

cloudera[®]

Cloudera Data Science Workbench

Important Notice

© 2010-2022 Cloudera, Inc. All rights reserved.

Cloudera, the Cloudera logo, and any other product or service names or slogans contained in this document are trademarks of Cloudera and its suppliers or licensors, and may not be copied, imitated or used, in whole or in part, without the prior written permission of Cloudera or the applicable trademark holder. If this documentation includes code, including but not limited to, code examples, Cloudera makes this available to you under the terms of the Apache License, Version 2.0, including any required notices. A copy of the Apache License Version 2.0, including any notices, is included herein. A copy of the Apache License Version 2.0 can also be found here: <https://opensource.org/licenses/Apache-2.0>

Hadoop and the Hadoop elephant logo are trademarks of the Apache Software Foundation. All other trademarks, registered trademarks, product names and company names or logos mentioned in this document are the property of their respective owners. Reference to any products, services, processes or other information, by trade name, trademark, manufacturer, supplier or otherwise does not constitute or imply endorsement, sponsorship or recommendation thereof by us.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Cloudera.

Cloudera may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Cloudera, the furnishing of this document does not give you any license to these patents, trademarks copyrights, or other intellectual property. For information about patents covering Cloudera products, see <http://tiny.cloudera.com/patents>.

The information in this document is subject to change without notice. Cloudera shall not be liable for any damages resulting from technical errors or omissions which may be present in this document, or from use of this document.

Cloudera, Inc.

**395 Page Mill Road
Palo Alto, CA 94306
info@cloudera.com
US: 1-888-789-1488
Intl: 1-650-362-0488
www.cloudera.com**

Release Information

Version: Cloudera Data Science Workbench 1.7.x
Date: March 22, 2022

Table of Contents

Cloudera Data Science Workbench Overview.....15

Architecture Overview.....	17
Cloudera Manager.....	17
Engines.....	18
Cloudera Data Science Workbench Web Application.....	19
CDS 2.x Powered by Apache Spark.....	19

Cloudera Data Science Workbench Release Notes.....21

Cloudera Data Science Workbench 1.7.2.....	21
<i>New Features and Changes in Cloudera Data Science Workbench 1.7.2.....</i>	<i>21</i>
<i>Issues Fixed in Cloudera Data Science Workbench 1.7.2.....</i>	<i>21</i>
Cloudera Data Science Workbench 1.7.1.....	22
<i>New Features and Changes in Cloudera Data Science Workbench 1.7.1.....</i>	<i>22</i>
<i>Issues Fixed in Cloudera Data Science Workbench 1.7.1.....</i>	<i>23</i>
Cloudera Data Science Workbench 1.6.1.....	23
<i>New Features and Changes in Cloudera Data Science Workbench 1.6.1.....</i>	<i>23</i>
<i>Issues Fixed in Cloudera Data Science Workbench 1.6.1.....</i>	<i>24</i>
Cloudera Data Science Workbench 1.6.0.....	25
<i>New Features and Changes in Cloudera Data Science Workbench 1.6.0.....</i>	<i>25</i>
<i>Incompatible Changes in Cloudera Data Science Workbench 1.6.0.....</i>	<i>28</i>
<i>Issues Fixed in Cloudera Data Science Workbench 1.6.0.....</i>	<i>28</i>
<i>Known Issues and Limitations in Cloudera Data Science Workbench 1.6.0.....</i>	<i>29</i>
Cloudera Data Science Workbench 1.5.0.....	29
<i>New Features and Changes in Cloudera Data Science Workbench 1.5.0.....</i>	<i>29</i>
<i>Incompatible Changes in Cloudera Data Science Workbench 1.5.0.....</i>	<i>31</i>
<i>Issues Fixed in Cloudera Data Science Workbench 1.5.0.....</i>	<i>31</i>
<i>Known Issues and Limitations in Cloudera Data Science Workbench 1.5.0.....</i>	<i>32</i>
Cloudera Data Science Workbench 1.4.3.....	32
<i>New Features and Changes in Cloudera Data Science Workbench 1.4.3.....</i>	<i>32</i>
<i>Issues Fixed in Cloudera Data Science Workbench 1.4.3.....</i>	<i>33</i>
<i>Known Issues and Limitations in Cloudera Data Science Workbench 1.4.3.....</i>	<i>37</i>
Cloudera Data Science Workbench 1.4.2.....	37
<i>New Features and Changes in Cloudera Data Science Workbench 1.4.2.....</i>	<i>37</i>
<i>Issues Fixed in Cloudera Data Science Workbench 1.4.2.....</i>	<i>38</i>
<i>Known Issues and Limitations in Cloudera Data Science Workbench 1.4.2.....</i>	<i>40</i>
Cloudera Data Science Workbench 1.4.0.....	41
<i>New Features in Cloudera Data Science Workbench 1.4.0.....</i>	<i>41</i>

- Incompatible Changes in Cloudera Data Science Workbench 1.4.0*.....42
- Issues Fixed in Cloudera Data Science Workbench 1.4.0*.....42
- Known Issues and Limitations in Cloudera Data Science Workbench 1.4.0*.....43
- Cloudera Data Science Workbench 1.3.1.....43
- New Features in Cloudera Data Science Workbench 1.3.1*.....43
- Issues Fixed in Cloudera Data Science Workbench 1.3.1*.....43
- Known Issues and Limitations in Cloudera Data Science Workbench 1.3.1*.....44
- Cloudera Data Science Workbench 1.3.0.....44
- New Features and Changes in Cloudera Data Science Workbench 1.3.0*.....44
- Issues Fixed in Cloudera Data Science Workbench 1.3.0*.....44
- Incompatible Changes in Cloudera Data Science Workbench 1.3.0*.....45
- Known Issues and Limitations in Cloudera Data Science Workbench 1.3.0*.....45
- Cloudera Data Science Workbench 1.2.2.....45
- New Features and Changes in Cloudera Data Science Workbench 1.2.2*.....45
- Issues Fixed In Cloudera Data Science Workbench 1.2.2*.....46
- Known Issues and Limitations in Cloudera Data Science Workbench 1.2.2*.....46
- Cloudera Data Science Workbench 1.2.1.....47
- Issues Fixed In Cloudera Data Science Workbench 1.2.1*.....47
- Incompatible Changes in Cloudera Data Science Workbench 1.2.1*.....47
- Known Issues and Limitations in Cloudera Data Science Workbench 1.2.1*.....47
- Cloudera Data Science Workbench 1.2.0.....47
- New Features and Changes in Cloudera Data Science Workbench 1.2.0*.....47
- Issues Fixed in Cloudera Data Science Workbench 1.2.0*.....48
- Incompatible Changes in Cloudera Data Science Workbench 1.2.0*.....49
- Known Issues and Limitations in Cloudera Data Science Workbench 1.2.0*.....49
- Cloudera Data Science Workbench 1.1.1.....49
- New Features in Cloudera Data Science Workbench 1.1.1*.....49
- Issues Fixed In Cloudera Data Science Workbench 1.1.1*.....49
- Known Issues and Limitations in Cloudera Data Science Workbench 1.1.1*.....49
- Cloudera Data Science Workbench 1.1.0.....49
- New Features and Changes in Cloudera Data Science Workbench 1.1.0*.....49
- Issues Fixed in Cloudera Data Science Workbench 1.1.0*.....50
- Incompatible Changes in Cloudera Data Science Workbench 1.1.0*.....50
- Known Issues and Limitations in Cloudera Data Science Workbench 1.1.0*.....51
- Cloudera Data Science Workbench 1.0.1.....51
- Issues Fixed in Cloudera Data Science Workbench 1.0.1*.....51
- Known Issues and Limitations in Cloudera Data Science Workbench 1.0.x*.....51
- Cloudera Data Science Workbench 1.0.0.....52
- Known Issues and Limitations in Cloudera Data Science Workbench 1.7.2.....52
- Installation*.....52
- Upgrades*.....52
- CDH Integration*.....53
- Cloudera Manager Integration*.....54
- CDS Powered By Apache Spark*.....55

<i>Crashes and Hangs</i>	56
<i>Third-party Editors</i>	56
<i>Engines</i>	57
<i>Experiments</i>	58
<i>GPU Support</i>	58
<i>Jobs</i>	59
<i>Models</i>	59
<i>Networking</i>	60
<i>Quotas</i>	61
<i>Security</i>	61
<i>Usability</i>	63
<i>Other</i>	64

Cloudera Data Science Workbench 1.7.2 Requirements and Supported

Platforms	65
Cloudera Manager and CDH Requirements.....	65
Operating System Requirements.....	65
JDK Requirements.....	66
Networking and Security Requirements.....	67
<i>Ports Required by Cloudera Data Science Workbench</i>	68
Recommended Hardware Configuration.....	69
Python Supported Versions.....	70
Docker and Kubernetes Support.....	70
Supported Browsers.....	70
Cloudera Altus Director Support (AWS and Azure Only).....	71
Recommended Configuration on Amazon Web Services (AWS).....	71
Recommended Configuration on Microsoft Azure.....	72

Installing Cloudera Data Science Workbench 1.7.2 on CDH.....73

Installing Cloudera Data Science Workbench 1.7.2.....	73
Multiple Cloudera Data Science Workbench Deployments.....	73
Airgapped Installations.....	73
Required Pre-Installation Steps.....	74
<i>Review Requirements and Supported Platforms</i>	74
<i>Set Up a Wildcard DNS Subdomain</i>	74
<i>Disable Untrusted SSH Access</i>	74
<i>Configure Block Devices</i>	75
<i>Install Cloudera Data Science Workbench</i>	75
Installing Cloudera Data Science Workbench 1.7.2 Using Cloudera Manager.....	75
<i>Prerequisites</i>	76
<i>Configure Apache Spark 2</i>	76

<i>Configure JAVA_HOME</i>	78
<i>Download and Install the Cloudera Data Science Workbench CSD</i>	78
<i>Install the Cloudera Data Science Workbench Parcel</i>	79
<i>Add the Cloudera Data Science Workbench Service</i>	80
<i>Create the Administrator Account</i>	82
<i>Next Steps</i>	82
Installing Cloudera Data Science Workbench 1.7.2 Using Packages	83
<i>Prerequisites</i>	83
<i>Configure Gateway Hosts Using Cloudera Manager</i>	83
<i>Install Cloudera Data Science Workbench on the Master Host</i>	84
<i>(Optional) Install Cloudera Data Science Workbench on Worker Hosts</i>	88
<i>Create the Administrator Account</i>	89
<i>Next Steps</i>	90

Upgrading to the Latest Version of Cloudera Data Science Workbench 1.7.2 on

CDH	91
Upgrading Cloudera Data Science Workbench from CDH 5 to CDH 6.....	91
Upgrading Cloudera Data Science Workbench 1.7.2 or higher from CDH 6 to CDP-DC 7.x.....	93
Upgrading Cloudera Data Science Workbench 1.7.2 Using Cloudera Manager.....	93
Migrating from an RPM-based Deployment to the Latest 1.7.2 CSD.....	97
Upgrading Cloudera Data Science Workbench 1.7.2 Using Packages.....	101

Deploying Cloudera Data Science Workbench 1.7.2 on Hortonworks Data

Platform	103
CDSW-on-HDP Architecture Overview.....	103
Supported Platforms and Requirements.....	104
<i>Platform Requirements</i>	104
<i>Operating System Requirements</i>	104
<i>Java Requirements</i>	105
<i>Network and Security Requirements</i>	105
<i>Hardware Requirements</i>	106
<i>Python Supported Versions</i>	107
Known Issues and Limitations.....	108
Installing Cloudera Data Science Workbench 1.7.2 on HDP	108
<i>Prerequisites</i>	108
<i>Add Gateway Hosts for Cloudera Data Science Workbench to Your HDP Cluster</i>	108
<i>Create HDFS User Directories</i>	109
<i>Install Cloudera Data Science Workbench on the Master Host</i>	109
<i>(Optional) Install Cloudera Data Science Workbench on Worker Hosts</i>	113
<i>Create the Site Administrator Account</i>	114
Upgrading to Cloudera Data Science Workbench 1.7.2 on HDP.....	114

Getting Started with a New Project on Cloudera Data Science Workbench.....	115
Upgrading a CDSW 1.7.2 Deployment from HDP 2 to HDP 3.....	115
Frequently Asked Questions (FAQs).....	116

Getting Started with Cloudera Data Science Workbench.....117

Sign up.....	117
Create a Project from a Built-in Template.....	117
Launch a Session to Run the Project.....	118
Next Steps.....	120

Managing Cloudera Data Science Workbench Users.....121

Managing your Personal Account.....	121
Managing Team Accounts.....	122
<i>Creating a Team</i>	122
<i>Modifying Team Account Settings</i>	123
Managing Users as a Site Administrator.....	123

Managing Projects in Cloudera Data Science Workbench.....126

Creating a Project	126
Adding Collaborators.....	127
Modifying Project Settings	128
Managing Files.....	128
<i>Disabling Project File Uploads and Downloads</i>	129
Custom Template Projects.....	129
Deleting a Project.....	129
Using the Workbench.....	130
<i>Launch a Session</i>	130
<i>Execute Code</i>	131
<i>Access the Terminal</i>	132
<i>Stop a Session</i>	132
<i>Jupyter Magic Commands</i>	133
Data Visualization.....	133
<i>Simple Plots</i>	134
<i>Saved Images</i>	134
<i>HTML Visualizations</i>	134
<i>IFrame Visualizations</i>	135
<i>Grid Displays</i>	136
<i>Documenting Your Analysis</i>	136
Using NVIDIA GPUs for Cloudera Data Science Workbench Projects.....	137
<i>Key Points to Note</i>	137
<i>Enabling Cloudera Data Science Workbench to use GPUs</i>	138

Web Applications Embedded in Cloudera Data Science Workbench.....	142
<i>Spark 2 Web UIs (CDSW_SPARK_PORT).....</i>	<i>143</i>
<i>TensorBoard, Shiny, and others (CDSW_APP_PORT or CDSW_READONLY_PORT).....</i>	<i>143</i>
<i>Example: A Shiny Application.....</i>	<i>143</i>
Accessing Web User Interfaces from Cloudera Data Science Workbench.....	145
<i>Cloudera Manager, Hue, and the Spark History Server.....</i>	<i>145</i>
Running Distributed ML Workloads on YARN.....	145
<i>Example: H2O.....</i>	<i>145</i>
Distributed Computing with Workers.....	146
<i>Workers API.....</i>	<i>146</i>
<i>Example: Worker Network Communications.....</i>	<i>148</i>

Collaborating on Projects with Cloudera Data Science Workbench.....150

Project Collaborators.....	150
<i>Restricting Collaborator and Administrator Access to Active Sessions.....</i>	<i>150</i>
Teams.....	151
Sharing Personal Projects.....	151
Forking Projects.....	151
Collaborating with Git.....	151
Sharing Job and Session Console Outputs.....	151
Using Git to Collaborate on Projects.....	152
<i>Importing a Project From Git.....</i>	<i>152</i>
<i>Linking an Existing Project to a Git Remote.....</i>	<i>152</i>

Editors.....154

Configure a Browser IDE as an Editor.....	155
<i>Test a Browser IDE in a Session Before Installation.....</i>	<i>155</i>
<i>Configure a Browser IDE at the Project Level.....</i>	<i>156</i>
<i>Configure a Browser IDE at the Engine Level.....</i>	<i>157</i>
<i>Configure Jupyter Notebook in a Customized Engine Image.....</i>	<i>160</i>
Configure a SSH Gateway to Use Local IDEs.....	160
<i>Configure and Use a Local IDE.....</i>	<i>160</i>
<i>Configure PyCharm as a Local IDE.....</i>	<i>160</i>

Importing Data into Cloudera Data Science Workbench.....165

Accessing Local Data from Your Computer.....	165
Accessing Data from HDFS.....	165
Accessing Data from Apache HBase.....	166
Accessing Data from Apache Hive.....	167
Accessing Data from Apache Impala.....	168
<i>Loading CSV Data into an Impala Table.....</i>	<i>168</i>

<i>Running Queries on Impala Tables</i>	168
Accessing Data in Amazon S3 Buckets.....	171
Accessing External SQL Databases.....	172
<i>R</i>	172
<i>Python</i>	172

Experiments.....173

Purpose.....	173
Concepts.....	173
Running an Experiment (QuickStart).....	174
Tracking Metrics.....	177
Saving Files.....	177
Disabling the Experiments Feature.....	177
Limitations.....	178
Debugging Issues with Experiments.....	178

Models.....180

Purpose.....	180
Concepts and Terminology.....	180
Creating and Deploying a Model (QuickStart).....	182
Calling a Model.....	184
Updating Active Models.....	186
Usage Guidelines.....	188
Known Issues and Limitations.....	189
Model Training and Deployment - Iris Dataset.....	190
<i>Create a Project</i>	190
<i>Train the Model</i>	190
<i>Deploy the Model</i>	192
Model Monitoring and Administration.....	193
<i>Monitoring Active Models</i>	193
<i>Deleting a Model</i>	194
<i>Disabling the Models Feature</i>	195
Debugging Issues with Models.....	195

Analytical Applications.....197

Limitations.....	198
------------------	-----

Managing Jobs and Pipelines in Cloudera Data Science Workbench.....199

Creating a Job.....	199
Creating a Pipeline.....	200

Viewing Job History.....	200
Cloudera Data Science Workbench Jobs API.....	200
<i>API Key Authentication</i>	200
<i>Starting a Job Run Using the API</i>	201
<i>Starting a Job Run Using Python</i>	202
<i>Limitations</i>	203

Cloudera Data Science Workbench Engines.....204

Basic Concepts and Terminology.....	204
Project Environments.....	205
<i>Environmental Variables</i>	205
<i>Dependencies</i>	205
<i>Configuring Engine Environments for Experiments and Models</i>	205
Engines for Experiments and Models.....	206
<i>Snapshot Code</i>	206
<i>Build Image</i>	206
<i>Run Experiment / Deploy Model</i>	208
Configuring Cloudera Data Science Workbench Engines.....	208
<i>Concepts and Terminology</i>	208
<i>Managing Engines</i>	208
Managing Engine Dependencies.....	211
<i>Installing Packages Directly Within Projects</i>	212
<i>Creating a Customized Engine with the Required Package(s)</i>	213
<i>Mounting Additional Dependencies from the Host</i>	214
<i>Managing Dependencies for Spark 2 Projects</i>	214
Engine Environment Variables.....	215
<i>Environment Variables from Cloudera Manager</i>	215
<i>Accessing Environmental Variables from Projects</i>	216
<i>Engine Environment Variables</i>	216
Installing Additional Packages.....	218
<i>Using Conda with Cloudera Data Science Workbench</i>	219
Customized Engine Images.....	220
<i>Creating a Customized Engine Image</i>	221
<i>End-to-End Example: MeCab</i>	222
<i>Limitations</i>	223
<i>Related Resources</i>	224
Cloudera Data Science Workbench Engine Versions and Packaging.....	224
<i>Base Engine 10</i>	224
<i>Base Engine 8</i>	226
<i>Base Engine 7</i>	227
<i>Base Engine 6</i>	229
<i>Base Engine 5</i>	230
<i>Base Engine 4</i>	231

<i>Base Engine 3</i>	233
<i>Base Engine 2</i>	234
<i>CUDA Engine - Technical Preview</i>	235
<i>Enabling GPUs for CDSW with the CUDA engine</i>	236

Using CDS 2.x Powered by Apache Spark.....238

<i>Configuring CDS 2.x Powered by Apache Spark 2</i>	238
<i>Spark Configuration Files</i>	239
<i>Managing Memory Available for Spark Drivers</i>	239
<i>Managing Dependencies for Spark 2 Jobs</i>	240
<i>Spark Logging Configuration</i>	240
<i>Running Spark Jobs on an HDP Cluster</i>	241
<i>Setting Up an HTTP Proxy for Spark 2</i>	241
<i>Using Spark 2 from Python</i>	241
<i>Setting Up a PySpark Project</i>	242
<i>Example: Montecarlo Estimation</i>	242
<i>Example: Locating and Adding JARs to Spark 2 Configuration</i>	243
<i>Example: Distributing Dependencies on a PySpark Cluster</i>	244
<i>Using Spark 2 from R</i>	246
<i>Installing sparklyr</i>	246
<i>Connecting to Spark 2</i>	246
<i>Using Spark 2 from Scala</i>	246
<i>Accessing Spark 2 from the Scala Engine</i>	246
<i>Example: Read Files from the Cluster Local Filesystem</i>	247
<i>Example: Using External Packages by Adding Jars or Dependencies</i>	247

Cloudera Data Science Workbench Administration Guide.....249

<i>Monitoring Cloudera Data Science Workbench Activity</i>	249
<i>Related Resources</i>	250
<i>Monitoring User Events</i>	250
<i>Tracked User Events</i>	251
<i>Configuring Quotas - Technical Preview</i>	255
<i>Managing the Cloudera Data Science Workbench Service in Cloudera Manager</i>	256
<i>Adding the Cloudera Data Science Workbench Service</i>	256
<i>Roles Associated with the Cloudera Data Science Workbench Service</i>	256
<i>Accessing Cloudera Data Science Workbench from Cloudera Manager</i>	257
<i>Configuring Cloudera Data Science Workbench Properties</i>	257
<i>Starting, Stopping, and Restarting the Service</i>	257
<i>Checking the Status of the CDSW Service</i>	258
<i>Managing Cloudera Data Science Workbench Worker Hosts</i>	258
<i>Health Tests</i>	258
<i>Tracking Disk Usage on the Application Block Device</i>	258

<i>Creating Diagnostic Bundles</i>	260
Data Collection in Cloudera Data Science Workbench.....	260
<i>Usage Tracking</i>	260
<i>Diagnostic Bundles</i>	260
Cloudera Data Science Workbench Email Notifications.....	262
Managing License Keys for Cloudera Data Science Workbench.....	262
<i>Trial License</i>	262
<i>Cloudera Enterprise License</i>	263
<i>Uploading License Keys</i>	263
User Access to Features.....	263

Cluster Management.....265

Cluster Monitoring with Grafana.....	265
Backup and Disaster Recovery for Cloudera Data Science Workbench.....	266
<i>Creating a Backup</i>	266
<i>Restoring from a Backup</i>	267
Cloudera Data Science Workbench Scaling Guidelines.....	267
Ports Used By Cloudera Data Science Workbench.....	267
Managing Cloudera Data Science Workbench Hosts.....	268
<i>Customize Workload Scheduling</i>	268
<i>Migrating a Deployment to a New Set of Hosts</i>	269
<i>Adding a Worker Host</i>	270
<i>Removing a Worker Host</i>	271
<i>Changing the Domain Name</i>	271
Migrating a Deployment to a New Set of Hosts.....	272
<i>Migrating a CSD Deployment</i>	272
<i>Migrating an RPM Deployment</i>	274
Rollback Cloudera Data Science Workbench to an Older Version.....	275
Uninstalling Cloudera Data Science Workbench.....	276
<i>CSD Deployments</i>	276
<i>RPM Deployments</i>	276

Cloudera Data Science Workbench Security Guide.....278

Security Model.....	278
<i>Wildcard DNS Subdomain Requirement</i>	279
Authentication.....	279
Authorization.....	280
<i>Cluster Authorization</i>	280
<i>User Role Authorization</i>	280
<i>Access Control for Teams and Projects</i>	280
Wire Encryption.....	284
<i>External Communications</i>	284

<i>Internal Communications</i>	284
Cloudera Data Science Workbench Gateway Host Security.....	284
Base Engine Image Security.....	285
Engine-level Security for Custom Editors and other Web Applications.....	285
Enabling TLS/SSL for Cloudera Data Science Workbench.....	285
<i>Private Key and Certificate Requirements</i>	286
<i>Configuring Internal Termination</i>	287
<i>Configuring External Termination</i>	288
<i>Known Issues and Limitations</i>	289
<i>Configuring Custom Root CA Certificate</i>	289
Configuring Cloudera Data Science Workbench Deployments Behind a Proxy.....	290
Hadoop Authentication with Kerberos for Cloudera Data Science Workbench.....	291
<i>Configure FreeIPA</i>	292
Configuring External Authentication with LDAP and SAML.....	293
<i>User Signup Process</i>	294
<i>Configuring LDAP/Active Directory Authentication</i>	294
<i>Configuring SAML Authentication</i>	297
<i>Debug Login URL</i>	299
Configuring HTTP Headers for Cloudera Data Science Workbench.....	300
<i>Enable HTTP Security Headers</i>	300
<i>Enable HTTP Strict Transport Security (HSTS)</i>	300
<i>Enable Cross-Origin Resource Sharing (CORS)</i>	300
Restricting User-Controlled Kubernetes Pods.....	301
<i>Allow containers to run as root</i>	301
<i>Allow "privileged" pod containers</i>	301
<i>Allow pod containers to mount unsupported volume types</i>	301
SSH Keys.....	302
<i>Personal Key</i>	302
<i>Team Key</i>	302
<i>Adding SSH Key to GitHub</i>	302
<i>SSH Tunnels</i>	302

Troubleshooting Cloudera Data Science Workbench.....304

Understanding Installation Warnings.....	304
Error Encountered Trying to Load Images when Initializing Cloudera Data Science Workbench.....	307
404 Not Found Error.....	307
Troubleshooting Kerberos Errors.....	308
Troubleshooting TLS/SSL Errors.....	308
Troubleshooting Issues with Workloads.....	309
Troubleshooting Issues with Models and Experiments.....	311

Cloudera Data Science Workbench Command Line Reference.....312

cdswct1 Command Line Interface Client.....313
Download and Configure the cdswct1.....314

Cloudera Data Science Workbench FAQs.....316

Where can I get a sample project to try out Cloudera Data Science Workbench?.....316
What are the software and hardware requirements for Cloudera Data Science Workbench?.....316
Can I run Cloudera Data Science Workbench on hosts shared with other Hadoop services?.....316
How does Cloudera Data Science Workbench use Docker and Kubernetes?.....316
Can I run Cloudera Data Science Workbench on my own Kubernetes cluster?.....316
Does Cloudera Data Science Workbench support REST API access?.....316
How do I contact Cloudera for issues regarding Cloudera Data Science Workbench?.....316

Cloudera Data Science Workbench Glossary.....318

Appendix: Apache License, Version 2.0.....320

Cloudera Data Science Workbench Overview

Cloudera Data Science Workbench is a secure, self-service enterprise data science platform that lets data scientists manage their own analytics pipelines, thus accelerating machine learning projects from exploration to production. It allows data scientists to bring their existing skills and tools, such as R, Python, and Scala, to securely run computations on data in Hadoop clusters. It enables data science teams to use their preferred data science packages to run experiments with on-demand access to compute resources. Models can be trained, deployed, and managed centrally for increased agility and compliance.

Built for the enterprise, Cloudera Data Science Workbench includes direct integration with the Cloudera platform for a complete machine learning workflow that supports collaborative development, and can run both in the public cloud and on-premises.

Demo - Watch this video for a quick 3 minute demo of Cloudera Data Science Workbench: [CDSW Quickstart](#)

The screenshot displays the Cloudera Data Science Workbench interface. At the top, there are navigation tabs for 'Documentation' and 'Projects'. Below this, a dashboard shows '2 sessions running', '0 jobs running', and resource usage: '2.00 vCPU' and '4.00 GIB / 15.17 GIB'. The main workspace is divided into several panes:

- Projects List:** A sidebar on the left lists various projects such as 'Python Template Project', 'Python Visualizations', 'Air Quality San Francisco', 'GitHub Data', 'Yahoo Search Data', and 'PySpark Tests'.
- Code Editor:** The central pane shows an R script named 'analysis.r' with code for setting up a workspace, loading data, and plotting 'Boston Median Housing Price' vs 'Crime'.
- Table View:** A table on the right displays data for 'boston' with columns: crim, indus, nox, rm, and lstat. The data is as follows:

X	crim	indus	nox	rm		
1	0.0063	18	2.31	0.538	6.575	
2	0.0273	0	7.07	0	0.469	6.421
3	0.0273	0	7.07	0	0.469	7.185
4	0.0324	0	2.18	0	0.458	6.998
5	0.069	0	2.18	0	0.458	7.147
6	0.0298	0	2.18	0	0.458	6.43
- Cluster Metadata:** A pane on the bottom left shows job history and cluster details, including a 'Job History' graph and 'Latest Run' information (Duration: 01:44, Run: 8286, Failed: 47).
- Explore:** A histogram on the bottom right visualizes the 'Boston Median Housing Price' distribution.

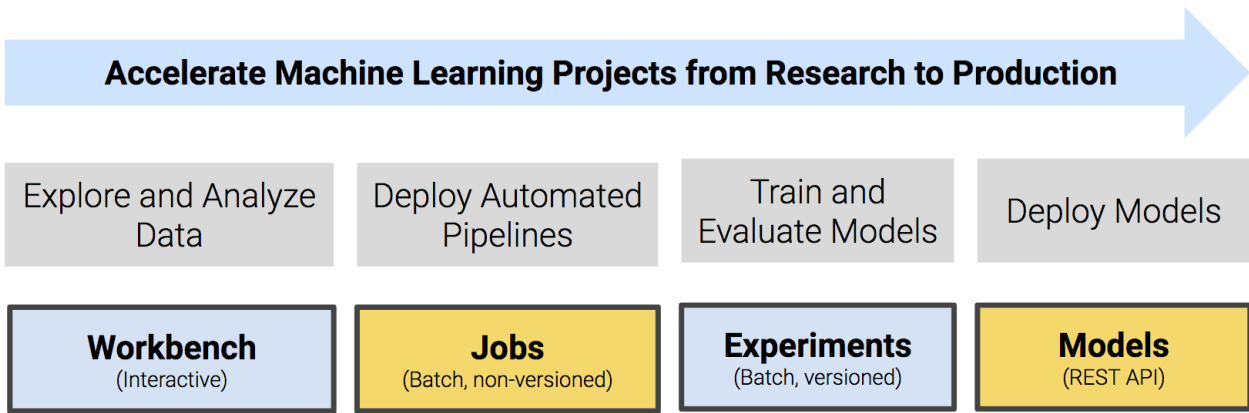
Typical Machine Learning Project Workflow

Machine learning is a discipline that uses computer algorithms to extract useful knowledge from data. There are many different types of machine learning algorithms, and each one works differently. In general however, machine learning algorithms begin with an initial hypothetical model, determine how well this model fits a set of data, and then work on improving the model iteratively. This training process continues until the algorithm can find no additional improvements, or until the user stops the process.

A typical machine learning project will include the following high-level steps that will transform a loose data hypothesis into a model that serves predictions.

1. Explore and experiment with and display findings of data
2. Deploy automated pipelines of analytics workloads
3. Train and evaluate models
4. Deploy models as REST APIs to serve predictions

With Cloudera Data Science Workbench, you can deploy the complete lifecycle of a machine learning project from research to deployment.



Core Capabilities of Cloudera Data Science Workbench

For Data Scientists

Projects
Organize your data science efforts as isolated projects, which might include reusable code, configuration, artifacts, and libraries. Projects can also be connected to GitHub repositories for integrated version control and collaboration.

Workbench
A workbench for data scientists and data engineers that includes support for:

- Interactive user sessions with Python, R, and Scala through flexible and extensible *engines*.
- Project workspaces powered by Docker containers for control over environment configuration. You can install new packages or run command-line scripts directly from the built-in terminal.
- Distributing computations to your Cloudera Manager cluster using CDS 2.x Powered by Apache Spark and Apache Impala.
- Sharing, publishing, and collaboration of projects and results.

Jobs
Automate analytics workloads with a lightweight job and pipeline scheduling system that supports real-time monitoring, job history, and email alerts.

Batch Experiments
Demo - [Experiments](#)
Use batch jobs to train and compare versioned, reproducible models. With experiments, data scientists can:

- Create versioned snapshots of model code, dependencies, and any configuration parameters required to train the model.
- Build and execute each training run in an isolated container.
- Track model metrics, performance, and model artifacts as required.

Models
Demo - [Model Deployment](#)
Deploy and serve models as REST APIs. Data scientists can select a specific Python or R function within a project file to be deployed as a model, and Cloudera Data Science Workbench will:

- Create a snapshot of the model code, saved model parameters, and dependencies.
- Build an immutable executable container with the trained model and serving code.
- Deploy the model as a REST API along with a specified number of replicas, automatically load balanced.
- Save the built model container, along with metadata such as who built or deployed it.
- Allow data scientists to test and share the model

For IT Administrators**Native Support for the Cloudera Enterprise Data Hub**

Direct integration with the Cloudera Enterprise Data Hub makes it easy for end users to interact with existing clusters, without having to bother IT or compromise on security. No additional setup is required. They can just start coding.

Enterprise Security

Cloudera Data Science Workbench can leverage your existing authentication systems such as SAML or LDAP/Active Directory. It also includes native support for Kerberized Hadoop clusters.

Native Spark 2 Support

Cloudera Data Science Workbench connects to existing Spark-on-YARN clusters with no setup required.

Flexible Deployment

Deploy on-premises or in the cloud (on IaaS) and scale capacity as workloads change.

Multitenancy Support

A single Cloudera Data Science Workbench deployment can support different business groups sharing common infrastructure without interfering with one another, or placing additional demands on IT.

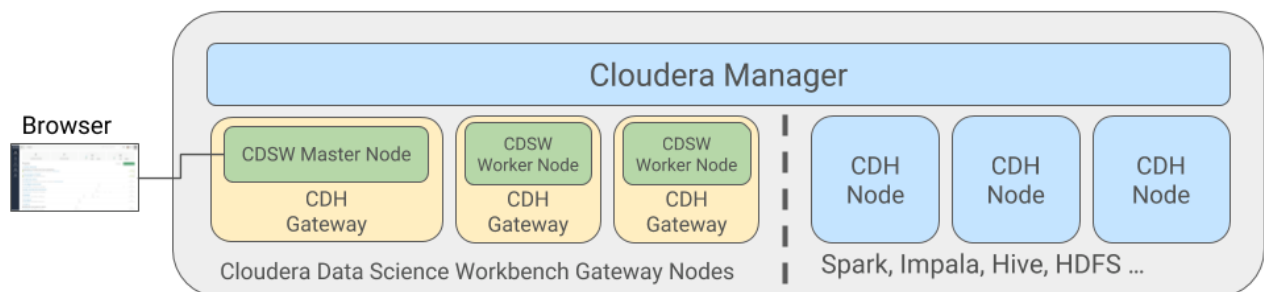
Architecture Overview



Important: The rest of this documentation assumes you are familiar with CDH and Cloudera Manager. If not, make sure you read the [documentation for CDH and Cloudera Manager](#) before you proceed.

Cloudera Manager

Cloudera Manager is an end-to-end application used for managing CDH clusters. When a CDH service (such as Impala, Spark, etc.) is added to the cluster, Cloudera Manager configures cluster hosts with one or more functions, called *roles*. In a Cloudera Manager cluster, a gateway role is one that designates that a host should receive client configuration for a CDH service even though the host does not have any role instances for that service running on it. Gateway roles provide the configuration required for clients that want to access the CDH cluster. Hosts that are designated with gateway roles for CDH services are referred to as gateway hosts.



Cloudera Data Science Workbench runs on one or more dedicated gateway hosts on CDH clusters. Each of these hosts has the Cloudera Manager Agent installed on them. The Cloudera Management Agent ensures that Cloudera Data Science Workbench has the libraries and configuration necessary to securely access the CDH cluster.

Cloudera Data Science Workbench does not support running any other services on these gateway hosts. Each gateway host must be dedicated solely to Cloudera Data Science Workbench. This is because user workloads require dedicated CPU and memory, which might conflict with other services running on these hosts. Any workloads that you run on Cloudera Data Science Workbench hosts will have immediate secure access to the CDH cluster.

From the assigned gateway hosts, one will serve as the *master* host while others will serve as *worker* hosts.

Cloudera Data Science Workbench Overview

Master Host

The master host keeps track of all critical persistent and stateful data within Cloudera Data Science Workbench. This data is stored at `/var/lib/cdsw`.

- **Project Files**

Cloudera Data Science Workbench uses an NFS server to store project files. Project files can include user code, any libraries you install, and small data files. The master host provides a persistent filesystem which is exported to worker hosts using NFS. This filesystem allows users to install packages interactively and have their dependencies and code available on all Cloudera Data Science Workbench nodes without any need for synchronization. The files for *all* the projects are stored on the master host at `/var/lib/cdsw/current/projects`. When a job or session is launched, the project's filesystem is mounted into an isolated Docker container at `/home/cdsw`.

- **Relational Database**

The Cloudera Data Science Workbench uses a PostgreSQL database that runs within a container on the master host at `/var/lib/cdsw/current/postgres-data`.

- **Livelog**

Cloudera Data Science Workbench allows users to work interactively with R, Python, and Scala from their browser and display results in realtime. This realtime state is stored in an internal database called Livelog, which stores data on the master host at `/var/lib/cdsw/current/livelog`. Users do not need to be connected to the server for results to be tracked or jobs to run.

Worker Hosts

While the master host stores the stateful components of the Cloudera Data Science Workbench, the worker hosts are transient. These can be added or removed as needed, which gives you flexibility with scaling the deployment. As the number of users and workloads increases, you can add more worker hosts to Cloudera Data Science Workbench over time.



Note: For proof-of-concept deployments, you can deploy a 1-host cluster with just a Master host. The Master host can run user workloads just as a worker host can when required for demonstration purposes. For production deployments, Cloudera requires to have a reserved, dedicated master host and separate worker host(s).

Even on multi-host deployments, the Master host doubles up to perform both functions: those of the Master [outlined above](#), and those of a worker. Starting with version 1.4.3, multi-host deployments can be customized to reserve the Master only for internal processes while user workloads are run exclusively on workers. For details, see [Reserving the Master Host for Internal CDSW Components](#) on page 269.

Engines

Cloudera Data Science Workbench engines are responsible for running R, Python, and Scala code written by users and intermediating access to the CDH cluster. You can think of an engine as a virtual machine, customized to have all the necessary dependencies to access the CDH cluster while keeping each project's environment entirely isolated. To ensure that every engine has access to the parcels and client configuration managed by the Cloudera Manager Agent, a number of folders are mounted from the host into the container environment. This includes the parcel path `/opt/cloudera`, client configuration, as well as the host's `JAVA_HOME`.

For more details on basic concepts and terminology related to engines in Cloudera Data Science Workbench, see [Cloudera Data Science Workbench Engines](#) on page 204.

Docker and Kubernetes

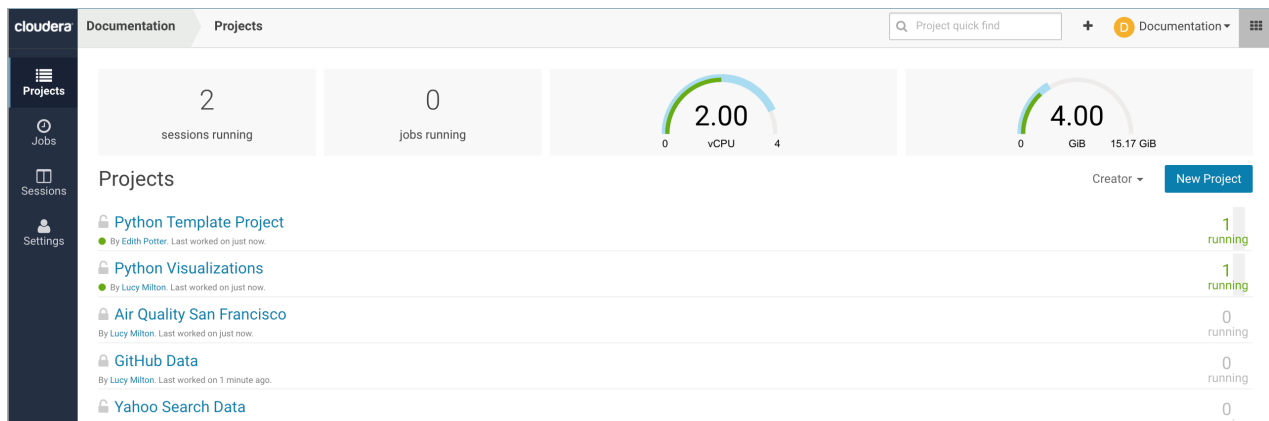
Cloudera Data Science Workbench uses Docker containers to deliver application components and run isolated user workloads. On a per project basis, users can run R, Python, and Scala workloads with different versions of libraries and system packages. CPU and memory are also isolated, ensuring reliable, scalable execution in a multi-tenant setting. Each Docker container running user workloads, also referred to as an *engine*, provides a visualized gateway with secure access to CDH cluster services such as HDFS, Spark 2, Hive, and Impala. CDH dependencies and client configuration, managed by Cloudera Manager, are mounted from the underlying gateway host. Workloads that leverage CDH services such as HDFS, Spark, Hive, and Impala are executed across the full CDH cluster.

To enable multiple users and concurrent access, Cloudera Data Science Workbench transparently subdivides and schedules containers across multiple hosts dedicated as gateway hosts. This scheduling is done using Kubernetes, a container orchestration system used internally by Cloudera Data Science Workbench. Neither Docker nor Kubernetes are directly exposed to end users, with users interacting with Cloudera Data Science Workbench through a web application.

Cloudera Data Science Workbench Web Application

The Cloudera Data Science Workbench web application is typically hosted on the master host, at `http://cdsw.<your_domain>.com`. The web application provides a rich GUI that allows you to create projects, collaborate with your team, run data science workloads, and easily share the results with your team. For a quick demonstration, either watch this [video](#) or read the [Quickstart Guide](#).

You can log in to the web application either as a site administrator or a regular user. See the [Administration](#) and [User Guides](#) respectively for more details on what you can accomplish using the web application.



CDS 2.x Powered by Apache Spark



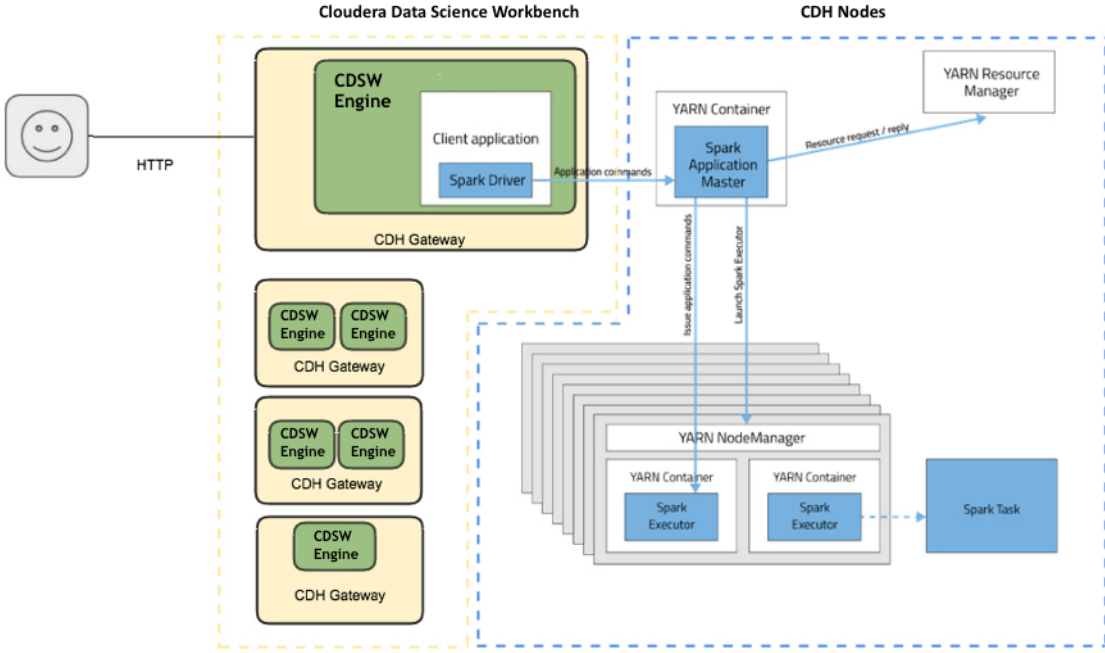
Important: The rest of this topic assumes you are familiar with Apache Spark and CDS 2.x Powered by Apache Spark. If not, make sure you read the [CDS 2.x documentation](#) before you proceed.

Apache Spark is a general purpose framework for distributed computing that offers high performance for both batch and stream processing. It exposes APIs for Java, Python, R, and Scala, as well as an interactive shell for you to run jobs.

Cloudera Data Science Workbench provides interactive and batch access to Spark 2. Connections are fully secure without additional configuration, with each user accessing Spark using their Kerberos principal. With a few extra lines of code, you can do anything in Cloudera Data Science Workbench that you might do in the Spark shell, as well as leverage all the benefits of the workbench. Your Spark applications will run in an isolated project workspace.

Cloudera Data Science Workbench's interactive mode allows you to launch a Spark application and work iteratively in R, Python, or Scala, rather than the standard workflow of launching an application and waiting for it to complete to view the results. Because of its interactive nature, Cloudera Data Science Workbench works with Spark on YARN's `client` mode, where the driver persists through the lifetime of the job and runs executors with full access to the CDH cluster resources. This architecture is illustrated the following figure:

Cloudera Data Science Workbench Overview



More resources:

- [Documentation for CDS 2.x Powered by Apache Spark](#)
- [Apache Spark 2 upstream documentation](#)

Cloudera Data Science Workbench Release Notes

These release notes provide information on new features, fixed issues and incompatible changes for all generally-available (GA) versions of Cloudera Data Science Workbench. For the current known issues and limitations, see [Known Issues and Limitations in Cloudera Data Science Workbench 1.7.2](#) on page 52.

Cloudera Data Science Workbench 1.7.2

This section lists the release notes for Cloudera Data Science Workbench 1.7.1.

New Features and Changes in Cloudera Data Science Workbench 1.7.2

- **Added support for CDP Data Center 7.0**

Cloudera Data Platform (CDP) Data Center is the on-premises version of Cloudera Data Platform. This new product combines the best of Cloudera Enterprise Data Hub and Hortonworks Data Platform Enterprise along with new features and enhancements across the stack. CDP Data Center is comprised of a variety of components such as Apache HDFS, Apache Hive 3, Apache HBase, and Apache Impala, along with many other components for specialized workloads. For details, see [CDP Data Center Overview](#).

You can use Cloudera Manager to [install CDSW 1.7.2 as a parcel](#) on CDP Data Center 7.0. RPM package installs are not supported.

- **Environment variables set at the Site Admin-level and project-level are now passed to models and experiments during the [container build process](#)**

Previously (CDSW 1.7.1 and lower), the environment variables set at the site admin level and project level did not automatically get pulled into the builds created for models and experiments. They needed to be explicitly coded into the [cdsw-build.sh file](#). With CDSW 1.7.2 and higher, experiments and models will automatically inherit these admin and project-level environment variables.

Note that custom mounts or environment variables configured in `cdsw.conf` (such as `NO_PROXY`, `HTTP(S)_PROXY`, etc.) are still not passed to the container builds for experiments and models (even though they are applied to sessions, jobs, and deployed models/experiments).

Issues Fixed in Cloudera Data Science Workbench 1.7.2

- Fixed an issue on multi-node CDSW 1.7.1 deployments where the [CDSW Web UI would not automatically come up after upgrading to CDSW 1.7.1](#).

Cloudera Bug: DSE-9587

- Fixed an issue where environmental variables set at the site admin level and at the project level would not get passed down to experiments and models at container build time.

Cloudera Bug: DSE-6708

- Fixed an issue where new users could not log in when the **Require invitation to sign up** checkbox was enabled. Additionally, the **Test LDAP Configuration** form did not return any error message if the user being tested wasn't already synced to the local CDSW database.

Cloudera Bug: DSE-3829

- Fixed an issue where license files could not be uploaded to CDSW through the UI.

Cloudera Bug: DSE-9874, DSE-8865

Cloudera Data Science Workbench 1.7.1



Note: Cloudera Data Science Workbench 1.7.1 is the next official release after Cloudera Data Science Workbench 1.6.x. Version 1.7.0 is no longer publicly available.

This section lists the release notes for Cloudera Data Science Workbench 1.7.1.

New Features and Changes in Cloudera Data Science Workbench 1.7.1

- **Supported upgrade paths to CDSW 1.7.1**

Cloudera Data Science Workbench only supports upgrades to version 1.7.1 from version 1.5.x and 1.6.x. If you are using an earlier version of CDSW, you must first upgrade to version 1.5.x or 1.6.x, and then upgrade to version 1.7.1.

- **Analytical Applications**

Cloudera Data Science Workbench now gives data scientists a way to create long-running standalone ML web applications/dashboards that can easily be shared with other business stakeholders. Applications can range from single visualizations embedded in reports, to rich dashboard solutions such as Tableau.

Applications stand alongside other existing forms of workloads in CDSW (sessions, jobs, experiments, models). For details, see [Analytical Applications](#) on page 197.

- **Monitoring CDSW with Grafana**

CDSW now leverages Prometheus and Grafana to provide a dashboard that allows you to monitor how CPU, memory, storage, and other resources are being consumed by CDSW deployment. For details, see [Cluster Monitoring with Grafana](#) on page 265.

- **Feature flag overrides**

This is a new property available in the CDSW service in Cloudera Manager. It can be used to enable/disable experimental features (such as [quotas](#)) and [disable usage metric collection](#) in diagnostic bundles.

- **Quotas**

CDSW site administrators can now enable CPU, GPU, and memory usage quotas per user. You can set default quotas for each user on the deployment as well as overriding custom quotas for specific users. For details, see [Configuring Quotas - Technical Preview](#) on page 255.



Important: The Quotas feature is in Technical Preview. It is not enabled by default. To enable quotas, use the **Feature flag overrides** property in Cloudera Manager.

- **Usage Metrics Collection**

By default, CDSW 1.7.1 now gathers highly redacted information on which feature is being used on your deployment. When you create a diagnostic bundle, this information is packed alongside the diagnostic information.

You can use the **Feature flag overrides** property in Cloudera Manager to [disable collection of usage metrics](#).

Engine Upgrade

Cloudera Data Science Workbench 1.7.1 (and later) ships **version 10** of the base engine image which includes the following versions of R and Python:

- **R** - 3.5.1
- **Python** - 2.7.17, 3.6.9

Pre-installed Packages in Engine 10

For details about the packages included in the base engine, see [Cloudera Data Science Workbench Engine Versions and Packaging](#) on page 224.

Issues Fixed in Cloudera Data Science Workbench 1.7.1

- Fixed routing issues on proxy-enabled setups. Starting with CDSW 1.7.0, all proxy enabled environments automatically choose a smaller CIDR range for Kubernetes pods (512) and service IPs (256) which also sets the `NO_PROXY` with 512 pod IPs and 256 service IPs.
 This also limits the maximum number of pods/engines that you can run at any given time. You can run approximately 512 pods and around 256 concurrent models at a time. Typically, this is sufficient to run Machine Learning workloads. However, if you need to run a larger concurrent load, then you can fine-tune these settings in the `cdsw.conf` file.
Cloudera Bug: DSE-6737
- Fixed an issue where Cloudera Manager's 5 minute timeout for generating support bundles would lead to CDSW data and metrics missing from the bundles.
Cloudera Bug: DSE-3160
- Fixed an issue where scheduled jobs would not start if scheduled in timezones other than UTC.
Cloudera Bug: DSE-8563
- Fixed an issue where inactive Jupyter sessions in the Workbench would behave in ways that were inconsistent with the rest of the application.
Cloudera Bug: DSE-7867
- Fixed an issue in version 1.6.1 where deployments using a custom Certificate Authority (signed by either their organisation's internal CA or a non-default CA) would have to explicitly set the `REQUESTS_CA_BUNDLE` environmental variable to force Python to use the system truststore.
Cloudera Bug: DSE-7441
- Fixed UI issues where the application did not open project files consistently or as expected.
Cloudera Bug: DSE-6274
- Fixed an issue where CSV files with Chinese characters could not be previewed in the Workbench or Files view.
Cloudera Bug: DSE-4892
- Fixed an issue where CDSW would retain session data for a long time which led to the `var/lib/cdsw` mount filling up with old data that affected application performance.
Cloudera Bug: DSE-3170
- CDSW now clears all iptables rules on application restart. This should fix recurring issues with Kerberos authentication and browser access to the Workbench.
Cloudera Bug: DSE-5095

Cloudera Data Science Workbench 1.6.1

This section lists the release notes for Cloudera Data Science Workbench 1.6.1.

New Features and Changes in Cloudera Data Science Workbench 1.6.1

- **Security**
 - **SAML Identity Provider Logout:** With version 1.6.1, a user clicking the **Sign Out** button on CDSW can also be logged out of their identity provider.

For instructions on how to configure this, see [SAML Identity Provider Configuration](#).

- **Root CA configuration:** Added a new field called **Root CA configuration** to the **Admin > Security** page. Organizations that use an internal custom Certificate Authority can use this field to paste in the contents of their internal CA's root certificate file.

For instructions, see [Configuring Custom Root CA Certificate](#) on page 289.

- **IPv6 Requirement:** Cloudera Data Science Workbench 1.6.x requires you to enable IPv6 on all CDSW gateway hosts. For instructions, refer the workaround provided here: [Known Issue: CDSW cannot start sessions due to connection errors](#).
- **Resource Usage captured in User Events:** You can now [query the `user_events` table](#) for resources used by each session, job, experiment, and model. To see which specific events have this information, see [Tracked User Events](#) on page 251.
- **Editors (Windows only):** For the Windows `cdswctl` client, CDSW now automatically adds the `.exe` extension to the file-name. The format of the downloaded file has also changed from `tar.gz` to `zip`.
- **Kubernetes**
Kubernetes has been upgraded to version 1.13.9.

Issues Fixed in Cloudera Data Science Workbench 1.6.1

- Fixed an issue where deployments using a custom Certificate Authority (signed by either their organisation's internal CA or a non-default CA) would see HTTP Error 500 when attempting to launch the Terminal or Jupyter Notebook sessions from the Workbench
With version 1.6.1, if you are using a custom CA, site administrators can go to the **Admin > Security** page and paste your internal CA's root certificate file contents into the **Root CA configuration** field.
The contents of this field are then inserted into the engine's root certificate store every time a session (or any workload) is launched. This allows processes inside the engine to communicate with the ingress controller.
Cloudera Bug: DSE-7237, DSE-7173
- Fixed an issue where the **Admin > License** page was displaying an incorrect number of CDSW users on a deployment. This count has now been updated to reflect only the total number of *active* users.
Cloudera Bug: DSE-6350
- Fixed an issue where setting `HADOOP_USER_NAME` to the CDSW username had certain unintended consequences. This fix now sets `HADOOP_USER_NAME` to the first part of the Kerberos principal in kerberized environments. In non-kerberized environments, it is still set to the CDSW username.
Cloudera Bug: DSE-6928
- Fixed an issue where sessions on non-kerberized environments would throw the following error even though no principal was provided: `Kerberos principal provided, but no krb5.conf and cluster is not Kerberized`.
Cloudera Bug: DSE-7236
- Fixed an issue in version 1.6.0 where GPUs were not being recognized on air-gapped environments.
Cloudera Bug: DSE-7138
- Fixed an issue where the transition to/from Daylight Savings Time would cause scheduled CDSW jobs to fail.
Cloudera Bug: DSE-3399
- Fixed an issue in CDSW 1.6.0 where CDSW sessions would fail to launch or the web application crashes due to a Node.js issue.
Cloudera Bug: DSE-7238

Engine Upgrade

Cloudera Data Science Workbench 1.7.1 (and later) ships **version 10** of the base engine image which includes the following versions of R and Python:

- **R** - 3.5.1
- **Python** - 2.7.17, 3.6.9

Engine 10 uses Ubuntu 18.04 as its base operating system. This is an upgrade from Engine 8 which used Ubuntu 16.04.

Cloudera Data Science Workbench 1.6.0

This section lists the release notes for Cloudera Data Science Workbench 1.6.0.

New Features and Changes in Cloudera Data Science Workbench 1.6.0

- **Bring Your Own Editor**

You can now take advantage of all the benefits of Cloudera Data Science Workbench while using an editor you are familiar with. This feature supports third-party IDEs that run on your local machine like PyCharm and browser-based IDEs such as Jupyter. Base Image v8 ships with Jupyter preconfigured and can be selected from the **Start Session** menu.

For details, see [Editors](#) on page 154.

- **Multiple Cloudera Data Science Workbench Deployments**

You can now have multiple Cloudera Data Science Workbench CSD deployments associated with one instance of Cloudera Manager.

For details, see [Multiple Cloudera Data Science Workbench Deployments](#) on page 73.

- **Audits**

Cloudera Data Science Workbench logs specific events, such as user logins and sharing, that you can view by querying a database. For more information, see [Monitoring User Events](#) on page 250 and [Tracked User Events](#) on page 251.

- **Expanded Support for Distributed Machine Learning**

Cloudera Data Science Workbench 1.6 (and higher) allows you to run distributed workloads with frameworks such as TensorFlowOnSpark, H2O, XGBoost, and so on. This is similar to what you can already do with Spark workloads that run on the attached CDH/HDP cluster. For details, see [Running Distributed ML Workloads on YARN](#) on page 145.

- **cdswctl CLI Client**

The `cdswctl` client provides an additional way to interact with your Cloudera Data Science Workbench deployment to perform certain actions. For example, you can use the `cdswctl` client to start an SSH-endpoint on your local machine and then connect a local IDE, such as PyCharm, to Cloudera Data Science Workbench.

You can download `cdswctl` from the Cloudera Data Science Workbench web UI and use it from your local machine. Note that this client differs from the `cdsw` CLI tool used to run commands such as `cdsw status`, which exists within the Cloudera Data Science Workbench deployment.

For details, see [cdswctl Command Line Interface Client](#) on page 313.

- **Status and Validate Commands**

The CDSW service in Cloudera Manager now includes two new commands that can be used to assess the status of your Cloudera Data Science Workbench deployment: **Status** and **Validate**. They are the equivalent of the `cdsw status` and `cdsw validate` commands that are available via the CLI.

For details, see [Checking the Status of the CDSW Service](#) on page 258.

- **Experiments**

- If your cluster has been equipped with GPUs, you can now use GPUs to run [experiments](#) on Cloudera Data Science Workbench.
- Tracked experiment files now refresh and appear automatically on the **Overview** page for a run of an experiment. Previously, you had to manually refresh the page after an experiment completes.

- **Command Line Interface (CLI) Changes - RPM Deployments only**

- The `cdsw reset` command has been removed and replaced by the `cdsw stop` command.
- The `cdsw init` command has been removed and replaced by the `cdsw start` command.

For details on how these commands behave on the master and worker hosts, refer to the [Cloudera Data Science Workbench Command Line Reference](#) on page 312.

- **Kubernetes and Weave**

Kubernetes has been upgraded to version 1.13.5. Weave Net has been upgraded to version 2.5.1. This upgrade resolves [Weave issue #2934](#).

- **Logs**

- **Staging Directory**

You can now configure the temporary directory that Cloudera Data Science Workbench uses to stage logs when collecting a diagnostic bundle. Old logs in the directory are deleted when a new diagnostic bundle is collected or when the size grows larger than 10 MB.

- **Logs tab**

Running sessions now display a **Logs** tab. This tab displays engine logs and, if applicable, Spark logs for the running session. Previously, if you wanted to access these logs, that required logging into the Cloudera Data Science Workbench host(s) and the Spark server.

For details, see [Diagnostic Bundles](#) on page 260.

- **Operating System**

Cloudera Data Science Workbench 1.6 supports RHEL and CentOS 7.6.

- **Workload Scheduling Changes**

- Starting with version 1.6, Cloudera Data Science Workbench allows you to specify a list of CDSW gateway hosts that are labeled as **Auxiliary Nodes**. These hosts will be deprioritized during workload scheduling. That is, they will be chosen to run workloads that can't be scheduled on any other hosts. For example, sessions with very large resource requests, or when the other hosts are fully utilized.

For details, see [Customize Workload Scheduling](#) on page 268.

- **Reserve Master Host**

Cloudera Data Science Workbench 1.4.3 introduced a new feature that allowed you to reserve the CDSW Master host for running internal application components. Starting with version 1.6, this feature can be enabled on CSD-based deployments using the **Reserve Master Host** property in Cloudera Manager. Safety valves are no longer needed.

For details, see [Reserving the Master Host for Internal CDSW Components](#) on page 269.

- **Security**

- **FreeIPA Support**

In addition to MIT Kerberos and Active Directory, Cloudera Data Science Workbench now also supports FreeIPA as an identity management system. For details, see [Configure FreeIPA](#) on page 292.

- **New User Role - Operator**

Version 1.6 includes a new access role called **Operator**. When a user is assigned the Operator role on a project, they will be able to start and stop pre-existing jobs and will have view-only access to project code, data, and results.

– Restricting User-Controlled Kubernetes Pods

Cloudera Data Science Workbench 1.6 includes three new properties that allow you to control the permissions granted to user-controlled Kubernetes pods. An example of a user-controlled pod is the engine pod, which provides the environment for sessions, jobs, etc. These pods are launched in a per-user Kubernetes namespace. Since the user has the ability to launch arbitrary pods, these settings restrict what those pods can do.

For details, see [Restricting User-Controlled Kubernetes Pods](#) on page 301.

– LDAP/SAML Configuration Changes

Previously, if you wanted to grant the site administrator role to users of an LDAP/SAML group, that group had to be listed under 2 properties: **LDAP/SAML Full Administrator Groups** and **LDAP/SAML User Groups**. If a group was only listed under **LDAP/SAML Full Administrator Groups**, and not under **LDAP/SAML User Groups**, users of that group would not be able to log in to CDSW.

With version 1.6, you do not need to list the admin groups under both properties. Users belonging to groups listed under **LDAP/SAML Full Administrator Groups** will be able to log in and have site administrator access to Cloudera Data Science Workbench as expected.

– Project and Team Creation

Site administrators can now restrict whether or not users can create projects or teams with the following properties on the **Settings** page:

- **Allow users to create projects**
- **Allow users to create teams**

For details, see [User Access to Features](#) on page 263.

– Session Tokens

The method by which the Cloudera Data Science Workbench web UI session tokens are stored has been hardened. Users must log out of the Cloudera Data Science Workbench web UI and back in after upgrading to version 1.6.0.

– Sharing

Site administrators can now control whether consoles can be shared with the **Allow console output sharing** property on the **Admin > Security** page. Disable this property to remove the **Share** button from the project workspace and workbench UI as well as disable access to all shared console outputs across the deployment. Note that re-enabling this property does not automatically grant access to previously shared consoles. You will need to manually share each console again.

– TLS/SSL

Cloudera Data Science Workbench now defaults to using TLS 1.2. The default cipher suites have also been upgraded to Mozilla's [Modern](#) cipher suites.

• IPv6 Requirement

Cloudera Data Science Workbench 1.6.x requires you to enable IPv6 on all CDSW gateway hosts. For instructions, refer the workaround provided here: [Known Issue: CDSW cannot start sessions due to connection errors](#).

• Spark UI

The Spark UI is now available as a tab within running sessions that use Spark.

Engine Upgrade

Cloudera Data Science Workbench 1.6.0 (and later) ships **version 8** of the base engine image which includes the following versions of R and Python:

- **R** - 3.5.1
- **Python** - 2.7.11, 3.6.8

Pre-installed Packages in Engine 8

For details about the packages included in the base engine, see [Cloudera Data Science Workbench Engine Versions and Packaging](#) on page 224.

(For Upgrades Only) Move Existing Projects to the Latest Base Engine Images

Make sure you test and upgrade existing projects to **Base Image v8 (Project Settings > Engine)** to take advantage of the latest fixes. There are two reasons to do this:

- **Container Security**

Security best practices dictate that engine containers should not run as the root user. Engines (v7 and lower) briefly initialize as the `root` user and then run as the `cdsw` user. Engines v8 (and higher) now follow the best practice and run only as the `cdsw` user. For more details, see [Restricting User-Created Pods](#).

- **CDH 6 Compatibility**

The base engine image you use must be compatible with the version of CDH you are running. This is especially important if you are running workloads on Spark. Older base engines (v6 and lower) cannot support the latest versions of CDH 6. If you want to run Spark workloads on CDH 6, you must upgrade your projects to base engine 7 (or higher).

Incompatible Changes in Cloudera Data Science Workbench 1.6.0

- **SLES 12 SP2, SP3 are not supported with Cloudera Data Science Workbench 1.6.0**

SLES 12 SP2 and SP3 have reached the [end of general support with SUSE](#) and will not be supported with Cloudera Data Science Workbench 1.6.0 (and higher).

- **GPU Setup Changes**

- `nvidia-docker1` is no longer supported.
- The **NVIDIA Library Path** property is no longer available.

Cloudera Data Science Workbench 1.6 ships with `nvidia-docker2` installed by default. The path to the NVIDIA library volumes is also set automatically when GPUs are enabled. Review the revised GPU setup steps here: [Enabling Cloudera Data Science Workbench to use GPUs](#) on page 138.

- The `CDSW_PUBLIC_PORT` environment variable has been deprecated and will be removed in a future release. Use `CDSW_APP_PORT` or `CDSW_READONLY_PORT` environment variables instead.

For details, see [Engine Environment Variables](#) on page 215.

Issues Fixed in Cloudera Data Science Workbench 1.6.0

- Fixed an issue where you had to include `pd.options.display.html.table_schema = True` to show a horizontal scroll bar for Pandas Dataframe if there were too many columns. You no longer have to include the property.

Cloudera Issue: DSE-3562

- Fixed an issue where the built-in Workbench editor did not properly recognize imported code that uses tabs instead of spaces. This also resolves navigation issues that occurred within the editor when working with imported code that uses tabs.

Cloudera Issue: DSE-2976, DSE-3221

- Fixed an issue where an email with attachments triggered by a job fail to send if the attachment is over 4 MB.

Cloudera Issue: DSE-5980, DSE-6003

- Fixed an issue where large R scripts hang when run in the built-in Workbench editor.

Cloudera Issue: DSE-2817

- Fixed an issue where `.md` files were not rendered in Markdown. Previously, only `README.md` was rendered correctly.

Cloudera Issue: DSE-3315

- Fixed an issue with `predict.py`, the model training script in the Python template project.

Cloudera Issue: DSE-5314

- Fixed an issue where logs generated by the Cloudera Data Science Workbench diagnostic bundle were occupying too much space in the `/var/log/cdsw` directory. The size of the generated bundle has been reduced and you can now configure a temporary staging directory to be used when a diagnostic bundle is generated.

Cloudera Issue: DSE-5921

- The `cdsw-build.sh` script used with models and experiments now runs as the `cdsw` user.

Cloudera Issue: DSE-4340

- The changes to GPU support in version 1.6 have also fixed an issue where GPUs were not automatically detected after a machine reboot.

Cloudera Issue: DSE-2847

- Fixed an issue where iFrame visualizations would not render in the Workbench due to the new HTTP security headers added in version 1.4.x.

Cloudera Issue: DSE-5274

Known Issues and Limitations in Cloudera Data Science Workbench 1.6.0

For a complete list, see [Known Issues and Limitations in Cloudera Data Science Workbench 1.7.2](#) on page 52.

Cloudera Data Science Workbench 1.5.0

This section lists the release notes for Cloudera Data Science Workbench 1.5.0.

New Features and Changes in Cloudera Data Science Workbench 1.5.0

- **Cloudera Enterprise 6.1 Support**

Cloudera Data Science Workbench is now supported with Cloudera Manager 6.1.x (and higher) and CDH 6.1.x (and higher). For details, see [Cloudera Manager and CDH Requirements](#) on page 65.

- **Cloudera Data Science Workbench on Hortonworks Data Platform (HDP)**

Cloudera Data Science Workbench can now be deployed on HDP 2.6.5 and HDP 3.1.0. For an architecture overview and installation instructions, see [Deploying Cloudera Data Science Workbench 1.7.2 on Hortonworks Data Platform](#) on page 103.

- **Security Enhancements**

- **Allow Site Administrators to Enable/Disable Project Uploads and Downloads** - By default, all Cloudera Data Science Workbench users are allowed to upload and download files to/from a project. Version 1.5 introduces

a new feature flag that allows site administrators to hide the UI features that let users upload and download project files.

Note that this feature flag only removes the relevant features from the Cloudera Data Science Workbench UI. It does not disable the ability to upload and download files through the backend web API.

For details on how to enable this feature, see [Disabling Project File Uploads and Downloads](#) on page 129.

- **OpenJDK Support**

Cloudera Data Science Workbench now supports Open JDK 8 on Cloudera Enterprise 5.16.1 (and higher). For details, see [Product Compatibility Matrix - Supported JDK Versions](#).

- **Engines**

- [Base engine upgraded with a new version of R - 3.5.1 \(Base Image v7\)](#)
- **Debugging Improvements** - Previously, engines and their associated logs were deleted immediately after an exit or a crash. With version 1.5, engines are now available for about 5 minutes after they have ended to allow you to collect the relevant logs.

Additionally, when an engine exits with a non-zero status code, the last 50 lines from the engine's logs are now printed to the Workbench console. Note that a non-zero exit code and the presence of engine logs in the Workbench does not always imply a problem with the code. Events such as session timeouts and out-of-memory issues are also assigned non-zero exit codes and will display engine logs.

- **Installation and Upgrade**

- **New Configuration Parameters** - Version 1.5 includes three new configuration parameters that can be used to specify the type of distribution you are running, the directory for the installed packages/parcels, and the path where Anaconda is installed (for HDP only).

- DISTRO
- DISTRO_DIR
- ANACONDA_DIR

Details and sample values for these properties have been added to the relevant installation topics for [CDH](#) and [HDP](#).

- **DOCKER_TMPDIR changed to `/var/lib/cdswh/tmp/docker`** - Previously the Cloudera Data Science Workbench installer would temporarily decompress the base engine image file to the `/var/lib/docker/tmp` directory. Starting with version 1.5, the installer will use the `/var/lib/cdswh/tmp/docker` directory instead. Make sure you have an Application block device mounted to `/var/lib/cdswh` as [recommended](#) so that installation/upgrade can proceed without issues.
- **Improved Validation Checks** - Improved the validation checks run by the installer and the error messages that are displayed during the installation process. Cloudera Data Science Workbench now:
 - Checks that space is available on the root directory, the Application Block Device and the Docker Block Device(s).
 - Checks that DNS forward and reverse lookup works for the Cloudera Data Science Workbench Domain and Master IP address provided.
 - Displays better error messages for the `cdsw status` and `cdsw validate` commands for easier debugging.

- **Command Line**

- **cdsw logs** - Previously, the `cdsw logs` command generated two log bundles - one in plaintext and one with sensitive information redacted. With version 1.5, the command now generates only a single bundle that has all the sensitive information redacted by default.

To turn off redaction of log files for internal use, you can use the new `--skip-redaction` option as follows:

```
cdsb logs --skip-redaction
```

- **Networking**

- Cloudera Data Science Workbench now uses DNS hostnames (not IP addresses) for internal communication between components. As a result, the wildcard DNS hostname configured for Cloudera Data Science Workbench must now be resolvable from both, the CDSW cluster, and your browser.
- Cloudera Data Science Workbench now enables IPv4 forwarding (`net.ipv4.conf.default.forwarding`) during the installation process.

Engine Upgrade

Cloudera Data Science Workbench 1.5.0 (and later) ships **version 7** of the base engine image which includes the following versions of R and Python:

- **R** - 3.5.1
- **Python** - 2.7.11, 3.6.1

Pre-installed Packages in Engine 7 - For details about the packages included in the base engine, see [Cloudera Data Science Workbench Engine Versions and Packaging](#) on page 224.

Upgrade Projects to Use the Latest Base Engine Images - Make sure you test and upgrade existing projects to **Base Image v7 (Project Settings > Engine)** to take advantage of the latest fixes.

Note that this is a required step if you are upgrading to using Cloudera Data Science Workbench on CDH 6.

The base engine image you use must be compatible with the version of CDH you are running. This is especially important if you are running workloads on Spark. Older base engines (v6 and lower) cannot support the latest versions of CDH 6. That is because these engines were configured to point to the Spark 2 parcel. However, on CDH 6 clusters, Spark is now packaged as a part of CDH 6 and the separate add-on Spark 2 parcel is no longer supported. If you want to run Spark workloads on CDH 6, you must upgrade your projects to base engine 7 (or higher).

Table 1: CDSW Base Engine Compatibility for Spark Workloads on CDH 5 and CDH 6

Base Engine Versions	CDH 5	CDH 6
Base engines 6 (and lower)	Yes	No
Base engines 7 (and higher)	Yes	Yes

Incompatible Changes in Cloudera Data Science Workbench 1.5.0

Deprecated Property - CDH Parcel Directory

The **CDH parcel directory** property is no longer available in the Site Administration panel at **Admin > Engines**. Depending on your deployment, use one of the following ways to configure this property:

- **CSD deployments:** If you are using the default parcel directory, `/opt/cloudera/parcels`, no action is required. If you want to use a custom location for the parcel directory, configure this in Cloudera Manager as documented [here](#).
- **RPM deployments:** If you are using the default parcel directory, `/opt/cloudera/parcels`, no action is required. If you want to specify a custom location for the parcel directory, configure the `DISTRO_DIR` property in the `cdsw.conf` file on both master and worker hosts. Run `cdsw restart` after you make this change.

Issues Fixed in Cloudera Data Science Workbench 1.5.0

- Fixed an issue with RPM installations where `NO_PROXY` settings were being ignored.

Cloudera Data Science Workbench Release Notes

Cloudera Bug: DSE-4444

- Fixed an issue where CDSW would not start because of IP issues with web pods. Version 1.5 fixes this by enabling IPv4 forwarding at startup.

Cloudera Bug: DSE-4609

- Fixed an issue where engines would get deleted immediately after an exit/crash and engine logs did not persist which made it difficult to debug issues with crashes or auto-restarts.

Cloudera Bug: DSE-4008, DSE-4417

- Fixed intermittent issues with starting and stopping Cloudera Data Science Workbench on CSD deployments.

Cloudera Bug: DSE-4426, DSE-4829

- Fixed an issue where Cloudera Data Science Workbench was reporting incorrect file sizes for files larger than 2 MB.

Cloudera Bug: DSE-4531, DSE-4532

- Fixed an issue where the Run New Experiment dialog box did not include the file selector and the Script name had to be typed in manually.

Cloudera Bug: DSE-3650

- Fixed an issue where underlying Kubernetes processes were running out of resources leading to Out of Memory (OOM) errors. Cloudera Data Science Workbench now reserves compute resources for Kubernetes components.

Cloudera Bug: DSE-4896, DSE-5001

- Fixed an issue where the `PYSPARK3_PYTHON` environment variable was not working as expected for Python 3 workloads.

Cloudera Bug: DSE-4329

- Fixed an issue where Docker commands would fail on Cloudera Data Science Workbench engines that are not available locally (such as custom engine images) when an HTTP/HTTPS proxy was in use.

Cloudera Bug: DSE-4427

- Fixed an issue where installation of the `XML` package would fail in the R kernel.

Cloudera Bug: DSE-2201

Known Issues and Limitations in Cloudera Data Science Workbench 1.5.0

For a complete list of the current known issues and limitations in Cloudera Data Science Workbench 1.5.x, see [Known Issues in Cloudera Data Science Workbench 1.5.x](#).

Cloudera Data Science Workbench 1.4.3

This section lists the release notes for Cloudera Data Science Workbench 1.4.3.

New Features and Changes in Cloudera Data Science Workbench 1.4.3

- **Reserve Master Host for Internal Application Components**

Cloudera Data Science Workbench now allows you to reserve the master host for running internal application components and services such as Livelog, the PostgreSQL database, and so on, while user workloads run exclusively on worker hosts.

By default, the master host runs both, user workloads as well as the application's internal services. However, depending on the size of your CDSW deployment and the number of workloads running at any given time, it's

possible that user workloads might dominate resources on the master host. Enabling this feature will ensure that CDSW's application components always have access to the resources they need on the master host and are not adversely affected by user workloads.



Important: This feature only applies to deployments with more than one Cloudera Data Science Workbench host. Enabling this feature on single-host deployments will leave Cloudera Data Science Workbench incapable of scheduling any workloads.

For details on how to enable this feature, see [Reserving the Master Host for Internal CDSW Components](#) on page 269.

- **Allow Only Session Creators to Execute Commands in Active Sessions**

By default, project contributors, project administrators, and site administrators have the ability to execute commands within your actively running sessions in the Workbench. Cloudera Data Science Workbench 1.4.3 introduces a new feature that allows site administrators to restrict this ability. When this feature is enabled, only the user that creates the session will be able to execute commands in that session. No other users, regardless of their permissions in the team or as project collaborators/administrators, will be able to execute commands on active sessions that were not created by them.

For details on how to enable this feature, see [Restricting Collaborator and Administrator Access to Active Sessions](#) on page 150.

Issues Fixed in Cloudera Data Science Workbench 1.4.3

TSB-349: SQL Injection Vulnerability in Cloudera Data Science Workbench

An SQL injection vulnerability was found in Cloudera Data Science Workbench. This would allow any authenticated user to run arbitrary queries against CDSW's internal database. The database contains user contact information, bcrypt-hashed CDSW passwords (in the case of local authentication), API keys, and stored Kerberos keytabs.

Products affected: Cloudera Data Science Workbench (CDSW)

Releases affected: CDSW 1.4.0, 1.4.1, 1.4.2

Users affected: All

Date/time of detection: 2018-10-18

Detected by: Milan Magyar (Cloudera)

Severity (Low/Medium/High): Critical (9.9): [CVSS:3.0/AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H](#)

Impact: An authenticated CDSW user can arbitrarily access and modify the CDSW internal database. This allows privilege escalation in CDSW, Kubernetes, and the Linux host; creation, deletion, modification, and exfiltration of data, code, and credentials; denial of service; and data loss.

CVE: CVE-2018-20091

Immediate action required:

1. Strongly consider performing a backup before beginning. We advise you to have a [backup](#) before performing any upgrade and before beginning this remediation work.
2. [Upgrade to Cloudera Data Science Workbench 1.4.3 \(or higher\)](#).
3. In an abundance of caution Cloudera recommends that you revoke credentials and secrets stored by CDSW. To revoke these credentials:
 - a. Change the password for any account with a keytab or kerberos credential that has been stored in CDSW. This includes the Kerberos principals for the associated CDH cluster if entered on the CDSW "Hadoop Authentication" user settings page.

- b. With Cloudera Data Science Workbench 1.4.3 running, run the following remediation script *on each CDSW host*, including the master and all workers: [Remediation Script for TSB-349](#)

Note: Cloudera Data Science Workbench will become unavailable during this time.

- c. The script performs the following actions:
 - a. If using local user authentication, logs out every user and resets their CDSW password.
 - b. Regenerates or deletes various keys for every user.
 - c. Resets secrets used for internal communications.
 - d. Fully stop and start Cloudera Data Science Workbench (a restart is not sufficient).
 - For CSD-based deployments, [restart the CDSW service](#) in Cloudera Manager.

OR

 - For RPM-based deployments, run `cdsw stop` followed by `cdsw start` on the CDSW master host.
 - e. If using internal TLS termination: [revoke and regenerate the CDSW TLS certificate and key](#).
 - f. For each user, revoke the previous CDSW-generated [SSH public key for git integration on the git side](#) (the private key in CDSW has already been deleted). A new SSH key pair has already been generated and should be installed in the old key's place.
 - g. Revoke and regenerate any credential stored within a CDSW project, including any passwords stored in projects' environment variables.
4. Verify all CDSW settings to ensure they are unchanged (e.g. SMTP server, authentication settings, custom docker images, host mounts, etc).
 5. Treat all CDSW hosts as potentially compromised with root access. Remediate per your policy.

Addressed in release/refresh/patch: Cloudera Data Science Workbench 1.4.3

For the latest update on this issue see the corresponding Knowledge article:

[TSB 2019-349: CDSW SQL Injection Vulnerability](#)

[TSB-350: Risk of Data Loss During Cloudera Data Science Workbench \(CDSW\) Shutdown and Restart](#)

Stopping Cloudera Data Science Workbench involves unmounting the NFS volumes that store CDSW project directories and then cleaning up a folder where CDSW stores its temporary state. However, due to a race condition, this NFS unmount process can take too long or fail altogether. If this happens, any CDSW projects that remain mounted will be deleted.

TSB-2018-346 was released in the time-frame of CDSW 1.4.2 to fix this issue, but it only turned out to be a partial fix. With CDSW 1.4.3, we have fixed the issue permanently. However, the [script](#) that was provided with TSB-2018-346 still ensures that data loss is prevented and must be used to shutdown/restart all the affected CDSW released listed below. The same script is also available under the **Immediate Action Required** section below.

Products affected: Cloudera Data Science Workbench

Releases affected: Cloudera Data Science Workbench versions

- 1.0.x
- 1.1.x
- 1.2.x
- 1.3.0, 1.3.1

- 1.4.0, 1.4.1, 1.4.2

Users affected: This potentially affects all CDSW users.

Detected by: Nehmé Tohmé (Cloudera)

Severity (Low/Medium/High): High

Impact: Potential data loss.

CVE: N/A

Immediate action required: If you are running any of the affected Cloudera Data Science Workbench versions, you must run the following script on the CDSW master host every time before you stop or restart Cloudera Data Science Workbench. Failure to do so can result in data loss.

This script should also be run before initiating a Cloudera Data Science Workbench upgrade. As always, we recommend [creating a full backup](#) prior to beginning an upgrade.

cdsw_protect_stop_restart.sh - Available for download at: [cdsw_protect_stop_restart.sh](#).

```
#!/bin/bash
```

```
set -e
```

```
cat << EXPLANATION
```

```
This script is a workaround for Cloudera TSB-346 and TSB-350. It protects your CDSW projects from a rare race condition that can result in data loss. Run this script before stopping the CDSW service, irrespective of whether the stop precedes a restart, upgrade, or any other task.
```

```
Run this script only on the master node of your CDSW cluster.
```

```
You will be asked to specify a target folder on the master node where the script will save a backup of all your project files. Make sure the target folder has enough free space to accommodate all of your project files. To determine how much space is required, run 'du -hs /var/lib/cdsw/current/projects' on the CDSW master node.
```

```
This script will first back up your project files to the specified target folder. It will then temporarily move your project files aside to protect against the data loss condition. At that point, it is safe to stop the CDSW service. After CDSW has stopped, the script will move the project files back into place.
```

```
Note: This workaround is not required for CDSW 1.4.3 and higher.
```

```
EXPLANATION
```

```
read -p "Enter target folder for backups: " backup_target
```

```
echo "Backing up to $backup_target..."
```

```
rsync -azp /var/lib/cdsw/current/projects "$backup_target"
```

```
read -n 1 -p "Backup complete. Press enter when you are ready to stop CDSW: "
```

```
echo "Deleting all Kubernetes resources..."
```

```
kubectl delete
```

```
configmaps,deployments,daemonsets,replicasets,services,ingress,secrets,persistentvolumes,persistentvolumeclaims,jobs --all
```

```
kubectl delete pods --all
```

```
echo "Temporarily saving project files to /var/lib/cdsw/current/projects_tmp..."
```

```
mkdir /var/lib/cdsw/current/projects_tmp
```

```
mv /var/lib/cdsw/current/projects/* /var/lib/cdsw/current/projects_tmp
```

```
echo -e "Please stop the CDSW service."
```

```
read -n 1 -p "Press enter when CDSW has stopped: "  
  
echo "Moving projects back into place..."  
mv /var/lib/cdsw/current/projects_tmp/* /var/lib/cdsw/current/projects  
rm -rf /var/lib/cdsw/current/projects_tmp  
  
echo -e "Done. You may now upgrade or start the CDSW service."  
echo -e "When CDSW is running, if desired, you may delete the backup data at  
$backup_target"
```

Addressed in release/refresh/patch: This issue is fixed in Cloudera Data Science Workbench 1.4.3.

Note that you are required to run the workaround script above when you upgrade from an affected version to a release with the fix. This helps guard against data loss when the affected version needs to be shut down during the upgrade process.

TSB-351: Unauthorized Project Access in Cloudera Data Science Workbench

Malicious CDSW users can bypass project permission checks and gain read-write access to any project folder in CDSW.

Products affected: Cloudera Data Science Workbench

Releases affected: Cloudera Data Science Workbench 1.4.0, 1.4.1, 1.4.2

Users affected: All CDSW Users

Date/time of detection: 10/29/2018

Detected by: Che-Yuan Liang (Cloudera)

Severity (Low/Medium/High): High (8.3: [CVSS:3.0/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:L](#))

Impact: Project data can be read or written (changed, destroyed) by any Cloudera Data Science Workbench user.

CVE: CVE-2018-20090

Immediate action required:

Upgrade to a version of Cloudera Data Science Workbench with the fix (version 1.4.3, 1.5.0, or higher).

Addressed in release/refresh/patch: Cloudera Data Science Workbench 1.4.3 (and higher)

For the latest update on this issue see the corresponding Knowledge article:

[TSB 2019-351: Unauthorized Project Access in Cloudera Data Science Workbench](#)

Other Notable Fixed Issues in Cloudera Data Science Workbench 1.4.3

- Fixed an issue where malicious Cloudera Data Science Workbench users were able to bypass project permission checks and gain read-write access to any project folder in Cloudera Data Science Workbench.
Cloudera Bug: DSE-5138
- Fixed an issue where Cloudera Data Science Workbench would become unresponsive because the web application was making too many simultaneous requests to the Kubernetes API server. CDSW now caches calls to the API and refreshes the cache periodically.
Cloudera Bug: DSE-5265, DSE-5269
- Fixed an issue where Cloudera Data Science Workbench workloads would intermittently crash with `Exit Code 2: Misuse of Shell builtins`.
Cloudera Bug: DSE-4709
- Fixed an issue where Cloudera Data Science Workbench would not start when internal TLS termination was enabled and the TLS private key/certificate pair in use did not include a trailing newline character.
Cloudera Bug: DSE-4853

Known Issues and Limitations in Cloudera Data Science Workbench 1.4.3

For a complete list of the current known issues and limitations in Cloudera Data Science Workbench 1.4.x, see [Known Issues and Limitations in Cloudera Data Science Workbench 1.7.2](#) on page 52.

Cloudera Data Science Workbench 1.4.2



Note: Cloudera Data Science Workbench 1.4.2 is the next official maintenance release after Cloudera Data Science Workbench 1.4.0. Version 1.4.1 is no longer publicly available.

This section lists the release notes for Cloudera Data Science Workbench 1.4.2.

New Features and Changes in Cloudera Data Science Workbench 1.4.2

- **Operating System:** Added support for RHEL / CentOS / Oracle Linux RHCK 7.5.
- **Engines**
 - Mounts - By default, host mounts (specified at **Admin > Engines > Mounts**) are loaded into engine containers with read-only permissions. With version 1.4.2, [a new checkbox](#) allows you to make these mounted directories available in engine containers with **read-write** permissions instead.
 - [Engine upgrade \(Base Image v6\)](#)
- **Models**
 - In Cloudera Data Science Workbench 1.4.0, model request sizes were limited to 100 KB. In version 1.4.2, this limit has now been increased to **5 MB**. To take advantage of this higher threshold, you will need to upgrade to Cloudera Data Science Workbench 1.4.2 and rebuild your existing models.
- **Security**

Added three new properties to the **Admin > Security** page that allow you to customize HTTP headers accepted by Cloudera Data Science Workbench.

 - Enable HTTP security headers
 - Enable cross-origin resource sharing (CORS)
 - Enable HTTP Strict Transport Security (HSTS)

For details, see [Configuring HTTP Headers for Cloudera Data Science Workbench](#) on page 300.

Engine Upgrade

Cloudera Data Science Workbench 1.4.2 ships **version 6** of the base engine image which includes the following versions of R and Python:

- **R** - 3.4.1
- **Python** - 2.7.11, 3.6.1

Pre-installed Packages in Engine 6 - For details about the packages included in the base engine, see [Cloudera Data Science Workbench Engine Versions and Packaging](#) on page 224.

Additionally, Cloudera Data Science Workbench will now alert you when a new engine version is available. Make sure you test and upgrade existing projects to **Base Image v6 (Project Settings > Engine)** to take advantage of the latest fixes.

Issues Fixed in Cloudera Data Science Workbench 1.4.2

TSB-346: Risk of Data Loss During Cloudera Data Science Workbench (CDSW) Shutdown and Restart

Stopping Cloudera Data Science Workbench involves unmounting the NFS volumes that store CDSW project directories and then cleaning up a folder where the kubelet stores its temporary state. However, due to a race condition, this NFS unmount process can take too long or fail altogether. If this happens, CDSW projects that remain mounted will be deleted by the cleanup step.

Products affected: Cloudera Data Science Workbench

Releases affected: Cloudera Data Science Workbench versions -

- 1.0.x
- 1.1.x
- 1.2.x
- 1.3.0, 1.3.1
- 1.4.0, 1.4.1

Users affected: This potentially affects all CDSW users.

Detected by: Nehmé Tohmé (Cloudera)

Severity (Low/Medium/High): High

Impact: If the NFS unmount fails during shutdown, data loss can occur. All CDSW project files might be deleted.

CVE: N/A

Immediate action required: If you are running any of the affected Cloudera Data Science Workbench versions, you must run the following script on the CDSW master host every time before you stop or restart Cloudera Data Science Workbench. Failure to do so can result in data loss.

This script should also be run before initiating a Cloudera Data Science Workbench upgrade. As always, we recommend [creating a full backup](#) prior to beginning an upgrade.

csw_protect_stop_restart.sh - Available for download at: [csw_protect_stop_restart.sh](#).

```
#!/bin/bash
```

```
set -e
```

```
cat << EXPLANATION
```

```
This script is a workaround for Cloudera TSB-346. It protects your CDSW projects from a rare race condition that can result in data loss. Run this script before stopping the CDSW service, irrespective of whether the stop precedes a restart, upgrade, or any other task.
```

```
Run this script only on the master node of your CDSW cluster.
```

```
You will be asked to specify a target folder on the master node where the script will save a backup of all your project files. Make sure the target folder has enough free space to accommodate all of your project files. To determine how much space is required, run 'du -hs /var/lib/csw/current/projects' on the CDSW master node.
```

```
This script will first back up your project files to the specified target folder. It will then temporarily move your project files aside to protect against the data loss condition. At that point, it is safe to stop the CDSW service. After CDSW has stopped, the script will move the project files back into place.
```

```
Note: This workaround is not required for CDSW 1.4.2 and higher.
```

EXPLANATION

```

read -p "Enter target folder for backups: " backup_target

echo "Backing up to $backup_target..."
rsync -azp /var/lib/cds/current/projects "$backup_target"

read -n 1 -p "Backup complete. Press enter when you are ready to stop CDSW: "

echo "Deleting all Kubernetes resources..."
kubectl delete
configmaps,deployments,daemonsets,replicasets,services,ingress,secrets,persistentvolumes,persistentvolumeclaims,jobs
--all
kubectl delete pods --all

echo "Temporarily saving project files to /var/lib/cds/current/projects_tmp..."
mkdir /var/lib/cds/current/projects_tmp
mv /var/lib/cds/current/projects/* /var/lib/cds/current/projects_tmp

echo -e "Please stop the CDSW service."

read -n 1 -p "Press enter when CDSW has stopped: "

echo "Moving projects back into place..."
mv /var/lib/cds/current/projects_tmp/* /var/lib/cds/current/projects
rm -rf /var/lib/cds/current/projects_tmp

echo -e "Done. You may now upgrade or start the CDSW service."
echo -e "When CDSW is running, if desired, you may delete the backup data at
$backup_target"

```

Addressed in release/refresh/patch: This issue is fixed in Cloudera Data Science Workbench 1.4.2.

Note that you are required to run the workaround script above when you upgrade from an affected version to a release with the fix. This helps guard against data loss when the affected version needs to be shut down during the upgrade process.

For the latest update on this issue see the corresponding Knowledge article:

[TSB 2018-346: Risk of Data Loss During Cloudera Data Science Workbench \(CDSW\) Shutdown and Restart](#)

TSB-328: Unauthenticated User Enumeration in Cloudera Data Science Workbench

Unauthenticated users can get a list of user accounts of Cloudera Data Science Workbench.

Products affected: Cloudera Data Science Workbench

Releases affected: Cloudera Data Science Workbench 1.4.0 (and lower)

Users affected: All users of Cloudera Data Science Workbench 1.4.0 (and lower)

Date/time of detection: June 11, 2018

Severity (Low/Medium/High): 5.3 (Medium) [CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N](#)

Impact: Unauthenticated user enumeration in Cloudera Data Science Workbench.

CVE: CVE-2018-15665

Immediate action required: [Upgrade](#) to the latest version of Cloudera Data Science Workbench (1.4.2 or higher).

Note that Cloudera Data Science Workbench 1.4.1 is no longer publicly available due to [TSB 2018-346: Risk of Data Loss During Cloudera Data Science Workbench \(CDSW\) Shutdown and Restart](#).

Addressed in release/refresh/patch: Cloudera Data Science Workbench 1.4.2 (and higher)

For the latest update on this issue see the corresponding Knowledge article:

[TSB 2018-318: Unauthenticated User Enumeration in Cloudera Data Science Workbench](#)

Other Notable Fixed Issues in Cloudera Data Science Workbench 1.4.2

- Fixed an issue where attempting to fork a large project would result in unexpected 'out of memory' errors.
Cloudera Bug: DSE-4464
- Fixed an issue in version 1.4.0 where Cloudera Data Science Workbench workloads would intermittently get stuck in the Scheduling state due to a Red Hat kernel slab leak.
Cloudera Bug: DSE-4098
- Fixed an issue in version 1.4.0 where the Hadoop username on non-kerberized clusters defaulted to `cdsw`. This was a [known issue](#) and has been fixed in version 1.4.2. The Hadoop username will now once again default to your Cloudera Data Science Workbench username.
Cloudera Bug: DSE-4240
- Fixed an issue in version 1.4.0 where creating a project using Git via SSH did not work.
Cloudera Bug: DSE-4278
- Fixed an issue in version 1.4.0 where environmental variables set in the **Admin** panel were not being propagated to projects (experiments, sessions, jobs) as expected.
Cloudera Bug: DSE-4422
- Fixed an issue in version 1.4.0 where Cloudera Data Science Workbench would not start when external TLS termination was enabled.
Cloudera Bug: DSE-4640
- Fixed an issue in version 1.4.0 where HTTP/HTTPS proxy settings in Cloudera Manager were erroneously escaped when propagated to Cloudera Data Science Workbench engines.
Cloudera Bug: DSE-4421
- Fixed an issue in version 1.4.0 where [SSH tunnels](#) did not work as expected.
Cloudera Bug: DSE-4741
- Fixed an issue in version 1.4.0 where copying multiple files into a folder resulted in unexpected behavior such as overwritten files and incorrect UI messages.
Cloudera Bug: DSE-4831
- Fixed an issue in version 1.4.0 where [workers](#) (in engines) and collection of usage metrics failed on TLS-enabled clusters.
Cloudera Bug: DSE-4293, DSE-4572
- Fixed an issue in version 1.4.0 where the **Files > New Folder** dialog box did not work.
Cloudera Bug: DSE-4807
- Fixed an issue in version 1.4.0 where deleting an experiment did not work from certain dashboards. Consequently, deleting the parent project would also fail in such cases.
Cloudera Bug: DSE-4227

Known Issues and Limitations in Cloudera Data Science Workbench 1.4.2

For a complete list of the current known issues and limitations in Cloudera Data Science Workbench 1.4.x, see [Known Issues and Limitations in Cloudera Data Science Workbench 1.7.2](#) on page 52.

Cloudera Data Science Workbench 1.4.0

This section lists the release notes for Cloudera Data Science Workbench 1.4.0.

New Features in Cloudera Data Science Workbench 1.4.0

- **Models and Experiments** - Cloudera Data Science Workbench 1.4 extends the machine learning platform experience from research to production. Now you can use Cloudera Data Science Workbench to build, train, and deploy models in a unified workflow.
 - [Experiments](#) - Train and compare versioned, reproducible models
 - [Models](#) - Deploy and manage models as REST APIs to serve predictions
- **External Authentication**
 - LDAP/SAML users can now restrict access to Cloudera Data Science Workbench to specific LDAP/SAML groups. Additionally, you can now specify groups that should automatically be granted site administrator privileges when they log in to Cloudera Data Science Workbench. For details, see [Configuring External Authentication with LDAP and SAML](#) on page 293.
 - Cloudera Data Science Workbench now supports multiple identity provider signing certificates for SAML 2.0 authentication.
 - Cloudera Data Science Workbench now supports SAML 2.0 Errata 05 E43 for SAML 2.0 authentication.
- **Projects and Workbench**
 - Site administrators can now disable individual built-in template projects by using a checkbox in the **Project Templates** table at **Admin > Settings**. Only enabled project templates will be displayed in the dropdown menu when [creating a new project](#).
 - The default `.gitignore` file that is created with each new project has been updated to:

```
R
node_modules
*.pyc
.*
!.gitignore
```

- Added support for multiple Terminal windows within a single session.

- **Networking**
 - Cloudera Data Science Workbench now supports DNS resolution of localhost to non-local IP address (not 127.0.0.1).
 - Cloudera Data Science Workbench now appends the following default values to the `NO_PROXY` parameter *if* any of the following properties are configured: `HTTP_PROXY`, `HTTPS_PROXY`, or `ALL_PROXY`.

```
"127.0.0.1,localhost,100.66.0.1,100.66.0.2,100.66.0.3,
100.66.0.4,100.66.0.5,100.66.0.6,100.66.0.7,100.66.0.8,100.66.0.9,
100.66.0.10,100.66.0.11,100.66.0.12,100.66.0.13,100.66.0.14,
100.66.0.15,100.66.0.16,100.66.0.17,100.66.0.18,100.66.0.19,
100.66.0.20,100.66.0.21,100.66.0.22,100.66.0.23,100.66.0.24,
100.66.0.25,100.66.0.26,100.66.0.27,100.66.0.28,100.66.0.29,
100.66.0.30,100.66.0.31,100.66.0.32,100.66.0.33,100.66.0.34,
100.66.0.35,100.66.0.36,100.66.0.37,100.66.0.38,100.66.0.39,
100.66.0.40,100.66.0.41,100.66.0.42,100.66.0.43,100.66.0.44,
100.66.0.45,100.66.0.46,100.66.0.47,100.66.0.48,100.66.0.49,
100.66.0.50,100.77.0.10,100.77.0.128,100.77.0.129,100.77.0.130,
100.77.0.131,100.77.0.132,100.77.0.133,100.77.0.134,100.77.0.135,
100.77.0.136,100.77.0.137,100.77.0.138,100.77.0.139"
```

Cloudera Data Science Workbench Release Notes

- **Installation Validation Checks** - Improved validation checks run during the installation process. Cloudera Data Science Workbench now:
 - Verifies that the wildcard DNS subdomain has been configured.
 - Verifies that `resolv.conf` is not pointing to `127.0.0.1`.
 - Validates iptables chains to ensure there are no custom rules being set.
 - Throws a warning if you are using a self-signed TLS certificate, an expired certificate, or if the certificate is not valid for the wildcard domain used for Cloudera Data Science Workbench.
- **Command Line** - Added a verbose option to the `cdsw status` command.

```
cdsw status [-v|--verbose]
```

- Kubernetes has been upgraded to version 1.8.12.

Engine Upgrade

Cloudera Data Science Workbench 1.4.0 (and later) ships **version 5** of the base engine image which includes the following versions of R and Python:

- **R** - 3.4.1
- **Python** - 2.7.11, 3.6.1

Pre-installed Packages in Engine 5 - For details about the packages included in the base engine, see [Cloudera Data Science Workbench Engine Versions and Packaging](#) on page 224.

Additionally, Cloudera Data Science Workbench will now alert you when a new engine version is available. Make sure you test and upgrade existing projects to **Base Image v5 (Project Settings > Engine)** to take advantage of the latest fixes.

Incompatible Changes in Cloudera Data Science Workbench 1.4.0

Host Mounts are now Read-Only in Engines - Previously, mounts (specified at **Admin > Engines > Mounts**) were loaded into engine containers with read-write permissions.

Starting with version 1.4.0, mount points are now loaded into engines with **read-only** permissions.

Issues Fixed in Cloudera Data Science Workbench 1.4.0

- Fixed an issue where Git would timeout when cloning a project took too long. The timeout has now been increased to 60 seconds when creating a new project from Git.
Cloudera Bug: DSE-3363
- Fixed an issue where manual parcel deployments could not detect parcel hash files with a `.sha1` extension.
Cloudera Bug: DSE-3301
- Fixed several usability issues (file create, save, and so on) with Internet Explorer 11.
Cloudera Bug: DSE-3426, DSE-3434
- Fixed an issue where CSD installations would fail to recognize Oracle Linux 7.3 as a supported operating system.
Cloudera Bug: DSE-3257
- Fixed an issue where Cloudera Data Science Workbench would hang with 100 percent CPU utilization.
Cloudera Bug: DSE-3450
- Fixed a SAML 2.0 configuration issue where uploading the identity provider metadata XML file did not update identity provider signing certificate and/or SSO URL on Cloudera Data Science Workbench correctly.
Cloudera Bug: DSE-3076

- Fixed an issue with SAML 2.0 authentication where the identity provider's signature was not being validated correctly.
Cloudera Bug: DSE-3694
- Fixed the **Save As** functionality in the project Workbench.
Cloudera Bug: DSE-3870
- Fixed an issue where if a user had some files opened in the Workbench in a previous session, and those files no longer existed in the project filesystem, a File Not Found error would occur when opening the Workbench.
Cloudera Bug: DSE-3835

Known Issues and Limitations in Cloudera Data Science Workbench 1.4.0

For a complete list of the current known issues and limitations in Cloudera Data Science Workbench 1.4.x, see [Known Issues and Limitations in Cloudera Data Science Workbench 1.7.2](#) on page 52.

Cloudera Data Science Workbench 1.3.1

This section lists the release notes for Cloudera Data Science Workbench 1.3.1.

New Features in Cloudera Data Science Workbench 1.3.1

- **Operating System:** Added support for RHEL / CentOS / Oracle Linux RHCK 7.5.
- **SAML**
 - Cloudera Data Science Workbench now supports multiple identity provider signing certificates for SAML 2.0 authentication.
 - Cloudera Data Science Workbench now supports SAML 2.0 Errata 05 E43 for SAML 2.0 authentication.

Issues Fixed in Cloudera Data Science Workbench 1.3.1

Remote Command Execution and Information Disclosure in Cloudera Data Science Workbench

A configuration issue in Kubernetes used by Cloudera Data Science Workbench can allow remote command execution and privilege escalation in CDSW. A separate information permissions issue can cause the LDAP bind password to be exposed to authenticated CDSW users when LDAP bind search is enabled.

Products affected: Cloudera Data Science Workbench

Releases affected: Cloudera Data Science Workbench 1.3.0 (and lower)

Users affected: All users of Cloudera Data Science Workbench 1.3.0 (and lower)

Date/time of detection: May 16, 2018

Severity (Low/Medium/High): High

Impact: Remote command execution and information disclosure

CVE: CVE-2018-11215

Immediate action required: [Upgrade](#) to the latest version of Cloudera Data Science Workbench (1.3.1 or higher) and [change the LDAP bind password](#) if previously configured in Cloudera Data Science Workbench.

Addressed in release/refresh/patch: Cloudera Data Science Workbench 1.3.1 (and higher)

For the latest update on this issue see the corresponding Knowledge Base article:

[TSB: 2018-313: Remote Command Execution and Information](#)

Cloudera Data Science Workbench Release Notes

Other Notable Fixed Issues in Cloudera Data Science Workbench 1.3.1

- Fixed an issue where CSD installations would fail to recognize Oracle Linux 7.3 as a supported operating system.
Cloudera Bug: DSE-3257
- Fixed several usability issues (file create, save, and so on) with Internet Explorer 11.
Cloudera Bug: DSE-3426, DSE-3434
- Fixed a SAML 2.0 configuration issue where uploading the identity provider metadata XML file did not update identity provider signing certificate and/or SSO URL on Cloudera Data Science Workbench correctly.
Cloudera Bug: DSE-3265
- Fixed an issue where the owner of a console output could not view their own shared consoles from sessions /job runs when sharing with **Specific user/team**.
Cloudera Bug: DSE-3143
- Fixed issue with missing connectors in Jobs dependency chart.
Cloudera Bug: DSE-3185

Known Issues and Limitations in Cloudera Data Science Workbench 1.3.1

For a list of the current known issues and limitations, refer the documentation for version 1.3.x at [Cloudera Data Science Workbench 1.3.x](#).

Cloudera Data Science Workbench 1.3.0

This section lists the release notes for Cloudera Data Science Workbench 1.3.0.

New Features and Changes in Cloudera Data Science Workbench 1.3.0

- Added support for SUSE Linux Enterprise Server 12 SP3.
- Site administrators can now add template projects that are customized for their organization's use-cases.
- Version 1.3 introduces a new environment variable for Python 3 sessions called `PYSPARK3_PYTHON`. Python 2 sessions will continue to use the default `PYSPARK_PYTHON` variable. This will allow you to configure distinct variables for Python 2 and Python 3 applications.
- In the Cloudera Manager CDSW service, the **Wildcard DNS Domain** property has been renamed to **Cloudera Data Science Workbench Domain**.
- Output for the `cdsw version` command now includes the type of deployment you are running – RPM or CSD.
- Added `log4j` and `spark-defaults` sample configuration to the PySpark and Scala template projects.

Issues Fixed in Cloudera Data Science Workbench 1.3.0

- Fixed an issue where the `cdsw status` command failed to run all the required system checks.
Cloudera Bug: DSE-3070
- Session lists now include additional metadata to help distinguish between different sessions.
Cloudera Bug: DSE-2814
- Pre-install validation checks have been improved to detect issues with `iptables` modules and Java settings.
Cloudera Bug: DSE-2293

- Fixed an issue with the `cdsw status` command output when TLS is enabled.
Cloudera Bug: DSE-3182
- CDS 2.2 Release 2 fixes the issue where a PySpark application could only be run *once* per active Workbench session.
Cloudera Bug: CDH-58475
- Fixed an issue that prevented Bokeh plots from rendering.
Cloudera Bug: DSE-3134
- Fixed an issue in Cloudera Data Science Workbench 1.2.2 that prevented WebSocket re-connections and caused console hangs.
Cloudera Bug: DSE-3085
- Improved CDSW service restart performance for CSD deployments.
Cloudera Bug: DSE-2937

Incompatible Changes in Cloudera Data Science Workbench 1.3.0

Deploying Cloudera Data Science Workbench with Cloudera Director 2.7

While this is not a Cloudera Data Science Workbench change, you should note that Cloudera Director 2.7 includes a new instance-level setting that sets the `mountAllUnmountedDisks` property to `false`:

```
normalizationConfig {  
  mountAllUnmountedDisks: false  
}
```

This means Cloudera Director 2.7 (and higher) users no longer need to set `lp.normalization.mountAllUnmountedDisksRequired` to `false` in the Cloudera Director server's `application.properties` file. Note that Cloudera Director 2.6 still requires this setting.

Known Issues and Limitations in Cloudera Data Science Workbench 1.3.0

For a list of the current known issues and limitations, refer the documentation for version 1.3.x at [Cloudera Data Science Workbench 1.3.x](#).

Cloudera Data Science Workbench 1.2.2

This section lists the release notes for Cloudera Data Science Workbench 1.2.2. The documentation for version 1.2.x can be found at [Cloudera Data Science Workbench 1.2.x](#).

New Features and Changes in Cloudera Data Science Workbench 1.2.2

- Added support for SUSE Linux Enterprise Server 12 SP2.
- Added support for multi-homed networks.
- Cloudera Director now allows you to deploy CSD-based Cloudera Data Science Workbench 1.2.x deployments on AWS. For more specifics on supported platforms, see [Cloudera Altus Director Support \(AWS and Azure Only\)](#) on page 71.
- Added a new environment variable called `MAX_TEXT_LENGTH` that allows you to set the maximum number of characters that can be displayed in a single text cell. By default, this value is set to 800,000 and any more characters will be truncated.

Engine Upgrade

Cloudera Data Science Workbench 1.2.2 (and later) ships **version 4** of the base engine image which includes bug fixes related to Python development and Kerberos authentication. Engine 4 ships the following versions of R and Python:

- **R** - 3.4.1
- **Python** - 2.7.11, 3.6.1

For details about the packages included in the base engine, see [Cloudera Data Science Workbench Engine Versions and Packaging](#) on page 224.

Make sure you upgrade existing projects to **Base Image v4 (Project Settings > Engine)** to take advantage of these fixes.

The new engine also changes how you configure and use Conda in Python sessions and extended engines. For more details, see [Using Conda with Cloudera Data Science Workbench](#) on page 219.

Issues Fixed In Cloudera Data Science Workbench 1.2.2

- Fixed an issue where Conda environmental variables were not being propagated to the Terminal correctly.
Cloudera Bug: DSE-2256
- Fixed an issue where GPUs were not being detected by Cloudera Data Science Workbench due to incorrect mount settings.
Cloudera Bug: DSE-2957
- Fixed an issue where jobs were failing due to Kerberos TGT renewal issues.
Cloudera Bug: DSE-1007
- Fixed an issue on Internet Explorer 10 and 11 where the browser would fail to render console output after launching too many interactive sessions.
Cloudera Bug: DSE-2998, DSE-2979
- Cloudera Data Science Workbench now correctly renders HTML that contains iFrames with the `srcdoc` attribute.
Cloudera Bug: DSE-2034
- Fixed an issue where logging in using LDAP/Active Directory would sometimes crash the Cloudera Data Science Workbench web application.
Cloudera Bug: DSE-2672
- The file tree in the Workbench now refreshes correctly when switching between sessions or launching a new session.
Cloudera Bug: DSE-2829
- Fixed a file descriptors leak that would cause the "Failed to get Kubernetes client configuration" error in Cloudera Manager.
Cloudera Bug: DSE-2910
- Fixed an issue where the host-controller process was consuming too much CPU. This was occurring due to a [bug](#) in the Kubernetes `client-go` library.
Cloudera Bug: DSE-2993

Known Issues and Limitations in Cloudera Data Science Workbench 1.2.2

For a list of known issues and limitations, refer the documentation for version 1.2.x at [Cloudera Data Science Workbench 1.2.x](#).

Cloudera Data Science Workbench 1.2.1

This section lists the release notes for Cloudera Data Science Workbench 1.2.1. The documentation for version 1.2.x can be found at [Cloudera Data Science Workbench 1.2.x](#).

Issues Fixed In Cloudera Data Science Workbench 1.2.1

- The **Master Node IPv4 Address** parameter has been added to Cloudera Manager's Add Service wizard and is now a required parameter for installation on AWS. This should fix any related installation issues for deployments on AWS.

Cloudera Bug: DSE-2879

- Fixed an issue with CSD-based deployments where certain operations would fail because the **Prepare Node** command was not installing all the required packages during First Run of the service. To see the updated list of packages that are now being installed by the **Prepare Node** command, refer the [CSD install guide](#).

Cloudera Bug: DSE-2869

- Fixed an issue where the `LD_LIBRARY_PATH` environmental variable was not getting propagated to CUDA engines.

Cloudera Bug: DSE-2828

- Fixed an issue where stopping Cloudera Data Science Workbench on worker hosts resulted in the application hanging indefinitely.

Cloudera Bug: DSE-2880

Incompatible Changes in Cloudera Data Science Workbench 1.2.1

Upgrading from Cloudera Data Science Workbench 1.2.0 to 1.2.1 on CSD-based deployments

After upgrading from Cloudera Data Science Workbench 1.2.0 to 1.2.1 on a CSD-based deployment, CLI commands might not work as expected due to missing binaries in the environment. Note that this issue does not affect fresh installs.

Known Issues and Limitations in Cloudera Data Science Workbench 1.2.1

For a list of known issues and limitations, refer the documentation for version 1.2.x at [Cloudera Data Science Workbench 1.2.x](#).

Cloudera Data Science Workbench 1.2.0

This section lists the release notes for Cloudera Data Science Workbench 1.2.0. The documentation for version 1.2.x can be found at [Cloudera Data Science Workbench 1.2.x](#).

New Features and Changes in Cloudera Data Science Workbench 1.2.0

- Cloudera Data Science Workbench is now available as an add-on service for Cloudera Manager. To this end, Cloudera Data Science Workbench is now distributed in a parcel that integrates with Cloudera Manager using a Custom Service Descriptor (CSD). You can use Cloudera Manager to [install, upgrade](#), and monitor Cloudera Data Science Workbench. [Diagnostic data bundles](#) can be generated and submitted to Cloudera through Cloudera Manager.
- Cloudera Data Science Workbench now enables secure sharing of job and session consoles. Additionally, site administrators can disable anonymous sharing from the Site Administrator dashboard (**Admin > Security**). See [Sharing Job and Session Console Outputs](#) on page 151.
- The **Admin > Usage** page now includes graphs for monitoring usage activity such as number of CPUs or GPUs used, memory usage, and total session runs, over customizable periods of time.

Cloudera Data Science Workbench Release Notes

- Cloudera Data Science Workbench now lets you configure session, job, and idle timeouts. These can be configured using [environmental variables](#) either for the entire deployment or per-project.
- The `cdsw enable` and `cdsw disable` commands are no longer needed. The master host will now automatically detect the IP addresses of worker hosts joining or leaving Cloudera Data Science Workbench. See the revised [Cloudera Data Science Workbench Command Line Reference](#) on page 312.
- The Kudu Python client is now included in the Cloudera Data Science Workbench base engine image.
- Interactive session names can now be modified by project contributors and admins. By default, session names are set to 'Untitled Session'.
- All-numeric usernames are now accepted.
- Kubernetes has been upgraded to version 1.6.11.

Engine Upgrade

- Cloudera Data Science Workbench 1.2.0 ships **version 3** of the base engine image which includes `matplotlib` improvements and the Kudu client libraries. Engine 3 ships the following versions of R and Python:
 - **R** - 3.4.1
 - **Python** - 2.7.11, 3.6.1

Make sure you upgrade existing projects to **Base Image v3 (Project Settings > Engine)** to take advantage of the new features and bug fixes included in the new engine.

Issues Fixed in Cloudera Data Science Workbench 1.2.0

Privilege Escalation and Database Exposure in Cloudera Data Science Workbench

Several web application vulnerabilities allowed malicious authenticated Cloudera Data Science Workbench (CDSW) users to escalate privileges in CDSW. In combination, such users could exploit these vulnerabilities to gain root access to CDSW hosts, gain access to the CDSW database which includes Kerberos keytabs of CDSW users and bcrypt hashed passwords, and obtain other privileged information such as session tokens, invitations tokens, and environmental variables.

Products affected: Cloudera Data Science Workbench

Releases affected: Cloudera Data Science Workbench 1.0.0, 1.0.1, 1.1.0, 1.1.1

Users affected: All users of Cloudera Data Science Workbench 1.0.0, 1.0.1, 1.1.0, 1.1.1

Date/time of detection: September 1, 2017

Detected by: NCC Group

Severity (Low/Medium/High): High

Impact: Privilege escalation and database exposure.

CVE: CVE-2017-15536

Addressed in release/refresh/patch: Cloudera Data Science Workbench 1.2.0 or higher.

Immediate action required: Upgrade to the latest version of Cloudera Data Science Workbench.

Other Notable Fixed Issues in Cloudera Data Science Workbench 1.2.0

- Fixed an issue where the Workbench editor screen jumps unexpectedly when typing or scrolling.
- Fixed auto-scroll behavior in the Workbench console. This was a browser compatibility issue that affected Chrome and Firefox, but not Safari.
- Fixed an issue where if a user logged out of Cloudera Data Science Workbench, and logged back in as a different user, they may see a `SecurityError` message in the Workbench.
- Fixed an issue that was preventing site administrators from uploading the SAML metadata file.

- Fixed several issues related to plotting with `matplotlib`. If you have previously used any workarounds for plotting, you might consider removing them now.
- Engines now use the same build of Kerberos utilities (`ktutil`, `kinit`, and `klist`) as the rest of Cloudera Data Science Workbench. This will improve logs obtained from `kinit` and make debugging Kerberos issues easier.
- `KRB5_TRACE` is now included in the error logs obtained when you `kinit`.
- Fixed an issue that was affecting health checks in deployments using AWS elastic load balancing.

Incompatible Changes in Cloudera Data Science Workbench 1.2.0

Proxy Configuration Change: If you are using a proxy server, you must ensure that the IP addresses for the web and Livelog services are skipped from the proxy.

Depending on your deployment (parcel or package), append the following IP addresses to either the **No Proxy** property in the Cloudera Manager CDSW service, or to the `NO_PROXY` parameter in `cdsw.conf`.

```
100.77.0.129
100.77.0.130
```

These have also been added to the installation instructions.

Known Issues and Limitations in Cloudera Data Science Workbench 1.2.0

For a list of known issues and limitations, refer the documentation for version 1.2.x at [Cloudera Data Science Workbench 1.2.x](#).

Cloudera Data Science Workbench 1.1.1

This section lists the release notes for Cloudera Data Science Workbench 1.1.1. The documentation for version 1.1.x can be found at [Cloudera Data Science Workbench 1.1.x](#).

New Features in Cloudera Data Science Workbench 1.1.1

- **Keytab Authentication** - With version 1.1.1, you can now authenticate yourself to the CDH cluster by uploading your Kerberos keytab to Cloudera Data Science Workbench. To use this feature, go to the top-right dropdown menu, click **Account settings** > **Hadoop Authentication**, enter your Kerberos principal and click **Upload Keytab**.

Issues Fixed In Cloudera Data Science Workbench 1.1.1

- Fixed an issue with airgapped installations where the installer could not pull the alpine 3.4 image into the airgapped environment.
- Fixed an issue where Cloudera Data Science Workbench would fail to log a command trace when the Kerberos process exits.
- Fixed authentication issues with older versions of MIT KDC.

Known Issues and Limitations in Cloudera Data Science Workbench 1.1.1

For a list of known issues and limitations, refer the documentation for version 1.1.x at [Cloudera Data Science Workbench 1.1.x](#).

Cloudera Data Science Workbench 1.1.0

This section lists the release notes for Cloudera Data Science Workbench 1.1.0. The documentation for version 1.1.x can be found at [Cloudera Data Science Workbench 1.1.x](#).

New Features and Changes in Cloudera Data Science Workbench 1.1.0

- Added support for RHEL/CentOS 7.3 and Oracle Linux 7.3.

Cloudera Data Science Workbench Release Notes

- Cloudera Data Science Workbench now allows you to run GPU-based workloads. For more details, see [Using NVIDIA GPUs for Cloudera Data Science Workbench Projects](#) on page 137.
- For Cloudera Manager and CDH clusters that are not connected to the Internet, Cloudera Data Science Workbench now supports fully offline installations. See the [installation guide](#) for more details.
- Web UIs for processing frameworks such as Spark 2, Tensorflow, and Shiny, are now embedded in Cloudera Data Science Workbench and can be accessed directly from active sessions and jobs. For more details, see [Accessing Web User Interfaces from Cloudera Data Science Workbench](#) on page 145.
- Added support for a Jobs REST API that lets you orchestrate jobs from 3rd party workflow tools. See [Cloudera Data Science Workbench Jobs API](#) on page 200.
- DataFrames are now scrollable in the workbench session output pane. For examples, see the section on [Grid Displays](#) on page 136.
- Added support for rich visualizations in Scala engine using Jupyter [jvm-repr](#). For an example, see [HTML Visualizations - Scala](#).
- `JAVA_HOME` is now set in `cdsw.conf`, and not from the Site Administrator dashboard (**Admin > Engines**).

Engine Upgrade

Cloudera Data Science Workbench 1.1.0 ships **version 2** of the base engine image that includes new versions of Pandas, seaborn, and assorted bug fixes. Engine 2 ships the following versions of R and Python:

- **R** - 3.3.0
- **Python** - 2.7.11, 3.6.1

Make sure you upgrade existing projects to **Base Image v2 (Project Settings > Engine)** to take advantage of the new features and bug fixes included in the new engine.

Issues Fixed in Cloudera Data Science Workbench 1.1.0

- Improved support for dynamic data visualizations in Python, including Bokeh.
- Fixed issues with the Python template project. The project now supports offline mode and will therefore work on airgapped clusters.
- Fixed issues related to cached responses in Internet Explorer 11.
- Fixed issues with Java symlinks outside of `JAVA_HOME`.
- The `cdsw status` command can now be run on worker hosts.
- Removed unauthenticated localhost access to Kubernetes.
- Fixed Kerberos authentication issues with specific enc-types and Active Directory.
- Removed restrictions on usernames with special characters for better compatibility with external authentication systems such as Active Directory.
- Fixed issues with LDAP configuration validation that caused application crashes.
- Improved LDAP test configuration form to avoid confusion on parameters being sent.

Incompatible Changes in Cloudera Data Science Workbench 1.1.0

- **Upgrading from version 1.0.x to 1.1.x**

During the upgrade process, you will encounter incompatibilities between the two versions of `cdsw.conf`. This is because even though you are installing the latest RPM, your previous configuration settings in `cdsw.conf` will

remain unchanged. Depending on the release you are upgrading from, you will need to modify `cdsw.conf` to ensure it passes the validation checks run by the 1.1.x release.

Key changes to note:

- `JAVA_HOME` is now a required parameter. Make sure you add `JAVA_HOME` to `cdsw.conf` before you start Cloudera Data Science Workbench.
- Previous versions allowed `MASTER_IP` to be set to a DNS hostname. If you are still using a DNS hostname, switch to an IP address.
- **Python engine updated in version 1.1.x**

Version 1.1.x includes an updated base engine image for Python which no longer uses the deprecated `pylab` mode in Jupyter to import the `numpy` and `matplotlib` functions into the global scope. With version 1.1.x, engines will now use built-in functions like `any` rather than the `pylab` counterpart, `numpy.any`. As a result of this change, you might see certain behavioral changes and differences in results between the two versions.

Also note that Python projects originally created with engine 1 will be running pandas version 0.19, and will not auto-upgrade to version 0.20 by simply selecting engine 2. You will also need to manually install version 0.20.1 of pandas when you launch a project session.

Known Issues and Limitations in Cloudera Data Science Workbench 1.1.0

For a list of known issues and limitations, refer the documentation for version 1.1.x at [Cloudera Data Science Workbench 1.1.x](#).

Cloudera Data Science Workbench 1.0.1

This section lists the release notes for Cloudera Data Science Workbench 1.0.1. The documentation for version 1.0.x can be found at [Cloudera Data Science Workbench 1.0.x](#).

Issues Fixed in Cloudera Data Science Workbench 1.0.1

- Fixed a random port conflict that could prevent Scala engines from running.
- Improved formatting of validation, and visibility of some errors.
- Fixed an issue with Firefox that was resulting in duplicate jobs on job creation.
- Removed the Mathjax external dependency on CDN.
- Improved `PATH` and `JAVA_HOME` handling that previously broke Hadoop CLIs.
- Fixed an issue with Java security policy files that caused Kerberos issues.
- Fixed an issue that caused `git clone` to fail on some repositories.
- Fixed an issue where updating LDAP admin settings deactivated the local fallback login.
- Fixed an issue where bad LDAP configuration crashed the application.
- Fixed an issue where job environmental variable settings did not persist.

Known Issues and Limitations in Cloudera Data Science Workbench 1.0.x

For a list of known issues and limitations, refer the documentation for version 1.0.x at [Cloudera Data Science Workbench 1.0.x](#).

Cloudera Data Science Workbench 1.0.0

Version 1.0 represents the first generally available (GA) release of Cloudera Data Science Workbench. For information about the main features and benefits of Cloudera Data Science Workbench, as well as an architectural overview of the product, see [Cloudera Data Science Workbench Overview](#) on page 15.

Known Issues and Limitations in Cloudera Data Science Workbench 1.7.2

This topic lists the current known issues and limitations in Cloudera Data Science Workbench 1.7.2. For previous versions, see:

- [Known Issues in Cloudera Data Science Workbench 1.6.x](#)
- [Known Issues in Cloudera Data Science Workbench 1.5.x](#)
- [Known Issues in Cloudera Data Science Workbench 1.4.x](#)
- [Known Issues in Cloudera Data Science Workbench 1.3.x](#)
- [Known Issues in Cloudera Data Science Workbench 1.2.x](#)
- [Known Issues in Cloudera Data Science Workbench 1.1.x](#)

Installation

During the Cloudera Data Science Workbench startup process, you might see certain timeout issues.

```
Pods not ready in cluster default ['role/<pod_name>'].
```

This is due to an issue with some pods taking longer to start up and other dependent processes timing out. Restart the CDSW service to get past this issue.

Cloudera Bug: DSE-6855

Upgrades

Please read the following upgrade issues before you begin the upgrade process:

Upgrades supported from CDSW 1.5.x (and higher) to CDSW 1.7.x

Cloudera Data Science Workbench only supports upgrades to version 1.7.x from version 1.5.x and 1.6.x. If you are using an earlier version, you must first upgrade to version 1.5.x or 1.6.x, and then upgrade to version 1.7.x.

[CDSW restart issue on multi-node deployments; CDSW Web UI does not automatically come up after upgrading to CDSW 1.7.1](#)

After upgrading multi-node deployments (1 CDSW Master, multiple Workers) to CDSW 1.7.1, the web application is not automatically accessible as expected. This happens because of a bug where the CDSW restart process does not open the HTTP/HTTPS port required by the web pod.

Affected Version: Cloudera Data Science Workbench 1.7.1

Fixed Version: Cloudera Data Science Workbench 1.7.2

Workaround: This is a one-time fix needed to solve the issue with the CDSW restart process.

1. Download the following patch files:

- [ingress-controller.yaml](#)
- [tcp-ingress-controller.yaml](#)

2. Copy the `ingress-controller.yaml` file to

`/etc/cdsw/patches/default/deployment/ingress-controller.yaml` on the CDSW master node.

3. Copy the `tcp-ingress-controller.yaml` file to `/etc/cdsw/patches/default/deployment/tcp-ingress-controller.yaml` on the CDSW master node.
4. [Restart Cloudera Data Science Workbench.](#)

Cloudera Bug: DSE-9587, DSE-9663

Domain name resolution issues after upgrading to CDSW 1.7.x; Pods stuck in CrashLoopBackOff state

After upgrading to CDSW 1.7.x, certain application pods (s2i-registry and image-puller) get stuck in CrashLoopBackOff state. This is due to an issue with the DNS resolver.

Workaround: Remove or comment out the `search` entry from the `/etc/resolv.conf` file.

```
# cat /etc/resolv.conf
.....
# search example.com
nameserver 192.0.2.1
nameserver 192.0.2.2
```

CDSW shows a warning message to update to a lower Base Image version after upgrading to CDSW 1.7.x

You may see the following warning message after upgrading to CDSW 1.7.x, asking you to update the Base Image version: There is a new version of Base Image available. Latest engine image is: "Base Image v9".

You can ignore this message because CDSW 1.7.x comes with the Base Image v10. However, if you choose to update to v9 and click **Update version**, then your host system would try to download the Base Image from the online docker repository: `docker.repository.cloudera.com/cdsw/engine:9`. And depending on the amount of time the host takes to pull the v9 image, your session may get stuck in a "Scheduling" state.

CDSW does not display this message when you newly install CDSW 1.7.x.

Cloudera Bug: DSE-10170

On a TLS-enabled cluster Cloudera Manager points the Cloudera Data Science Workbench web UI to `http://` instead of `https://`

After upgrading the Cloudera Data Science Workbench parcel and CSD to 1.7.x, the link to the Cloudera Data Science Workbench web UI from Cloudera Manager redirects to `http://cdsw.your-company.com` instead of `https://cdsw.your-company.com` on a TLS-enabled cluster.

Workaround: You can manually enter the complete domain name with the `https` protocol in your web browser. Alternatively, contact Cloudera Support to obtain a hotfix and the instructions to apply the patch. Quote the following issue while raising the support request: ENGESC-199.

Cloudera Bug: ENGESC-199

CDH Integration

CDH client configuration changes require a full Cloudera Data Science Workbench restart

Cloudera Data Science Workbench does not automatically detect configuration changes on the CDH cluster. Therefore, any changes made to CDH services, ranging from updates to service configuration properties to complete CDH or CDS parcel upgrades, must be followed by a full reset of Cloudera Data Science Workbench.

Workaround: Depending on your deployment, use one of the following sets of steps to perform a full reset of Cloudera Data Science Workbench. Note that this reset does not impact your data in any way.

- **CSD Deployments** - To reset Cloudera Data Science Workbench using Cloudera Manager:
 1. Log into the Cloudera Manager Admin Console.

2. On the Cloudera Manager homepage, click



to the right of the **CDSW** service and select **Restart**. Confirm your choice on the next screen and wait for the action to complete.

OR

- **RPM Deployments** - Run the following steps on the Cloudera Data Science Workbench master host:

```
cdsw stop
cdsw start
```

Cloudera Manager Integration

CSD distribution/activation fails on mixed-OS clusters when there are third-party parcels running on OSs that are not supported by Cloudera Data Science Workbench

For example, adding a new CDSW gateway host on a RHEL 6 cluster running RHEL-6 compatible parcels will fail. This is because Cloudera Manager will not allow distribution of the RHEL 6 parcels on the new host which will likely be running a CDSW-compatible operating system such as RHEL 7.

Workaround: To ensure adding a new CDSW gateway host is successful, you must create a copy of the 'incompatible' third-party parcel files and give them the corresponding RHEL 7 names so that Cloudera Manager allows them to be distributed on the new gateway host. Use the following sample instructions to do so:

1. SSH to the Cloudera Manager Server host.
2. Navigate to the directory that contains all the parcels. By default, this is `/opt/cloudera/parcel-repo`.

```
cd /opt/cloudera/parcel-repo
```

3. Make a copy of the incompatible third-party parcel with the new name. For example, if you have a RHEL 6 parcel that cannot be distributed on a RHEL 7 CDSW host:

```
cp <PARCELNAME.cdh5.x.x.p0.123>-el6.parcel <PARCELNAME.cdh5.x.x.p0.123>-el7.parcel
```

4. Repeat the previous step for parcel's SHA file.

```
cp <PARCELNAME.cdh5.x.x.p0.123>-el6.parcel.sha <PARCELNAME.cdh5.x.x.p0.123>-el7.parcel.sha
```

5. Update the new files' owner and permissions to match those of existing parcels in the `/opt/cloudera/parcel-repo` directory.

```
chown cloudera-scm:cloudera-scm <PARCELNAME.cdh5.x.x.p0.123>-el7.parcel
chown cloudera-scm:cloudera-scm <PARCELNAME.cdh5.x.x.p0.123>-el7.parcel.sha
chmod 640 <PARCELNAME.cdh5.x.x.p0.123>-el7.parcel
chmod 640 <PARCELNAME.cdh5.x.x.p0.123>-el7.parcel.sha
```

You should now be able to add new gateway hosts for Cloudera Data Science Workbench to your cluster.

Cloudera Bug: OPSAPS-42130, OPSAPS-31880

CDSW Service health status after a restart does not match the actual state of the application

After a restart, the Cloudera Data Science Workbench service in Cloudera Manager will display **Good** health even though the Cloudera Data Science Workbench web application might need a few more minutes to get ready to serve requests.

CDS Powered By Apache Spark

Scala sessions can fail if dependencies take longer than 15 minutes

If the dependencies in `spark-defaults.conf` (`spark.jars`, `spark.packages`, etc) take longer than 15 minutes to resolve, then scala sessions will fail the first time.

Workaround: Use one of the following workarounds:

- Restart the session.
- Mount the Spark dependency directory from the CDSW host machines.

CDSW specific `log4j.properties` file is picked up on the executors

SPARK-11105 implements automatic upload/pickup of the `log4j.properties` file on the executors. This causes issues as CDSW adds an extra appender to store the Spark driver logs in the container under `/tmp`.

```
log4j.appender.file=org.apache.log4j.FileAppender
log4j.appender.file.append=true
log4j.appender.file.file=/tmp/spark-driver.log
..
```

When this configuration is used a `/tmp/spark-driver.log` file is created on the Executor's host with the kinitd user which can result in permission errors when multiple users run Spark applications in CDSW.

Workaround: Append the following lines to the `spark-defaults.conf` file:

C6.xx versions

```
spark.yarn.am.extraJavaOptions=-Dlog4j.configuration=file:///etc/spark/conf/log4j.properties
spark.executor.extraJavaOptions=-Dlog4j.configuration=file:///etc/spark/conf/log4j.properties
```

c5.xx versions

```
spark.yarn.am.extraJavaOptions=-Dlog4j.configuration=file:///etc/spark2/conf/log4j.properties
spark.executor.extraJavaOptions=-Dlog4j.configuration=file:///etc/spark2/conf/log4j.properties
```

Cloudera Bug: DSE-5900

On TLS-enabled CDSW deployments, the embedded Spark UI does not work

If you have a TLS-enabled CDSW deployment, the embedded Spark UI tab does not render as expected.

Workaround: To work around this issue, launch the Spark UI in a separate tab and append `/jobs` after the URL. For example, if your `engineID` is `tb0z9ydiua5q9v2d` and the DOMAIN is `example.com` then view the Spark UI at:
`https://spark-tb0z9ydiua5q9v2d.example.com/jobs/`

Alternative workaround: To view running Spark jobs, navigate to **Spark History Server UI > Show Incomplete Applications > Application ID**

Affected Versions: This issue affects **CDSW 1.6.x** and **CDSW 1.7.x** on the following platforms:

- **CDH 5:** CDS 2.4 release 2 (and lower)
- **CDH 6:** Versions of Spark that ship with CDH 6.0.x, CDH 6.1.x, CDH 6.2.1 (and lower), CDH 6.3.2 (and lower)

Solution: Upgrade to CDSW version 1.7.1 or higher, and either:

- CDH version 6.4.0, 6.2.2, 6.3.3 or higher
- CDH 5 with Spark 2.4 release 3

Spark lineage collection is not supported with Cloudera Data Science Workbench

Lineage collection is enabled by default in Spark 2.3. This feature does not work with Cloudera Data Science Workbench because the lineage log directory is not automatically mounted into CDSW engines when a session/job is started.

Affected Versions: CDS 2.3 release 2 (and higher) Powered By Apache Spark

With Spark 2.3 release 3 (or higher), if Spark cannot find the lineage log directory, it will automatically disable lineage collection for that application. Spark jobs will continue to execute in Cloudera Data Science Workbench, but lineage information will not be collected.

With Spark 2.3 release 2, Spark jobs will fail in Cloudera Data Science Workbench. Either upgrade to Spark 2.3 release 3 which includes a partial fix (as described above) or use one of the following workarounds to disable Spark lineage:

Workaround 1: Disable Spark Lineage Per-Project in Cloudera Data Science Workbench

To do this, set `spark.lineage.enabled` to `false` in a `spark-defaults.conf` file in your Cloudera Data Science Workbench project. This will need to be done individually for each project as required.

Workaround 2: Disable Spark Lineage for the Cluster

1. Log in to Cloudera Manager and go to the Spark 2 service.
2. Click **Configuration**.
3. Search for the **Enable Lineage Collection** property and uncheck the checkbox to disable lineage collection.
4. Click **Save Changes**.
5. Go back to the Cloudera Manager homepage and [restart the CDSW service](#) for this change to go into effect.

Cloudera Bug: DSE-3720, CDH-67643

Monitoring Spark Applications invoked from CDSW

To monitor `spark_on_yarn` applications invoked from CDSW, an embedded Spark UI is displayed right next to the session/job. This was achieved by disabling RM proxy. However, with this change, attempts to access the same Spark application using the RM UI will result in Error 500 (connection refused).

Affected Versions: CDSW 1.6 and higher.

Workaround: If the Administrator wants to troubleshoot a running `spark-on-yarn` application invoked by an end-user from the workbench, the user must share their session using the **Share** button on the right side of the console. An alternate workaround which will not provide realtime updates is to access the Spark Application UI from the Spark History Server UI > Incomplete Applications.

Cloudera Bug: DSE-4979

Crashes and Hangs

- **Third-party security and orchestration software (such as McAfee, Tanium, Symantec) can lead to CDSW crashing randomly**

Workaround: Disable all third-party security agents on CDSW hosts.

Cloudera Bug: DSE-8550

- High I/O utilization on the application block device can cause the application to stall or become unresponsive. Users should read and write data directly from HDFS rather than staging it in their project directories.
- Installing ipywidgets or a Jupyter notebook into a project can cause Python engines to hang due to an unexpected configuration. The issue can be resolved by deleting the installed libraries from the R engine terminal.

Third-party Editors

- Logs generated by a browser IDE do not appear within the IDE. They are displayed in the **Logs** tab for the session.

Cloudera Bug: DSE-6570

- Sessions with Browser IDEs running do not adhere to the limit set in `IDLE_MAXIMUM_MINUTES`. Session logs show the warning message that states that the idle session will timeout, but the timeout does not occur. The session continues to run and consume resources until the timeout set in `SESSION_MAXIMUM_MINUTES` is reached. Ensure that you manually stop a session after you are finished, so that the resources are available to other users.

Cloudera Bug: DSE-6651

- Sessions with Browser IDEs running time out with no warning after the time limit set in `SESSION_MAXIMUM_MINUTES` is reached, regardless of whether or not the session is idle. Periodically stop the browser IDE and session manually to avoid reaching `SESSION_MAXIMUM_MINUTES`.

Cloudera Bug: DSE-6652

- The lack of a ROOT CA certificate can cause issues with terminals and the Jupyter editor after upgrading CDSW.

Problem: After upgrading from CDSW version 1.5 to version 1.7.1, the terminal does not open for any kernel, and the Jupyter notebook does not work.

Workaround: In CDSW, go to **Admin > Security**, and paste the internal CA root certificate file contents directly into the **Root CA** configuration field. You should be able to launch a new session and start the terminal or launch the Jupyter editor. It is not necessary to restart CDSW. This procedure is described at [Configuring Custom Root CA Certificate](#) on page 289

Engines

- Configuring duplicate mount points in the site admin panel (**Admin > Engines > Mounts**) results in sessions crashing in the workbench.

Cloudera Bug: DSE-3308

- Spawning remote workers fails in R when the `env` parameter is not set. For more details, see [Distributed Computing with Workers](#) on page 146.

Cloudera Bug: DSE-3384

- Autofs mounts are not supported with Cloudera Data Science Workbench.

Cloudera Bug: DSE-2238

- When using Conda to install Python packages, you must specify the Python version to match the Python versions shipped in the engine image (2.7.11 and 3.6.1). If not specified, the conda-installed Python version will not be used within a project. Pip (pip and pip3) does not face this issue.
- When engine version 8 (or higher) is used, and the [Allow containers to run as root property](#) is disabled, the creation of containers that run with root privileges is prevented. Additionally, the elevation of privileges from the `cdsw` user to `root` (for example, using a `setuid` binary) is also prevented.

As a result, running the `ping` command, which is actually a `setuid` binary, will fail in engine 8 (or higher) when [Allow containers to run as root property](#) is disabled.

```
$ ping www.google.com
Ping: icmp open socket: Operation not permitted.
```

Custom Engine Images

- Cloudera Data Science Workbench only supports customized engines that are based on the Cloudera Data Science Workbench base image.
- Cloudera Data Science Workbench does not support creation of custom engines larger than 10 GB.

Cloudera Bug: DSE-4420

- Cloudera Data Science Workbench does not support pulling images from registries that require Docker credentials.

Cloudera Bug: DSE-1521

- The contents of certain pre-existing standard directories such as `/home/cdsw`, `/tmp`, `/opt/cloudera`, and so on, cannot be modified while creating customized engines. This means any files saved in these directories will not be accessible from sessions that are running on customized engines.

Workaround: Create a new custom directory in the Dockerfile used to create the customized engine, and save your files to that directory. *Or*, create a new custom directory on all the Cloudera Data Science Workbench gateway hosts and save your files to those directories. Then, [mount this directory](#) to the custom engine.

Experiments

- **(If [quotas](#) are enabled)** Experiments that are stuck in the **Scheduled** state due to lack of resources do not automatically start even if you free up existing resources.

Workaround: Stop the experiment that is stuck in the **Scheduled** state. Then manually reschedule the experiment.

Cloudera Bug: DSE-8736

- Experiments do not store snapshots of project files. You cannot automatically restore code that was run as part of an experiment.
- Experiments will fail if your project filesystem is too large for the Git [snapshot](#) process. As a general rule, any project files (code, generated model artifacts, dependencies, etc.) larger than 50 MB must be part of your project's `.gitignore` file so that they are not included in snapshots for experiment builds.
- Experiments cannot be deleted. As a result, be conscious of how you use the `track_metrics` and `track_file` functions.
 - Do not track files larger than 50MB.
 - Do not track more than 100 metrics per experiment. Excessive metric calls from an experiment may cause Cloudera Data Science Workbench to hang.
- The Experiments table will allow you to display only three metrics at a time. You can select which metrics are displayed from the **metrics** dropdown. If you are tracking a large number of metrics (100 or more), you might notice some performance lag in the UI.
- Arguments are not supported with Scala experiments.
- The `track_metrics` and `track_file` functions are not supported with Scala experiments.
- The UI does not display a confirmation when you start an experiment or any alerts when experiments fail.

GPU Support

Only CUDA-enabled NVIDIA GPU hardware is supported

Cloudera Data Science Workbench only supports CUDA-enabled NVIDIA GPU cards.

Heterogeneous GPU hardware is not supported

You must use the same GPU hardware across a single Cloudera Data Science Workbench deployment.

GPU image for CDSW does not work with TensorFlow

The `LD_LIBRARY_PATH` environment variable is not set properly in the technical preview GPU image (docker.repository.cloudera.com/cdsw/cuda-engine:10) which is needed for the TensorFlow framework to work.



Note: If you are using Pytorch, then you may not face this issue.

Workaround: To use the technical preview GPU image (`docker.repository.cloudera.com/cdsw/cuda-engine:10`) with TensorFlow:

1. Install TensorFlow by running the following command:

```
pip3 install tensorflow
```

2. Add the following to the `LD_LIBRARY_PATH` environment variable:

```
LD_LIBRARY_PATH=/usr/local/cuda/lib64:/usr/local/cuda/lib:/usr/lib/hadoop/lib/native
```

Jobs

- Job notification emails fail intermittently when attachments are included. Emails are delivered either with blank attachments or no attachments at all.
Cloudera Bug: DSE-9469, DSE-8806
- Cloudera Data Science Workbench does not support changing your API key, or having multiple API keys.
- Currently, you cannot use the Jobs API to create a job, stop a job, or get the status of a job.
- Jobs pipeline visualization in CML and CDSW no longer shows dependent jobs. Dependencies only show the first step in the chain. Previously (before upgrade to 1.7), the UI displayed the whole chain of jobs. Jobs still run in the correct order, but the UI is no longer clear. Associated Bug: DSE-8003

Models

- **Known Issues with Model Builds and Deployed Models**

- Unable to create a model with the name of a deleted Model.

Workaround: For now, Models shall have unique names across the lifespan of the cluster installation.

Cloudera Bug: DSE-4237

- Re-deploying or re-building models results in model downtime (usually brief).
- Re-starting Cloudera Data Science Workbench does not automatically restart active models. These models must be manually restarted so they can serve requests again.
Cloudera Bug: DSE-4950
- Model deployment will fail if your project filesystem is too large for the Git [snapshot](#) process. As a general rule, any project files (code, generated model artifacts, dependencies, etc.) larger than 50 MB must be part of your project's `.gitignore` file so that they are not included in snapshots for model builds.
- Model builds will fail if your project filesystem includes a `.git` directory (likely hidden or nested). Typical build stage errors include:

```
Error: 2 UNKNOWN: Unable to schedule build: [Unable to create a checkpoint of current source: [Unable to push sources to git server: ...
```

To work around this, rename the `.git` directory (for example, `NO.git`) and re-build the model.

Cloudera Bug: DSE-4657

- JSON requests made to active models should not be more than 5 MB in size. This is because JSON is not suitable for very large requests and has high overhead for binary objects such as images or video. Call the model with a reference to the image or video, such as a URL, instead of the object itself.

- Any external connections, for example, a database connection or a Spark context, must be managed by the model's code. Models that require such connections are responsible for their own setup, teardown, and refresh.
- Model logs and statistics are only preserved so long as the individual replica is active. Cloudera Data Science Workbench may restart a replica at any time it is deemed necessary (such as bad input to the model).

• Limitations

- Scala models are not supported.
- [Spawning worker threads](#) is not supported with models.
- Models deployed using Cloudera Data Science Workbench are not highly-available.
- Dynamic scaling and auto-scaling are not currently supported. To change the number of replicas in service, you will have to re-deploy the build.

Networking

• CDSW cannot launch sessions due to connection errors resulting from a segfault

Sample error:

```
transport: Error while dialing dial tcp 100.77.93.252:20051: connect: connection refused
```

Workaround: Enable IPv6 on all CDSW hosts

1. Double-check that IPv6 is currently disabled during boot time, i.e. `ipv6.disable` should be equal to 1.

```
$ dmesg
[ 0.000000] Kernel command line: BOOT_IMAGE=/boot/vmlinuz-3.10.0-514.el7.x86_64
root=UUID=3e109aa3-f171-4614-ad07-c856f20f9d25 ro console=tty0 crashkernel=auto
console=ttyS0,115200 ipv6.disable=1
```

```
$ cat /proc/cmdline
....ipv6.disable=1
```



Note: If IPv6 is not explicitly disabled at the boot time, you don't have to follow the subsequent steps. The following steps are only required when "ipv6.disable" is set as 1 in the grub conf file.

2. Edit `/etc/default/grub` and delete the `ipv6.disable=1` entry from `GRUB_CMDLINE_LINUX`. For example:

```
GRUB_CMDLINE_LINUX="rd.lvm.lv=rhel/swap crashkernel=auto rd.lvm.lv=rhel/root"
```

3. Run the `grub2-mkconfig` command to regenerate the `grub.cfg` file:

```
grub2-mkconfig -o /boot/grub2/grub.cfg
```

Alternatively, on UEFI systems, you would run the following command:

```
grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
```

4. Follow the above steps for both CDSW Master and Worker nodes.
5. [Stop the Cloudera Data Science Workbench service](#).
6. Reboot all the Cloudera Data Science Workbench hosts to enable IPv6 support.
7. [Start the Cloudera Data Science Workbench service](#). Run `dmesg` on the CDSW hosts to ensure there are no segfault errors seen.



Note: In RHEL7, IPv6 support is built-in and no longer provided as a standalone kernel module (like it used to in RHEL6). Using the steps above we are only enabling the functionality and there is no need to enable IPv6 on the interfaces or via `sysctl`. The reason IPv6 functionality is required is due to the use of upgraded node.js version in CDSW 1.6 and above wherein the gRPC initialization code assumes that the IPv6 functionality is present even if IPv6 is not actually used for communication.

Cloudera Bug: DSE-7238, DSE-7455

- Custom `/etc/hosts` entries on Cloudera Data Science Workbench hosts do not propagate to sessions and jobs running in containers.

Cloudera Bug: DSE-2598

- Initialisation of Cloudera Data Science Workbench (`cdsw init`) will fail if localhost does not resolve to `127.0.0.1`.
- Cloudera Data Science Workbench does not support DNS servers running on `127.0.0.1:53`. This IP address resolves to the container localhost within Cloudera Data Science Workbench containers. As a workaround, use either a non-loopback address or a remote DNS server.
- Kubernetes throws the following error when `/etc/resolv.conf` lists more than three domains:

```
Resolv.conf file '/etc/resolv.conf' contains search line consisting of more than 3 domains!
```

Due to a limitation in the `libc` resolver, only two DNS servers are supported in `/etc/resolv.conf`. Kubernetes uses one additional entry for the cluster DNS.

Quotas

- If custom quota for a user is enabled, and the quotas feature is then disabled, the custom quota setting continues to remain in effect. That is, even if the quotas feature is disabled, the user will still see the `cpu/memory limit reached` error when they reach the previously set custom quota limit.

Workaround: If you want to disable quotas, first manually delete each custom quota row and then switch Quotas toggle to **OFF**. To remove custom quotas, click the vertical ellipses at the end of each custom quota row and choose **Remove**.

Cloudera Bug: DSE-9063

- (If [quotas](#) are enabled) Experiments that are stuck in the **Scheduled** state due to lack of resources do not automatically start even if you free up existing resources.

Workaround: Stop the experiment that is stuck in the **Scheduled** state. Then manually reschedule the experiment.

Cloudera Bug: DSE-8736

Security

Working in the terminal or an editor should not count as idle session

If a user opens a workbench and is either working exclusively in the terminal or just editing files, Cloudera Data Science Workbench counts that time as idle time and the user gets kicked out after the configured max idle timeout.

Workaround:

- Increase the idle session timeout by adding a new environmental variable `IDLE_MAXIMUM_MINUTES`. Click **CDSW > Project > Settings > Environmental variables**.



Note: This approach would require you to keep your containers running.

You can set the value of the variables `IDLE_MAXIMUM_MINUTES` or `SESSION_MAXIMUM_MINUTES` to their maximum allowed value, which is 35000 (~3 weeks).

- Alternatively, run a simple script inside CDSW session to keep the session alive. Opening the Cloudera Data Science Workbench and create a file as shown here (assuming Python project), and then run it in the Workbench.

```
import time
time.sleep(10000)
```

Cloudera Bug: DSE-3080

SSH access to Cloudera Data Science Workbench hosts must be disabled

The container runtime and application data storage is not fully secure from untrusted users who have SSH access to the gateway hosts. Therefore, SSH access to the gateway hosts for untrusted users should be disabled for security and resource utilization reasons.

TLS/SSL

- Self-signed certificates where the Certificate Authority is not part of the user's trust store are not supported for TLS termination. For more details, see [Enabling TLS/SSL - Limitations](#).
- Cloudera Data Science Workbench does not support the use of encrypted private keys for TLS.

Cloudera Bug: DSE-1708

- A "certificate has expired" error displays when you log in to the Cloudera Data Science Workbench web UI. This issue can occur if Cloudera Data Science Workbench exceeds 365 days of continuous uptime because the internal certificate for Kubernetes expires after 1 year.

Workaround: Restart the Cloudera Data Science Workbench deployment.

- For CSD installations, restart the Cloudera Data Science Workbench service in Cloudera Manager.
- For RPM installations, run the following command on the Master host:

```
cdsw restart
```

Kerberos

- Using Kerberos plugin modules in `krb5.conf` is not supported.
- Modifying the `default_ccache_name` parameter in `krb5.conf` does not work in Cloudera Data Science Workbench. Only the default path for this parameter, `/tmp/krb5cc_${uid}`, is supported.
- PowerBroker-equipped Active Directory is not supported.

Cloudera Bug: DSE-1838

- When you upload a Kerberos keytab to authenticate yourself to the CDH cluster, Cloudera Data Science Workbench might display a fleeting error message ('cancelled') in the bottom right corner of the screen, even if authentication was successful. This error message can be ignored.

Cloudera Bug: DSE-2344

Usability

- Environment variables with the dollar (\$) character are not parsed correctly by CDSW. For example, if you set `PASSWORD="pass$123"` in the project environment variables, and then try to read it using the `echo` command, you see the following output: `pass23`

Workaround: Use one of the following commands to print the \$ sign:

```
echo 24 | xxd -r -p
or
echo JAo= | base64 -d
```

Insert the value of the environment variable by wrapping it in the command substitution using `$()` or `` ``. For example, if you want to set the environment variable to `ABC$123`, specify:

```
ABC$(echo 24 | xxd -r -p)123
or
ABC`echo 24 | xxd -r -p`123
```

- You see the HTTP 404 error on navigating to other tabs within the CDSW web UI after updating the project name on the **Project Settings** page. This is because the backend API still uses the older project name.

Workaround: After you save the new project name, manually reload the webpage before navigating to the other tabs or pages within the CDSW web UI.

Cloudera Bug: DSE-11911

- In some cases, the application switcher (grid icon) does not show any other applications, such as Hue or Ranger.

Cloudera Bug: DSE-865

- Scala sessions hang when running large scripts (longer than 100 lines) in the Workbench editor.

Workaround 1:

Execute the script in manually-selected chunks. For example, highlight the first 50 lines and select **Run > Run Line(s)**.

Workaround 2:

Restructure your code by moving content into imported functions so as to bring the size down to under 100 lines.

- The R engine is unable to display multi-byte characters in plots. Examples of multi-byte characters include languages such as Korean, Japanese, and Chinese.

Workaround: Use the [showtext](#) R package to support more fonts and characters. For example, to display Korean characters:

```
install.packages('showtext')
library(showtext)
font_add_google("Noto Sans KR", "noto")
showtext_auto()
```

Cloudera Bug: DSE-7308

- In a scenario where 100s of users are logged in and creating processes, the `nproc` and `nofile` limits of the system may be reached. Use `ulimits` or other methods to increase the maximum number of processes and open files that can be created by a user on the system.
- When rebooting, Cloudera Data Science Workbench hosts can take a significant amount of time (about 30 minutes) to become ready.
- Long-running operations such as `fork` and `clone` can time out when projects are large or connections outlast the HTTP timeouts of reverse proxies.
- The Scala kernel does not support auto-complete features in the editor.

Cloudera Data Science Workbench Release Notes

- Scala and R code can sometimes indent incorrectly in the workbench editor.

Cloudera Bug: DSE-1218

Other

- The Cloudera Data Visualization application quits when it hits a session timeout.

Workaround: Upgrade to CDSW 1.8.x or later.

Cloudera Bug: DSE-8943

Cloudera Data Science Workbench 1.7.2 Requirements and Supported Platforms



Important: For requirements and supported platforms specific to Hortonworks Data Platform, see [Deploying Cloudera Data Science Workbench 1.7.2 on Hortonworks Data Platform](#) on page 103.

This topic lists the software and hardware configuration required to successfully install and run Cloudera Data Science Workbench. Cloudera Data Science Workbench does not support hosts or clusters that do not conform to the requirements listed on this page.

Cloudera Manager and CDH Requirements

Cloudera Data Science Workbench 1.7.2 is supported on the following versions of CDH and Cloudera Manager:

Type	CDH	Cloudera Manager
CSD Deployments	<ul style="list-style-type: none"> CDH 5.10 or higher CDH 6.1.x or higher Cloudera Runtime Data Center 7.0.3 or higher 	<ul style="list-style-type: none"> Cloudera Manager 5.x: 5.16.2.4505 or higher Cloudera Manager 6.1.x: 6.1.1.4505 or higher Cloudera Manager 6.2.x: 6.2.1.4505 or higher Cloudera Manager 6.3.x+: 6.3.3 or higher Cloudera Manager Data Center 7.0.3 or higher
RPM Deployments	<ul style="list-style-type: none"> CDH 5.10 or higher CDH 6.1.x or higher 	<ul style="list-style-type: none"> Cloudera Manager 5.11 or higher Cloudera Manager 6.1.x or higher

All cluster hosts must be managed by Cloudera Manager. Note that all Cloudera Data Science Workbench administrative tasks require root access to the cluster's gateway hosts where Cloudera Data Science Workbench is installed. Therefore, Cloudera Data Science Workbench does not support single-user mode installations.

Apache Spark Requirements

CDH Version	Spark 2 Compatibility
CDH 5	CDS 2.1.x Powered by Apache Spark (and higher)
CDH 6	<p>On CDH 6 clusters, Apache Spark 2 is packaged with CDH and can no longer be installed separately.</p> <p>To find out which version of Spark 2 ships with your version of CDH 6, refer the CDH 6 Packaging Information guide.</p>
Cloudera Runtime 7	<p>On Cloudera Runtime clusters, Apache Spark 2 is packaged with Cloudera Runtime and can no longer be installed separately.</p> <p>To find out which version of Spark 2 ships with your version of Cloudera Runtime, refer the Cloudera Runtime Component Versions guide.</p>

Operating System Requirements

Cloudera Data Science Workbench 1.7.2 is supported on the following operating systems. A gateway host that is dedicated to running Cloudera Data Science Workbench must use one of the following supported versions even if the

Cloudera Data Science Workbench 1.7.2 Requirements and Supported Platforms

remaining CDH hosts in your cluster are running any of the other operating systems supported by Cloudera Enterprise [5](#) or [6](#).



Note: Before upgrading the host's operating system or installing a patch, you must stop all CDSW services.

Operating System	Versions	Notes
RHEL / CentOS / Oracle Linux RHCK	7.8, 7.7, 7.6, 7.5, 7.4, 7.3, 7.2	<ul style="list-style-type: none">When IPv6 is disabled, CDSW installations on RHEL/CentOS 7.3 fail due to an issue in kernel versions 3.10.0-514 - 3.10.0-693. For details, see https://access.redhat.com/solutions/3039771.CDSW installations on RHEL/CentOS 7.2 might fail due to an issue with certain versions of the <code>nfs-utils</code> package. To fix the issue, either downgrade the <code>nfs-utils</code> package or upgrade to a version with the fix. View the complete Red Hat bug report here .
Oracle Linux (UEK - default)	7.3	-



Note: **SUSE Linux Enterprise Server (SLES)** - SLES 12 SP2 and SP3 have reached the [end of general support with SUSE](#) and are not supported with Cloudera Data Science Workbench 1.6.0 (and higher).

Cloudera Data Science Workbench [publishes placeholder parcels](#) for other operating systems as well. However, note that these do not work and have only been included to support mixed-OS clusters.

Additional OS-level Settings

- Enable memory cgroups on your operating system.
- Disable swap for optimum stability and performance. For instructions, see [Setting the vm.swappiness Linux Kernel Parameter](#).
- Cloudera Data Science Workbench uses **uid 8536** and **uid 28536** for internal service accounts. Make sure that these user IDs are not assigned to any other service or user account.
- Cloudera recommends that all users have the `max-user-processes` ulimit set to at least 65536.
- Cloudera recommends that all users have the `max-open-files` ulimit set to 1048576.

JDK Requirements

The entire CDH cluster, including Cloudera Data Science Workbench gateway hosts, must use the same version of JDK. Points to remember:

- [Oracle JDK 7](#) is supported across all versions of Cloudera Manager 5 and CDH 5. [Oracle JDK 8](#) is supported in Cloudera Enterprise 5.3.x and higher. Note the [JDK 8 Requirement for Spark 2.2 \(or higher\)](#) on page 67.
- OpenJDK 8 is supported in Cloudera Enterprise 5.16.1 and higher. OpenJDK 7 is not supported.
- For Red Hat/CentOS deployments in particular, **Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction** must be enabled on the Cloudera Data Science Workbench gateway hosts.

For more specifics on the versions of Oracle JDK and OpenJDK recommended for CDH and Cloudera Manager clusters, and instructions on how to install the Java Cryptography Extension, see the Cloudera Product Compatibility Matrix - [Supported JDK Versions](#).

JDK 8 Requirement for Spark 2.2 (or higher)

CSD-based deployments:

On CSD-based deployments, Cloudera Manager automatically detects the path and version of Java installed on Cloudera Data Science Workbench gateway hosts. You do not need to explicitly set the value for `JAVA_HOME` unless you want to use a custom location, use JRE, or (in the case of Spark 2) force Cloudera Manager to use JDK 1.8 as explained below.

To upgrade your entire CDH cluster to JDK 1.8, see [Upgrading to Oracle JDK 1.8](#).

Package-based deployments:

Set `JAVA_HOME` to the JDK 8 path in `cdsw.conf` during the [installation process](#). If you need to modify `JAVA_HOME` after the fact, [restart the master and worker hosts](#) to have the changes go into effect.

Networking and Security Requirements



Important: Make sure that your networking/security settings on Cloudera Data Science Workbench gateway hosts are not being overwritten behind-the-scenes, either by any automated scripts, or by other high-priority configuration such as `/etc/sysctl.conf`, `/etc/krb5.conf`, or `/etc/hosts.deny`.

- Enable IPv6 on all Cloudera Data Science Workbench gateway hosts. For instructions, refer the workaround provided here: [Known Issue: CDSW cannot start sessions due to connection errors](#).
- All Cloudera Data Science Workbench gateway hosts must be part of the same datacenter and use the same network. Hosts from different data-centers or networks can result in unreliable performance.
- A wildcard subdomain such as `*.cdsw.company.com` must be configured. Wildcard subdomains are used to provide isolation for user-generated content.

The wildcard DNS hostname configured for Cloudera Data Science Workbench must be resolvable from both, the CDSW cluster, and your browser.

- Disable all pre-existing `iptables` rules. While Kubernetes makes extensive use of `iptables`, it's difficult to predict how pre-existing `iptables` rules will interact with the rules inserted by Kubernetes. Therefore, Cloudera recommends you to disable all pre-existing rules before you proceed with the installation.

It is recommended to save the `iptables` and check whether the changes have been written to the `/etc/sysconfig/iptables` file before you disable them. If you disable the `iptables` without saving, then the settings can get erased upon system reboot.

1. Save the `iptables` by running the following command:

```
service iptables save
```

2. Verify whether the changes have been written to the file by running the following command:

```
ls -l /etc/sysconfig/iptables
```

3. Disable the `iptables` by running the following commands:

```
sudo iptables -P INPUT ACCEPT
sudo iptables -P FORWARD ACCEPT
sudo iptables -P OUTPUT ACCEPT
sudo iptables -t nat -F
sudo iptables -t mangle -F
sudo iptables -F
sudo iptables -X
```

- Cloudera Data Science Workbench sets the following `sysctl` options in `/etc/sysctl.d/k8s.conf`:

Cloudera Data Science Workbench 1.7.2 Requirements and Supported Platforms

- net.bridge.bridge-nf-call-iptables=1
- net.bridge.bridge-nf-call-ip6tables=1
- net.ipv4.ip_forward=1
- net.ipv4.conf.default.forwarding=1

Underlying components of Cloudera Data Science Workbench (Docker, Kubernetes, and NFS) require these options to work correctly. Make sure they are not overridden by high-priority configuration such as `/etc/sysctl.conf`.

- SELinux must either be disabled or run in *permissive* mode.
- Multi-homed networks are supported with Cloudera Data Science Workbench 1.2.2 (and higher). However, you will need to explicitly configure the private IP address of the worker nodes in the kubelet start script as follows:

```
# vi /opt/cloudera/parcels/CDSW/scripts/start-kubelet-worker-standalone-core.sh
88 kubelet_opts+=(--v=2)
89 kubelet_opts+=(--node-ip=172.x.x.x)
```

- Firewall restrictions must be disabled across Cloudera Data Science Workbench and CDH/HDP cluster hosts. For more details on cluster communication, see [Ports Required by Cloudera Data Science Workbench](#) on page 68.
- Untrusted (non-sudo) SSH access to Cloudera Data Science Workbench hosts must be disabled to ensure a secure deployment.

Cloudera Data Science Workbench assumes that users only access the gateway hosts through the web application. Untrusted users with SSH access to a Cloudera Data Science Workbench host can gain full access to the cluster, including access to other users' workloads.

- `localhost` must resolve to `127.0.0.1`.
- Forward and reverse DNS lookup must be enabled for the Cloudera Data Science Workbench domain name and IP address (CDSW master host).
- Cloudera Data Science Workbench does not support DNS servers running on `127.0.0.1:53`. This IP address resolves to the container localhost within Cloudera Data Science Workbench containers. As a workaround, use either a non-loopback address or a remote DNS server.
- All third-party security software (such as McAfee, Tanium, Symantec, etc.) must be disabled on CDSW hosts. Failure to do so can result in Cloudera Data Science Workbench failing randomly.

Cloudera Data Science Workbench does not support hosts or clusters that do not conform to these restrictions.

Ports Required by Cloudera Data Science Workbench

Cloudera Data Science Workbench runs on gateway hosts in a CDH/HDP cluster. As such, Cloudera Data Science Workbench acts as a gateway and requires full connectivity to cluster services such as Impala, Spark 2, etc. Additionally, in the case of Spark 2, cluster hosts will require access to the Spark driver running on a set of random ports (20050-32767) on Cloudera Data Science Workbench hosts.

Firewall restrictions must be disabled across Cloudera Data Science Workbench and CDH/HDP cluster hosts. Internally, the Cloudera Data Science Workbench master and worker hosts require full connectivity with no firewalls. Externally, end users connect to Cloudera Data Science Workbench exclusively through a web server running on the master host, and therefore do not need direct access to any other internal Cloudera Data Science Workbench or CDH services.

This information has been summarized in the following table.

Components	Details
Communication with the CDH / HDP cluster	CDH / HDP -> Cloudera Data Science Workbench The CDH/HDP cluster must have access to the Spark driver that runs on Cloudera Data Science Workbench hosts, on a set of randomized ports in the range, 20050-32767 .
	Cloudera Data Science Workbench -> CDH / HDP As a gateway service, Cloudera Data Science Workbench must have access to all the ports used by CDH and Cloudera Manager .

Components	Details
Communication with the Web Browser	The Cloudera Data Science Workbench web application is available at port 80 . HTTPS access is available over port 443 .

Recommended Hardware Configuration



Important:

- Allocate separate CDH gateway hosts for Cloudera Data Science Workbench. Do not reuse existing hosts that are already running other CDH services. Doing this can lead to port conflicts, unreliable execution of user workloads, and out-of-memory errors.
- For proof-of-concept deployments, you can deploy a 1-host cluster with just a Master host. The Master host can run user workloads just as a worker host can when required for demonstration purposes. For production deployments, Cloudera requires to have a reserved, dedicated master host and separate worker host(s).
- All Cloudera Data Science Workbench gateway hosts must be part of the same datacenter and use the same network. Hosts from different data-centers or networks can result in unreliable performance.
- Starting with version 1.4.3, multi-host CDSW deployments can be customized to reserve the Master only for internal processes while user workloads are run exclusively on workers. For details, see [Reserving the Master Host for Internal CDSW Components](#) on page 269.

Cloudera Data Science Workbench hosts are added to your CDH cluster as gateway hosts. The recommended minimum hardware configuration for Cloudera Data Science Workbench gateway hosts is:

Resource Type	Master	Workers	Notes
Supported architecture			Intel 64 bit, AMD 64
CPU	16+ CPU (vCPU) cores	16+ CPU (vCPU) cores	
RAM	32+ GB	32+ GB	
Disk Space			
Root Volume	100+ GB	100+ GB	If you are going to partition the root volume, make sure you allocate at least 20 GB to / so that the installer can proceed without running out of space.
Application Block Device	1 TB	-	The Application Block Device is only required on the Master where it is mounted to <code>/var/lib/cdsw</code> . You will be asked to create a <code>/var/lib/cdsw</code> directory on all the Worker hosts during the installation process. However, they do not need to be mounted to a block device. It is only used to store client configuration for HDP cluster services on Workers.
Docker Block Device	1 TB	1 TB	The Docker Block Device is required on <i>all</i> Master and Worker hosts.

Scaling Guidelines

Cloudera Data Science Workbench 1.7.2 Requirements and Supported Platforms

New hosts can be added and removed from a Cloudera Data Science Workbench deployment without interrupting any jobs already scheduled on existing hosts. Therefore, it is rather straightforward to increase capacity based on observed usage. At a minimum, Cloudera recommends you allocate at least 1 CPU core and 2 GB of RAM per concurrent session or job. CPU can burst above a 1 CPU core share when spare resources are available. Therefore, a 1 CPU core allocation is often adequate for light workloads. Allocating less than 2 GB of RAM can lead to out-of-memory errors for many applications.

As a general guideline, Cloudera recommends hosts with RAM between 60GB and 256GB, and between 16 and 48 cores. This provides a useful range of options for end users. Note that SSDs are *strongly recommended* for application data storage. Using standard HDDs can sometimes result in poor application performance.

For some data science and machine learning applications, users can collect a significant amount of data in memory within a single R or Python process, or use a significant amount of CPU resources that cannot be easily distributed into the CDH cluster. If individual users frequently run larger workloads or run workloads in parallel over long durations, increase the total resources accordingly. Understanding your users' concurrent workload requirements or observing actual usage is the best approach to scaling Cloudera Data Science Workbench.

Python Supported Versions

The default Cloudera Data Science Workbench engine currently includes **Python 2.7.17** and **Python 3.6.9**. CDSW supports what comes bundled with the base image. To use PySpark with lambda functions that run within the CDH cluster, the Spark executors must have access to a matching version of Python. For many common operating systems, the default system Python will not match the minor release of Python included in Data Science Workbench.

To ensure that the Python versions match, Python can either be installed on every CDH host or made available per job run using Spark's ability to distribute dependencies. Given the size of a typical isolated Python environment and the desire to avoid repeated uploads from gateway hosts, Cloudera recommends installing Python 2.7 and 3.6 on the cluster if you are using PySpark with lambda functions.

You can install Python 2.7 and 3.6 on the cluster using any method and set the corresponding `PYSPARK_PYTHON` environment variable in your project. Cloudera Data Science Workbench 1.3 (and higher) include a separate environment variable for Python 3 sessions called `PYSPARK3_PYTHON`. Python 2 sessions continue to use the default `PYSPARK_PYTHON` variable. This will allow you to run Python 2 and Python 3 sessions in parallel without either variable being overridden by the other.

For an example on distributing Python dependencies dynamically, see [Example: Distributing Dependencies on a PySpark Cluster](#) on page 244.

Anaconda

Continuum Analytics and Cloudera have partnered to create an [Anaconda parcel](#) for CDH to enable simple distribution, installation, and management of popular Python packages and their dependencies. Note that this parcel is not directly supported by Cloudera.

Docker and Kubernetes Support

Cloudera Data Science Workbench only supports the versions of [Docker](#) and [Kubernetes](#) that are shipped with each release. Upgrading Docker or Kubernetes, or running on third-party Kubernetes clusters is not supported.

Supported Browsers

- Chrome (latest stable version)
- Firefox (latest released version and latest ESR version)
- Safari 9+

Cloudera Altus Director Support (AWS and Azure Only)

Altus Director support for Cloudera Data Science Workbench is available for the following platforms:

- **Amazon Web Services (AWS)** - Cloudera Altus Director 2.6.0 (and higher)
- **Microsoft Azure** - Cloudera Altus Director 2.7 (and higher)
- Cloudera Manager 5.13.1 (and higher)
- CSD-based Cloudera Data Science Workbench 1.2.x (and higher)

Deploying Cloudera Data Science Workbench with Altus Director

Points to note when using Altus Director to install Cloudera Data Science Workbench:

- **(Required for Director 2.6)** Before you run the command to bootstrap a new cluster, set the `lp.normalization.mountAllUnmountedDisksRequired` property to `false` in the Altus Director server's `application.properties` file, and then *restart Altus Director*.

Higher versions of Altus Director do not require this step. Altus Director 2.7 (and higher) include an instance-level setting called `mountAllUnmountedDisks` that must be set to **false** as demonstrated in the following sample configuration files.
- Depending on your cloud platform, you can use one of the following sample configuration files to deploy a Cloudera Manager cluster with Cloudera Data Science Workbench.
 - AWS - [aws.cdsw.conf](#)
 - Azure - [azure.cdsw.conf](#)

Note that these sample files are tailored to Altus Director 2.7 (and higher) and they install a very limited CDH cluster with just the following services: HDFS, YARN, and Spark 2. You can extend them as needed to match your use case.

Related Topics:

- [Using Products outside CDH with Altus Director](#)
- [Altus Director CLI](#)
- [The Altus Director Configuration File](#)
- [Setting Altus Director Properties](#)

Recommended Configuration on Amazon Web Services (AWS)

On AWS, Cloudera Data Science Workbench must be used with persistent/long-running Apache Hadoop clusters only.

CDH and Cloudera Manager Hosts

- For instructions on deploying CDH and Cloudera Manager on AWS, refer the [Cloudera Reference Architecture for AWS deployments](#).

Cloudera Data Science Workbench Hosts

- **Operations**
 - Use Cloudera Director to orchestrate operations. Use Cloudera Manager to monitor the cluster.
- **Networking**
 - No security group or network restrictions between hosts.
 - HTTP connectivity to the corporate network for browser access. Do not use proxies or manual SSH tunnels.
- **Recommended Instance Types**

Cloudera Data Science Workbench 1.7.2 Requirements and Supported Platforms

- m4.4xlarge–m4.16xlarge

In this case, bigger is better. That is, one m4.16large is better than four m4.4xlarge hosts. AWS pricing scales linearly, and larger instances have more EBS bandwidth.

- **Storage**
 - 100 GB root volume block device (gp2) on all hosts
 - 500 GB Docker block devices (gp2) on all hosts
 - 1 TB Application block device (io1) on master host

Recommended Configuration on Microsoft Azure

CDH and Cloudera Manager Hosts

- For instructions on deploying CDH and Cloudera Manager on Azure, refer the [Cloudera Reference Architecture for Azure deployments](#).

Cloudera Data Science Workbench Hosts

- **Operations**
 - Use Cloudera Director to orchestrate operations. Use Cloudera Manager to monitor the cluster.
- **Networking**
 - No security group or network restrictions between hosts.
 - HTTP connectivity to the corporate network for browser access. Do not use proxies or manual SSH tunnels.
- **Recommended Instance Types**
 - DS13-DS14 v2 instances on all hosts.
- **Storage**
 - P30 premium storage for the Application and Docker block devices.

Cloudera Data Science Workbench *requires* premium disks for its block devices on Azure. Standard disks can lead to unacceptable performance even on small clusters.

Installing Cloudera Data Science Workbench 1.7.2 on CDH

This topic walks you through the installation paths available for Cloudera Data Science Workbench 1.7.2. It also describes the steps needed to configure your cluster gateway hosts and block devices before you can begin installing the Cloudera Data Science Workbench parcel/package.



Note: As of February 1, 2021, all downloads of CDH and Cloudera Manager require a username and password and use a modified URL. You must use the modified URL, including the username and password when downloading the repository contents described below. You may need to upgrade Cloudera Manager to a newer version that uses the modified URLs.

This can affect new installations, upgrades, adding new hosts to a cluster, and adding a cluster.

For more information, see *Updating an existing CDH/Cloudera Manager deployment to access downloads with authentication* for [CDH5](#) or [CDH6](#).

For installation and upgrades, you must manually add the **Remote Parcel Repository URLs** for your CDSW version to Cloudera Manager.

Installing Cloudera Data Science Workbench 1.7.2

You can use one of the following ways to install Cloudera Data Science Workbench 1.7.2:

- **Using a Custom Service Descriptor (CSD) and Parcel** - Starting with version 1.2.x, Cloudera Data Science Workbench is available as an add-on service for Cloudera Manager. Two files are required for this type of installation: a CSD JAR file that contains all the configuration needed to describe and manage the new Cloudera Data Science Workbench service, and the Cloudera Data Science Workbench parcel. To install this service, first download and copy the CSD file to the Cloudera Manager Server host. Then use Cloudera Manager to distribute the Cloudera Data Science Workbench parcel to the relevant gateway hosts.

Note that this installation mode does not apply to CDSW-on-HDP deployments.

or

- **Using a Package (RPM)** - You can install the Cloudera Data Science Workbench package directly on your cluster's gateway or edge hosts. In this case, you will not be able to manage the Cloudera Data Science Workbench service from a cluster manager such as Cloudera Manager or Ambari.

To begin the installation process, continue reading [Required Pre-Installation Steps](#) on page 74.

Multiple Cloudera Data Science Workbench Deployments

Starting with version 1.6, you can add more than one Cloudera Data Science Workbench CSD deployment to a single instance of Cloudera Manager. The deployments must install the same version of CDSW.

To add a second Cloudera Data Science Workbench to Cloudera Manager, complete the [Required Pre-Installation Steps](#) on page 74 for a second set of gateway hosts. Then, install the parcel and add the service as described in [the CSD installation](#).

Airgapped Installations

Sometimes organizations choose to restrict parts of their network from the Internet for security reasons. Isolating segments of a network can provide assurance that valuable data is not being compromised by individuals out of maliciousness or for personal gain. However, in such cases isolated hosts are unable to access Cloudera repositories

Installing Cloudera Data Science Workbench 1.7.2 on CDH

for new installations or upgrades. Effective version 1.1.1, Cloudera Data Science Workbench supports installation on CDH clusters that are not connected to the Internet.

For CSD-based installs in an airgapped environment, put the Cloudera Data Science Workbench parcel into a [new hosted or local parcel repository](#), and then configure the Cloudera Manager Server to target this newly-created repository.

Required Pre-Installation Steps

The rest of this topic describes the steps you should take to review your platforms and configure your gateway hosts before you begin to install Cloudera Data Science Workbench.



Note: For proof-of-concept deployments, you can deploy a 1-host cluster with just a Master host. The Master host can run user workloads just as a worker host can when required for demonstration purposes. For production deployments, Cloudera requires to have a reserved, dedicated master host and separate worker host(s).

Even on multi-host deployments, the Master host doubles up to perform both functions: those of the Master [outlined here](#), and those of a worker.

Review Requirements and Supported Platforms

Review the complete list of [Cloudera Data Science Workbench 1.7.2 Requirements and Supported Platforms](#) on page 65 before you proceed with the installation.

Set Up a Wildcard DNS Subdomain

Cloudera Data Science Workbench uses DNS to route HTTP requests to specific engines and services. Wildcard subdomains (such as `*.cdsw.<your_domain>.com`) are required in order to provide isolation for user-generated content. In particular, wildcard subdomains help:

- Securely expose interactive session services, such as visualizations, the terminal, and web UIs such as TensorBoard, Shiny, Plotly, and so on.
- Securely isolate user-generated content from the application.

To set up subdomains for Cloudera Data Science Workbench, configure your DNS server with an A record for a wildcard DNS name such as `*.cdsw.<your_domain>.com` for the master host, and a second A record for the root entry of `cdsw.<your_domain>.com`.

For example, if your master IP address is `172.46.47.48`, you'd configure two A records as follows:

```
cdsw.<your_domain>.com.    IN A 172.46.47.48
*.cdsw.<your_domain>.com.  IN A 172.46.47.48
```

You can also use a wildcard CNAME record if it is supported by your DNS provider.

Starting with version 1.5, the wildcard DNS hostname configured for Cloudera Data Science Workbench must now be resolvable from both, the CDSW cluster, and your Computer from where you are accessing the CDSW UI.

Disable Untrusted SSH Access

Cloudera Data Science Workbench assumes that users only access the gateway hosts through the web application. Untrusted users with SSH access to a Cloudera Data Science Workbench host can gain full access to the cluster, including access to other users' workloads. Therefore, untrusted (non-sudo) SSH access to Cloudera Data Science Workbench hosts must be disabled to ensure a secure deployment.

For more information on the security capabilities of Cloudera Data Science Workbench, see the [Cloudera Data Science Workbench Security Guide](#) on page 278.

Configure Block Devices

Docker Block Device

The Cloudera Data Science Workbench installer will format and mount Docker on each gateway host. Make sure there is no important data stored on these devices. **Do not mount or format the Docker Block Devices before installing Cloudera Data Science Workbench.**

Every Cloudera Data Science Workbench gateway host must have one or more block devices with at least 1 TB dedicated to storage of Docker images. The Docker block devices store the Cloudera Data Science Workbench Docker images including the Python, R, and Scala engines. Each engine image can occupy 15GB.

Application Block Device or Mount Point

The master host on Cloudera Data Science Workbench requires at least 1 TB for database and project storage. This recommended capacity is contingent on the expected number of users and projects on the cluster. While large data files should be stored on HDFS, it is not uncommon to find gigabytes of data or libraries in individual projects. Running out of storage will cause the application to fail. Cloudera recommends allocating at least 5 GB per project and at least 1 TB of storage in total. Make sure you continue to carefully [monitor disk space usage](#) and I/O using Cloudera Manager.

Cloudera Data Science Workbench stores all application data at `/var/lib/cdsw`. On a CSD-based deployment, this location is not configurable. The Application Block Device should be formatted before installing Cloudera Data Science Workbench. Cloudera Data Science Workbench will assume the system administrator has formatted and mounted one or more block devices to `/var/lib/cdsw` on the *master* host. Note that Application Block Device mounts are not required on worker hosts.

Regardless of the application data storage configuration you choose, `/var/lib/cdsw` must be stored on a separate block device. Given typical database and user access patterns, an SSD is *strongly* recommended.

The `/var/lib/cdsw` directory contains persistent information such as database, configurations, image details and so on. By default, data in `/var/lib/cdsw` is not backed up or replicated to HDFS or other hosts. Reliable storage and backup strategy is critical for production installations. For more information, see [Backup and Disaster Recovery for Cloudera Data Science Workbench](#) on page 266. To migrate the user-related information to a new host, you can transfer the `/var/lib/cdsw` directory to the new host.

UUID cannot be used in place of the disk name to identify a block device in CDSW. You can determine whether a given path is a block device or not by using the following expression:

```
if [ -b $FILE ]
```

If this expression returns `True` or `1`, then the file exists and is a block special file. This means that the parameter is a path to the block device and cannot be a plain UUID.

Install Cloudera Data Science Workbench

To use the Cloudera Manager CSD and parcel to install Cloudera Data Science Workbench, follow the steps at [Installation and Upgrade Using Cloudera Manager](#).

OR

To install the Cloudera Data Science Workbench package on the cluster gateway hosts, follow the steps at [Installation and Upgrade Using Packages](#).

Installing Cloudera Data Science Workbench 1.7.2 Using Cloudera Manager



Note: Cloudera Data Science Workbench 1.7.1 is the next official release after Cloudera Data Science Workbench 1.6.x. Version 1.7.0 is no longer publicly available.

Use the following steps to install Cloudera Data Science Workbench using Cloudera Manager.

Prerequisites

Before you begin installing Cloudera Data Science Workbench, make sure you have completed the steps to [secure your hosts, set up DNS subdomains, and configure block devices](#).



Note: For proof-of-concept deployments, you can deploy a 1-host cluster with just a Master host. The Master host can run user workloads just as a worker host can when required for demonstration purposes. For production deployments, Cloudera requires to have a reserved, dedicated master host and separate worker host(s).

Configure Apache Spark 2

Depending on the platform you are using, use one of the following sections to configure Apache Spark 2.

CDH 5

1. Install and configure the CDS 2.x Powered by Apache Spark parcel and CSD. For instructions, see [Installing CDS 2.x Powered by Apache Spark](#).



Important: Do not install CDS 2.x if you are using CDH 6. Spark 2 ships as part of the CDH 6 parcel; the add-on parcel is no longer required. To see which version of Spark 2 ships with CDH, refer the [CDH 6 Packaging documentation](#).

2. To be able to use Spark 2, each user must have their own `/home` directory in HDFS. If you sign in to Hue first, these directories will automatically be created for you. Alternatively, you can have cluster administrators create these directories.

```
hdfs dfs -mkdir /user/<username>
hdfs dfs -chown <username>:<username> /user/<username>
```

If you are using CDS 2.3 release 2 (or higher), review the associated known issues here: [CDS Powered By Apache Spark](#) on page 55.

3. Use Cloudera Manager to create add gateway hosts to your CDH cluster.
 1. Create a new [host template](#) that includes gateway roles for HDFS, YARN, and Spark 2.
(Required for CDH 6 and CDP Data Center) If you want to run workloads on dataframe-based tables, such as tables from PySpark, sparklyr, SparkSQL, or Scala, you must also add the Hive gateway role to the template.
 2. Use the instructions at [Adding a Host to the Cluster](#) to add gateway hosts to the cluster. Apply the template created in the previous step to these gateway hosts. If your cluster is kerberized, confirm that the [krb5.conf](#) file on your gateway hosts is correct.
4. Test Spark 2 integration on the gateway hosts.
 1. SSH to a gateway host.
 2. If your cluster is kerberized, run `kinit` to authenticate to the CDH cluster's Kerberos Key Distribution Center. The Kerberos ticket you create is not visible to Cloudera Data Science Workbench users.
 3. Submit a test job to Spark by executing the following command:

CDH 5

```
spark2-submit --class org.apache.spark.examples.SparkPi --master yarn \
--deploy-mode client
/opt/cloudera/parcels/SPARK2/lib/spark2/examples/jars/spark-example*.jar 100
```

To view a sample command, click

```
spark2-submit --class org.apache.spark.examples.SparkPi --master yarn \
--deploy-mode client
/opt/cloudera/parcels/SPARK2/lib/spark2/examples/jars/spark-examples*.jar 100
```

CDH 6 and CDP Data Center 7

```
spark-submit --class org.apache.spark.examples.SparkPi --master yarn \
--deploy-mode client SPARK_HOME/lib/spark-examples*.jar 100
```

To view a sample command, click

```
spark-submit --class org.apache.spark.examples.SparkPi --master yarn \
--deploy-mode client /opt/cloudera/parcels/CDH/lib/spark/examples/jars/spark-examples*.jar
100
```

4. View the status of the job in the CLI output or in the Spark web UI to confirm that the host you want to use for the Cloudera Data Science Workbench master functions properly as a Spark gateway.

To view sample CLI output, click

```
19/02/15 09:37:39 INFO spark.SparkContext: Running Spark version 2.4.0-cdh6.1.0
19/02/15 09:37:39 INFO spark.SparkContext: Submitted application: Spark Pi
...
19/02/15 09:37:40 INFO util.Utils: Successfully started service 'sparkDriver' on port
37050.
...
19/02/15 09:38:06 INFO scheduler.DAGScheduler: Job 0 finished: reduce at SparkPi.scala:38,
took 18.659033 s
```

CDH 6 or CDP Data Center 7

1. To be able to use Spark 2, each user must have their own `/home` directory in HDFS. If you sign in to Hue first, these directories will automatically be created for you. Alternatively, you can have cluster administrators create these directories.

```
hdfs dfs -mkdir /user/<username>
hdfs dfs -chown <username>:<username> /user/<username>
```

2. Use Cloudera Manager to create add gateway hosts to your CDH cluster.
 1. Create a new [host template](#) that includes gateway roles for HDFS, YARN, and Spark 2.

If you want to run workloads on dataframe-based tables, such as tables from PySpark, sparklyr, SparkSQL, or Scala, you must also add the Hive gateway role to the template.
 2. Use the instructions at [Adding a Host to the Cluster](#) to add gateway hosts to the cluster. Apply the template created in the previous step to these gateway hosts. If your cluster is kerberized, confirm that the [krb5.conf](#) file on your gateway hosts is correct.
3. Test Spark 2 integration on the gateway hosts.
 1. SSH to a gateway host.
 2. If your cluster is kerberized, run `kinit` to authenticate to the CDH cluster's Kerberos Key Distribution Center. The Kerberos ticket you create is not visible to Cloudera Data Science Workbench users.
 3. Submit a test job to Spark by executing the following command:

```
spark-submit --class org.apache.spark.examples.SparkPi --master yarn \
--deploy-mode client SPARK_HOME/lib/spark-examples*.jar 100
```

Installing Cloudera Data Science Workbench 1.7.2 on CDH

To view a sample command, click

```
spark-submit --class org.apache.spark.examples.SparkPi --master yarn \
--deploy-mode client /opt/cloudera/parcels/CDH/lib/spark/examples/jars/spark-examples*.jar
100
```

4. View the status of the job in the CLI output or in the Spark web UI to confirm that the host you want to use for the Cloudera Data Science Workbench master functions properly as a Spark gateway.

To view sample CLI output, click

```
19/02/15 09:37:39 INFO spark.SparkContext: Running Spark version 2.4.0-cdh6.1.0
19/02/15 09:37:39 INFO spark.SparkContext: Submitted application: Spark Pi
...
19/02/15 09:37:40 INFO util.Utils: Successfully started service 'sparkDriver' on port
37050.
...
19/02/15 09:38:06 INFO scheduler.DAGScheduler: Job 0 finished: reduce at SparkPi.scala:38,
took 18.659033 s
```

Configure JAVA_HOME

On CSD-based deployments, Cloudera Manager automatically detects the path and version of Java installed on Cloudera Data Science Workbench gateway hosts. You do not need to explicitly set the value for `JAVA_HOME` unless you want to use a custom location, use JRE, or (in the case of Spark 2) force Cloudera Manager to use JDK 1.8 as explained below.

Setting a value for JAVA_HOME - The value for `JAVA_HOME` depends on whether you are using JDK or JRE. For example, if you're using JDK 1.8_162, set `JAVA_HOME` to `/usr/java/jdk1.8.0_162`. If you are only using JRE, set it to `/usr/java/jdk1.8.0_162/jre`.

Issues with Spark 2.2 and higher - Spark 2.2 (and higher) requires JDK 1.8. However, if a host has both JDK 1.7 and JDK 1.8 installed, Cloudera Manager might choose to use JDK 1.7 over JDK 1.8. If you are using Spark 2.2 (or higher), this will create a problem during the first run of the service because Spark will not work with JDK 1.7. To work around this, explicitly configure Cloudera Manager to use JDK 1.8 on the gateway hosts that are running Cloudera Data Science Workbench.

For instructions on how to set `JAVA_HOME`, see [Configuring a Custom Java Home Location in Cloudera Manager](#).

To upgrade the whole CDH cluster to JDK 1.8, see [Upgrading to JDK 1.8](#).

Download and Install the Cloudera Data Science Workbench CSD

1. Download the Cloudera Data Science Workbench CSD. Make sure you download the CSD that corresponds to the version of CDH or Cloudera Runtime you are using.

- **CDP Data Center**

```
https://archive.cloudera.com/p/cdsw1/1.7.2/csd/CLOUDERA_DATA_SCIENCE_WORKBENCH-CDPDC-1.7.2.jar
```

OR

- **CDH 6**

```
https://archive.cloudera.com/p/cdsw1/1.7.2/csd/CLOUDERA_DATA_SCIENCE_WORKBENCH-CDH6-1.7.2.jar
```

OR

- **CDH 5**

```
https://archive.cloudera.com/p/cdsw1/1.7.2/csd/CLOUDERA_DATA_SCIENCE_WORKBENCH-CDH5-1.7.2.jar
```

- Log on to the Cloudera Manager Server host, and place the CSD file under `/opt/cloudera/csd`, which is the default location for CSD files. To configure a custom location for CSD files, refer to the Cloudera Manager documentation at [Configuring the Location of Custom Service Descriptor Files](#).
- Set the file ownership to `cloudera-scm:cloudera-scm` with permission 644.

Set the file ownership.

CDH

```
chown cloudera-scm:cloudera-scm CLOUDERA_DATA_SCIENCE_WORKBENCH-CDH<X>-1.7.<Y>.jar
```

CDP Data Center

```
chown cloudera-scm:cloudera-scm CLOUDERA_DATA_SCIENCE_WORKBENCH-CDPDC-1.7.<Y>.jar
```

- Set the file permissions.

CDH

```
chmod 644 CLOUDERA_DATA_SCIENCE_WORKBENCH-CDH<X>-1.7.<Y>.jar
```

CDP Data Center

```
chmod 644 CLOUDERA_DATA_SCIENCE_WORKBENCH-CDPDC-1.7.<Y>.jar
```

- Restart the Cloudera Manager Server:

```
service cloudera-scm-server restart
```

- Log into the Cloudera Manager Admin Console and restart the Cloudera Management Service:
 - Select **Clusters > Cloudera Management Service**.
 - Select **Actions > Restart**.

Install the Cloudera Data Science Workbench Parcel

- Log into the Cloudera Manager Admin Console.
- Click **Hosts > Parcels** in the main navigation bar.
- Add the remote parcel repository URL to Cloudera Manager. For detailed steps, click
 - On the **Parcels** page, click **Configuration**.
 - In the **Remote Parcel Repository URLs** list, click the addition symbol to open an additional row.
 - Enter the path to the repository. Cloudera Data Science Workbench publishes placeholder parcels for other operating systems as well. However, note that these do not work and have only been included to support mixed-OS clusters.

Version	Remote Parcel Repository URL
Cloudera Data Science Workbench 1.7.2	<code>https://archive.cloudera.com/p/cdsw1/1.7.2/parcels/</code>
Cloudera Data Science Workbench 1.7.1	<code>https://archive.cloudera.com/p/cdsw1/1.7.1/parcels/</code>

4. Click **Save Changes**.

5. Go to the **Hosts > Parcels** page. The external parcel should now appear in the set of parcels available for download.

4. Click **Download**. Once the download is complete, click **Distribute** to distribute the parcel to all the CDH hosts in your cluster. Then click **Activate**. For more detailed information on each of these tasks, see [Managing Parcels](#).

For airgapped installations, [create your own local repository](#), put the Cloudera Data Science Workbench parcel there, and then configure the Cloudera Manager Server to target this newly-created repository.

Add the Cloudera Data Science Workbench Service

Perform the following steps to add the Cloudera Data Science Workbench service to your cluster:

1. Log into the Cloudera Manager Admin Console.
2. On the **Home > Status** tab, click



to the right of the cluster name and select **Add a Service** to launch the wizard. A list of services will be displayed.

3. Select the Cloudera Data Science Workbench service and click **Continue**.
4. Select the services which the new CDSW service should depend on. At a minimum, the HDFS, Spark 2, and YARN services are required for the CDSW service to run successfully. Click **Continue**.

(Required for CDH 6) If you want to run SparkSQL workloads, you must also add the Hive service as a dependency.

5. Assign the Cloudera Data Science Workbench roles, HDFS, Spark 2, and YARN, to gateway hosts.

Master

Assign the Master role to a gateway host that is the designated Master host. This is the host that should have the Application Block Device mounted to it.

Worker

Assign the Worker role to any other gateway hosts that will be used for Cloudera Data Science Workbench. Note that Worker hosts are not required for a fully-functional Cloudera Data Science Workbench deployment. For proof-of-concept deployments, you can deploy a 1-host cluster with just a Master host. The Master host can run user workloads just as a worker host can.

Even if you are setting up a multi-host deployment, **do not assign the Master and Worker roles to the same host**. By default, the Master host doubles up to perform both functions: those of the Master, [outlined here](#), and those of a worker.

Docker Daemon

This role runs underlying Docker processes on *all* Cloudera Data Science Workbench hosts. The Docker Daemon role must be assigned to **every** Cloudera Data Science Workbench gateway host.

On First Run, Cloudera Manager will automatically assign this role to each Cloudera Data Science Workbench gateway host. However, if any [more hosts are added](#) or reassigned to Cloudera Data Science Workbench, you must explicitly assign the Docker Daemon role to them.
















Application

This role runs the Cloudera Data Science Workbench application. This role runs only on the CDSW Master host.

On First Run, Cloudera Manager will assign the Application role to the host running the Cloudera Data Science Workbench Master role. The Application role is always assigned to the same host as the Master. Consequently, this role must never be assigned to a Worker host.

The following image shows the role assignments for a Cloudera Data Science Workbench Master host and Worker host:

Roles

Hosts	Count	Roles
10.10.10.10	1	 B  G  NN  RM  SM  G
10.10.10.11	1	 A  DD  M
10.10.10.12	1	 DD  W  G
10.10.10.13	2	 DN  G  NM

This table is grouped by hosts having the same roles assigned to them.

6. Configure the following parameters and click **Continue**.

Properties	Description
Cloudera Data Science Workbench Domain	<p>DNS domain configured to point to the master host.</p> <p>If the previously configured DNS subdomain entries are <code>cdsw.<your_domain>.com</code> and <code>*.cdsw.<your_domain>.com</code>, then this parameter should be set to <code>cdsw.<your_domain>.com</code>.</p> <p>Users' browsers contact the Cloudera Data Science Workbench web application at <code>http://cdsw.<your_domain>.com</code>.</p> <p>This domain for DNS only and is unrelated to Kerberos or LDAP domains.</p>
Master Node IPv4 Address	<p>IPv4 address for the master host that is reachable from the worker host. <i>By default, this field is left blank and Cloudera Manager uses the IPv4 address of the Master host.</i></p>

Properties	Description
	Within an AWS VPC, set this parameter to the internal IP address of the master host; for instance, if your hostname is <code>ip-10-251-50-12.ec2.internal</code> , set this property to the corresponding IP address, <code>10.251.50.12</code> .
Install Required Packages	<p>When this parameter is enabled, the Prepare Node command will install all the required package dependencies on First Run. If you choose to disable this property, you must manually install the following packages on <i>all</i> gateway hosts running Cloudera Data Science Workbench roles:</p> <pre>nfs-utils libseccomp lvm2 bridge-utils libtool-ltdl iptables rsync policycoreutils-python selinux-policy-base selinux-policy-targeted ntp ebtables bind-utils nmap-ncat openssl e2fsprogs redhat-lsb-core conntrack-tools socat</pre>
Docker Block Device	<p>Block device(s) for Docker images. Use the full path to specify the image(s), for instance, <code>/dev/xvde</code>.</p> <p>The Cloudera Data Science Workbench installer will format and mount Docker on each gateway host that is assigned the Docker Daemon role. <i>Do not mount these block devices prior to installation.</i></p>

- The wizard will now begin a First Run of the Cloudera Data Science Workbench service. This includes deploying client configuration for HDFS, YARN and Spark 2, installing the package dependencies on all hosts, and formatting the Docker block device. The wizard will also assign the Application role to the host running Master and the Docker Daemon role to *all* the gateway hosts running Cloudera Data Science Workbench.
- Once the First Run command has completed successfully, click **Finish** to go back to the Cloudera Manager home page.

Create the Administrator Account

After your installation is complete, set up the initial administrator account. Go to the Cloudera Data Science Workbench web application at `http://cdsw.<your_domain>.com`.

You must access Cloudera Data Science Workbench from the **Cloudera Data Science Workbench Domain** configured when setting up the service, and not the hostname of the master host. Visiting the hostname of the master host will result in a 404 error.

The first account that you create becomes the site administrator. You may now use this account to create a new project and start using the workbench to run data science workloads. For a brief example, see [Getting Started with the Cloudera Data Science Workbench](#).

Next Steps

As a site administrator, you can invite new users, monitor resource utilization, secure the deployment, and upload a license key for the product. Depending on the size of your deployment, you might also want to customize how Cloudera Data Science Workbench schedules workloads on your gateway hosts. For more details on these tasks, see:

- [Site Administration](#)
- [Customize Workload Scheduling](#) on page 268
- [Security](#)

You can also start using the product by configuring your personal account and creating a new project. For a quickstart that walks you through creating and running a simple template project, see [Getting Started with Cloudera Data Science Workbench](#) on page 117. For more details on collaborating with teams, working on projects, and sharing results, see the [Managing Cloudera Data Science Workbench Users](#) on page 121.

Installing Cloudera Data Science Workbench 1.7.2 Using Packages



Note: Cloudera Data Science Workbench 1.7.1 is the next official release after Cloudera Data Science Workbench 1.6.x. Version 1.7.0 is no longer publicly available.

Use the following steps to install the latest Cloudera Data Science Workbench 1.7.2 using RPM packages.

Prerequisites

Before you begin installing Cloudera Data Science Workbench, make sure you have completed the steps to [configure your hosts and block devices](#).



Note: For proof-of-concept deployments, you can deploy a 1-host cluster with just a Master host. The Master host can run user workloads just as a worker host can when required for demonstration purposes. For production deployments, Cloudera requires to have a reserved, dedicated master host and separate worker host(s).

Configure Gateway Hosts Using Cloudera Manager

Cloudera Data Science Workbench hosts must be added to your CDH cluster as gateway hosts, with gateway roles properly configured. To configure gateway hosts:

1. If you have not already done so and plan to use PySpark, install either the [Anaconda parcel](#) or Python (versions 2.7.11 and 3.6.1) on your CDH cluster. For more information see, [Python Supported Versions](#) on page 70.
2. Configure Apache Spark on your gateway hosts.
 - a. **(CDH 5 only)** Install and configure the CDS 2.x Powered by Apache Spark parcel and CSD. For instructions, see [Installing CDS 2.x Powered by Apache Spark](#).



Important: Do not install CDS 2.x if you are using CDH 6. Spark 2 ships as part of the CDH 6 package; the add-on parcel is no longer required. To see which version of Spark 2 ships with CDH, refer the [CDH 6 Packaging documentation](#).

- b. **(Required for CDH 5 and CDH 6)** To be able to use Spark 2, each user must have their own `/home` directory in HDFS. If you sign in to Hue first, these directories will automatically be created for you. Alternatively, you can have cluster administrators create these directories.

```
hdfs dfs -mkdir /user/<username>
hdfs dfs -chown <username>:<username> /user/<username>
```

If you are using CDS 2.3 release 2 (or higher), review the associated known issues here: [CDS Powered By Apache Spark](#) on page 55.

3. Use Cloudera Manager to create add gateway hosts to your CDH cluster.
 1. Create a new [host template](#) that includes gateway roles for HDFS, YARN, and Spark 2.

Installing Cloudera Data Science Workbench 1.7.2 on CDH

(Required for CDH 6) If you want to run workloads on dataframe-based tables, such as tables from PySpark, sparklyr, SparkSQL, or Scala, you must also add the Hive gateway role to the template.

2. Use the instructions at [Adding a Host to the Cluster](#) to add gateway hosts to the cluster. Apply the template created in the previous step to these gateway hosts. If your cluster is kerberized, confirm that the [krb5.conf](#) file on your gateway hosts is correct.

4. Test Spark 2 integration on the gateway hosts.

1. SSH to a gateway host.
2. If your cluster is kerberized, run `kinit` to authenticate to the CDH cluster's Kerberos Key Distribution Center. The Kerberos ticket you create is not visible to Cloudera Data Science Workbench users.
3. Submit a test job to Spark by executing the following command:

CDH 5

```
spark2-submit --class org.apache.spark.examples.SparkPi --master yarn \  
--deploy-mode client \  
/opt/cloudera/parcels/SPARK2/lib/spark2/examples/jars/spark-example*.jar 100
```

To view a sample command, click

```
spark2-submit --class org.apache.spark.examples.SparkPi --master yarn \  
--deploy-mode client \  
/opt/cloudera/parcels/SPARK2/lib/spark2/examples/jars/spark-examples*.jar 100
```

CDH 6

```
spark-submit --class org.apache.spark.examples.SparkPi --master yarn \  
--deploy-mode client $SPARK_HOME/lib/spark-examples*.jar 100
```

To view a sample command, click

```
spark-submit --class org.apache.spark.examples.SparkPi --master yarn \  
--deploy-mode client /opt/cloudera/parcels/CDH/lib/spark/examples/jars/spark-examples*.jar 100
```

4. View the status of the job in the CLI output or in the Spark web UI to confirm that the host you want to use for the Cloudera Data Science Workbench master functions properly as a Spark gateway.

To view sample CLI output, click

```
19/02/15 09:37:39 INFO spark.SparkContext: Running Spark version 2.4.0-cdh6.1.0  
19/02/15 09:37:39 INFO spark.SparkContext: Submitted application: Spark Pi  
...  
19/02/15 09:37:40 INFO util.Utils: Successfully started service 'sparkDriver' on port 37050.  
...  
19/02/15 09:38:06 INFO scheduler.DAGScheduler: Job 0 finished: reduce at SparkPi.scala:38, took 18.659033 s
```

Install Cloudera Data Science Workbench on the Master Host

Use the following steps to install Cloudera Data Science Workbench on the master host. **Note that airgapped clusters and non-airgapped clusters use different files for installation.**

1. **Non-airgapped Installation** - Download the Cloudera Data Science Workbench repo file (`cloudera-cdsw.repo`) from the following location:

```
https://archive.cloudera.com/p/cdsw1/1.7.2/redhat7/yum/cloudera-cdsw.repo
```

Airgapped installation - For airgapped installations, download the Cloudera Data Science Workbench RPM file from the following location:

```
https://archive.cloudera.com/p/cdsw1/1.7.2/redhat7/yum/RPMS/x86_64/
```



Important: Make sure all Cloudera Data Science Workbench hosts (master and worker) are running the same version of Cloudera Data Science Workbench.

2. Skip this step for airgapped installations. Add the Cloudera Public GPG repository key. This key verifies that you are downloading genuine packages.

```
sudo rpm --import
https://archive.cloudera.com/p/cdsw1/1.7.2/redhat7/yum/RPM-GPG-KEY-cloudera
```

3. **Non-airgapped Installation** - Install the latest RPM with the following command:

```
sudo yum install cloudera-data-science-workbench
```

Airgapped Installation - Copy the RPM downloaded in the previous step to the appropriate gateway host. Then, use the complete filename to install the package. For example:


```
sudo yum install cloudera-data-science-workbench-1.7.2.12345.rpm
```

For guidance on any warnings displayed during the installation process, see [Understanding Installation Warnings](#) on page 304.

4. Edit the configuration file at `/etc/cdsw/config/cdsw.conf`. The following table lists the configuration properties that can be configured in `cdsw.conf`.

Table 2: cdsw.conf Properties

Properties	Description
Required Configuration	
DOMAIN	Wildcard DNS domain configured to point to the master host. If the wildcard DNS entries are configured as <code>cdsw.<company>.com</code> and <code>*.cdsw.<company>.com</code> , then DOMAIN should be set to <code>cdsw.<company>.com</code> . Users' browsers should then contact the Cloudera Data Science Workbench web application at <code>http://cdsw.<company>.com</code> . This domain for DNS and is unrelated to Kerberos or LDAP domains.
MASTER_IP	IPv4 address for the master host that is reachable from the worker hosts. Within an AWS VPC, MASTER_IP should be set to the internal IP address of the master host; for instance, if your hostname is <code>ip-10-251-50-12.ec2.internal</code> , set MASTER_IP to the corresponding IP address, <code>10.251.50.12</code> .
DISTRO	The Hadoop distribution installed on the cluster. Set this property to CDH.
DOCKER_BLOCK_DEVICES	Block device(s) for Docker images (space separated if there are multiple). Use the full path to specify the image(s), for instance, <code>/dev/xvde</code> .
JAVA_HOME	Path where Java is installed on the Cloudera Data Science Workbench hosts. The value for JAVA_HOME depends on whether you are using JDK or JRE. For example, if you're using JDK 1.8_162, set JAVA_HOME to <code>/usr/java/jdk1.8.0_162</code> . If you are only using JRE, set it to <code>/usr/java/jdk1.8.0_162/jre</code> .

Properties	Description
	Note that <i>Spark 2.2 (and higher) requires JDK 1.8</i> . For more details on the specific versions of Oracle JDK recommended for CDH and Cloudera Manager clusters, see the Cloudera Product Compatibility Matrix - Supported JDK Versions .
Optional Configuration	
APPLICATION_BLOCK_DEVICE	<p>(Master Host Only) Configure a block device for application state.</p> <p>If this property is left blank, the filesystem mounted at <code>/var/lib/cdsw</code> on the master host will be used to store all user data. For production deployments, Cloudera strongly recommends you use this option with a dedicated SSD block device for the <code>/var/lib/cdsw</code> mount.</p> <p><i>(Not recommended)</i> If set, Cloudera Data Science Workbench will format the provided block device as <code>ext4</code>, mount it to <code>/var/lib/cdsw</code>, and store all user data on it. This option has only been provided for proof-of-concept setups, and Cloudera is not responsible for any data loss.</p> <p>Use the full path to specify the mount point, for instance, <code>/dev/xvdf</code>.</p>
RESERVE_MASTER	<p>Set this property to <code>true</code> to reserve the master host for Cloudera Data Science Workbench's internal components and services such as Livelog, the PostgreSQL database, and so on. User workloads will now run exclusively on worker hosts, while the master is reserved for internal application services.</p> <p>Note that this property is not yet available as a configuration property in Cloudera Manager. However, you can use an Advanced Configuration Snippet (Safety Valve) to configure this as described here: Reserving the Master Host for Internal CDSW Components on page 269.</p> <div style="border: 1px solid orange; padding: 10px; margin-top: 10px;"> <p> Important: This feature only applies to deployments with more than one Cloudera Data Science Workbench host. Enabling this feature on single-host deployments will leave Cloudera Data Science Workbench incapable of scheduling any workloads.</p> </div>
DISTRO_DIR	Path where the Hadoop distribution is installed on the Cloudera Data Science Workbench hosts. For CDH clusters, the default location of the parcel directory is <code>/opt/cloudera/parcels</code> . Specify this property only if you are using a non-default location.
ANACONDA_DIR	Path where the Anaconda package is installed. On CDH clusters, Anaconda is installed as a parcel in Cloudera Manager. Therefore, this parameter does not apply and must be left blank.
TLS_ENABLE	<p>Enable and enforce HTTPS (TLS/SSL) for web access.</p> <p>Set to <code>true</code> to enable and enforce HTTPS access to the web application.</p> <p>You can also set this property to <code>true</code> to enable external TLS termination. For more details on TLS termination, see Enabling TLS/SSL for Cloudera Data Science Workbench on page 285.</p>
TLS_CERT	Certificate and private key for internal TLS termination.
TLS_KEY	Setting <code>TLS_CERT</code> and <code>TLS_KEY</code> will enable internal TLS termination. You must also set <code>TLS_ENABLE</code> to <code>true</code> above to enable and enforce internal termination. Set these only if you are not terminating TLS externally.

Properties	Description
	<p>Make sure you specify the full path to the certificate and key files, which must be in PEM format.</p> <p>For details on certificate requirements and enabling TLS termination, see Enabling TLS/SSL for Cloudera Data Science Workbench on page 285.</p>
TLS_ROOTCA	<p>If <code>TLS_CERT</code> is signed by a non-public or Internal Custom Certificate Authority, set this to a <code>.pem</code> file containing the root certificate (trust chain) for that certificate authority.</p> <p>The contents of this field are then inserted into the engine's root certificate store every time a session (or any workload) is launched. This allows processes inside the engine to communicate securely with the ingress controller.</p>
HTTP_PROXY HTTPS_PROXY	<p>If your deployment is behind an HTTP or HTTPS proxy, set the respective <code>HTTP_PROXY</code> or <code>HTTPS_PROXY</code> property in <code>/etc/cdsw/config/cdsw.conf</code> to the hostname of the proxy you are using.</p> <pre>HTTP_PROXY="<http://proxy_host>:<proxy-port>" HTTPS_PROXY="<http://proxy_host>:<proxy-port>"</pre> <p>If you are using an intermediate proxy such as Cntlm to handle NTLM authentication, add the Cntlm proxy address to the <code>HTTP_PROXY</code> or <code>HTTPS_PROXY</code> fields in <code>cdsw.conf</code>.</p> <pre>HTTP_PROXY="http://localhost:3128" HTTPS_PROXY="http://localhost:3128"</pre> <p>If the proxy server uses TLS encryption to handle connection requests, you will need to add the proxy's root CA certificate to your host's store of trusted certificates. This is because proxy servers typically sign their server certificate with their own root certificate. Therefore, any connection attempts will fail until the Cloudera Data Science Workbench host trusts the proxy's root CA certificate. If you do not have access to your proxy's root certificate, contact your Network / IT administrator.</p> <p>To enable trust, copy the proxy's root certificate to the trusted CA certificate store (<code>ca-trust</code>) on the Cloudera Data Science Workbench host.</p> <pre>cp /tmp/<proxy-root-certificate>.cert /etc/pki/ca-trust/source/anchors/</pre> <p>Use the following command to rebuild the trusted certificate store.</p> <pre>update-ca-trust extract</pre>
ALL_PROXY	<p>If a SOCKS proxy is in use, set to <code>socks5://<host>:<port>/</code>.</p>
NO_PROXY	<p>Comma-separated list of hostnames that should be skipped from the proxy.</p> <p>Starting with version 1.4, if you have defined a proxy in the <code>HTTP_PROXY(S)</code> or <code>ALL_PROXY</code> properties, Cloudera Data Science Workbench automatically appends the following list of IP addresses to the <code>NO_PROXY</code> configuration. Note that this is the minimum required configuration for this field.</p>


Properties	Description
	<p>This list includes 127.0.0.1, localhost, and any private Docker registries and HTTP services inside the firewall that Cloudera Data Science Workbench users might want to access from the engines.</p> <pre>"127.0.0.1,localhost,100.66.0.1,100.66.0.2,100.66.0.3,100.66.0.4,100.66.0.5,100.66.0.6,100.66.0.7,100.66.0.8,100.66.0.9,100.66.0.10,100.66.0.11,100.66.0.12,100.66.0.13,100.66.0.14,100.66.0.15,100.66.0.16,100.66.0.17,100.66.0.18,100.66.0.19,100.66.0.20,100.66.0.21,100.66.0.22,100.66.0.23,100.66.0.24,100.66.0.25,100.66.0.26,100.66.0.27,100.66.0.28,100.66.0.29,100.66.0.30,100.66.0.31,100.66.0.32,100.66.0.33,100.66.0.34,100.66.0.35,100.66.0.36,100.66.0.37,100.66.0.38,100.66.0.39,100.66.0.40,100.66.0.41,100.66.0.42,100.66.0.43,100.66.0.44,100.66.0.45,100.66.0.46,100.66.0.47,100.66.0.48,100.66.0.49,100.66.0.50,100.77.0.10,100.77.0.128,100.77.0.129,100.77.0.130,100.77.0.131,100.77.0.132,100.77.0.133,100.77.0.134,100.77.0.135,100.77.0.136,100.77.0.137,100.77.0.138,100.77.0.139"</pre>
NVIDIA_GPU_ENABLE	<p>Set this property to <code>true</code> to enable GPU support for Cloudera Data Science Workbench workloads. When this property is enabled on a host is equipped with GPU hardware, the GPU(s) will be available for use by Cloudera Data Science Workbench hosts.</p> <p>If this property is set to <code>true</code> on a host that does not have GPU support, there will be no effect. By default, this property is set to <code>false</code>.</p> <p>For detailed instructions on how to enable GPU-based workloads on Cloudera Data Science Workbench, see Using NVIDIA GPUs for Cloudera Data Science Workbench Projects on page 137.</p>

5. Initialize and start Cloudera Data Science Workbench.

```
cdsw start
```

The application will take a few minutes to bootstrap. You can watch the status of application installation and startup with `watch cdsw status`.

(Optional) Install Cloudera Data Science Workbench on Worker Hosts

 **Note:** For proof-of-concept deployments, you can deploy a 1-host cluster with just a Master host. The Master host can run user workloads just as a worker host can when required for demonstration purposes. For production deployments, Cloudera requires to have a reserved, dedicated master host and separate worker host(s).

Cloudera Data Science Workbench supports adding and removing additional worker hosts at any time. Worker hosts allow you to transparently scale the number of concurrent workloads users can run.

Use the following steps to add worker hosts to Cloudera Data Science Workbench. **Note that airgapped clusters and non-airgapped clusters use different files for installation.**

- 1. Non air-gapped Installation** - Download the Cloudera Data Science Workbench repo file (`cloudera-cdsw.repo`) from the following location:

```
https://archive.cloudera.com/p/cdsw1/1.7.2/redhat7/yum/cloudera-cdsw.repo
```

Airgapped installation - For airgapped installations, download the Cloudera Data Science Workbench RPM file from the following location:

```
https://archive.cloudera.com/p/cdsw1/1.7.2/redhat7/yum/RPMS/x86_64/
```




Important: Make sure all Cloudera Data Science Workbench hosts (master and worker) are running the same version of Cloudera Data Science Workbench.

2. Skip this step for airgapped installations. Add the Cloudera Public GPG repository key. This key verifies that you are downloading genuine packages.

```
sudo rpm --import
https://archive.cloudera.com/p/cdsw1/1.7.2/redhat7/yum/RPM-GPG-KEY-cloudera
```

3. **Non-airgapped Installation** - Install the latest RPM with the following command:

```
sudo yum install cloudera-data-science-workbench
```

Airgapped Installation - Copy the RPM downloaded in the previous step to the appropriate gateway host. Then, use the complete filename to install the package. For example:

```
sudo yum install cloudera-data-science-workbench-1.7.2.12345.rpm
```

For guidance on any warnings displayed during the installation process, see [Understanding Installation Warnings](#) on page 304.

4. Copy `cdsw.conf` file from the master host:

```
scp root@cdsw-host-1.<company>.com:/etc/cdsw/config/cdsw.conf /etc/cdsw/config/cdsw.conf
```

After initialization, the `cdsw.conf` file includes a generated bootstrap token that allows worker hosts to securely join the cluster. You can get this token by copying the configuration file from master and ensuring it has 600 permissions.

If your hosts have heterogeneous block device configurations, modify the Docker block device settings in the worker host configuration file after you copy it. Worker hosts do not need application block devices, which store the project files and database state, and this configuration option is ignored.

5. Create `/var/lib/cdsw` on the worker host. This directory must exist on all worker hosts. Without it, the next step that registers the worker host with the master will fail.

Unlike the master host, the `/var/lib/cdsw` directory on worker hosts does **not** need to be mounted to an Application Block Device. It is only used to store CDH client configuration on workers.

6. On the *worker* host, run the following command to add the host to the cluster:

```
cdsw join
```

This causes the worker hosts to register themselves with the Cloudera Data Science Workbench master host and increase the available pool of resources for workloads.

7. Return to the master host and verify the host is registered with this command:

```
cdsw status
```

Create the Administrator Account

Installation typically takes 30 minutes, although it might take an additional 60 minutes for the R, Python, and Scala engine to be available on all hosts.

After your installation is complete, set up the initial administrator account. Go to the Cloudera Data Science Workbench web application at `http://cdsw.<company>.com`.



Note: You must access Cloudera Data Science Workbench from the `DOMAIN` configured in `cdsw.conf`, and not the hostname of the master host. Visiting the hostname of the master host will result in a 404 error.

The first account that you create becomes the site administrator. You may now use this account to create a new project and start using the workbench to run data science workloads. For a brief example, see [Getting Started with the Cloudera Data Science Workbench](#).

Next Steps

As a site administrator, you can invite new users, monitor resource utilization, secure the deployment, and upload a license key for the product. Depending on the size of your deployment, you might also want to customize how Cloudera Data Science Workbench schedules workloads on your gateway hosts. For more details on these tasks, see:

- [Site Administration](#)
- [Customize Workload Scheduling](#) on page 268
- [Security](#)

You can also start using the product by configuring your personal account and creating a new project. For a quickstart that walks you through creating a simple template project, see [Getting Started with Cloudera Data Science Workbench](#) on page 117. For more details on collaborating with teams, working on projects, and sharing results, see the [Managing Cloudera Data Science Workbench Users](#) on page 121.

Upgrading to the Latest Version of Cloudera Data Science Workbench 1.7.2 on CDH

This topic walks you through the upgrade paths available for Cloudera Data Science Workbench 1.7.2. Depending on your existing deployment, choose from one of the upgrade paths listed in the following table.



Note: As of February 1, 2021, all downloads of CDH and Cloudera Manager require a username and password and use a modified URL. You must use the modified URL, including the username and password when downloading the repository contents described below. You may need to upgrade Cloudera Manager to a newer version that uses the modified URLs.

This can affect new installations, upgrades, adding new hosts to a cluster, and adding a cluster.

For more information, see *Updating an existing CDH/Cloudera Manager deployment to access downloads with authentication* for [CDH5](#) or [CDH6](#).

For installation and upgrades, you must manually add the Remote Parcel Repository URLs for your CDSW version to Cloudera Manager.

Upgrading from CDH 5 > CDH 6 - If you are currently running Cloudera Data Science Workbench on CDH 5 and want to upgrade your cluster to CDH 6, also see [Upgrading Cloudera Data Science Workbench from CDH 5 to CDH 6](#) on page 91 or <https://docs.cloudera.com/documentation/enterprise/release-notes/topics/cm-retrofit-auth-downloads.html>.

Upgrade Path	Link to Instructions
CSD > CSD Upgrading from an existing CSD-based deployment to the latest 1.7.2 CSD and parcel.	Upgrading Cloudera Data Science Workbench 1.7.2 Using Cloudera Manager on page 93
RPM > CSD Migrating from an RPM-based deployment to the latest 1.7.2 CSD and parcel-based deployment.	Migrating from an RPM-based Deployment to the Latest 1.7.2 CSD on page 97
RPM > RPM Upgrading an existing RPM-based deployment to the latest 1.7.2 RPM.	Upgrading Cloudera Data Science Workbench 1.7.2 Using Packages on page 101 Note that you cannot use Cloudera Manager for this upgrade path.

Upgrading Cloudera Data Science Workbench from CDH 5 to CDH 6

This section provides a general outline on how to go about upgrading your Cloudera Data Science Workbench cluster from CDH 5 to CDH 6. Refer the relevant linked Cloudera Manager and CDH upgrade documentation for the detailed steps required for this procedure.

Upgrading CSD Deployments from CDH 5 to CDH 6

Starting with version 1.5, Cloudera Data Science Workbench publishes two separate CSD files: one for CDH 5 and one for CDH 6. Check the CSD file name to ensure that you are using the correct CSD file for your cluster. For example:

- **CDH 6** - CLOUDERA_DATA_SCIENCE_WORKBENCH_CDH6_1.x.y.jar
- **CDH 5** - CLOUDERA_DATA_SCIENCE_WORKBENCH_CDH5_1.x.y.jar

Upgrading to the Latest Version of Cloudera Data Science Workbench 1.7.2 on CDH

Use the following path to upgrade from running a CSD-based Cloudera Data Science Workbench deployment on CDH 5 to running on CDH 6:

1. [Upgrade to Cloudera Manager 6.1 \(or higher\)](#).
2. Stop Cloudera Data Science Workbench.
3. Remove both Spark gateway roles from all CDSW hosts.
4. Delete the `/etc/spark` and `/etc/spark2` directories.
5. Download **both** of the CDSW Cloudera Data Science Workbench CSD files for the latest version. For example:
 - CDSW1.8-CDH6..jar
 - CDSW1.8-CDH5..jar

At this point, you should have three CSV files. One original file (for example, CDSW1.5-CDH5..jar) and two new files (for example, CDSW1.8-CDH6..jar and CDSW1.8-CDH5..jar).

6. Log on to the Cloudera Manager Server host, and place the new CDSW files under `/opt/cloudera/csd`, which is the default location for CSDs.
7. Restart the Cloudera Manager Server.
8. [Upgrade to Cloudera Data Science Workbench 1.5 \(or higher\)](#). During the upgrade process, as you install, distribute, and activate the new parcel, take care to ensure that both the CDSW CSDs (for CDH 5 and CDH 6) are present on the Cloudera Manager Server host.
9. [Use the Cloudera Manager Upgrade Wizard to upgrade from CDH 5 to CDH 6.1 \(or higher\)](#). As part of the upgrade, the wizard will also remove the Spark 2 parcel from all your cluster hosts. With CDH 6, Spark 2 ships as a part of CDH. The add-on parcel is no longer required.

Cloudera Manager 6 can differentiate between the two active CSDs and will select the right one based on the version of CDH running. Because you already have the CDH 6-compatible CSD installed, no further steps are needed.

- 10 (Optional) Remove any existing CDH 5 CSDs from the Cloudera Manager Server host.
- 11 Add the Spark gateway back in.
- 12 Redeploy the client configurations.
- 13 Restart Cloudera Data Science Workbench.

Upgrading RPM Deployments from CDH 5 to CDH 6

Cloudera Data Science Workbench ships a single RPM that can be used to install CDSW on both, CDH 5, and CDH 6 clusters. The upgrade path for RPM deployments is:

1. [Upgrade to Cloudera Manager 6.1 \(or higher\)](#).
2. [Use the Cloudera Manager Upgrade Wizard to upgrade from CDH 5 to CDH 6 \(or higher\)](#). As part of the upgrade, the wizard will also remove the Spark 2 parcel from all your cluster hosts. This is because with CDH 6, Spark 2 ships as a part of CDH. The add-on parcel is no longer required.
3. [Upgrade to the latest Cloudera Data Science Workbench RPM](#).

Upgrading RPM Deployments from CDH 5 to CDH 6

Cloudera Data Science Workbench ships a single RPM that can be used to install CDSW on both, CDH 5, and CDH 6 clusters. The upgrade path for RPM deployments is:

1. [Upgrade to Cloudera Manager 6.1 \(or higher\)](#).
2. [Use the Cloudera Manager Upgrade Wizard to upgrade from CDH 5 to CDH 6 \(or higher\)](#). As part of the upgrade, the wizard will also remove the Spark 2 parcel from all your cluster hosts. This is because with CDH 6, Spark 2 ships as a part of CDH. The add-on parcel is no longer required.

3. [Upgrade to the latest Cloudera Data Science Workbench RPM.](#)

Upgrading Cloudera Data Science Workbench 1.7.2 or higher from CDH 6 to CDP-DC 7.x

The following general outline describes how to upgrade your Cloudera Data Science Workbench cluster from CDH 6 to CDP-DC. Refer to the relevant linked Cloudera Manager and CDH upgrade documentation for detailed steps required for this procedure.

Before upgrading your CDSW deployment, do the following:

1. Migrate CDSW from RPM to CSD: see the instructions to [migrate to CSD files](#).
2. Download the correct CDSW CSD files for CDP-DC. The correct CSD file names are:
 - For CDH 6: `CLOUDERA_DATA_SCIENCE_WORKBENCH_CDH6_1.x.y.jar`
 - For CDP-DC: `CLOUDERA_DATA_SCIENCE_WORKBENCH_CDPDC_1.x.y.jar`

Upgrade cluster from CDH 6 to CDP-DC 7.x:

1. Stop Cloudera Data Science Workbench.
2. Log on to the Cloudera Manager Server host.
3. Place the CSD files in `/opt/cloudera/csd`. This is the default location for CSD files.
4. Restart Cloudera Manager Server.
5. [Upgrade to Cloudera Data Science Workbench 1.7.2 \(or higher\)](#). During the upgrade process, as you install, distribute, and activate the new parcel, take care to ensure that both CDSW CSD files (for CDH 6 and CDP-DC) are present on the Cloudera Manager Server host.
6. Upgrade CDP-DC based on the CDP-DC [upgrade documentation](#).
7. Remove any existing CDH 6 CSDs from the Cloudera Manager Server host. This step is optional.
8. Restart Cloudera Data Science Workbench.

Upgrading Cloudera Data Science Workbench 1.7.2 Using Cloudera Manager



Important:

- Cloudera Data Science Workbench only supports upgrades to version 1.7.x from version 1.5.x and 1.6.x. If you are using a lower version, you must first upgrade to version 1.5.x or 1.6.x, and then upgrade to version 1.7.x.
- Cloudera Data Science Workbench 1.7.1 is the next official release after Cloudera Data Science Workbench 1.6.x. Version 1.7.0 is no longer publicly available.

This topic describes how to upgrade a CSD and parcel-based deployment to the latest version of Cloudera Data Science Workbench 1.7.2.

1. Before you begin the upgrade process, make sure you read the [Cloudera Data Science Workbench Release Notes](#) on page 21 relevant to the version you are upgrading to/from.
2. [Stop the Cloudera Data Science Workbench service in Cloudera Manager.](#)
3. **(Upgrading from CDSW 1.7.1 with patch)** Perform this step only if you are upgrading from CDSW 1.7.1 and applied the patches documented here: [CDSW restart issue on multi-node deployments; CDSW Web UI does not automatically come up after upgrading to CDSW 1.7.1](#) on page 52
 - a. Delete the 2 patch files: `/etc/cdsw/patches/default/deployment/ingress-controller.yaml` and `/etc/cdsw/patches/default/deployment/tcp-ingress-controller.yaml`.
 - b. Delete every empty folder from the `/etc/cdsw/patches` directory.

c. Delete the `/etc/cdsw/patches` directory if it is empty.

4. **(Strongly Recommended)** On the master host, backup all your application data that is stored in the `/var/lib/cdsw` directory.

To create the backup, run the following command on the *master* host:

```
tar -cvzf cdsw.tar.gz -C /var/lib/cdsw/ .
```

5. Deactivate the existing Cloudera Data Science Workbench parcel. Go to the Cloudera Manager Admin Console. In the top navigation bar, click **Hosts** > **Parcels**.

Locate the current active CDSW parcel and click **Deactivate**. On the confirmation pop-up, select **Deactivate Only** and click **OK**.

6. Download and save the latest Cloudera Data Science Workbench CSD to the Cloudera Manager Server host.

a. Download the Cloudera Data Science Workbench CSD. Make sure you download the CSD that corresponds to the version of CDH or Cloudera Runtime you are using.

- **CDP Data Center**

```
https://archive.cloudera.com/p/cdsw1/1.7.2/csd/CLOUDERA_DATA_SCIENCE_WORKBENCH-CDPDC-1.7.2.jar
```

OR

- **CDH 6**

```
https://archive.cloudera.com/p/cdsw1/1.7.2/csd/CLOUDERA_DATA_SCIENCE_WORKBENCH-CDH6-1.7.2.jar
```

OR

- **CDH 5**

```
https://archive.cloudera.com/p/cdsw1/1.7.2/csd/CLOUDERA_DATA_SCIENCE_WORKBENCH-CDH5-1.7.2.jar
```

b. Log on to the Cloudera Manager Server host, and place the CSD file under `/opt/cloudera/csd`, which is the default location for CSD files.

c. Delete any CSD files belonging to older versions of Cloudera Data Science Workbench from `/opt/cloudera/csd`.

This is required because older versions of the CSD will not work with the latest Cloudera Data Science Workbench parcel. Make sure your CSD and parcel are always the same version.

After you delete the file(s) belonging to the older version, you should still have two files for the current version; one for CDH5 (for example, CDSW1.8-CDH5..jar) and one for CDH6 (for example, CDSW1.8-CDH6..jar). You'll need both of these files to upgrade.



Note: If you have previously configured a custom location for CSD files, place the CSD file there, and delete any CSDs belonging to older versions of Cloudera Data Science Workbench. For help, refer the Cloudera Manager documentation at [Configuring the Location of Custom Service Descriptor Files](#).

d. Set the CSD file ownership to `cloudera-scm:cloudera-scm` with permission 644.

e. Restart the Cloudera Manager Server:

```
service cloudera-scm-server restart
```

f. Log into the Cloudera Manager Admin Console and restart the Cloudera Management Service.

a. Select **Clusters** > **Cloudera Management Service**.

b. Select Actions > Restart.

7. Distribute and activate the new parcel on your cluster.

- a. Log into the Cloudera Manager Admin Console.
- b. Click **Hosts > Parcels** in the main navigation bar.
- c. *If the latest CDSW parcel is already available in this list, you can skip this step.*

Add the Cloudera Data Science Workbench parcel repository URL to Cloudera Manager.

- 1. On the **Parcels** page, click **Configuration**.
- 2. In the **Remote Parcel Repository URLs** list, click the addition symbol to create a new row.
- 3. Enter the path to the repository.

Version	Remote Parcel Repository URL
Cloudera Data Science Workbench 1.7.2	<code>https://archive.cloudera.com/p/cdsw1/1.7.2/parcels/</code>
Cloudera Data Science Workbench 1.7.1	<code>https://archive.cloudera.com/p/cdsw1/1.7.1/parcels/</code>

4. Click Save Changes.

- d. Go back to the **Hosts > Parcels** page. The latest parcel should now appear in the set of parcels available for download. Click **Download**. Once the download is complete, click **Distribute** to distribute the parcel to all the CDH hosts in your cluster. Then click **Activate**. For more detailed information on each of these tasks, see [Managing Parcels](#).

8. Run the Prepare Node command on all Cloudera Data Science Workbench hosts.

- 1. Before you run **Prepare Node**, you must make sure that the command is allowed to install all the required packages on your cluster. This is controlled by the **Install Required Packages** property.
 - 1. Navigate to the CDSW service.
 - 2. Click **Configuration**.
 - 3. Search for the **Install Required Packages** property. If this property is enabled, you can move on to the next step and run **Prepare Node**.

However, if the property has been disabled, you must either enable it or manually install the following packages on *all* Cloudera Data Science Workbench gateway hosts.

```
nfs-utils
libseccomp
lvm2
bridge-utils
libtool-ltdl
iptables
rsync
policycoreutils-python
selinux-policy-base
selinux-policy-targeted
ntp
e2fsprogs
redhat-lsb-core
conntrack-tools
socat
```

2. Run the **Prepare Node** command.

1. In Cloudera Manager, navigate to the Cloudera Data Science Workbench service.
2. Click the **Instances** tab.
3. Use the checkboxes to select all host instances and click **Actions for Selected (x)**.
4. Click **Prepare Node**. Once again, click **Prepare Node** to confirm the action.

9. Log into the Cloudera Manager Admin Console and restart the Cloudera Data Science Workbench service.

a. On the **Home > Status** tab, click



to the right of the **CDSW** service and select **Restart** from the dropdown.

- #### b. Confirm your choice on the next screen. Note that a complete restart of the service will take time. Even though the CDSW service status shows **Good Health**, the application itself will take some more time to get ready.

10 Upgrade Projects to Use the Latest Base Engine Images

If the release you have just upgraded to includes a new version of the base engine image (see [release notes](#)), you will need to manually configure existing projects to use the new engine. Cloudera recommends you do so to take advantage of any new features and bug fixes included in the newly released engine. For example:

• **Container Security**

Security best practices dictate that engine containers should not run as the root user. Engines (v7 and lower) briefly initialize as the `root` user and then run as the `cdsw` user. Engines v8 (and higher) now follow the best practice and run only as the `cdsw` user. For more details, see [Restricting User-Created Pods](#).

• **CDH 6 Compatibility**

The base engine image you use must be compatible with the version of CDH you are running. This is especially important if you are running workloads on Spark. Older base engines (v6 and lower) cannot support the latest versions of CDH 6. If you want to run Spark workloads on CDH 6, you must upgrade your projects to base engine 7 (or higher).

• **Editors**

Engines v8 (and higher) ships with the browser-based IDE, Jupyter, preconfigured and can be selected from the **Start Session** menu.

To upgrade a project to the new engine, go to the project's **Settings > Engine** page and select the new engine from the dropdown. If any of your projects are using custom [extended engines](#), you will need to modify them to use the new base engine image.

11 (GPU-enabled Deployments) Remove `nvidia-docker1` and Upgrade NVIDIA Drivers to 410.xx or higher

Perform the following steps to make sure you can continue to leverage GPUs for workloads on Cloudera Data Science Workbench 1.6 (and higher).

1. Remove `nvidia-docker1`. Cloudera Data Science Workbench (version 1.6 and higher) ships with `nvidia-docker2` installed by default.

```
yum remove nvidia-docker
```

Perform this step on *all* hosts that have GPUs attached to them.

2. Upgrade your NVIDIA driver to version 410.xx (or higher). This must be done because `nvidia-docker2` does not support lower versions of NVIDIA drivers.
 - a. Stop Cloudera Data Science Workbench.

Depending on your deployment, either stop the CDSW service in Cloudera Manager (for CSDs) or run `cdsw stop` on the Master host (for RPMs).

- b. Reboot the GPU-enabled hosts.
- c. Install a supported version of the NVIDIA driver (410.xx or higher) on all GPU-enabled hosts.
- d. Start Cloudera Data Science Workbench.

Depending on your deployment, either start the CDSW service in Cloudera Manager (for CSDs) or run `cdsw start` on the Master host (for RPMs).

Migrating from an RPM-based Deployment to the Latest 1.7.2 CSD



Important:

- Cloudera Data Science Workbench only supports upgrades to version 1.7.x from version 1.5.x and 1.6.x. If you are using a lower version, you must first upgrade to version 1.5.x or 1.6.x, and then upgrade to version 1.7.x.
- Cloudera Data Science Workbench 1.7.1 is the next official release after Cloudera Data Science Workbench 1.6.x. Version 1.7.0 is no longer publicly available.

This topic describes how to migrate from an RPM-based deployment to the latest 1.7.2 CSD and parcel-based deployment.

1. Before you begin the migration process, make sure you read the [Cloudera Data Science Workbench Release Notes](#) on page 21 relevant to the version you are migrating to/from.
2. Save a backup of the Cloudera Data Science Workbench configuration file located at `/etc/cdsw/config/cdsw.conf`.
3. [Stop the Cloudera Data Science Workbench service in Cloudera Manager](#).
4. **(Upgrading from CDSW 1.7.1 with patch)** Perform this step only if you are upgrading from CDSW 1.7.1 and applied the patches documented here: [CDSW restart issue on multi-node deployments; CDSW Web UI does not automatically come up after upgrading to CDSW 1.7.1](#) on page 52
 - a. Delete the 2 patch files: `/etc/cdsw/patches/default/deployment/ingress-controller.yaml` and `/etc/cdsw/patches/default/deployment/tcp-ingress-controller.yaml`.
 - b. Delete every empty folder from the `/etc/cdsw/patches` directory.
 - c. Delete the `/etc/cdsw/patches` directory if it is empty.
5. **(Strongly Recommended)** On the master host, backup all your application data that is stored in the `/var/lib/cdsw` directory.

To create the backup, run the following command on the *master* host:

```
tar -cvzf cdsw.tar.gz -C /var/lib/cdsw/ .
```


6. Save a backup of the Cloudera Data Science Workbench configuration file at `/etc/cdsw/config/cdsw.conf`.
7. Uninstall the previous release of Cloudera Data Science Workbench. Perform this step on the master host, as well as all the worker hosts.

```
yum remove cloudera-data-science-workbench
```

8. Install the latest version of Cloudera Data Science Workbench using the CSD and parcel. Note that when you are configuring role assignments for the Cloudera Data Science Workbench service, **the Master role must be assigned to the same host that was running as master prior to the upgrade**.

For installation instructions, see [Installing Cloudera Data Science Workbench 1.7.2 Using Cloudera Manager](#) on page 75. You might be able to skip the first few steps assuming you have the wildcard DNS domain and block devices already set up.

9. Use your copy of the backup `cdsw.conf` created in Step 3 to recreate those settings in Cloudera Manager by configuring the corresponding properties under the Cloudera Data Science Workbench service.
 - a. Log into the Cloudera Manager Admin Console.
 - b. Go to the Cloudera Data Science Workbench service.
 - c. Click the **Configuration** tab.
 - d. The following table lists all the `cdsw.conf` properties and their corresponding Cloudera Manager properties (in bold). Use the search box to bring up the properties you want to modify.
 - e. Click **Save Changes**.

cdsw.conf Property	Corresponding Cloudera Manager Property and Description
TLS_ENABLE	<p>Enable TLS: Enable and enforce HTTPS (TLS/SSL) access to the web application (optional). Both internal and external termination are supported. To enable internal termination, you must also set the TLS Certificate for Internal Termination and TLS Key for Internal Termination parameters. If these parameters are not set, terminate TLS using an external proxy.</p> <p>For more details on TLS termination, see Enabling TLS/SSL for Cloudera Data Science Workbench on page 285.</p>
TLS_CERT TLS_KEY	<p>TLS Certificate for Internal Termination, TLS Key for Internal Termination</p> <p>Complete path to the certificate and private key (in PEM format) to be used for internal TLS termination. Set these parameters only if you are not terminating TLS externally. You must also set the Enable TLS property to enable and enforce termination. The certificate must include both <code>DOMAIN</code> and <code>*.DOMAIN</code> as hostnames.</p> <p>Self-signed certificates are not supported unless trusted fully by clients. Accepting an invalid certificate manually can cause connection failures for unknown subdomains. Set these only if you are not terminating TLS externally. For details on certificate requirements and enabling TLS termination, see Enabling TLS/SSL for Cloudera Data Science Workbench on page 285.</p>
TLS_ROOTCA	<div style="border: 1px solid #ccc; padding: 10px; margin-bottom: 10px;"> <p> Note: This property is not available in Cloudera Manager. It must be set in the Cloudera Data Science Workbench Site Administration panel after installation is complete. For instructions, see Configuring Custom Root CA Certificate on page 289.</p> </div> <p>If your organization uses an internal custom Certificate Authority, you can use this field to paste in the contents of your internal CA's root certificate file.</p> <p>The contents of this field are then inserted into the engine's root certificate store every time a session (or any workload) is launched. This allows processes inside the engine to communicate securely with the ingress controller.</p>
HTTP_PROXY HTTPS_PROXY	<p>HTTP Proxy, HTTPS Proxy</p> <p>If your deployment is behind an HTTP or HTTPS proxy, set the respective HTTP Proxy or HTTPS Proxy property to the hostname of the proxy you are using.</p> <div style="border: 1px dashed #ccc; padding: 5px; margin-top: 10px;"> <pre>http://<proxy_host>:<proxy-port></pre> </div>

cdsw.conf Property	Corresponding Cloudera Manager Property and Description
	<p><i>or</i></p> <pre>https://<proxy_host>:<proxy_port></pre> <p>If you are using an intermediate proxy such as Cntlm to handle NTLM authentication, add the Cntlm proxy address to the HTTP Proxy or HTTPS Proxy fields. That is, either <code>http://localhost:3128</code> or <code>https://localhost:3128</code> respectively.</p> <p>If the proxy server uses TLS encryption to handle connection requests, you will need to add the proxy's root CA certificate to your host's store of trusted certificates. This is because proxy servers typically sign their server certificate with their own root certificate. Therefore, any connection attempts will fail until the Cloudera Data Science Workbench host trusts the proxy's root CA certificate. If you do not have access to your proxy's root certificate, contact your Network / IT administrator.</p> <p>To enable trust, copy the proxy's root certificate to the trusted CA certificate store (<code>ca-trust</code>) on the Cloudera Data Science Workbench host.</p> <pre>cp /tmp/<proxy-root-certificate>.cert /etc/pki/ca-trust/source/anchors/</pre> <p>Use the following command to rebuild the trusted certificate store.</p> <pre>update-ca-trust extract</pre>
ALL_PROXY	<p>SOCKS Proxy: If a SOCKS proxy is in use, set this parameter to <code>socks5://<host>:<port>/</code>.</p>
NO_PROXY	<p>No Proxy: Comma-separated list of hostnames that should be skipped from the proxy.</p> <p>Starting with version 1.4, if you have defined a proxy in the <code>HTTP_PROXY(S)</code> or <code>ALL_PROXY</code> properties, Cloudera Data Science Workbench automatically appends the following list of IP addresses to the <code>NO_PROXY</code> configuration. Note that this is the minimum required configuration for this field.</p> <p>This list includes <code>127.0.0.1</code>, <code>localhost</code>, and any private Docker registries and HTTP services inside the firewall that Cloudera Data Science Workbench users might want to access from the engines.</p> <pre>"127.0.0.1,localhost,100.66.0.1,100.66.0.2,100.66.0.3,100.66.0.4,100.66.0.5,100.66.0.6,100.66.0.7,100.66.0.8,100.66.0.9,100.66.0.10,100.66.0.11,100.66.0.12,100.66.0.13,100.66.0.14,100.66.0.15,100.66.0.16,100.66.0.17,100.66.0.18,100.66.0.19,100.66.0.20,100.66.0.21,100.66.0.22,100.66.0.23,100.66.0.24,100.66.0.25,100.66.0.26,100.66.0.27,100.66.0.28,100.66.0.29,100.66.0.30,100.66.0.31,100.66.0.32,100.66.0.33,100.66.0.34,100.66.0.35,100.66.0.36,100.66.0.37,100.66.0.38,100.66.0.39,100.66.0.40,100.66.0.41,100.66.0.42,100.66.0.43,100.66.0.44,100.66.0.45,100.66.0.46,100.66.0.47,100.66.0.48,100.66.0.49,100.66.0.50,100.77.0.10,100.77.0.128,100.77.0.129,100.77.0.130,100.77.0.131,100.77.0.132,100.77.0.133,100.77.0.134,100.77.0.135,100.77.0.136,100.77.0.137,100.77.0.138,100.77.0.139"</pre>
NVIDIA_GPU_ENABLE	<p>Enable GPU Support: When this property is enabled, GPUs installed on Cloudera Data Science Workbench hosts will be available for use in its workloads. By default, this parameter is disabled.</p> <p>For instructions on how to enable GPU-based workloads on Cloudera Data Science Workbench, see Using NVIDIA GPUs for Cloudera Data Science Workbench Projects on page 137.</p>

10 Cloudera Manager will prompt you to restart the service if needed.

11 Upgrade Projects to Use the Latest Base Engine Images

If the release you have just upgraded to includes a new version of the base engine image (see [release notes](#)), you will need to manually configure existing projects to use the new engine. Cloudera recommends you do so to take advantage of any new features and bug fixes included in the newly released engine. For example:

- **Container Security**

Security best practices dictate that engine containers should not run as the root user. Engines (v7 and lower) briefly initialize as the `root` user and then run as the `cdsw` user. Engines v8 (and higher) now follow the best practice and run only as the `cdsw` user. For more details, see [Restricting User-Created Pods](#).

- **CDH 6 Compatibility**

The base engine image you use must be compatible with the version of CDH you are running. This is especially important if you are running workloads on Spark. Older base engines (v6 and lower) cannot support the latest versions of CDH 6. If you want to run Spark workloads on CDH 6, you must upgrade your projects to base engine 7 (or higher).

- **Editors**

Engines v8 (and higher) ships with the browser-based IDE, Jupyter, preconfigured and can be selected from the **Start Session** menu.

To upgrade a project to the new engine, go to the project's **Settings > Engine** page and select the new engine from the dropdown. If any of your projects are using custom [extended engines](#), you will need to modify them to use the new base engine image.

12 (GPU-enabled Deployments) Remove nvidia-docker1 and Upgrade NVIDIA Drivers to 410.xx or higher

Perform the following steps to make sure you can continue to leverage GPUs for workloads on Cloudera Data Science Workbench 1.6 (and higher).

1. Remove `nvidia-docker1`. Cloudera Data Science Workbench (version 1.6 and higher) ships with `nvidia-docker2` installed by default.

```
yum remove nvidia-docker
```

Perform this step on *all* hosts that have GPUs attached to them.

2. Upgrade your NVIDIA driver to version 410.xx (or higher). This must be done because `nvidia-docker2` does not support lower versions of NVIDIA drivers.

- a. Stop Cloudera Data Science Workbench.

Depending on your deployment, either stop the CDSW service in Cloudera Manager (for CSDs) or run `cdsw stop` on the Master host (for RPMs).

- b. Reboot the GPU-enabled hosts.

- c. Install a supported version of the NVIDIA driver (410.xx or higher) on all GPU-enabled hosts.

- d. Start Cloudera Data Science Workbench.

Depending on your deployment, either start the CDSW service in Cloudera Manager (for CSDs) or run `cdsw start` on the Master host (for RPMs).

Upgrading Cloudera Data Science Workbench 1.7.2 Using Packages



Important:

- Cloudera Data Science Workbench only supports upgrades to version 1.7.x from version 1.5.x and 1.6.x. If you are using a lower version, you must first upgrade to version 1.5.x or 1.6.x, and then upgrade to version 1.7.x.
- Cloudera Data Science Workbench 1.7.1 is the next official release after Cloudera Data Science Workbench 1.6.x. Version 1.7.0 is no longer publicly available.

This topic describes how to upgrade an RPM-based deployment to the latest version of Cloudera Data Science Workbench 1.7.2.

Before you start upgrading Cloudera Data Science Workbench, read the [Cloudera Data Science Workbench Release Notes](#) on page 21 relevant to the version you are upgrading to.

1. Run the following command on *all* Cloudera Data Science Workbench hosts (master and workers) to stop Cloudera Data Science Workbench.

```
cdsw stop
```

2. **(Upgrading from CDSW 1.7.1 with patch)** Perform this step only if you are upgrading from CDSW 1.7.1 and applied the patches documented here: [CDSW restart issue on multi-node deployments; CDSW Web UI does not automatically come up after upgrading to CDSW 1.7.1](#) on page 52

- a. Delete the 2 patch files: `/etc/cdsw/patches/default/deployment/ingress-controller.yaml` and `/etc/cdsw/patches/default/deployment/tcp-ingress-controller.yaml`.
- b. Delete every empty folder from the `/etc/cdsw/patches` directory.
- c. Delete the `/etc/cdsw/patches` directory if it is empty.

3. **(Strongly Recommended)** On the master host, backup all your application data that is stored in the `/var/lib/cdsw` directory.

To create the backup, run the following command on the *master* host:

```
tar -cvzf cdsw.tar.gz -C /var/lib/cdsw/ .
```

4. Save a backup of the Cloudera Data Science Workbench configuration file at `/etc/cdsw/config/cdsw.conf`.
5. Uninstall the previous release of Cloudera Data Science Workbench. Perform this step on the master host, as well as all the worker hosts.

```
yum remove cloudera-data-science-workbench
```

6. Install the latest version of Cloudera Data Science Workbench on the master host and on all the worker hosts. During the installation process, you might need to resolve certain incompatibilities in `cdsw.conf`. Even though you will be installing the latest RPM, your previous configuration settings in `cdsw.conf` will remain unchanged. Depending on the release you are upgrading from, you will need to modify [cdsw.conf](#) to ensure it passes the validation checks run by the 1.7.2 release.

To install the latest version of Cloudera Data Science Workbench, follow the same process to install the package as you would for a fresh installation.

- a. [Install Cloudera Data Science Workbench on the Master Host](#) on page 84
- b. [\(Optional\) Install Cloudera Data Science Workbench on Worker Hosts](#) on page 88.

7. **Upgrade Projects to Use the Latest Base Engine Images**

If the release you have just upgraded to includes a new version of the base engine image (see [release notes](#)), you will need to manually configure existing projects to use the new engine. Cloudera recommends you do so to take advantage of any new features and bug fixes included in the newly released engine. For example:

- **Container Security**

Security best practices dictate that engine containers should not run as the root user. Engines (v7 and lower) briefly initialize as the `root` user and then run as the `cdsw` user. Engines v8 (and higher) now follow the best practice and run only as the `cdsw` user. For more details, see [Restricting User-Created Pods](#).

- **CDH 6 Compatibility**

The base engine image you use must be compatible with the version of CDH you are running. This is especially important if you are running workloads on Spark. Older base engines (v6 and lower) cannot support the latest versions of CDH 6. If you want to run Spark workloads on CDH 6, you must upgrade your projects to base engine 7 (or higher).

- **Editors**

Engines v8 (and higher) ships with the browser-based IDE, Jupyter, preconfigured and can be selected from the **Start Session** menu.

To upgrade a project to the new engine, go to the project's **Settings > Engine** page and select the new engine from the dropdown. If any of your projects are using custom [extended engines](#), you will need to modify them to use the new base engine image.

8. (GPU-enabled Deployments) Remove `nvidia-docker1` and Upgrade NVIDIA Drivers to 410.xx or higher

Perform the following steps to make sure you can continue to leverage GPUs for workloads on Cloudera Data Science Workbench 1.6 (and higher).

1. Remove `nvidia-docker1`. Cloudera Data Science Workbench (version 1.6 and higher) ships with `nvidia-docker2` installed by default.

```
yum remove nvidia-docker
```

Perform this step on *all* hosts that have GPUs attached to them.

2. Upgrade your NVIDIA driver to version 410.xx (or higher). This must be done because `nvidia-docker2` does not support lower versions of NVIDIA drivers.
 - a. Stop Cloudera Data Science Workbench.

Depending on your deployment, either stop the CDSW service in Cloudera Manager (for CSDs) or run `cdsw stop` on the Master host (for RPMs).

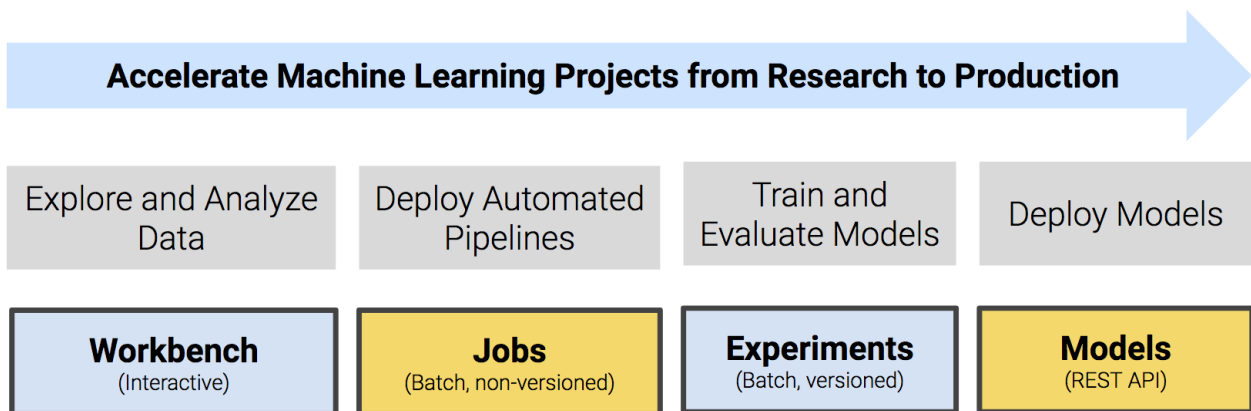
- b. Reboot the GPU-enabled hosts.
- c. Install a supported version of the NVIDIA driver (410.xx or higher) on all GPU-enabled hosts.
- d. Start Cloudera Data Science Workbench.

Depending on your deployment, either start the CDSW service in Cloudera Manager (for CSDs) or run `cdsw start` on the Master host (for RPMs).

Deploying Cloudera Data Science Workbench 1.7.2 on Hortonworks Data Platform

Cloudera Data Science Workbench is a secure, self-service enterprise data science platform that lets data scientists manage their own analytics pipelines, thus accelerating machine learning projects from exploration to production. It allows data scientists to bring their existing skills and tools, such as R, Python, and Scala, to securely run computations on data in Hadoop clusters. It enables data science teams to use their preferred data science packages to run experiments with on-demand access to compute resources. Models can be trained, deployed, and managed centrally for increased agility and compliance.

Starting with version 1.5, Cloudera Data Science Workbench includes direct integration with the Hortonworks Data Platform [for a complete machine learning workflow](#) that supports collaborative development and can run both in the public cloud and on-premises.



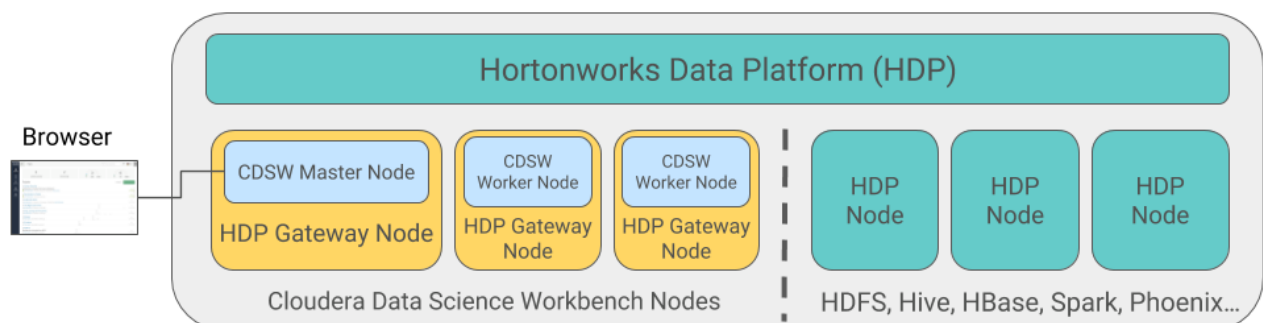
A detailed overview of the core capabilities of Cloudera Data Science Workbench is available here: [Core Capabilities of Cloudera Data Science Workbench](#).

On this page:

The rest of this topic describes how Cloudera Data Science Workbench can be deployed on HDP. It walks you through a brief architecture overview, the installation requirements, and the limitations associated with such deployments. It also includes instructions on how to install the Cloudera Data Science Workbench package on HDP.

CDSW-on-HDP Architecture Overview

Cloudera Data Science Workbench runs on one or more dedicated *gateway/edge* hosts on HDP clusters. A gateway host is one that does not have any cluster services running on them. They only run the clients for cluster services (such as the HDFS Client, YARN Client, Spark2 Client and so on). These clients ensure that Cloudera Data Science Workbench has all the libraries and configuration files necessary to securely access the HDP cluster and their respective services.



Deploying Cloudera Data Science Workbench 1.7.2 on Hortonworks Data Platform

Cloudera Data Science Workbench does not support running any other services on these gateway hosts. Each gateway host must be dedicated solely to Cloudera Data Science Workbench. This is because user workloads require dedicated CPU and memory, which might conflict with other services running on these hosts.

From the gateway hosts assigned to Cloudera Data Science Workbench, one will serve as the *master* host, which also runs the CDSW web application, while others will serve as *worker* hosts. You should note that worker hosts are not required for a fully-functional Cloudera Data Science Workbench deployment. For proof-of-concept deployments you can deploy a 1-host cluster with just a Master host. The Master host can run user workloads just as a worker can.

Refer the following topics for more details on master and worker hosts, and the other components (such as engines for workload execution) that make up Cloudera Data Science Workbench.

Related Topics:

- [CDSW Master and Worker Hosts](#)
- [CDSW Engines: For Python, R, and Scala Workloads](#)

Supported Platforms and Requirements

This topic lists the software and hardware configuration required to successfully install and run Cloudera Data Science Workbench on the Hortonworks Data Platform. Cloudera Data Science Workbench and HDP do not support hosts or clusters that do not conform to the requirements listed on this page.

Platform Requirements

- Cloudera Data Science Workbench 1.7.2 (and higher)
- Hortonworks Data Platform 2.6.5, 3.1.0, 3.1.5
 - Apache Spark 2 - Use the version of Spark that ships with the version of HDP you are running. Refer the HDP component version lists for details: [HDP 2.6.5](#), [HDP 3.1.0](#), [HDP 3.1.5](#).
 - Spark 1.x is not supported.
 - Apache Hive - Dataframe-based tables (such as tables from PySpark, sparklyr, or Scala) require Hive to be installed on your CDH cluster because of a dependency on the Hive Metastore.

Operating System Requirements

Cloudera Data Science Workbench 1.7.2 is supported on HDP on the following operating systems:

Operating System	Versions	Notes
RHEL / CentOS / Oracle Linux RHCK	7.7, 7.6, 7.5, 7.4, 7.3, 7.2	<ul style="list-style-type: none">• Cloudera Data Science Workbench installations on RHEL/CentOS 7.2 might fail due to an issue with certain versions of the <code>nfs-utils</code> package. To fix the issue, either downgrade the <code>nfs-utils</code> package or upgrade to a version with the fix. <p>View the complete Red Hat bug report here.</p>
Oracle Linux (UEK - default)	7.3	-

Additional OS-level Settings

- Enable memory cgroups on your operating system.
- Disable swap for optimum stability and performance. For instructions, see [Setting the vm.swappiness Linux Kernel Parameter](#).
- Cloudera Data Science Workbench uses **uid 8536** for an internal service account. Make sure that this user ID is not assigned to any other service or user account.
- Cloudera recommends that all users have the `max-user-processes` ulimit set to at least 65536.

- Cloudera recommends that all users have the `max-open-files` ulimit set to 1048576.

Java Requirements

The entire cluster, including Cloudera Data Science Workbench gateway hosts, must use either **Oracle JDK 8** or **OpenJDK 8**.

For Red Hat/CentOS deployments, **Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction** must be enabled on the Cloudera Data Science Workbench gateway hosts. This is to ensure that JDK uses the same default encryption type (`aes256-cts`) as that used by Red Hat/CentOS operating systems, Kerberos, and the rest of the cluster. For instructions, see [Installing the JCE on Ambari](#).

The `JAVA_HOME` configuration property must be configured as part of the [CDSW installation process](#) and must match the `JAVA_HOME` environmental variable configured for your HDP cluster. If you need to modify `JAVA_HOME` after the fact, [restart the master and worker hosts](#) to have the changes go into effect.

Related topics:

- [Changing the JDK version on an Existing HDP Cluster](#)

Network and Security Requirements



Important: Make sure that your networking/security settings on Cloudera Data Science Workbench gateway hosts are not being overwritten behind-the-scenes, either by any automated scripts, or by other high-priority configuration such as `/etc/sysctl.conf`, `/etc/krb5.conf`, or `/etc/hosts.deny`.

- Cloudera Data Science Workbench requires DNS to resolve all hostnames in your CDH cluster. CDSW does not allow using `/etc/hosts` for this.
- **(New in Cloudera Data Science Workbench 1.6.x and higher)** Enable IPv6 on all Cloudera Data Science Workbench gateway hosts. For instructions, refer the workaround provided here: [Known Issue: CDSW cannot start sessions due to connection errors](#).
- All Cloudera Data Science Workbench gateway hosts must be part of the same datacenter and use the same network. Hosts from different data-centers or networks can result in unreliable performance.
- A wildcard subdomain such as `*.cdsw.company.com` must be configured. Wildcard subdomains are used to provide isolation for user-generated content.

Starting with version 1.5, the wildcard DNS hostname configured for Cloudera Data Science Workbench must now be resolvable from both, the CDSW cluster, and your browser.

- Disable all pre-existing `iptables` rules. While Kubernetes makes extensive use of `iptables`, it's difficult to predict how pre-existing `iptables` rules will interact with the rules inserted by Kubernetes. Therefore, Cloudera recommends you use the following commands to disable all pre-existing rules before you proceed with the installation.

```
sudo iptables -P INPUT ACCEPT
sudo iptables -P FORWARD ACCEPT
sudo iptables -P OUTPUT ACCEPT
sudo iptables -t nat -F
sudo iptables -t mangle -F
sudo iptables -F
sudo iptables -X
```

- Cloudera Data Science Workbench sets the following `sysctl` options in `/etc/sysctl.d/k8s.conf`:

- `net.bridge.bridge-nf-call-iptables=1`
- `net.bridge.bridge-nf-call-ip6tables=1`
- `net.ipv4.ip_forward=1`
- `net.ipv4.conf.default.forwarding=1`

Deploying Cloudera Data Science Workbench 1.7.2 on Hortonworks Data Platform

Underlying components of Cloudera Data Science Workbench (Docker, Kubernetes, and NFS) require these options to work correctly. Make sure they are not overridden by high-priority configuration such as `/etc/sysctl.conf`.

- SELinux must either be disabled or run in *permissive* mode.
- Multi-homed networks are supported with Cloudera Data Science Workbench 1.2.2 (and higher).
- Firewall restrictions must be disabled across Cloudera Data Science Workbench and HDP hosts. Internally, the Cloudera Data Science Workbench master and worker hosts require full connectivity with no firewalls. Externally, end users connect to Cloudera Data Science Workbench exclusively through a web server running on the master host, and therefore do not need direct access to any other internal Cloudera Data Science Workbench or HDP services.

Review the complete list of ports required by Cloudera Data Science Workbench at [Ports Used By Cloudera Data Science Workbench](#) on page 267.

- Untrusted (non-sudo) SSH access to Cloudera Data Science Workbench hosts must be disabled to ensure a secure deployment.

Cloudera Data Science Workbench assumes that users only access the gateway hosts through the web application. Untrusted users with SSH access to a Cloudera Data Science Workbench host can gain full access to the cluster, including access to other users' workloads.

- `localhost` must resolve to `127.0.0.1`.
- Forward and reverse DNS lookup must be enabled for the Cloudera Data Science Workbench domain name and IP address (CDSW master host).
- Cloudera Data Science Workbench does not support DNS servers running on `127.0.0.1:53`. This IP address resolves to the container `localhost` within Cloudera Data Science Workbench containers. As a workaround, use either a non-loopback address or a remote DNS server.

Cloudera Data Science Workbench does not support hosts or clusters that do not conform to these restrictions.

Hardware Requirements

Cloudera Data Science Workbench hosts are added to your cluster as gateway hosts. The table below lists the recommended minimum hardware configuration for Cloudera Data Science Workbench gateway hosts.



Important:

- Allocate separate gateway hosts on your cluster for Cloudera Data Science Workbench. Do not reuse existing hosts that are already running other HDP services. Doing this can lead to port conflicts, unreliable execution of user workloads, and out-of-memory errors.
- All Cloudera Data Science Workbench gateway hosts must be part of the same datacenter and use the same network. Hosts from different data-centers or networks can result in unreliable performance.

Resource Type	Master	Workers	Notes
CPU	16+ CPU (vCPU) cores	16+ CPU (vCPU) cores	
RAM	32+ GB	32+ GB	
Disk Space			
Root Volume	100+ GB	100+ GB	If you are going to partition the root volume, make sure you allocate at least 20 GB to / so that the installer can proceed without running out of space.

Resource Type	Master	Workers	Notes
Application Block Device	1 TB	-	<p>The Application Block Device is only required on the Master where it is mounted to <code>/var/lib/cdsw</code>.</p> <p>You will be asked to create a <code>/var/lib/cdsw</code> directory on all the Worker hosts during the installation process. However, they do not need to be mounted to a block device. This directory is only used to store client configuration for HDP cluster services on Workers.</p>
Docker Block Device	1 TB	1 TB	The Docker Block Device is required on <i>all</i> Master and Worker hosts.

Related Topics:

- [Cloudera Data Science Workbench Scaling Guidelines](#) on page 267

Python Supported Versions

The default Cloudera Data Science Workbench engine includes **Python 2.7.11** and **Python 3.6.1**. To use PySpark within the HDP cluster, the Spark executors must have access to a matching version of Python. For many common operating systems, the default system Python will not match the minor release of Python included in Cloudera Data Science Workbench.

To ensure that the Python versions match, Python can either be installed on every HDP host or made available per job run using Spark's ability to distribute dependencies. Given the size of a typical isolated Python environment and the desire to avoid repeated uploads from gateway hosts, Cloudera recommends installing Python 2.7 and 3.6 on the cluster if you are using PySpark with lambda functions.

You can install Python 2.7 and 3.6 on the cluster using any method and set the corresponding `PYSPARK_PYTHON` [environment variable](#) in your project. Cloudera Data Science Workbench includes a separate environment variable for Python 3 sessions called `PYSPARK3_PYTHON`. Python 2 sessions continue to use the default `PYSPARK_PYTHON` variable. This will allow you to run Python 2 and Python 3 sessions in parallel without either variable being overridden by the other.

Anaconda

Anaconda is a package manager that makes it easier to install, distribute, and manage popular Python libraries and their dependencies. You can use Anaconda for package management with Cloudera Data Science Workbench, but it is not required.

**Note:**

The latest Anaconda package ships with Python 3.7. However, Cloudera Data Science Workbench only supports Python 2.7.11 and Python 3.6.1. For instructions on how to install Anaconda with Python 3.6, refer this section in the Anaconda FAQs: [How do I get Anaconda with Python 3.6?](#)

You can install Anaconda before you install Cloudera Data Science Workbench or after. Once Anaconda is installed, perform the following steps to configure Cloudera Data Science Workbench to work with Anaconda:

1. Install the Anaconda package on *all* cluster hosts. For installation instructions, refer to the [Anaconda installation documentation](#).
2. Set the `ANACONDA_DIR` property in the Cloudera Data Science Workbench configuration file: `cdsw.conf`. This can be done when you first configure `cdsw.conf` during the installation or later.
3. [Restart Cloudera Data Science Workbench](#) to have this change go into effect.

Known Issues and Limitations

- Cloudera Data Science Workbench cannot be managed by Apache Ambari.
- Apache Phoenix requires additional configuration to run commands successfully from within Cloudera Data Science Workbench engines (sessions, jobs, experiments, models).

Workaround: Explicitly set `HBASE_CONF_PATH` to a valid path before running Phoenix commands from engines.

```
export HBASE_CONF_PATH=/usr/hdp/hbase/<hdp_version>/0/
```

Installing Cloudera Data Science Workbench 1.7.2 on HDP



Note: Cloudera Data Science Workbench 1.7.1 is the next official release after Cloudera Data Science Workbench 1.6.x. Version 1.7.0 is no longer publicly available.

Use the following steps to install the Cloudera Data Science Workbench RPM package on an HDP cluster.

Prerequisites

Before you begin installing Cloudera Data Science Workbench, perform the following steps to set up a wildcard DNS subdomain for CDSW, disable untrusted SSH access to the hosts, and configure the Application and Docker block devices.

- [Set Up a Wildcard DNS Subdomain](#) on page 74
- [Disable Untrusted SSH Access](#) on page 74
- [Configure the Application and Docker Block Devices](#)

Add Gateway Hosts for Cloudera Data Science Workbench to Your HDP Cluster

To add new hosts to act as Gateway hosts for your cluster:

1. Log in to the Ambari Server.
2. Go to the **Hosts** page and select **Actions > + Add New Hosts**.
3. On the **Install Options** page, enter the fully-qualified domain names for your new hosts.

The wizard also needs the private key file you created when you set up password-less SSH. Using the host names and key file information, the wizard can locate, access, and interact securely with all the hosts in the cluster. Alternatively, you can [manually install and start the Ambari agents](#) on all the new hosts.

Click **Register and Confirm**.

For more detailed instructions, refer to [Install Options](#).

4. The **Confirm Hosts** page prompts you to confirm that Ambari has located the correct hosts for your cluster and to check those hosts to make sure they have the correct directories, packages, and processes required to continue the install. When you are satisfied with the list of hosts, click **Next**.

For detailed instructions, refer to [Confirm Hosts](#).

5. On the **Assign Slaves and Clients** page, select the Clients that should be installed on the new hosts. To install clients on all hosts, select the **Client** checkbox for every host. You can use the **all** option for each available client to expedite this.

Make sure no other services are running on these hosts. To make this easier, select the **none** option for all other services.

6. On the **Configurations** page, select the [configuration groups](#) for the new hosts.

7. The **Review** page displays the host assignments you have made. Check to make sure everything is correct. If you need to make changes, use the left navigation bar to return to the appropriate screen.

When you are satisfied with your choices, click **Deploy**.

8. The **Install, Start and Test** page displays progress as the clients are installed and deployed on each host. When the process is complete, click **Next**.
9. The **Summary** page provides you a list of the accomplished tasks. Click **Complete** and you will be directed back to the **Hosts** page.

Create HDFS User Directories

To run workloads that leverage HDP cluster services, make sure that HDFS directories (`/user/<username>`) are created for each user so that they can seamlessly connect to HDP from Cloudera Data Science Workbench.

Perform the following steps for each user directory that must be created.

1. SSH to a host in the cluster that includes the HDFS client.
2. Switch to the `hdfs` system account user:

```
su - hdfs
```

3. Create an HDFS directory for the user. For example, you would create the following directory for the default user `admin`:

```
hdfs dfs -mkdir /user/admin
```

4. Assign ownership of the new directory to the user. For example, for the new `/user/admin` directory, make the `admin` user the owner of the directory:

```
hdfs dfs -chown admin:hadoop /user/admin
```

Install Cloudera Data Science Workbench on the Master Host

Use the following steps to install Cloudera Data Science Workbench on the master host. **Note that airgapped clusters and non-airgapped clusters use different files for installation.**

1. **Non-airgapped Installation** - Download the Cloudera Data Science Workbench repo file (`cloudera-cdsw.repo`) from the following location:

```
https://archive.cloudera.com/p/cdsw1/1.7.2/redhat7/yum/cloudera-cdsw.repo
```

Airgapped installation - For airgapped installations, download the Cloudera Data Science Workbench RPM file from the following location:

```
https://archive.cloudera.com/p/cdsw1/1.7.2/redhat7/yum/RPMS/x86_64/
```



Important: Make sure all Cloudera Data Science Workbench hosts (master and worker) are running the same version of Cloudera Data Science Workbench.

2. Skip this step for airgapped installations. Add the Cloudera Public GPG repository key. This key verifies that you are downloading genuine packages.

```
sudo rpm --import
https://archive.cloudera.com/p/cdsw1/1.7.2/redhat7/yum/RPM-GPG-KEY-cloudera
```

3. Non-airgapped Installation - Install the latest RPM with the following command:

```
sudo yum install cloudera-data-science-workbench
```

Airgapped Installation - Copy the RPM downloaded in the previous step to the appropriate gateway host. Then, use the complete filename to install the package. For example:


```
sudo yum install cloudera-data-science-workbench-1.7.2.12345.rpm
```

For guidance on any warnings displayed during the installation process, see [Understanding Installation Warnings](#) on page 304.

4. Edit the configuration file at `/etc/cdswh/config/cdswh.conf`. The following table lists the configuration properties that can be configured in `cdsw.conf`.

Table 3: cdswh.conf Properties

Properties	Description
Required Configuration	
DOMAIN	Wildcard DNS domain configured to point to the master host. If the wildcard DNS entries are configured as <code>cdsw.<your_domain>.com</code> and <code>*.cdsw.<your_domain>.com</code> , then DOMAIN should be set to <code>cdsw.<your_domain>.com</code> . Users' browsers should then contact the Cloudera Data Science Workbench web application at <code>http://cdsw.<your_domain>.com</code> . This domain for DNS and is unrelated to Kerberos or LDAP domains.
MASTER_IP	IPv4 address for the master host that is reachable from the worker hosts. Within an AWS VPC, MASTER_IP should be set to the internal IP address of the master host; for instance, if your hostname is <code>ip-10-251-50-12.ec2.internal</code> , set MASTER_IP to the corresponding IP address, <code>10.251.50.12</code> .
DISTRO	The Hadoop distribution installed on the cluster. Set this property to HDP.
DOCKER_BLOCK_DEVICES	Block device(s) for Docker images (space separated if there are multiple). Use the full path to specify the image(s), for instance, <code>/dev/xvde</code> .
JAVA_HOME	Path where Java is installed on the Cloudera Data Science Workbench hosts. This path must match the JAVA_HOME environment variable that is configured for your HDP cluster. You can find the value in <code>hadoop-env.sh</code> on any node in the HDP cluster. Note that Spark 2.3 requires JDK 1.8. For more details on the specific versions of Oracle JDK recommended for HDP clusters, see the Hortonworks Support Matrix - https://supportmatrix.hortonworks.com/ .
Optional Configuration	
APPLICATION_BLOCK_DEVICE	(Master Host Only) Configure a block device for application state. If this property is left blank, the filesystem mounted at <code>/var/lib/cdsw</code> on the master host will be used to store all user data. For production deployments, Cloudera strongly recommends you use this option with a dedicated SSD block device for the <code>/var/lib/cdsw</code> mount. <i>(Not recommended)</i> If set, Cloudera Data Science Workbench will format the provided block device as <code>ext4</code> , mount it to <code>/var/lib/cdsw</code> , and store all user data on it.

Properties	Description
	<p>This option has only been provided for proof-of-concept setups, and Cloudera is not responsible for any data loss.</p> <p>Use the full path to specify the mount point, for instance, <code>/dev/xvdf</code>.</p>
RESERVE_MASTER	<p>Set this property to <code>true</code> to reserve the master host for Cloudera Data Science Workbench's internal components and services, such as Livelog, the PostgreSQL database, and so on. User workloads will now run exclusively on worker hosts, while the master is reserved for internal application services.</p> <div style="border: 1px solid orange; padding: 10px; margin: 10px 0;"> <p> Important: This feature only applies to deployments with more than one Cloudera Data Science Workbench host. Enabling this feature on single-host deployments will leave Cloudera Data Science Workbench incapable of scheduling any workloads.</p> </div>
DISTRO_DIR	<p>Path where the Hadoop distribution is installed on the Cloudera Data Science Workbench hosts. For HDP clusters, the default location of the packages is <code>/usr/hdp</code>. Specify this property only if you are using a non-default location.</p>
ANACONDA_DIR	<p>Path where Anaconda is installed. Set this property only if you are using Anaconda for package management.</p> <p>By default, the Anaconda package is installed at: <code>/home/<your-username>/anaconda<2 or 3></code>. Refer to the Anaconda FAQs for more details.</p> <p>If you choose to start using Anaconda anytime post-installation, you must set this property and then restart Cloudera Data Science Workbench to have this change take effect.</p>
TLS_ENABLE	<p>Enable and enforce HTTPS (TLS/SSL) for web access.</p> <p>Set to <code>true</code> to enable and enforce HTTPS access to the web application.</p> <p>You can also set this property to <code>true</code> to enable external TLS termination. For more details on TLS termination, see Enabling TLS/SSL for Cloudera Data Science Workbench on page 285.</p>
TLS_CERT TLS_KEY	<p>Certificate and private key for internal TLS termination.</p> <p>Setting <code>TLS_CERT</code> and <code>TLS_KEY</code> will enable internal TLS termination. You must also set <code>TLS_ENABLE</code> to <code>true</code> above to enable and enforce internal termination. Set these only if you are not terminating TLS externally.</p> <p>Make sure you specify the full path to the certificate and key files, which must be in PEM format.</p> <p>For details on certificate requirements and enabling TLS termination, see Enabling TLS/SSL for Cloudera Data Science Workbench on page 285.</p>
HTTP_PROXY HTTPS_PROXY	<p>If your deployment is behind an HTTP or HTTPS proxy, set the respective <code>HTTP_PROXY</code> or <code>HTTPS_PROXY</code> property in <code>/etc/cdsw/config/cdsw.conf</code> to the hostname of the proxy you are using.</p> <div style="border: 1px dashed blue; padding: 10px; margin: 10px 0;"> <pre>HTTP_PROXY="<http://proxy_host>:<proxy_port>" HTTPS_PROXY="<http://proxy_host>:<proxy_port>"</pre> </div>

Properties	Description
	<p>If you are using an intermediate proxy, such as Cntlm, to handle NTLM authentication, add the Cntlm proxy address to the <code>HTTP_PROXY</code> or <code>HTTPS_PROXY</code> fields in <code>cdsw.conf</code>.</p> <pre data-bbox="630 323 1143 380"> HTTP_PROXY="http://localhost:3128" HTTPS_PROXY="http://localhost:3128" </pre> <p>If the proxy server uses TLS encryption to handle connection requests, you will need to add the proxy's root CA certificate to your host's store of trusted certificates. This is because proxy servers typically sign their server certificate with their own root certificate. Therefore, any connection attempts will fail until the Cloudera Data Science Workbench host trusts the proxy's root CA certificate. If you do not have access to your proxy's root certificate, contact your Network / IT administrator.</p> <p>To enable trust, copy the proxy's root certificate to the trusted CA certificate store (<code>ca-trust</code>) on the Cloudera Data Science Workbench host.</p> <pre data-bbox="630 709 1430 766"> cp /tmp/<proxy-root-certificate>.cert /etc/pki/ca-trust/source/anchors/ </pre> <p>Use the following command to rebuild the trusted certificate store.</p> <pre data-bbox="630 856 971 884"> update-ca-trust extract </pre>
ALL_PROXY	<p>If a SOCKS proxy is in use, set to <code>socks5://<host>:<port>/</code>.</p>
NO_PROXY	<p>Comma-separated list of hostnames that should be skipped from the proxy.</p> <p>Starting with version 1.4, if you have defined a proxy in the <code>HTTP_PROXY(S)</code> or <code>ALL_PROXY</code> properties, Cloudera Data Science Workbench automatically appends the following list of IP addresses to the <code>NO_PROXY</code> configuration. Note that this is the minimum required configuration for this field.</p> <p>This list includes <code>127.0.0.1</code>, <code>localhost</code>, and any private Docker registries and HTTP services inside the firewall that Cloudera Data Science Workbench users might want to access from the engines.</p> <pre data-bbox="630 1297 1458 1612"> "127.0.0.1,localhost,100.66.0.1,100.66.0.2,100.66.0.3,100.66.0.4,100.66.0.5,100.66.0.6,100.66.0.7,100.66.0.8,100.66.0.9,100.66.0.10,100.66.0.11,100.66.0.12,100.66.0.13,100.66.0.14,100.66.0.15,100.66.0.16,100.66.0.17,100.66.0.18,100.66.0.19,100.66.0.20,100.66.0.21,100.66.0.22,100.66.0.23,100.66.0.24,100.66.0.25,100.66.0.26,100.66.0.27,100.66.0.28,100.66.0.29,100.66.0.30,100.66.0.31,100.66.0.32,100.66.0.33,100.66.0.34,100.66.0.35,100.66.0.36,100.66.0.37,100.66.0.38,100.66.0.39,100.66.0.40,100.66.0.41,100.66.0.42,100.66.0.43,100.66.0.44,100.66.0.45,100.66.0.46,100.66.0.47,100.66.0.48,100.66.0.49,100.66.0.50,100.77.0.10,100.77.0.128,100.77.0.129,100.77.0.130,100.77.0.131,100.77.0.132,100.77.0.133,100.77.0.134,100.77.0.135,100.77.0.136,100.77.0.137,100.77.0.138,100.77.0.139" </pre>
NVIDIA_GPU_ENABLE	<p>Set this property to <code>true</code> to enable GPU support for Cloudera Data Science Workbench workloads. When this property is enabled on a host is equipped with GPU hardware, the GPU(s) will be available for use by Cloudera Data Science Workbench hosts.</p> <p>If this property is set to <code>true</code> on a host that does not have GPU support, there will be no effect. By default, this property is set to <code>false</code>.</p>

Properties	Description
	For detailed instructions on how to enable GPU-based workloads on Cloudera Data Science Workbench, see Using NVIDIA GPUs for Cloudera Data Science Workbench Projects on page 137.
NVIDIA_LIBRARY_PATH	Complete path to the NVIDIA driver libraries.

5. Initialize and start Cloudera Data Science Workbench.

```
cdsw start
```

The application will take a few minutes to bootstrap. You can watch the status of application installation and startup with `watch cdsw status`.

(Optional) Install Cloudera Data Science Workbench on Worker Hosts



Note: For proof-of-concept deployments, you can deploy a 1-host cluster with just a Master host. The Master host can run user workloads just as a worker host can when required for demonstration purposes. For production deployments, Cloudera requires to have a reserved, dedicated master host and separate worker host(s).

Cloudera Data Science Workbench supports adding and removing additional worker hosts at any time. Worker hosts allow you to transparently scale the number of concurrent workloads users can run.

Use the following steps to add worker hosts to Cloudera Data Science Workbench. **Note that airgapped clusters and non-airgapped clusters use different files for installation.**

- 1. Non-airgapped Installation** - Download the Cloudera Data Science Workbench repo file (`cloudera-cdsw.repo`) from the following location:

```
https://archive.cloudera.com/p/cdsw1/1.7.2/redhat7/yum/cloudera-cdsw.repo
```

Airgapped installation - For airgapped installations, download the Cloudera Data Science Workbench RPM file from the following location:

```
https://archive.cloudera.com/p/cdsw1/1.7.2/redhat7/yum/RPMS/x86_64/
```



Important: Make sure all Cloudera Data Science Workbench hosts (master and worker) are running the same version of Cloudera Data Science Workbench.

- 2. Skip this step for airgapped installations.** Add the Cloudera Public GPG repository key. This key verifies that you are downloading genuine packages.

```
sudo rpm --import
https://archive.cloudera.com/p/cdsw1/1.7.2/redhat7/yum/RPM-GPG-KEY-cloudera
```

- 3. Non-airgapped Installation** - Install the latest RPM with the following command:

```
sudo yum install cloudera-data-science-workbench
```

Airgapped Installation - Copy the RPM downloaded in the previous step to the appropriate gateway host. Then, use the complete filename to install the package. For example:

```
sudo yum install cloudera-data-science-workbench-1.7.2.12345.rpm
```

Deploying Cloudera Data Science Workbench 1.7.2 on Hortonworks Data Platform

For guidance on any warnings displayed during the installation process, see [Understanding Installation Warnings](#) on page 304.

4. Copy `cdsw.conf` file from the master host:

```
scp root@<cdsw-master-hostname.your_domain.com>:/etc/cdsw/config/cdsw.conf  
/etc/cdsw/config/cdsw.conf
```

After initialization, the `cdsw.conf` file includes a generated bootstrap token that allows worker hosts to securely join the cluster. You can get this token by copying the configuration file from master and ensuring it has 600 permissions.

If your hosts have heterogeneous block device configurations, modify the Docker block device settings in the worker host configuration file after you copy it. Worker hosts do not need application block devices, which store the project files and database state, and this configuration option is ignored.

5. Create `/var/lib/cdsw` on the worker host. This directory must exist on all worker hosts. Without it, the next step that registers the worker host with the master will fail.

Unlike the master host, the `/var/lib/cdsw` directory on worker hosts does **not** need to be mounted to an Application Block Device. It is only used to store client configuration for HDP services on workers.

6. On the *worker* host, run the following command to add the host to the cluster:

```
cdsw join
```

This causes the worker hosts to register themselves with the Cloudera Data Science Workbench master host and increase the available pool of resources for workloads.

7. Return to the master host and verify the host is registered with this command:

```
cdsw status
```

Create the Site Administrator Account

Installation typically takes 30 minutes although it might take an additional 60 minutes for the R, Python, and Scala engine to be available on all hosts.

After installation is complete, go to the Cloudera Data Science Workbench web application at `http://cdsw.<your_domain>.com`.



Note: You must access Cloudera Data Science Workbench from the `DOMAIN` [previously configured in `cdsw.conf`](#), and not the hostname of the master host. Visiting the hostname of the master host will result in a 404 error.

Sign up for a new account. The first account that you create becomes the site administrator. As a site administrator, you can invite new users, secure the deployment, and [upload a license key](#) for the product. For more details on these tasks, see the [Administration](#) and [Security](#) guides.

Upgrading to Cloudera Data Science Workbench 1.7.2 on HDP



Note: Cloudera Data Science Workbench 1.7.1 is the next official release after Cloudera Data Science Workbench 1.6.x. Version 1.7.0 is no longer publicly available.

Before you start upgrading Cloudera Data Science Workbench, read the [Cloudera Data Science Workbench Release Notes](#) on page 21 relevant to the version you are upgrading to.

1. Run the following command on *all* Cloudera Data Science Workbench hosts (master and workers) to stop Cloudera Data Science Workbench.

```
cdsw stop
```

2. **(Strongly Recommended)** On the master host, backup all your application data that is stored in the `/var/lib/cdsw` directory.

To create the backup, run the following command on the *master* host:

```
tar -cvzf cdsw.tar.gz -C /var/lib/cdsw/ .
```

3. Save a backup of the Cloudera Data Science Workbench configuration file at `/etc/cdsw/config/cdsw.conf`.
4. Uninstall the previous release of Cloudera Data Science Workbench. Perform this step on the master host, as well as all the worker hosts.

```
yum remove cloudera-data-science-workbench
```

5. Install the latest version of Cloudera Data Science Workbench on the master host and on all the worker hosts. During the installation process, you might need to resolve certain incompatibilities in `cdsw.conf`. Even though you will be installing the latest RPM, your previous configuration settings in `cdsw.conf` will remain unchanged. Depending on the release you are upgrading from, you will need to modify `cdsw.conf` to ensure it passes the validation checks run by the 1.7.2 release.

To install the latest version of Cloudera Data Science Workbench, follow the same process to install the package as you would for a fresh installation: [Installing Cloudera Data Science Workbench 1.7.2 on HDP](#) on page 108

Getting Started with a New Project on Cloudera Data Science Workbench

To sign up, open the Cloudera Data Science Workbench web application in a browser. The application is typically hosted on the master host at `http://cdsw.<your_domain>.com`. The first time you log in, you will be prompted to create a username and password. Note that if your site administrator has configured your deployment to require invitations, you will need an invitation link to sign up.

You can use this account to create a new project and start using the workbench to run data science workloads. Watch the following video for a quick demo (demo starts at 00:30): [CDSW Quickstart Demo](#)

Related Documentation:

The Cloudera Data Science Workbench user guide includes detailed instructions that should help you get started with running workloads on Cloudera Data Science Workbench.

- [Creating and Managing Projects](#)
- [Collaborating on Projects](#)
- [Accessing Data from HDFS, Hive, HBase, etc.](#)
- Model Training and Deployment: [Experiments](#), [Models](#)

Upgrading a CDSW 1.7.2 Deployment from HDP 2 to HDP 3

Cloudera Data Science Workbench ships a single RPM that can be used to install CDSW on both, HDP 2, and HDP 3 clusters. The upgrade path for RPM deployments is:

1. [Upgrade from HDP 2.6.5 \(or higher\) to HDP 3.1.x \(or higher\)](#).
2. Restart Cloudera Data Science Workbench. Check Ambari to ensure that there are no more stale configs on the cluster hosts.

Frequently Asked Questions (FAQs)

Does CDSW-on-HDP require a license key?

Cloudera Data Science Workbench is fully functional during a 60-day, non-renewable trial period. The trial period starts when you create your first user. When the 60-day period ends, functionality will be limited. You will not be able to create any new projects or schedule any more workloads.

At this point, you must obtain a Cloudera Enterprise license and upload it to Cloudera Data Science Workbench. Cloudera Data Science Workbench will then go back to being fully functional.

For details, see [Managing License Keys for Cloudera Data Science Workbench](#) on page 262.

How do I file a support case for CDSW-on-HDP?

If you have encountered an issue, you can create a support ticket in the [Cloudera Support portal](#).

Before you log a support ticket, run the following command on the master host to create a tarball with diagnostic information for your Cloudera Data Science Workbench installation. Attach the resulting bundle to the support case you create.

```
cdsw logs
```

You can also use [SmartSense](#) to collect diagnostic data and cluster metrics from Ambari.

Getting Started with Cloudera Data Science Workbench

This topic provides a suggested method for quickly getting started with data science projects on Cloudera Data Science Workbench.

Watch the following video for a quick demo of the steps described in this topic: [CDSW Quickstart Demo](#)

Sign up

To sign up, open the Cloudera Data Science Workbench web application in a browser. The application is typically hosted on the master host at `http://cdsw.<your_domain>.com`. The first time you log in, you will be prompted to create a username and password. Note that the first account created will receive [site administrator](#) privileges.

If your site administrator has configured your deployment to require invitations, you will need an invitation link to sign up.

Create a Project from a Built-in Template

Cloudera Data Science Workbench is organized around projects. Projects hold all the code, configuration, and libraries needed to reproducibly run analyses.

To help you get started, Cloudera Data Science Workbench includes sample template projects in R, Python, PySpark, and Scala. Using a template project gives you the impetus to start using the Cloudera Data Science Workbench right away.

Create a Template Project

Create a New Project

Account
Documentation

Project Name
Python 2 Template Test

Project Visibility

- Private** - Only added collaborators can view the project.
- Team** - All members of your team can view this project.
- Public** - All authenticated users can view this project.

Initial Setup

Blank **Template** Local Git

Python

Templates include example code to help you get started.

Create Project

To create a template project:

Getting Started with Cloudera Data Science Workbench

1. Sign in to Cloudera Data Science Workbench.
2. Click **New Project**.
3. Enter the account and project name.
4. Under the **Template** tab, you can choose one of the programming languages to create a project from one of the built-in templates. Alternatively, if your site administrator has added any custom template projects, those will also be available in this dropdown list.
5. Click **Create Project**.

After creating your project, you see your project files and the list of jobs defined in your project. These project files are stored on an internal NFS server, and are available to all your project sessions and jobs, regardless of the gateway hosts they run on. Any changes you make to the code or libraries you install into your project will be immediately available when running an engine.

Launch a Session to Run the Project

Cloudera Data Science Workbench provides an interactive environment tailored for data science called the *workbench*. It supports R, Python, and Scala engines, one of which we will use to run the template project.

Workbench

The screenshot displays the Cloudera Data Science Workbench interface. On the left, a file explorer shows the project file system with files like `analysis.py`, `README.md`, and `seaborn-data`. The main area is a code editor showing a Python script for data analysis and visualization. The script includes imports for `pandas` and `sns`, data loading, and plotting. On the right, a terminal window shows the execution of the script, with output including the Pearson correlation coefficient (`pearsonr = 0.68; p = 6.7e-34`) and a plot titled "Tips Regression" showing a scatter plot with a regression line and marginal histograms.

Project file system

Terminal access

Terminal access to running engine

Interactive command prompt

Perform the following steps to run the project:

Open the Workbench to Launch a Session

To run the project code, open the workbench and launch a new session.

1. Navigate to the new project's **Overview** page.
2. Click **Open Workbench**.
3. **Launch a New Session**

Start New Session

Engine Image - Configure
Base Image v4 - docker.repository.cloudera.com/cdsw/engine:4

Select Engine Kernel

Python 2
 Python 3
 Scala
 R

Select Engine Profile

1 vCPU / 2 GiB Memory

Launch Session


1. Use **Select Engine Kernel** to choose the programming language that your project uses.
2. Use **Select Engine Profile** to select the number of [CPU cores](#) and memory to be used.
3. Click **Launch Session**.

The command prompt at the bottom right of your browser window will turn green when the engine is ready. Sessions typically take between 10 and 20 seconds to start.

Execute Project Code

You can enter and execute code using either the editor or the command prompt. The editor is best used for code you want to keep, while the command prompt is best for quick interactive exploration.

Editor - To run code in the editor:

1. Select a script from the project files on the left sidebar.
2. To run the whole script click  on the top navigation bar, or, highlight the code you want to run and press **Ctrl+Enter** (Windows/Linux) or **cmd+Enter** (macOS).

Command Prompt - The command prompt functions largely like any other. Enter a command and press **Enter** to execute it. If you want to enter more than one line of code, use **Shift+Enter** to move to the next line. The output of your code, including plots, appears in the console.

```
> ls
analysis.py  README.md  seaborn-data/
> !pip install beautifulsoup4
```

← Enter a command and press Enter

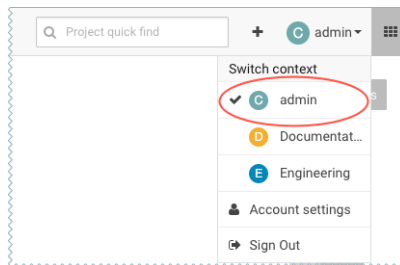
Code Autocomplete - The Python and R kernels include support for automatic code completion, both in the editor and the command prompt. Use **single tab** to display suggestions and **double tab** for autocomplete.

Managing Cloudera Data Science Workbench Users

User Contexts

The Cloudera Data Science Workbench UI uses the notion of contexts to separate your personal account from any team accounts you belong to. Depending on the context you are in, you will be able to modify settings for either your personal account, or a team account, and see the projects created in each account. Shared personal projects will show up in your personal account context.

Context changes in the UI are subtle, so if you're wondering where a project or setting lives, first make sure you are in the right context. The application header will tell you which context you are currently in. You can switch to a different context by going to the drop-down menu in the upper right-hand corner of the page.



Personal

Your personal context can be used to work on private projects, experiments, and so on. For related management tasks, see [Managing your Personal Account](#) on page 121.

Team

Teams allow streamlined administration of projects. If you are going to collaborate with teammates on more than one project, you should create a team account and work on shared projects under the team context. For related tasks, see [Managing Team Accounts](#) on page 122.

User Roles

Currently, Cloudera Data Science Workbench has only one deployment-wide specialised user role: site administrator. All other roles are created within the scope of a particular team or project and do not affect the rest of the deployment.

Site Administrators

Site administrators are superusers who have complete access to *all* activity on the deployment. By default, the first user account that signs up for Cloudera Data Science Workbench becomes a site administrator. They can add/invite more users, disable existing users, assign site administrator privileges to others, and so on. For instructions, see [Managing Users as a Site Administrator](#) on page 123.

Roles within Teams and Projects

Within the context of each team and project, users can be assigned Viewer, Operator, Contributor, or Admin roles. These roles are only applicable within the limited scope of the team/project. More details on the privileges associated with each of these roles are included in the [Creating Teams](#) and [Project Collaborators](#) topics respectively.

Managing your Personal Account

To manage your personal account settings:

1. Sign in to Cloudera Data Science Workbench.
2. From the upper right drop-down menu, switch context to your personal account.
3. Click **Settings**.

Profile

You can modify your name, email, and bio on this page.

Teams

This page lists the teams you are a part of and the role assigned to you for each team.

SSH Keys

Your public SSH key resides here. SSH keys provide a useful way to access to external resources such as databases or remote Git repositories. For instructions, see [SSH Keys](#) on page 302.

Hadoop Authentication

Enter your Hadoop credentials here to authenticate yourself against the cluster KDC. For more information, see [Hadoop Authentication with Kerberos for Cloudera Data Science Workbench](#) on page 291.



Important: You can also access your personal account settings by clicking **Account settings** in the upper right-hand corner drop-down menu. This option will always take you to your personal settings page, irrespective of the context you are currently in.

Managing Team Accounts

Users who work together on more than one project and want to facilitate collaboration can create a Team. Teams allow streamlined administration of projects. Team projects are owned by the team, rather than an individual user. Team administrators can add or remove members at any time, assigning each member different permissions.

Members	Type	Actions
Ambreen	Admin	change delete
Christopher	Admin	change delete
John	Contributor	change delete

Only team members can access an team's projects.

Creating a Team

To create a team:

1. Click the plus sign (+) in the title bar, to the right of the **Search** field.
2. Select **Create Team**.
3. Enter a **Team Name**.
4. Click **Create Team**.
5. Add or invite team members. Team members can have one of the following privilege levels:
 - **Viewer** - Read-only access to team projects. Cannot create new projects within the team but can be added to existing ones.

- **Operator** - Read-only access to team projects. Cannot create new projects within the team but can be added to existing ones. Additionally, Operators can start and stop existing jobs in the projects that they have access to.
- **Contributor** - Write-level access to all team projects to all team projects with **Team** or **Public** visibility. Can create new projects within the team. They can also be added to existing team projects.
- **Admin** - Has complete access to all team projects, can add new team members, and modify team account information.

6. Click **Done**.



Note: Team administrators and team members with Admin or Contributor privileges on projects have access to all your sessions and can execute commands within your active sessions. Cloudera Data Science Workbench 1.4.3 introduces a new feature that allows site administrators to restrict this ability by allowing only session creators to execute commands within their own active sessions. For details, see [Restricting Access to Active Sessions](#).

Modifying Team Account Settings

Team administrators can modify account information, add or invite new team members, and view/edit privileges of existing members. To make these changes:

1. From the upper right drop-down menu, switch context to the team account.
2. Click **Settings** to open up the Account Settings dashboard.

Profile

Modify the team description on this page.

Members

You can add new team members on this page, and modify privilege levels for existing members.

SSH Keys

The team's public SSH key resides here. Team SSH keys provide a useful way to give an entire team access to external resources such as databases. For instructions, see [SSH Keys](#) on page 302. Generally, team SSH keys should not be used to authenticate against Git repositories. Use your personal key instead.

Managing Users as a Site Administrator

Required Role: [Site Administrator](#)

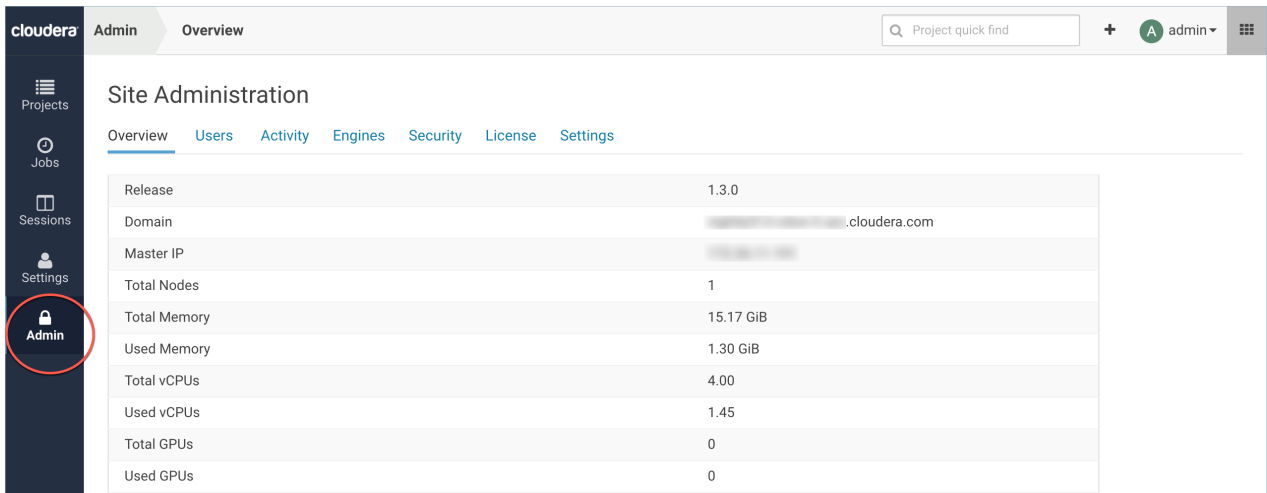
This topic describes how to manage Cloudera Data Science Workbench users as a site administrator. By default, the first user account that signs up for the Cloudera Data Science Workbench becomes a site administrator. Site administrators can manage other users, monitor resources, secure access to the deployment, and upload license keys for the product.



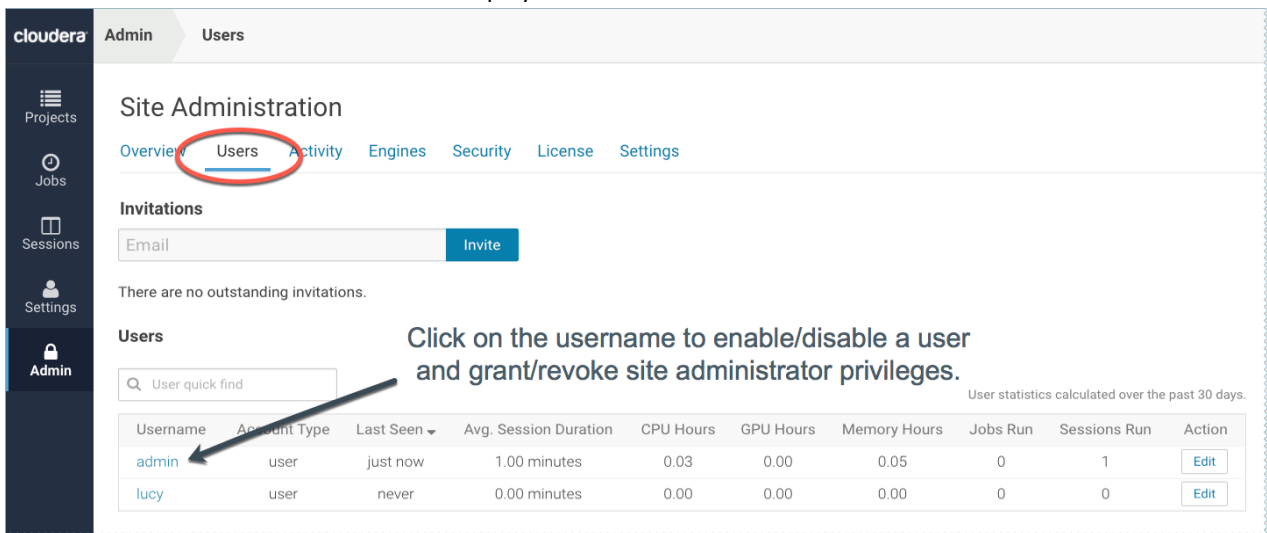
Important: Site administrators have complete access to *all* activity on the deployment. This includes access to all teams and projects on the deployment, even if they have not been explicitly added as team members or collaborators.

To access the site administrator dashboard:

1. Go to the Cloudera Data Science Workbench web application (<http://cdsw.company.com>) and log in as a site administrator.
2. On the left sidebar, click **Admin**. You will see an array of tabs for all the tasks you can perform as a site administrator.



As a site administrator you can add new users, assign or modify privileges for existing users, and monitor user activity on the Cloudera Data Science Workbench deployment.



Adding New Users

To invite new users, navigate to the **Admin > Users** tab. Under **Invitations**, enter the name or email ID of the person you want to invite and click **Invite**. This tab will show you a list of all outstanding invitations. Once an invitation has been accepted, the record will no longer show up on this page. The **Users** tab also displays a list of users of the application. Click on a username to see more details about the user.

If you want new users to join by invitation only, go to the **Admin > Settings** tab and check the **Require invitation to sign up** checkbox to require invitation tokens for account creation. By default, invitations are sent from `noreply@your-cds-w-domain`. To modify this default, see [Cloudera Data Science Workbench Email Notifications](#) on page 262.

Assigning the Site Administrator role to an Existing User

To make a regular user a site administrator:

1. Sign in to Cloudera Data Science Workbench as a site administrator.
2. Click **Admin**.
3. Click the **Users** tab.
4. Click on the username of the user who you want to make a site administrator.
5. Select the **Site Administrator** checkbox.

6. Click **Update**.

Disabling User Accounts

Use the following instructions to disable user accounts. Note that disabled users cannot login and do not count towards named users for licensing.

1. Sign in to Cloudera Data Science Workbench as a site administrator.
2. Click **Admin**.
3. Click the **Users** tab.
4. Click on the username of the user who you want to disable.
5. Select the **Disabled** checkbox.
6. Click **Update**.

Monitoring Users

The **Users** tab on the admin dashboard displays the complete list of users. You can see which users are currently active, and when a user last logged in to the Cloudera Data Science Workbench. To modify a user's username, email or permissions, click the **Edit** button under the Action column.

Managing Projects in Cloudera Data Science Workbench

Projects form the heart of Cloudera Data Science Workbench. They hold all the code, configuration, and libraries needed to reproducibly run analyses. Each project is independent, ensuring users can work freely without interfering with one another or breaking existing workloads.

This topic describes how to create and manage projects in Cloudera Data Science Workbench.

Creating a Project

To create a Cloudera Data Science Workbench project:

1. Go to Cloudera Data Science Workbench and on the left sidebar, click **Projects**.
2. Click **New Project**.
3. If you are a member of a team, from the drop-down menu, select the **Account** under which you want to create this project. If there is only one account on the deployment, you will not see this option.
4. Enter a **Project Name**.
5. Select **Project Visibility** from one of the following options.
 - **Private** - Only [project collaborators](#) can view or edit the project.
 - **Team** - If the project is created under a team account, all members of the team can view the project. Only explicitly-added collaborators can edit the project.
 - **Public** - All authenticated users of Cloudera Data Science Workbench will be able to view the project. Collaborators will be able to edit the project.
6. Under **Initial Setup**, you can either create a blank project, or select one of the following sources for your project files.
 - **Built-in Templates** - Template projects contain example code that can help you get started with the Cloudera Data Science Workbench. They are available in R, Python, PySpark, and Scala. Using a template project is not required, but it helps you start using the Cloudera Data Science Workbench right away.

Custom Templates - Starting with version 1.3, site administrators can add template projects that are customized for their organization's use-cases. For details, see [Custom Template Projects](#) on page 129.
 - **Local** - If you have an existing project on your local disk, use this option to upload compressed files or folders to Cloudera Data Science Workbench.
 - **Git** - If you already use Git for version control and collaboration, you can continue to do so with the Cloudera Data Science Workbench. Specifying a Git URL will clone the project into Cloudera Data Science Workbench. If you use a Git SSH URL, your personal private SSH key will be used to clone the repository. This is the recommended approach. However, you must add the public SSH key from your personal Cloudera Data Science Workbench account to the remote Git hosting service before you can clone the project. Specify your username and password in the URL as follows:

```
http://username:password@server/path/project.git
```

7. Click **Create Project**. After the project is created, you can see your project files and the list of jobs defined in your project.

Note that as part of the project filesystem, Cloudera Data Science Workbench also creates the following `.gitignore` file.

```
R
node_modules
*.pyc
```

```
.*
!.gitignore
```

- 8. (Optional)** To work with team members on a project, use the instructions in the following section to add them as collaborators to the project.

Adding Collaborators

If you want to work closely with colleagues on a particular project, use the following steps to add them to the project.

1. Navigate to the project overview page.
2. Click **Team** to open the Collaborators page.
3. Search for collaborators by either name or email address and click **Add**.

For a project created under your personal account, anyone who belongs to your organization can be added as a collaborator. For a project created under a team account, you can only add collaborators that already belong to the team. If you want to work on a project that requires collaborators from different teams, create a new team with the required members, and then create a project under that account. If your project was created from a Git repository, each collaborator must create the project from the same central Git repository.

You can grant project collaborators one of three levels of access:

- **Viewer** - Read-only access to code, data, and results.
- **Operator** - Read-only access to code, data, and results. Additionally, Operators can start and stop existing jobs in the projects that they have access to.
- **Contributor** - Can view, edit, create, and delete files and environmental variables, run sessions/experiments/jobs/models and execute code in running jobs. Additionally, Contributors can set the default engine for the project.
- **Admin** - Has complete access to all aspects of the project. This includes the ability to add new collaborators, and delete the entire project.



Warning:

Collaborating Securely on Projects

Before adding project collaborators, you must remember that assigning the *Contributor* or *Admin* role to a project collaborator is the same as giving them write access to your data in CDH. This is because project contributors and project administrators have write access to all your project code (including any library code that you might not be actively inspecting). For example, a contributor/admin could modify project file(s) to insert code that deletes some data on the CDH cluster. The next time you launch a session and run the same code, it will appear as though you deleted the data yourself.

Additionally, project collaborators also have access to all actively running sessions and jobs. This means that a malicious user can easily impersonate you by accessing one of *your* active sessions. Therefore, it is extremely important to restrict project access to trusted collaborators only. Note that Cloudera Data Science Workbench 1.4.3 introduces a new feature that allows site administrators to restrict this ability by allowing only session creators to execute commands within their own active sessions. For details, see [Restricting Access to Active Sessions](#).

For these reasons, Cloudera recommends using Git to collaborate securely on shared projects. This will also help avoid file modification conflicts when your team is working on more elaborate projects.

For more information on collaborating effectively, see [Collaborating on Projects with Cloudera Data Science Workbench](#) on page 150.

Modifying Project Settings

Project contributors and administrators can modify aspects of the project environment such as the engine being used to launch sessions, the environment variables, and create SSH tunnels to access external resources. To make these changes:

1. Switch context to the account where the project was created.
2. Click **Projects**.
3. From the list of projects, select the one you want to modify.
4. Click **Settings** to open up the Project Settings dashboard.

Options

Modify the project name and its privacy settings on this page.

Engine

Cloudera Data Science Workbench ensures that your code is always run with the specific engine version you selected. You can select the version here. For advanced use cases, Cloudera Data Science Workbench projects can use custom Docker images for their projects. Site administrators can whitelist images for use in projects, and project administrators can use this page to select which of these whitelisted images is installed for their projects. For an example, see [Customized Engine Images](#) on page 220.

Environment - If there are any environmental variables that should be injected into all the engines running this project, you can add them to this page. For more details, see [Engine Environment Variables](#) on page 215.

Tunnels

In some environments, external databases and data sources reside behind restrictive firewalls. Cloudera Data Science Workbench provides a convenient way to connect to such resources using your SSH key. For instructions, see [SSH Tunnels](#) on page 302.

Delete Project

This page can only be accessed by project administrators. Remember that deleting a project is irreversible. All files, data, sessions, and jobs will be lost.

Managing Files



Important: For use cases beyond simple projects, Cloudera strongly recommends using [Git](#) to manage your projects using version control.

Cloudera Data Science Workbench allows you to move, rename, copy, and delete files within the scope of the project where they live. You can also upload new files to a project, or download project files. Files can only be uploaded within the scope of a single project. Therefore, to access a script or data file from multiple projects, you will need to manually upload it to all the relevant projects.

1. Switch context to the account where the project was created.
2. Click **Projects**.
3. From the list of projects, click on the project you want to modify. This will take you to the project overview.
4. Click **Files**.

Upload Files to a Project

Click **Upload**. Select **Files** or **Folder** from the dropdown, and choose the files or folder you want to upload from your local filesystem.

In addition to uploading files or a folder, you can upload a `.tar` file of multiple files and folders. After you select and upload the `.tar` file, you can use a terminal session to extract the contents:

1. On the project overview page, click **Open Workbench** and select a running session or create a new one.

2. Click **Terminal access**.
3. In the terminal window, extract the contents of the `.tar` file:

```
tar -xvf <file_name>.tar.gz
```

The extracted files are now available for the project.

Download Project Files

Click **Download** to download the entire project in a `.zip` file. To download only a specific file, select the checkbox next to the file(s) to be download and click **Download**.

You can also use the checkboxes to **Move**, **Rename**, or **Delete** files within the scope of this project.

Disabling Project File Uploads and Downloads

Required Role: [Site Administrator](#)

By default, all Cloudera Data Science Workbench users are allowed to upload and download files to/from a project. Version 1.5 introduces a new feature flag that allows site administrators to hide the UI features that let users upload and download project files.

Note that this feature flag only removes the relevant features from the Cloudera Data Science Workbench UI. It does not disable the ability to upload and download files through the backend web API.



Note: You cannot disable file upload and download when using the Jupyter Notebook.

To disable project file uploads and downloads:

1. Go to **Admin > Security**.
2. Under the **File Upload/Download** section, disable the **Allow file upload/download through UI** checkbox.

Custom Template Projects

Required Role: [Site Administrator](#)

Site administrators can add template projects that have been customized for their organization's use-cases. These custom project templates can be added in the form of a Git repository.

To add a new template project, go to **Admin > Settings**. Under the **Project Templates** section, provide a template name, the URL to the project's Git repository, and click **Add**.

The added templates will become available in the Template tab on the [Create Project](#) page. Site administrators can add, edit, or delete custom templates, but not the built-in ones. However, individual built-in templates can be disabled using a checkbox in the **Project Templates** table at **Admin > Settings**.

Deleting a Project



Important: Deleting a project is an irreversible action. All files, data, and history related to the project will be lost. This includes any jobs, sessions or models you created within the project.

To delete a project:

1. Go to the project **Overview** page.
2. On the left sidebar, click **Settings**.
3. Go to the **Delete Project**.

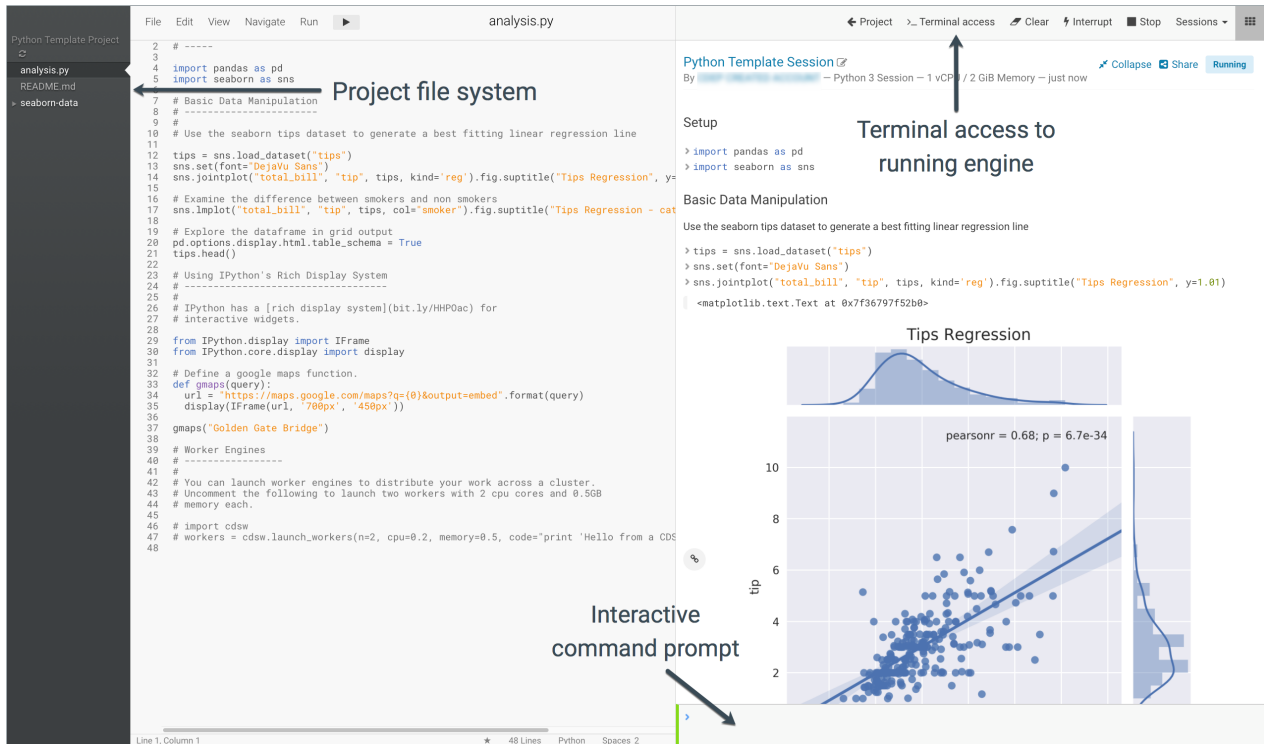
4. Click **Delete Project** and click **OK** to confirm.

Using the Workbench

The workbench console provides an interactive environment tailored for data science, supporting R, Python and Scala. It currently supports R, Python, and Scala [engines](#). You can use these engines in isolation, as you would on your laptop, or connect to your CDH cluster using Cloudera Distribution of Apache Spark 2 and other libraries.

The workbench UI includes four primary components:

- An editor where you can edit your scripts.
- A console where you can track the results of your analysis.
- A command prompt where you can enter commands interactively.
- A terminal where you can use a Bash shell.



Typically, you would use the following steps to run a project in the workbench:

Launch a Session



Note: Shell start up files are not run during session start up. CDSW sessions are not bash shells, so shell start up files such as `bashrc`, `zsh`, and `ksh` are not run. If you want to set an environment variable, you can use the CDSW environment variables feature. This will ensure the environment variable is injected in all contexts: sessions, terminals, experiments, models, jobs, etc. If you want to run more complicated code during startup (for example, conditional statements), consider using `PYTHONSTARTUP`, see [Startup.py](#) or `Rprofile`, see [Managing R with .Rprofile](#), [.Renviron](#), [Rprofile.site](#), [Renviron.site](#), [rsession.conf](#), and [repos.conf](#).

1. Navigate to your project's **Overview** page.
2. Click **Open Workbench**.
3. **Launch a New Session**

Start New Session

Engine Image - Configure
Base Image v4 - docker.repository.cloudera.com/cdsw/engine:4

Select Engine Kernel

Python 2
 Python 3
 Scala
 R

Select Engine Profile

1 vCPU / 2 GiB Memory

Launch Session

1. Use **Select Engine Kernel** to choose the programming language that your project uses.
2. Make sure that you select **Workbench** as the **Editor**.
3. Use **Select Engine Profile** to select the number of [CPU cores](#) and memory to be used.
4. Click **Launch Session**.

The command prompt at the bottom right of your browser window will turn green when the engine is ready. Sessions typically take between 10 and 20 seconds to start.

Execute Code

You can enter and execute code at the command prompt or the editor. The editor is best for code you want to keep, while the command prompt is best for quick interactive exploration.

Command Prompt - The command prompt functions largely like any other. Enter a command and press **Enter** to execute it. If you want to enter more than one line of code, use **Shift+Enter** to move to the next line. The output of your code, including plots, appears in the console.


```

> ls
analysis.py  README.md  seaborn-data/
> !pip install beautifulsoup4
  
```

← Enter a command and press Enter

If you created your project from a template, you should see project files in the editor. You can open a file in the editor by clicking the file name in the file navigation bar on the left.

Editor - To run code from the editor:

1. Select a script from the project files on the left sidebar.
2. To run the whole script click  on the top navigation bar, or, highlight the code you want to run and press **Ctrl+Enter** (Windows/Linux) or **cmd+Enter** (macOS).

Scala

```
quit()
```

Sessions automatically stop after an hour of inactivity.

Jupyter Magic Commands

Cloudera Data Science Workbench's Scala and Python kernels are based on Jupyter kernels. Jupyter kernels support varying magic commands that extend the core language with useful shortcuts. This section details the magic commands (magics) supported by Cloudera Data Science Workbench.

Line magics begin with a single %: for example, `%timeit`. *Cell magics* begin with a double %: for example, `%%bash`.

Python

In the default Python engine, Cloudera Data Science Workbench supports most line magics, but no cell magics.

Cloudera Data Science Workbench supports the shell magic `!`: for example, `!ls -alh /home/cdsw`.

Cloudera Data Science Workbench supports the help magics `?` and `??`: for example, `?numpy` and `??numpy`. `?` displays the docstring for its argument. `??` attempts to print the source code. You can get help on magics using the `?` prefix: for example, `?%timeit`.

Cloudera Data Science Workbench supports the line magics listed at <https://ipython.org/ipython-doc/3/interactive/magics.html#line-magics>, with the following exceptions:

- `%colors`
- `%debug`
- `%edit`
- `%gui`
- `%history`
- `%install_default_config`
- `%install_profiles`
- `%lsmagic`
- `%macro`
- `%matplotlib`
- `%notebook`
- `%page`
- `%pastebin`
- `%pdb`
- `%prun`
- `%pylab`
- `%recall`
- `%rerun`
- `%save`
- `%sc`

Scala

Cloudera Data Science Workbench's Scala kernel is based on Apache Toree. It supports the line magics documented in the Apache Toree [magic tutorial](#).

Data Visualization

Each language on Cloudera Data Science Workbench has a visualization system that you can use to create plots, including rich HTML visualizations.

Simple Plots

To create a simple plot, run a console in your favorite language and paste in the following code sample:

R

```
# A standard R plot
plot(rnorm(1000))

# A ggplot2 plot
library("ggplot2")
qplot(hp, mpg, data=mtcars, color=am,
facets=gear~cyl, size=I(3),
xlab="Horsepower", ylab="Miles per Gallon")
```

Python

```
import matplotlib.pyplot as plt
import random
plt.plot([random.normalvariate(0,1) for i in xrange(1,1000)])
```

Cloudera Data Science Workbench processes each line of code individually (unlike notebooks that process code per-cell). This means if your plot requires multiple commands, you will see incomplete plots in the workbench as each line is processed.

To get around this behavior, wrap all your plotting commands in one Python function. Cloudera Data Science Workbench will then process the function as a whole, and not as individual lines. You should then see your plots as expected.

Saved Images

You can also display images, using a command in the following format:

R

```
library("cdsw")

download.file("https://upload.wikimedia.org/wikipedia/commons/2/29/Minard.png",
"/cdn/Minard.png")
image("Minard.png")
```

Python

```
import urllib
from IPython.display import Image
urllib.urlretrieve("http://upload.wikimedia.org/wikipedia/commons/2/29/Minard.png",
"Minard.png")

Image(filename="Minard.png")
```

HTML Visualizations

Your code can generate and display HTML. To create an HTML widget, paste in the following:

R

```
library("cdsw")
html('<svg><circle cx="50" cy="50" r="50" fill="red" /></svg>')
```

Python

```
from IPython.display import HTML
HTML('<svg><circle cx="50" cy="50" r="50" fill="red" /></svg>')
```

Scala

Cloudera Data Science Workbench allows you to build visualization libraries for Scala using [jvm-repr](#). The following example demonstrates how to register a custom HTML representation with the "text/html" mimetype in Cloudera Data Science Workbench. This output will render as HTML in your workbench session.

```
//HTML representation
case class HTML(html: String)

//Register a displayer to render html
Displayers.register(classOf[HTML],
  new Displayer[HTML] {
    override def display(html: HTML): java.util.Map[String, String] = {
      Map(
        "text/html" -> html.html
      ).asJava
    }
  })

val helloHTML = HTML("<h1> <em> Hello World </em> </h1>")
display(helloHTML)
```

Iframe Visualizations



Note:

Cloudera Data Science Workbench versions 1.4.2 (and higher) added a new feature that allowed users to [enable HTTP security headers](#) for responses to Cloudera Data Science Workbench. This setting is enabled by default. However, the X-Frame-Options header added as part of this feature blocks rendering of iFrames injected by third-party data visualization libraries.

To work around this issue, a site administrator can go to the **Admin > Security** page and disable the **Enable HTTP security headers** property. Restart Cloudera Data Science Workbench for this change to take effect.

Most visualizations require more than basic HTML. Embedding HTML directly in your console also risks conflicts between different parts of your code. The most flexible way to embed a web resource is using an [Iframe](#):

R

```
library("cdsw")
iframe(src="https://www.youtube.com/embed/8pHzROP1D-w", width="854px", height="510px")
```

Python

```
from IPython.display import HTML
HTML('<iframe width="854" height="510"
src="https://www.youtube.com/embed/8pHzROP1D-w"></iframe>')
```

You can generate HTML files within your console and display them in IFrames using the `/cdn` folder. The `cdn` folder persists and services static assets generated by your engine runs. For instance, you can embed a full HTML file with IFrames.

R

```
library("cdsw")
f <- file("/cdn/index.html")
html.content <- paste("<p>Here is a normal random variate:", rnorm(1), "</p>")
writeLines(c(html.content), f)
close(f)
iframe("index.html")
```

Python

```
from IPython.display import HTML
import random

html_content = "<p>Here is a normal random variate: %f </p>" % random.normalvariate(0,1)

file("/cdn/index.html", "w").write(html_content)
HTML("<iframe src=index.html>")
```

Cloudera Data Science Workbench uses this feature to support many rich plotting libraries such as `htmlwidgets`, `Bokeh`, and `Plotly`.

Grid Displays

Cloudera Data Science Workbench supports native grid displays of DataFrames across several languages.

Python

Using DataFrames with the `pandas` package requires per-session activation:

```
import pandas as pd
pd.DataFrame(data=[range(1,100)])
```

For PySpark DataFrames, use `pandas` and run `df.toPandas()` on a PySpark DataFrame. This will bring the DataFrame into local memory as a pandas DataFrame.



Note:

A Python project originally created with engine 1 will be running `pandas` version 0.19, and will not auto-upgrade to version 0.20 by simply selecting engine 2 in the project's **Settings > Engine** page.

The `pandas` data grid setting only exists starting in version 0.20.1. To upgrade, manually install version 0.20.1 at the session prompt.

```
!pip install pandas==0.20.1
```

R

In R, DataFrames will display as grids by default. For example, to view the Iris data set, you would just use:

```
iris
```

Similar to PySpark, bringing Sparklyr data into local memory with `as.data.frame` will output a grid display.

```
sparkly_df %>% as.data.frame
```

Scala

Calling the `display()` function on an existing dataframe will trigger a collect, much like `df.show()`.

```
val df = sc.parallelize(1 to 100).toDF()
display(df)
```

Documenting Your Analysis

Cloudera Data Science Workbench supports Markdown documentation of your code written in comments. This allows you to generate reports directly from valid Python and R code that runs anywhere, even outside Cloudera Data Science Workbench. To add documentation to your analysis, create comments in [Markdown](#) format:

R

```
# Heading
# -----
#
# This documentation is important.
#
# Inline math:  $e^x$ 
#
# Display math:  $y = \Sigma x + \epsilon$ 
print("Now the code!")
```

Python

```
# Heading
# -----
#
# This documentation is important.
#
# Inline math:  $e^x$ 
#
# Display math:  $y = \Sigma x + \epsilon$ 
print("Now the code!")
```

Using NVIDIA GPUs for Cloudera Data Science Workbench Projects

Minimum Required Roles: [Cloudera Manager Cluster Administrator](#), [CDSW Site Administrator](#)

A GPU is a specialized processor that can be used to accelerate highly parallelized computationally-intensive workloads. Because of their computational power, GPUs have been found to be particularly well-suited to [deep learning](#) workloads. Ideally, CPUs and GPUs should be used in tandem for data engineering and data science workloads. A typical machine learning workflow involves data preparation, model training, model scoring, and model fitting. You can use existing general-purpose CPUs for each stage of the workflow, and optionally accelerate the math-intensive steps with the selective application of special-purpose GPUs. For example, GPUs allow you to accelerate model fitting using frameworks such as [Tensorflow](#), [PyTorch](#), [Keras](#), [MXNet](#), and [Microsoft Cognitive Toolkit \(CNTK\)](#).

By enabling GPU support, data scientists can share GPU resources available on Cloudera Data Science Workbench hosts. Users can request a specific number of GPU instances, up to the total number available on a host, which are then allocated to the running session or job for the duration of the run. Projects can use isolated versions of libraries, and even different CUDA and cuDNN versions via Cloudera Data Science Workbench's [extensible engine](#) feature.

Prerequisite

This topic assumes you have already installed or upgraded to the latest version of Cloudera Data Science Workbench.

Key Points to Note

- Cloudera Data Science Workbench only supports CUDA-enabled NVIDIA GPU cards.
- Cloudera Data Science Workbench does not support heterogeneous GPU hardware in a single deployment.
- Cloudera Data Science Workbench does not include an engine image that supports NVIDIA libraries. Create your own custom CUDA-capable engine image using the instructions described in this topic.
- Cloudera Data Science Workbench does not install or configure the NVIDIA drivers on the Cloudera Data Science Workbench gateway hosts. These depend on your GPU hardware and will have to be installed by your system administrator. The steps provided in this topic are generic guidelines that will help you evaluate your setup.
- The instructions described in this topic require Internet access. If you have an airgapped deployment, you will be required to manually download and load the resources onto your hosts.

- For a list of known issues associated with this feature, refer Known Issues - [GPU Support](#) on page 58.

Enabling Cloudera Data Science Workbench to use GPUs

To enable GPU usage on Cloudera Data Science Workbench, perform the following steps to provision the Cloudera Data Science Workbench hosts. As noted in the following instructions, certain steps must be repeated on all gateway hosts that have GPU hardware installed on them.

The steps described in this document have been tested and validated on the following setup:

CDSW	OS & Kernel	NVIDIA Driver	CUDA
1.7.x (engine 10)	RHEL 7.4 3.10.0-862.9.1.el7.x86_64	418.56	CUDA 10.1
1.7.x (engine 10)	RHEL 7.6 3.10.0-957.12.2.el7.x86_64	418.56	CUDA 10.1

For more compatibility information across NVIDIA Drivers and CUDA, refer the [NVIDIA documentation: CUDA Compatibility](#).

Set Up the Operating System and Kernel

Perform this step on all hosts with GPU hardware installed on them.

1. Install the `kernel-devel` package.

```
sudo yum install -y kernel-devel-`uname -r`
```

If the previous command fails to find a matching version of the `kernel-devel` package, list all the `kernel/kernel-devel` versions that are available from the RHEL/CentOS package repositories, and pick the desired version to install.

You can use a bash script as demonstrated here to do this:

```
if ! yum install kernel-devel-`uname -r`; then
  yum install -y kernel kernel-devel; retVal=$?
  if [ $retVal -eq 0 ]; then echo "Reboot is required since new version of kernel was installed"; fi
fi
```

2. If you upgraded to a new kernel version in the previous step, run the following command to reboot.

```
sudo reboot
```

3. Install the Development tools package.

```
sudo yum groupinstall -y "Development tools"
```

Install the NVIDIA Driver on GPU Hosts

Perform this step on all hosts with GPU hardware installed on them.

Cloudera Data Science Workbench does not ship with any of the NVIDIA drivers needed to enable GPUs for general purpose processing. System administrators are expected to install the version of the drivers that are compatible with the CUDA libraries that will be consumed on each host.

Use the [NVIDIA UNIX Driver archive](#) to find out which driver is compatible with your GPU card and operating system. To download and install the NVIDIA driver, make sure you **follow the instructions on the respective driver's download page**. It is crucial that you download the correct version.

For example, if you use the `.run` file method (Linux 64 bit), you would download and install the driver as follows:

```
wget http://us.download.nvidia.com/.../NVIDIA-Linux-x86_64-<driver_version>.run
export NVIDIA_DRIVER_VERSION=<driver_version>
chmod 755 ./NVIDIA-Linux-x86_64-$NVIDIA_DRIVER_VERSION.run
./NVIDIA-Linux-x86_64-$NVIDIA_DRIVER_VERSION.run -asq
```

Once the installation is complete, run the following command to verify that the driver was installed correctly:

```
/usr/bin/nvidia-smi
```

Enable GPU Support in Cloudera Data Science Workbench

Minimum Required Cloudera Manager Role: [Cluster Administrator](#)

Depending on your deployment, use one of the following sets of steps to enable Cloudera Data Science Workbench to identify the GPUs installed:

CSD Deployments

1. Ensure that the Docker daemon and worker node roles are installed on the GPU node.

You might need to restart CDSW after you install the Docker daemon and worker node roles and before enabling GPU support.

2. Go to the CDSW service in Cloudera Manager. Click **Configuration**. Search for the following property and enable it:

Enable GPU Support	Use the checkbox to enable GPU support for Cloudera Data Science Workbench workloads. When this property is enabled on a host that is equipped with GPU hardware, the GPU(s) will be available for use by Cloudera Data Science Workbench.
---------------------------	--

3. [Restart the CDSW service](#) in Cloudera Manager.
4. [Test whether Cloudera Data Science Workbench is detecting GPUs.](#)

RPM Deployments

1. Set the following parameter in `/etc/cdsw/config/cdsw.conf` on *all* Cloudera Data Science Workbench hosts. You must make sure that `cdsw.conf` is consistent across all hosts, irrespective of whether they have GPU hardware installed on them.

NVIDIA_GPU_ENABLE	Set this property to <code>true</code> to enable GPU support for Cloudera Data Science Workbench workloads. When this property is enabled on a host that is equipped with GPU hardware, the GPU(s) will be available for use by Cloudera Data Science Workbench.
-------------------	--

2. On the *master* host, run the following command to restart Cloudera Data Science Workbench.

```
cdsw restart
```

If you modified `cdsw.conf` on a *worker* host, run the following commands to make sure the changes go into effect:

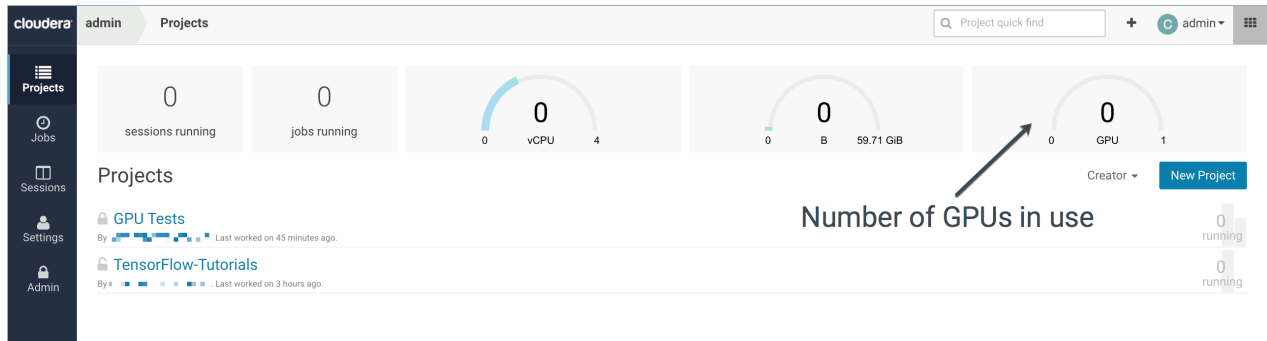
```
cdsw stop
cdsw join
```

3. Use the following section to test whether Cloudera Data Science Workbench can now detect GPUs.

Managing Projects in Cloudera Data Science Workbench

Test whether Cloudera Data Science Workbench can detect GPUs

Once Cloudera Data Science Workbench has successfully restarted, if NVIDIA drivers have been installed on the Cloudera Data Science Workbench hosts, Cloudera Data Science Workbench will now be able to detect the GPUs available on its hosts.



Additionally, the output of this command will also indicate that there are hosts with GPUs present.

```
cdsw status
```

Create a Custom CUDA-capable Engine Image



Note: Before you proceed, review the list of known issues and limitations associated with custom engines [here](#).

The base engine image (`docker.repository.cloudera.com/cdsw/engine:<version>`) that ships with Cloudera Data Science Workbench will need to be extended with CUDA libraries to make it possible to use GPUs in jobs and sessions.

The following sample Dockerfile illustrates an engine on top of which machine learning frameworks such as Tensorflow and PyTorch can be used. This Dockerfile uses a deep learning library from NVIDIA called [NVIDIA CUDA Deep Neural Network \(cuDNN\)](#). For detailed information about compatibility between NVIDIA driver versions and CUDA, refer the [cuDNN installation guide \(prerequisites\)](#).

When creating the Dockerfile for the custom image, you must delete the Cloudera repository that is inaccessible because of the paywall by running the following:

```
RUN rm /etc/apt/sources.list.d/*
```

Make sure you also check with the machine learning framework that you intend to use in order to know which version of cuDNN is needed. As an example, Tensorflow's NVIDIA hardware and software requirements for GPU support are listed in the [Tensorflow documentation here](#). Additionally, the Tensorflow version compatibility matrix for CUDA and cuDNN is documented [here](#).

The following sample Dockerfile uses NVIDIA's official Dockerfiles for [CUDA and cuDNN images](#).

cuda.Dockerfile

```
FROM docker.repository.cloudera.com/cdsw/engine:10
RUN rm /etc/apt/sources.list.d/*
RUN apt-get update && apt-get install -y --no-install-recommends \
gnupg2 curl ca-certificates && \
curl -fsSL
https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64/7fa2af80.pub
| apt-key add - && \
echo "deb https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64 /"
> /etc/apt/sources.list.d/cuda.list && \
echo "deb
https://developer.download.nvidia.com/compute/machine-learning/repos/ubuntu1804/x86_64
/" > /etc/apt/sources.list.d/nvidia-ml.list && \
apt-get purge --autoremove -y curl && \
```

```

rm -rf /var/lib/apt/lists/*

ENV CUDA_VERSION 10.1.243
LABEL com.nvidia.cuda.version="${CUDA_VERSION}"

ENV CUDA_PKG_VERSION 10-0=${CUDA_VERSION}-1
RUN apt-get update && apt-get install -y --no-install-recommends \
  cuda-cudart-${CUDA_PKG_VERSION} \
  cuda-libraries-${CUDA_PKG_VERSION} && \
  ln -s cuda-10.1 /usr/local/cuda && \
  rm -rf /var/lib/apt/lists/*

RUN echo "/usr/local/cuda/lib64" >> /etc/ld.so.conf.d/cuda.conf && \
  ldconfig

RUN echo "/usr/local/nvidia/lib" >> /etc/ld.so.conf.d/nvidia.conf && \
  echo "/usr/local/nvidia/lib64" >> /etc/ld.so.conf.d/nvidia.conf

ENV PATH /usr/local/nvidia/bin:/usr/local/cuda/bin:${PATH}
ENV LD_LIBRARY_PATH /usr/local/nvidia/lib:/usr/local/nvidia/lib64

RUN echo "deb
http://developer.download.nvidia.com/compute/machine-learning/repos/ubuntu1604/x86_64
/" > /etc/apt/sources.list.d/nvidia-ml.list

ENV CUDNN_VERSION 7.6.5.32
LABEL com.nvidia.cudnn.version="${CUDNN_VERSION}"

RUN apt-get update && apt-get install -y --no-install-recommends \
  libcudnn7=${CUDNN_VERSION}-1+cuda10.1 && \
  apt-mark hold libcudnn7 && \
  rm -rf /var/lib/apt/lists/*

```

You can now build a [custom engine image](#) out of `cuda.Dockerfile` using the following sample command:

```
docker build --network host -t <company-registry>/cdsw-cuda:10 . -f cuda.Dockerfile
```

Push this new engine image to a public Docker registry so that it can be made available for Cloudera Data Science Workbench workloads. For example:

```
docker push <company-registry>/cdsw-cuda:10
```

Site Admins: Add the Custom CUDA Engine to your Cloudera Data Science Workbench Deployment

Required CDSW Role: [Site Administrator](#)

After you've created the custom CUDA engine, a site administrator must add this new engine to Cloudera Data Science Workbench.

1. Sign in to Cloudera Data Science Workbench.
2. Click **Admin**.
3. Go to the **Engines** tab.
4. Under **Engine Images**, add the custom CUDA-capable engine image created in the previous step. This allows project administrators across the deployment to start using this engine in their jobs and sessions.
5. Site administrators can also set a limit on the maximum number of GPUs that can be allocated per session or job. From the **Maximum GPUs per Session/Job** dropdown, select the maximum number of GPUs that can be used by an engine.
6. Click **Update**.

Project Admins: Enable the CUDA Engine for your Project

Project administrators can use the following steps to make it the CUDA engine the default engine used for workloads within a particular project.

1. Navigate to your project's **Overview** page.

2. Click **Settings**.
3. Go to the **Engines** tab.
4. Under **Engine Image**, select the CUDA-capable engine image from the dropdown.

Test the CUDA Engine

You can use the following simple examples to test whether the new CUDA engine is able to leverage GPUs as expected.

1. Go to a project that is using the CUDA engine and click **Open Workbench**.
2. Launch a new session with GPUs.
3. Run the following command in the workbench command prompt to verify that the driver was installed correctly:

```
! /usr/bin/nvidia-smi
```

4. Use any of the following code samples to confirm that the new engine works with common deep learning libraries.

Pytorch

```
!pip3 install torch
from torch import cuda
assert cuda.is_available()
assert cuda.device_count() > 0
print(cuda.get_device_name(cuda.current_device()))
```

Tensorflow

```
!pip3 install tensorflow-gpu==2.1.0
from tensorflow.python.client import device_lib
assert 'GPU' in str(device_lib.list_local_devices())
device_lib.list_local_devices()
```

Keras

```
!pip3 install keras
from keras import backend
assert len(backend.tensorflow_backend._get_available_gpus()) > 0
print(backend.tensorflow_backend._get_available_gpus())
```

Web Applications Embedded in Cloudera Data Science Workbench

This topic describes how Cloudera Data Science Workbench allows you to embed web applications for frameworks such as Spark 2, TensorFlow, Shiny, and so on within sessions.

Many data science libraries and processing frameworks include user interfaces to help track progress of your jobs and break down workflows. These are instrumental in debugging and using the platforms themselves. For example, Spark provides a Spark Web UI to monitor running applications and TensorFlow visualizations can be run on TensorBoard. Other web application frameworks such as Shiny and Flask are popular ways for data scientists to display additional interactive analysis in the languages they already know.

Cloudera Data Science Workbench allows you to access these web UIs directly from sessions and jobs. This feature is particularly helpful when you want to monitor and track progress for batch jobs. Even though jobs don't give you access to the interactive workbench console, you can still track long running jobs through the UI. However, note that the UI is only active so long as the job/session is active. If your session times out after 60 minutes (default timeout value), so will the UI.

Cloudera Data Science Workbench exposes web applications for Spark and other machine learning frameworks as described here.

Spark 2 Web UIs (CDSW_SPARK_PORT)

Spark 2 exposes one web UI for each Spark application driver running in Cloudera Data Science Workbench. The UI will be running within the container, on the port specified by the environmental variable `CDSW_SPARK_PORT`. By default, `CDSW_SPARK_PORT` is set to **20049**. The web UI will exist only as long as a SparkContext is active within a session. The port is freed up when the SparkContext is shutdown.

Spark 2 web UI is available as a tab in the session, or alternatively in browsers at `https://spark- \langle CDSW_ENGINE_ID \rangle . \langle CDSW_DOMAIN \rangle` . For a running job, navigate to the **Job Overview** page and click the **History** tab. Click on the running job and select **Spark UI**.

TensorBoard, Shiny, and others (CDSW_APP_PORT or CDSW_READONLY_PORT)



Note: Previously, this section referred to `CDSW_PUBLIC_PORT`, which is deprecated as of Cloudera Data Science Workbench 1.6.0. Based on your use case, use `CDSW_APP_PORT` or `CDSW_READONLY_PORT` instead. However, the example below still uses `CDSW_PUBLIC_PORT`.

`CDSW_APP_PORT` and `CDSW_READONLY_PORT` are environment variables that point to general purpose public ports. Any HTTP services running in containers that bind to `CDSW_APP_PORT` or `CDSW_READONLY_PORT` are available in browsers at: `http:// \langle CDSW_ENGINE_ID \rangle . \langle CDSW_DOMAIN \rangle` . Therefore, TensorBoard, Shiny, Flask or any other web framework accompanying a project can be accessed directly from within a session or job, as long as it is run on `CDSW_APP_PORT` or `CDSW_READONLY_PORT`.

`CDSW_APP_PORT` is meant for applications that grant some level of control to the project, such as access to the active session or terminal. `CDSW_READONLY_PORT` must be used for applications that grant read-only access to project results.

To access the UI while you are in an active session, click the grid icon in the upper right hand corner of the Cloudera Data Science Workbench web application, and select the UI from the dropdown. For a job, navigate to the job overview page and click the **History** tab. Click on a job run to open the session output for the job. You can now click the grid icon in the upper right hand corner of the Cloudera Data Science Workbench web application to access the UI for this session.

Limitations with port availability

Cloudera Data Science Workbench exposes only one port per-access level. This means, in version 1.6.0, you can run a maximum of 3 web applications simultaneously:

- one on `CDSW_APP_PORT`, which can be used for applications that grant some level of control over the project to Contributors and Admins,
- one on `CDSW_READONLY_PORT`, which can be used for applications that only need to give read-only access to project collaborators,
- and, one on the now-deprecated `CDSW_PUBLIC_PORT`, which is accessible by all users.

However, by default the [editors](#) feature (introduced in version 1.6) runs third-party browser-based editors on `CDSW_APP_PORT`. Therefore, for projects that are already using browser-based third-party editors, you are left with only 2 other ports to run applications on: `CDSW_READONLY_PORT` and `CDSW_PUBLIC_PORT`. Keep in mind the level of access you want to grant users when you are selecting one of these ports for a web application.

Example: A Shiny Application

This example demonstrates how to create and run a Shiny application and view the associated UI while in an active session.

Create a new, blank project and run an R console. Create the files, `ui.R` and `server.R`, in the project, and copy the contents of the following example files provided by [Shiny by RStudio](#):

R

```
# ui.R
```

```
library(shiny)

# Define UI for application that draws a histogram
shinyUI(fluidPage(

  # Application title
  titlePanel("Hello Shiny!"),

  # Sidebar with a slider input for the number of bins
  sidebarLayout(
    sidebarPanel(
      sliderInput("bins",
                  "Number of bins:",
                  min = 1,
                  max = 50,
                  value = 30)
    ),

    # Show a plot of the generated distribution
    mainPanel(
      plotOutput("distPlot")
    )
  )
))
```

R

```
# server.R

library(shiny)

# Define server logic required to draw a histogram
shinyServer(function(input, output) {

  # Expression that generates a histogram. The expression is
  # wrapped in a call to renderPlot to indicate that:
  #
  # 1) It is "reactive" and therefore should re-execute automatically
  #    when inputs change
  # 2) Its output type is a plot

  output$distPlot <- renderPlot({
    x <- faithful[, 2] # Old Faithful Geyser data
    bins <- seq(min(x), max(x), length.out = input$bins + 1)

    # draw the histogram with the specified number of bins
    hist(x, breaks = bins, col = 'darkgray', border = 'white')
  })
})
```

Run the following code in the interactive workbench prompt to install the Shiny package, load the library into the engine, and run the Shiny application.

R

```
install.packages('shiny')

library('shiny')

runApp(port=as.numeric(Sys.getenv("CDSW_PUBLIC_PORT")), host="0.0.0.0",
       launch.browser="FALSE")
```

Finally, click the grid icon in the upper right hand corner of the Cloudera Data Science Workbench web application, and select the Shiny UI, **Hello Shiny!**, from the dropdown. The UI will be active as long as the session is still running.

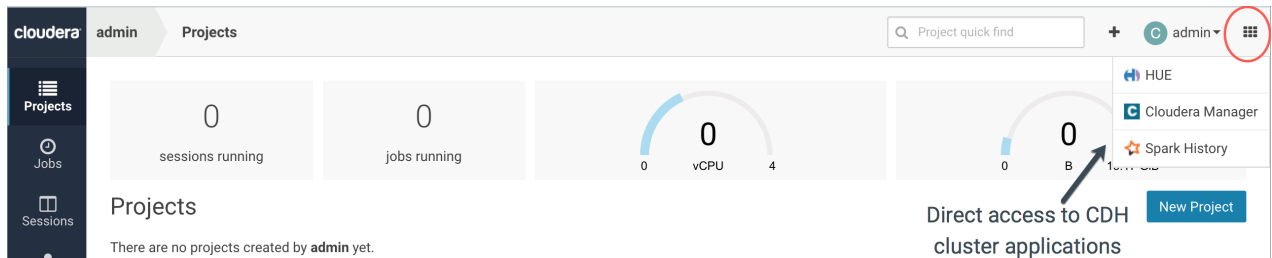
Accessing Web User Interfaces from Cloudera Data Science Workbench

This topic describes how to access user interfaces for applications such as Cloudera Manager, Hue, and so on, directly from the Cloudera Data Science Workbench UI .

Cloudera Manager, Hue, and the Spark History Server

Cloudera Data Science Workbench also gives you a way to access your CDH cluster's Cloudera Manager and Hue UIs from within the Cloudera Data Science Workbench application. Spark 2 provides a UI that displays information and logs for completed Spark applications, which is useful for debugging and performance monitoring. This UI, called the History Server, runs on the CDH cluster, on a configurable host and port.

To access these applications, click the grid icon in the upper right hand corner of the Cloudera Data Science Workbench web application, and select the UI you want to visit from the dropdown.



Running Distributed ML Workloads on YARN

Cloudera Data Science Workbench 1.6 (and higher) allows you to run distributed machine learning workloads on the CDH/HDP cluster with frameworks such as TensorFlowOnSpark, H2O, XGBoost, and so on. This is similar to what you can already do with [Spark workloads](#) that run on the attached CDH/HDP cluster.

To support this, Cloudera Data Science Workbench now forwards three extra ports from the host to each engine. The ports numbers for these ports are stored in the following environmental variables:

- CDSW_HOST_PORT_0
- CDSW_HOST_PORT_1
- CDSW_HOST_PORT_2

The engine's IP address is stored in CDSW_IP_ADDRESS and the host's IP address is stored in CDSW_HOST_IP_ADDRESS.

The information in these environmental variables can be used to make services running in the engine available to services running in the CDH cluster.

Example: H2O

The following shell script shows you how to use this new feature to run a distributed H2O workload. You can run this script in any active session.

```
#!/bin/bash
wget https://h2o-release.s3.amazonaws.com/h2o/rel-yates/4/h2o-3.24.0.4-cdh6.0.zip
unzip h2o-3.24.0.4-cdh6.0.zip
hadoop jar h2o-3.24.0.4-cdh6.0/h2odriver.jar \
-nodes 1 \
-mapperXmx 1g \
-extdriverif $CDSW_HOST_IP_ADDRESS \
-driverif $CDSW_IP_ADDRESS \
-driverport $CDSW_HOST_PORT_0 \
-disown
```

```
# Clean up
yarn application -kill \
$(yarn application -list 2>/dev/null | grep H2O | awk ' {print $1;}
')
```

Distributed Computing with Workers

Cloudera Data Science Workbench provides basic support for launching multiple engine instances, known as *workers*, from a single interactive session. Any R or Python session can be used to spawn workers. These workers can be configured to run a script (e.g. a Python file) or a command when they start up.

Workers can be launched using the `launch_workers` function. Other supported functions are `list_workers` and `stop_workers`. Output from all the workers is displayed in the workbench console of the session that launched them. These workers are terminated when the session exits.

Using Workers for Machine Learning

The simplest example of using this feature would involve launching multiple workers from a session, where each one prints 'hello world' and then terminates right after. To extend this example, you can remove the print command and configure the workers to run a more elaborate script instead. For example, you can set up a queue of parameters (inputs to a function) in your main interactive session, and then configure the workers to run a script that pulls parameters off the queue, applies a function, and keeps doing this until the parameter queue is empty. This generic idea can be applied to multiple real-world use-cases. For example, if the queue is a list of URLs and the workers apply a function that scrapes a URL and saves it to a database, CDSW can easily be used to do parallelized web crawling.

Hyperparameter optimization is a common task in machine learning, and workers can use the same parameter queue pattern described above to perform this task. In this case, the parameter queue would be a list of possible values of the hyperparameters of a machine learning model. Each worker would apply a function that trains a machine learning model. The workers run until the queue is empty, and save snapshots of the model and its performance.

Workers API

This section lists the functions available as part of the workers API.

Launch Workers

Launches worker engines into the cluster.

Syntax

```
launch_workers(n, cpu, memory, nvidia_gpu=0, kernel="python3", script="", code="",
env={})
```

Parameters

- **n (int)** - The number of engines to launch.
- **cpu (float)** - The number of CPU cores to allocate to the engine.
- **memory (float)** - The number of gigabytes of memory to allocate to the engine.
- **nvidia_gpu (int, optional)** - The number of GPU's to allocate to the engine.
- **kernel (str, optional)** - The kernel. Can be "r", "python2", "python3" or "scala".
- **script (str, optional)** - The name of a Python source file the worker should execute as soon as it starts up.
- **code (str, optional)** - Python code the engine should execute as soon as it starts up. If a script is specified, code will be ignored.
- **env (dict, optional)** - Environment variables to set in the engine.

Example Usage

Python

```
import cdsw
workers = cdsw.launch_workers(n=2, cpu=0.2, memory=0.5, code="print('Hello from a CDSW Worker')")
```

R

```
library("cdsw")
workers <- launch.workers(n=2, cpu=0.2, memory=0.5, env="", code="print('Hello from a CDSW Worker')")
```



Note: Due to a bug, the `env` parameter must be defined when calling the `launch.workers` function in R. If you do not wish to pass environment variables, simply set it to an empty string. When not defined, the `env` parameter is serialized internally into a format that is incompatible with Cloudera Data Science Workbench. This bug does not affect the Python engine.

List Workers

Returns all information on all the workers in the cluster.

Syntax

```
list_workers()
```

Await Workers

Waits for workers to either reach the `running` status, or to complete and exit.

Syntax

```
await_workers(ids, wait_for_completion=True, timeout_seconds=60)
```

Parameters

- `ids`: int or list of worker descriptions, optional The id's of the worker engines to stop or the worker's description dicts as returned by `launch_workers` or `list_workers`. If not provided, all workers in the cluster will be stopped.
- `wait_for_completion`: boolean, optional If True, will wait for all workers to exit successfully. If False, will wait for all workers to reach the `running` status. Defaults to True.
- `timeout_seconds`: int, optional Maximum number of seconds to wait for workers to reach the desired status. Defaults to 60. If equal to 0, there is no timeout. Workers that have not reached the desired status by the timeout will be returned in the `failures` key. See the return value documentation.

Returns

- `dict` - A dict with keys `workers` and `failures`. The `workers` key contains a list of dicts describing the workers that reached the desired status. The `failures` key contains a list of descriptions of the workers that did not.



Note: If `wait_for_completion` is False, the workers in the 'workers' key will contain a key called 'ip_address' which contains each worker's external IP address. This can be useful for running distributed frameworks on workers.

Stop Workers

Stops worker engines.

Syntax

```
stop_workers(*worker_id)
```

Parameter

- **worker_id (int, optional)** - The ID numbers of the worker engines that must be stopped. If an ID is not provided, all the worker engines on the cluster will be stopped.

Example: Worker Network Communications

Workers are a low-level feature to help use higher level libraries that can operate across multiple hosts. As such, you will generally want to use workers only to launch the backends for these libraries.

To help you get your workers or distributed computing framework components talking to one another, every worker engine run includes an environmental variable `CDSW_MASTER_IP` with the fully addressable IP of the master engine. Every engine has a dedicated IP access with no possibility of port conflicts.

For instance, the following are trivial examples of two worker engines talking to the master engine.

R

From the master engine, the following `master.r` script will launch two workers and accept incoming connections from them.

```
# master.r

library("cdsw")

# Launch two CDSW workers. These are engines that will run in
# the same project, execute a given code or script, and exit.
workers <- launch.workers(n=2, cpu=0.2, memory=0.5, env="", script="worker.r")

# Accept two connections, one from each worker. Workers will
# execute worker.r.
for(i in c(1,2)) {
  # Receive a message from each worker and return a response.
  con <- socketConnection(host="0.0.0.0", port = 6000, blocking=TRUE, server=TRUE,
open="r+")
  data <- readLines(con, 1)
  print(paste("Server received:", data))
  writeLines("Hello from master!", con)
  close(con)
}
```

The workers will execute the following `worker.r` script and respond to the master.

```
# worker.r

print(Sys.getenv("CDSW_MASTER_IP"))
con <- socketConnection(host=Sys.getenv("CDSW_MASTER_IP"), port = 6000, blocking=TRUE,
server=FALSE, open="r+")
write_resp <- writeLines("Hello from Worker", con)
server_resp <- readLines(con, 1)
print(paste("Worker received: ", server_resp))
close(con)
```

Python

From the master engine, the following `master.py` script will launch two workers and accept incoming connections from them.

```
# master.py

import cdsw, socket
```

```

# Launch two CDSW workers. These are engines that will run in
# the same project, execute a given code or script, and exit.
workers = cdsw.launch_workers(n=2, cpu=0.2, memory=0.5, script="worker.py")

# Listen on TCP port 6000
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(("0.0.0.0", 6000))
s.listen(1)

# Accept two connections, one from each worker. Workers will
# execute worker.py.
conn, addr = s.accept()
for i in range(2):
    # Receive a message from each worker and return a response.
    data = conn.recv(20)
    if not data: break
    print("Master received:", data)
    conn.send("Hello From Server!".encode())
conn.close()

```

The workers will execute the following `worker.py` script and respond to the master.

```

# worker.py

import os, socket

# Open a TCP connection to the master.
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((os.environ["CDSW_MASTER_IP"], 6000))

# Send some data and receive a response.
s.send("Hello From Worker!".encode())
data = s.recv(1024)
s.close()

print("Worker received:", data)

```

Collaborating on Projects with Cloudera Data Science Workbench

Cloudera Data Science Workbench supports several collaboration models.

Project Collaborators

If you want to work closely with trusted colleagues on a particular project, you can add them to the project as collaborators. For instructions, see [Adding Collaborators](#) on page 127.



Warning:

Collaborating Securely on Projects

Before adding project collaborators, you must remember that assigning the *Contributor* or *Admin* role to a project collaborator is the same as giving them write access to your data in CDH. This is because project contributors and project administrators have write access to all your project code (including any library code that you might not be actively inspecting). For example, a contributor/admin could modify project file(s) to insert code that deletes some data on the CDH cluster. The next time you launch a session and run the same code, it will appear as though you deleted the data yourself.

Additionally, project collaborators also have access to all actively running sessions and jobs. This means that a malicious user can easily impersonate you by accessing one of *your* active sessions. Therefore, it is extremely important to restrict project access to trusted collaborators only. Note that Cloudera Data Science Workbench 1.4.3 introduces a new feature that allows site administrators to restrict this ability by allowing only session creators to execute commands within their own active sessions. For details, see [Restricting Access to Active Sessions](#).

For these reasons, Cloudera recommends using Git to collaborate securely on shared projects. This will also help avoid file modification conflicts when your team is working on more elaborate projects.

Restricting Collaborator and Administrator Access to Active Sessions

Required Role: [Site Administrator](#)

By default, the following Cloudera Data Science Workbench users have the ability to **execute commands** within any active sessions you have created:

- All Site Administrators
- Users who have been assigned Admin or Contributor privileges for the project where the session is created.
- For team projects, Team Admins have complete access to all team projects and any active sessions running within these projects. Additionally, any team members who have been assigned the Admin or Contributor roles for your projects will also have the ability to execute commands within your active sessions.

Starting with Cloudera Data Science Workbench 1.4.3, site administrators can now restrict this ability by allowing only the user who launched the session to execute commands within their own active sessions. To enable this restriction:

1. Log into Cloudera Data Science Workbench with site administrator privileges.
2. Click **Admin > Security**.
3. Under the **General** section, select the checkbox to enable the **Only session creators can execute commands on active sessions** property.

When this property is enabled, only the user that creates a session will be able to execute commands in that session. No other users, regardless of their permissions in the team or as project collaborators, will be able to execute commands on active sessions that are not created by them. Even site administrators will not be able to execute commands in other users' active sessions. However, keep in mind that all site administrators still have access to the Site Administrator dashboard and can reverse this change at any time.

Teams

Users who work together on more than one project and want to facilitate collaboration can create a Team. Teams allow streamlined administration of projects. Team projects are owned by the team, rather than an individual user. Team administrators can add or remove members at any time, assigning each member different permissions.

For more details, see:

- [Creating a Team](#) on page 122
- [Modifying Team Account Settings](#) on page 123

Sharing Personal Projects

When you create a project in your personal context, Cloudera Data Science Workbench asks you to assign one of the following visibility levels to the project - Private or Public. Public projects on Cloudera Data Science Workbench grant read-level access to *everyone* with access to the Cloudera Data Science Workbench application. That means everyone can view the project's files and results, but only those whom you have explicitly [added as a collaborator](#) can edit files, run engines, or view the project's environment variables.

You can include a markdown-formatted `README.md` file in public projects to document your project's purpose and usage.

If you are a project admin, you can set a project's visibility to **Public** from the **Project > Settings > Options** page. For instructions, see [Modifying Project Settings](#) on page 128.

Forking Projects

You can fork another user's project by clicking **Fork** on the **Project** page. Forking creates a new project under your account that contains all the files, libraries, configuration, and jobs from the original project.

Creating sample projects that other users can fork helps to bootstrap new projects and encourage common conventions.



Note: An issue exists where a timeout might occur when forking large projects.

Collaborating with Git

Cloudera Data Science Workbench provides seamless access to Git projects. Whether you are working independently, or as part of a team, you can leverage all of benefits of version control and collaboration with Git from within Cloudera Data Science Workbench. Teams that already use Git for collaboration can continue to do so. Each team member will need to create a separate Cloudera Data Science Workbench project from the central Git repository.

For anything but simple projects, Cloudera recommends using Git for version control. You should work on Cloudera Data Science Workbench the same way you would work locally, and for most data scientists and developers that means using Git.

For more details, see [Using Git to Collaborate on Projects](#) on page 152.

Sharing Job and Session Console Outputs

Cloudera Data Science Workbench lets you easily share the results of your analysis with one click. Using rich visualizations and documentation comments, you can arrange your console log so that it is a readable record of your analysis and results. This log continues to be available even after the session stops. This method of sharing allows you to show colleagues and collaborators your progress without your having to spend time creating a report.

Collaborating on Projects with Cloudera Data Science Workbench

To share results from an interactive session, click **Share** at the top of the console page. From here you can generate a link that includes a secret token that gives access to that particular console output. For jobs results, you can either share a link to the latest job result or a particular job run. To share the latest job result, click the **Latest Run** link for a job on the Overview page. This link will always have the latest job results. To share a particular run, click on a job run in the job's **History** page and share the corresponding link.

You can share console outputs with one of the following sets of users.

- **All anonymous users with the link** - By default, Cloudera Data Science Workbench allows anonymous access to shared consoles. However, site administrators can disable anonymous sharing at any time by going to **Admin > Security**, disabling the **Allow anonymous access to shared console outputs** checkbox, and clicking **Disable anonymous access** to confirm.

Once anonymous sharing has been disabled, all existing publicly shared console outputs will be updated to be viewable only by authenticated users.

- **All authenticated users with the link** - This means any user with a Cloudera Data Science Workbench account will have access to the shared console.
- **Specific users and teams** - Click **Change** to search for users and teams to give access to the shared console. You can also come back to the session and revoke access from a user or team the same way.

Sharing Data Visualizations

If you want to share a single data visualization rather than an entire console, you can embed it in another web page. Click the small circular 'link' button located to the left of most rich visualizations to view the HTML snippet that you can use to embed the visualization.

Using Git to Collaborate on Projects

Cloudera Data Science Workbench provides seamless access to Git projects. Whether you are working independently, or as part of a team, you can leverage all of benefits of version control and collaboration with Git from within Cloudera Data Science Workbench. Teams that already use Git for collaboration can continue to do so. Each team member will need to create a separate Cloudera Data Science Workbench project from the central Git repository.

For anything but simple projects, Cloudera recommends using Git for version control. You should work on Cloudera Data Science Workbench the same way you would work locally, and for most data scientists and developers that means using Git.

Cloudera Data Science Workbench does not include significant UI support for Git, but instead allows you to use the full power of the command line. If you run an engine and open a [terminal](#), you can run any Git command, including `init`, `add`, `commit`, `branch`, `merge` and `rebase`. Everything should work exactly as it does locally, except that you are running on a distributed edge host directly connected to your Apache Hadoop cluster.

Importing a Project From Git

When you create a project, you can optionally supply an HTTPS or SSH Git URL that points to a remote repository. The new project is a clone of that remote repository. You can commit, push and pull your code by running a console and opening a [terminal](#).

Using SSH - If you want to use SSH to clone the repo, you will need to first add your personal Cloudera Data Science Workbench SSH key to your GitHub account. For instructions, see [Adding SSH Key to GitHub](#) on page 302.

If you see Git commands hanging indefinitely, check with your cluster administrators to make sure that the SSH ports on the Cloudera Data Science Workbench hosts are not blocked.

Linking an Existing Project to a Git Remote

If you did not create your project from a Git repository, you can link an existing project to a Git remote (for example, `git@github.com:username/repo.git`) so that you can push and pull your code.

To link to a Git remote:

1. Launch a new session.
2. Open a [terminal](#).
3. Enter the following commands:

Shell

```
git init
git add *
git commit -a -m 'Initial commit'
git remote add origin git@github.com:username/repo.git
```

You can run `git status` after `git init` to make sure your `.gitignore` includes a folder for libraries and other non-code artifacts.

Editors

In addition to the native Cloudera Data Science Workbench editor, you can configure Cloudera Data Science Workbench to work with third-party, browser-based IDEs such as Jupyter and also certain local IDEs that run on your machine, such as PyCharm. In the Cloudera Data Science Workbench documentation, browser-based IDEs such as Jupyter and RStudio are referred to as **browser IDEs**, whereas IDEs such as PyCharm that run on your local machine outside the browser are referred to as **local IDEs**.

You can use the browser or local IDE of your choice to edit and run code interactively. When you bring your own editor, you still get many of the benefits of Cloudera Data Science Workbench behind an editor interface you are familiar with:

- Dependency management that lets you share code with confidence
- CDH client configurations
- Automatic Kerberos authentication through Cloudera Data Science Workbench
- Reuse code in other Cloudera Data Science Workbench features such as experiments and jobs
- Collaboration features such as teams
- Compliance with IT rules for where compute, data, and/or code must reside. For example, compute occurs within the Cloudera Data Science Workbench deployment, not the local machine. Browser IDEs run within a Cloudera Data Science Workbench session and follow all the same compliance rules. Local IDEs, on the other hand, can bring data or code to a user's machine. Therefore, Site Administrators can opt to disable local IDEs to balance user productivity with compliance concerns.

Note that you can only edit and run code interactively with the IDEs. Tasks such as creating a project or deploying a model require the Cloudera Data Science Workbench web UI and cannot be completed through an editor.

The configuration for an IDE depends on which type of editor you want to use:

Workbench editor

The Workbench editor is the built-in editor for Cloudera Data Science Workbench. No additional configuration is required to use it. When you launch a session, select the Workbench editor.

Third-party, browser-based IDEs

Browser IDEs are editors such as Jupyter or RStudio. When you use a browser IDE, it runs within a session and allows you to edit and run code interactively. Changes that you make in the editor are propagated to the Cloudera Data Science Workbench project. Base Engine Image v8 ships with Jupyter preconfigured as a browser IDE. You can select it when you start a session or add a different browser IDE. For more information, see [Configure a Browser IDE as an Editor](#) on page 155.

Keep the following in mind when using browser IDEs:

- **Engine Version Requirements**
 - Browser-based IDEs that are [configured using custom engines](#) require Base Engine Image v8 or higher.
 - Browser-based IDEs that are [configured directly within individual projects](#) do not require a specific engine image. However, Cloudera recommends you use the latest engine image.
- When you are finished using a browser IDE, you must exit the IDE properly, including saving your work if necessary. Do not just stop the Cloudera Data Science Workbench session. Doing so will cause you to lose your session state.
- Depending on the behavior of the browser IDE, multiple users within a project may overwrite each other's state.
- Browser IDEs do not adhere to the timeout set in `IDLE_MAXIMUM_MINUTES`. Instead, they use the timeout set in `SESSION_MAXIMUM_MINUTES`, which is 7 days by default. Cloudera recommends that users stop their session manually after using a browser-based editor. Running sessions continue to consume resources and may impact other users.
- Logs for browser IDEs are available on the **Logs** tab of the session window. This includes information that the IDE may generate, such as error messages, in addition to any Cloudera Data Science Workbench logs.

Local IDE Editors on your machine that can use SSH-based remote editing

These editors, referred to as Local IDEs in the documentation, are editors such as PyCharm that run on your local machine. They connect to the Cloudera Data Science Workbench with an SSH endpoint and allow you to edit and run code interactively. You must manually configure some sort of file sync and ignore list between your local machine and Cloudera Data Science Workbench. You can use functionality within the local IDE, such as PyCharm's sync, or external tools that can sync via the SSH endpoint, such as mutagen.

Keep the following in mind before setting up local IDEs:

- Local IDEs do not require a specific engine image, but Cloudera recommends you use the latest engine image.
- Site Administrators should work with IT to determine the data access policies for your organization. For example, your data policy may not allow users to sync certain files to their machines from Cloudera Data Science Workbench. Verify that users understand the requirements and adhere to them when configuring their file sync behavior.
- Users should ensure that the IDEs they want to use support SSH. For example, VS Code supports "remote development over SSH," and PyCharm supports using a "remote interpreter over SSH."

For more information, see [Configure a SSH Gateway to Use Local IDEs](#) on page 160.

Configure a Browser IDE as an Editor

When you use a browser IDE, changes that you make in the editor are propagated to the Cloudera Data Science Workbench project. For example, if you create a new `.py` file or modify an existing one with the third-party editor, the changes are propagated to Cloudera Data Science Workbench. When you run the code from the notebook, execution is pushed from the notebook to Cloudera Data Science Workbench.

Base Engine Image v8 (and later) comes preconfigured with Jupyter. Jupyter can be selected in place of the built-in Workbench editor when you launch a session, and no additional configuration is required.



Note: If you create a customized engine image by extending the CDSW base image, Jupyter Notebook will still be installed on this customized engine image. However, CDSW will not automatically list Jupyter Notebook in the dropdown list of editors on the **Launch New Session** page in projects that are configured to use this customized engine image. You must configure the custom engine image to use Jupyter Notebook. For details, see [Configure Jupyter Notebook in a Customized Engine Image](#) on page 160.

You can configure additional IDEs to be available from the dropdown. You have two configuration options:

- **Project Level:** You can configure an editor at the project level so that any session launched within that project can use the editor configured. Other projects across the deployment will not be able to use any editors configured in such a manner. For steps, see [Configure a Browser IDE at the Project Level](#) on page 156.
- **Engine Level:** You can create a custom engine configured with the editor so that any project across the deployment that uses this custom engine can also use the editor configured. This might be the only option in case of certain browser IDEs (such as RStudio) that require root permission to install and therefore cannot be directly installed within the project. For steps, see [Configure a Browser IDE at the Engine Level](#) on page 157.

Cloudera recommends you first test the browser IDE you intend to install in a session before you install it to the project or build a custom engine with it. For steps, see [Test a Browser IDE in a Session Before Installation](#) on page 155.

Test a Browser IDE in a Session Before Installation

This process can be used to ensure that a browser IDE works as expected before you install it to a project or to a customized engine image. This process is not meant for browser IDEs that require root permission to install, such as RStudio.

These steps are only required if you want to use an editor that does not come pre-installed as part of the default engine image. Perform the following steps to configure an editor for your session:

1. Ensure that your browser accepts pop-up windows and cookies from Cloudera Data Science Workbench web UI.
2. Open the Cloudera Data Science Workbench web UI.
3. Go to your project and launch a session with the kernel of your choice and the **Workbench** editor. Alternatively, open an existing session.
4. In the interactive command prompt or terminal for the session, install the editor you want to use. See the documentation for your editor for specific instructions.

For example:

Jupyter Lab

Python 2

The following example command installs Jupyter Lab for Python 2:

```
!pip install jupyterlab
```

Python 3

The following example command installs Jupyter Lab for Python 3:

```
!pip3 install jupyterlab
```

5. After the installation completes, enter the command to start the server for the notebook on the port specified in the **CDSW_APP_PORT** environment variable on IP address 127.0.0.1.

For example, the following command starts the server for Jupyter Lab on the port specified in the **CDSW_APP_PORT** environment variable:

```
!/home/cdsw/.local/bin/jupyter-lab --no-browser --ip=127.0.0.1 --port=${CDSW_APP_PORT} --NotebookApp.token= --NotebookApp.allow_remote_access=True --log-level=ERROR
```

6. Click on the grid icon in the top right.
You should see the editor in the drop-down menu. If you select the editor, it opens in a new browser tab.

Configure a Browser IDE at the Project Level



Note: The following steps are only required if you want to use an editor that does not come pre-installed as part of the default engine image that Cloudera Data Science Workbench ships with.

Perform the following steps to configure an editor at the project level:

1. **(Recommended)** [Test a Browser IDE in a Session Before Installation](#) on page 155
2. Install the IDE of your choice to the project. For information about how to install additional packages to a project, see [Installing Additional Packages](#) on page 218.
3. Open the Cloudera Data Science Workbench web UI.
4. Go to the project you want to configure an editor for.
5. Go to **Settings > Editors** and click **New Editor**.
6. Complete the fields:
 - **Name:** Provide a name for the editor. This is the name that appears in the dropdown menu for **Editors** when you start a new session.
 - **Command:** Enter the command to start the server for the editor on the Cloudera Data Science Workbench public port specified in the **CDSW_APP_PORT** environment variable (default 8081).

For example, the following command starts Jupyter Lab on the port specified by the `CDSW_APP_PORT` environment variable:

```
/home/cdsw/.local/bin/jupyter-lab --no-browser --ip=127.0.0.1 --port=${CDSW_APP_PORT}
--NotebookApp.token= --NotebookApp.allow_remote_access=True --log-level=ERROR
```

This is the same command you used to start the IDE to test it in a session.

7. Save the changes.

When a user starts a new session, the editor you added is available in the list of editors. Browsers must be configured to accept cookies and allow pop-up windows from the Cloudera Data Science Workbench web UI.

Configure a Browser IDE at the Engine Level



Note: The following steps are only required if you want to use an editor that does not come pre-installed as part of the default engine image that Cloudera Data Science Workbench ships with.

You can make a browser IDE available to *any project* within a Cloudera Data Science Workbench deployment by creating a customized engine image, installing the editor to it, and then whitelisting the custom image for projects as needed. Additionally, browser IDEs that require root permission to install, such as RStudio, can only be used as part of a customized engine image.

When a user launches a session, they can select the customized engine with the editors available. The following steps describe how to build a customized engine image for RStudio:

1. Create a Dockerfile for the new custom image. Note that the base engine image uses Ubuntu.

The following sample Dockerfile is for RStudio:

```
FROM docker.repository.cloudera.com/cdsw/engine:10

WORKDIR /tmp

#The RUN commands that install an editor
#For example: RUN apt-get install myeditor

RUN apt-get update && \
    apt-get install -y --no-install-recommends \
        libapparmor1 \
        libclang-dev \
        lsb-release \
        psmisc \
        sudo && \
    apt-get clean && \
    apt-get autoremove && \
    rm -rf /var/lib/apt/lists/*

RUN wget --quiet
https://download2.rstudio.org/server/bionic/amd64/rstudio-server-1.2.5033-amd64.deb && \
    dpkg -i rstudio-server-1.2.5033-amd64.deb && \
    rm rstudio-server-1.2.5033-amd64.deb

COPY rserver.conf /etc/rstudio/rserver.conf

COPY rstudio-cdsw /usr/local/bin/rstudio-cdsw

RUN chmod +x /usr/local/bin/rstudio-cdsw
```

2. Create `rserver.conf`:

```
# Must match CDSW_APP_PORT
www-port=8090
server-app-armor-enabled=0
server-daemonize=0
```

```
www-address=127.0.0.1
auth-none=1
auth-validate-users=0
```

Make sure that the `www-port` property matches the port set in the `CDSW_APP_PORT` environment variable (default 8090).

3. Create `rstudio-cdsw`:

```
#!/bin/bash

# This saves RStudio's user runtime information to /tmp, which ensures several
# RStudio sessions can run in the same project simultaneously
mkdir -p /tmp/rstudio/sessions/active
mkdir -p /home/cdsw/.rstudio/sessions
if [ -d /home/cdsw/.rstudio/sessions/active ]; then rm -rf
/home/cdsw/.rstudio/sessions/active; fi
ln -s /tmp/rstudio/sessions/active /home/cdsw/.rstudio/sessions/active

# This ensures RStudio picks up the environment. This may not be necessary if
# you are installing RStudio Professional. See
# https://docs.rstudio.com/ide/server-pro/r-sessions.html#customizing-session-launches.
# SPARK_DIST_CLASSPATH is treated as a special case to workaroud a bug in R
# with very long environment variables.
env | grep -v ^SPARK_DIST_CLASSPATH >> /usr/local/lib/R/etc/Renviron.site
echo "Sys.setenv(\"SPARK_DIST_CLASSPATH\"=\"\${SPARK_DIST_CLASSPATH}\")" >>
/usr/local/lib/R/etc/Rprofile.site

# Now start RStudio
/usr/sbin/rstudio-server start
```

4. Build the Dockerfile:

```
docker build -t <image-name>:<tag> . -f Dockerfile
```

If you want to build your image on a Cloudera Data Science Workbench gateway host, you must add the `--network=host` option to the build command:

```
docker build --network=host -t <image-name>:<tag> . -f Dockerfile
```

5. Distribute the image:

- Push the image to a public registry such as DockerHub.

For instructions, refer the Docker documentation: [docker push](#).

- Push the image to your company's Docker registry.

When using this method, make sure to tag your image with the following schema:

```
docker tag <image-name> <company-registry>/<user-name>/<image-name>:<tag>
```

Once the image has been tagged properly, use the following command to push the image:

```
docker push <company-registry>/<user-name>/<image-name>:<tag>
```

- Distribute the image manually:

1. Save the docker image as a tarball on the host where it was built

```
docker image save -o ./<new_customized_engine>.tar <image-name>
```

- Distribute the image to *all* the Cloudera Data Science Workbench gateway hosts.

```
scp ./<new_customized_engine>.tar root@<cdsw.your_company.com>:/tmp/
```

- Load the image on *all* the Cloudera Data Science Workbench gateway hosts.

```
docker load --input /tmp/./<new_customized_engine>.tar
```

- To verify that the image was successfully distributed and loaded, run:

```
docker images
```

- Whitelist the image in Cloudera Data Science Workbench:

- Log in to the Cloudera Data Science Workbench web UI as a site administrator.
- Click **Admin > Engines**.
- Add `<company-registry>/<user-name>/<image-name>:<tag>` to the list of whitelisted engine images.

- Whitelist the new engine for a project:

- Go to the project **Settings** page.
- Click **Engines**.
- Select the new engine from the dropdown list of available Docker images. This engine will now be used to launch sessions within this project.

- Configure project(s) to use RStudio. When this is done, you will be able to select RStudio from the dropdown list of editors on the **Launch New Session** page. There are two ways to do this: for an individual project, or for all projects that use this engine.

Configure RStudio for an individual project

- Go to the project **Settings > Editors**.
- Click **New Editor**.
- Complete the fields:
 - Name:** Provide a name for the editor. For example, **RStudio**. This is the name that appears in the dropdown menu for **Editors** when you start a new session.
 - Command:** Enter the command to start the server for the editor.

For example, the following command will start RStudio:

```
/usr/local/bin/rstudio-cdsw
```

- Click **Save**.

Configure RStudio for all projects that use this engine

- Log in to the Cloudera Data Science Workbench web UI as a site administrator.
- Click **Admin > Engines**.
- Under **Engine Images**, click the **Edit** button for the engine image that you whitelisted here in a previous step.
- Click **New Editor**.
 - Name:** Provide a name for the editor. For example, **RStudio**. This is the name that appears in the dropdown menu for **Editors** when you start a new session.
 - Command:** Enter the command to start the server for the editor.

For example, the following command will start RStudio:

```
/usr/local/bin/rstudio-cdsw
```

- e) Click **Save**, then click **Save** again.

For more information about how to create a customized engine image and limitations, see [Customized Engine Images](#) on page 220

Configure Jupyter Notebook in a Customized Engine Image

CDSW's base Image v8 (and later) come with Jupyter Notebook pre-installed on them. If you create a customized engine image by extending this base image, Jupyter Notebook will still be installed on this customized engine image. However, CDSW will not automatically list Jupyter Notebook in the dropdown list of editors on the **Launch New Session** page in projects that are configured to use this customized engine image. You must use the following steps to configure the custom engine image to use Jupyter Notebook.

1. Log in to the Cloudera Data Science Workbench web UI as a site administrator.
2. Click **Admin > Engines**.
3. Under **Engine Images**, click the **Edit** button for the customized engine image that you want to configure for Jupyter Notebook.
4. Click **New Editor**.
 - **Name:** Enter **Jupyter Notebook**. This is the name that appears in the dropdown menu for **Editors** when you start a new session.
 - **Command:** Enter the command to start Jupyter Notebook.

```
/usr/local/bin/jupyter-notebook --no-browser --ip=127.0.0.1 --port=${CDSW_APP_PORT}
--NotebookApp.token= --NotebookApp.allow_remote_access=True --log-level=ERROR
```

5. Click **Save**, then click **Save** again.

Configure a SSH Gateway to Use Local IDEs

Cloudera Data Science Workbench relies on the SSH functionality of the local IDEs to connect to the SSH endpoint on your local machine created with the `cdswctl` client. Users establish an SSH endpoint on their machine with the `cdswctl` client. This endpoint acts as the bridge that connects the IDE on your machine and the Cloudera Data Science Workbench deployment.

Configure and Use a Local IDE

The specifics for how to configure a local IDE to work with Cloudera Data Science Workbench are dependent on the local IDE you want to use. The following steps are a high-level description of the steps a user must complete:

1. Establish an SSH endpoint with the [cdswctl Command Line Interface Client](#) on page 313.
2. Configure the local IDE to use Cloudera Data Science Workbench as the remote interpreter.
3. Optionally, sync files with tools (like mutagen, SSHFS, or the functionality built into your IDE) from Cloudera Data Science Workbench to your local machine. Ensure that you adhere to IT policies.
4. Edit the code in the local IDE and run the code interactively on Cloudera Data Science Workbench.
5. Sync the files you edited locally to Cloudera Data Science Workbench.
6. Use the Cloudera Data Science Workbench web UI to perform actions such as deploying a model that uses the code you edited.

You can see an end-to-end example for PyCharm configuration here: [Configure PyCharm as a Local IDE](#) on page 160.

Configure PyCharm as a Local IDE

Cloudera Data Science Workbench supports using local IDEs on your machine that allow remote execution and/or file sync over SSH, such as PyCharm. This topic describes the tasks you need to perform to configure Cloudera Data Science Workbench to act as a remote SSH interpreter for PyCharm. Once finished, you can use PyCharm to edit and sync the

changes to Cloudera Data Science Workbench. To perform actions such as deploying a model, use the Cloudera Data Science Workbench web UI.



Note: These instructions were written for the Professional Edition of PyCharm Version 2019.1. See the documentation for your version of PyCharm for specific instructions.

Before you begin, ensure that the following prerequisites are met:

- You have an edition of PyCharm that supports SSH, such as the Professional Edition.
- You have an SSH public/private key pair for your local machine that is compatible with PyCharm. If you use OpenSSH to generate the key, include the `-m PEM` option because PyCharm does not support modern (RFC 4716) OpenSSH keys.
- You have Contributor permissions for an existing Cloudera Data Science project. Alternatively, create a new project you have access to.

Download `cdswctl` and Add an SSH Key

1. Open the Cloudera Data Science Workbench web UI and go to **Settings > Remote Editing** for your user account.
2. Download `cdswctl` client for your operating system.

If you are using the macOS executable, `cdswctl` will be unsigned and therefore cannot be launched on the recent version of macOS without performing the following additional steps:

- a) In the Finder on your Mac and locate the app you want to open.

Don't use Launchpad to do this. Launchpad doesn't allow you to access the shortcut menu.

- b) Control-click the app icon, then choose **Open** from the shortcut menu.
- c) Click **Open**.

The app is saved as an exception to your security settings, and you can open it in the future by double-clicking it just as you can any registered app.

3. In the terminal, run `cat ~/.ssh/id_rsa.pub`. If you used a different filename above when generating the key, use that filename instead. This command prints the key as a string.
4. Copy the key. It should resemble the following: `ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCh2J5mW3i3BgtZ25/F0sxywpLVkx1RgmZunI`
5. In **SSH public keys for session access**, paste the key.

Cloudera Data Science Workbench uses the SSH public key to authenticate your CLI client session, including the SSH endpoint connection to the Cloudera Data Science Workbench deployment. Any SSH endpoints that are running when you add an SSH public key must also be restarted.

Initialize an SSH Connection to Cloudera Data Science Workbench

The following task describes how to establish an SSH endpoint for Cloudera Data Science Workbench. Creating an SSH endpoint is the first step to configuring a remote editor for Cloudera Data Science Workbench.

1. Log in to Cloudera Data Science Workbench with the CLI client. Depending on your deployment, make sure you add `http` or `https` to the URL as shown below:

```
cdswctl login -n <username> -u http(s)://cdsw.your_domain.com
```

For example, the following command logs the user `sample_user` into the `https://cdsw.your_domain.com` deployment:

```
cdswctl login -n sample_user -u https://cdsw.your_domain.com
```

2. Create a local SSH endpoint to Cloudera Data Science Workbench. Run the following command:

```
cdswctl ssh-endpoint -p <username>/<project_name> [-c <CPU_cores>] [-m <memory_in_GB>]
[-g <number_of_GPUs>]
```

The command uses the following defaults for optional parameters:

- CPU cores: 1
- Memory: 1 GB
- GPUs: 0

For example, the following command starts a session for the logged-in user `sample_user` under the `customerchurn` project with .5 cores, .75 GB of memory, 0 GPUs, and the Python3 kernel:

```
cdswctl ssh-endpoint -p customerchurn -c 0.5 -m 0.75
```

To create an SSH endpoint in a project owned by another user or a team, for example `finance`, prepend the username to the project and separate them with a forward slash:

```
cdswctl ssh-endpoint -p finance/customerchurn -c 0.5 -m 0.75
```

This command creates session in the project `customerchurn` that belongs to the team `finance`.

Information for the SSH endpoint appears in the output:

```
...
You can SSH to it using
    ssh -p <some_port> cdsw@localhost
...
```

3. Open a new command prompt and run the outputted command from the previous step:

```
ssh -p <some_port> cdsw@localhost
```

For example:

```
ssh -p 9750 cdsw@localhost
```

You will be prompted for the passphrase for the SSH key you entered in the Cloudera Data Science web UI.

Once you are connected to the endpoint, you are logged in as the `cdsw` user and can perform actions as though you are accessing the terminal through the Cloudera Data Science Workbench web UI.

4. Test the connection.

If you run `ls`, the project files associated with the session you created are shown. If you run `whoami`, the command returns the `cdsw` user.

Add Cloudera Data Science Workbench as an Interpreter for PyCharm

Before you begin, ensure that the SSH endpoint for Cloudera Data Science Workbench is running on your local machine. In PyCharm, you can configure an SSH interpreter. Cloudera Data Science Workbench uses this method to connect to PyCharm and act as its interpreter. These instructions were written for the Professional Edition of PyCharm Version 2019.1 and are meant as a starting point. If additional information is required, see the documentation for your version of PyCharm for specific instructions.

1. Verify that the SSH endpoint for Cloudera Data Science Workbench is running with `cdswctl`. If the endpoint is not running, start it.
2. Open PyCharm.
3. Create a new project.
4. Expand **Project Interpreter** and select **Existing interpreter**.

5. Click on ... and select **SSH Interpreter**

6. Select **New server configuration** and complete the fields:

- **Host:** localhost
- **Port:** `<port_number>`

This is the port number provided by `cdswctl`.

- **Username:** `cdsw`

7. Select **Key pair** and complete the fields using the RSA private key that corresponds to the public key you added to the **Remote Editing** tab in the Cloudera Data Science Workbench web UI.

For macOS users, you must add your RSA private key to your keychain. In a terminal window, run the following command:

```
ssh-add -K <path to your private key>/<private_key>
```

8. Complete the wizard. Based on the Python version you want to use, enter one of the following parameters:

- For Python 2: `/usr/local/bin/python`
- For Python 3: `/usr/local/bin/python3`

You are returned to the **New Project** window. **Existing interpreter** is selected, and you should see the connection to Cloudera Data Science Workbench in the **Interpreter** field.

9. In the **Remote project location** field, specify the following directory:

```
/home/cdsw
```

10 Create the project.

(Optional) Configure the Sync Between Cloudera Data Science Workbench and PyCharm

Before you configure syncing behavior between the remote editor and Cloudera Data Science Workbench, ensure that you understand the policies set forth by IT and the Site Administrator. For example, a policy might require that data remains within the Cloudera Data Science Workbench deployment but allow you to download and edit code. Configuring what files PyCharm ignores can help you adhere to IT policies.

1. In your project, go to **Preferences**.

Depending on your operating system, **Preferences** may be called **Settings**.

2. Go to **Build, Execution, Deployment** and select **Deployment**.

3. On the **Connection** tab, add the following path to the **Root path** field:

```
/home/cdsw
```

4. On the **Excluded Paths** tab, add any paths you want to exclude.

Cloudera recommends excluding the following paths at a minimum:

- `/home/cdsw/.local`
- `/home/cdsw/.cache`
- `/home/cdsw/.ipython`
- `/home/cdsw/.ipython`
- `/home/cdsw/.oracle_jre_usage`
- `/home/cdsw/.pip`
- `/home/cdsw/.pycharm_helpers`

5. Optionally, add a **Deployment path** on the **Mappings** tab if the code for your Cloudera Data Science Workbench project lives in a subdirectory of the root path.

6. Expand **Deployment** in the left navigation and go to **Options > Upload changed files automatically to the default server** and set the behavior to adhere to the policies set forth by IT and the Site Administrator.

Cloudera recommends setting the behavior to **Automatic upload** because the data remains on the cluster while your changes get uploaded.

7. Sync for the project file(s) to your machine and begin editing.

Importing Data into Cloudera Data Science Workbench

Cloudera Data Science Workbench allows you to run analytics workloads on data imported from local files, Apache HBase, Apache Kudu, Apache Impala, Apache Hive or other external data stores such as Amazon S3.

Accessing Local Data from Your Computer

If you want to perform analytics operations on existing data files (.csv, .txt, etc.) from your computer, you can upload these files directly to your Cloudera Data Science Workbench project. Go to the project's Overview page. Under the Files section, click **Upload** and select the relevant data files to be uploaded.

The following sections use the [tips.csv](#) dataset to demonstrate how to work with local data stored within your project. Upload this dataset to the `data` folder in your project before you run these examples.

Pandas (Python)

```
import pandas as pd

tips = pd.read_csv('data/tips.csv')

tips \
    .query('sex == "Female"') \
    .groupby('day') \
    .agg({'tip' : 'mean'}) \
    .rename(columns={'tip': 'avg_tip_dinner'}) \
    .sort_values('avg_tip_dinner', ascending=False)
```

dplyr (R)

```
library(readr)
library(dplyr)

# load data from .csv file in project
tips <- read_csv("data/tips.csv")

# query using dplyr
tips %>%
  filter(sex == "Female") %>%
  group_by(day) %>%
  summarise(
    avg_tip = mean(tip, na.rm = TRUE)
  ) %>%
  arrange(desc(avg_tip))
```

Accessing Data from HDFS

There are many ways to access HDFS data from R, Python, and Scala libraries. The following code samples demonstrate how to count the number of occurrences of each word in a simple text file in HDFS.

Navigate to your project and click **Open Workbench**. Create a file called `sample_text_file.txt` and save it to your project in the `data` folder. Now write this file to HDFS. You can do this in one of the following ways:

- Click **Terminal** above the Cloudera Data Science Workbench console and enter the following command to write the file to HDFS:

```
hdfs dfs -put data/sample_text_file.txt /tmp
```

OR

Importing Data into Cloudera Data Science Workbench

- Use the workbench command prompt:

Python Session

```
!hdfs dfs -put data/sample_text_file.txt /tmp
```

R Session

```
system("hdfs dfs -put data/tips.csv /user/hive/warehouse/tips/")
```

The following examples use Python and Scala to read `sample_text_file.txt` from HDFS (written above) and perform the count operation on it.

Python

```
from __future__ import print_function
import sys, re
from operator import add
from pyspark.sql import SparkSession

spark = SparkSession\
    .builder\
    .appName("PythonWordCount")\
    .getOrCreate()

# Access the file
lines = spark.read.text("/tmp/sample_text_file.txt").rdd.map(lambda r: r[0])
counts = lines.flatMap(lambda x: x.split(' ')) \
    .map(lambda x: (x, 1)) \
    .reduceByKey(add) \
    .sortBy(lambda x: x[1], False)
output = counts.collect()
for (word, count) in output:
    print("%s: %i" % (word, count))

spark.stop()
```

Scala

```
//count lower bound
val threshold = 2

// read the file added to hdfs
val tokenized = sc.textFile("/tmp/sample_text_file.txt").flatMap(_.split(" "))

// count the occurrence of each word
val wordCounts = tokenized.map(_._1).reduceByKey(_ + _)

// filter out words with fewer than threshold occurrences
val filtered = wordCounts.filter(_._2 >= threshold)

System.out.println(filtered.collect().mkString(", "))
```

Accessing Data from Apache HBase

This section demonstrates how to use the HappyBase Python library to access data from HBase.

Load Data into HBase Table

For this example, we're going to import data from a CSV file into HBase using the `importTsv` package.

1. Log into Cloudera Data Science Workbench and launch a Python 3 session within a new/existing project.
2. For this example, we will be using the following sample CSV file. Create the following `employees.csv` file in your project.

employees.csv

```
1, Lucy, Engineering
2, Milton, Engineering
3, Edith, Support
```

3. In the workbench, click **Terminal access**. Perform the following steps in the Terminal:

a. Start the HBase shell and create a new blank table called `employees`.

```
hbase shell
create 'employees', 'name', 'department'
exit
```

b. Load `employees.csv` into HDFS.

```
hdfs dfs -put employees.csv /tmp
```

c. Use [ImportTsv](#) to load data from HDFS (`/tmp/employees.csv`) into the HBase table created in the previous step.

```
hbase org.apache.hadoop.hbase.mapreduce.ImportTsv -Dimporttsv.separator=', '
-Dimporttsv.columns=HBASE_ROW_KEY,name,department employees /tmp/employees.csv
```

d. Go back to the HBase shell and run the following command to make sure data was loaded into the HBase table.

```
hbase shell
scan 'employees'
```

Query Data Using HappyBase

1. Launch a Python 3 session and use the workbench command prompt to install the `happybase` package.

```
!pip3 install happybase
```

2. Use `happybase` to connect to the `employees` table created in the previous step.

Python

```
import happybase
connection = happybase.Connection(host='<hbase_thrift_server_hostname>', port=9090,
autoconnect=True)
table = connection.table('employees')
rows = table.rows(['1', '2', '3'])
for key, data in rows:
    print(key, data)
```

Accessing Data from Apache Hive

The following code sample demonstrates how to establish a connection with the Hive metastore and access data from tables in Hive.

Python

```
import os
!pip3 install impyla
!pip3 install thrift_sasl
```

Importing Data into Cloudera Data Science Workbench

```
import os
import pandas
from impala.dbapi import connect
from impala.util import as_pandas

# Specify HIVE_HS2_HOST host name as an environment variable in your project settings
HIVE_HS2_HOST='<hiveserver2_hostname>'

# This connection string depends on your cluster setup and authentication mechanism
conn = connect(host=HIVE_HS2_HOST,
               port=10000,
               auth_mechanism='GSSAPI',
               kerberos_service_name='hive')
cursor = conn.cursor()
cursor.execute('SHOW TABLES')
tables = as_pandas(cursor)
tables
```

Accessing Data from Apache Impala

In this section, we take some sample data in the form of a CSV file, save the contents of this file to a table in Impala, and then use some common Python and R libraries to run simple queries on this data.

Loading CSV Data into an Impala Table

For this demonstration, we will be using the [tips.csv](#) dataset. Use the following steps to save this file to a project in Cloudera Data Science Workbench, and then load it into a table in Apache Impala.

1. [Create a new Cloudera Data Science Workbench project.](#)
2. Create a folder called `data` and upload `tips.csv` to this folder. For detailed instructions, see [Managing Project Files](#).
3. The next steps require access to services on the CDH cluster. If Kerberos has been enabled on the cluster, enter your credentials (username, password/keytab) in Cloudera Data Science Workbench to enable access. For instructions, see [Hadoop Authentication with Kerberos for Cloudera Data Science Workbench](#) on page 291.
4. Navigate back to the project Overview page and click **Open Workbench**.
5. Launch a new session (Python or R).
6. Open the **Terminal**.
 - a. Run the following command to create an empty table in Impala called `tips`. Replace `<impala_daemon_hostname>` with the hostname for your Impala daemon.

```
impala-shell -i <impala_daemon_hostname>:21000 -q '
CREATE TABLE default.tips (
  `total_bill` FLOAT,
  `tip` FLOAT,
  `sex` STRING,
  `smoker` STRING,
  `day` STRING,
  `time` STRING,
  `size` TINYINT)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ","
LOCATION "hdfs:///user/hive/warehouse/tips/";'
```

- b. Run the following command to load data from the `/data/tips.csv` file into the Impala table.

```
hdfs dfs -put data/tips.csv /user/hive/warehouse/tips/
```

Running Queries on Impala Tables

This section demonstrates how to run queries on the `tips` table created in the previous section using some common Python and R libraries such as Pandas, Impyla, Sparklyr and so on. All the examples in this section run the same query, but use different libraries to do so.

PySpark (Python)

```

from pyspark.sql import SparkSession

spark = SparkSession.builder.master('yarn').getOrCreate()

# load data from .csv file in HDFS
# tips = spark.read.csv("/user/hive/warehouse/tips/", header=True, inferSchema=True)

# OR load data from table in Hive metastore
tips = spark.table('tips')

from pyspark.sql.functions import col, lit, mean

# query using DataFrame API
tips \
    .filter(col('sex').like("%Female%")) \
    .groupBy('day') \
    .agg(mean('tip').alias('avg_tip')) \
    .orderBy('avg_tip',ascending=False) \
    .show()

# query using SQL
spark.sql('''
SELECT day,AVG(tip) AS avg_tip \
FROM tips \
WHERE sex LIKE "%Female%" \
GROUP BY day \
ORDER BY avg_tip DESC''').show()

spark.stop()

```

Impyla (Python)

Python 2

```

# (Required) Install the impyla package
# !pip install impyla
# !pip install thrift_sasl
import os
import pandas
from impala.dbapi import connect
from impala.util import as_pandas

# Connect to Impala using Impyla
# Secure clusters will require additional parameters to connect to Impala.
# Recommended: Specify IMPALA_HOST as an environment variable in your project settings

IMPALA_HOST = os.getenv('IMPALA_HOST', '<impala_daemon_hostname>')
conn = connect(host=IMPALA_HOST, port=21050)

# Execute using SQL
cursor = conn.cursor()

cursor.execute('SELECT day,AVG(tip) AS avg_tip \
               FROM tips \
               WHERE sex ILIKE "%Female%" \
               GROUP BY day \
               ORDER BY avg_tip DESC')

# Pretty output using Pandas
tables = as_pandas(cursor)
tables

```

Python 3

Importing Data into Cloudera Data Science Workbench

The specific library versions shown in this example are needed for Impyla to work correctly with Python 3.

```
# Install Libraries
!pip3 install impyla==0.13.8
!pip3 install thrift_sasl==0.2.1
!pip3 install thrift==0.9.3
!pip3 install sasl==0.2.1

# Connect to Impala
from impala.dbapi import connect

conn = connect(host='host',
               port=21050,
               auth_mechanism='GSSAPI',
               use_ssl=True,
               kerberos_service_name='impala')

# Execute Query
sql = "select * from table"

cursor = conn.cursor()
cursor.execute(sql)
results = cursor.fetchall()
```

Ibis (Python)

```
# (Required) Install the ibis-framework[impala] package
# !pip3 install ibis-framework[impala]

import ibis
import os
ibis.options.interactive = True
ibis.options.verbose = True

# Connection to Impala
# Secure clusters will require additional parameters to connect to Impala.
# Recommended: Specify IMPALA_HOST as an environment variable in your project settings

IMPALA_HOST = os.getenv('IMPALA_HOST', '<impala_daemon_hostname>')
con = ibis.impala.connect(host=IMPALA_HOST, port=21050, database='default')
con.list_tables()

tips = con.table('tips')

tips \
    .filter(tips.sex.like(['%Female%'])) \
    .group_by('day') \
    .aggregate( \
        avg_tip=tips.tip.mean() \
    ) \
    .sort_by(ibis.desc('avg_tip')) \
    .execute()
```

Sparklyr (R)

```
# (Required) Install the sparklyr package
# install.packages("sparklyr")

library(stringr)
library(sparklyr)
library(dplyr)

spark <- spark_connect(master = "yarn")

# load data from file in HDFS
tips <- spark_read_csv(
  sc = spark,
  name = "tips",
  path = "/user/hive/warehouse/tips/")
```

```

)

# OR load data from table
tips <- tbl(spark, "tips")

# query using dplyr
tips %>%
  filter(sex %like% "%Female%") %>%
  group_by(day) %>%
  summarise(
    avg_tip = mean(tip, na.rm = TRUE)
  ) %>%
  arrange(desc(avg_tip))

# query using SQL
tbl(spark, sql("
  SELECT day,AVG(tip) AS avg_tip \
  FROM tips \
  WHERE sex LIKE '%Female%' \
  GROUP BY day \
  ORDER BY avg_tip DESC"))

spark_disconnect(spark)

```

Accessing Data in Amazon S3 Buckets

Every language in Cloudera Data Science Workbench has libraries available for uploading to and downloading from Amazon S3.

To work with S3:

1. Add your Amazon Web Services [access keys](#) to your project's [environment variables](#) as `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`.
2. Pick your favorite language from the code samples below. Each one downloads the R 'Old Faithful' dataset from S3.

R

```

library("devtools")
install_github("armstrtw/AWS.tools")

Sys.setenv("AWSACCESSKEY"=Sys.getenv("AWS_ACCESS_KEY_ID"))
Sys.setenv("AWSSECRETKEY"=Sys.getenv("AWS_SECRET_ACCESS_KEY"))

library("AWS.tools")

s3.get("s3://sense-files/faithful.csv")

```

Python

```

# Install Boto to the project
!pip install boto

# Create the Boto S3 connection object.
from boto.s3.connection import S3Connection
aws_connection = S3Connection()

# Download the dataset to file 'faithful.csv'.
bucket = aws_connection.get_bucket('sense-files')
key = bucket.get_key('faithful.csv')
key.get_contents_to_filename('/home/cdsw/faithful.csv')

```

Accessing External SQL Databases

Every language in Cloudera Data Science Workbench has multiple client libraries available for SQL databases.

If your database is behind a firewall or on a secure server, you can connect to it by creating an [SSH tunnel](#) to the server, then connecting to the database on `localhost`.

If the database is password-protected, consider storing the password in an environmental variable to avoid displaying it in your code or in consoles. The examples below show how to retrieve the password from an [environment variable](#) and use it to connect.

R

```
# dplyr lets you program the same way with local data frames and remote SQL databases.
install.packages("dplyr")
library("dplyr")
db <- src_postgres(dbname="test_db", host="localhost", port=5432, user="cdswuser",
password=Sys.getenv("POSTGRES_PASSWORD"))
flights_table <- tbl(db, "flights")
select(flights_table, year:day, dep_delay, arr_delay)
```

Python

You can access data using [pyodbc](#) or [SQLAlchemy](#)

```
# pyodbc lets you make direct SQL queries.
!wget https://pyodbc.googlecode.com/files/pyodbc-3.0.7.zip
!unzip pyodbc-3.0.7.zip
!cd pyodbc-3.0.7;python setup.py install --prefix /home/cdsw
import os

# See http://www.connectionstrings.com/ for information on how to construct ODBC
connection strings.
db = pyodbc.connect("DRIVER={PostgreSQL
Unicode};SERVER=localhost;PORT=5432;DATABASE=test_db;USER=cdswuser;OPTION=3;PASSWORD=%s"
% os.getenv("POSTGRES_PASSWORD"))
cursor = cnxn.cursor()
cursor.execute("select user_id, user_name from users")

# sqlalchemy is an object relational database client that lets you make database queries
in a more Pythonic way.
!pip install sqlalchemy
import os

import sqlalchemy
from sqlalchemy.orm import sessionmaker
from sqlalchemy import create_engine
db = create_engine("postgresql://cdswuser:%s@localhost:5432/test_db" %
os.getenv("POSTGRES_PASSWORD"))
session = sessionmaker(bind=db)
user = session.query(User).filter_by(name='ed').first()
```

Experiments

Starting with version 1.4, Cloudera Data Science Workbench allows data scientists to run batch experiments that track different versions of code, input parameters, and output (both metrics and files).

Demo: Watch the following video for a quick demonstration of the steps described in this topic: [Experiments with Cloudera Data Science Workbench](#)

Related:

- [Engines for Experiments and Models](#) on page 206
- [Debugging Issues with Experiments](#) on page 178

Purpose

Challenge

As data scientists iteratively develop models, they often experiment with datasets, features, libraries, algorithms, and parameters. Even small changes can significantly impact the resulting model. This means data scientists need the ability to iterate and repeat similar experiments in parallel and on demand, as they rely on differences in output and scores to tune parameters until they obtain the best fit for the problem at hand. Such a training workflow requires versioning of the file system, input parameters, and output of each training run.

Without versioned experiments you would need intense process rigor to consistently track training artifacts (data, parameters, code, etc.), and even then it might be impossible to reproduce and explain a given result. This can lead to wasted time/effort during collaboration, not to mention the compliance risks introduced.

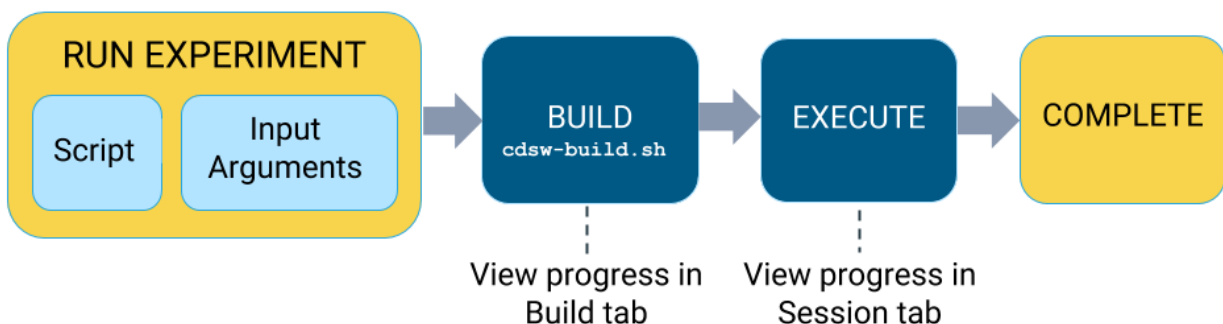
Solution

Starting with version 1.4, Cloudera Data Science Workbench uses *experiments* to facilitate ad-hoc batch execution and model training. Experiments are batch executed workloads where the code, input parameters, and output artifacts are versioned. This feature also provides a lightweight ability to track output data, including files, metrics, and metadata for comparison.

Concepts

The term *experiment* refers to a non interactive batch execution script that is versioned across input parameters, project files, and output. Batch experiments are associated with a specific project (much like sessions or jobs) and have no notion of scheduling; they run at creation time. To support versioning of the project files and retain run-level artifacts and metadata, each experiment is executed in an isolated container.

Lifecycle of an Experiment



The rest of this section describes the different stages in the lifecycle of an experiment - from launch to completion.

1. Launch Experiment

In this step you will select a script from your project that will be run as part of the experiment, and the resources (memory/GPU) needed to run the experiment. The engine kernel will be selected by default based on your script. For detailed instructions on how to launch an experiment, see [Running an Experiment \(QuickStart\)](#) on page 174.

2. Build

When you launch the experiment, Cloudera Data Science Workbench first builds a new versioned engine image where the experiment will be executed in isolation. This new engine includes:

- the base engine image used by the project (check **Project > Settings**)
- a snapshot of the project filesystem
- environmental variables inherited from the project.
- packages explicitly specified in the project's build script (`cdsw-build.sh`)

It is your responsibility to provide the complete list of dependencies required for the experiment via the `cdsw-build.sh` file. As part of the engine's build process, Cloudera Data Science Workbench will run the `cdsw-build.sh` script and install the packages or libraries requested there on the new image.

Note that custom mounts or environment variables configured in `cdsw.conf` (such as `NO_PROXY`, `HTTP(S)_PROXY`, etc.) are still not passed to the container builds for experiments and models (even though they are applied to sessions, jobs, and deployed models/experiments).

For details about the build process and examples on how to specify dependencies, see [Engines for Experiments and Models](#) on page 206.

3. Schedule

Once the engine is built the experiment is scheduled for execution like any other job or session. Once the requested CPU/GPU and memory have been allocated to the experiment, it will move on to the execution stage.

Note that if your deployment is running low on memory and CPU, your runs may spend some time in this stage.

4. Execute

This is the stage where the script you have selected will be run in the newly built engine environment. This is the same output you would see if you had executed the script in a session in the Workbench console.

You can watch the execution in progress in the individual run's **Session** tab.

You can also go to the project **Overview > Experiments** page to see a table of all the experiments launched within that project and their current status.

Run ID: A numeric ID that tracks *all* experiments launched on a Cloudera Data Science Workbench deployment. It is not limited to the scope of a single user or project.

Running an Experiment (QuickStart)

The following steps describe how to launch an experiment from the Workbench console. In this example we are going to run a simple script that adds all the numbers passed as arguments to the experiment.

1. Go to the project **Overview** page.
2. Click **Open Workbench**.
3. Create/modify any project code as needed. You can also launch a session to simultaneously test code changes on the interactive console as you launch new experiments.

As an example, you can run this Python script that accepts a series of numbers as command-line arguments and prints their sum.

add.py

```
import sys
import cdsw

args = len(sys.argv) - 1
sum = 0
x = 1

while (args >= x):
    print ("Argument %i: %s" % (x, sys.argv[x]))
    sum = sum + int(sys.argv[x])
    x = x + 1

print ("Sum of the numbers is: %i." % sum)
```

To test the script, launch a Python session and run the following command from the workbench command prompt:

```
!python add.py 1 2 3 4
```

4. Click **Run Experiment**. If you're already in an active session, click **Run > Run Experiment**. Fill out the following fields:

- **Script** - Select the file that will be executed for this experiment.
- **Arguments** - If your script requires any command line arguments, enter them here.



Note: Arguments are not supported with Scala experiments.

- **Engine Kernel and Engine Profile** - Select the kernel and computing resources needed for this experiment.

For this example we will run the `add.py` script and pass some numbers as arguments.

5. Click **Start Run**.

6. To track progress for the run, go back to the project **Overview**. On the left navigation bar click **Experiments**. You should see the experiment you've just run at the top of the list. Click on the Run ID to view an overview for each individual run. Then click **Build**.

On this **Build** tab you can see realtime progress as Cludera Data Science Workbench builds the Docker image for this experiment. This allows you to debug any errors that might occur during the build stage.

Run-4 [New Experiment](#)

Overview **Session** Build

```

Sending build context to Docker daemon 14.34 MB

Step 1/5 : FROM docker.repository.cloudera.com/cdsw/engine:5
---> da62f78351e0
Step 2/5 : COPY sources /home/cdsw
---> dc0d62362524
Removing intermediate container 52edb8ecef1f
Step 3/5 : WORKDIR /home/cdsw
---> 604125f1bd47
Removing intermediate container 934e47683660
Step 4/5 : RUN chown -R 8536:8536 /home/cdsw
---> Running in 80097e84eb11

```

7. Once the Docker image is ready, the run will begin execution. You can track progress for this stage by going to the **Session** tab.

For example, the **Session** pane output from running `add.py` is:

Run-4 [New Experiment](#)

Overview **Session** Build

```

> import sys
> import cdsw
> args = len(sys.argv) - 1
> sum = 0
> x = 1
> while (args >= x):
    print ("Parameter %i: %s" % (x, sys.argv[x]))
    sum = sum + int(sys.argv[x])
    x = x + 1

Parameter 1: 18
Parameter 2: 90
Parameter 3: 34

> print ("Sum of the numbers is: %i." % sum)

Sum of the numbers is: 142.

```

8. (Optional) The `cdsw` library that is bundled with Cloudera Data Science Workbench includes some built-in functions that you can use to compare experiments and save any files from your experiments.

For example, to track the sum for each run, add the following line to the end of the `add.py` script.

```
cdsw.track_metric("Sum", sum)
```

This will be tracked in the **Experiments** table:

Run	Script	Arguments	Kernel	Comment	Submitter	Created At	Sum	Status	Duration
12	add.py	1 2 3 4 5	python3		admin	6/11/18 1:48 PM		Scheduling	
11	add.py	66 42 97 104 2004	python3		admin	6/11/18 1:44 PM	2313	Success	0 mins
10	add.py	4 60 72	python3		admin	6/11/18 1:43 PM	136	Success	0 mins
9	add.py	22 96 78	python3		admin	6/11/18 1:43 PM	196	Success	0 mins

For more details, see [Tracking Metrics](#) on page 177 and [Saving Files](#) on page 177.

Tracking Metrics

The `cdsw` library includes a `track_metric` function that can be used to log up to 50 metrics associated with a run, thus allowing accuracy and scores to be tracked over time.

The function accepts input in the form of key value pairs.

```
cdsw.track_metric(key, value)
```

Python

```
cdsw.track_metric("R_squared", 0.79)
```

R

```
cdsw::track.metric("R_squared", 0.62)
```

These metrics will be available on the project's **Experiments** tab where you can view, sort, and filter experiments on the values. The table on the **Experiments** page will allow you to display only three metrics at a time. You can select which metrics are displayed from the **metrics** dropdown.



Note: This function is not supported with Scala experiments.

Saving Files

Cloudera Data Science Workbench allows you to select which artifacts you'd like to access and evaluate after an experiment is complete. These artifacts could be anything from a text file to an image or a model that you have built through the run.

The `cdsw` library includes a `track_file` function that can be used to specify which artifacts should be retained after the experiment is complete.

Python

```
cdsw.track_file('model.pkl')
```

R

```
cdsw::track.file('model.pkl')
```

Specified artifacts can be accessed from the run's Overview page. These files can also be saved to the top-level project filesystem and downloaded from there.



Note: This function is not supported with Scala experiments.

Disabling the Experiments Feature

Required Role: [Site Administrator](#)



Important: The feature flag mentioned here only hides the Experiments feature from the UI. It will not stop any experiments that have already been queued for execution.

To disable this feature on your Cloudera Data Science Workbench deployment:

1. Log in to Cloudera Data Science Workbench.
2. Click **Admin > Settings**.
3. Under the **Feature Flags** section, disable the **Enable users to run experiments** checkbox.

Limitations

- (If [quotas](#) are enabled) Experiments that are stuck in the **Scheduled** state due to lack of resources do not automatically start even if you free up existing resources.

Workaround: Stop the experiment that is stuck in the **Scheduled** state. Then manually reschedule the experiment.

Cloudera Bug: DSE-8736

- Experiments do not store snapshots of project files. You cannot automatically restore code that was run as part of an experiment.
- Experiments will fail if your project filesystem is too large for the Git [snapshot](#) process. As a general rule, any project files (code, generated model artifacts, dependencies, etc.) larger than 50 MB must be part of your project's `.gitignore` file so that they are not included in snapshots for experiment builds.
- Experiments cannot be deleted. As a result, be conscious of how you use the `track_metrics` and `track_file` functions.
 - Do not track files larger than 50MB.
 - Do not track more than 100 metrics per experiment. Excessive metric calls from an experiment may cause Cloudera Data Science Workbench to hang.
- The Experiments table will allow you to display only three metrics at a time. You can select which metrics are displayed from the **metrics** dropdown. If you are tracking a large number of metrics (100 or more), you might notice some performance lag in the UI.
- Arguments are not supported with Scala experiments.
- The `track_metrics` and `track_file` functions are not supported with Scala experiments.
- The UI does not display a confirmation when you start an experiment or any alerts when experiments fail.

Debugging Issues with Experiments

This topic lists some common issues to watch out for during an experiment's [build and execution process](#):

Experiment spends too long in Scheduling/Built stage

If your experiments are spending too long in any particular stage, check the resource consumption statistics for the cluster. When the cluster starts to run out of resources, often experiments (and other entities like jobs, models) will spend too long in the queue before they can be executed.

Resource consumption by experiments (and jobs, sessions) can be tracked by site administrators on the **Admin > Activity** page.

Experiment fails in the Build stage

During the build stage Cloudera Data Science Workbench creates a new Docker image for the experiment. You can track progress for this stage on each experiment's **Build** page. The build logs on this page should help point you in the right direction.

Common issues that might cause failures at this stage include:

- Lack of execute permissions on the build script itself.
- Inability to reach the Python package index or R mirror when installing packages.
- Typo in the name of the build script (`cdsw-build.sh`). Note that the build process will only execute a script called `cdsw-build.sh`; not any other bash scripts from your project.
- Using `pip3` to install packages in `cdsw-build.sh`, but selecting a Python 2 kernel when you actually launch the experiment. Or vice versa.

Experiment fails in the Execute stage

Each experiment includes a **Session** page where you can track the output of the experiment as it executes. This is similar to the output you would see if you test the experiment in the workbench console. Any runtime errors will display on the **Session** page just as they would in an interactive session.

Models

Starting with version 1.4, Cloudera Data Science Workbench allows data scientists to build, deploy, and manage models as REST APIs to serve predictions.

Demo: Watch the following video for a quick demonstration of the steps described in this topic: [Model Deployment with Cloudera Data Science Workbench](#)

Related:

- [Engines for Experiments and Models](#) on page 206
- [Model Training and Deployment Example - Iris Dataset](#)
- [Model Monitoring and Administration](#) on page 193
- [Debugging Issues with Models](#) on page 195

Purpose

Challenge

Data scientists often develop models using a variety of Python/R open source packages. The challenge lies in actually exposing those models to stakeholders who can test the model. In most organizations, the model deployment process will require assistance from a separate DevOps team who likely have their own policies about deploying new code.

For example, a model that has been developed in Python by data scientists might be rebuilt in another language by the devops team before it is actually deployed. This process can be slow and error-prone. It can take months to deploy new models, if at all. This also introduces compliance risks when you take into account the fact that the new re-developed model might not be even be an accurate reproduction of the original model.

Once a model has been deployed, you then need to ensure that the devops team has a way to rollback the model to a previous version if needed. This means the data science team also needs a reliable way to retain history of the models they build and ensure that they can rebuild a specific version if needed. At any time, data scientists (or any other stakeholders) must have a way to accurately identify which version of a model is/was deployed.

Solution

Starting with version 1.4, Cloudera Data Science Workbench allows data scientists to build and deploy their own models as REST APIs. Data scientists can now select a Python or R function within a project file, and Cloudera Data Science Workbench will:

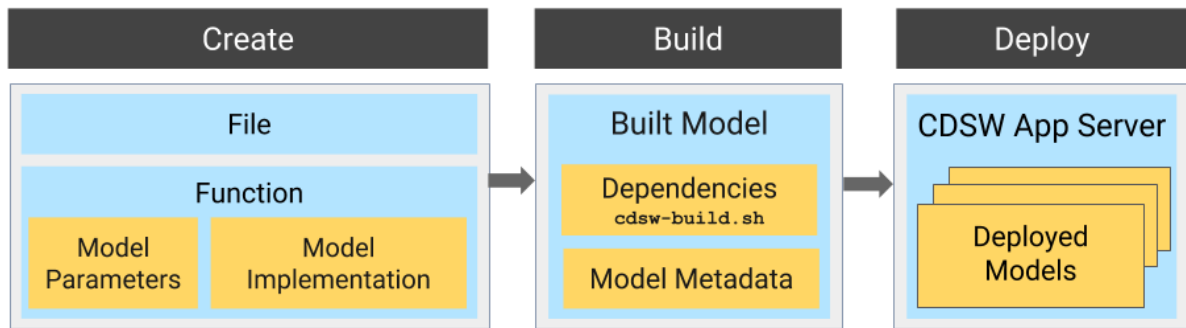
- Create a snapshot of model code, model parameters, and dependencies.
- Package a trained model into an immutable artifact and provide basic serving code.
- Add a REST endpoint that automatically accepts input parameters matching the function, and that returns a data structure that matches the function's return type.
- Save the model along with some metadata.
- Deploy a specified number of model API replicas, automatically load balanced.

Concepts and Terminology

Model

Model is a high level abstract term that is used to describe several possible incarnations of objects created during the model deployment process. For the purpose of this discussion you should note that 'model' does not always refer to a specific artifact. More precise terms (as defined later in this section) should be used whenever possible.

Stages of the Model Deployment Process



The rest of this section contains supplemental information that describes the model deployment process in detail.

Create

- **File** - The R or Python file containing the function to be invoked when the model is started.
- **Function** - The function to be invoked inside the file. This function should take a single JSON-encoded object (for example, a python dictionary) as input and return a JSON-encodable object as output to ensure compatibility with any application accessing the model using the API. JSON decoding and encoding for model input/output is built into Cloudera Data Science Workbench.

The function will likely include the following components:

- **Model Implementation**

The code for implementing the model (e.g. decision trees, k-means). This might originate with the data scientist or might be provided by the engineering team. This code implements the model's predict function, along with any setup and teardown that may be required.

- **Model Parameters**

A set of parameters obtained as a result of model training/fitting (using [experiments](#)). For example, a specific decision tree or the specific centroids of a k-means clustering, to be used to make a prediction.

Build

This stage takes as input the file that calls the function and returns an artifact that implements a single concrete model, referred to as a model *build*.

- **Built Model**

A built model is a static, immutable artifact that includes the model implementation, its parameters, any runtime dependencies, and its metadata. If any of these components need to be changed, for example, code changes to the implementation or its parameters need to be retrained, a new build must be created for the model. Model builds are versioned using *build numbers*.

To create the model build, Cloudera Data Science Workbench creates a Docker image based on the engine designated as the project's default engine. This image provides an isolated environment where the model implementation code will run.

To configure the image environment, you can specify a list of dependencies to be installed in a build script called `cdsw-build.sh`.

For details about the build process and examples on how to install dependencies, see [Engines for Experiments and Models](#) on page 206.

- **Build Number:**

Build numbers are used to track different versions of builds within the scope of a single model. They start at 1 and are incremented with each new build created for the model.

Deploy

This stage takes as input the memory/CPU resources required to power the model, the number of replicas needed, and deploys the model build created in the previous stage to a REST API.

- **Deployed Model**

A deployed model is a model build in execution. A built model is deployed in a model serving environment, likely with multiple replicas.

- **Environmental Variable**

You can set environmental variables each time you deploy a model. Note that models also inherit any environment variables set at the [project and global level](#). However, in case of any conflicts, variables set per-model will take precedence.



Note:

- If you are using any model-specific environmental variables, these must be specified every time you re-deploy a model. Models do not inherit environmental variables from previous deployments.
- Note that custom mounts or environment variables configured in `cdsw.conf` (such as `NO_PROXY`, `HTTP(S)_PROXY`, etc.) are still not passed to the container builds for experiments and models (even though they are applied to sessions, jobs, and deployed models/experiments).

- **Model Replicas**

The engines that serve incoming requests to the model. Note that each replica can only process one request at a time. Multiple replicas are essential for load-balancing, fault tolerance, and serving concurrent requests. Cloudera Data science Workbench allows you to deploy a maximum of 9 replicas per model.

- **Deployment ID**

Deployment IDs are numeric IDs used to track models deployed across Cloudera Data Science Workbench. They are not bound to a model or project.

Creating and Deploying a Model (QuickStart)

Using Cloudera Data Science Workbench, you can create any function within a script and deploy it to a REST API. In a machine learning project, this will typically be a predict function that will accept an input and return a prediction based on the model's parameters.

For the purpose of this quick start demo we are going to create a very simple function that adds two numbers and deploy it as a model that returns the sum of the numbers. This function will accept two numbers in JSON format as input and return the sum.

1. Create a new project. Note that models are always created within the context of a project.
2. Click **Open Workbench** and launch a new Python 3 session.
3. Create a new file within the project called `add_numbers.py`. This is the file where we define the function that will be called when the model is run. For example:

add_numbers.py

```
def add(args):
    result = args["a"] + args["b"]
    return result
```



Note: In practice, do not assume that users calling the model will provide input in the correct format or enter good values. Always perform input validation.

4. Before deploying the model, test it by running the `add_numbers.py` script, and then calling the `add` function directly from the interactive workbench session. For example:

```
add({"a": 3, "b": 5})
```

The screenshot shows a Jupyter Notebook interface. On the left, a code editor displays a Python function:

```
1 # Function to add two numbers
2
3 def add(args):
4     result = args["a"] + args["b"]
5     return result
6
```

On the right, the terminal output shows the function being called twice:

```
> add({"a": 3, "b": 5})
8
> add({"a": 4, "b": 7})
11
```

5. Deploy the `add` function to a REST endpoint.

- a. Go to the project **Overview** page.
- b. Click **Models > New Model**.
- c. Give the model a Name and Description.
- d. Enter details about the model that you want to build. In this case:
 - **File:** `add_numbers.py`
 - **Function:** `add`
 - **Example Input:** `{"a": 3, "b": 5}`
 - **Example Output:** `8`

The screenshot shows the 'New Model' configuration form. It has four main sections:

- File ***: `add_numbers.py`
- Function ***: `add`
- Example Input ?**: `{ "a": 3, "b": 5 }`
- Example Output ?**: `8`

- e. Select the resources needed to run this model, including any replicas for load balancing.
 - f. Click **Deploy Model**.
6. Click on the model to go to its **Overview** page. Click **Builds** to track realtime progress as the model is built and deployed. This process essentially creates a Docker container where the model will live and serve requests.

The screenshot shows the 'Builds' tab for a model named 'Add Two Numbers'. At the top right, there are buttons for 'Building', 'Stop', and 'Deploy New Build'. Below the title, there are navigation tabs: 'Overview', 'Deployments', 'Builds' (selected), 'Monitoring', and 'Settings'. A table lists the build details:

Build	Status	File	Function	Kernel	Engine Image	Created By	Created At	Comment	Actions
1	Building	add_numbers.py	add	python3	Base Image v	ambreen	Jun 5, 2018, 5:44 PM	Initial revision.	Delete

Below the table, a terminal window shows the following build logs:

```

Sending build context to Docker daemon 15.05 MB

Step 1/16 : FROM docker.repository.cloudera.com/cdsw/engine:
---> f8955770daa1
Step 2/16 : ENTRYPOINT node /app/model-runtime/model-server.js
---> Running in 58038f1e58d5

```

- Once the model has been deployed, go back to the model **Overview** page and use the **Test Model** widget to make sure the model works as expected.

If you entered example input when creating the model, the Input field will be pre-populated with those values. Click **Test**. The result returned includes the output response from the model, as well as the ID of the replica that served the request.

Model response times depend largely on your model code. That is, how long it takes the model function to perform the computation needed to return a prediction. It is worth noting that model replicas can only process one request at a time. Concurrent requests will be queued until the model can process them.

Calling a Model

This section lists some requirements for model requests and how to test a model using Cloudera Data Science Workbench.

- [\(Requirement\) JSON for Model Requests/Responses](#)
- [\(Requirement\) Model Access Key](#)
- [Test Calls to a Model](#)

(Requirement) JSON for Model Requests/Responses

Every model function in Cloudera Data Science Workbench takes a single argument in the form of a JSON-encoded object, and returns another JSON-encoded object as output. This format ensures compatibility with any application accessing the model using the API, and gives you the flexibility to define how JSON data types map to your model's datatypes.

Model Requests

When making calls to a model, keep in mind that JSON is not suitable for very large requests and has high overhead for binary objects such as images or video. Consider calling the model with a reference to the image or video such as a URL instead of the object itself. Requests to models should not be more than 5 MB in size. Performance may degrade and memory usage increase for larger requests.



Note: In Cloudera Data Science Workbench 1.4.0, model request sizes were limited to 100 KB. With version 1.4.2 (and higher), this limit has been increased to 5 MB. To take advantage of this higher threshold, you will need to upgrade to version 1.4.2 (or higher) and rebuild your existing models.

Ensure that the JSON request represents all objects in the request or response of a model call. For example, JSON does not natively support dates. In such cases consider passing dates as strings, for example in ISO-8601 format, instead.

For a simple example of how to pass JSON arguments to the model function and make calls to deployed model, see [Creating and Deploying a Model \(QuickStart\)](#) on page 182.

Model Responses

Models return responses in the form of a JSON-encoded object. Model response times depend on how long it takes the model function to perform the computation needed to return a prediction. Model replicas can only process one request at a time. Concurrent requests are queued until a replica is available to process them.

When Cloudera Data Science Workbench receives a call request for a model, it attempts to find a free replica that can answer the call. If the first arbitrarily selected replica is busy, Cloudera Data Science Workbench will keep trying to contact a free replica for 30 seconds. If no replica is available, Cloudera Data Science Workbench will return a `model.busy` error with HTTP status code 429 (Too Many Requests). If you see such errors, [re-deploy the model build](#) with a higher number of replicas.

(Requirement) Access Key

Each model in Cloudera Data Science Workbench has a unique access key associated with it. This access key serves two purposes: 1) it is a unique identifier for the model, and 2) it serves as an authentication token that allows you to make calls to the model.

Models deployed using Cloudera Data Science Workbench are not public. In order to call an active model your request must include the model's access key for authentication (as demonstrated in the sample calls above).

To locate the access key for a model, go to the model **Overview** page and click **Settings**.

The screenshot shows the 'Add Two Numbers' model page in Cloudera Data Science Workbench. The 'Settings' tab is selected and circled in red. Below the tabs, there are fields for 'Name' (Add Two Numbers) and 'Description' (This model takes two numbers as input and returns their sum.). At the bottom, the 'Access Key' is displayed as 'mqw7e9nwjw8ym1sz0sfqgzqh9sshj26'. A 'Regenerate' button is next to it. An arrow points from the text 'Access Key required to make requests to this model.' to the access key field.



Important:

Only one access key per model is active at any time. If you regenerate the access key, you will need to re-distribute this access key to users/applications using the model.

Alternatively, you can use this mechanism to revoke access to a model by regenerating the access key. Anyone with an older version of the key will not be able to make calls to the model.

Testing Calls to a Model

Cloudera Data Science Workbench provides two ways to test calls to a model:

- **Test Model Widget**

On each model's **Overview** page, Cloudera Data Science Workbench provides a widget that makes a sample call to the deployed model to ensure it is receiving input and returning results as expected.

Test Model

Input

```
{
  "a": 3,
  "b": 5
}
```

Test Reset

Result

Status	● success
Response	8
Replica ID	add-two-numbers-1-1-86b9b58b7b-g6s8r

- **Sample Request Strings**

On the model **Overview** page, Cloudera Data Science Workbench also provides sample curl and POST request strings that you can use to test calls to the model. Copy/paste the `curl` request directly into a Terminal to test the call.

Note that these sample requests already include the example input values you entered while building the model, and the [access key](#) required to query the model.

Overview Deployments Builds Monitoring Settings

Description predict

Sample Code

Shell
Python
R

```
curl -H "Content-Type: application/json" -X POST http://modelservice.s[REDACTED]
[REDACTED].cloudera.com/model -d '{"accessKey": "motvod88m3jnpdvkci0r3934r
ij5z7at", "request": {"param": "value"}}'
```

Updating Active Models

Active Model - A model that is in the **Deploying**, **Deployed**, or **Stopping** stages.

You can make changes to a model even after it has been deployed and is actively serving requests. Depending on business factors and changing resource requirements, such changes will likely range from changes to the model code itself, to simply modifying the number of CPU/GPUs requested for the model. In addition, you can also stop and restart active models.

Depending on your requirement, you can perform one of the following actions:

Re-deploy an Existing Build

Re-deploying a model involves re-publishing a previously-deployed model in a new serving environment - this is, with an updated number of replicas or memory/CPU/GPU allocation. For example, circumstances that require a re-deployment might include:

- An active model that previously requested a large number of CPUs/GPUs that are not being used efficiently.
- An active model that is dropping requests because it is falling short of replicas.
- An active model needs to be rolled back to one of its previous versions.



Warning: Currently, Cloudera Data Science Workbench only allows one active deployment per model. This means when you re-deploy a build, the current active deployment will go offline until the re-deployment process is complete and the new deployment is ready to receive requests. Prepare for model downtime accordingly.

To re-deploy an existing model:

1. Go to the model **Overview** page.
2. Click **Deployments**.
3. Select the version you want to deploy and click **Re-deploy this Build**.



Note: If you are using any model-specific environmental variables, these must be specified every time you re-deploy a model. Models do not inherit environmental variables from previous deployments.

Deployed Stop Restart Deploy New Build

[Overview](#) [Deployments](#) [Builds](#) [Monitoring](#) [Settings](#)

Id	Build	Status	Deployed At	Stopped At	Deployed By
3	2	Deployed	Jun 7, 2018, 1:49 PM		ambreen
2	1	Stopped	Jun 7, 2018, 11:16 AM	Jun 7, 2018, 1:46 PM	ambreen
1	1	Stopped	Jun 5, 2018, 6:45 PM	Jun 7, 2018, 11:13 AM	ambreen

Model

Id: 1

Name: Add Two Numbers

Description: This model takes two numbers as input and returns their sum.

Re-deploy This Build

4. Modify the model serving environment as needed.
5. Click **Deploy Model**.

Deploy a New Build for a Model

Deploying a new build for a model involves both, re-building the Docker image for the model, *and* deploying this new build. Note that this is not required if you only need to update the resources allocated to the model. As an example, changes that require a new build might include:

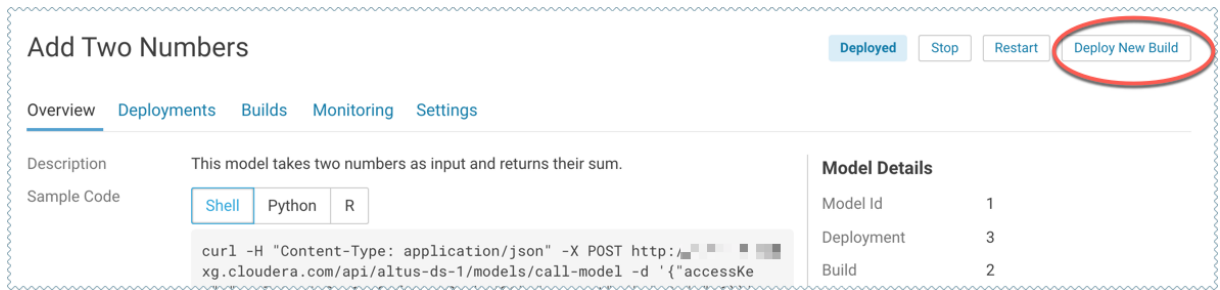
- Code changes to the model implementation.
- Renaming the function that is used to invoke the model.



Warning: Currently, Cloudera Data Science Workbench does not allow you to create a new build for a model without also deploying it. This combined with the fact that you can only have one active deployment per model means that once the new model is built, the current active deployment will go offline so that the new build can be deployed. Prepare for model downtime accordingly.

To create a new build and deploy it:

1. Go to the model **Overview** page.
2. Click **Deploy New Build**.



3. Complete the form and click **Deploy Model**.

Stop a Model

To stop a model (all replicas), go to the model **Overview** page and click **Stop**. Click **OK** to confirm.

Restart a Model

To restart a model (all replicas), go to the model **Overview** page and click **Restart**. Click **OK** to confirm.

Restarting a model does not let you make any code changes to the model. It should primarily be used as a way to quickly re-initialize or re-connect to resources.

Usage Guidelines

This section calls out some important guidelines you should keep in mind when you start deploying models with Cloudera Data Science Workbench.

Model Code

Models in Cloudera Data Science Workbench are designed to execute any code that is wrapped into a function. This means you can potentially deploy a model that returns the result of a `SELECT * query` on a very large table. However, Cloudera strongly recommends against using the models feature for such use cases.

As a best practice, your models should be returning simple JSON responses in near-real time speeds (within a fraction of a second). If you have a long-running operation that requires extensive computing and takes more than 15 seconds to complete, consider using batch jobs instead.

Model Artifacts

Once you start building larger models, make sure you are storing these model artifacts in HDFS, S3, or any other external storage. Do not use the project filesystem to store large output artifacts.

In general, any project files larger than 50 MB must be part of your project's `.gitignore` file so that they are not included in [snapshots](#) for future experiments/model builds. Note that in case your models require resources that are stored outside the model itself, it is up to you to ensure that these resources are available and immutable as model replicas may be restarted at any time.

Resource Consumption and Scaling

Models should be treated as any other long-running applications that are continuously consuming memory and computing resources. If you are unsure about your resource requirements when you first deploy the model, start with a single replica, monitor its usage, and scale as needed.

If you notice that your models are getting stuck in various stages of the deployment process, check the [monitoring](#) page to make sure that the cluster has sufficient resources to complete the deployment operation.

Security Considerations

As stated previously, models do not impose any limitations on the code they can execute. Additionally, models run with the permissions of the user that creates the model (same as sessions and jobs). Therefore, be conscious of potential data leaks especially when querying underlying data sets to serve predictions.

Cloudera Data Science Workbench models are not public by default. Each model has an [access key](#) associated with it. Only users/applications who have this key can make calls to the model. Be careful with who has permission to view this key.

Cloudera Data Science Workbench also prints `stderr/stdout logs` from models to an output pane in the UI. Make sure you are not writing any sensitive information to these logs.

Deployment Considerations

Cloudera Data Science Workbench does not currently support high availability for models. Additionally, there can only be one active deployment per model at any given time. This means you should plan for model downtime if you want to deploy a new build of the model or re-deploy with more/less replicas.

Keep in mind that models that have been developed and trained using Cloudera Data Science Workbench are essentially Python/R code that can easily be persisted and exported to external environments using popular serialization formats such as Pickle, PMML, ONNX, and so on.

Known Issues and Limitations

• Known Issues with Model Builds and Deployed Models

- Unable to create a model with the name of a deleted Model.

Workaround: For now, Models shall have unique names across the lifespan of the cluster installation.

Cloudera Bug: DSE-4237

- Re-deploying or re-building models results in model downtime (usually brief).
- Re-starting Cloudera Data Science Workbench does not automatically restart active models. These models must be manually restarted so they can serve requests again.

Cloudera Bug: DSE-4950

- Model deployment will fail if your project filesystem is too large for the Git [snapshot](#) process. As a general rule, any project files (code, generated model artifacts, dependencies, etc.) larger than 50 MB must be part of your project's `.gitignore` file so that they are not included in snapshots for model builds.
- Model builds will fail if your project filesystem includes a `.git` directory (likely hidden or nested). Typical build stage errors include:

```
Error: 2 UNKNOWN: Unable to schedule build: [Unable to create a checkpoint of current source: [Unable to push sources to git server: ...
```

To work around this, rename the `.git` directory (for example, `NO.git`) and re-build the model.

Cloudera Bug: DSE-4657

- JSON requests made to active models should not be more than 5 MB in size. This is because JSON is not suitable for very large requests and has high overhead for binary objects such as images or video. Call the model with a reference to the image or video, such as a URL, instead of the object itself.
- Any external connections, for example, a database connection or a Spark context, must be managed by the model's code. Models that require such connections are responsible for their own setup, teardown, and refresh.
- Model logs and statistics are only preserved so long as the individual replica is active. Cloudera Data Science Workbench may restart a replica at any time it is deemed necessary (such as bad input to the model).

• Limitations

- Scala models are not supported.

- [Spawning worker threads](#) is not supported with models.
- Models deployed using Cloudera Data Science Workbench are not highly-available.
- Dynamic scaling and auto-scaling are not currently supported. To change the number of replicas in service, you will have to re-deploy the build.

Model Training and Deployment - Iris Dataset

This topic uses Cloudera Data Science Workbench's built-in Python template project to walk you through an end-to-end example where we use experiments to develop and train a model, and then deploy it using Cloudera Data Science Workbench.

This example uses the canonical [Iris](#) dataset from [Fisher and Anderson](#) to build a model that predicts the width of a flower's petal based on the petal's length.

Create a Project

The scripts for this example are available in the Python template project that ships with Cloudera Data Science Workbench. First, create a new project from the Python template:

Once you've created the project, go to the project's **Files** page. The following files are used for the demo:

- `cdsw-build.sh` - A custom [build script](#) used for models and experiments. Pip installs our dependencies, primarily the `scikit-learn` library.
- `fit.py` - A model training example to be run as an experiment. Generates the `model.pkl` file that contains the fitted parameters of our model.
- `predict.py` - A sample function to be deployed as a model. Uses `model.pkl` produced by `fit.py` to make predictions about petal width.

Train the Model

Run experiments using `fit.py` to develop a model. The `fit.py` script tracks metrics, mean squared error (MSE) and R^2 , to help compare the results of different experiments. It also writes the fitted model to a `model.pkl` file.

To run an experiment:

1. Navigate to the Iris project's **Overview > Experiments** page.
2. Click **Run Experiment**.
3. Fill out the form as follows and click **Start Run**. Make sure you use the Python 3 kernel.

Run New Experiment

Script: fit.py

Arguments: Enter Arguments

Engine Kernel:

- Python 2
- Python 3
- Scala
- R

Engine Profile: 1 vCPU / 2 GiB Memory

Comment: Testing a run ..

Buttons: Cancel, Start Run

4. The new experiment should now show up on the **Experiments** table. Click on the Run ID to go to the experiment's **Overview** page. The **Build** and **Session** tabs display realtime progress as the experiment builds and executes.
5. Once the experiment has completed successfully, go back to its **Overview** page. The tracked metrics show us that our test set had an MSE of ~ 0.0078 and an R^2 of ~ 0.0493 . For the purpose of this demo, let's consider this an accurate enough model to deploy and use for predictions.

Run-21

Overview Session Build

Configuration

Script	fit.py
Arguments	
Comment	
Build Snapshot	cd61c8ac443de924189a55c5562d29268bbcb539
Created At	6/21/18 6:06 PM
Submitter	admin

Metrics

mean_sq_err	0.007866659505691643
r2	0.04934628330010382

Output

model.pkl

Add to Project

6. Once you have finished training and comparing metrics from different experiments, go to the experiment that generated the best model. From the experiment's **Overview** page, select the `model.pkl` file and click **Add to Project**.

This saves the model to the project filesystem, available on the project's **Files** page. We will now deploy this model as a REST API that can serve predictions.

Deploy the Model

To deploy the model we use the `predict.py` script from the Python template project. This script contains the `predict` function that accepts petal length as input and uses the model built in the previous step to predict petal width.

1. Navigate to the Iris project's **Overview > Models** page.
2. Click **New Model** and fill out the fields. Make sure you use the Python 3 kernel. For example:

3. Deploy the model.
4. Click on the model to go to its **Overview** page. As the model builds you can track progress on the **Build** page. Once deployed, you can see the replicas deployed on the **Monitoring** page.
5. To test the model, use the **Test Model** widget on the model's **Overview** page.

Test Model

Input

```
{
  "petal_length": 5.4
}
```

Test
Reset

Result

Status	● success
Response	1.8826221434150965
Replica ID	predict-petal-width-2-9-7cf557b957-5wj

Model Monitoring and Administration

This topic describes how to monitor active models and some tasks related to general model administration:

Monitoring Active Models

Active Model - A model that is in the **Deploying**, **Deployed**, or **Stopping** stages.

Cloudera Data Science Workbench provides two ways to monitor active models:

Monitoring Individual Models

When a model is deployed, Cloudera Data Science Workbench allows you to specify a number of replicas that will be deployed to serve requests. For each active model, you can monitor its replicas by going to the model's **Monitoring** page. On this page you can track the number of requests being served by each replica, success and failure rates, and their associated `stderr` and `stdout` logs. Depending on future resource requirements, you can increase or decrease the number of replicas by [re-deploying the model](#).

The most recent logs are at the top of the pane (see image). `stderr` logs are displayed next to a red bar while `stdout` logs are by a green bar. Note that model logs and statistics are only preserved so long as the individual replica is active. When a replica restarts (for example, in case of bad input) the logs also start with a clean slate.

The screenshot shows the 'Add Two Numbers' model monitoring page. At the top right, there are buttons for 'Deployed', 'Stop', 'Restart', and 'Deploy New Build'. Below the title, there are tabs for 'Overview', 'Deployments', 'Builds', 'Monitoring' (selected), and 'Settings'. A table displays the status of three replicas, all of which are 'Ready' and have processed 6 requests with 100% success and 0 failures. Below the table, there is a 'Streams' section with checkboxes for 'stdout' and 'stderr'. The log output shows several successful 'Added the numbers' messages and one informational message about model initialization.

Replica	Status	Received	Processed	Success	Failure	Error	Busy	Not Ready	Restart
add-numbers-1-7-664d6854bd-4z2bv	Ready	6	6 (100 %)	6 (100 %)	0	0	0	0	0
add-numbers-1-7-664d6854bd-qgg5j	Ready	5	5 (100 %)	5 (100 %)	0	0	0	0	0
add-numbers-1-7-664d6854bd-qxlnr	Ready	5	5 (100 %)	5 (100 %)	0	0	0	0	0

```

Streams:  stdout  stderr
2018-06-08 11:28:01.998 Added the numbers - success!
2018-06-08 11:23:56.804 Added the numbers - success!
2018-06-08 11:23:51.799 Added the numbers - success!
2018-06-08 11:14:21.244 Added the numbers - success!
2018-06-08 11:13:51.420 2018-06-08 18:13:51,420 36 INFO Model.Runtime Finish Model initialization
2018-06-08 11:13:51.420 Model_ready
  
```

Monitoring All Active Models

Required Role: [Site Administrator](#)

To see a complete list of all the models that have been deployed on a deployment, and review resource usage across the deployment by models alone, go to **Admin > Models**. On this page, site administrators can also **Stop/Restart/Rebuild** any of the currently deployed models.

The screenshot shows the 'Site Administration' page with the 'Models' tab selected. It features four summary cards: '1 Active Models', '3 Active Model Replicas', '3 Total Requested CPU', and '6 Total Requested Memory (GiB)'. Below these cards is a table of active models. The 'Add Two Numbers' model is listed with 3/3 replicas, 3 CPU, and 6 GiB memory. The 'Models' tab in the navigation bar is circled in red.

Model	Project	Status	Replicas	CPU	Memory	Deployed By	Last Deployed	Actions
Add Two Numbers	Model Sample Project	Deployed	3 / 3	3	6 GiB		Jun 7, 2018, 1:49 PM	Stop

Deleting a Model



Important:

- You must stop all active deployments before you delete a model. If not stopped, active models will continue serving requests and consuming resources even though they do not show up in Cloudera Data Science Workbench UI.
- Deleted models are not actually removed from disk. That is, this operation will not free up storage space.

Deleting a model removes all of the model's builds and its deployment history from Cloudera Data Science Workbench.

To delete a model, go to the model **Overview > Settings** and click **Delete Model**.

You can also delete specific builds from a model's history by going to the model's **Overview > Build** page.

Disabling the Models Feature

Required Role: [Site Administrator](#)



Important: The feature flag mentioned here only hides the Models feature from the UI. It will not stop any active models that have already been deployed. Make sure you stop all active models from the **Admin > Models** page before you disable the feature.

To disable this feature on your Cloudera Data Science Workbench deployment:

1. Log in to Cloudera Data Science Workbench.
2. Click **Admin > Settings**.
3. Under the **Feature Flags** section, disable the **Enable users to create models** checkbox.

Debugging Issues with Models

This topic describes some common issues to watch out for during different stages of the model [build and deployment process](#).

As a general rule, if your model spends too long in any of the afore-mentioned stages, check the resource consumption statistics for the cluster. When the cluster starts to run out of resources, often models will spend some time in a queue before they can be executed.

Resource consumption by active models on a deployment can be tracked by site administrators on the **Admin > Models** page.

Building

Live progress for this stage can be tracked on the model's **Build** tab. It shows the details of the [build process](#) that creates a new Docker image for the model. Potential issues:

- If you specified a custom build script (`cdsw-build.sh`), ensure that the commands inside the script complete successfully.
- If you are in an environment with restricted network connectivity, you might need to manually upload dependencies to your project and install them from local files.

Pushing

Once the model has been built, it is copied to an internal Docker registry to make it available to all the Cloudera Data Science Workbench hosts. Depending on network speeds, your model may spend some time in this stage.

Deploying

If you see issues occurring when Cloudera Data Science Workbench is attempting to start the model, use the following guidelines to begin troubleshooting:

- Make sure your model code works in a workbench session. To do this, launch a new session, run your model file, and then interactively call your target function with the input object. For a simple example, see the [Model Quickstart](#).
- Ensure that you do not have any syntax errors. For python, make sure you have the kernel with the appropriate python version (Python 2 or Python 3) selected for the syntax you have used.
- Make sure that your `cdsw-build.sh` file provides a complete set of dependencies. Dependencies manually installed during a session on the workbench are not carried over to your model. This is to ensure a clean, isolated, build for each model.

- If your model accesses resources such as data on the CDH cluster or an external database make sure that those resources can accept the load your model may exert on them.

Deployed

Once a model is up and running, you can track some basic logs and statistics on the model's [Monitoring](#) page. In case issues arise:

- Check that you are handling bad input from users. If your function throws an exception, Cloudera Data Science Workbench will restart your model to attempt to get back to a known good state. The user will see an unexpected model shutdown error.

For most transient issues, model replicas will respond by restarting on their own before they actually crash. This auto-restart behavior should help keep the model online as you attempt to debug runtime issues.

- Make runtime troubleshooting easier by printing errors and output to `stderr` and `stdout`. You can catch these on each model's **Monitoring** tab. Be careful not to log sensitive data here.
- The **Monitoring** tab also displays the status of each replica and will show if the replica cannot be scheduled due to a lack of cluster resources. It will also display how many requests have been served/dropped by each replica.

When Cloudera Data Science Workbench receives a call request for a model, it attempts to find a free replica that can answer the call. If the first arbitrarily selected replica is busy, Cloudera Data Science Workbench will keep trying to contact a free replica for 30 seconds. If no replica is available, Cloudera Data Science Workbench will return a `model.busy` error with HTTP status code 429 (Too Many Requests). If you see such errors, [re-deploy the model build](#) with a higher number of replicas.

Analytical Applications

This feature gives data scientists a way to create web applications/dashboards and easily share them with other business stakeholders. Applications can range from single visualizations embedded in reports, to rich dashboard solutions such as Tableau. They can be interactive or non-interactive.

Applications stand alongside other existing forms of workloads in CDSW (sessions, jobs, experiments, models). Like all other workloads, applications must be created within the scope of a project. Each application is launched within its own isolated engine. Additionally, like models, engines launched for applications do not time out automatically. They will run as long as the web application needs to be accessible by any users and must be stopped manually when needed.



Note: Only logged-in CDSW users can view the Applications.

Testing applications before you deploy

Before you deploy an application using the steps described here, make sure your application has been thoroughly tested. You can use sessions to develop, test, and debug your applications. You can test web apps by embedding them in sessions as described here: [Web Applications Embedded in Cludera Data Science Workbench](#) on page 142

1. Go to a project's **Overview** page.
2. Click **Applications**.
3. Click **New Application**.
4. Fill out the following fields.
 - **Name:** Enter a name for the application.
 - **Subdomain:** Enter a subdomain that will be used to construct the URL for the web application. For example, if you use `test-app` as the subdomain, the application will be accessible at `test-app.<csw-domain-name>`.
Subdomains should be valid DNS hostname characters: letters from a to z, digits from 0 to 9, and the hyphen.
 - **Description:** Enter a description for the application.
 - **Script:** Select a script that hosts a web application on either `CDSW_READONLY_PORT` or `CDSW_APP_PORT`. Applications running on either of these ports are available to any users with at least read access to the project. The Python template project includes an `entry.py` script that you can use to test this out.



Note: CDSW does not prevent you from running an application that allows a read-only user (i.e. Viewers) to modify files belonging to the project. It is up to you to make the application truly read-only in terms of files, models, and other resources belonging to the project.

- **Engine Kernel and Engine Profile:** Select the kernel and computing resources needed for this application.

5. Click **Create Application**.

In a few minutes, you should see the application status change to **Running** on the **Applications** page. Click on the name of the application to access the web application interface.

You can **Stop**, **Restart**, or **Delete** an application from the **Applications** page.

If you want to make changes to an existing application, click **Overview** under the application name. Then go to the **Settings** tab to make any changes and update the application.

Limitations

This topic lists all the limitations associated with the Applications feature.

- **Securing project resources**

CDSW applications are accessible by any user with read-only (or higher) access to the project. However, CDSW does not actively prevent you from running an application that allows a read-only user (for example, Viewers) to modify files belonging to the project. It is up to you to make the application truly read-only in terms of files, models, and other resources belonging to the project.

- **Port availability**

CDSW exposes only 3 ports per project. This means, you can run a maximum of 3 web applications simultaneously:

- one on `CDSW_APP_PORT`
- one on `CDSW_READONLY_PORT`
- and, one on the now-deprecated `CDSW_PUBLIC_PORT`

However, by default the editors feature runs third-party browser-based editors on `CDSW_APP_PORT`. Therefore, for projects that are already using browser-based third-party editors, you are left with only 2 other ports to run applications on: `CDSW_READONLY_PORT` and `CDSW_PUBLIC_PORT`.

Managing Jobs and Pipelines in Cloudera Data Science Workbench

Cloudera Data Science Workbench allows you to automate analytics workloads with a built-in job and pipeline scheduling system that supports real-time monitoring, job history, and email alerts. A *job* automates the action of launching an engine, running a script, and tracking the results, all in one batch process. Jobs are created within the purview of a single project and can be configured to run on a recurring schedule. You can customize the engine environment for a job, set up email alerts for successful or failed job runs, and email the output of the job to yourself or a colleague.

As data science projects mature beyond ad hoc scripts, you might want to break them up into multiple steps. For example, a project may include one or more data acquisition, data cleansing, and finally, data analytics steps. For such projects, Cloudera Data Science Workbench allows you to schedule multiple jobs to run one after another in what is called a *pipeline*, where each job is dependent on the output of the one preceding it.

Creating a Job

Jobs are created within the scope of a project. When you create a job, you will be asked to select a script to execute as part of the job, and create a schedule for when the job should run. Optionally, you can configure a job to be dependent on another existing job, thus creating a pipeline of tasks to be accomplished in a sequence. Note that the script files and any other job dependencies must exist within the scope of the same project.

1. Navigate to the project for which you want to create a job.
2. On the left-hand sidebar, click **Jobs**.
3. Click **New Job**.
4. Enter a **Name** for the job.
5. Select a script to execute for this job by clicking on the folder icon. You will be able to select a script from a list of files that are already part of the project. To upload more files to the project, see [Managing Files](#) on page 128.
6. Depending on the code you are running, select an **Engine Kernel** for the job from one of the following options: Python 2, Python 3, R, or Scala.
7. Select a **Schedule** for the job runs from one of the following options.
 - **Manual** - Select this option if you plan to run the job manually each time.
 - **Recurring** - Select this option if you want the job to run in a recurring pattern every X minutes, or on an hourly, daily, weekly or monthly schedule.
 - **Dependent** - Use this option when you are building a pipeline of jobs to run in a predefined sequence. From a dropdown list of existing jobs in this project, select the job that this one should depend on. Once you have configured a dependency, this job will run only after the preceding job in the pipeline has completed a successful run.
8. Select an **Engine Profile** to specify the number of cores and memory available for each session.
9. Enter an optional timeout value in minutes.
10. Click **Set environment variables** if you want to set any values to override the overall project environment variables.
11. Specify a list of **Job Report Recipients** to whom you can send email notifications with detailed job reports for job success, failure, or timeout. You can send these reports to yourself, your team (if the project was created under a team account), or any other external email addresses.
12. Add any **Attachments** such as the console log to the job reports that will be emailed.
13. Click **Create Job**.

Starting with version 1.1.x, you can use the Jobs API to schedule jobs from third party workflow tools. For details, see [Cloudera Data Science Workbench Jobs API](#) on page 200.

Creating a Pipeline

The **Jobs** overview presents a list of all existing jobs created for a project along with a dependency graph to display any pipelines you've created. Job dependencies do not need to be configured at the time of job creation. Pipelines can be created after the fact by modifying the jobs to establish dependencies between them. From the job overview, you can modify the settings of a job, access the history of all job runs, and view the session output for individual job runs.

Let's take an example of a project that has two jobs, Read Weblogs and Write Weblogs. Given that you must read the data before you can run analyses and write to it, the Write Weblogs job should only be triggered after the Read Weblogs job completes a successful run. To create such a two-step pipeline:

1. Navigate to the project where the Read Weblogs and Write Weblogs jobs were created.
2. Click **Jobs**.
3. From the list of jobs, select Write Weblogs.
4. Click the **Settings** tab.
5. Click on the Schedule dropdown and select **Dependent**. Select Read Weblogs from the dropdown list of existing jobs in the project.
6. Click **Update Job**.

Viewing Job History

1. Navigate to the project where the job was created.
2. Click **Jobs**.
3. Select the relevant job.
4. Click the **History** tab. You will see a list of all the job runs with some basic information such as who created the job, run duration, and status. Click individual runs to see the session output for each run.

Cloudera Data Science Workbench Jobs API

Cloudera Data Science Workbench exposes a REST API that allows you to schedule jobs from third-party workflow tools. You must authenticate yourself before you can use the API to submit a job run request. The Jobs API supports [HTTP Basic Authentication](#), accepting the same users and credentials as Cloudera Data Science Workbench.

API Key Authentication

Cloudera recommends using your API key for requests instead of your actual username/password so as to avoid storing and sending your credentials in plaintext. The API key is a randomly generated token that is unique to each user. It must be treated as highly sensitive information because it can be used to start jobs via the API. To look up your Cloudera Data Science Workbench API key:

1. Sign in to Cloudera Data Science Workbench.
2. From the upper right drop-down menu, switch context to your personal account.
3. Click **Settings**.
4. Select the **API Key** tab.

The following example demonstrates how to construct an HTTP request using the standard [basic authentication](#) technique. Most tools and libraries, such as Curl and Python Requests, support basic authentication and can set the required **Authorization** header for you. For example, with `curl` you can pass the API Key to the `--user` flag and leave the password field blank.

```
curl -v -XPOST http://cdsw.example.com/api/v1/<path_to_job> --user "<API_KEY>:"
```

To access the API using a library that does not provide Basic Authentication convenience methods, set the request's **Authorization** header to `Basic <API_KEY_encoded_in_base64>`. For example, if your API key is

uysgxtj7jzkps96njextnxxmq05usp0b, set **Authorization** to Basic
dX1zZ3h0ajdqemtwczk2bmp1eHRueHhtcTAldXNwMGI6.

Starting a Job Run Using the API

Once a job has been [created and configured](#) through the Cloudera Data Science Workbench web application, you can start a run of the job through the API. This will constitute sending a POST request to a job start URL of the form:
`http://cdsw.example.com/api/v1/projects/<$USERNAME>/<$PROJECT_NAME>/jobs/<$JOB_ID>/start.`

To construct a request, use the following steps to derive the username, project name, and job ID from the job's URL in the web application.

1. Log in to the Cloudera Data Science Workbench web application.
2. Switch context to the team/personal account where the parent project lives.
3. Select the project from the list.
4. From the project's **Overview**, select the job you want to run. This will take you to the job **Overview** page. The URL for this page is of the form: `http://cdsw.example.com/<$USERNAME>/<$PROJECT_NAME>/jobs/<$JOB_ID>`.
5. Use the `$USERNAME`, `$PROJECT_NAME`, and `$JOB_ID` parameters from the job Overview URL to create the following job start URL:

`http://cdsw.example.com/api/v1/projects/<$USERNAME>/<$PROJECT_NAME>/jobs/<$JOB_ID>/start.`

For example, if your job Overview page has the URL

`http://cdsw.example.com/alice/sample-project/jobs/123`, then a sample POST request would be of the form:

```
curl -v -XPOST http://cdsw.example.com/api/v1/projects/alice/sample-project/jobs/123/start \
--user "<API_KEY>:" --header "Content-type: application/json"
```

Note that the request must have the **Content-Type** header set to `application/json`, even if the request body is empty.

Setting Environment Variables

You can set environment variables for a job run by passing parameters in the API request body in a JSON-encoded object with the following format.

```
{
  "environment": {
    "ENV_VARIABLE": "value 1",
    "ANOTHER_ENV_VARIABLE": "value 2"
  }
}
```

The values set here will override the defaults set for the project and the job in the web application. This request body is optional and can be left blank.

Be aware of potential conflicts with existing defaults for environment variables that are crucial to your job, such as `PATH` and the [CDSW * variables](#).

Sample Job Run

As an example, let's assume user Alice has created a project titled Risk Analysis. Under the Risk Analysis project, Alice has created a job with the ID, 208. Using `curl`, Alice can use her API Key (`uysgxtj7jzkps96njextnxxmq05usp0b`) to create an API request as follows:

```
curl -v -XPOST http://cdsw.example.com/api/v1/projects/alice/risk-analysis/jobs/208/start \
--user "uysgxtj7jzkps96njextnxxmq05usp0b:" --header "Content-type: application/json" \
--data '{"environment": {"START_DATE": "2017-01-01", "END_DATE": "2017-01-31"}}'
```

Managing Jobs and Pipelines in Cloudera Data Science Workbench

In this example, `START_DATE` and `END_DATE` are environment variables that are passed as parameters to the API request in a JSON object.

In the resulting HTTP request, `curl` automatically encodes the **Authorization** request header in base64 format.

```
* Connected to cds.example.com (10.0.0.3) port 80 (#0)
* Server auth using Basic with user 'uysgxtj7jzkps96njextnxxmq05usp0b'
> POST /api/v1/projects/alice/risk-analysis/jobs/21/start HTTP/1.1
> Host: cds.example.com
> Authorization: Basic dXlzZ3h0ajdqemtwczk2bmp1eHRueHhtcTAldXNwMGI6
> User-Agent: curl/7.51.0
> Accept: */*
> Content-type: application/json
>
< HTTP/1.1 200 OK
< Access-Control-Allow-Origin: *
< Content-Type: application/json; charset=utf-8
< Date: Mon, 10 Jul 2017 12:00:00 GMT
< Vary: Accept-Encoding
< Transfer-Encoding: chunked
<
{
  "engine_id": "cwg6wclmg0x482u0"
}
```

You can confirm that the job was started by going to the Cloudera Data Science Workbench web application.

Starting a Job Run Using Python

To start a job run using Python, Cloudera recommends using [Requests](#), an HTTP library for Python; it comes with a convenient API that makes it easy to submit job run requests to Cloudera Data Science Workbench. Extending the Risk Analysis example from the previous section, the following sample Python code will create an HTTP request to run the job with the job ID, 208.

Python 2

```
# example.py

import requests
import json

HOST = "http://cds.example.com"
USERNAME = "alice"
API_KEY = "uysgxtj7jzkps96njextnxxmq05usp0b"
PROJECT_NAME = "risk-analysis"
JOB_ID = "208"

url = "/".join([HOST, "api/v1/projects", USERNAME, PROJECT_NAME, "jobs", JOB_ID, "start"])
job_params = {"START_DATE": "2017-01-01", "END_DATE": "2017-01-31"}
res = requests.post(
    url,
    headers = {"Content-Type": "application/json"},
    auth = (API_KEY, ""),
    data = json.dumps({"environment": job_params})
)

print "URL", url
print "HTTP status code", res.status_code
print "Engine ID", res.json().get('engine_id')
```

When you run the code, you should see output of the form:

```
python example.py
```

```
URL http://cds.example.com/api/v1/projects/alice/risk-analysis/jobs/208/start
HTTP status code 200
Engine ID r1lw5q3q589ryg9o
```

Limitations

- Job notification emails fail intermittently when attachments are included. Emails are delivered either with blank attachments or no attachments at all.
Cloudera Bug: DSE-9469, DSE-8806
- Cloudera Data Science Workbench does not support changing your API key, or having multiple API keys.
- Currently, you cannot use the Jobs API to create a job, stop a job, or get the status of a job.
- Jobs pipeline visualization in CML and CDSW no longer shows dependent jobs. Dependencies only show the first step in the chain. Previously (before upgrade to 1.7), the UI displayed the whole chain of jobs. Jobs still run in the correct order, but the UI is no longer clear. Associated Bug: DSE-8003

Cloudera Data Science Workbench Engines

In the context of Cloudera Data Science Workbench, engines are responsible for running data science workloads and intermediating access to the underlying CDH cluster. This topic gives an overview of engines and walks you through some of the ways you can customize engine environments to meet the requirements of your users and projects.

Basic Concepts and Terminology

Base Engine Image

The base engine image is a Docker image that contains all the building blocks needed to launch a Cloudera Data Science Workbench session and run a workload. It consists of *kernels* for Python, R, and Scala along with additional libraries that can be used to run common data analytics operations. When you launch a session to run a project, an *engine* is kicked off from a container of this image. The base image itself is built and shipped along with Cloudera Data Science Workbench.

New versions of the base engine image are released periodically. However, existing projects are not automatically upgraded to use new engine images. Older images are retained to ensure you are able to test code compatibility with the new engine before upgrading to it manually.

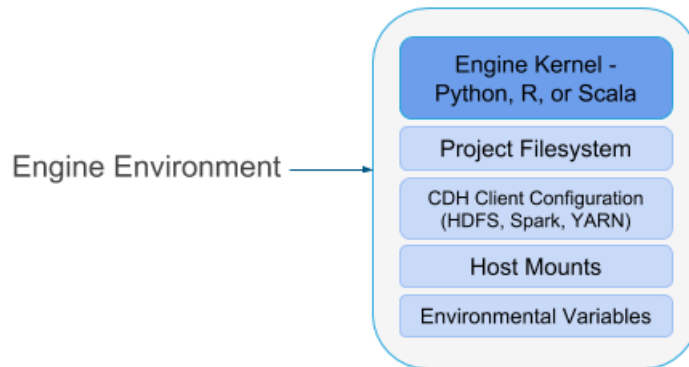
For more details on the libraries shipped within the base engine image, see [Cloudera Data Science Workbench Engine Versions and Packaging](#) on page 224.

Engine

The term *engine* refers to a virtual machine-style environment that is created when you run a project (via session or job) in Cloudera Data Science Workbench. You can use an engine to run R, Python, and Scala workloads on data stored in the underlying CDH cluster.

Cloudera Data Science Workbench allows you to run code using either a *session* or a *job*. A session is a way to interactively launch an engine and execute code while a job lets you batch process those actions and schedule them to run recursively. Each session and job launches its own engine that lives as long as the workload is running (or until it times out).

A running engine includes the following components:



- **Kernel**

Each engine runs a kernel with an R, Python or Scala process that can be used to execute code within the engine. The kernel launched differs based on the option you select (either Python 2/3, PySpark, R, or Scala) when you launch the session or configure a job.

The Python kernel is based on the Jupyter IPython kernel; the R kernel is custom-made for CDSW; and the Scala kernel is based on the Apache Toree kernel.

- **Project Filesystem Mount**

Cloudera Data Science Workbench uses a *persistent* filesystem to store project files such as user code, installed libraries, or even small data files. Project files are stored on the master host at `/var/lib/cdsw/current/projects`.

Every time you launch a new session or run a job for a project, a new engine is created, and the project filesystem is mounted into the engine's environment at `/home/cdsw`. Once the session/job ends, the only project artifacts that remain are a log of the workload you ran, and any files that were generated or modified, including libraries you might have installed. All of the installed dependencies persist through the lifetime of the project. The next time you launch a session/job for the same project, those dependencies will be mounted into the engine environment along with the rest of the project filesystem.

- **CDH and Host Mounts**

To ensure that each engine is able to access the CDH cluster, a number of folders are mounted from the CDSW gateway host into the engine's environment. For example, on a CSD deployment, this includes the path to the parcel repository (`/opt/cloudera`), client configurations for HDFS, Spark, YARN, as well as the host's `JAVA_HOME`.

Cloudera Data Science Workbench works out-of-the-box for CDH clusters that use the default file system layouts configured by Cloudera Manager. If you customized your CDH cluster's filesystem layout (for example, modified the CDH parcel directory) or if there are other files on the hosts that should be mounted into the engines, use the Site Administration panel to include them.

For detailed instructions, see [CDH Parcel Directory and Host Mounts](#).

Project Environments

This section describes how you can configure engine environments to meet the requirements of a project. This can be done by using environmental variables and by installing dependencies.

Environmental Variables

Environmental variables help you customize engine environments, both globally and for individual projects/jobs. For example, if you need to configure a particular timezone for a project or increase the length of the session/job timeout windows, you can use environmental variables to do so. Environmental variables can also be used to assign variable names to secrets, such as passwords or authentication tokens, to avoid including these directly in the code.

For a list of the environmental variables you can configure and instructions on how to configure them, see [Engine Environment Variables](#) on page 215.

Dependencies

You can provide packages, such as Python libraries, in addition to the pre-installed package through the following methods:

- Directly installing packages within projects
- Creating a custom engine with the required packages
- Mounting a path from the host which contains additional packages

One method may be more appropriate for your deployment than another method. For more information about each option, see [Managing Engine Dependencies](#) on page 211.

Configuring Engine Environments for Experiments and Models

To allow for versioning of experiments and models, Cloudera Data Science Workbench executes each experiment and model in a completely isolated engine. Every time a model or experiment is kicked off, Cloudera Data Science Workbench creates a new isolated Docker image where the model or experiment is executed. These engines are built by extending the project's designated default engine image to include the code to be executed and any dependencies as specified.

For details on how this process works and how to configure these environments, see [Engines for Experiments and Models](#) on page 206.

Engines for Experiments and Models

In Cloudera Data Science Workbench, models, experiments, jobs, and sessions are all created and executed within the context of a project. We've described the different ways in which you can customize a project's engine environment for sessions and jobs [here](#). However, engines for models and experiments are completely isolated from the rest of the project.

Every time a model or experiment is kicked off, Cloudera Data Science Workbench creates a new isolated Docker image where the model or experiment is executed. This isolation in build and execution makes it possible for Cloudera Data Science Workbench to keep track of input and output artifacts for every experiment you run. In case of models, versioned builds give you a way to retain build history for models and a reliable way to rollback to an older version of a model if needed.



The rest of this topic describes the engine build process that occurs when you kick off a model or experiment.

Snapshot Code

When you first launch an experiment or model, Cloudera Data Science Workbench takes a Git snapshot of the project filesystem at that point in time. It is important to note that this Git server functions behind the scenes and is completely separate from any other Git version control system you might be using for the project as a whole.

However, this Git snapshot will recognize the `.gitignore` file defined in the project. This means if there are any artifacts (files, dependencies, etc.) larger than 50 MB stored directly in your project filesystem, make sure to add those files or folders to `.gitignore` so that they are not recorded as part of the snapshot. This ensures that the experiment or model environment is truly isolated and does not inherit dependencies that have been previously installed in the project workspace.

By default, each project is created with the following `.gitignore` file:

```
R
node_modules
*.pyc
.*
!.gitignore
```

Augment this file to include any extra dependencies you have installed in your project workspace to ensure a truly isolated workspace for each model or experiment.

Multiple `.gitignore` files

A project can include multiple `.gitignore` files. However, there can only be one `.git` directory in the project, located at the project root, `/home/cdsw/.git`, otherwise Experiment and Model deployment fails.

If you create a blank project, and then want to clone a repo into it, clone a single project to the root of the workspace (`/home/cdsw/.git`) to ensure that Experiments and Models work.

Build Image

Once the code snapshot is available, Cloudera Data Science Workbench creates a new Docker image with a copy of the snapshot. This new image is based off the project's designated default engine image (configured at **Project Settings > Engine**). The image environment can be customized by using environmental variables and a *build script* that specifies which packages should be included in the new image.

Environmental Variables

Previously (CDSW 1.7.1 and lower), the environment variables set at the site admin level and project level did not automatically get pulled into the builds created for models and experiments. They needed to be explicitly coded into the `cdsw-build.sh` file. With CDSW 1.7.2 and higher, experiments and models will automatically inherit these admin and project-level environment variables.

Note that custom mounts or environment variables configured in `cdsw.conf` (such as `NO_PROXY`, `HTTP(S)_PROXY`, etc.) are still not passed to the container builds for experiments and models (even though they are applied to sessions, jobs, and deployed models/experiments).

For more information, see [Engine Environment Variables](#) on page 215.

Build Script - `cdsw-build.sh`

As part of the Docker build process, Cloudera Data Science Workbench runs a build script called `cdsw-build.sh` file. You can use this file to customize the image environment by specifying any dependencies to be installed for the code to run successfully. One advantage to this approach is that you now have the flexibility to use different tools and libraries in each consecutive training run. Just modify the build script as per your requirements each time you need to test a new library or even different versions of a library.



Important:

- The `cdsw-build.sh` script does not exist by default -- it has to be created by you within each project as needed.
- The name of the file is not customizable. It must be called `cdsw-build.sh`.

The following sections demonstrate how to specify dependencies in Python and R projects so that they are included in the build process for models and experiments.

Python 3

For Python, create a `requirements.txt` file in your project with a list of packages that must be installed. For example:

```
beautifulsoup4==4.6.0
seaborn==0.7.1
```

Figure 1: requirements.txt

Then, create a `cdsw-build.sh` file in your project and include the following command to install the dependencies listed in `requirements.txt`.

```
pip3 install -r requirements.txt
```

Figure 2: cdsw-build.sh

Now, when `cdsw-build.sh` is run as part of the build process, it will install the `beautifulsoup4` and `seaborn` packages to the new image built for the experiment/model.

R

For R, create a script called `install.R` with the list of packages that must be installed. For example:

```
install.packages(repos="https://cloud.r-project.org", c("tidyr", "stringr"))
```

Figure 3: install.R

Then, create a `cdsw-build.sh` file in your project and include the following command to run `install.R`.

```
Rscript install.R
```

Figure 4: `cdsw-build.sh`

Now, when `cdsw-build.sh` is run as part of the build process, it will install the `tidyr` and `stringr` packages to the new image built for the experiment/model.

If you do not specify a build script, the build process will still run to completion, but the Docker image will not have any additional dependencies installed. At the end of the build process, the built image is then pushed to an internal Docker registry so that it can be made available to all the Cloudera Data Science Workbench hosts. This push is largely transparent to the end user.



Note: If you want to test your code in an interactive session before you run an experiment or deploy a model, run the `cdsw-build.sh` script directly in the workbench. This will allow you to test code in an engine environment that is similar to one that will eventually be built by the model/experiment build process.

Run Experiment / Deploy Model

Once the Docker image has been built and pushed to the internal registry, the experiment/model can now be executed within this isolated environment.

In case of experiments, you can track live progress as the experiment executes in the experiment's **Session** tab.

Unlike experiments, models do not display live execution progress in a console. Behind the scenes, Cloudera Data Science Workbench will move on to deploying the model in a serving environment based on the computing resources and replicas you requested. Once deployed you can go to the model's **Monitoring** page to view statistics on the number of requests served/dropped and `stderr/stdout` logs for the model replicas.

Configuring Cloudera Data Science Workbench Engines

This topic describes how to configure and manage engines in Cloudera Data Science Workbench. Cloudera Data Science Workbench currently supports R, Python, and Scala engines. You can use these engines to run data science projects either in isolation, as you would on your laptop, or connect to your CDH cluster using Cloudera Distribution of Apache Spark 2 and other libraries.

Concepts and Terminology

Review basic concepts and terminology related to engines at [Cloudera Data Science Workbench Engines](#) on page 204.

Managing Engines

Required Role: [Site Administrator](#)

Site administrators and project administrators are responsible for making sure that all projects on the deployment have access to the engines they need. Site admins can create engine profiles, determine the default engine version to be used across the deployment, and white-list any custom engines that teams require. As a site administrator, you can also customize engine environments by setting global environmental variables and configuring any files/folders that need to be mounted into project environments on run time.

Managing Engine Profiles

Engine profiles define how many [vCPUs](#) and how much memory Cloudera Data Science Workbench will reserve for a particular session/job. As a site administrator you can create several different vCPU, GPU, and memory configurations which will be available when launching a session/job. When launching a new session, users will be able to select one of the available engine profiles depending on their project's requirements.

The screenshot shows the 'Site Administration' interface with the 'Engines' tab selected. The 'Engines Profiles' table is as follows:

Description	vCPU (burstable)	Memory (GiB)
1 vCPU / 2 GiB Memory	1	2
2 vCPU / 4 GiB Memory	2	4
4 vCPU / 8 GiB Memory	4	8
8 vCPU / 16 GiB Memory	8	16
0.25 vCPU / 1.75 GiB Memory	0.25	1.75
1 vCPU (burstable), 1.75 GiB memory	<input type="text" value="1"/>	<input type="text" value="1.75"/>

The 'Start New Session' dropdown menu is open, showing the following options:

- Engine Image - Configure
 - Base Image v4 - docker.repository.cloudera.com/cdsw/engine:4
- Select Engine Kernel
 - Python 2
 - Python 3
 - Scala
 - R
- Select Engine Profile
 - 1 vCPU / 2 GiB Memory
 - 2 vCPU / 4 GiB Memory
 - 4 vCPU / 8 GiB Memory
 - 8 vCPU / 16 GiB Memory
 - 0.25 vCPU / 1.75 GiB Memory

To create engine profiles, go to the **Admin > Engines** page, under **Engines Profiles**. Cloudera recommends that all profiles include at least 2 GB of RAM to avoid out of memory errors for common user operations.

You will see the option to add GPUs to the engine profiles only if your Cloudera Data Science Workbench hosts are equipped with GPUs, and you have enabled them for use by setting the relevant properties either in Cloudera Manager (for CSD) or in [cdsw.conf](#) (for RPM).

Managing Engine Images

By default, Cloudera Data Science Workbench ships a base engine image that includes *kernels* for Python, R, and Scala, along with some [additional libraries](#) that can be used to run common data analytics operations. Occasionally, new engine versions are released and shipped with Cloudera Data Science Workbench releases.

Engine images are available in the Site Administrator panel at **Admin > Engines**, under the **Engine Images** section. As a site administrator, you can select which engine version is used by default for new projects. Furthermore, project administrators can explicitly select which engine image should be used as the default image for a project. To do so, go to the project's Overview page and click **Settings** on the left navigation bar.

If a user publishes a new custom Docker image, site administrators are responsible for white-listing such images for use across the deployment. For more information on creating and managing custom Docker images, see [Customized Engine Images](#) on page 220.

Configuring the Engine Environment

This section describes some of the ways you can configure engine environments to meet the requirements of your projects.

Environmental Variables

For information on how environmental variables can be used to configure engine environments in Cloudera Data Science Workbench, see [Engine Environment Variables](#) on page 215.

CDH Parcel Directory

Starting with Cloudera Data Science Workbench 1.5, the **CDH parcel directory** property is no longer available in the Site Administration panel. By default, Cloudera Data Science Workbench looks for the CDH parcel at `/opt/cloudera/parcels`.

If you want to use a custom location for your parcels, use one of the following methods to configure this custom location:

CSD deployments: If you are using the default parcel directory, `/opt/cloudera/parcels`, no action is required. If you want to use a custom location for the parcel directory, configure this in Cloudera Manager as documented [here](#).

OR

RPM deployments: If you are using the default parcel directory, `/opt/cloudera/parcels`, no action is required. If you want to specify a custom location for the parcel directory, configure the `DISTRO_DIR` property in the `cdsw.conf` file on both master and worker hosts. Run `cdsw restart` after you make this change.

Configuring Host Mounts

By default, Cloudera Data Science Workbench will automatically mount the CDH parcel directory and client configuration for required services such as HDFS, Spark, and YARN into each project's engine. However, if users want to reference any additional files/folders on the host, site administrators will need to configure them here so that they are loaded into engine containers at runtime. Note that the directories specified here will be available to all projects across the deployment.



Note: Host mounts must be mounted over NFS on all CDSW hosts. Each user session mounts the folder from the server on which it's running. This mount only works if all CDSW hosts have the mounted folder.

To configure additional mounts, go to **Admin > Engines** and add the paths to be mounted from the host to the **Mounts** section.

Mounts

Select extra folders to be mounted from the host to all sessions and jobs.

Path	Write Access	Actions
/tmp/test	<input checked="" type="checkbox"/>	Delete
/tmp/readonly	<input type="checkbox"/>	Delete
<input type="text"/>	<input type="checkbox"/>	Add

By default, these folders are mounted into engines with read-only permissions. Use the checkbox to enable write access.

The following table summarizes how mounts are loaded into engine containers in current and previous Cloudera Data Science Workbench releases.

CDSW Version	Mount Point Permissions in Engines
1.4.2 (and higher)	By default, mount points are loaded into engine containers with read-only permissions. CDSW 1.4.2 (and higher) also include a Write Access checkbox (see image) that you can use to enable read-write access for individual mounted directories. Note that these permissions will apply to all projects across the deployment.
1.4.0	Mount points are loaded into engine containers with read-only permissions.
1.3.x (and lower)	Mount points are loaded into engine containers with read-write permissions.

Points to Remember:

- When adding host mounts, try to be as generic as possible without mounting common system files. For example, if you want to add several files under `/etc/spark2-conf`, you can simplify and mount the `/etc/spark2-conf` directory; but adding the parent `/etc` might prevent the engine from running.

As a general rule, do not mount full system directories that are already in use; such as `/var`, `/tmp`, or `/etc`. This also serves to avoid accidentally exposing confidential information in running sessions. Do not set `JAVA_HOME` to a path under these directories because CDSW sessions will mount the `JAVA_HOME` location from the host in the session engine container which can interfere with the operation of the engine container.

- Do not add duplicate mount points. This will lead to sessions crashing in the workbench.

Configuring Shared Memory Limit for Docker Images

You can increase the shared memory size for the sessions, experiments, and jobs running within an Engine container within your project. For Docker, the default size of the available shared memory is 64MB.

To increase the shared memory limit:

1. From CDSW web UI, go to **Projects > Project Settings > Engine > Advanced Settings**.
2. Specify the shared memory size in the **Shared Memory Limit** field.
3. Click **Save Advanced Settings** to save the configuration and exit.

This mounts a volume with the `tmpfs` file system to `/dev/shm` and Kubernetes will enforce the given limit. The maximum size of this volume is the half of your physical RAM in the node without the swap.

Setting Time Zones for Sessions and Jobs

The default time zone for Cloudera Data Science Workbench sessions is UTC. This is the default regardless of the time zone setting on the Master host.

To change to your preferred time zone, for example, Pacific Standard Time (PST), navigate to **Admin > Engines**. Under the **Environmental Variables** section, add a new variable with the name set to `TZ` and value set to `America/Los_Angeles`, and click **Add**.

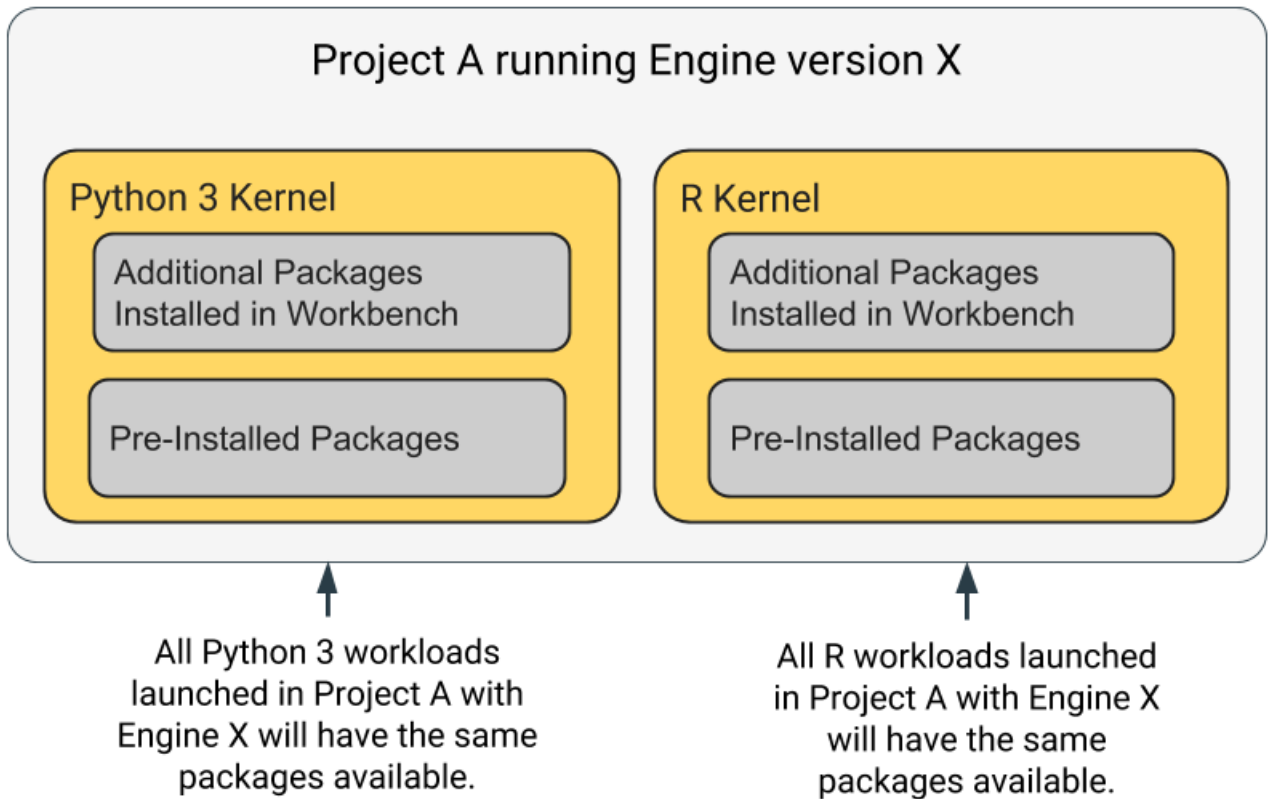
Managing Engine Dependencies

This page describes the options available to you for mounting a project's dependencies into its engine environment:



Important: Even though experiments and models are created within the scope of a project, the engines they use are completely isolated from those used by sessions or jobs launched within the same project. For details, see [Engines for Experiments and Models](#) on page 206.

Installing Packages Directly Within Projects



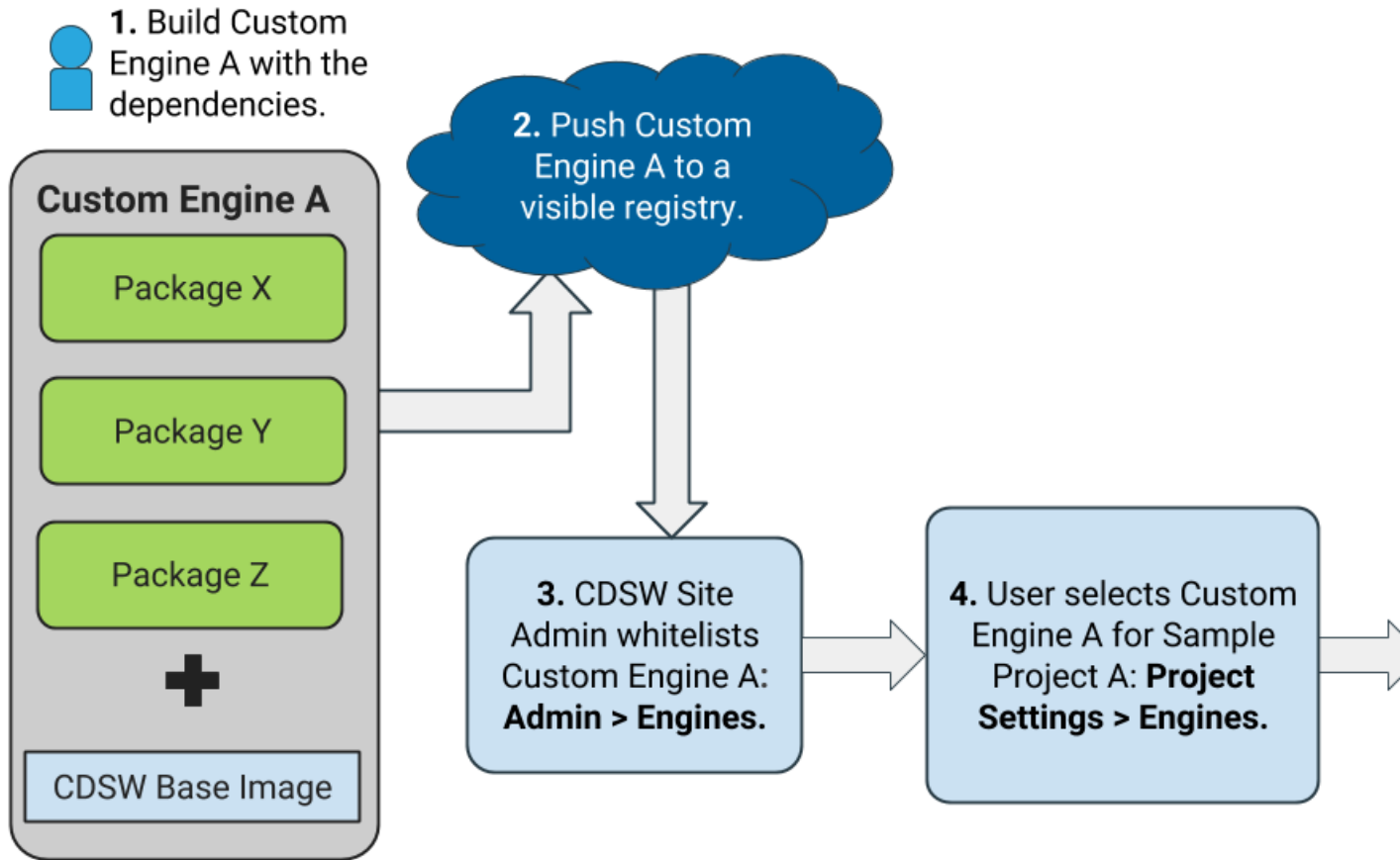
Cloudera Data Science Workbench engines are preloaded with a few common packages and libraries for R, Python, and Scala. In addition to these, Cloudera Data Science Workbench allows you to install any other packages or libraries required by your projects just as you would on your local computer. Each project's environment is completely isolated from others, which means you can install different versions of libraries pinned to different projects.

Libraries can be installed from the workbench using the inbuilt interactive command prompt or the terminal. Any dependencies installed this way are mounted to the project environment at `/home/cdsw`. Alternatively, you could choose to use a package manager such as [Conda](#) to install and maintain packages and their dependencies.

Note that overriding pre-installed packages by installing packages directly in the workbench can have unwanted side effects. It is not recommended or supported.

For detailed instructions, see [Installing Additional Packages](#) on page 218.

Creating a Customized Engine with the Required Package(s)



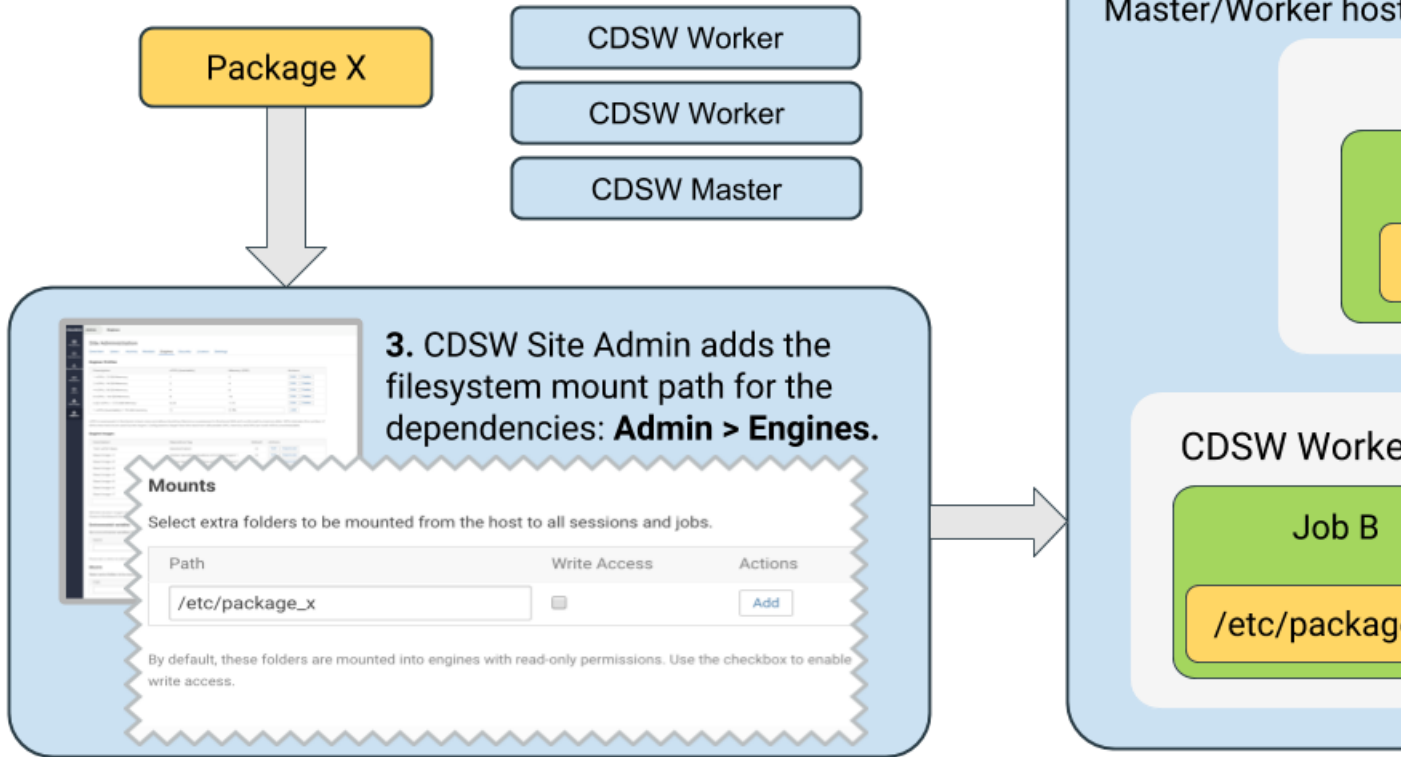
Directly installing a package to a project as described above might not always be feasible. For example, packages that require root access to be installed, or that must be installed to a path outside `/home/cdsw` (outside the project mount), cannot be installed directly from the workbench. For such circumstances, Cloudera recommends you extend the base Cloudera Data Science Workbench engine image to build a customized image with all the required packages installed to it.

This approach can also be used to accelerate project setup across the deployment. For example, if you want multiple projects on your deployment to have access to some common dependencies out of the box or if a package just has a complicated setup, it might be easier to simply provide users with an engine environment that has already been customized for their project(s).

For detailed instructions with an example, see [Customized Engine Images](#) on page 220

Mounting Additional Dependencies from the Host

1. CDSW Site Admin selects a sub-tree of the filesystem to act as the mount path for dependencies.
2. Site Admin (as a root user) installs Package X to that sub-tree on **ALL** CDSW Gateway hosts.



As described previously, all Cloudera Data Science Workbench projects run within an engine. By default, Cloudera Data Science Workbench automatically mounts the CDH parcel directory and client configuration for required services, such as HDFS, Spark, and YARN, into the engine. However, if users want to reference any additional files/folders on the host, site administrators need to explicitly load them into engine containers at runtime.

Limitation

If you use this option, you must self-manage the dependencies installed on the gateway hosts to ensure consistency across them all. Cloudera Data Science Workbench cannot manage or control the packages with this method. For example, you must manually ensure that version mismatches do not occur. Additionally, this method can lead to issues with experiment repeatability since CDSW does not control or keep track of the packages on the host. Contrast this with [Installing Packages Directly Within Projects](#) on page 212 and [Creating a Customized Engine with the Required Package\(s\)](#) on page 213, where Cloudera Data Science Workbench is aware of the packages and manages them.

For instructions on how to configure host mounts, see [Configuring Host Mounts](#) on page 210. Note that the directories specified here will be available to all projects across the deployment.

Managing Dependencies for Spark 2 Projects

With Spark projects, you can add external packages to Spark executors on startup. To add external dependencies to Spark jobs, specify the libraries you want added by using the appropriate configuration parameters in a `spark-defaults.conf` file.

For a list of the relevant properties and examples, see [Managing Dependencies for Spark 2 Jobs](#) on page 240.

Engine Environment Variables

Environmental variables allow you to customize engine environments for projects. For example, if you need to configure a particular timezone for a project, or increase the length of the session/job timeout windows, you can use environmental variables to do so. Environmental variables can also be used to assign variable names to secrets such as passwords or authentication tokens to avoid including these directly in the code.

In general, Cloudera recommends that you do not include passwords, tokens, or any other secrets directly in your code because anyone with read access to your project will be able to view this information. A better place to store secrets is in your project's environment variables, where only project collaborators and admins have view access. They can therefore be used to securely store confidential information such as your AWS keys or database credentials.

Cloudera Data Science Workbench allows you to define environmental variables for the following scopes:

Global

A site administrator for your Cloudera Data Science Workbench deployment can set environmental variables on a global level. These values will apply to every project on the deployment.

To set global environmental variables, go to **Admin > Engines**.

Project

Project administrators can set project-specific environmental variables to customize the engines launched for a project. Variables set here will override the global values set in the site administration panel.

To set environmental variables for a project, go to the project's Overview page and click **Settings > Engine**.

Job

Environments for individual jobs within a project can be customized while creating the job. Variables set per-job will override the project-level and global settings.

To set environmental variables for a job, go to the job's Overview page and click **Settings > Set Environmental Variables**.

Experiments

Engines created for execution of experiments are completely isolated from the project. However, these engines inherit values from environmental variables set at the project-level and/or global level. Variables set at the project-level will override the global values set in the site administration panel.

Models

Model environments are completely isolated from the project. Environmental variables for these engines can be configured during the [build stage](#) of the model deployment process. Models will also inherit any environment variables set at the project and global level. However, variables set per-model build will override other settings.



Important:

Note that custom mounts or environment variables configured in `cdsw.conf` (such as `NO_PROXY`, `HTTP(S)_PROXY`, etc.) are still not passed to the container builds for experiments and models (even though they are applied to sessions, jobs, and deployed models/experiments).

Environment Variables from Cloudera Manager

In addition to the environment variables that you can specify with different scopes, Cloudera Data Science Workbench inherits a set of environment variables from Cloudera Manager:

- `HTTP_PROXY`
- `HTTPS_PROXY`

- ALL_PROXY
- NO_PROXY

For information about what these variables are used for, see [Configuring Cloudera Data Science Workbench Deployments Behind a Proxy](#) on page 290.

Site and project administrators can change these values by manually modifying them at the project or global level. The values set within Cloudera Data Science Workbench take precedence over the ones inherited from Cloudera Manager.

Accessing Environmental Variables from Projects

Environmental variables are injected into every engine launched for a project, contingent on the scope at which the variable was set (global, project, etc.). The following code samples show how to access a sample environment variable called `DATABASE_PASSWORD` from your project code.

R

```
database.password <- Sys.getenv("DATABASE_PASSWORD")
```

Python

```
import os
database_password = os.environ["DATABASE_PASSWORD"]
```

Scala

```
System.getenv("DATABASE_PASSWORD")
```

Appending Values to Environment Variables:

You can also set environment variables to append to existing values instead of replacing them. For example, when setting the `LD_LIBRARY_PATH` variable, you can set the value to `LD_LIBRARY_PATH:/path/to/set`.


Engine Environment Variables

The following table lists Cloudera Data Science Workbench environment variables that you can use to customize your experience within the Workbench console. These can be set either as a site administrator or within the scope of a project or a job.

Environment Variable	Description
MAX_TEXT_LENGTH	Maximum number of characters that can be displayed in a single text cell. By default, this value is set to 800,000 and any more characters will be truncated. Default: 800,000
SESSION_MAXIMUM_MINUTES	Maximum number of minutes a session can run before it times out. Default: 60*24*7 minutes (7 days) Maximum Value: 35,000 minutes
JOB_MAXIMUM_MINUTES	Maximum number of minutes a job can run before it times out. Default: 60*24*7 minutes (7 days) Maximum Value: 35,000 minutes
IDLE_MAXIMUM_MINUTES	Maximum number of minutes a session can remain idle before it exits. An idle session is defined as no browser interaction. Contrast this to <code>session_maximum_minutes</code> which is the total time the session is open, regardless of browser interaction.

Environment Variable	Description
	<p>Default: 60 minutes</p> <p>Maximum Value: 35,000 minutes</p>
CONDA_DEFAULT_ENV	Points to the default Conda environment so you can use Conda to install/manage packages in the Workbench. For more details on when to use this variable, see Using Conda with Cloudera Data Science Workbench on page 219.

Per-Engine Environmental Variables: In addition to the previous table, there are some more built-in environmental variables that are set by the Cloudera Data Science Workbench application itself and do not need to be modified by users. These variables are set *per-engine* launched by Cloudera Data Science Workbench and only apply within the scope of each engine.

Environment Variable	Description
CDSW_PROJECT	The project to which this engine belongs.
CDSW_ENGINE_ID	The ID of this engine. For sessions, this appears in your browser's URL bar.
CDSW_MASTER_ID	If this engine is a worker, this is the CDSW_ENGINE_ID of its master.
CDSW_MASTER_IP	If this engine is a worker, this is the IP address of its master.
CDSW_PUBLIC_PORT	<div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;">  Note: This property is deprecated as of Cloudera Data Science Workbench 1.6.0. See CDSW_APP_PORT and CDSW_READONLY_PORT for alternatives. </div> <p>A port on which you can expose HTTP services in the engine to browsers. HTTP services that bind CDSW_PUBLIC_PORT will be available in browsers at: <code>http(s)://<CDSW_ENGINE_ID>.<CDSW_DOMAIN></code>. By default, CDSW_PUBLIC_PORT is set to 8080. Setting CDSW_PUBLIC_PORT to a non-default port number is not supported. Use <code>0.0.0.0</code> as the IP.</p> <p>A direct link to these web services will be available from the grid icon in the upper right corner of the Cloudera Data Science Workbench web application, as long as the job or session is still running. For more details, see Accessing Web User Interfaces from Cloudera Data Science Workbench on page 145.</p>
CDSW_APP_PORT	<p>A port on which you can expose HTTP services in the engine to browsers. HTTP services that bind CDSW_APP_PORT will be available in browsers at: <code>http(s)://<CDSW_ENGINE_ID>.<CDSW_DOMAIN></code>. Use this port for applications that grant some control to the project, such as access to the session or terminal. Use <code>127.0.0.1</code> as the IP.</p> <p>A direct link to these web applications will be available from the grid icon in the upper right corner of the Cloudera Data Science Workbench web application as long as the job or session runs. Even if the web application itself does not have authentication enabled, only project Contributors and Admins will be able to access it. For more details, see Accessing Web User Interfaces from Cloudera Data Science Workbench on page 145.</p> <p>Note that if the Site Administrator has enabled Allow only session creators to execute commands on active sessions, then the UI is only available to the session creator. Other users will not be able to access it.</p>

Environment Variable	Description
CDSW_READONLY_PORT	<p>A port on which you can expose HTTP services in the engine to browsers. HTTP services that bind <code>CDSW_READONLY_PORT</code> will be available in browsers at: <code>http(s)://<\$CDSW_ENGINE_ID>.<\$CDSW_DOMAIN></code>. Use this port for applications that grant read-only access to project results. Use <code>127.0.0.1</code> as the IP.</p> <p>A direct link to these web applications will be available to users with from the grid icon in the upper right corner of the Cloudera Data Science Workbench web application as long as the job or session runs. Even if the web application itself does not have authentication enabled, only project collaborators will be able to access the application. For more details, see Accessing Web User Interfaces from Cloudera Data Science Workbench on page 145.</p>
CDSW_DOMAIN	The domain on which Cloudera Data Science Workbench is being served. This can be useful for iframing services, as demonstrated in the Shiny example .
CDSW_CPU_MILLICORES	The number of CPU cores allocated to this engine, expressed in thousandths of a core.
CDSW_MEMORY_MB	The number of megabytes of memory allocated to this engine.
CDSW_IP_ADDRESS	Other engines in the Cloudera Data Science Workbench cluster can contact this engine on this IP address.

Installing Additional Packages

Cloudera Data Science Workbench engines are preloaded with a few common packages and libraries for R, Python, and Scala. However, a key feature of Cloudera Data Science Workbench is the ability of different projects to install and use libraries pinned to specific versions, just as you would on your local computer.

Generally, Cloudera recommends you install *all* required packages locally into your project. This will ensure you have the exact versions you want and that these libraries will not be upgraded when Cloudera upgrades the base engine image. You only need to install libraries and packages once per project. From then on, they are available to any new engine you spawn throughout the lifetime of the project.

You can install additional libraries and packages from the workbench, using either the command prompt or the terminal. Alternatively, you might choose to use a package manager such as [Conda](#) to install and maintain packages and their dependencies. For some basic usage guidelines for Conda, see [Using Conda with Cloudera Data Science Workbench](#) on page 219.



Note:

Cloudera Data Science Workbench does not currently support installation of packages that require root access to the hosts. For such use-cases, you will need to create a new custom engine that extends the base engine image to include the required packages. For instructions, see [Creating Custom Engine Images](#).

To install a package from the command prompt:

1. Navigate to your project's **Overview** page. Click **Open Workbench** and launch a session.
2. At the [command prompt](#) in the bottom right, enter the command to install the package. Some examples using Python and R have been provided.

R

```
# Install from CRAN
install.packages("ggplot2")
```

```
# Install using devtools
install.packages('devtools')
library(devtools)
install_github("hadley/ggplot2")
```

Python 2

```
# Installing from console using ! shell operator and pip:
!pip install beautifulsoup

# Installing from terminal
pip install beautifulsoup
```

Python 3

```
# Installing from console using ! shell operator and pip3:
!pip3 install beautifulsoup4

# Installing from terminal
pip3 install beautifulsoup4
```

(Python Only) Using a Requirements File

For a Python project, you can specify a list of the packages you want in a `requirements.txt` file that lives in your project. The packages can be installed all at once using `pip/pip3`.

1. Create a new file called `requirements.txt` file within your project:

```
beautifulsoup4==4.6.0
seaborn==0.7.1
```

2. To install the packages in a Python 3 engine, run the following command in the workbench command prompt.

```
!pip3 install -r requirements.txt
```

For Python 2 engines, use `pip`.

```
!pip install -r requirements.txt
```

Using Conda with Cloudera Data Science Workbench

Cloudera Data Science Workbench recommends using `pip` for package management along with a `requirements.txt` file (as described in the previous section). However, for users that prefer Conda, the default engine in Cloudera Data Science Workbench includes two environments called `python2.7`, and `python3.6`. These environments are added to `sys.path`, depending on the version of Python selected when you launch a new session.

In Python 2 and Python 3 sessions and attached terminals, Cloudera Data Science Workbench automatically sets the `CONDA_DEFAULT_ENV` and `CONDA_PREFIX` environment variables to point to Conda environments under `/home/cdsw/.conda`.

However, Cloudera Data Science Workbench does not automatically configure Conda to pin the actual Python version. Therefore if you are using Conda to install a package, you must specify the version of Python. For example, to use Conda to install the `feather-format` package into the `python3.6` environment, run the following command in the Workbench command prompt:

```
!conda install -y -c conda-forge python=3.6.1 feather-format
```

To install a package into the `python2.7` environment, run:

```
!conda install -y -c conda-forge python=2.7.11 feather-format
```

Note that on `sys.path`, pip packages have precedence over conda packages.



Note:

- If your project is using an older base engine image (version 3 and lower), you will need to specify both the Python version as well as the Conda environment. For example:

```
!conda install -y -c conda-forge --name python3.6 python=3.6.1
feather-format
```

The Conda environment is also required when you create an extensible engine using Conda (as described in the following section).

- Cloudera Data Science Workbench does not automatically configure a Conda environment for R and Scala sessions and attached terminals. If you want to use Conda to install packages from an R or Scala session or terminal, you must manually configure Conda to install packages into the desired environment.

Creating an Extensible Engine With Conda

Cloudera Data Science Workbench also allows you to [extend its base engine image](#) to include packages of your choice using Conda. To create an extended engine:

1. Add the following lines to a Dockerfile to extend the base engine, push the engine image to your Docker registry, and whitelist the new engine for your project. For more details on this step, see [Extensible Engines](#).

Python 2

```
RUN mkdir -p /opt/conda/envs/python2.7
RUN conda install -y nbconvert python=2.7.11 -n python2.7
```

Python 3

```
RUN mkdir -p /opt/conda/envs/python3.6
RUN conda install -y nbconvert python=3.6.1 -n python3.6
```

2. Set the `PYTHONPATH` environmental variable as shown below. You can set this either globally in the site administrator dashboard, or for a specific project by going to the project's **Settings > Engine** page.

Python 2

```
PYTHONPATH=$PYTHONPATH:/opt/conda/envs/python2.7/lib/python2.7/site-packages
```

Python 3

```
PYTHONPATH=$PYTHONPATH:/opt/conda/envs/python3.6/lib/python3.6/site-packages
```

Customized Engine Images

By default, Cloudera Data Science Workbench engines are preloaded with a few common packages and libraries for R, Python, and Scala. In addition to these, Cloudera Data Science Workbench also allows you to install any other packages or libraries that are required by your projects. However, directly installing a package to a project as described above might not always be feasible. For example, packages that require root access to be installed, or that must be installed to a path outside `/home/cdsw` (outside the project mount), cannot be installed directly from the workbench.

For such circumstances, Cloudera Data Science Workbench allows you to extend the base Docker image and create a new Docker image with all the libraries and packages you require. Site administrators can then whitelist this new image

for use in projects, and project administrators set the new white-listed image to be used as the default engine image for their projects. For an end-to-end example of this process, see [End-to-End Example: MeCab](#) on page 222.



Note: You will need to remove any unnecessary Cloudera sources or repositories that are inaccessible because of the paywall.

Note that this approach can also be used to accelerate project setup across the deployment. For example, if you want multiple projects on your deployment to have access to some common dependencies (package or software or driver) out of the box, or even if a package just has a complicated setup, it might be easier to simply provide users with an engine that has already been customized for their project(s).

Creating a Customized Engine Image

This section walks you through the steps required to create your own custom engine based on the Cloudera Data Science Workbench base image. For a complete example, see [End-to-End Example: MeCab](#) on page 222.

Create a Dockerfile for the New Custom Image

The first step when building a customized image is to create a Dockerfile that specifies which packages you would like to install in addition to the base image.

For example, the following Dockerfile installs the `beautifulsoup4` package on top of the base Ubuntu image that ships with Cloudera Data Science Workbench.

```
# Dockerfile
# Specify a Cloudera Data Science Workbench base image
FROM docker.repository.cloudera.com/cdsw/engine:8
RUN rm /etc/apt/sources.list.d/*
# Update packages on the base image and install beautifulsoup4
RUN apt-get update
RUN pip install beautifulsoup4 && pip3 install beautifulsoup4
```

Build the New Image

A new custom Docker image can be built on any host where Docker binaries are installed. To install these binaries, run the following command on the host where you want to build the new image:

```
docker build -t <image-name>:<tag> . -f Dockerfile
```

If your Dockerfile makes any outside connection (for example, `apt-get update`, `pip install`, `curl`), you must add the `--network=host` option to the build command:

```
docker build --network=host -t<image-name>:<tag> . -f Dockerfile
```

Distribute the Image

Once you have built a new custom engine, use one of the following ways to distribute the new image to all your Cloudera Data Science Workbench hosts:

Push the image to a public registry such as DockerHub

For instructions, refer the Docker documentation: [docker push](#).

Push the image to your company's Docker registry

When using this method, make sure to tag your image with the following schema:

```
docker tag <image-name> <company-registry>/<user-name>/<image-name>:<tag>
```

Once the image has been tagged properly, use the following command to push the image:

```
docker push <company-registry>/<user-name>/<image-name>:<tag>
```

The MeCab example at the end of this topic uses this method.

Distribute the image manually

Use the following steps to manually distribute the image on the cluster:

1. Save the docker image as a tarball on the host where it was built

```
docker image save -o ./<new_customized_engine>.tar <image-name>
```

2. Distribute the image to *all* the Cloudera Data Science Workbench gateway hosts.

```
scp ./<new_customized_engine>.tar root@<cdsw.your_company.com>:/tmp/
```

3. Load the image on *all* the Cloudera Data Science Workbench gateway hosts.

```
docker load --input /tmp/./<new_customized_engine>.tar
```

4. To verify that the image was successfully distributed and loaded, run:

```
docker images
```

Whitelist the Image in Cloudera Data Science Workbench

White-listing a customized image in Cloudera Data Science Workbench is a two-step process.

1. Whitelist Image for the Deployment

First, a site administrator will need to clear the new image for use on the deployment.

1. Log in as a site administrator.
2. Click **Admin > Engines**.
3. Add <company-registry>/<user-name>/<image-name>:<tag> to the list of whitelisted engine images.

2. Whitelist Image for Per-Project

If you want to start using the image in a project, the project administrator will need to set this image as the default image for the project.

1. Go to the project **Settings** page.
2. Click **Engines**.
3. Select the new customized engine from the dropdown list of available Docker images. Sessions and jobs you run in your project will now have access to this engine.

End-to-End Example: MeCab

This section demonstrates how to customize the Cloudera Data Science Workbench base engine image to include the [MeCab](#) (a Japanese text tokenizer) library.

This is a sample Dockerfile that adds MeCab to the Cloudera Data Science Workbench base image.

```
# Dockerfile
FROM docker.repository.cloudera.com/cdsw/engine:8
RUN rm /etc/apt/sources.list.d/*
RUN apt-get update && \
    apt-get install -y -q mecab \
        libmecab-dev \
        mecab-ipadic-utf8 && \
```

```

apt-get clean && \
rm -rf /var/lib/apt/lists/*
RUN cd /tmp && \
git clone --depth 1 https://github.com/neologd/mecab-ipadic-neologd.git && \
/tmp/mecab-ipadic-neologd/bin/install-mecab-ipadic-neologd -y -n -p
/var/lib/mecab/dic/neologd && \
rm -rf /tmp/mecab-ipadic-neologd
RUN pip install --upgrade pip
RUN pip install mecab-python==0.996

```

To use this image on your Cloudera Data Science Workbench project, perform the following steps.

1. Build a new image with the Dockerfile.

```
docker build --network=host -t <company-registry>/user/cdsw-mecab:latest . -f Dockerfile
```

2. Push the image to your company's Docker registry.

```
docker push <your-company-registry>/user/cdsw-mecab:latest
```

3. Whitelist the image, `<your-company-registry>/user/cdsw-mecab:latest`. Only a site administrator can do this.

Go to **Admin > Engines** and add `<company-registry>/user/cdsw-mecab:latest` to the list of whitelisted engine images.

Engine Images			
Description	Repository:Tag	Default	Actions
Test LaTeX deps	latextest:latest	<input type="radio"/>	Edit Deprecate
Base Image v1	docker.repository.cloudera.com/cdsw/engine:1	<input type="radio"/>	Edit Deprecate
Base Image v2	docker.repository.cloudera.com/cdsw/engine:2	<input type="radio"/>	Edit Deprecate
Base Image v3	docker.repository.cloudera.com/cdsw/engine:3	<input type="radio"/>	Edit Deprecate
Base Image v4	docker.repository.cloudera.com/cdsw/engine:4	<input checked="" type="radio"/>	Edit Deprecate
Custom Image - MeCab	<input type="text" value="<your-company-registry>/user/cdsw-mecab:latest"/>	<input type="radio"/>	Add

4. Ask a project administrator to set the new image as the default for your project. Go to the project **Settings**, click **Engines**, and select `company-registry/user/cdsw-mecab:latest` from the dropdown.

Engine Image

Select the Docker image that Cloudera Data Science Workbench should use to run sessions and jobs in this project. If you'd like to use a different image, contact your site administrator.

- ✓ Test LaTeX deps, latextest:latest
- Base Image v1, docker.repository.cloudera.com/cdsw/engine:1
- Base Image v2, docker.repository.cloudera.com/cdsw/engine:2
- Base Image v3, docker.repository.cloudera.com/cdsw/engine:3
- Base Image v4, docker.repository.cloudera.com/cdsw/engine:4
- Custom Image - MeCab, <your-company-registry>/user/cdsw-mecab:latest**

You should now be able to run this project on the customized MeCab engine.

Limitations

- Cloudera Data Science Workbench only supports customized engines that are based on the Cloudera Data Science Workbench base image.
- Cloudera Data Science Workbench does not support creation of custom engines larger than 10 GB.

Cloudera Bug: DSE-4420

- Cloudera Data Science Workbench does not support pulling images from registries that require Docker credentials.

Cloudera Bug: DSE-1521

- The contents of certain pre-existing standard directories such as `/home/cdsw`, `/tmp`, `/opt/cloudera`, and so on, cannot be modified while creating customized engines. This means any files saved in these directories will not be accessible from sessions that are running on customized engines.

Workaround: Create a new custom directory in the Dockerfile used to create the customized engine, and save your files to that directory. *Or*, create a new custom directory on all the Cloudera Data Science Workbench gateway hosts and save your files to those directories. Then, [mount this directory](#) to the custom engine.

Related Resources

- This Cloudera Engineering Blog post on [Customizing Docker Images in Cloudera Data Science Workbench](#) describes an end-to-end example on how to build and publish a customized Docker image and use it as an engine in Cloudera Data Science Workbench.
- For an example of how to extend the base engine image to include Conda, see [Creating an Extensible Engine With Conda](#) on page 220.

Cloudera Data Science Workbench Engine Versions and Packaging

This topic lists the packages included in the Python and R kernels of the base engine images that ship with each release of Cloudera Data Science Workbench.

Base Engine 10

This section lists the Python, R, and Scala libraries that ship with engine 10.

Python Libraries in Base Engine 10

Engine 10 ships **Python 2.7.17 and 3.6.9**. This section lists the libraries that ship with the Python kernels in engine 10.

Items in **bold** indicate a new version since the last release.

Python3

Library	Version
ipython	5.1.0
requests	2.22.0
Flask	0.12.0
simplejson	3.16.0
numpy	1.17.2
pandas	0.25.1
pandas-datareader	0.8.1
py4j	0.10.8.1
futures	2.1.4
matplotlib	2.0.0
seaborn	0.9.0
Cython	0.29.13
kudu-python	1.2.0

Python2

Library	Version
ipython	5.1.0
requests	2.22.0
Flask	0.12.0
simplejson	3.16.0
numpy	1.16.5
pandas	0.24.2
pandas-datareader	0.8.0
py4j	0.10.8.1
futures	3.3.0
matplotlib	2.0.0
seaborn	0.9.0
Cython	0.29.13
kudu-python	1.2.0

R Libraries in Base Engine 10

Engine 10 ships **R version 3.5.1**. This section lists the libraries that ship with the R kernel in engine 8.

Items in **bold** indicate a new version since the last release.

Package	Version
RCurl	1.95.4.12
caTools	1.17.1.3
svTools	0.9.5
png	0.1.7
RJSONIO	1.3.1.3
ggplot2	3.2.1
cluster	2.1.0
codetools	0.2.16
foreign	0.8.73
dplyr	0.8.3
httr	1.4.1
httpuv	1.5.2
jsonlite	1.6
magrittr	1.5
knitr	1.26
purrr	0.3.3
tm	0.7.7

Package	Version
proxy	0.4.23
data.table	1.12.8
stringr	1.4.0
Rook	1.1.1
rJava	0.9.11
devtools	2.2.1

Scala in Base Engine 10

The Scala kernel is based on the [Apache Toree 0.1.x kernel](#). For details on how to add more dependencies to the kernel, see [Example: Using External Packages by Adding Jars or Dependencies](#) on page 247.

Base Engine 8

This section lists the Python, R, and Scala libraries that ship with engine 8.

Python Libraries in Base Engine 8

Engine 8 ships **Python 2.7.11 and 3.6.8**. This section lists the libraries that ship with the Python kernels in engine 8.

Items in **bold** indicate a new version since the last release.

Library	Version
ipython	5.1.0
requests	2.13.0
Flask	0.12.0
simplejson	3.10.0
numpy	1.13.3
pandas	0.20.1
pandas-datareader	0.2.1
py4j	0.10.7
futures	2.1.4
matplotlib	2.0.0
seaborn	0.8.0
Cython	0.25.2
kudu-python	1.2.0

R Libraries in Base Engine 8

Engine 8 ships **R version 3.5.1**. This section lists the libraries that ship with the R kernel in engine 8.

Items in **bold** indicate a new version since the last release.

Package	Version
RCurl	1.95.4.12
caTools	1.17.1.2

Package	Version
svTools	0.9.5
png	0.1.7
RJSONIO	1.3.1.2
ggplot2	3.1.1
cluster	2.0.9
codetools	0.2.16
foreign	0.8.71
dplyr	0.8.1
httr	1.4.0
httpuv	1.5.1
jsonlite	1.6
magrittr	1.5
knitr	1.23
purrr	0.3.2
tm	0.7.6
proxy	0.4.23
data.table	1.12.2
stringr	1.4.0
Rook	1.1.1
rJava	0.9.11
devtools	2.0.2

Scala in Base Engine 8

The Scala kernel is based on the [Apache Toree 0.1.x kernel](#). For details on how to add more dependencies to the kernel, see [Example: Using External Packages by Adding Jars or Dependencies](#) on page 247.

Base Engine 7

This section lists the Python, R, and Scala libraries that ship with engine 7.

Python Libraries in Base Engine 7

Engine 7 ships **Python 2.7.11 and 3.6.1**. This section lists the libraries that ship with the Python kernels in engine 7.

Items in **bold** indicate a new version since the last release.

Library	Version
ipython	5.1.0
requests	2.13.0
Flask	0.12.0
simplejson	3.10.0
numpy	1.12.1

Library	Version
pandas	0.20.1
pandas-datareader	0.2.1
py4j	0.10.7
futures	2.1.4
matplotlib	2.0.0
seaborn	0.8.0
Cython	0.25.2
kudu-python	1.2.0

R Libraries in Base Engine 7

Engine 7 ships **R version 3.5.1**. This section lists the libraries that ship with the R kernel in engine 7.

Items in **bold** indicate a new version since the last release.

Package	Version
RCurl	1.95.4.11
caTools	1.17.1.1
svTools	0.9.5
png	0.1.7
RJSONIO	1.3.1.1
ggplot2	3.1.0
cluster	2.0.7.1
codetools	0.2.16
foreign	0.8.71
dplyr	0.7.8
httr	1.4.0
httpuv	1.4.5.1
jsonlite	1.6
magrittr	1.5
knitr	1.21
purrr	0.2.5
tm	0.7.6
proxy	0.4.22
data.table	1.12.0
stringr	1.3.1
Rook	1.1.1
rJava	0.9.10
devtools	2.0.1

Scala in Base Engine 7

The Scala kernel is based on the [Apache Toret 0.1.x kernel](#). For details on how to add more dependencies to the kernel, see [Example: Using External Packages by Adding Jars or Dependencies](#) on page 247.

Base Engine 6

This section lists the Python, R, and Scala libraries that ship with engine 6.

Python Libraries in Base Engine 6

Engine 6 ships **Python 2.7.11 and 3.6.1**. This section lists the libraries that ship with the Python kernels in engine 6.

Items in **bold** indicate a new version since the last release.

Library	Version
ipython	5.1.0
requests	2.13.0
Flask	0.12.0
simplejson	3.10.0
numpy	1.12.1
pandas	0.20.1
pandas-datareader	0.2.1
py4j	0.10.7
futures	2.1.4
matplotlib	2.0.0
seaborn	0.8.0
Cython	0.25.2
kudu-python	1.2.0

R Libraries in Base Engine 6

Engine 6 ships **R version 3.4.1**. This section lists the libraries that ship with the R kernel in engine 6.

Items in **bold** indicate a new version since the last release.

Package	Version
RCurl	1.95.4.10
caTools	1.17.1
svTools	0.9.5
png	0.1.7
RJSONIO	1.3.0
ggplot2	3.0.0
cluster	2.0.7.1
codetools	0.2.15
foreign	0.8.70
dplyr	0.7.6

Package	Version
httr	1.3.1
httpuv	1.4.4.2
jsonlite	1.5
magrittr	1.5
knitr	1.20
purrr	0.2.5
tm	0.7.4
proxy	0.4.22
data.table	1.11.4
stringr	1.3.1
Rook	1.1.1
rJava	0.9.10
devtools	1.13.6

Scala in Base Engine 6

The Scala kernel is based on the [Apache Toree 0.1.x kernel](#). For details on how to add more dependencies to the kernel, see [Example: Using External Packages by Adding Jars or Dependencies](#) on page 247.

Base Engine 5

This section lists the Python, R, and Scala libraries that ship with engine 5.

Python Libraries in Base Engine 5

Engine 5 ships **Python 2.7.11 and 3.6.1**. This section lists the libraries that ship with the Python kernels in engine 5.

Items in **bold** indicate a new version since the last release.

Library	Version
ipython	5.1.0
requests	2.13.0
Flask	0.12.0
simplejson	3.10.0
numpy	1.12.1
pandas	0.20.1
pandas-datareader	0.2.1
py4j	0.10.7
futures	2.1.4
matplotlib	2.0.0
seaborn	0.8.0
Cython	0.25.2
kudu-python	1.2.0

R Libraries in Base Engine 5

Engine 5 ships **R version 3.4.1**. This section lists the libraries that ship with the R kernel in engine 5.

Items in **bold** indicate a new version since the last release.

Package	Version
RCurl	1.95.4.10
caTools	1.17.1
svTools	0.9.4
png	0.1.7
RJSONIO	1.3.0
ggplot2	2.2.1
cluster	2.0.7.1
codetools	0.2.15
foreign	0.8.70
dplyr	0.7.5
httr	1.3.1
httpuv	1.4.3
jsonlite	1.5
magrittr	1.5
knitr	1.20
purrr	0.2.5
tm	0.7.3
proxy	0.4.22
data.table	1.11.4
stringr	1.3.1
Rook	1.1.1
rJava	0.9.10
devtools	1.13.5

Scala in Base Engine 5

The Scala kernel is based on the [Apache Toree 0.1.x kernel](#). For details on how to add more dependencies to the kernel, see [Example: Using External Packages by Adding Jars or Dependencies](#) on page 247.

Base Engine 4

This section lists the Python, R, and Scala libraries that ship with engine 4.

Python Libraries in Base Engine 4

Engine 4 ships **Python 2.7.11 and 3.6.1**. This section lists the libraries that ship with the Python kernels in engine 4.

Library	Version
ipython	5.1.0

Library	Version
requests	2.13.0
Flask	0.12.0
simplejson	3.10.0
numpy	1.12.1
pandas	0.20.1
pandas-datareader	0.2.1
py4j	0.10.4
futures	2.1.4
matplotlib	2.0.0
seaborn	0.8.0
Cython	0.25.2
kudu-python	1.2.0

R Libraries in Base Engine 4

Engine 4 ships **R version 3.4.1**. This section lists the libraries that ship with the R kernel in engine 4.

Package	Version
RCurl	1.95.4.10
caTools	1.17.1
svTools	0.9.4
png	0.1.7
RJSONIO	1.3.0
ggplot2	2.2.1
cluster	2.0.6
codetools	0.2.15
foreign	0.8.69
dplyr	0.7.4
httr	1.3.1
httpuv	1.3.5
jsonlite	1.5
magrittr	1.5
knitr	1.18
purrr	0.2.4
tm	0.7.3
proxy	0.4.21
data.table	1.10.4.3
stringr	1.2.0

Package	Version
Rook	1.1.1
rJava	0.9.9
devtools	1.13.4

Scala in Base Engine 4

The Scala kernel is based on the [Apache Toree 0.1.x kernel](#). For details on how to add more dependencies to the kernel, see [Example: Using External Packages by Adding Jars or Dependencies](#) on page 247.

Base Engine 3

This section lists the Python, R, and Scala libraries that ship with engine 3.

Python Libraries in Base Engine 3

Engine 3 ships **Python 2.7.11 and 3.6.1**. This section lists the libraries that ship with the Python kernels in engine 3.

Library	Version
ipython	5.1.0
requests	2.13.0
Flask	0.12.0
simplejson	3.10.0
numpy	1.12.1
pandas	0.20.1
pandas-datareader	0.2.1
py4j	0.10.4
futures	2.1.4
matplotlib	2.0.0
seaborn	0.8.0
Cython	0.25.2
kudu-python	1.2.0

R Libraries in Base Engine 3

Engine 3 ships **R version 3.4.1**. This section lists the libraries that ship with the R kernel in engine 3.

Package	Version
RCurl	1.95.4.8
caTools	1.17.1
svTools	0.9.4
png	0.1.7
RJSONIO	1.3.0
ggplot2	2.2.1
cluster	2.0.6

Package	Version
codetools	0.2.15
foreign	0.8.69
dplyr	0.7.4
httr	1.3.1
httpuv	1.3.5
jsonlite	1.5
magrittr	1.5
knitr	1.17
purrr	0.2.4
tm	0.7.3
proxy	0.4.20
data.table	1.10.4.3
stringr	1.2.0
Rook	1.1.1
rJava	0.9.9
devtools	1.13.4

Scala in Base Engine 3

The Scala kernel is based on the [Apache Toree 0.1.x kernel](#). For details on how to add more dependencies to the kernel, see [Example: Using External Packages by Adding Jars or Dependencies](#) on page 247.

Base Engine 2

This section lists the Python, R, and Scala libraries that ship with engine 2.

Python 2 Libraries in Base Engine 2

Engine 2 ships **Python 2.7.11 and 3.6.1**. This section lists the libraries that ship with the Python kernels in engine 2.

Library	Version
ipython	5.1.0
requests	2.13.0
Flask	0.12.0
simplejson	3.10.0
numpy	1.12.1
pandas	0.20.1
pandas-datareader	0.2.1
py4j	0.10.4
futures	2.1.4
matplotlib	2.0.0
seaborn	0.7.1

Library	Version
Cython	0.23.4

R Libraries in Base Engine 2

Engine 2 ships **R version 3.3.0**. This section lists the libraries that ship with the R kernel in engine 2.

Package	Version
RCurl	1.95.4.8
caTools	1.17.1
svTools	0.9.4
png	0.1.7
RJSONIO	1.3.0
ggplot2	2.2.1
cluster	2.0.6
codetools	0.2.15
foreign	0.8.69
dplyr	0.7.2
httr	1.2.1
httpuv	1.3.5
jsonlite	1.5
magrittr	1.5
knitr	1.16
purrr	0.2.3
data.table	1.10.4
stringr	1.2.0
Rook	1.1.1
rJava	0.9.8
devtools	1.13.3

Scala in Base Engine 2

The Scala kernel is based on the [Apache Toret 0.1.x kernel](#). For details on how to add more dependencies to the kernel, see [Example: Using External Packages by Adding Jars or Dependencies](#) on page 247.

CUDA Engine - Technical Preview

To make it easier for users to get started with using GPUs on CDSW, the Cloudera engineering team is working on a new custom engine that comes enabled with CUDA out of the box. Previously, users were expected to [build their own CUDA engine](#).



Important: This engine is still under development and is not recommended for use in production environments.

Compatibility Information

The first version of this engine is built on top of CDSW base engine:10 and ships with CUDA 10.1. It has been tested with CDSW 1.7.x.

Custom Engine	CDSW	NVIDIA Driver
cuda-engine:10 - Based on CDSW engine:10 - Ships CUDA 10.1	1.7.x	418.39 (or higher) NVIDIA/CUDA compatibility matrix

Enabling GPUs for CDSW with the CUDA engine

This section gives you a modified set of steps to be used when you want to use the CUDA engine to enable GPUs for CDSW workloads. If you already have

Set up the CDSW hosts

The first few steps of this process are the same as that required for the traditional GPU set up process. **If you have already performed these steps for an existing project that uses GPUs, you can move on to the next section: [Site Admins: Add the Custom CUDA Engine to your Cloudera Data Science Workbench Deployment](#) on page 141**

1. [Set Up the Operating System and Kernel](#) on page 138
2. [Install the NVIDIA Driver on GPU Hosts](#) on page 138 - Make sure you are using a driver that is compatible with the CUDA engine.
3. [Enable GPU Support in Cloudera Data Science Workbench](#) on page 139

Site Admins: Add the CUDA Engine to your Cloudera Data Science Workbench Deployment

Required CDSW Role: [Site Administrator](#)

After you've created the CUDA engine, a site administrator must add this new engine to Cloudera Data Science Workbench.

1. Sign in to Cloudera Data Science Workbench.
2. Click **Admin**.
3. Go to the **Engines** tab.
4. Under **Engine Images**, add `docker.repository.cloudera.com/cds/cuda-engine:10` to the list of images.
5. Click **Update**.

Airgapped CDSW Deployments: Once these steps have been performed, the CUDA engine will be pulled from Cloudera's public Docker registry. If you have an airgapped deployment, you will also need to manually distribute this image to every CDSW host. For sample steps, see [Distribute the Image](#) on page 221.

Project Admins: Enable the CUDA Engine for your Project

Project administrators can use the following steps to make it the CUDA engine the default engine used for workloads within a particular project.

1. Navigate to your project's **Overview** page.
2. Click **Settings**.
3. Go to the **Engines** tab.
4. Under **Engine Image**, select the CUDA-capable engine image from the dropdown.

Test the CUDA Engine

You can use the following simple examples to test whether the new CUDA engine is able to leverage GPUs as expected.

1. Go to a project that is using the CUDA engine and click **Open Workbench**.
2. Launch a new session with GPUs.

3. Run the following command in the workbench command prompt to verify that the driver was installed correctly:

```
! /usr/bin/nvidia-smi
```

4. Use any of the following code samples to confirm that the new engine works with common deep learning libraries.

Pytorch

```
!pip3 install torch
from torch import cuda
assert cuda.is_available()
assert cuda.device_count() > 0
print(cuda.get_device_name(cuda.current_device()))
```

Tensorflow

```
!pip3 install tensorflow-gpu==2.1.0
from tensorflow.python.client import device_lib
assert 'GPU' in str(device_lib.list_local_devices())
device_lib.list_local_devices()
```

Keras

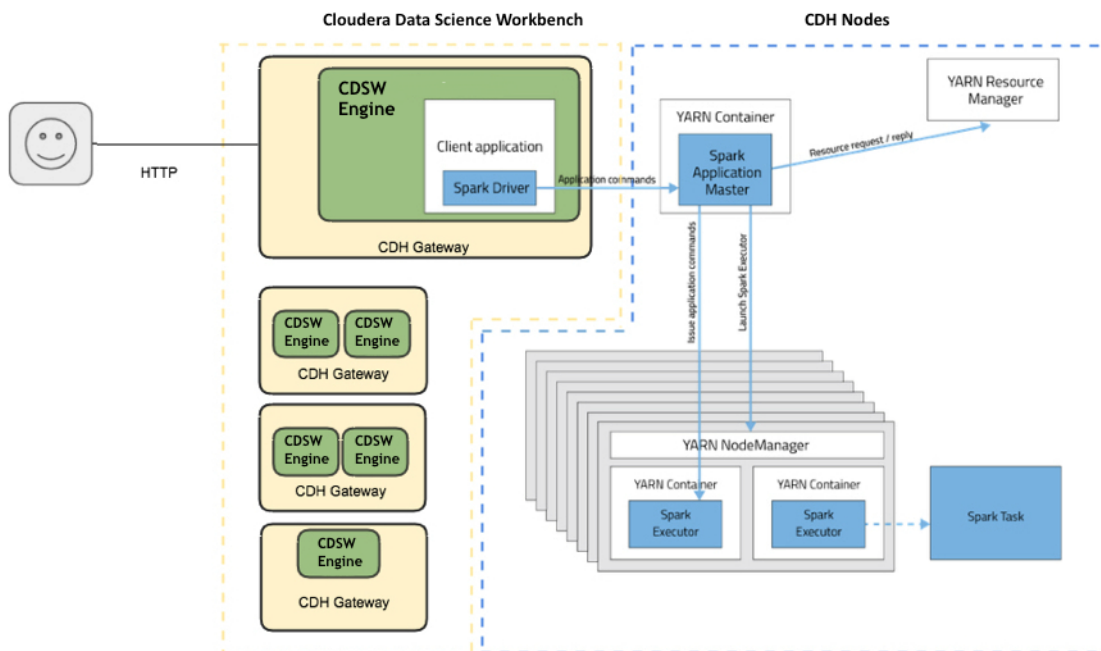
```
!pip3 install keras
from keras import backend
assert len(backend.tensorflow_backend._get_available_gpus()) > 0
print(backend.tensorflow_backend._get_available_gpus())
```

Using CDS 2.x Powered by Apache Spark

Apache Spark is a general purpose framework for distributed computing that offers high performance for both batch and stream processing. It exposes APIs for Java, Python, R, and Scala, as well as an interactive shell for you to run jobs.

Cloudera Data Science Workbench provides interactive and batch access to Spark 2. Connections are fully secure without additional configuration, with each user accessing Spark using their Kerberos principal. With a few extra lines of code, you can do anything in Cloudera Data Science Workbench that you might do in the Spark shell, as well as leverage all the benefits of the workbench. Your Spark applications will run in an isolated project workspace.

Cloudera Data Science Workbench's interactive mode allows you to launch a Spark application and work iteratively in R, Python, or Scala, rather than the standard workflow of launching an application and waiting for it to complete to view the results. Because of its interactive nature, Cloudera Data Science Workbench works with Spark on YARN's `client` mode, where the driver persists through the lifetime of the job and runs executors with full access to the CDH cluster resources. This architecture is illustrated the following figure:



The rest of this guide describes how to set Spark 2 environment variables, manage package dependencies, and how to configure logging. It also consists of instructions and sample code for running R, Scala, and Python projects from Spark 2.

Cloudera Data Science Workbench allows you to access the Spark History Server and even transient per-session UIs for Spark 2 directly from the workbench console. For more details, see [Accessing Web User Interfaces from Cloudera Data Science Workbench](#) on page 145.

Configuring CDS 2.x Powered by Apache Spark 2

This topic describes how to set Spark 2 environment variables, manage package dependencies for Spark 2 jobs, and how to configure logging.

Spark Configuration Files

Cloudera Data Science Workbench supports configuring Spark 2 properties on a per project basis with the `spark-defaults.conf` file. If there is a file called `spark-defaults.conf` in your project root, this will be automatically be added to the global Spark defaults. To specify an alternate file location, set the environmental variable, `SPARK_CONFIG`, to the path of the file relative to your project. If you're accustomed to submitting a Spark job with key-values pairs following a `--conf` flag, these can also be set in a `spark-defaults.conf` file instead. For a list of valid key-value pairs, refer the Spark [configuration reference documentation](#).

Administrators can set environment variable paths in the `/etc/spark2/conf/spark-env.sh` file.

You can also use Cloudera Manager to configure `spark-defaults.conf` and `spark-env.sh` globally for all Spark applications as follows.



Important: Cloudera Data Science Workbench does not automatically detect configuration changes on the CDH cluster. Therefore, any changes made to CDH services must be followed by a full reset of Cloudera Data Science Workbench. Review the associated known issue here: [CDH Integration Issues](#).

Configuring Global Properties Using Cloudera Manager

Configure client configuration properties for all Spark applications in `spark-defaults.conf` as follows:

1. Go to the Cloudera Manager Admin Console.
2. Navigate to the Spark service.
3. Click the **Configuration** tab.
4. Search for the **Spark Client Advanced Configuration Snippet (Safety Valve) for spark-conf/spark-defaults.conf** property.
5. Specify properties described in [Application Properties](#). If more than one role group applies to this configuration, edit the value for the appropriate role group.
6. Click **Save Changes** to commit the changes.
7. Deploy the client configuration.
8. [Restart Cloudera Data Science Workbench](#).

For more information on using a `spark-defaults.conf` file for Spark jobs, visit the [Apache Spark 2 reference documentation](#).

Configuring Spark Environment Variables Using Cloudera Manager

Configure service-wide environment variables for all Spark applications in `spark-env.sh` as follows:

1. Go to the Cloudera Manager Admin Console.
2. Navigate to the Spark 2 service.
3. Click the **Configuration** tab.
4. Search for the **Spark Service Advanced Configuration Snippet (Safety Valve) for spark-conf/spark-env.sh** property and add the paths for the environment variables you want to configure.
5. Click **Save Changes** to commit the changes.
6. Restart the service.
7. Deploy the client configuration.
8. [Restart Cloudera Data Science Workbench](#).

Managing Memory Available for Spark Drivers

By default, the amount of memory allocated to Spark driver processes is set to a 0.8 fraction of the total memory allocated for the engine container. If you want to allocate more or less memory to the Spark driver process, you can override this default by setting the `spark.driver.memory` property in `spark-defaults.conf` (as described above).

Managing Dependencies for Spark 2 Jobs

As with any Spark job, you can add external packages to the executor on startup. To add external dependencies to Spark jobs, specify the libraries you want added by using the appropriate configuration parameter in a `spark-defaults.conf` file. The following table lists the most commonly used configuration parameters for adding dependencies and how they can be used:

Property	Description
<code>spark.files</code>	Comma-separated list of files to be placed in the working directory of each Spark executor.
<code>spark.submit.pyFiles</code>	Comma-separated list of <code>.zip</code> , <code>.egg</code> , or <code>.py</code> files to place on <code>PYTHONPATH</code> for Python applications.
<code>spark.jars</code>	Comma-separated list of local jars to include on the Spark driver and Spark executor classpaths.
<code>spark.jars.packages</code>	Comma-separated list of Maven coordinates of jars to include on the Spark driver and Spark executor classpaths. When configured, Spark will search the local Maven repo, and then Maven central and any additional remote repositories configured by <code>spark.jars.ivy</code> . The format for the coordinates are <code>groupId:artifactId:version</code> .
<code>spark.jars.ivy</code>	Comma-separated list of additional remote repositories to search for the coordinates given with <code>spark.jars.packages</code> .

Example `spark-defaults.conf`

Here is a sample `spark-defaults.conf` file that uses some of the Spark configuration parameters discussed in the previous section to add external packages on startup.

```
spark.jars.packages org.scala-lang:scala-lang-http_2.11:2.3.0
spark.jars my_sample.jar
spark.files data/test_data_1.csv,data/test_data_2.csv
```

spark.jars.packages

The `scala-lang` package will be downloaded from Maven central and included on the Spark driver and executor classpaths.

spark.jars

The pre-existing jar, `my_sample.jar`, residing in the root of this project will be included on the Spark driver and executor classpaths.

spark.files

The two sample data sets, `test_data_1.csv` and `test_data_2.csv`, from the `/data` directory of this project will be distributed to the working directory of each Spark executor.

For more advanced configuration options, visit the [Apache Spark 2 reference documentation](#).

Spark Logging Configuration

Cloudera Data Science Workbench allows you to update Spark's internal logging configuration on a per-project basis. Spark 2 uses Apache Log4j, which can be configured through a properties file. By default, a `log4j.properties` file found in the root of your project will be appended to the existing Spark logging properties for every session and job. To specify a custom location, set the environmental variable `LOG4J_CONFIG` to the file location relative to your project.

The [Log4j documentation](#) has more details on logging options.

Increasing the log level or pushing logs to an alternate location for troublesome jobs can be very helpful for debugging. For example, this is a `log4j.properties` file in the root of a project that sets the logging level to INFO for Spark jobs.

```
shell.log.level=INFO
```

PySpark logging levels should be set as follows:

```
log4j.logger.org.apache.spark.api.python.PythonGatewayServer=<LOG_LEVEL>
```

And Scala logging levels should be set as:

```
log4j.logger.org.apache.spark.repl.Main=<LOG_LEVEL>
```

Running Spark Jobs on an HDP Cluster

Execution errors might occur due to inadequate resource threshold values set for YARN.

In order to run Spark jobs on an HDP cluster, apply the following changes:

1. From Ambari, go to **Views > YARN Queue Manager** service.
2. Choose or add a queue. You can configure root, default, or individual queues.
3. Under the **Resources** section, increase the value in the **Maximum AM Resource** field so that the queues have enough resources to execute. For example, you can set it to 80%.

Child queues can inherit the settings from the parent queue.

Setting Up an HTTP Proxy for Spark 2

In Cloudera Data Science Workbench clusters that use an HTTP proxy, follow these steps to support web-related actions in Spark. You must set the Spark configuration parameter `extraJavaOptions` on your gateway hosts.

To set up a Spark proxy:

1. Log in to Cloudera Manager.
2. Go to **Spark2 > Configuration**.
3. Filter the properties with **Scope > Gateway** and **Category > Advanced**.
4. Scroll down to **Spark 2 Client Advanced Configuration Snippet (Safety Valve) for spark2-conf/spark-defaults.conf**.
5. Enter the following configuration code, substituting your proxy host and port values:

```
spark.driver.extraJavaOptions= \
-Dhttp.proxyHost=<YOUR HTTP PROXY HOST> \
-Dhttp.proxyPort=<HTTP PORT> \
-Dhttps.proxyHost=<YOUR HTTPS PROXY HOST> \
-Dhttps.proxyPort=<HTTPS PORT>
```

6. Click **Save Changes**.
7. Choose **Actions > Deploy Client Configuration**.

Using Spark 2 from Python

Cloudera Data Science Workbench supports using Spark 2 from Python via PySpark.

Setting Up a PySpark Project

PySpark Environment Variables

The default Cloudera Data Science Workbench engine currently includes **Python 2.7.17** and **Python 3.6.9**. CDSW supports what comes bundled with the base image. To use PySpark with lambda functions that run within the CDH cluster, the Spark executors must have access to a matching version of Python. For many common operating systems, the default system Python will not match the minor release of Python included in Data Science Workbench.

To ensure that the Python versions match, Python can either be installed on every CDH host or made available per job run using Spark's ability to distribute dependencies. Given the size of a typical isolated Python environment and the desire to avoid repeated uploads from gateway hosts, Cloudera recommends installing Python 2.7 and 3.6 on the cluster if you are using PySpark with lambda functions.

You can install Python 2.7 and 3.6 on the cluster using any method and set the corresponding `PYSPARK_PYTHON` environment variable in your project. Cloudera Data Science Workbench 1.3 (and higher) include a separate environment variable for Python 3 sessions called `PYSPARK3_PYTHON`. Python 2 sessions continue to use the default `PYSPARK_PYTHON` variable. This will allow you to run Python 2 and Python 3 sessions in parallel without either variable being overridden by the other.

Creating and Running a PySpark Project

To get started quickly, use the **PySpark template project** to create a new project. For instructions, see [Creating a Project](#) on page 126.

To run a PySpark project, navigate to the project's overview page, open the workbench console and launch a Python session. For detailed instructions, see [Using the Workbench](#) on page 130.

Testing a PySpark Project in Spark Local Mode

Spark's local mode is often useful for testing and debugging purposes. Use the following sample code snippet to start a PySpark session in local mode.

```
from pyspark.sql import SparkSession

spark = SparkSession\
    .builder \
    .appName("LocalSparkSession") \
    .master("local") \
    .getOrCreate()
```

For more details, refer the Spark documentation: [Running Spark Applications](#).

Example: Montecarlo Estimation

Within the template PySpark project, `pi.py` is a classic example that calculates Pi using the [Montecarlo Estimation](#).

What follows is the full, annotated code sample that can be saved to the `pi.py` file.

```
## Estimating  $\pi$ 
#
# This PySpark example shows you how to estimate  $\pi$  in parallel
# using Monte Carlo integration.

from __future__ import print_function
import sys
from random import random
from operator import add
# Connect to Spark by creating a Spark session
from pyspark.sql import SparkSession

spark = SparkSession\
    .builder\
    .appName("PythonPi")\
```

```

        .getOrCreate()

partitions = int(sys.argv[1]) if len(sys.argv) > 1 else 2
n = 100000 * partitions

def f(_):
    x = random() * 2 - 1
    y = random() * 2 - 1
    return 1 if x ** 2 + y ** 2 < 1 else 0

# To access the associated SparkContext
count = spark.sparkContext.parallelize(range(1, n + 1), partitions).map(f).reduce(add)
print("Pi is roughly %f" % (4.0 * count / n))

spark.stop()

```

Example: Locating and Adding JARs to Spark 2 Configuration

This example shows how to discover the location of JAR files installed with Spark 2, and add them to the Spark 2 configuration.

```

# # Using Avro data
#
# This example shows how to use a JAR file on the local filesystem on
# Spark on Yarn.

from __future__ import print_function
import os,sys
import os.path
from functools import reduce
from pyspark.sql import SparkSession
from pyspark.files import SparkFiles

# Add the data file to HDFS for consumption by the Spark executors.
!hdfs dfs -put resources/users.avro /tmp

# Find the example JARs provided by the Spark parcel. This parcel
# is available on both the driver, which runs in Cloudera Data Science Workbench, and
# the
# executors, which run on Yarn.
exampleDir = os.path.join(os.environ["SPARK_HOME"], "examples/jars")
exampleJars = [os.path.join(exampleDir, x) for x in os.listdir(exampleDir)]

# Add the Spark JARs to the Spark configuration to make them available for use.
spark = SparkSession\
    .builder\
    .config("spark.jars", ",".join(exampleJars))\
    .appName("AvroKeyInputFormat")\
    .getOrCreate()
sc = spark.sparkContext

# Read the schema.
schema = open("resources/user.avsc").read()
conf = {"avro.schema.input.key": schema }

avro_rdd = sc.newAPIHadoopFile(
    "/tmp/users.avro", # This is an HDFS path!
    "org.apache.avro.mapreduce.HadoopKeyInputFormat",
    "org.apache.avro.mapred.HadoopKey",
    "org.apache.hadoop.io.NullWritable",
    keyConverter="org.apache.spark.examples.pythonconverters.AvroWrapperToJavaConverter",
    conf=conf)
output = avro_rdd.map(lambda x: x[0]).collect()
for k in output:
    print(k)
spark.stop()

```

Example: Distributing Dependencies on a PySpark Cluster

Although Python is a popular choice for data scientists, it is not straightforward to make a Python library available on a distributed PySpark cluster. To determine which dependencies are required on the cluster, you must understand that Spark code applications run in Spark executor processes distributed throughout the cluster. If the Python code you are running uses any third-party libraries, Spark executors require access to those libraries when they run on remote executors.

This example demonstrates a way to run the following Python code (`nltk_sample.py`), that includes pure Python libraries ([nltk](#)), on a distributed PySpark cluster.

1. (Prerequisites)

- Make sure the [Anaconda parcel](#) has been [distributed and activated](#) on your cluster.
- Create a new project in Cloudera Data Science Workbench. In that project, create a new file called `nltk_sample.py` with the following sample script.

`nltk_sample.py`

```
# This code uses NLTK, a Python natural language processing library.
# NLTK is not installed with conda by default.
# You can use 'import' within your Python UDFs, to use Python libraries.
# The goal here is to distribute NLTK with the conda environment.

import os
import sys
from pyspark.sql import SparkSession

spark = SparkSession.builder \
    .appName("spark-nltk") \
    .getOrCreate()

data = spark.sparkContext.textFile('1970-Nixon.txt')

def word_tokenize(x):
    import nltk
    return nltk.word_tokenize(x)

def pos_tag(x):
    import nltk
    return nltk.pos_tag([x])

words = data.flatMap(word_tokenize)
words.saveAsTextFile('nixon_tokens')

pos_word = words.map(pos_tag)
pos_word.saveAsTextFile('nixon_token_pos')
```

2. Go to the project you created and launch a new PySpark session.

3. Click **Terminal Access** and run the following command to pack the Python environment into conda.

```
conda create -n nltk_env --copy -y -q python=2.7.11 nltk numpy
```

The `--copy` option allows you to copy whole dependent packages into certain directory of a conda environment. If you want to add extra pip packages without conda, you should copy packages manually after using `pip install`. In Cloudera Data Science Workbench, pip will install packages into the `~/ .local` directory in a project.

```
pip install some-awesome-package
cp -r ~/ .local/lib ~/ .conda/envs/nltk_env/
```

Zip the conda environment for shipping on PySpark cluster.

```
cd ~/ .conda/envs
zip -r ../../nltk_env.zip nltk_env
```

4. (Specific to NLTK) For this example, you can use NLTK data as input.

```
cd ~/
source activate nltk_env

# download nltk data
(nltk_env)$ python -m nltk.downloader -d nltk_data all
(nltk_env)$ hdfs dfs -put nltk_data/corpora/state_union/1970-Nixon.txt ./

# archive nltk data for distribution
cd ~/nltk_data/tokenizers/
zip -r ../../tokenizers.zip *
cd ~/nltk_data/taggers/
zip -r ../../taggers.zip *
```

5. Set spark-submit options in spark-defaults.conf.

```
spark.yarn.appMasterEnv.PYSPARK_PYTHON=./NLTK/nltk_env/bin/python
spark.yarn.appMasterEnv.NLTK_DATA=./
spark.executorEnv.NLTK_DATA=./
spark.yarn.dist.archives=nltk_env.zip#NLTK,tokenizers.zip#tokenizers,taggers.zip#taggers
```

With these settings, PySpark unzips `nltk_env.zip` into the NLTK directory. `NLTK_DATA` is the environmental variable where NLTK data is stored.

6. In Cloudera Data Science Workbench, set the `PYSPARK_PYTHON` environment variable to the newly-created environment. To do this, navigate back to the Project Overview page and click **Settings > Engine > Environment Variables**. Set `PYSPARK_PYTHON` to `./NLTK/nltk_env/bin/python` and click **Add**. Then click **Save Environment**.
7. Launch a new PySpark session and run the `nltk_sample.py` script in the workbench. You can test whether the script ran successfully using the following command:

```
!hdfs dfs -cat ./nixon_tokens/* | head -n 20
```

```
Annual
Message
to
the
Congress
on
the
State
of
the
Union
.
January
22
'
1970
Mr.
Speaker
'
Mr.

! hdfs dfs -cat nixon_token_pos/* | head -n 20
[(u'Annual', 'JJ')]
[(u'Message', 'NN')]
[(u'to', 'TO')]
[(u'the', 'DT')]
[(u'Congress', 'NNP')]
[(u'on', 'IN')]
[(u'the', 'DT')]
[(u'State', 'NNP')]
[(u'of', 'IN')]
[(u'the', 'DT')]
[(u'Union', 'NN')]
[(u'.', '.')]
[(u'January', 'NNP')]
```

Using CDS 2.x Powered by Apache Spark

```
[(u'22', 'CD')]
[(u',', ',')]
[(u'1970', 'CD')]
[(u'Mr.', 'NNP')]
[(u'Speaker', 'NN')]
[(u',', ',')]
[(u'Mr.', 'NNP')]
```

Using Spark 2 from R

R users can access Spark 2 using [sparklyr](#). Although Cloudera does not ship or support sparklyr, we do recommend using sparklyr as the R interface for Cloudera Data Science Workbench.

Installing sparklyr

Install the latest version of sparklyr as follows.

```
install.packages("sparklyr")
```



Note: The `spark_apply()` function requires the R Runtime environment to be pre-installed on your cluster. This will likely require intervention from your cluster administrator. For details, refer the [RStudio documentation](#).

Connecting to Spark 2

You can connect to local instances of Spark 2 as well as remote clusters.

```
## Connecting to Spark 2
# Connect to an existing Spark 2 cluster in YARN client mode using the spark_connect
function.
library(sparklyr)
system.time(sc <- spark_connect(master = "yarn-client"))
# The returned Spark 2 connection (sc) provides a remote dplyr data source to the Spark
2 cluster.
```

For a complete example, see [Sparklyr \(R\)](#) on page 170.

Using Spark 2 from Scala

This topic describes how to set up a Scala project for CDS 2.x Powered by Apache Spark along with a few associated tasks. Cloudera Data Science Workbench provides an interface to the Spark 2 shell (v 2.0+) that works with Scala 2.11.

Accessing Spark 2 from the Scala Engine

Unlike PySpark or Sparklyr, you can access a SparkContext assigned to the `spark` (SparkSession) and `sc` (SparkContext) objects on console startup, just as when using the Spark shell. By default, the application name will be set to `CDSW_sessionID`, where `sessionID` is the id of the session running your Spark code. To customize this, set the `spark.app.name` property to the desired application name in a `spark-defaults.conf` file.

`Pi.scala` is a classic starting point for calculating Pi using [Montecarlo Estimation](#).

This is the full, annotated code sample.

```
//Calculate pi with Monte Carlo estimation
import scala.math.random

//make a very large unique set of 1 -> n
val partitions = 2
```

```

val n = math.min(100000L * partitions, Int.MaxValue).toInt
val xs = 1 until n

//split up n into the number of partitions we can use
val rdd = sc.parallelize(xs, partitions).setName("'N values rdd'")

//generate a random set of points within a 2x2 square
val sample = rdd.map { i =>
  val x = random * 2 - 1
  val y = random * 2 - 1
  (x, y)
}.setName("'Random points rdd'")

//points w/in the square also w/in the center circle of r=1
val inside = sample.filter { case (x, y) => (x * x + y * y < 1) }.setName("'Random points
inside circle'")
val count = inside.count()

//Area(circle)/Area(square) = inside/n => pi=4*inside/n
println("Pi is roughly " + 4.0 * count / n)

```

Key points to note:

- `import scala.math.random`

Importing included packages works just as in the shell, and need only be done once.

- **Spark context (sc).**

You can access a SparkContext assigned to the variable `sc` on console startup.

```
val rdd = sc.parallelize(xs, partitions).setName("'N values rdd'")
```

Example: Read Files from the Cluster Local Filesystem

Use the following command in the terminal to read text from the local filesystem. The file must exist on all hosts, and the same path for the driver and executors. In this example you are reading the file `ebay-xbox.csv`.

```
sc.textFile("file:///tmp/ebay-xbox.csv")
```

Example: Using External Packages by Adding Jars or Dependencies

External libraries are handled through line magics. Line magics in the Toree kernel are prefixed with `%`.

Adding Remote Packages

You can use Apache Toree's `AddDeps` magic to add dependencies from Maven central. You must specify the company name, artifact ID, and version. To resolve any transitive dependencies, you must explicitly specify the `--transitive` flag.

```

%AddDeps org.scalaj scalaj-http_2.11 2.3.0
import scalaj.http._
val response: HttpResponse[String] = Http("http://www.omdbapi.com/").param("t", "crimson
tide").asString
response.body
response.code
response.headers
response.cookies

```

Using CDS 2.x Powered by Apache Spark

Adding Remote or Local JARs

You can use the `AddJars` magic to distribute local or remote JARs to the kernel and the cluster. Using the `-f` option ignores cached JARs and reloads.

```
%AddJar http://example.com/some_lib.jar -f  
%AddJar file:/path/to/some/lib.jar
```


Cloudera Data Science Workbench Administration Guide

The following topics describe some common administrative tasks that can only be performed by a Cloudera Data Science Workbench [site administrator](#).

Monitoring Cloudera Data Science Workbench Activity

Required Role: [Site Administrator](#)

The **Admin > Overview** tab displays basic information about your deployment, such as the number of users signed up, the number of teams and projects created, memory used, and some average job scheduling and run times. You can also see the version of Cloudera Data Science Workbench you are currently running.

The **Admin > Users** tab displays information about users and their resource use. The tab includes the following columns, among others:

- **CPU Time:** The amount of time, rounded to one hour increments, that a workspace is utilizing a CPU. If the session uses 100% of CPUs available, and there are two CPUs in the instance, then one hour of use results in two hours of CPU time. Time spent idling also counts towards this metric. This metric is tracked as a 30-day moving average.
- **GPU Time:** The same as CPU Time, but with GPUs.
- **Memory Time:** 1 GB of memory allocated to the user's engines per hour. If a 2 GB session is used for one hour, it is counted as two hours of memory time. This metric is also tracked as a 30-day moving average.

The **Admin > Activity** tab of the dashboard displays the following time series charts. These graphs should help site administrators identify basic usage patterns, understand how cluster resources are being utilized over time, and how they are being distributed among teams and users.



Note: Opening the Activity page can cause a brief period of slowness in the rest of the CDSW UI if the CDSW cluster has been in use for a long time. You can avoid or reduce this slow down by purging the cluster's old data. Otherwise, for clusters with many users and projects, use the Activity page sparingly.

The screenshot shows the Cloudera Admin console interface. The top navigation bar includes 'Admin' and 'Usage' tabs. A search bar and a user profile icon are visible. The main content area is titled 'Site Administration' and has several sub-tabs: 'Overview', 'Users', 'Activity' (circled in red), 'Engines', 'Security', 'License', and 'Settings'. Below the tabs, there are filters for 'CPU', 'Memory', 'GPU', 'Runs', and 'Lag', and a 'Date Range' selector set to '2018-02-15 - 2018-02-22'. The main chart is a line graph titled 'Total CPU' showing 'CPU in Use' over time. Below the chart is a table of active jobs:

Name	Creator	Team	Project	Language	CPU	Mem	GPU	Created At	Duration	Status	Actions
Write Weblogs			Engine Packaging	python2	1	2	0	2/22/18 1:18 PM	running	Running	Open Stop
Read Weblogs			Metrics	python2	1	2	0	2/22/18 1:18 PM	running	Running	Open Stop



Important: The graphs and numbers on the **Admin > Activity** page do not account for any resources used by active models on the deployment. For that information, go to **Admin > Models** page.

- **CPU** - Total number of CPUs requested by sessions running at this time.

Note that code running inside an n-CPU session, job, experiment or model replica can access at least n CPUs worth of CPU time. Each user pod can utilize all of its host's CPU resources except the amount requested by other user workloads or Cloudera Data Science Workbench application components. For example, a 1-core Python session can use more than 1 core if other cores have not been requested by other user workloads or CDSW application components.

- **Memory** - Total memory (in GiB) requested by sessions running at this time.
- **GPU** - Total number of GPUs requested by sessions running at this time.
- **Runs** - Total number of sessions and jobs running at this time.
- **Lag** - Depicts session scheduling and startup times.
 - **Scheduling Duration:** The amount of time it took for a session pod to be scheduled on the cluster.
 - **Starting Duration:** The amount of time it took for a session to be ready for user input. This is the amount of time since a pod was scheduled on the cluster until code could be executed.

Related Resources

- **Models** - [Monitoring Active Models](#) on page 193.
- **Tracking Disk Usage** - [Tracking Disk Usage on the Application Block Device](#) on page 258

Monitoring User Events

You can query the PostgreSQL database that is embedded within the Cloudera Data Science Workbench deployment to monitor or audit user events. This requires root access to the Cloudera Data Science Workbench Master host.

1. SSH to the Cloudera Data Science Workbench Master host and log in as `root`.

```
ssh root@<cdsw_master_host_domain_name>
```

2. Get the name of the database pod:

```
kubectl get pods -l role=db
```

The command returns information similar to the following example:

NAME	READY	STATUS	RESTARTS	AGE
db-86bbb69b54-d5q88	1/1	Running	0	4h46m

3. Enter the following command to log into the database as the `sense` user:

```
kubectl exec <database pod> -ti -- psql -U sense
```

For example, the following command logs in to the database on pod `db-86bbb69b54-d5q88`:

```
kubectl exec db-86bbb69b54-d5q88 -ti -- psql -U sense
```

You are logged into the database as the `sense` user.

4. Run queries against the `user_events` table.

For example, run the following query to view the most recent user event:

```
select * from user_events order by created_at DESC LIMIT 1
```

The command returns information similar to the following:

```
id          | 3658
user_id     | 273
ipaddr      | ::ffff:127.0.0.1
user_agent  | node-superagent/2.3.0
event_name  | model created
description | {"model": "Simple Model
1559154287-ex5yn", "modelId": "50", "userType": "NORMAL", "username": "LucyMilton"}
created_at  | 2019-05-29 18:24:47.65449
```

5. (Optional) Export the user events to a CSV file for further analysis:

a) While still logged into the database shell, copy the `user_events` table to a CSV file:

```
copy user_events to '/tmp/user_events.csv' DELIMITER ',' CSV HEADER;
```

b) Exit the PostgreSQL shell. Type `\q` and press ENTER.

c) Find the Docker container that the database runs in:

```
docker ps | grep db-86bbb
```

The command returns output similar to the following:

```
8c56d04bbd58 c230b2f564da "docker-entrypoint..." 7 days ago Up 7 days
k8s_db_db-86bbb69b54-fcfm6_default_8b2dd23d-88b9-11e9-bc34-0245eb679f96_0
```

The first entry in **bold** is the container ID.

d) Copy the `user_events.csv` file out of the container into a temporary directory on the Master host:

```
docker cp <container_ID>:/tmp/user_events.csv /tmp/user_events.csv
```

For example:

```
docker cp 8c56d04bbd58:/tmp/user_events.csv /tmp/user_events.csv
```

e) Use SCP to copy `/tmp/user_events.csv` from the Cloudera Data Science Workbench Master host to a destination of your choice.

For example, run the following command on your local machine to copy `user_events.csv` to a local directory:

```
scp root@<cdsw_master_host_domain_name>:/tmp/user_events.csv /path/to/local/directory/
```

For information about the different user events, see [Tracked User Events](#) on page 251.

Tracked User Events

The tables on this page describe the user events that are logged by Cloudera Data Science Workbench.

Table 4: Database Columns

When you query the `user_events` table, the following information can be returned:

Information	Description
id	The ID assigned to the event.
user_id	The UUID of the user who triggered the event.

Information	Description
ipaddr	The IP address of the user or component that triggered the event. 127.0.0.1 indicates an internal component.
user agent	The user agent for this action, such as the web browser. For example: <div style="border: 1px dashed gray; padding: 5px; margin: 5px 0;"> <pre>Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.103 Safari/537.36</pre> </div>
event_name	The event that was logged. The tables on this page list possible events.
description	This field contains the model name and ID, the user type (NORMAL or ADMIN), and the username.
created_at	The date (YYYY-MM-DD format) and time (24-hour clock) the event occurred .

Table 5: Events Related to Engines

Event	Description
engine environment vars updated	-
engine mount created	-
engine mount deleted	-
engine mount updated	-
engine profile created	-
engine profile deleted	-
engine profile updated	-

Table 6: Events Related to Experiments

Event	Description
experiment run created	-Starting with version 1.6.1, this event also captures resource usage (cpu, memory, gpu) per experiment.
experiment run repeated	-Starting with version 1.6.1, this event also captures resource usage (cpu, memory, gpu) per experiment.
experiment run cancelled	-

Table 7: Events Related to Files

Event	Description
file downloaded	-
file updated	-
file deleted	-
file copied	-
file renamed	-

Event	Description
file linked	The logged event indicates when a symlink is created for a file or directory.
directory uploaded	-

Table 8: Events Related to Models

Event	Description
model created	-Starting with version 1.6.1, this event also captures resource usage (cpu, memory, gpu) per model created.
model deleted	-

Table 9: Events Related to Jobs

Event	Description
job created	-
job started	-Starting with version 1.6.1, this event also captures resource usage (cpu, memory, gpu) per job started.
stopped all runs for job	-
job shared with user	-
job unshared with user	-
job sharing updated	The logged event indicates when the sharing status for a job is changed from one of the following options to another: <ul style="list-style-type: none"> • All anonymous users with the link • All authenticated users with the link • Specific users and teams

Table 10: Events Related to Licenses

Event	Description
license created	-
license deleted	-

Table 11: Events Related to Projects

Event	Description
project created	-
project updated	-
project deleted	-
collaborator added	-
collaborator removed	-
collaborator invited	-

Table 12: Events Related to Sessions

Event	Description
session launched	-Starting with version 1.6.1, this event also captures resource usage (cpu, memory, gpu) per session launched.
session terminated	-
session stopped	-
session shared with user	-
session unshared with user	-
update session sharing status	The logged event indicates when the sharing status for a session is changed from one of the following options to another: <ul style="list-style-type: none"> • All anonymous users with the link • All authenticated users with the link • Specific users and teams

Table 13: Events Related to Admin Settings

site config updated	The logged event indicates when a setting on the Admin Settings page is changed.
---------------------	---

Table 14: Events Related to Teams

Event	Description
add member to team	-
delete team member	-
update team member	-

Table 15: Events Related to Users

Event	Description
forgot password	-
password reset	-
update user	If the logged event shows that a user is banned, that means that the user account has been deactivated and does not count toward the license.
user signup	-
user login	The logged event includes the authorization method, LDAP/SAML or local.
user logout	-
ldap/saml user creation	The logged event indicates when a user is created with LDAP or SAML.

Configuring Quotas - Technical Preview



Important: Quotas is an experimental feature. It is not considered ready for production.

Required Role: Site Administrator

This topic describes how to configure CPU, GPU, and memory quotas for users on a CDSW deployment. You can set default quotas for each user on the deployment as well as overriding custom quotas for specific users who might require more resources. Quotas can be enabled from the Site Administration panel.

Before you begin

By default, the quotas feature is hidden. To access this feature, first go to the CDSW service in Cloudera Manager and use the **Feature flag overrides** property to enable access.

1. Log in to Cloudera Manager.
2. Go to the **CDSW** service.
3. Click **Configuration**.
4. Search for the **Feature flag overrides** property. Add the following JSON code to enable quotas.

```
{"quotas": true}
```

5. Click **Save Changes**.
6. [Restart the CDSW service](#).

You should now be able to see the Quotas tab in the Site Admin panel in Cloudera Data Science Workbench.

Enabling Default Quotas for all CDSW Users

To enable CPU, GPU, and memory quotas for users on your CDSW deployment:

1. Log into Cloudera Data Science Workbench with site administrator privileges.
2. Click **Admin > Quotas**.
3. Switch the toggle to **ON**. This applies a default quota of **2 vCPU, 8 GB memory** to each user on the deployment.

If your deployment was provisioned with GPUs, a default quota of **0 GPU** per user will apply. If you want users to have access to GPUs, you must modify the default quotas as described in the next step.
4. If you want to change the default quotas, click on **Default (per user)**. CDSW displays the **Edit default quota dialog** box.
5. Enter the CPU, Memory, and GPU quota values that should apply to all users of the deployment.
6. Click **Update**.

Results

Enabling quotas will only affect new workloads. If users have already scheduled workloads that exceed the new quota limits, those will continue to run uninterrupted. If a user is over their limit, they will not be able to schedule any more workloads.

Enabling Custom Quotas for Specific Users

This section shows you how to enable custom quotas for specific users on your CDSW deployment. You can use this feature to help certain users who regularly require more resources than the defaults set in the previous section. Any custom values set here will override the default quotas.

To enable custom quotas:

1. Log into Cloudera Data Science Workbench with site administrator privileges.

2. Click **Admin > Quotas**.
3. Under the **Custom Quota** section, click **Add Custom Quota**.
4. Enter the Username, CPU, Memory, and GPU values that apply to the custom quota. Then click **Add**.

You can use a decimal floating point value (x.01) only for the CPU and Memory settings, not for GPUs.

5. Repeat Steps 3 and 4 to add custom quotas as needed.

You can also duplicate an existing custom quota setting for a new user. Click the vertical ellipses at the end of the custom quota row and choose **Duplicate**, specify a new username, and click **Add**.

Results

Enabling custom quotas will only affect new workloads. If users have already scheduled workloads that exceed the new quota limits, those will continue to run uninterrupted. If a user is over their limit, they will not be able to schedule any more workloads.

Modifying Default and Custom Quotas

Editing Quotas

Modifying existing quotas will only affect new workloads. To modify an existing quota setting, click the vertical ellipses at the end of the row and choose **Edit**. Modify the parameters as needed and click **Update** to save your changes. Note that these modifications will only affect new workloads.

Deleting Quotas

To delete a custom quota, click the vertical ellipses at the end of the custom quota row and choose **Remove**.

Managing the Cloudera Data Science Workbench Service in Cloudera Manager

This topic describes how to configure and manage Cloudera Data Science Workbench using Cloudera Manager. The contents of this topic only apply to CSD-based deployments. If you installed Cloudera Data Science Workbench using the RPM, the Cloudera Data Science Workbench service will not be available to you in Cloudera Manager.

Adding the Cloudera Data Science Workbench Service

Cloudera Data Science Workbench is available as an add-on service for Cloudera Manager. To install Cloudera Data Science Workbench, you require the following files: a CSD JAR file that contains all the configuration needed to describe and manage the new Cloudera Data Science Workbench service, and the Cloudera Data Science Workbench parcel.

To install this service, first download and copy the CSD file to the Cloudera Manager Server host. Then use Cloudera Manager to distribute the Cloudera Data Science Workbench parcel to the relevant gateway hosts. You can then use Cloudera Manager's **Add Service** wizard to add the Cloudera Data Science Workbench service to your cluster.

For the complete set of instructions, see [Install Cloudera Data Science Workbench](#) on page 75.

Roles Associated with the Cloudera Data Science Workbench Service

Master

Runs the Kubernetes master components on the CDSW master host.

The Master role must only be assigned to the Cloudera Data Science Workbench master host.

Worker

Runs the Kubernetes worker/host components on the CDSW worker hosts.

The Worker role must be assigned to all Cloudera Data Science Workbench worker hosts. **Do not assign the Master and Worker roles to the same host.** Even if you are running a single-host proof-of-concept deployment, the single Master host will be able to run user workloads just as a worker host can.

Docker Daemon

Runs underlying Docker processes on *all* Cloudera Data Science Workbench hosts.

The Docker Daemon role must be assigned to every Cloudera Data Science Workbench gateway host.

Application

Runs the Cloudera Data Science Workbench web application. The Application role must only be assigned to the Cloudera Data Science Workbench master host.

As of version 1.6, the Application role can be restarted independently of the other roles. However, the Master role must not be restarted independently of the Application role.

Similarly, do not attempt to restart the underlying Docker Daemon role while the Master/Worker roles are still running on a host. This will result in the operation hanging indefinitely. To avoid this, always perform a full service restart.

Accessing Cloudera Data Science Workbench from Cloudera Manager

1. Log into the Cloudera Manager Admin Console.
2. Go to the **CDSW** service.
3. Click **CDSW Web UI** to visit the Cloudera Data Science Workbench web application.

Configuring Cloudera Data Science Workbench Properties

In a CSD-based deployment, Cloudera Manager allows you to configure Cloudera Data Science Workbench properties without having to directly edit any configuration file.

1. Log into the Cloudera Manager Admin Console.
2. Go to the **CDSW** service.
3. Click the **Configuration** tab.
4. Use the search bar to look for the property you want to configure. You can use Cloudera Manager to [configure proxies](#), [enable TLS](#), reserve the master host, and [enable GPU support](#) for Cloudera Data Science Workbench.

If you have recently migrated from an RPM-based deployment to a CSD-based deployment, a list of the properties in `cdsw.conf`, along with their corresponding properties in Cloudera Manager can be found in the upgrade guide [here](#).

5. Click **Save Changes**.

Starting, Stopping, and Restarting the Service



Important: On Cloudera Data Science Workbench 1.4.0 (and lower), **do not** stop or restart Cloudera Data Science Workbench without using the [cdsw_protect_stop_restart.sh](#) script. This is to help avoid the data loss issue detailed in [TSB-346](#).

To start, stop, and restart the Cloudera Data Science Workbench service:

1. Log into the Cloudera Manager Admin Console.
2. On the **Home > Status** tab, click



to the right of the **CDSW** service and select the action (**Start**, **Stop**, or **Restart**) you want to perform from the dropdown.

3. Confirm your choice on the next screen. When you see a **Finished** status, the action is complete.

Points to Remember

- After a restart, the Cloudera Data Science Workbench service in Cloudera Manager will display **Good** health even though the Cloudera Data Science Workbench web application might need a few more minutes to get ready to serve requests.
- The CDSW service must be restarted every time client configuration is redeployed to the Cloudera Data Science Workbench hosts.

Checking the Status of the CDSW Service

Starting with version 1.6, the CDSW service in Cloudera Manager includes the following commands:

- **Status:** Checks the current status of Cloudera Data Science Workbench.
- **Validate:** Runs common diagnostic checks to ensure all internal components are configured and running as expected.

To run these commands on the Cloudera Data Science Workbench service:

1. Log into the Cloudera Manager Admin Console.
2. On the **Home** > **Status** tab, click



to the right of the **CDSW** service and select the action (**Status** or **Validate**) you want to perform from the dropdown.

3. Confirm your choice on the next screen. When you see a **Finished** status, the action is complete. If the commands fail, click on the **stdout** tab to view the complete output from the commands.

Managing Cloudera Data Science Workbench Worker Hosts

You can add or remove workers from Cloudera Data Science Workbench using Cloudera Manager. For instructions, see:

- [Adding a Worker Host](#)
- [Removing a Worker Host](#)

Health Tests

Cloudera Manager runs a few health tests to confirm whether Cloudera Data Science Workbench and its components (Master and Workers) are running, and ready to serve requests.

You can choose to enable or disable individual or summary health tests, and in some cases specify what should be included in the calculation of overall health for the service, role instance, or host. See [Configuring Monitoring Settings](#) for more information.

Tracking Disk Usage on the Application Block Device

This section demonstrates how to use Cloudera Manager to chart disk usage on the Application block device over time, and to create a trigger to notify cluster administrators when free space on the block device falls below a certain threshold. The latter is particularly important because once the Application block device runs out of disk space, Cloudera Data Science Workbench will stop launching any new sessions or jobs. Advance notifications will give administrators a chance to expand the block device or cleanup existing data before Cloudera Data Science Workbench users run into any problems.

Create a Chart to Track Disk Usage on the Application Block Device

The following steps use Cloudera Manager's Chart Builder to track disk usage on the Application Block Device (mounted to `/var/lib/cdsw` on the CDSW master host) over time.

1. Log into the Cloudera Manager Admin Console.
2. Click **Charts** > **Chart Builder**.

- Enter a [tsquery](#) that charts disk usage on the block device. For example, the following tsquery creates a chart to track unallocated disk space on the Application block device.

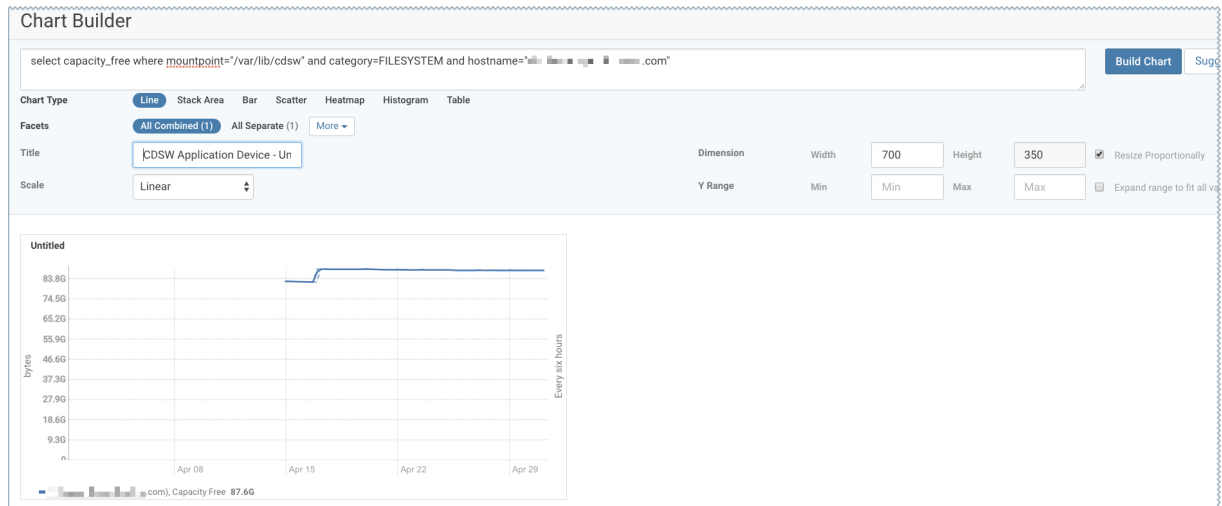
```
select capacity_free where mountpoint="/var/lib/cdsw" and category=FILESYSTEM and hostname="<CDSW_Master_hostname>"
```

Alternatively, you could use the following tsquery to track the disk space already in use on the block device.

```
select capacity, capacity_used where mountpoint="/var/lib/cdsw" and category=FILESYSTEM and hostname="<CDSW_Master_hostname>"
```

Make sure you insert the hostname for your master host as indicated in the queries.

- Click **Build Chart**. You should see a preview of the chart below.



- Click **Save**.
- Enter a name for the chart.
- Select **Add chart to another dashboard**. From the dropdown list of available System Dashboards, select **CDH Cloudera Data Science Workbench Status Page**.
- Click **Save Chart**. If you navigate back to the CDSW service page, you should now see the new chart on this page.

For more details about Cloudera Manager's Chart Builder, see the following topic in the Cloudera Manager documentation: [Charting Time Series Data](#).

Create a Trigger to Notify Cluster Administrators when Free Space Runs Low

The following steps create a [trigger](#) to alert Cloudera Manager cluster administrators when free space on the Application Block Device has fallen below a specific threshold.

- Log in to Cloudera Manager and go to the CDSW service page.
- Click **Create Trigger**.
- Give the trigger a name.
- Modify the Expression field to include a condition for the trigger to fire. For example, if the trigger should fire when unallocated disk space on the Application Block Device falls below 250GB, the expression should be:

```
IF (select capacity_free where mountpoint="/var/lib/cdsw" and category=FILESYSTEM and hostname="<CDSW_Master_hostname>" and LAST (capacity_free) < 250GB) DO health:concerning
```

On the right hand side of the page, you should see a preview of the query you have entered and a chart that displays the result of the query as in the following sample image. Note that if the query is incorrect or incomplete you will not see the preview on the right.

The screenshot displays the 'Create New Trigger' configuration page. It includes a 'Name' field with the value 'CDSW Application Device Memory Low'. The 'Expression' field contains a tsquery: `IF (select capacity_free where mountpoint='/var/lib/cdsw' and category=FILESYSTEM and hostname='...com' and LAST (capacity_free) < 55GB) DO health:concerning`. The 'Preview' section shows a green checkmark and the text 'CDSW Application Device Memory Low trigger is not firing.' Below this, a 'Charts' section displays a line graph for 'capacity_free' in bytes. The y-axis ranges from 18.6G to 74.5G. A red horizontal line indicates a threshold at 55.9G. The x-axis shows time intervals from 12:02 to 12:03. A legend at the bottom of the chart identifies the red line as 'Threshold: capacity_free < 5905... 59.1B'.

5. Click **Create Trigger**. If you navigate back to the CDSW service page, you should now see the new trigger in the list of Health Tests.

For more details about Triggers, refer the following topic in the Cloudera Manager documentation: [Triggers](#).

Creating Diagnostic Bundles

Diagnostic data for Cloudera Data Science Workbench is now available as part of the Cloudera Manager diagnostic bundle. For details on usage and diagnostic data collection in Cloudera Data Science Workbench, see [Data Collection in Cloudera Data Science Workbench](#) on page 260.

Data Collection in Cloudera Data Science Workbench

Cloudera Data Science Workbench collects usage and diagnostic data in two ways, both of which can be controlled by system administrators.

Usage Tracking

Cloudera Data Science Workbench collects aggregate usage data by sending limited tracking events to Google Analytics and Cloudera servers. No customer data or personal information is sent as part of these bundles.

Disable Usage Tracking

1. Log in to Cloudera Data Science Workbench as a site administrator.
2. Click **Admin**.
3. Click the **Settings** tab.
4. Uncheck **Send usage data to Cloudera**.

In addition to this, Cloudera Manager also collects and sends anonymous usage information using Google Analytics to Cloudera. If you are running a CSD-based deployment and want to disable data collection by Cloudera Manager, see [Managing Anonymous Usage Data Collection](#).

Diagnostic Bundles

Diagnostic bundles are used to aid in debugging issues filed with Cloudera Support. They can be created using either Cloudera Manager (only for CSD-based deployments), or the command line. This section also provides details on the information collected by Cloudera Data Science workbench as part of these bundles.

Using Cloudera Manager

If you are working on a CSD-based deployment, Cloudera Data Science Workbench logs and diagnostic data are available as part of the diagnostic bundles created by Cloudera Manager. By default, Cloudera Manager is configured to collect diagnostic data weekly and to send it to Cloudera *automatically*. You can schedule the frequency of data collection on a daily, weekly, or monthly schedule, or even disable the scheduled collection of data. To learn how to configure the frequency of data collection or disable it entirely, see [Diagnostic Data Collection by Cloudera Manager](#).

You can also manually trigger a collection and transfer of diagnostic data to Cloudera at any time. For instructions, see [Manually Collecting and Sending Diagnostic Data to Cloudera](#).



Note: If Cloudera Data Science Workbench data is missing from Cloudera Manager diagnostic bundles, it might be due to [this known issue](#).

You can configure what directory Cloudera Manager uses as a staging directory for diagnostic bundles. Change the **Log Staging Directory** property for your Cloudera Data Science Workbench instance in Cloudera Manager to set a different directory.

Usage Metrics

Starting with version 1.7, CDSW also gathers highly redacted information on which feature is being used. When you create a diagnostic bundle, this information is packed alongside the diagnostic information.

If you want to disable collection of usage metrics, configure the **Feature flag overrides** property in Cloudera Manager as described here:

1. Log in to Cloudera Manager.
2. Go to the **CDSW** service.
3. Click **Configuration**.
4. Search for the **Feature flag overrides** property. Add the following JSON code to disable usage me.

```
{"usage-reporting": false}
```

5. Click **Save Changes**.
6. [Restart the CDSW service](#).

Using the Command Line

Diagnostic bundles can be created by system administrators using the `cdsw logs` command. By default, sensitive information will be redacted from the log files in the bundle. This is the bundle that you should attach to any case opened with Cloudera Support. The filename of this generated bundle will be of the form, `cdsw-logs-$hostname-$date-$time.redacted.tar.gz`.

If you want to turn off redaction of log files, you can use the `-x|--skip-redaction` option as demonstrated below.

```
cdsw logs --skip-redaction
```

The diagnostic bundle is only meant for internal use. It should be retained at least for the duration of the support case, in case any critical information was redacted. However, it can be shared with Cloudera at your discretion. The filename of this bundle will be of the form, `cdsw-logs-$hostname-$date-$time.tar.gz`.

The contents of both archives are stored in text and can easily be inspected by system administrators. Both forms are designed to be easily diff-able.

Usage Metrics

Starting with version 1.7, CDSW also gathers highly redacted information on which feature is being used. When you create a diagnostic bundle, this information is packed alongside the diagnostic information. If you want to turn off

collection of information on feature usage, use the `-u|--skip-usage-events` flag when you generate the diagnostic bundle. For example:

```
cdsw logs --skip-usage-events
```

Information Collected in Diagnostic Bundles

Cloudera Data Science Workbench diagnostic bundles collect the following information:

- System information such as hostnames, operating system, kernel modules and settings, available hardware, and system logs.
- Cloudera Data Science Workbench version, status information, configuration, and the results of install-time validation checks.
- Details about file systems, devices, and mounts in use.
- CDH cluster configuration, including information about Java, Kerberos, installed parcels, and CDH services such as Spark 2.
- Network configuration and status, including interfaces, routing configuration, and reachability.
- Status information for system services such as Docker, Kubernetes, NFS, and NTP.
- Listings for processes, open files, and network sockets.
- Reachability, configuration, and logs for Cloudera Data Science Workbench application components.
- Hashed Cloudera Data Science Workbench user names.
- Information about Cloudera Data Science Workbench workloads (sessions, jobs, experiments, models, applications), including editor, kernel type, engine, ownership, termination status, and performance data.

Cloudera Data Science Workbench Email Notifications

Required Role: [Site Administrator](#)

Go to the **Admin > Settings** tab to specify an email address for outbound invitations and job notifications.

By default, all emails are sent from `noreply@your-cdsw-domain`. However, if your SMTP domain is different from the Cloudera Data Science Workbench domain, or it does not allow spoofing, you will need to explicitly specify the email address at the No Reply Email field.

Cloudera Data Science Workbench sends email notifications when you add collaborators to a project, share a project with a colleague, and for job status updates (email recipients are [configured per-job](#)). Emails are not sent when you create a new project. Email preferences cannot currently be configured at an individual user level.

Managing License Keys for Cloudera Data Science Workbench

Cloudera Data Science Workbench requires a Cloudera Enterprise license. To obtain a Cloudera Enterprise license, either fill in this [form](#), or call 866-843-7207. Note that only one license key can be used at a time.

After an initial trial period of 60 days, you must upload a license key to continue to use Cloudera Data Science Workbench.

Trial License

Cloudera Data Science Workbench is fully functional during a 60-day, non-renewable trial period. The trial period starts when you create your first user.

If 60 days or fewer remain on the license, a badge in the lower left corner of the dashboard displays the number of days remaining. The initial trial period is 60 days, so the remaining days are always displayed during the trial period.

When a trial license expires, functionality is limited as follows:

- A warning banner notifies you that the license has expired and suggests you contact the site administrator or upload a license key.
- You cannot create new users, projects, sessions, or jobs.
- Existing users can log in and view their projects and files.
- You cannot run existing jobs.

At this point you can obtain a Cloudera Enterprise license and upload it to Cloudera Data Science Workbench using the steps described below. Cloudera Data Science Workbench will then go back to being fully functional.

Cloudera Enterprise License

When an Enterprise license expires, a warning banner displays, but all product features remain fully functional.

Contact [Cloudera Support](#) to receive an updated license.

Uploading License Keys

To upload the license key:

1. Go to **Admin > License**.
2. Click **Upload License**.
3. Select the license file to be uploaded and click **Upload**.

User Access to Features

Cloudera Data Science Workbench provides Site Administrators with the ability to restrict or control specific functionality that non-Site Administrator users have access to. This is done through the **Security** and **Settings** tabs on the **Admin** page.

Table 16: Security Tab

Property	Description
Allow remote editing	Disable this property to prevent users from connecting to the Cloudera Data Science Workbench deployment with <code>cdswctl</code> and using local IDEs, such as PyCharm.
Allow only session creators to execute commands on active sessions	By default, a user's permission to active sessions in a project is the same as the user's permission to that project, which is determined by the combination of the user's permission as a project collaborator, the user's permission in the team if this is a team project, and whether the user is a Site Administrator. By checking this checkbox, only the user that created the active session will be able to execute commands in that session. No other users, regardless of their permissions in the team or as project collaborators, will be able to execute commands on active sessions that are not created by them. Even Site Administrators will not be able to execute commands in other users' active sessions.
Allow console output sharing	Disable this property to remove the Share button from the project workspace and workbench UI as well as disable access to all shared console outputs across the deployment. Note that re-enabling this property does not automatically grant access to previously shared consoles. You will need to manually share each console again.

Property	Description
Allow anonymous access to shared console outputs	Disable this property to require users to be logged in to access shared console outputs.
Allow file upload/download through UI	Use this checkbox to show/hide file upload/download UI in the project workspace. When disabled, Cloudera Data Science Workbench API will forbid request of downloading file(s) as attachment. Note that the backend API to upload/edit/read the project files are intact regardless of this change in order to support basic Cloudera Data Science Workbench functionality such as file edit/read.

Table 17: Settings Tab

Property	Description
Allow users to create public projects	Disable this property to restrict users from creating new public projects. Site Administrators will have to create any new public projects.
Allow users to create projects	Disable this property to restrict users from creating new projects. Site Administrators will have to create any new projects.
Allow users to create teams	Disable this property to restrict users from creating new teams. Site Administrators will have to create any new teams.
Allow users to run experiments	Disable this property to hide the Experiments feature in the UI. Note that this property does not affect any active experiments.
Allow users to create models	Disable this property to hide the Models feature in the UI. Note that this property does not affect any active models. In particular, if you do not stop active models before hiding the Models feature, they continue to serve requests and consume computing resources in the background.

Cluster Management

This section describes some common tasks and guideline related to cluster management for Cloudera Data Science Workbench.

For a basic overview of how Cloudera Data Science Workbench fits onto a Cloudera Manager and CDH cluster, refer the [Architecture Overview](#) on page 17.

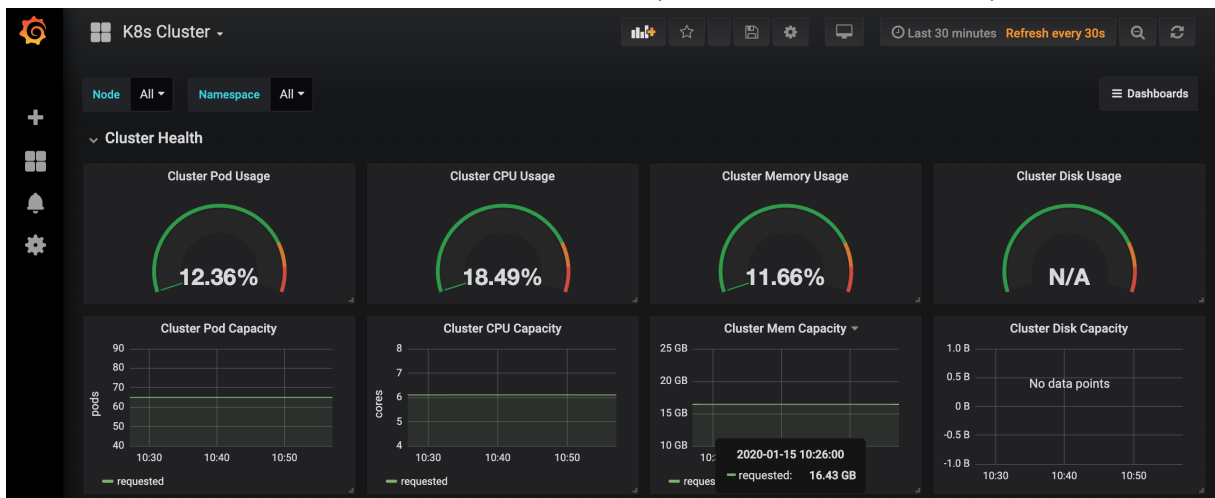
Cluster Monitoring with Grafana

Cloudera Data Science Workbench leverages Prometheus and Grafana to provide a dashboard that allows you to monitor how CPU, memory, storage, and other resources are being consumed by your CDSW deployment. Prometheus is an internal data source that is auto-populated with resource consumption data for each deployment. Grafana is the monitoring dashboard that allows you to create visualizations for resource consumption data from Prometheus.

By default, CDSW provides you with three Grafana dashboards: K8s Cluster, K8s Containers, and K8s Node. You can extend these dashboards or create more panels for other metrics. For more information, see the [Grafana documentation](#).

To access the Grafana dashboard for your deployment:

1. Log into Cloudera Data Science Workbench with site administrator privileges.
2. Click **Admin > Overview**
3. Click the **Grafana dashboard** link. This will take you to the built-in Grafana server.
4. To see all the available dashboards, click **Home > K8 Cluster** (or K8s Containers or K8s Node).



K8s Cluster

Provides metrics for the following:

- Overview of nodes, pods, and containers
- CPU capacity usage
- Memory capacity usage
- Pod capacity usage
- Disk capacity usage

K8s Containers

Provides metrics for the following:

- Memory usage per pod
- CPU Usage per pod

- Read/Write IOPS per pod

K8s Node

Provides metrics for the following:

- CPU usage per node
- Memory Usage per node
- Read/Write IOPS per node
- Available memory per node
- Network traffic per node

Backup and Disaster Recovery for Cloudera Data Science Workbench

All application data for Cloudera Data Science Workbench, including project files and database state, is stored on the master host at `/var/lib/cdsw`. Given typical access patterns, it is strongly recommended that `/var/lib/cdsw` be stored on a dedicated SSD block device or SSD RAID configuration. Because application data is not replicated to HDFS or backed up by default, site administrators must enable a backup strategy to meet any disaster recovery scenarios.

Cloudera strongly recommends both regular backups and backups before upgrades and is not responsible for any data loss.

Creating a Backup

1. Cloudera Data Science Workbench 1.4.2 or lower

Do not stop or restart Cloudera Data Science Workbench without using the [cdsw_protect_stop_restart.sh](#) script. This is to help avoid the data loss issue detailed in [TSB-346](#).

Run the script on your master host and stop Cloudera Data Science Workbench (instructions below) only when instructed to do so by the script. Then proceed with step 2 of this process.

Cloudera Data Science Workbench 1.4.3 or higher

Depending on your deployment, use one of the following sets of instructions to stop the application.

To stop Cloudera Data Science Workbench:

- **CSD** - Log in to Cloudera Manager. On the **Home** > **Status** tab, click



to the right of the **CDSW** service and select **Stop** from the dropdown. Wait for the action to complete.

OR

- **RPM** - Run the following command on the master host:

```
cdsw stop
```

2. To create the backup, run the following command on the *master* host:

```
tar -cvzf cdsw.tar.gz -C /var/lib/cdsw/ .
```



Note: Using tar to create the backup preserves important file metadata such as file ownership. Other methods of copying/saving files might not preserve this information. This metadata is required for tasks such as [migrating CDSW](#) to another cluster.

Restoring from a Backup

You can restore across versions. For example, you can restore a tarball from CDSW 1.5 to, say, version 1.6. To restore from a backup:

1. Stop the Cloudera Data Science Workbench.

1. **CSD** - Log in to Cloudera Manager. On the **Home** > **Status** tab, click to the right of the CDSW service and select **Stop** from the dropdown. Wait for the action to complete. OR

2. **RPM** - Run the following command on the master host:

```
cdsw stop
```

2. After stopping CDSW, and before running the following tar command, wait 2-5 minutes (depending on your disk speed) to ensure that all data from CDSW is successfully written to the disks. Otherwise the tar command may not capture all recent changes.

3. Restore the tarball to the following location: `var/lib/cdsw` by using the following command:

```
tar -xvzf cdsw.tar.gz -C /var/lib/cdsw/
```

4. Start the Cloudera Data Science Workbench.

Cloudera Data Science Workbench Scaling Guidelines

New hosts can be added and removed from a Cloudera Data Science Workbench deployment without interrupting any jobs already scheduled on existing hosts. Therefore, it is rather straightforward to increase capacity based on observed usage. At a minimum, Cloudera recommends you allocate at least 1 CPU core and 2 GB of RAM per concurrent session or job. CPU can burst above a 1 CPU core share when spare resources are available. Therefore, a 1 CPU core allocation is often adequate for light workloads. Allocating less than 2 GB of RAM can lead to out-of-memory errors for many applications.

As a general guideline, Cloudera recommends hosts with RAM between 60GB and 256GB, and between 16 and 48 cores. This provides a useful range of options for end users. SSDs are strongly recommended for application data storage.



Note: At present, Kubernetes does not automatically pick the CPU/RAM that is dynamically added to the virtual machines (VM) while the VM is on. Therefore, you need to restart the CDSW service/role on the node on which you have updated the capacity.

For some data science and machine learning applications, users can collect a significant amount of data in memory within a single R or Python process, or use a significant amount of CPU resources that cannot be easily distributed into the CDH cluster. If individual users frequently run larger workloads or run workloads in parallel over long durations, increase the total resources accordingly. Understanding your users' concurrent workload requirements or observing actual usage is the best approach to scaling Cloudera Data Science Workbench.

Ports Used By Cloudera Data Science Workbench

Cloudera Data Science Workbench runs on gateway hosts in a CDH/HDP cluster. As such, Cloudera Data Science Workbench acts as a gateway and requires full connectivity to cluster services such as Impala, Spark 2, etc. Additionally, in the case of Spark 2, cluster hosts will require access to the Spark driver running on a set of random ports (20050-32767) on Cloudera Data Science Workbench hosts.

Firewall restrictions must be disabled across Cloudera Data Science Workbench and CDH/HDP cluster hosts. Internally, the Cloudera Data Science Workbench master and worker hosts require full connectivity with no firewalls. Externally,

end users connect to Cloudera Data Science Workbench exclusively through a web server running on the master host, and therefore do not need direct access to any other internal Cloudera Data Science Workbench or CDH services.

This information has been summarized in the following table.

Components	Details
Communication with the CDH / HDP cluster	CDH / HDP -> Cloudera Data Science Workbench The CDH/HDP cluster must have access to the Spark driver that runs on Cloudera Data Science Workbench hosts, on a set of randomized ports in the range, 20050-32767 .
	Cloudera Data Science Workbench -> CDH / HDP As a gateway service, Cloudera Data Science Workbench must have access to all the ports used by CDH and Cloudera Manager .
Communication with the Web Browser	The Cloudera Data Science Workbench web application is available at port 80 . HTTPS access is available over port 443 .

Managing Cloudera Data Science Workbench Hosts

This topic describes how to perform some common tasks related to managing Cloudera Data Science Workbench hosts.

Customize Workload Scheduling

Starting with version 1.6, Cloudera Data Science Workbench allows you to specify a list of CDSW gateway hosts that are labeled as **Auxiliary Nodes**. These hosts will be deprioritized during workload scheduling. That is, they will be chosen to run workloads that can't be scheduled on any other hosts. For example, sessions with very large resource requests, or when the other hosts are fully utilized. This means, Cloudera Data Science Workbench will use the following order of preference when scheduling non-GPU workloads (session, job, experiment, or model):

Worker Hosts > Master Host > GPU-equipped Hosts | Labeled Auxiliary Hosts

When selecting a host to schedule an engine, Cloudera Data Science Workbench will give first preference to unlabeled Worker hosts. If Workers are unavailable or at capacity, CDSW will then leverage the Master host. And finally, any GPU-equipped hosts OR labeled auxiliary hosts will be leveraged.

Points to Note:

- **GPU-equipped Hosts** - Hosts equipped with GPUs will be **labeled auxiliary by default** so as to reserve them for GPU-intensive workloads. They do not need to be explicitly configured to be labeled. A GPU-equipped host and a labeled auxiliary host will be given equal priority when scheduling workloads.
- **Master Host** - The Master host **must not** be labeled an auxiliary node. If you want to reserve the Master for running internal Cloudera Data Science Workbench application components, use the [Reserve Master Host](#) property.

Labeling Auxiliary Hosts

Before you proceed, make sure you have reviewed the guidelines on [customizing workload scheduling](#) in Cloudera Data Science Workbench.

Depending on your deployment type, use one of the following sets of instructions to use this feature:

CSD Deployments

On CSD deployments, use the **Auxiliary Nodes** property in the CDSW service in Cloudera Manager to specify a comma-separated list of auxiliary hosts.

1. Log into the Cloudera Manager Admin Console.
2. Go to the **CDSW** service.
3. Click the **Configuration** tab.

4. Search for the following property: **Auxiliary Notes**.
5. Enter the hostnames that you want to label as auxiliary.
6. Click **Save Changes**.
7. Restart the CDSW service to have this change go into effect.

RPM Deployments

On RPM, deployments, use the `AUXILIARY_NODES` property in `cdsw.conf` to specify a comma-separated list of auxiliary hosts.

Reserving the Master Host for Internal CDSW Components



Important: This feature only applies to deployments with more than one Cloudera Data Science Workbench host. Enabling this feature on single-host deployments will leave Cloudera Data Science Workbench incapable of scheduling any workloads.

Cloudera Data Science Workbench allows you to reserve the master host for running internal application components and services such as Livelog, the PostgreSQL database, and so on, while user workloads run exclusively on worker hosts.

By default, the master host runs both, user workloads as well as the application's internal services. However, depending on the size of your CDSW deployment and the number of workloads running at any given time, it's possible that user workloads might dominate resources on the master host. Enabling this feature will ensure that CDSW's application components always have access to the resources they need on the master host and are not adversely affected by user workloads.

Depending on your deployment type, use one of the following sets of instructions to enable this feature:

CSD Deployments

On CSD-based deployments, this feature can be enabled in Cloudera Manager. Note that this feature is not yet available as a configuration property in Cloudera Manager. However, you can use an Advanced Configuration Snippet (Safety Valve) to configure this as follows:

1. Log into the Cloudera Manager Admin Console.
2. Go to the **CDSW** service.
3. Click the **Configuration** tab.
4. Search for the following property: **Reserve Master Host**. Select the checkbox to enable it.
5. Click **Save Changes**.
6. Restart the CDSW service to have this change go into effect.

RPM Deployments

To enable this feature on RPM-based deployments, go to the `/etc/cdsw/config/cdsw.conf` file and set the `RESERVE_MASTER` property to `true`.

Migrating a Deployment to a New Set of Hosts

The following topics describe how to migrate a Cloudera Data Science Workbench deployment to a new set of gateway hosts.

- [Migrating a CSD Deployment](#) on page 272
- [Migrating an RPM Deployment](#) on page 274

Adding a Worker Host



Note: For proof-of-concept deployments, you can deploy a 1-host cluster with just a Master host. The Master host can run user workloads just as a worker host can when required for demonstration purposes. For production deployments, Cloudera requires to have a reserved, dedicated master host and separate worker host(s).

Using Cloudera Manager

Perform the following steps to add a new worker host to Cloudera Data Science Workbench.

1. Log in to the Cloudera Manager Admin Console.
2. [Add a new host to your cluster](#). Make sure this is a gateway host and you are not running any services on this host.
3. Assign the HDFS, YARN, and Spark 2 gateway roles to the new host. For instructions, refer the Cloudera Manager documentation at [Adding a Role Instance](#).



Note: If you are using Spark 2.2 (or higher), review [JDK 8 Requirement for Spark 2.2 \(or higher\)](#) on page 67 for any additional steps you might need to perform to configure `JAVA_HOME` on the new nodes.

4. Go to the Cloudera Data Science Workbench service.
5. Click the **Instances** tab.
6. Click **Add Role Instances**.
7. Assign the Worker and Docker Daemon roles to the new host. Click **Continue**.
8. Review your changes and click **Continue**. The wizard finishes by performing any actions necessary to add the new role instances. **Do not start the new roles at this point. You must run the Prepare Node command as described in the next steps before the roles are started.**
9. The new host must have the following packages installed on it.

```
nfs-utils
libseccomp
lvm2
bridge-utils
libtool-ltdl
iptables
rsync
policycoreutils-python
selinux-policy-base
selinux-policy-targeted
ntp
etables
bind-utils
nmap-ncat
openssl
e2fsprogs
redhat-lsb-core
conntrack-tools
socat
```

You must either manually install these packages now, *or*, allow Cloudera Manager to install them in the next step.

If you choose the latter, make sure that Cloudera Manager has the permission needed to install the required packages. To do so, go to the Cloudera Data Science Workbench service and click **Configuration**. Search for the **Install Required Packages** property and make sure it is enabled.

10. Click **Instances** and select the new host. From the list of available actions, select the **Prepare Node** command to install the required packages on the new node.
11. On the **Instances** page, select the new role instances and click **Actions for Selected > Start**.



Note: It can take several minutes to initialize the host, load Docker images, and be available after you start it. Cloudera Manager may show the host in a healthy state (or green) while the host is being initialized in the background.

Using Packages

On an RPM deployment, the procedure to add a worker host to an existing deployment is the same as that required when you first install Cloudera Data Science Workbench on a worker. For instructions, see [Installing Cloudera Data Science Workbench on a Worker Host](#).

Removing a Worker Host

Using Cloudera Manager

Perform the following steps to remove a worker host from Cloudera Data Science Workbench.

1. Log into the Cloudera Manager Admin Console.
2. Click the **Instances** tab.
3. Select the Docker Daemon and Worker roles on the host to be removed from Cloudera Data Science Workbench.
4. Select **Actions for Selected** > **Stop** and click **Stop** to confirm the action. Click **Close** when the process is complete.
5. On the **Instances** page, re-select the Docker Daemon and Worker roles that were stopped in the previous step.
6. Select **Actions for Selected** > **Delete** and then click **Delete** to confirm the action.

Using Packages

To remove a worker host:

1. On the master host, run the following command to delete the worker host:

```
kubect1 delete node <worker_host_domain_name>
```

2. Reset the worker host.

```
cdsw stop
```

Changing the Domain Name

Cloudera Data Science Workbench allows you to change the domain of the web console.

Using Cloudera Manager

1. Log into the Cloudera Manager Admin Console.
2. Go to the Cloudera Data Science Workbench service.
3. Click the **Configuration** tab.
4. Search for the **Cloudera Data Science Workbench Domain** property and modify the value to reflect the new domain.
5. Click **Save Changes**.
6. Restart the Cloudera Data Science Workbench service to have the changes go into effect.

Using Packages

1. Open `/etc/cdsw/config/cdsw.conf` and set the `DOMAIN` variable to the new domain name.

```
DOMAIN="cdsw.<your_new_domain>.com"
```

2. Run the following commands to have the new domain name go into effect.

```
cdsw stop
cdsw start
```

Migrating a Deployment to a New Set of Hosts

This section describes how to migrate a Cloudera Data Science Workbench deployment to a new set of gateway hosts.

Migrating a CSD Deployment

This section describes how to migrate a CSD-based Cloudera Data Science Workbench service to a new set of gateway hosts.

Add and Set Up the New Hosts

1. [Add new hosts to your cluster as needed](#). Make sure they are gateway hosts that have been assigned gateway roles for HDFS, YARN, and Spark 2. Do not run any other services on these hosts.
2. Set up the new hosts as per the Cloudera Data Science Workbench hardware requirements listed [here](#).
 - [Disable Untrusted SSH Access](#) on page 74 on the new hosts.
 - [Configure Block Devices](#) on page 75 on the new hosts.

Copy the JDK to the new host

Copy the `/usr/java` directory to the new host.

Copy the DNS Nameserver to the new host

Copy the `/etc/resolv.conf` file to the new host.

Copy the Kerberos Configurations

Copy the `/etc/jkr5.conf` file to the new host.

Stop the CDSW Service



Important: On Cloudera Data Science Workbench 1.4.0 (and lower), **do not** stop or restart Cloudera Data Science Workbench without using the [cdsw_protect_stop_restart.sh](#) script. This is to help avoid the data loss issue detailed in [TSB-346](#).

Use Cloudera Manager to stop all roles of the CDSW service.

1. Log into the Cloudera Manager Admin Console.
2. On the **Home > Status** tab, click



to the right of the **CDSW** service and select **Stop** from the dropdown.

3. Confirm your choice on the next screen. When you see a **Finished** status, the action is complete.

Backup Application Data

In Cloudera Data Science Workbench all stateful data is stored on the master host at `/var/lib/cdsw`. Backup the contents of this directory before you begin the migration process.

1. Stop Cloudera Data Science Workbench.

2. After stopping CDSW, and before running the following tar command, wait 2-5 minutes (depending on your disk speed) to ensure that all data from CDSW is successfully written to the disks. Otherwise the tar command may not capture all recent changes.
3. To create the backup, run the following command on the *master* host:

```
tar -cvzf cdsw.tar.gz -C /var/lib/cdsw/ .
```

Delete CDSW Roles from Existing Hosts

1. Log into the Cloudera Manager Admin Console.
2. Go to the CDSW service.
3. Click the **Instances** tab.
4. Select all the role instances.
5. Select **Actions for Selected** > **Delete**. Click **Delete** to confirm the deletion.

Move Backup to the New Master

Copy the backup taken previously to the host that will be the new Cloudera Data Science Workbench master. Unpack the contents of the backup into `/var/lib/cdsw`.

```
tar xvzf cdsw.tar.gz -C /var/lib/cdsw
```

Update DNS Records for the New Master

Update your DNS records with the IP address for the new master host.

Add Role Instances for the New Hosts

1. Log into the Cloudera Manager Admin Console.
2. Go to the CDSW service.
3. Click the **Instances** tab.
4. Click **Add Role Instances**. Assign the Cloudera Data Science Workbench Master, Application, and Docker Daemon roles to the new master host. If you want to configure worker hosts, assign the Cloudera Data Science Workbench Worker and Docker Daemon roles to the new workers.



Important: Do not assign the Master and Worker roles to the same host. Even if you only have one host for Cloudera Data Science Workbench, the Master can automatically perform the functions of a Worker host as needed.

5. Click **Continue**. On the Review Changes page, review the configuration changes to be applied. The wizard finishes by performing any actions necessary to add the new role instances.

Do not start the new roles at this point. You must run the Prepare Node command as described in the next step before the roles are started.

Run the Prepare Node command on the New Hosts

The new hosts must have the following packages installed on it.

```
nfs-utils
libseccomp
lvm2
bridge-utils
libtool-ltdl
iptables
rsync
policycoreutils-python
selinux-policy-base
selinux-policy-targeted
```

```
ntp
e2fsprogs
bind-utils
nmap-ncat
openssl
redhat-lsb-core
conntrack-tools
socat
```

You can either manually install these packages now, *or*, allow Cloudera Manager to install them as part of the **Prepare Node** command later in this step.

If you choose the latter, make sure that Cloudera Manager has the permissions needed to install the required packages. To do so, go to the CDSW service and click **Configuration**. Search for the **Install Required Packages** property and make sure it is enabled.

Then run the **Prepare Node** command on the new hosts.

1. Go to the CDSW service.
2. Click **Instances**.
3. Select all the role instances.
4. Select **Actions for Selected** > **Prepare Node**. This will install the required set of packages on all the new hosts.

Start the CDSW Service

1. Log into the Cloudera Manager Admin Console.
2. On the **Home** > **Status** tab, click



to the right of the **CDSW** service and select **Start** from the dropdown.

3. Confirm your choice on the next screen. When you see a **Finished** status, the action is complete.

Migrating an RPM Deployment

This section describes how to migrate an RPM-based Cloudera Data Science Workbench service to a new set of gateway hosts.

Add and Set Up the New Hosts

1. [Add new hosts to your cluster as needed](#). Make sure they are gateway hosts that have been assigned gateway roles for HDFS, YARN, and Spark 2. Do not run any other services on these hosts.
2. Set up the new hosts as per the Cloudera Data Science Workbench hardware requirements listed [here](#).
 - [Disable Untrusted SSH Access](#) on page 74 on the new hosts.
 - [Configure Block Devices](#) on page 75 on the new hosts.

Copy the JDK to the new host

Copy the `/usr/java` directory to the new host.

Copy the DNS Nameserver to the new host

Copy the `/etc/resolv.conf` file to the new host.

Copy the Kerberos Configurations

Copy the `/etc/jkr5.conf` file to the new host.

Stop Cloudera Data Science Workbench



Important: On Cloudera Data Science Workbench 1.4.0 (and lower), **do not** stop or restart Cloudera Data Science Workbench without using the [cdsw_protect_stop_restart.sh](#) script. This is to help avoid the data loss issue detailed in [TSB-346](#).

Run the following command on the master host to stop Cloudera Data Science Workbench.

```
cdsw stop
```

Backup Application Data

In Cloudera Data Science Workbench all stateful data is stored on the master host at `/var/lib/cdsw`. Backup the contents of this directory before you begin the migration process.

1. Stop Cloudera Data Science Workbench.
2. After stopping CDSW, and before running the following tar command, wait 2-5 minutes (depending on your disk speed) to ensure that all data from CDSW is successfully written to the disks. Otherwise the tar command may not capture all recent changes.
3. To create the backup, run the following command on the *master* host:

```
tar -cvzf cdsw.tar.gz -C /var/lib/cdsw/ .
```

Remove Cloudera Data Science Workbench from Existing Hosts

Run the following commands on the existing master and any worker hosts you want to migrate.

```
cdsw stop
yum remove cloudera-data-science-workbench
```

Move Backup to New Master

Copy the backup taken in the previous step to the host that will be the new Cloudera Data Science Workbench master. Unpack the contents of the backup into `/var/lib/cdsw`.

```
tar xvzf cdsw.tar.gz -C /var/lib/cdsw
```

Update DNS Records for the New Master

Update your DNS records with the IP address for the new master host.

Install Cloudera Data Science Workbench on New Master Host

For instructions, see [Installing Cloudera Data Science Workbench 1.7.2 Using Packages](#) on page 83.

Rollback Cloudera Data Science Workbench to an Older Version

All stateful data for Cloudera Data Science Workbench is stored in the `/var/lib/cdsw` directory on the Master host. The contents of this directory are forward compatible, which is what allows for upgrades. However, they are *not* backward compatible. Therefore, to rollback Cloudera Data Science Workbench to a previous version, you must have a backup of the `/var/lib/cdsw` directory, taken prior to the last upgrade.

In general, the steps required to restore a previous version of Cloudera Data Science Workbench are:

1. Depending on your deployment, either uninstall the RPM or deactivate the current CDSW parcel in Cloudera Manager.
2. On the master host, restore the backup copy you have of `/var/lib/cdsw`. Note that any changes made after this backup will be lost.

3. Install a version of Cloudera Data Science Workbench that is *equal to or greater than* the version of the `/var/lib/cdsw` backup.

Uninstalling Cloudera Data Science Workbench

This topic describes how to uninstall Cloudera Data Science Workbench from a cluster.

CSD Deployments

1. Log in to the Cloudera Manager Admin Console.
- 2.



Important: On Cloudera Data Science Workbench 1.4.0 (and lower), **do not** stop or restart Cloudera Data Science Workbench without using the [cdsw_protect_stop_restart.sh](#) script. This is to help avoid the data loss issue detailed in [TSB-346](#).

On the **Home > Status** tab, click



to the right of the **CDSW** service and select **Stop** from the dropdown and confirm that you want to stop the service.

3. (**Strongly Recommended**) On the master host, backup the contents of the `/var/lib/cdsw` directory. This is the directory that stores all your application data.

To create the backup, run the following command on the *master* host:

```
tar -cvzf cdsw.tar.gz -C /var/lib/cdsw/ .
```

4. Go back to Cloudera Manager and click **Hosts > Parcels**.
5. Go to the CDSW parcel and click **Deactivate**.
6. Select **Deactivate Only** from the list.
7. Click the



to the right of the **Activate** button and select **Remove From Hosts**.

8. Click **OK** to confirm.
9. Go back to the **Home > Status** page and click



to the right of the **CDSW** service. Select **Delete** from the dropdown and confirm that you want to delete the service.

- 10 Remove all your user data that is stored in `/var/lib/cdsw` on the *master* host from the deployment. This step will permanently remove all user data.

```
sudo rm -rf /var/lib/cdsw
```

For more details on how to uninstall Cloudera Manager and its components, see [Uninstalling Cloudera Manager and Managed Software](#).

RPM Deployments

- 1.



Important: On Cloudera Data Science Workbench 1.4.0 (and lower), **do not** stop or restart Cloudera Data Science Workbench without using the [cdsw_protect_stop_restart.sh](#) script. This is to help avoid the data loss issue detailed in [TSB-346](#).

Run the following command on the master host to stop Cloudera Data Science Workbench.

```
cdsw stop
```

- 2. (Strongly Recommended)** On the master host, backup the contents of the `/var/lib/cdsw` directory. This is the directory that stores all your application data.

To create the backup, run the following command on the *master* host:

```
tar -cvzf cdsw.tar.gz -C /var/lib/cdsw/ .
```

- 3.** To uninstall Cloudera Data Science Workbench, run the following commands on the master host and all the worker hosts.

```
cdsw stop  
yum remove cloudera-data-science-workbench
```

- 4.** Remove all your user data that is stored in `/var/lib/cdsw` on the *master* host from the deployment. This step will permanently remove all user data.

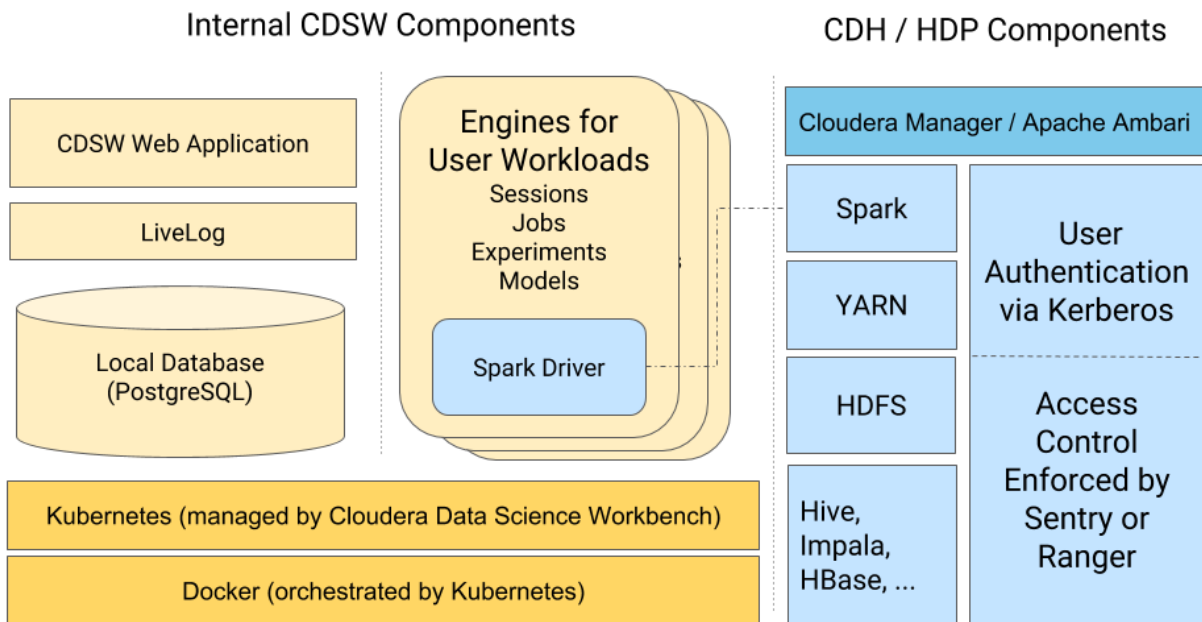
```
sudo rm -Rf /var/lib/cdsw
```

Cloudera Data Science Workbench Security Guide

This topic provides an overview of the Cloudera Data Science Workbench security model and describes how Cloudera Data Science Workbench leverages the [security and governance capabilities of your Cloudera Enterprise cluster](#) to deliver a secure, robust, collaborative data science platform for the enterprise.

Security Model

Cloudera Data Science Workbench uses Kubernetes to schedule and manage Docker containers. All Cloudera Data Science Workbench components (web application, PostgreSQL database, LiveLog, and so on) execute inside Docker containers. These are represented on the left-hand side of the diagram. Similarly, the environments that users operate in (via sessions, jobs, experiments, models), also run within isolated Docker containers that we call engines.



This architecture allows Cloudera Data Science Workbench to leverage the isolation properties of Docker to achieve notable security benefits.

Benefits of the Docker Isolation Model - Docker containers share the underlying host operating system, but are isolated from the rest of the host and from each other. Each container gets its own:

- **Isolated File System:** The Docker container does not see the host file system, but instead sees only the filesystem provided by the container and any host volumes that you have explicitly mounted into the container. This means a user launching a Cloudera Data Science Workbench session will only have access to the project files, and any specific host volumes you have chosen to mount into the session engine. They will not otherwise have access to the underlying host filesystem.
- **Isolated Process Namespace:** Docker containers cannot affect any processes running either on the host operating system or in other containers. Cloudera Data Science Workbench creates a new container each time a session/job/experiment/model is launched. This means user workloads can run in complete isolation from each other.

For more details on the Docker security model, see [Docker Security Overview](#).

Wildcard DNS Subdomain Requirement

When you first set up Cloudera Data Science Workbench, you are asked to create a wildcard DNS entry for the Cloudera Data Science Workbench domain. Cloudera Data Science Workbench uses these wildcard subdomains (`*.cdsw.<your_domain>.com`) to route HTTP requests to engines and services launched by users.

Every time users launch workloads (session/job/experiment/model) on Cloudera Data Science Workbench, a new engine is created for each workload. These engines are isolated Docker containers where users can execute code. Each engine is assigned its own unique, randomly-generated ID, which is saved to the `CDSW_ENGINE_ID` [environmental variable](#). This ID is also used to create a unique subdomain for each engine. These subdomains are of the form: `<CDSW_ENGINE_ID>.cdsw.<your_domain>.com`.

Assigning a unique subdomain to each engine allows Cloudera Data Science Workbench to:

- Securely expose interactive session services, such as visualizations, the terminal, and web UIs such as TensorBoard, Shiny, Plotly, and so on;
- Prevent cross-site scripting (XSS) attacks by securely isolating user-generated content from the Cloudera Data Science Workbench application.

It is important to note that because there is no limit to the number of workloads (i.e. engines) users can launch, Cloudera Data Science Workbench requires the ability to randomly generate large numbers of engine IDs (and their subdomains) on-demand. Therefore, creating a wildcard DNS subdomain is essential for Cloudera Data Science Workbench to function successfully.

Additionally, if you want to enable TLS for your deployment, your TLS certificate will need to include both, the Cloudera Data Science Workbench domain, as well as the wildcard for all first-level subdomains. This is required so that your browser can trust communications with the `<CDSW_ENGINE_ID>.cdsw.<your_domain>.com` subdomains.

Authentication

Authentication occurs in various forms in the Cloudera Data Science Workbench application. These are:

- **User Login:** Occurs when users log in to the Cloudera Data Science Workbench Web UI and authenticate themselves to the application. Cloudera Data Science Workbench works with the following authentication back-ends: the local CDSW database, LDAP/AD, and SAML. Using either [LDAP or SAML](#) is recommended, as it eases the administrative burden of managing identity in Cloudera Data Science Workbench.
- **SSH Key Authentication:** Each user account is assigned an [SSH key pair](#) for use in sessions. This SSH key pair can be used to authenticate to an external version control system such as Git. Only the public key appears in the UI; the private key is loaded into user sessions when launched.
- **Hadoop Cluster Authentication:** Authentication to the underlying CDH/HDP cluster is handled via Kerberos. To authenticate themselves to the cluster, users must provide a Kerberos principal and keytab/password. These credentials are stored in the internal Cloudera Data Science Workbench database. Note that Cloudera Data Science Workbench user sessions are only provided with the Kerberos credential-cache file which does not store the user's password. For more details, see [Hadoop Authentication with Kerberos for Cloudera Data Science Workbench](#) on page 291.
- **API Authentication:** Each user account is assigned an API Key that can be used for authentication when communicating with the Cloudera Data Science Workbench API. For more details, see [Cloudera Data Science Workbench Jobs API](#) on page 200.
- **Per-Model Authentication:** Each model is assigned a unique Access Key. This access key serves as an authentication token that allows users to make calls to the model once it has been deployed. For details, see [Model Access Key](#).

Authorization

This section describes how Cloudera Data Science Workbench handles authorization for users attempting to connect to the attached CDH/HDP cluster, and how Cloudera Data Science Workbench users and teams are granted access to projects.

Cluster Authorization

Cloudera Data Science Workbench connects to your cluster like any other client. It relies on your cluster's built-in security layer to enforce strong authentication and authorization for connections from Cloudera Data Science Workbench.

Cloudera Data Science Workbench users attempting to connect to a secure CDH or HDP cluster must first [authenticate themselves using their Kerberos credentials](#). Once a user has been successfully authenticated, Cloudera Data Science Workbench then depends on your cluster's built-in authorization tools (such as HDFS ACLs, Sentry, Ranger, and so on) to enforce their own existing access control rules when a Cloudera Data Science Workbench user attempts to access data on the cluster.

For example, let's assume you are running a secure CDH cluster with Impala (secured by Sentry). On this cluster, only the `impala-admin` group has been granted the required Sentry privileges to modify data in Impala. If a Cloudera Data Science Workbench user runs a Python job that attempts to modify data in an Impala table, Sentry privileges will be enforced. That is, if the user is not a member of the `impala-admin` group, the access request will be denied.

Similarly, if a Cloudera Data Science Workbench user is attempting to submit Spark jobs on secure HDP clusters, Ranger policies (via the respective HDFS and YARN plugins) will be enforced to decide whether the user has the permission to access the requested directories in HDFS, and whether they can submit jobs to the Spark queue.

This means as long as your cluster components have been secured with access control rules (via ACLs or Sentry or Ranger) that forbid specific users/groups from performing certain actions on the cluster, then all access attempts from Cloudera Data Science Workbench users will also be subject to those rules.

User Role Authorization

Cloudera Data Science Workbench has one major specialized user role across the deployment: [site administrators](#). Site administrators are superusers who have complete access to all activity on your Cloudera Data Science Workbench deployment. This includes all the configuration settings, projects (private and public), and workloads running on the deployment.

When LDAP or SAML based authentication is used, it is possible to restrict the users who can log into Cloudera Data Science Workbench based on their LDAP/SAML group membership. Additionally, you can specify LDAP/SAML groups that should automatically be granted site administrator privileges when they log in to Cloudera Data Science Workbench.

Apart from site administrators, every other Cloudera Data Science Workbench user is granted access to projects either as a project collaborator or as part of a team. Continue reading to find out how these project and team permissions interact with each other.

Access Control for Teams and Projects

When a team or project is created, the Team/Project Admin role is assigned to the user who created it. Other Team/Project Admins can be assigned later but there must always be at least one user assigned as Admin for the team or project. Team and project administrators then decide what level of access other users are granted per-team or per-project.

Project Access Levels

Users who are explicitly added to a project are referred to as project collaborators. Project collaborators can be assigned one of the following levels of access:

- **Viewer** - Read-only access to code, data, and results.
- **Operator** - Read-only access to code, data, and results. Additionally, Operators can start and stop existing jobs in the projects that they have access to.

- **Contributor** - Can view, edit, create, and delete files and environmental variables, run sessions/experiments/jobs/models and execute code in running jobs. Additionally, Contributors can set the default engine for the project.
- **Admin** - Has complete access to all aspects of the project. This includes the ability to add new collaborators, and delete the entire project.

Team Access Levels

Users who are explicitly added to a team are referred to as team members. Team members can be assigned one of the following levels of access:

- **Viewer** - Read-only access to team projects. Cannot create new projects within the team but can be added to existing ones.
- **Operator** - Read-only access to team projects. Cannot create new projects within the team but can be added to existing ones. Additionally, Operators can start and stop existing jobs in the projects that they have access to.
- **Contributor** - Write-level access to all team projects to all team projects with **Team** or **Public** visibility. Can create new projects within the team. They can also be added to existing team projects.
- **Admin** - Has complete access to all team projects, can add new team members, and modify team account information.

Project Visibility Levels

Projects can be created either in your personal [context](#), or in a team context. Furthermore, projects can be created with one of the following visibility levels:

- **Private** - Private projects can be created either in your personal context, or in a team context. They can only be accessed by project collaborators.
- **Team** - Team projects can only be created in a team context. They can be viewed by all members of the team.
- **Public** - Public projects can be created either in your personal context, or in a team context. They can be viewed by all authenticated Cloudera Data Science Workbench users.

It is important to remember that irrespective of the visibility level of the project, site administrators will always have complete Admin-level access to all projects on Cloudera Data Science Workbench. Additionally, depending on the visibility level of the project, and the context in which it was created, a few other users/team members might also have Contributor or Admin-level access to your project by default.

Use the following table to find out who might have default access to your projects on Cloudera Data Science Workbench.

Project Visibility	Access Levels for Cloudera Data Science Workbench Users
Private Visibility	<p>Private Projects Created in Personal Context</p> <p>The following user roles will have access to private projects in your personal context:</p> <p>Admin Access</p> <ul style="list-style-type: none"> • Site Administrators • Project Admins <p>Contributor Access</p> <ul style="list-style-type: none"> • Collaborators explicitly added to the project and given Contributor access. <p>Operator Access</p> <ul style="list-style-type: none"> • Operators explicitly added to the project and given Operator access. <p>Viewer Access</p> <ul style="list-style-type: none"> • Viewers explicitly added to the project and given Viewer access.
	<p>Private Projects Created in a Team Context</p>

Project Visibility	Access Levels for Cloudera Data Science Workbench Users
	<p>For private projects created within a team context, project-level permissions granted by Project Admins will take precedence over team-level permissions. The only exception to this rule are users who are Team Admins. Team Admins will always have Admin-level access to all projects within their team context, irrespective of the access level granted to them per-project.</p> <p>The following user roles will have access to private projects created within a team context:</p> <p>Admin Access</p> <ul style="list-style-type: none"> • Site Administrators • Project Admins • Team Admins <p>Contributor Access</p> <ul style="list-style-type: none"> • Collaborators explicitly added to the project and given Contributor access. <p>Operator Access</p> <ul style="list-style-type: none"> • Operators explicitly added to the project and given Operator access. <p>Viewer Access</p> <ul style="list-style-type: none"> • Viewers explicitly added to the project and given Viewer access.
<p>Team Visibility</p>	<p>Team Projects</p> <p>Projects with Team visibility can only be created in a team context. For team projects, both team access levels and project access levels must be taken into consideration to determine who has access to these projects.</p> <p>Points to note:</p> <ul style="list-style-type: none"> • Team members do not need to be explicitly added as project collaborators to have access to a team project. While you can explicitly invite specific team members to collaborate on your project, it is important to remember that <i>all</i> team members will have some level of access to your project. <p>The project Collaborators page does not list all team members; it only lists those you have explicitly added as collaborators. However, team members will still have access to all team projects. By default, their level of access to the projects is the same as their level of access to the team.</p> <ul style="list-style-type: none"> • Project Admins cannot downgrade access levels for team members. If project-level permissions don't match up to team-level permissions, team permissions will take precedence. <p>For example, if you add a Team Contributor as a collaborator to a team project, but only give them Project Viewer permission, the user will still have Contributor-level access to the project. Similarly, Team Admins will always have Admin-level access to all projects within their team context, irrespective of the access level granted to them per-project.</p> <ul style="list-style-type: none"> • Project Admins also cannot upgrade access levels for team members with Viewer-level access to the team. That is, Team Viewers cannot be given Contributor or Admin access to any team projects. <p>The following user roles will have access to team projects on Cloudera Data Science Workbench:</p> <p>Admin Access</p> <ul style="list-style-type: none"> • Site Administrators • Team Admins

Project Visibility	Access Levels for Cloudera Data Science Workbench Users
	<ul style="list-style-type: none"> Project Admins <p>Contributor Access</p> <ul style="list-style-type: none"> All team members with Contributor access. <p>Operator Access</p> <ul style="list-style-type: none"> All team members with Operator access. <p>Viewer Access</p> <ul style="list-style-type: none"> All team members with Viewer access.
Public Visibility	<p>Public Projects Created in Personal Context</p> <p>The following user roles will have access to public projects on Cloudera Data Science Workbench:</p> <p>Admin Access</p> <ul style="list-style-type: none"> Site Administrators Project Admins <p>Contributor Access</p> <ul style="list-style-type: none"> Collaborators explicitly added to the project and given Contributor access. <p>Operator Access</p> <ul style="list-style-type: none"> Operators explicitly added to the project and given Operator access. <p>Viewer Access</p> <ul style="list-style-type: none"> All authenticated CDSW users. <hr/> <p>Public Projects Created in a Team Context</p> <p>The following user roles will have access to public projects created in team contexts:</p> <p>Admin Access</p> <ul style="list-style-type: none"> Site Administrators Project Admins Team Admins <p>Contributor Access</p> <ul style="list-style-type: none"> All team members with Contributor access. <p>The team/project access rules and nuances described in the Team section apply here as well.</p> <p>Operator Access</p> <ul style="list-style-type: none"> All team members with Operator access. <p>Viewer Access</p> <ul style="list-style-type: none"> All authenticated CDSW users.



Note: Restricting Access to Active Sessions

Users with Admin or Contributor-level permissions on projects have access to all of the project's active sessions and can execute commands within these active sessions. Cloudera Data Science Workbench (1.4.3 and higher) includes a feature that allows site administrators to restrict this ability by allowing only session creators to execute commands within their own active sessions. For details on how to enable this, see [Restricting Access to Active Sessions](#).

Wire Encryption

External Communications

Cloudera Data Science Workbench uses HTTP and WebSockets (WS) to support interactive connections to the Cloudera Data Science Workbench web application. However, these connections are not secure by default.

For secure, encrypted communication, Cloudera Data Science Workbench can be configured to use a TLS termination proxy to handle incoming connection requests. The termination proxy server will decrypt incoming connection requests and forward them to the Cloudera Data Science Workbench web application.

The Cloudera Data Science Workbench documentation describes two different approaches to TLS termination: [internal and external TLS termination](#). Both provide a secure TLS connection between users and Cloudera Data Science Workbench. If you require more control over the TLS protocol and cipher suite, we recommend external termination. Both approaches require TLS certificates that list both, the Cloudera Data Science Workbench domain, as well as a wildcard for all first-level subdomains. For example, if the Cloudera Data Science Workbench domain is `cdsw.<your_domain>.com`, then the TLS certificate must include both `cdsw.<your_domain>.com` and `*.cdsw.<your_domain>.com`.

Browser Security

Cloudera Data Science Workbench also allows you to customize the HTTP headers accepted by Cloudera Data Science Workbench. The list of security headers enabled by default can be found in the documentation here: [HTTP Headers](#). Disabling these features could leave your Cloudera Data Science Workbench deployment vulnerable to clickjacking, cross-site scripting (XSS), or any other injection attacks.

Internal Communications

Internal communications between some Cloudera Data Science Workbench components are protected by mutually authenticated TLS.

The underlying Kubernetes cluster has a root Certificate Authority (CA) that is used to validate certificates for internal Kubernetes components. For details, refer the Kubernetes documentation here: [TLS certificates](#).

Cloudera Data Science Workbench Gateway Host Security

The [Cloudera Data Science Workbench master host](#) stores all the critical, stateful, persistent data for a Cloudera Data Science Workbench deployment. This data includes your deployment secrets, such as, Kerberos credentials, encrypted passwords, SSH and API keys, and so on. While the Cloudera Data Science Workbench worker hosts do not store the same secrets, they also store sensitive information. Therefore, protecting the master and worker hosts is extremely important. Cloudera recommends the following security best practices for all the Cloudera Data Science Workbench hosts:

- Disable untrusted SSH access to the Cloudera Data Science Workbench hosts. Cloudera Data Science Workbench assumes that users only access the gateway hosts through the web application. Users with SSH access to a Cloudera Data Science Workbench host can gain full access to the cluster, including access to other users' workloads. Therefore, untrusted (non-sudo) SSH access to Cloudera Data Science Workbench hosts must be disabled to ensure a secure deployment.

- Tightly control root access via sudo.
- Uninstall or disable any other unnecessary services running on the Cloudera Data Science Workbench gateway hosts.
- Keep the hosts' operating system updated to avoid security vulnerabilities.
- Monitor user login activity on the system.

Host Mounts

Cloudera Data Science Workbench allows site administrators to expose part of the host's file system into users' engine containers at runtime. This is done using the [host mounts](#) feature. It is worth noting that directories mounted using this feature are then available to *all* projects across the deployment. Use this feature with great care and ensure that no sensitive information is mounted.

Base Engine Image Security

The base engine image is a Docker image that contains all the building blocks needed to launch a Cloudera Data Science Workbench session and run a workload. It consists of kernels for Python, R, and Scala, and some common third-party libraries and packages that can be used to run common data analytics operations. Additionally, to provide a flexible, collaborative environment, data science teams can install their own preferred data science packages, just as they would on their local computers, to run workloads on data in the associated Hadoop cluster.

This base image itself is built and shipped along with Cloudera Data Science Workbench. Cloudera Data Science Workbench strives to ship a base engine image free from security vulnerabilities. Our engine images are regularly scanned for potential security vulnerabilities and we have been incorporating the recommendations from these scans into the product. For organizations that require more control over the contents of the engines used by Cloudera Data Science Workbench users, we recommend building your own [customized engine images](#).

Engine-level Security for Custom Editors and other Web Applications

Enabling TLS/SSL for Cloudera Data Science Workbench

Cloudera Data Science Workbench uses HTTP and WebSockets (WS) to support interactive connections to the Cloudera Data Science Workbench web application. However, these connections are not secure by default. This topic describes how you can use [TLS/SSL](#) to enforce secure encrypted connections, using HTTPS and WSS (WebSockets over TLS), to the Cloudera Data Science Workbench web application.

Starting with version 1.6, Cloudera Data Science Workbench defaults to using TLS 1.2. The default cipher suites have also been upgraded to Mozilla's [Modern](#) cipher suites.

Cloudera Data Science Workbench can be configured to use a TLS termination proxy to handle incoming connection requests. The termination proxy server will decrypt incoming connection requests and forward them to the Cloudera Data Science Workbench web application. A TLS termination proxy can be internal or external.



Note: If you are using an internal custom Certificate Authority, you must add your CA to the "TLS_ROOTCA" configuration. See [Configuring Custom Root CA Certificate](#) on page 289 for further information.

Internal Termination

An internal termination proxy will be run by Cloudera Data Science Workbench's built-in load balancer, called the ingress controller, on the master host. The ingress controller is primarily responsible for routing traffic and load balancing

between Cloudera Data Science Workbench's web service backend. Once configured, as shown in the instructions that follow, it will start terminating HTTPS traffic as well. The primary advantage of internal termination approach is simplicity.

External Termination

External TLS termination can be provided through a number of different approaches. Common examples include:

- Load balancers, such as the AWS Elastic Load Balancer
- Modern firewalls
- Reverse web proxies, such as `nginx`
- VPN appliances supporting TLS/SSL VPN

Organizations that require external termination will often have standardized on single approach for TLS. The primary advantage of this approach is that it allows such organizations to integrate with Cloudera Data Science Workbench without violating their IT department's policies for TLS. For example, with an external termination proxy, Cloudera Data Science Workbench does not need access to the TLS private key.

Load balancers and proxies often require a URL they can ping to validate the status of the web service backend. For instance, you can configure a load balancer to send an HTTP GET request to `/internal/load-balancer/health-ping`. If the response is 200 (OK), that means the backend is healthy. Note that, as with all communication to the web backend from the load balancer when TLS is terminated externally, this request should be sent over HTTP and not HTTPS.

Note that any terminating load balancer must provide the following header fields so that Cloudera Data Science Workbench can detect the IP address and protocol used by the client:

- X-Forwarded-For (client's IP address),
- X-Forwarded-Proto (client's requested protocol, i.e. HTTPS),
- X-Forwarded-Host (the "Host" header of the client's original request).

See [Configuring HTTP Headers for Cloudera Data Science Workbench](#) on page 300 for more details on how to customize HTTP headers required by Cloudera Data Science Workbench.

Related topic: [Troubleshooting TLS/SSL Errors](#) on page 308

Private Key and Certificate Requirements

The TLS certificate issued by your CA must list both, the Cloudera Data Science Workbench, as well as a wildcard for all first-level subdomains. For example, if the Cloudera Data Science Workbench domain is `cdsw.company.com`, then the TLS certificate must include both `cdsw.company.com` and `*.cdsw.company.com`.

Creating a Certificate Signing Request (CSR) and Key/Certificate Pair

Use the following steps to create a Certificate Signing Request (CSR) to submit to your CA. Then, create a private key/certificate pair that can be used to authenticate incoming communication requests to Cloudera Data Science Workbench.



Important: Make sure you use `openssl`, and not `keytool`, to perform these steps. Keytool does not support a wildcard Subject Alternative Name (SAN) and cannot create flat files.

1. Create a `cdsw.cnf` file and populate it with the required configuration parameters including the SAN field values.

```
vi cdsw.cnf
```

2. Copy and paste the default `openssl.cnf` from: <http://web.mit.edu/crypto/openssl.cnf>.
3. Modify the following sections and save the `cdsw.cnf` file:

```
[ CA_default ]
default_md = sha2
```

```
[ req ]
default_bits = 2048
distinguished_name = req_distinguished_name
req_extensions = req_ext

[ req_distinguished_name ]
countryName = Country Name (2 letter code)
stateOrProvinceName = State or Province Name (full name)
localityName = Locality Name (eg, city)
organizationName = Organization Name (eg, company)
commonName = Common Name (e.g. server FQDN or YOUR name)

[ req_ext ]
subjectAltName = @alt_names

[alt_names]
DNS.1 = *.cdsw.company.com
DNS.2 = cdsw.company.com
```

Key points to note:

- The domains set in the `DNS.1` and `DNS.2` entries above must match the `DOMAIN` set in `cdsw.conf`.
- The `default_md` parameter must be set to `sha256` at a minimum. Older hash functions such as SHA1 are deprecated and will be rejected by browsers, either currently or in the very near future.
- The `commonName` (CN) parameter will be ignored by browsers. You must use Subject Alternative Names.

4. Run the following command to generate the CSR.

```
openssl req -out cert.csr -newkey rsa:2048 -nodes -keyout private.key -config cdsw.cnf
```

This command generates the private key and the CSR in one step. The `-nodes` switch disables encryption of the private key (which is not supported by Cloudera Data Science Workbench at this time).

5. Use the CSR and private key generated in the previous step to request a certificate from the CA. If you have access to your organization's internal CA or PKI, use the following command to request the certificate. If you do not have access, or are using a third-party/commercial CA, use your organization's respective internal process to submit the request.

```
openssl x509 -req -days 365 -in cert.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out <your_tls_cert>.crt -sha256 -extfile cdsw.cnf -extensions req_ext
```

6. Run the following command to verify that the certificate issued by the CA lists both the required domains, `cdsw.company.com` and `*.cdsw.company.com`, under X509v3 Subject Alternative Name.

```
openssl x509 -in <your_tls_cert>.crt -noout -text
```

You should also verify that a valid hash function is being used to create the certificate. For SHA-256, the value under Signature Algorithm will be `sha256WithRSAEncryption`.

Configuring Internal Termination

Depending on your deployment (CSD or RPM), use one of the following sets of instructions to configure internal termination.

CSD Deployments

To enable internal termination, configure the following properties in the CDSW service in Cloudera Manager.

1. Log in to the Cloudera Manager Admin Console.
2. Navigate to the CDSW service and click **Configuration**.
3. Search for the following properties and configure as required.

- **Enable TLS** - When enabled, this property enforces HTTPS and WSS connections. The server will now redirect any HTTP request to HTTPS and generate URLs with the appropriate protocol.
- **TLS Key for Internal Termination** - Set to the path of the TLS private key.
- **TLS Certificate for Internal Termination** - Set to the path of the TLS certificate.

Certificates and keys must be in PEM format.

4. Click **Save Changes**.
5. Restart the CDSW service.

RPM Deployments

To enable internal termination, configure the following properties in `cdsw.conf` (on all Cloudera Data Science Workbench hosts).

- `TLS_ENABLE` - When set to `true`, this property enforces HTTPS and WSS connections. The server will now redirect any HTTP request to HTTPS and generate URLs with the appropriate protocol.
- `TLS_KEY` - Set to the path of the TLS private key.
- `TLS_CERT` - Set to the path of the TLS certificate.

Certificates and keys must be in PEM format.

You can configure these properties either as part of the [installation process](#) or after the fact. If you make any changes to `cdsw.conf` after installation is complete, make sure to [restart the master and worker hosts as needed](#).

Configuring External Termination

Depending on your deployment (CSD or RPM), use one of the following sets of instructions to configure external termination.

CSD Deployments

To enable external termination, configure the following property in the CDSW service in Cloudera Manager.

1. Log in to the Cloudera Manager Admin Console.
2. Navigate to the CDSW service and click **Configuration**.
3. Search for the following properties and configure as required.
 - **Enable TLS** - When enabled, this property enforces HTTPS and WSS connections. The server will now redirect any HTTP request to HTTPS and generate URLs with the appropriate protocol.The **TLS Key for Internal Termination** and **TLS Certificate for Internal Termination** properties must be left blank.

4. Click **Save Changes**.
5. Restart the CDSW service.

RPM Deployments

To enable external termination, configure the following property in `cdsw.conf` (on all Cloudera Data Science Workbench hosts).

- `TLS_ENABLE` - When set to `true`, this property enforces HTTPS and WSS connections. The server will now redirect any HTTP request to HTTPS and generate URLs with the appropriate protocol.

The `TLS_KEY` and `TLS_CERT` properties must be left blank.

You can configure this property either as part of the [installation process](#) or after the fact. If you make any changes to `cdsw.conf` after installation is complete, make sure to [restart the master and worker hosts as needed](#).

Known Issues and Limitations

- Communication within the Cloudera Data Science Workbench cluster is not encrypted.
- Cloudera Data Science Workbench does not support encrypted private keys with internal TLS termination. If you require an encrypted private key, use external TLS termination with a terminating proxy that does support encrypted private keys.
- Troubleshooting can be difficult because browsers do not typically display helpful security errors with WebSockets. Often they will just silently fail to connect.
- **Self-signed certificates**

In general, browsers do not support self-signed certificates for WSS. Your certificate must be signed by a Certificate Authority (CA) that your users' browsers will trust. Cloudera Data Science Workbench will not function properly if browsers silently abort WebSockets connections.

If you are using a TLS certificate that has been used to sign itself, and is not signed by a CA in the trust store, then the browser will display a dialog asking if you want to trust the certificate provided by Cloudera Data Science Workbench. This means you are using a self-signed certificate, which is *not supported and will not work*. In this case WSS connections will likely be aborted silently, regardless of your response (Ignore/Accept) to the dialog.

As long as you have a TLS certificate signed by a CA certificate in the trust store, it will be supported and will work with Cloudera Data Science Workbench. For example, if you need to use a certificate signed by your organization's internal CA, make sure that all your users import your root CA certificate into their machine's trust store. This can be done using the Keychain Access application on Macs or the Microsoft Management Console on Windows.

Configuring Custom Root CA Certificate

If your organization uses its own custom Certificate Authority, CDSW engines will not be able to automatically recognise the custom CA's root certificate. This topic describes how to add your internal root CA certificate to CDSW so that it is inserted into the engine's root certificate store every time a session (or any workload) is launched. This will allow processes inside the engine to communicate securely with the ingress controller.



Note: You can also make CDSW aware of your internal root CA by pointing the path to your custom root CA certificate using the property `TLS_ROOTCA` in your CDSW configuration. See [Configuring cdsw.conf properties](#). This ensures that when CDSW starts, the db-migrate pod will copy the contents of your rootCA to **CDSW web UI > Admin > Root CA Configuration** which can then be inserted into the engine's root certificate store every time a session (or any workload) is launched.

1. Log in to CDSW as a site administrator.
2. Go to **Admin > Security**.
3. Under the **Root CA Configuration** section, paste in the contents of your organization's internal root CA certificate file.

The contents of the certificate should remain in `.CRT` format. For example:

```
---BEGIN CERTIFICATE---
XXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXX
...
---END CERTIFICATE---
```

The contents of this field are then inserted into the engine's root certificate store every time a session (or any workload) is launched. This allows processes inside the engine to communicate with the ingress controller.

4. Click **Update**.

Restart any existing sessions and re-build any existing models to ensure that the newly launched engines pick up this change.

Configuring Cloudera Data Science Workbench Deployments Behind a Proxy

If your deployment is behind an HTTP or HTTPS proxy, you must configure the hostname of the proxy you are using in Cloudera Data Science Workbench as follows.

```
HTTP_PROXY="<http://proxy_host>:<proxy-port>"
HTTPS_PROXY="<http://proxy_host>:<proxy-port>"
```

Depending on your deployment, use one of the following methods to configure the proxy in Cloudera Science Workbench:

- **CSD** - Set the **HTTP Proxy** or **HTTPS Proxy** properties in the Cloudera Manager's CDSW service.
- **RPM** - Set the `HTTP_PROXY` or `HTTPS_PROXY` properties in `/etc/cdsw/config/cdsw.conf` on *all* Cloudera Data Science Workbench gateway hosts.

Intermediate Proxy: If you are using an intermediate proxy such as [Cntlm](#) to handle NTLM authentication, add the Cntlm proxy address to these fields.

```
HTTP_PROXY="http://localhost:3128"
HTTPS_PROXY="http://localhost:3128"
```

Supporting a TLS-Enabled Proxy Server:

If the proxy server uses TLS encryption to handle connection requests, you will need to add the proxy's root CA certificate to your host's store of trusted certificates. This is because proxy servers typically sign their server certificate with their own root certificate. Therefore, any connection attempts will fail until the Cloudera Data Science Workbench host trusts the proxy's root CA certificate. If you do not have access to your proxy's root certificate, contact your Network / IT administrator.

To enable trust, perform the following steps on the master and worker hosts.

1. Copy the proxy's root certificate to the trusted CA certificate store (`ca-trust`) on the Cloudera Data Science Workbench host.

```
cp /tmp/<proxy-root-certificate>.cert /etc/pki/ca-trust/source/anchors/
```

2. Use the following command to rebuild the trusted certificate store.

```
update-ca-trust extract
```

3. If you will be using [custom engine images](#) that will be pulled from a Docker repository, add the proxy's root certificates to a directory under `/etc/docker/certs.d`. For example, if your Docker repository is at `docker.repository.mycompany.com`, create the following directory structure:

```
/etc/docker/certs.d
|-- docker.repository.mycompany.com          # Directory named after Docker repository
    |-- <proxy-root-certificate>.cert        # Docker-related root CA certificates
```

This step is not required with the standard engine images because they are included in the Cloudera Data Science Workbench RPM.

4. Re-initialize Cloudera Data Science Workbench to have this change go into effect.

```
cdsw start
```

Configure Hostnames to be Skipped from the Proxy

Starting with version 1.4, if you have defined a proxy in the `HTTP_PROXY(S)` or `ALL_PROXY` properties, Cloudera Data Science Workbench automatically appends the following list of IP addresses to the `NO_PROXY` configuration. Note that this is the minimum required configuration for this field.

```
"127.0.0.1,localhost,100.66.0.1,100.66.0.2,100.66.0.3,
100.66.0.4,100.66.0.5,100.66.0.6,100.66.0.7,100.66.0.8,100.66.0.9,
100.66.0.10,100.66.0.11,100.66.0.12,100.66.0.13,100.66.0.14,
100.66.0.15,100.66.0.16,100.66.0.17,100.66.0.18,100.66.0.19,
100.66.0.20,100.66.0.21,100.66.0.22,100.66.0.23,100.66.0.24,
100.66.0.25,100.66.0.26,100.66.0.27,100.66.0.28,100.66.0.29,
100.66.0.30,100.66.0.31,100.66.0.32,100.66.0.33,100.66.0.34,
100.66.0.35,100.66.0.36,100.66.0.37,100.66.0.38,100.66.0.39,
100.66.0.40,100.66.0.41,100.66.0.42,100.66.0.43,100.66.0.44,
100.66.0.45,100.66.0.46,100.66.0.47,100.66.0.48,100.66.0.49,
100.66.0.50,100.77.0.10,100.77.0.128,100.77.0.129,100.77.0.130,
100.77.0.131,100.77.0.132,100.77.0.133,100.77.0.134,100.77.0.135,
100.77.0.136,100.77.0.137,100.77.0.138,100.77.0.139"
```

This list includes `127.0.0.1`, `localhost`, and any private Docker registries and HTTP services inside the firewall that Cloudera Data Science Workbench users might want to access from the engines.

To configure any additional hostnames that should be skipped from the proxy, use one of the following methods depending on your deployment:

- On a **CSD** deployment, use the Cloudera Manager CDSW service's **No Proxy** property to specify a comma-separated list of hostnames.
- On an **RPM** deployment, configure the `NO_PROXY` field in `cdsw.conf` on *all* Cloudera Data Science Workbench hosts.

Hadoop Authentication with Kerberos for Cloudera Data Science Workbench

Cloudera Data Science Workbench users can authenticate themselves using Kerberos against the cluster KDC defined in the host's `/etc/krb5.conf` file. Cloudera Data Science Workbench does not assume that your Kerberos principal is always the same as your login information. Therefore, you will need to make sure Cloudera Data Science Workbench knows your Kerberos identity when you sign in.

To authenticate against your cluster's Kerberos KDC, go to the top-right dropdown menu, click **Account settings** > **Hadoop Authentication**, and enter your Kerberos principal. To authenticate, either enter your password or click **Upload Keytab** to upload the keytab file directly to Cloudera Data Science Workbench. Once successfully authenticated, Cloudera Data Science Workbench uses your stored credentials to ensure that you are secure when running your workloads.

When you authenticate with Kerberos, Cloudera Data Science Workbench will store your keytab in an internal database. When you subsequently run an engine, the keytab is used by a Cloudera Data Science Workbench sidecar container to generate ticket-granting tickets for use by your code. Ticket-granting tickets allow you to access resources such as Spark, Hive, and Impala, on Kerberized CDH clusters.

While you can view your current ticket-granting ticket by typing `klist` in an engine terminal, there is no way for you or your code to view your keytab. This prevents malicious code and users from stealing your keytab.

**Important:**

- **(New in 1.6.1)** CDSW 1.6.1 fixes an issue where setting `HADOOP_USER_NAME` to the CDSW username had certain unintended consequences. This fix now sets `HADOOP_USER_NAME` to the first part of the Kerberos principal in kerberized environments. In non-kerberized environments, it is still set to the CDSW username.
- If the `/etc/krb5.conf` file is not available on all Cloudera Data Science Workbench hosts, authentication will fail.
- If you do not see the **Hadoop Authentication** tab, make sure you are accessing your personal account's settings from the top right menu. If you have selected a team account, the tab will not be visible when accessing the Team Settings from the left sidebar.
- When you upload a Kerberos keytab to authenticate yourself to the CDH cluster, Cloudera Data Science Workbench might display a fleeting error message ('cancelled') in the bottom right corner of the screen, even if authentication was successful. This error message can be ignored.

UI Behavior for Non-Kerberized Clusters

The contents of the **Hadoop Authentication** tab change depending on whether the cluster is kerberized. For a secure cluster with Kerberos enabled, the **Hadoop Authentication** tab displays a **Kerberos** section with fields to enter your Kerberos principal and username. However, if Cloudera Data Science Workbench cannot detect a `krb5.conf` file on the host, it will assume the cluster is not kerberized, and the **Hadoop Authentication** tab will display **Hadoop Username Override** configuration instead.

For a non-kerberized cluster, by default, your Hadoop username will be set to your Cloudera Data Science Workbench username. To override this default and set an alternative `HADOOP_USER_NAME`, go to the **Hadoop Username Override** setting at **Account settings > Hadoop Authentication**.

If the **Hadoop Authentication** tab is incorrectly displaying Kerberos configuration fields for a non-kerberized cluster, make sure the `krb5.conf` file is not present on the host running Cloudera Data Science Workbench. If you do find any instances of `krb5.conf` on the host, depending on your deployment, perform one of the following sets of actions:

- On CSD deployments, go to Cloudera Manager and stop the CDSW service. Remove the `krb5.conf` file(s) from the Cloudera Data Science Workbench gateway host, and then start the CDSW in Cloudera Manager.
- OR
- On RPM deployments, run `cdsw stop`, remove the `krb5.conf` file(s) from the Cloudera Data Science Workbench gateway host, and run `cdsw start`.

You should now see the expected **Hadoop Username Override** configuration field.

Limitations

- Cloudera Data Science Workbench does not support the use of [Kerberos plugin modules](#) in `krb5.conf`.

Configure FreeIPA

In addition to MIT Kerberos and Active Directory, Cloudera Data Science Workbench also supports FreeIPA as an identity management system. However, this support comes with one major caveat: if your Kerberos configuration file (`/etc/krb5.conf`) contains references to any external files that reside on the host operating system, Kerberos authentication could fail. This is because those files will not automatically be mounted into the engines where Cloudera Data Science Workbench runs workloads. As a result, any utilities or plugins referenced in this manner will not work.

Therefore, to enable FreeIPA support you must perform the following steps.

1. **Modify `krb5.conf` to remove references to external files**

You do not need to edit the `krb5.conf` file on the host operating system. Instead, make a copy of the file, and make your changes there. Points to note:

include and includedir directives

While the `include` and `includedir` directives do typically reference external files, CDSW does account for those directives. Therefore, they are safe to use and no changes need to be made here.

[plugins] directives

The `[plugins]` will always refer to a shared library on the host, which will not be available inside engines. An example of this is:

```
[plugins]
localauth = {
module = sssd:/usr/lib64/sss/modules/sss_krb5_localauth_plugin.so
}
```

You must remove the entire `[plugins]` section from `krb5.conf`. It is not needed for the commands used by Cloudera Data Science Workbench.

PKINIT

The [PKINIT](#) option can also point to external files which will not exist by default in CDSW engines. An example of such a configuration is:

```
[libdefaults]
EXAMPLE.COM = {
    pkinit_anchors = FILE:/usr/local/example.com.crt
}
```

If the realm that uses PKINIT is not one that CDSW users will need a keytab for, it can be removed from the `krb5.conf` file. Otherwise, users will need to create a keytab outside of CDSW and upload it to the **Settings > Hadoop Authentication** page.

default_ccache_name directive

A `default_ccache_name` using the Linux-specific `KEYRING` directive does not work with Cloudera Data Science Workbench. An example of this line is:

```
default_ccache_name = KEYRING:persistent:%
```

You must remove this line from the `krb5.conf` file; the default value will work properly inside CDSW engines.



Note: If you require other configurations of `krb5.conf` that have not been discussed here, the recommended alternative method is to manually upload a keytab on the **User Settings > Hadoop Authentication** page in the Cloudera Data Science Workbench web UI.

2. Copy the contents of `krb5.conf` to the Site Administration panel

1. Log into Cloudera Data Science Workbench as a site administrator.
2. Click **Admin > Security**.
3. Copy the contents of the modified `krb5.conf` from Step 1 to the **Kerberos Configuration** text box. Click **Update**.

The contents of this text box will now be used as the `krb5.conf` file in engines launched for user workloads.

Configuring External Authentication with LDAP and SAML

Cloudera Data Science Workbench supports user authentication against its internal local database, and against external services such as Active Directory, OpenLDAP-compatible directory services, and SAML 2.0 Identity Providers. By default,

Cloudera Data Science Workbench performs user authentication against its internal local database. This topic describes the signup process for the first user, how to configure authentication using LDAP, Active Directory or SAML 2.0, and an optional workaround that allows site administrators to bypass external authentication by logging in using the local database in case of misconfiguration.

User Signup Process

The first time you visit the Cloudera Data Science Workbench web console, the first account that you sign up with is a local administrator account. If in the future you intend to use external services for authentication, Cloudera recommends you use exclusive username & email combinations, rather than site administrators' work email addresses, for both the first site administrator account, and any other local accounts created before switching to external authentication. If the username/email combinations are not unique, an email address might end up being associated with different usernames, one for the external authentication service provider and one for a local Cloudera Data Science Workbench account. This will prevent the user from logging into Cloudera Data Science Workbench with their credentials for the external authentication service.

The link to the signup page is only visible on the login page when the authentication type is set to `local`. When you enable external services for authentication, signing up through the local database is disabled, and user accounts are automatically created upon their first successful login.

Optionally, site administrators can use a **Require invitation to sign up** flag under the **Admin > Settings** tab to require invitation tokens for account creation. When this flag is enabled, only users that are invited by site administrators can login to create an account, regardless of the authentication type.



Important: If you forget the original credentials, or make a mistake with LDAP or SAML configuration, you can use the workaround described in [Debug Login URL](#) on page 299.

When you switch to using external authentication methods such as LDAP or SAML 2.0, user accounts will be automatically created upon each user's first successful login. Cloudera Data Science Workbench will extract user attributes such as username, email address and full name from the authentication responses received from the LDAP server or SAML 2.0 Identity Provider and use them to create the user accounts.

Configuring LDAP/Active Directory Authentication

Cloudera Data Science Workbench supports both search bind and direct bind operations to authenticate against an LDAP or Active Directory directory service. The search bind authentication mechanism performs an `ldapssearch` against the directory service, and binds using the found [Distinguished Name \(DN\)](#) and password provided. The direct bind authentication mechanism binds to the LDAP server using a username and password provided at login.

You can configure Cloudera Data Science Workbench to use external authentication methods by clicking the **Admin** link on the left sidebar and selecting the **Security** tab. Select **LDAP** from the list to start configuring LDAP properties.

LDAP General Settings

- **LDAP Server URI:** Required. The URI of the LDAP/Active Directory server against which Cloudera Data Science Workbench should authenticate. For example, `ldaps://ldap.company.com:636`.
- **Use Direct Bind:** If checked, the username and password provided at login are used with the LDAP Username Pattern for binding to the LDAP server. If unchecked, Cloudera Data Science Workbench uses the search bind mechanism and two configurations, LDAP Bind DN and LDAP Bind Password, are required to perform the `ldapssearch` against the LDAP server.
- **LDAP Bind DN:** Required when using search bind. The DN to bind to for performing `ldapssearch`. For example, `cn=admin,dc=company,dc=com`.
- **LDAP Bind Password:** Required when using search bind. This is the password for the LDAP Bind DN.
- **LDAP Search Base:** Required. The base DN from which to search for the provided LDAP credentials. For example, `ou=Engineering,dc=company,dc=com`.
- **LDAP User Filter:** Required. The [LDAP filter](#) for searching for users. For example, `(&(sAMAccountName={0})(objectclass=person))`. The `{0}` placeholder will be replaced with the username provided at login.

- **LDAP User Username Attribute:** Required. The case-sensitive username attribute of the LDAP directory service. This is used by Cloudera Data Science Workbench to perform the bind operation and extract the username from the response. Common values are `uid`, `sAMAccountName`, or `userPrincipalName`.

General Settings

LDAP Server URI *

Use Direct Bind

By default, Cloudera Data Science Workbench searches for the users by binding to the provided **LDAP Bind DN** and **LDAP Bind Password**. When checked, Cloudera Data Science Workbench will attempt to search for the user by binding to the user-provided username and password. In such case, please make sure the users have permissions to search for themselves in the LDAP server.

LDAP Bind DN *

Update LDAP Bind Password

LDAP User Search Base *

LDAP User Search Filter

LDAP User Username Attribute *

When you select **Use Direct Bind**, Cloudera Data Science Workbench performs a direct bind to the LDAP server using the LDAP Username Pattern with the credentials provided on login (not **LDAP Bind DN** and **LDAP Bind Password**).

By default, Cloudera Data Science Workbench performs an LDAP search using the bind DN and credentials specified for the **LDAP Bind DN** and **LDAP Bind Password** configurations. It searches the subtree, starting from the base DN specified for the **LDAP Search Base** field, for an entry whose attribute specified in **LDAP User Username Attribute**, has the same value as the username provided on login. Cloudera Data Science Workbench then validates the user-provided password against the DN found as a result of the search.

LDAP Over SSL (LDAPS)

To support secure communication between Cloudera Data Science Workbench and the LDAP/Active Directory server, Cloudera Data Science Workbench might require a CA certificate to be able to validate the identity of the LDAP/Active Directory service.

- **CA Certificate:** If the certificate of your LDAP/Active Directory service was signed by a trusted or commercial Certificate Authority (CA), it is not necessary to upload the CA certificate here. However, if your LDAP/Active Directory certificate was signed by a self-signed CA, you must upload the self-signed CA certificate to Cloudera Data Science Workbench in order to use LDAP over SSL (LDAPS).

LDAP Group Settings

In addition to the general LDAP settings, you can use the following group settings to restrict the access to Cloudera Data Science Workbench to certain groups in LDAP:

- **LDAP Group Search Base:** The base distinguished name (DN) where Cloudera Data Science Workbench will search for groups.
- **LDAP Group Search Filter:** The LDAP filter that Cloudera Data Science Workbench will use to determine whether a user is affiliated to a group.

A group object in LDAP or Active Directory typically has one or more **member** attributes that stores the DNs of users in the group. If **LDAP Group Search Filter** is set to **member={0}**, Cloudera Data Science Workbench will automatically substitute the **{0}** placeholder for the DN of the authenticated user.

- **LDAP User Groups:** A list of LDAP groups whose users have access to Cloudera Data Science Workbench. When this property is set, only users that successfully authenticate themselves AND are affiliated to at least one of the groups listed here, will be able to access Cloudera Data Science Workbench.

If this property is left empty, all users that can successfully authenticate themselves to LDAP will be able to access Cloudera Data Science Workbench.

- **LDAP Full Administrator Groups:** A list of LDAP groups whose users are automatically granted the [site administrator](#) role on Cloudera Data Science Workbench.

The groups listed under **LDAP Full Administrator Groups** do *not* need to be listed again under the **LDAP User Groups** property.

If you want to restrict access to Cloudera Data Science Workbench to members of a group whose DN is:

```
CN=CDSWUsers , OU=Groups , DC=company , DC=com
```

And automatically grant site administrator privileges to members of a group whose DN is:

```
CN=CDSWAdmins , OU=Groups , DC=company , DC=com
```

Add the CNs of both groups to the following settings in Cloudera Data Science Workbench:

- **LDAP User Groups:** CDSWUsers
- **LDAP Full Administrator Groups:** CDSWAdmins

Figure 5: Example

How Login Works with LDAP Group Settings Enabled

With LDAP Group settings enabled, the login process in Cloudera Data Science Workbench works as follows:

1. Authentication with LDAP

When an unauthenticated user first accesses Cloudera Data Science Workbench, they are sent to the login page where they can login by providing a username and password.

Cloudera Data Science Workbench will search for the user by binding to the LDAP Bind DN and verify the username/password credentials provided by the user.

2. Authorization Check for Access to Cloudera Data Science Workbench

If the user is authenticated successfully, Cloudera Data Science Workbench will then use the **LDAP Group Search Filter** to search for all groups the user is affiliated to, in the DN provided by **LDAP Group Search Base**.

The list of LDAP groups the user belongs to is then compared to the pre-authorized lists of groups specified in the **LDAP User Groups** and **LDAP Full Administrator Groups** properties.

If there is a match with a group listed under **LDAP User Groups**, this user will be allowed to access Cloudera Data Science Workbench as a regular user.

If there is a match with a group listed under **LDAP Full Administrator Groups**, this user will be allowed to access Cloudera Data Science Workbench as a site administrator.

Test LDAP Configuration

You can test your LDAP/Active Directory configuration by entering your username and password in the **Test LDAP Configuration** section. This form simulates the user login process and allows you to verify the validity of your LDAP/Active Directory configuration without opening a new window.

Before using this form, make sure you click **Update** to save the LDAP configuration you want to test.

Configuring SAML Authentication

Cloudera Data Science Workbench supports the [Security Assertion Markup Language \(SAML\)](#) for [Single Sign-on \(SSO\)](#) authentication; in particular, between an identity provider (IDP) and a service provider (SP). The SAML specification defines three roles: the principal (typically a user), the IDP, and the SP. In the use case addressed by SAML, the principal (user agent) requests a service from the service provider. The service provider requests and obtains an identity assertion from the IDP. On the basis of this assertion, the SP can make an access control decision—in other words it can decide whether to perform some service for the connected principal.

The primary SAML use case is called web browser single sign-on (SSO). A user with a user agent (usually a web browser) requests a web resource protected by a SAML SP. The SP, wanting to know the identity of the requesting user, issues an authentication request to a SAML IDP through the user agent. In the context of this terminology, Cloudera Data Science Workbench operates as a SP.

Cloudera Data Science Workbench supports both SP- and IDP-initiated SAML 2.0-based SSO. Its [Assertion Consumer Service \(ACS\)](#) API endpoint is for consuming assertions received from the Identity Provider. If your Cloudera Data Science Workbench domain root were `cdsw.company.com`, then this endpoint would be available at `http://cdsw.company.com/api/v1/saml/acs`. SAML 2.0 metadata is available at `http://cdsw.company.com/api/v1/saml/metadata` for IDP-initiated SSO. Cloudera Data Science Workbench uses [HTTP Redirect Binding](#) for authentication requests and expects to receive responses from [HTTP POST Binding](#).

When Cloudera Data Science Workbench receives the SAML responses from the Identity Provider, it expects to see at least the following user attributes in the SAML responses:

- The unique identifier or username. Valid attributes are:
 - `uid`
 - `urn:oid:0.9.2342.19200300.100.1.1`
- The email address. Valid attributes are:
 - `mail`
 - `email`
 - `urn:oid:0.9.2342.19200300.100.1.3`
- The common name or full name of the user. Valid attributes are:
 - `cn`
 - `urn:oid:2.5.4.3`

In the absence of the `cn` attribute, Cloudera Data Science Workbench will attempt to use the following user attributes, if they exist, as the full name of the user:

- The first name of the user. Valid attributes are:
 - `givenName`
 - `urn:oid:2.5.4.42`
- The last name of the user. Valid attributes are:
 - `sn`
 - `urn:oid:2.5.4.4`

Configuration Options

Use the following properties to configure SAML authentication and authorization in Cloudera Data Science Workbench. For an overview of the login process, see [How Login Works with SAML Group Settings Enabled](#) on page 299.

Cloudera Data Science Workbench Settings

- **Entity ID:** Required. A globally unique name for Cloudera Data Science Workbench as a Service Provider. This is typically the URI.

- **NameID Format:** Optional. The name identifier format for both Cloudera Data Science Workbench and Identity Provider to communicate with each other regarding a user. Default:
`urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress.`
- **Authentication Context:** Optional. [SAML authentication context](#) classes are URIs that specify authentication methods used in SAML authentication requests and authentication statements. Default:
`urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport.`

Signing SAML Authentication Requests

- **CDSW Private Key for Signing Authentication Requests:** Optional. If you upload a private key, you must upload a corresponding certificate as well so that the Identity Provider can use the certificate to verify the authentication requests sent by Cloudera Data Science Workbench. You can upload the private key used for both signing authentication requests sent to Identity Provider and decrypting assertions received from the Identity Provider.
- **CDSW Certificate for Signature Validation:** Required if the Cloudera Data Science Workbench Private Key is set, otherwise optional. You can upload a certificate in the [PEM format](#) for the Identity Provider to [verify the authenticity](#) of the authentication requests generated by Cloudera Data Science Workbench. The uploaded certificate is made available at the `http://cdsw.company.com/api/v1/saml/metadata` endpoint.

SAML Assertion Decryption

Cloudera Data Science Workbench uses the following properties to support SAML assertion encryption & decryption.

- **CDSW Certificate for Encrypting SAML Assertions** - Must be configured on the Identity Provider so that Identity Provider can use it for encrypting SAML assertions for Cloudera Data Science Workbench
- **CDSW Private Key for Decrypting SAML Assertions** - Used to decrypt the encrypted SAML assertions.

Identity Provider

- **Identity Provider SSO URL:** Required. The entry point of the Identity Provider in the form of URI.
- **Identity Provider Logout URL:** Optional. When this URL is provided, and the **Enable SAML Logout** checkbox is enabled, a user clicking the **Sign Out** button on CDSW will also be logged out of the identity provider.
- **Identity Provider Signing Certificate:** Optional. Administrators can upload the [X.509](#) certificate of the Identity Provider for Cloudera Data Science Workbench to validate the incoming SAML responses.

Cloudera Data Science Workbench extracts the Identity Provider SSO URL and Identity Provider Signing Certificate information from the uploaded Identity Provider Metadata file. Cloudera Data Science Workbench also expects all Identity Provider metadata to be defined in a `<md:EntityDescriptor>` XML element with the namespace `"urn:oasis:names:tc:SAML:2.0:metadata"`, as defined in the [SAMLMeta-xsd schema](#).

For on-premises deployments, you must provide a certificate and private key, generated and signed with your trusted Certificate Authority, for Cloudera Data Science Workbench to establish secure communication with the Identity Provider.

- **Enable SAML Logout:** Optional. When this checkbox is enabled, and the **Identity Provider Logout URL** is provided, a user clicking the **Sign Out** button on CDSW will also be logged out of the identity provider. As a result of this, the user might also be logged out from any other services that they authenticate to using the same identity provider. For this feature to work, the identity provider must support Single Logout Service with HTTP-Redirect binding.

Authorization

When you're using SAML 2.0 authentication, you can use the following properties to restrict the access to Cloudera Data Science Workbench to certain groups of users:

- **SAML Attribute Identifier for User Role:** The Object Identifier (OID) of the user attribute that will be provided by your identity provider for identifying a user's role/affiliation. You can use this field in combination with the following **SAML User Groups** property to restrict access to Cloudera Data Science Workbench to only members of certain groups.

For example, if your identity provider returns the `OrganizationalUnitName` user attribute, you would specify the OID of the `OrganizationalUnitName`, which is `urn:oid:2.5.4.11`, as the value for this property.

- **SAML User Groups:** A list of groups whose users have access to Cloudera Data Science Workbench. When this property is set, only users that are successfully authenticated AND are affiliated to at least one of the groups listed here, will be able to access Cloudera Data Science Workbench.

For example, if your identity provider returns the `OrganizationalUnitName` user attribute, add the value of this attribute to the **SAML User Groups** list to restrict access to Cloudera Data Science Workbench to that group.

If this property is left empty, all users that can successfully authenticate themselves will be able to access Cloudera Data Science Workbench.

- **SAML Full Administrator Groups:** A list of groups whose users are automatically granted the [site administrator](#) role on Cloudera Data Science Workbench.

The groups listed under **SAML Full Administrator Groups** do *not* need to be listed again under the **SAML User Groups** property.

How Login Works with SAML Group Settings Enabled

With SAML Group settings enabled, the login process in Cloudera Data Science Workbench works as follows:

1. Authentication by Identity Provider

When an unauthenticated user accesses Cloudera Data Science Workbench, they are first sent to the identity provider's login page, where the user can login as usual.

Once successfully authenticated by the identity provider, the user is sent back to Cloudera Data Science Workbench along with a SAML assertion that includes, amongst other things, a list of the user's attributes.

2. Authorization Check for Access to Cloudera Data Science Workbench

Cloudera Data Science Workbench will attempt to look up the value of the **SAML Attribute Identifier for User Role** in the SAML assertion and check to see whether that value, which could be one or more group names, exists in the **SAML User Groups** and **SAML Full Administrator Groups** whitelists.

If there is a match with a group listed under **SAML User Groups**, this user will be allowed to access Cloudera Data Science Workbench as a regular user.

If there is a match with a group listed under **SAML Full Administrator Groups**, this user will be allowed to access Cloudera Data Science Workbench as a site administrator.

Debug Login URL

When using external authentication, such as LDAP, Active Directory or SAML 2.0, even a small mistake in authentication configurations in either Cloudera Data Science Workbench or the Identity Provider could potentially block all users from logging in.

Cloudera Data Science Workbench provides an optional fallback debug login URL for site administrators to log in against the local database with their username/password created during the signup process before changing the external authentication method. The debug login URL is `http://cdsw.company.com/login?debug=1`. If you do not remember the original password, you can reset it by going directly to `http://cdsw.company.com/forgot-password`. When configured to use external authentication, the link to the forgot password page is disabled on the login page for security reasons.

Disabling the Debug Login Route

Optionally, the debug login route can be disabled to prevent users from accessing Cloudera Data Science Workbench via local database when using external authentication. In case of external authentication failures, when the debug login route is disabled, root access to the master host is required to re-enable the debug login route.

Contact Cloudera Support for more information.

Configuring HTTP Headers for Cloudera Data Science Workbench

Required Role: [Site Administrator](#)

Cloudera Data Science Workbench 1.4.2 (and higher) include three properties that allow you to customize the HTTP headers accepted by Cloudera Data Science Workbench. They are available under the site administrator panel at **Admin > Security**.



Important: Any changes to the following properties require a full restart of Cloudera Data Science Workbench. For CSD deployments, go to Cloudera Manager and restart the CDSW service. For RPM deployments, run `cdsw restart` on the master host.

Enable HTTP Security Headers

When **Enable HTTP security headers** is enabled, the following HTTP headers will be included in HTTP responses from servers:

- X-XSS-Protection
- X-DNS-Prefetch-Control
- X-Frame-Options
- X-Download-Options
- X-Content-Type-Options

This property is **enabled by default**.

Disabling this property could leave your Cloudera Data Science Workbench deployment vulnerable to clickjacking, cross-site scripting (XSS), or any other injection attacks.

Enable HTTP Strict Transport Security (HSTS)



Note: Without TLS/SSL enabled, configuring this property will have no effect on your browser.

When both [TLS/SSL](#) and this property (**Enable HTTP Strict Transport Security (HSTS)**) are enabled, Cloudera Data Science Workbench will inform your browser that it should never load the site using HTTP. Additionally, all attempts to access Cloudera Data Science Workbench using HTTP will automatically be converted to HTTPS.

This property is **disabled by default**.

If you ever need to downgrade to back to HTTP, use the following sequence of steps: First, deactivate this checkbox to disable HSTS and restart Cloudera Data Science Workbench. Then, load the Cloudera Data Science Workbench web application in each browser to clear the respective browser's HSTS setting. Finally, disable TLS/SSL across the cluster. Following this sequence should help avoid a situation where users get locked out of their accounts due to browser caching.

Enable Cross-Origin Resource Sharing (CORS)

Most modern browsers implement the [Same-Origin Policy](#), which restricts how a document or a script loaded from one origin can interact with a resource from another origin. When the **Enable cross-origin resource sharing** property is enabled on Cloudera Data Science Workbench, web servers will include the `Access-Control-Allow-Origin:`

* HTTP header in their HTTP responses. This gives web applications on different domains permission to access the Cloudera Data Science Workbench API through browsers.

This property is **enabled by default**.

If this property is disabled, web applications from different domains will not be able to programmatically communicate with the Cloudera Data Science Workbench API through browsers.

Restricting User-Controlled Kubernetes Pods

Required Role: [Site Administrator](#)

Cloudera Data Science Workbench 1.6.0 (and higher) includes three properties that allow you to control the permissions granted to user-controlled Kubernetes pods. An example of a user-controlled pod is the engine pod, which provides the environment for sessions, jobs, etc. These pods are launched in a per-user Kubernetes namespace. Since the user has the ability to launch arbitrary pods, these settings restrict what those pods can do.

They are available under the site administrator panel at **Admin > Security** under the **Control of User-Created Kubernetes Pods** section.

Do not modify these settings unless you need to run pods that require special privileges. Enabling any of these properties puts CDSW user data at risk.

Allow containers to run as root

Security best practices dictate that engine containers should not run as the root user. Previously, engines (v7 and lower) would briefly initialize as the `root` user and then run as the `cdsw` user. With engine v8 (and higher), engines now follow the best practice and run only as the `cdsw` user.

Use the following sections to determine whether you need to perform any steps to take advantage of this feature.

New Deployments - Version 1.6.0 (or higher)

Cloudera Data Science Workbench 1.6 (and higher) ships with engine v8 (and higher). On such deployments, all projects should already be using the latest engine versions. Therefore, this property should be left **disabled**.

Upgrades from Version 1.5.x (and lower) to 1.6.0 (or higher)

For deployments that have upgraded from Cloudera Data Science Workbench 1.5 (or lower), it is likely that projects on your deployment are still using base engine v7 (or lower). On such deployments, this property will be enabled by default.

Perform the following steps so that you can disable this property to take advantage of the security fix:

1. Upgrade to Cloudera Data Science Workbench 1.6 (or higher).
2. Test and upgrade all projects to engine v8 (or higher). If you are using custom engines, you will need to rebuild these engines using engine v8 or higher as the base image.
3. Go to **Admin > Security**. Under the **Control of User-Created Kubernetes Pods** section, disable the **Allow containers to run as root** checkbox.

Allow "privileged" pod containers

Pod containers that are "privileged" are extraordinarily powerful. Processes within such containers get almost the same privileges that are available to processes outside the container. If this property is enabled, a privileged container could potentially access all data on the host.

This property is **disabled by default**.

Allow pod containers to mount unsupported volume types

The volumes that can be mounted inside a container in a Kubernetes pod are already heavily restricted. Access is normally denied to volume types that are unfamiliar, such as GlusterFS, Cinder, Fibre Channel, etc. If this property is enabled, pods will be able to mount all unsupported volume types.

This property is **disabled by default**.

SSH Keys

This topic describes the different types of SSH keys used by Cloudera Data Science Workbench, and how you can use those keys to authenticate to an external service such as GitHub.

Personal Key

Cloudera Data Science Workbench automatically generates an SSH [key pair](#) for your user account. You can rotate the key pair and view your public key on your user settings page. It is not possible for anyone to view your private key.

Every console you run has your account's private key loaded into its [SSH-agent](#). Your consoles can use the private key to authenticate to external services, such as GitHub. For instructions, see [Adding SSH Key to GitHub](#) on page 302.

Team Key

Like Cloudera Data Science Workbench users, each Cloudera Data Science Workbench team has an associated SSH key. You can access the public key from the team's account settings. Click **Account**, then select the team from the drop-down menu at the upper right corner of the page.

Team SSH keys provide a useful way to give an entire team access to external resources such as databases or GitHub repositories (as described in the next section). When you launch a console in a project owned by a team, you can use that team's SSH key from within the console.

Adding SSH Key to GitHub

If you want to use GitHub repositories to create new projects or collaborate on projects, use the following instructions to add your Cloudera Data Science Workbench SSH public key to your GitHub account:

1. Sign in to Cloudera Data Science Workbench.
2. Go to the upper right drop-down menu and switch context to the account whose key you want to add.
3. On the left sidebar, click **Settings**.
4. Go to the **SSH Keys** tab and copy your public SSH key.
5. Sign in to your GitHub account and add the Cloudera Data Science Workbench key copied in the previous step to your GitHub account. For instructions, refer the GitHub documentation on [adding SSH keys to GitHub](#).

SSH Tunnels



Important: SSH tunnels do not work in Cloudera Data Science Workbench 1.4.0. This issue has been fixed in Cloudera Data Science Workbench 1.4.2 (and higher).

In some environments, external databases and data sources reside behind restrictive firewalls. A common pattern is to provide access to these services using a bastion host with only the SSH port open. This introduces complexity for end users who must manually set up SSH tunnels to access data. Cloudera Data Science Workbench provides a convenient way to connect to such resources.

From the **Project > Settings > Tunnels** page, you can use your SSH key to connect Cloudera Data Science Workbench to an external database or cluster by creating an SSH tunnel. If you create an [SSH tunnel](#) to an external server in one of your projects, then all engines that you run in that project are able to connect securely to a port on that server by connecting to a local port. The encrypted tunnel is completely transparent to the user or code.

To create an automatic SSH tunnel:

1. Open the **Project Settings** page.
2. Open the **Tunnels** tab.
3. Click **New Tunnel**.
4. Enter the server IP Address or DNS hostname.

5. Enter your username on the server.
6. Enter the local port that should be proxied, and to which remote port on the server.

Then, on the remote server, configure SSH to accept password-less logins using your individual or team SSH key. Often, you can do so by appending the SSH key to the file `/home/username/.ssh/authorized_keys`.

Troubleshooting Cloudera Data Science Workbench

Depending on your deployment, use one of the following courses of action to start debugging issues with Cloudera Data Science Workbench.

- **CSD Deployments**

- Check the current status of the application and run validation checks. You can use the [Status and Validate](#) commands in Cloudera Manager to do so.
- Make sure your Cloudera Data Science Workbench configuration is correct. Log into Cloudera Manager and review configuration for the CDSW service.

- **RPM Deployments**

- Check the status of the application.

```
cdsw status
```

- SSH to your master host and run the following host validation command to check that the key services are running:

```
cdsw validate
```

- Make sure your Cloudera Data Science Workbench configuration is correct.

```
cat /etc/cdsw/config/cdsw.conf
```

The following sections describe solutions to potential problems and error messages you may encounter while installing, configuring or using Cloudera Data Science Workbench.

Understanding Installation Warnings

This section describes solutions to some warnings you might encounter during the installation process.

Preexisting iptables rules not supported

```
WARNING: Cloudera Data Science Workbench requires iptables, but does not support preexisting iptables rules.
```

Kubernetes makes extensive use of iptables. However, it's hard to know how pre-existing iptables rules will interact with the rules inserted by Kubernetes. Therefore, Cloudera recommends you run the following commands to clear all pre-existing rules before you proceed with the installation.

```
sudo iptables -F INPUT ACCEPT
sudo iptables -F FORWARD ACCEPT
sudo iptables -F OUTPUT ACCEPT
sudo iptables -t nat -F
sudo iptables -t mangle -F
sudo iptables -F
sudo iptables -X
```

The warning can be ignored after you clear the pre-existing rules or are sure that there are no pre-existing iptables rules.

Remove the entry corresponding to `/dev/xvdc` from `/etc/fstab`

Cloudera Data Science Workbench installs a custom filesystem on its Application and Docker block devices. These filesystems will be used to store user project files and Docker engine images respectively. Therefore, Cloudera Data Science Workbench requires complete access to the block devices. To avoid losing any existing data, make sure the block devices allocated to Cloudera Data Science Workbench are reserved only for the workbench.

Linux `sysctl` kernel configuration errors

Kubernetes and Docker require non-standard kernel configuration. Depending on the existing state of your kernel, this might result in `sysctl` errors such as:

```
sysctl net.bridge.bridge-nf-call-iptables must be set to 1
```

This is because the settings in `/etc/sysctl.conf` conflict with the settings required by Cloudera Data Science Workbench. Cloudera cannot make a blanket recommendation on how to resolve such errors because they are specific to your deployment. Cluster administrators may choose to either remove or modify the conflicting value directly in `/etc/sysctl.conf`, remove the value from the conflicting configuration file, or even delete the module that is causing the conflict.

To start diagnosing the issue, run the following command to see the list of configuration files that are overwriting values in `/etc/sysctl.conf`.

```
SYSTEMD_LOG_LEVEL=debug /usr/lib/systemd/systemd-sysctl
```

You will see output similar to:

```
Parsing /usr/lib/sysctl.d/00-system.conf
Parsing /usr/lib/sysctl.d/50-default.conf
Parsing /etc/sysctl.d/99-sysctl.conf
Overwriting earlier assignment of net/bridge/bridge-nf-call-ip6tables in file
'/etc/sysctl.d/99-sysctl.conf'.
Overwriting earlier assignment of net/bridge/bridge-nf-call-ip6tables in file
'/etc/sysctl.d/99-sysctl.conf'.
Overwriting earlier assignment of net/bridge/bridge-nf-call-ip6tables in file
'/etc/sysctl.d/99-sysctl.conf'.
Parsing /etc/sysctl.d/k8s.conf
Overwriting earlier assignment of net/bridge/bridge-nf-call-iptables in file
'/etc/sysctl.d/k8s.conf'.
Parsing /etc/sysctl.conf
Overwriting earlier assignment of net/bridge/bridge-nf-call-ip6tables in file
'/etc/sysctl.conf'.
Overwriting earlier assignment of net/bridge/bridge-nf-call-ip6tables in file
'/etc/sysctl.conf'.
Setting 'net/ipv4/conf/all/promote_secondaries' to '1'
Setting 'net/ipv4/conf/default/promote_secondaries' to '1'
...
```

`/etc/sysctl.d/k8s.conf` is the configuration added by Cloudera Data Science Workbench. Administrators must make sure that no other file is overwriting values set by `/etc/sysctl.d/k8s.conf`.

CDH parcels not found at `/opt/cloudera/parcels`

There are two possible reasons for this warning:

- If you are using a custom parcel directory, you can ignore the warning and proceed with the installation. Once the Cloudera Data Science Workbench is running, set the path to the CDH parcel in the admin dashboard. See [Configuring the Engine Environment](#) on page 209.
- This warning can be an indication that you have not added gateway roles to the Cloudera Data Science Workbench hosts. In this case, do not ignore the warning. Exit the installer and go to Cloudera Manager to add gateway roles to the cluster. See [Configure Gateway Hosts Using Cloudera Manager](#) on page 83.

Troubleshooting Cloudera Data Science Workbench

CDSW docker daemons fail to start

CDSW docker daemons fail to start with the following error:

```
Error starting daemon: error initializing graphdriver: devmapper: Unable to take ownership of thin-pool (docker-thinpool) that already has used data blocks.
```

This issue occurs when the block devices you specified for the **Docker Block Device** field already have data on them. This is a safeguard to prevent block devices from being wiped inadvertently. Note that resolving this resolving this issue involves deleting data from the block devices.

To resolve this issue, perform the following steps:

1. Verify that it is okay to delete the data on the block device.
2. SSH to the Cloudera Data Science Workbench master host.
3. Run the following script:

```
/opt/cloudera/parcels/CDSW/scripts/teardown-docker.sh
```

4. In the Cloudera Manager Admin Console, select the Cloudera Data Science Workbench service.
5. On the **Instances** tab, select the Docker Daemons.
6. Click **Actions for Selected (n) > Prepare Node**.
7. Start the Cloudera Data Science Workbench service by clicking **Actions > Start**.

User Process Limit

During host validation, you may encounter the following warning message:

```
{WARN} Cloudera Data Science Workbench recommends that all users have a max-user-processes limit of at least 65536.
```

This message appears if the user process limit is under 65536. You can increase the user process limit by adding the following line to the `/etc/security/limits.conf` file:

```
ulimit -u 65536
```

Set this configuration on every Cloudera Data Science Workbench host. You can also edit `/etc/security/limits.conf` to configure the user process limit.

Open Files Limit

During host validation, you may encounter the following warning message:

```
{WARN} Cloudera Data Science Workbench recommends that all users have a max-open-files limit set to 1048576.
```

This message appears if the open files limit is under 1048576. Note that on HDP clusters, the open file limit recommendation is 10000 at a minimum. Cloudera recommends a higher limit for clusters with Cloudera Data Science Workbench.

You can configure the file limit with the following command:

```
ulimit -n 1048576
```

Set this configuration on every Cloudera Data Science Workbench host. You can also edit `/etc/security/limits.conf` to configure the open files limit.

Disable SE Linux

During installation, you may encounter the following message:

```
Please disable SELinux by setting SELINUX=disabled|permissive in /etc/selinux/config,
then reboot or using setenforce 0 command"
```

SELinux enforces additional control policies for what a user, process, or daemon can do. If SELinux is enabled or not in permissive mode, Cloudera Data Science Workbench may not have the proper permissions to run.

To resolve this issue, you must change the SELinux mode on every host by doing one of the following:

- Edit the configuration file for SELinux and set it to disabled or permissive. Note that if you set SELinux to permissive mode, events such as access denials will be logged, but the denial will not be enforced. You can find the SELinux configuration file in the following location: `/etc/selinux/config`.
- Run the following command: `setenforce 0`. This command disables SELinux completely.

DNS is not configured properly

During installation, you might encounter the messages such as:

```
DNS doesn't resolve <CDSW_domain> to <CDSW_Master_IP_address>; DNS is not configured
properly
```

or

```
DNS doesn't resolve <CDSW_Master_IP_address> to <CDSW_domain>; DNS is not configured
properly"
```

This indicates that the CDSW domain name configured does not resolve to the IP address of the Master host. You must enable DNS forward and reverse lookup for the CDSW domain and IP address to proceed.

Error Encountered Trying to Load Images when Initializing Cloudera Data Science Workbench

Here are some sample error messages you might see when initializing Cloudera Data Science Workbench:

```
Error encountered while trying to load images.: 1
```

```
Unable to load images from [/etc/cdsw/images/cdsw_<version>.tar.gz].: 1
```

```
Error processing tar file(exit status 1): write ../../tar: no space left on device
```

These errors are an indication that the root volume is running out of space when trying to initialize Cloudera Data Science Workbench. During the initialization process, the Cloudera Data Science Workbench installer temporarily decompresses the engine image file located in `/etc/cdsw/images` to the `/var/lib/cdsw/docker-tmp/` directory.

If you have previously partitioned the root volume (which should be [at least 100 GB](#)), make sure you allocate at least 20 GB to `/var/lib/cdsw/docker-tmp/` so that the installer can proceed without running out of space.

404 Not Found Error

The 404 Not Found error might appear in the browser when you try to reach the Cloudera Data Science Workbench web application.

Troubleshooting Cloudera Data Science Workbench

This error is an indication that your installation of Cloudera Data Science Workbench was successful, but there was a mismatch in the domain configured in `cdsw.conf` and the domain referenced in the browser. To fix the error, go to `/etc/cdsw/config/cdsw.conf` and check that the URL you supplied for the `DOMAIN` property matches the one you are trying to use to reach the web application. This is the wildcard domain dedicated to Cloudera Data Science Workbench, not the hostname of the master host.

If this requires a change to `cdsw.conf`, after saving the changes run `cdsw stop` followed by `cdsw start`.

Troubleshooting Kerberos Errors

HDFS commands fail with Kerberos errors even though Kerberos authentication is successful in the web application

If Kerberos authentication is successful in the web application, and the output of `klist` in the engine reveals a valid-looking TGT, but commands such as `hdfs dfs -ls /` still fail with a Kerberos error, it is possible that your cluster is missing the [Java Cryptography Extension \(JCE\) Unlimited Strength Jurisdiction Policy File](#). The JCE policy file is required when Red Hat uses AES-256 encryption. This library should be installed on each cluster host and will live under `$JAVA_HOME`. For more information, see [Using AES-256 Encryption](#).

Cannot find renewable Kerberos TGT

Cloudera Data Science Workbench runs its own Kerberos TGT renewer which produces non-renewable TGT. However, this confuses Hadoop's renewer which looks for renewable TGTs. If the Spark 2 logging level is set to `WARN` or lower, you may see exceptions such as:

```
16/12/24 16:38:40 WARN security.UserGroupInformation: Exception encountered while running
the renewal command. Aborting renew thread. ExitCodeException exitCode=1: kinit: Resource
temporarily unavailable while renewing credentials

16/12/24 16:41:23 WARN security.UserGroupInformation: PrivilegedActionException
as:user@CLLOUDERA.LOCAL (auth:KERBEROS) cause:javax.security.sasl.SaslException: GSS
initiate failed [Caused by GSSException: No valid credentials provided (Mechanism level:
Failed to find any Kerberos tgt)]
```

This is *not* a bug. Spark 2 workloads will not be affected by this. Access to Kerberized resources should also work as expected.

Troubleshooting TLS/SSL Errors

This section describes some common issues with TLS configuration on Cloudera Data Science Workbench. Common errors include:

- Cloudera Data Science Workbench initialisation fails with an error such as:

```
Error preparing server: tls: failed to parse private key
```

- Your browser reports that the Cloudera Data Science Workbench web application is not secure even though you have enabled TLS settings as per [Enabling TLS/SSL for Cloudera Data Science Workbench](#) on page 285.

Possible Causes and Solutions

- **Certificate does not include the wildcard domain** - Confirm that the TLS certificate issued by your CA lists both, the Cloudera Data Science Workbench domain, as well as a wildcard for all first-level subdomains. For example, if your Cloudera Data Science Workbench domain is `cdsw.company.com`, then the TLS certificate must include both `cdsw.company.com` and `*.cdsw.company.com`.

- **Path to the private key and/or certificate is incorrect** - Confirm that the path to the private key file is correct by comparing the path and file name to the values for `TLS_KEY` and/or `TLS_CERT` in `cdsw.conf` or Cloudera Manager. For example:

```
TLS_CERT="/path/to/cert.pem"
TLS_KEY="/path/to/private.key"
```

- **Private key file does not have the right permissions** - The private key file must have read-only permissions. Set it as follows:

```
chmod 444 private.key
```

- **Private key is encrypted** - Cloudera Data Science Workbench does not support encrypted private keys. Check to see if your private key is encrypted:

```
cat private.key
```

```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4, ENCRYPTED
DEK-Info: DES-EDE3-CBC,11556F53E4A2824A
```

If the private key is encrypted as shown above, use the following steps to decrypt it:

1. Make a backup of the private key file.

```
mv private.key private.key.encrypted
```

2. Decrypt the backup private key and save the file to `private.key`. You will be asked to enter the private key password.

```
openssl rsa -in private.key.encrypted -out private.key
```

- **Private key and certificate are not related** - Check to see if the private key matches the public key in the certificate.

1. Print a hash of the private key modulus.

```
openssl rsa -in private.key -noout -modulus | openssl md5
```

```
(stdin)= 7a8d72ed61bb4be3c1f59e4f0161c023
```

2. Print a hash of the public key modulus.

```
openssl x509 -in cert.pem -noout -modulus | openssl md5
```

```
(stdin)= 7a8d72ed61bb4be3c1f59e4f0161c023
```

If the `md5` hash output of both keys is different, they are not related to each other, and will not work. You must revoke the old certificate, [regenerate a new private key and Certificate Signing Request \(CSR\)](#), and then apply for a new certificate.

Troubleshooting Issues with Workloads

This section describes some potential issues data scientists might encounter once the application is running workloads.

404 error in Workbench after starting an engine

This is typically caused because a wildcard DNS subdomain was not set up before installation. While the application will largely work, the engine consoles are served on subdomains and will not be routed correctly unless a wildcard DNS entry pointing to the master host is properly configured. You might need to wait 30-60 minutes until the DNS entries propagate. For instructions, see [Set Up a Wildcard DNS Subdomain](#) on page 74.

Engines cannot be scheduled due to lack of CPU or memory

A symptom of this is the following error message in the Workbench: "Unschedulable: No node in the cluster currently has enough CPU or memory to run the engine."

Either shut down some running sessions or jobs or provision more hosts for Cloudera Data Science Workbench.

Workbench prompt flashes red and does not take input

The Workbench prompt flashing red indicates that the session is not currently ready to take input.

Cloudera Data Science Workbench does not currently support non-REPL interaction. One workaround is to skip the prompt using appropriate command-line arguments. Otherwise, consider using the [terminal](#) to answer interactive prompts.

PySpark jobs fail due to HDFS permission errors

```
: org.apache.hadoop.security.AccessControlException: Permission denied: user=alice,
access=WRITE, inode="/user":hdfs:supergroup:drwxr-xr-x
```

(Required for CDH 5 and CDH 6) To be able to use Spark 2, each user must have their own `/home` directory in HDFS. If you sign in to Hue first, these directories will automatically be created for you. Alternatively, you can have cluster administrators create these directories.

```
hdfs dfs -mkdir /user/<username>
hdfs dfs -chown <username>:<username> /user/<username>
```

PySpark jobs fail due to Python version mismatch

```
Exception: Python in worker has different version 2.6 than that in driver 2.7, PySpark
cannot run with different minor versions
```

One solution is to install the matching Python 2.7 version on all the cluster hosts. Another, more recommended solution is to install the Anaconda parcel on all CDH cluster hosts. Cloudera Data Science Workbench Python engines will use the version of Python included in the Anaconda parcel which ensures Python versions between driver and workers will always match. Any library paths in workloads sent from drivers to workers will also match because Anaconda is present in the same location across all hosts. Once the parcel has been installed, set the `PYSPARK_PYTHON` environment variable in the Cloudera Data Science Workbench Admin dashboard. Alternatively, you can [use Cloudera Manager](#) to set the path.

Jobs fail due to incorrect JAVA_HOME on HDP

Commands, such as `hdfs` commands, and jobs fail with an error similar to the following message:

```
ERROR: JAVA_HOME /usr/lib/jvm/java does not exist.
```

The `JAVA_HOME` path you configure for Cloudera Data Science Workbench in `cdsw.conf` must match the `JAVA_HOME` configured by `hadoop-env.sh` for the HDP cluster. After you update `JAVA_HOME` in `cdsw.conf`, you must restart Cloudera Data Science Workbench. For more information, see [Changes to cdsw.conf](#).

The `user_events` table is growing in size and affecting performance

The `user_events` table is used to monitor and audit user events. It can grow in size in long running deployments and can decrease performance. To clean the table manually:

1. SSH to the Cloudera Data Science Workbench Master host and log in as `root`.

```
ssh root@<csw_master_host_domain_name>
```

2. Get the name of the database pod:

```
kubectl get pods -l role=db
```

The command returns information similar to the following example:

NAME	READY	STATUS	RESTARTS	AGE
db-6d56584f76-phn2f	1/1	Running	0	4h46m

3. Enter the following command to log into the database as the `sense` user::

```
kubectl exec -it <database pod> -- psql -U sense
```

4. Run the following query to get the number of rows from the `user_events` table:

```
select count(id) from user_events;
```

5. Delete the records older than 30 days by running the following query:

```
delete from user_events where created_at < NOW() - INTERVAL '30 days';
```

Troubleshooting Issues with Models and Experiments

See the following topics:

- [Debugging Issues with Experiments](#) on page 178
- [Debugging Issues with Models](#) on page 195

Cloudera Data Science Workbench Command Line Reference

This topic describes the commands available to the Cloudera Data Science Workbench command line utility `cdsw` that exists within a Cloudera Data Science Workbench deployment. This utility is meant to manage your Cloudera Data Science Workbench cluster. Running `cdsw` without any arguments will print a brief description of each command.

In addition, there is a `cdswctl` CLI client that offers different functionality that is meant for use by data scientists to manage their sessions. For information about that see [cdswctl Command Line Interface Client](#) on page 313

Start, Stop, Restart for CSD Deployments: The commands available for a CSD-based deployment are only a subset of those available for an RPM deployment. For example, the CLI for CSD deployments does not have commands such as `cdsw start`, `cdsw stop`, and `cdsw restart` available. Instead, these actions must be executed through the Cloudera Data Science Workbench service in Cloudera Manager. For instructions, see [Starting, Stopping, and Restarting the Service](#) on page 257.



Important: On Cloudera Data Science Workbench 1.4.0 (and lower), **do not** stop or restart Cloudera Data Science Workbench without using the [cdsw_protect_stop_restart.sh](#) script. This is to help avoid the data loss issue detailed in [TSB-346](#).

All of the following commands can be used in an RPM-based deployment. Those available for CSD-based deployments have been marked in the table.

Command	Description and Usage
<code>cdsw start</code>	Initializes and bootstraps the master host. Use this command to start Cloudera Data Science Workbench. Also see, Additional Usage Notes on page 313
<code>cdsw stop</code>	De-registers, resets, and stops a host. On a worker host, this command will remove the worker from the cluster. On the master host, this command will bring down the application and effectively tear down the Cloudera Data Workbench deployment.
<code>cdsw restart</code>	Run on the master host to restart application components. To restart a worker host, use <code>cdsw stop</code> , followed by <code>cdsw join</code> . These commands have been explained further in this topic.
<code>cdsw join</code>	Initializes a worker host. After a worker host has been added, run this command on the <i>worker</i> host to add it to the Cloudera Data Science Workbench cluster. This registers the worker hosts with the master, and increases the available pool of resources for workloads.
<code>cdsw status</code>	■ Displays the current status of the application. Starting with version 1.4, you can use <code>cdsw status -v</code> or <code>cdsw status --verbose</code> for more detailed output. The <code>cdsw status</code> command is not supported on worker hosts.
<code>cdsw validate</code>	■ Performs diagnostic checks for common errors that might be preventing the application from running as expected. This command should typically be run as the first step to troubleshooting any problems with the application, as indicated by <code>cdsw status</code> .

Command	Description and Usage
<code>cdsw logs</code>	<ul style="list-style-type: none"> Creates a tarball with diagnostic information for your Cloudera Data Science Workbench installation. If you file a case with Cloudera Support, run this command on each host and attach the resulting bundle to the case. For more details on the information collected in these bundles, see Data Collection in Cloudera Data Science Workbench on page 260.
<code>cdsw version</code>	<ul style="list-style-type: none"> Displays the version number and type of Cloudera Data Science Workbench deployment (RPM or CSD).
<code>cdsw help</code>	<ul style="list-style-type: none"> Displays the inline help options for the Cloudera Data Science Workbench CLI.

Additional Usage Notes



Note: These notes apply only to RPM-based deployments. In case of CSD-based deployments where you cannot directly modify `cdsw.conf`, Cloudera Manager will prompt you if the Cloudera Data Science Workbench service needs to be restarted.

Changes to `cdsw.conf`: Make sure `cdsw.conf` is consistent across all Cloudera Data Science Workbench hosts. Any changes made to the file must be copied over to all the other hosts.

- **Master Host** - Changes to the `JAVA_HOME`, `MASTER_IP`, `DOCKER_BLOCK_DEVICES`, and `APPLICATION_BLOCK_DEVICE` parameters in `cdsw.conf` require a re-initialization of the master host.

```
cdsw stop
cdsw start
```

Changes to other `cdsw.conf` parameters such as domain name changes, or TLS and HTTP proxy changes, require a restart of the application components.

```
cdsw restart
```

- **Worker Host** - Changes to `cdsw.conf` on a worker host, require a restart of the worker host as follows:

```
cdsw stop
cdsw join
```

`cdswctl` Command Line Interface Client

Cloudera Data Science Workbench 1.6 and later ships with a CLI client that you can download from the Cloudera Data Science Workbench web UI. The `cdswctl` client can perform the following tasks:

- Logging in
- Creating an SSH endpoint
- Listing sessions that are starting or running
- Starting or stopping a session

Other actions, such as creating a project, require you to use the Cloudera Data Science Workbench web UI. For information about the available commands, run the following command:

```
cdswctl help
```

Download and Configure the `cdswctl`

Before you begin, ensure that the following prerequisites are met:

- You have an SSH public/private key pair for your local machine.
- You have Contributor permissions for an existing Cloudera Data Science project. Alternatively, create a new project you have access to.
- The Site Administrator has not disabled remote editing for Cloudera Data Science Workbench.

(Optional) Generate an SSH Public/Private Key

This task is optional. If you already have an SSH public/private key pair, skip this task. The steps to create an SSH public/private key pair differ based on your operating system. The following instructions are meant to be an example and are written for macOS using `ssh-keygen`.

1. Open **Terminal**.
2. Run the following command and complete the fields:

```
ssh-keygen -t rsa -f ~/.ssh/id_rsa
```

Keep the following guidelines in mind:

- Make sure that the SSH key you generate meets the requirements for the local IDE you want to use. For example, PyCharm requires the `-m PEM` option because PyCharm does not support modern (RFC 4716) OpenSSH keys.
- Provide a passphrase when you generate the key pair. Use this passphrase when prompted for the SSH key passphrase.
- Save the SSH key to the default `~/.ssh` location.

Download `cdswctl` and Add an SSH Key

1. Open the Cloudera Data Science Workbench web UI and go to **Settings > Remote Editing** for your user account.
2. Download `cdswctl` client for your operating system.

If you are using the macOS executable, `cdswctl` will be unsigned and therefore cannot be launched on the recent version of macOS without performing the following additional steps:

- a) In the Finder on your Mac and locate the app you want to open.

Don't use Launchpad to do this. Launchpad doesn't allow you to access the shortcut menu.

- b) Control-click the app icon, then choose **Open** from the shortcut menu.
- c) Click **Open**.

The app is saved as an exception to your security settings, and you can open it in the future by double-clicking it just as you can any registered app.

3. In the terminal, run `cat ~/.ssh/id_rsa.pub`. If you used a different filename above when generating the key, use that filename instead. This command prints the key as a string.
4. Copy the key. It should resemble the following: `ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCh2J5mW3i3BgtZ25/F0sxywpLVkx1RgmZunI`
5. In **SSH public keys for session access**, paste the key.

Cloudera Data Science Workbench uses the SSH public key to authenticate your CLI client session, including the SSH endpoint connection to the Cloudera Data Science Workbench deployment. Any SSH endpoints that are running when you add an SSH public key must also be restarted.

Initialize an SSH Connection to Cloudera Data Science Workbench

The following task describes how to establish an SSH endpoint for Cloudera Data Science Workbench. Creating an SSH endpoint is the first step to configuring a remote editor for Cloudera Data Science Workbench.

1. Log in to Cloudera Data Science Workbench with the CLI client. Depending on your deployment, make sure you add `http` or `https` to the URL as shown below:

```
cdswctl login -n <username> -u http(s)://cdsw.your_domain.com
```

For example, the following command logs the user `sample_user` into the `https://cdsw.your_domain.com` deployment:

```
cdswctl login -n sample_user -u https://cdsw.your_domain.com
```

2. Create a local SSH endpoint to Cloudera Data Science Workbench. Run the following command:

```
cdswctl ssh-endpoint -p <username>/<project_name> [-c <CPU_cores>] [-m <memory_in_GB>] [-g <number_of_GPUs>]
```

The command uses the following defaults for optional parameters:

- CPU cores: 1
- Memory: 1 GB
- GPUs: 0

For example, the following command starts a session for the logged-in user `sample_user` under the `customerchurn` project with .5 cores, .75 GB of memory, 0 GPUs, and the Python3 kernel:

```
cdswctl ssh-endpoint -p customerchurn -c 0.5 -m 0.75
```

To create an SSH endpoint in a project owned by another user or a team, for example `finance`, prepend the username to the project and separate them with a forward slash:

```
cdswctl ssh-endpoint -p finance/customerchurn -c 0.5 -m 0.75
```

This command creates session in the project `customerchurn` that belongs to the team `finance`.

Information for the SSH endpoint appears in the output:

```
...
You can SSH to it using
    ssh -p <some_port> cdsw@localhost
...
```

3. Open a new command prompt and run the outputted command from the previous step:

```
ssh -p <some_port> cdsw@localhost
```

For example:

```
ssh -p 9750 cdsw@localhost
```

You will be prompted for the passphrase for the SSH key you entered in the Cloudera Data Science web UI.

Once you are connected to the endpoint, you are logged in as the `cdsw` user and can perform actions as though you are accessing the terminal through the Cloudera Data Science Workbench web UI.

4. Test the connection.

If you run `ls`, the project files associated with the session you created are shown. If you run `whoami`, the command returns the `cdsw` user.

Cloudera Data Science Workbench FAQs

Where can I get a sample project to try out Cloudera Data Science Workbench?

Cloudera Data Science Workbench ships with sample project templates that you can use to try running workloads. These are currently available in Python, R, and Scala. See [Create a Project from a Built-in Template](#) on page 117.

What are the software and hardware requirements for Cloudera Data Science Workbench?

For detailed information on the software and hardware required to successfully run Cloudera Data Science Workbench, see [Cloudera Data Science Workbench 1.7.2 Requirements and Supported Platforms](#) on page 65.

Can I run Cloudera Data Science Workbench on hosts shared with other Hadoop services?

No. Cloudera does not support running Cloudera Data Science Workbench on non-dedicated hosts. Running other services on Cloudera Data Science Workbench hosts can lead to unreliable execution of workloads and difficult to debug out-of-memory errors.

How does Cloudera Data Science Workbench use Docker and Kubernetes?

Cloudera Data Science Workbench uses [Docker](#) and [Kubernetes](#) to manage containers. Currently, Cloudera Data Science Workbench only supports the versions of Docker and Kubernetes that are shipped with each release. Upgrading Docker, or Kubernetes, or running on third-party Kubernetes clusters is not supported.

Cloudera does not support Kubernetes or Docker for running any other workloads beyond those on Cloudera Data Science Workbench.

Can I run Cloudera Data Science Workbench on my own Kubernetes cluster?

This is not supported.

Does Cloudera Data Science Workbench support REST API access?

Starting with version 1.1.0, Cloudera Data Science Workbench supports a Jobs REST API that lets you orchestrate jobs from 3rd party workflow tools. For more details, see [Cloudera Data Science Workbench Jobs API](#) on page 200.

Other means of API access to Cloudera Data Science Workbench are not supported at this time.

How do I contact Cloudera for issues regarding Cloudera Data Science Workbench?

Cloudera Support

If you are a Cloudera customer, you can register for an account to create a support ticket at the [support portal](#).

Before you log a support ticket, run the following command on the master host to create a tarball with diagnostic information for your Cloudera Data Science Workbench installation.

```
cdsw logs
```

Attach the resulting bundle to the support case you create.

Cloudera Community

Register for the [Cloudera Community forums](#) and post your questions or feedback on the [Cloudera Data Science Workbench board](#).

Cloudera Data Science Workbench Glossary

Terms related to Cloudera Data Science Workbench:

Cloudera Data Science Workbench

Cloudera Data Science Workbench is a product that enables fast, easy, and secure self-service data science for the enterprise. It allows data scientists to bring their existing skills and tools, such as R, Python, and Scala, to securely run computations on data in Hadoop clusters.

site administrator

A Cloudera Data Science Workbench user with all-access permissions. Site administrators can add or disable users/teams, monitor and manage resource usage, secure access to the deployment, and more. The Site Administration dashboard is only accessible to site administrators.

API

Cloudera Data Science Workbench exposes a limited REST API that allows you to schedule existing jobs from third-party workflow tools.

cluster

Refers to the CDH cluster managed by Cloudera Manager, including the gateway hosts that are running Cloudera Data Science Workbench.

context

Cloudera Data Science Workbench uses the notion of contexts to separate your personal account from any team accounts you belong to. This gives you leave to run experiments in your own personal context, while you can simultaneously collaborate with others in your organization within a team context.

engine

In Cloudera Data Science Workbench, engines are responsible for running R, Python, and Scala code written by users and for facilitating access to the CDH cluster. Each engine functions as an isolated virtual machine, customized to have all the necessary dependencies to access the CDH cluster while keeping each project's environment entirely isolated. The only artifacts that remain after an engine runs is a log of the analysis and any files that were generated or modified inside the project's filesystem, which is mounted to each engine at `/home/cdsw`.

experiment

Experiments are batch executed workloads that help facilitate model training in Cloudera Data Science Workbench.

gateway host

On a Cloudera Manager cluster, a gateway host is one that has been assigned a gateway role for a CDH service. Such a host will receive client configuration for that CDH service even though the host does not have any role instances for that service running on it.

Cloudera Data Science Workbench runs on dedicated gateway hosts on a CDH cluster. These hosts are assigned gateway roles for the Spark and HDFS services so that Cloudera Data Science Workbench has the client configuration required to access the CDH cluster.

job

Jobs are *sessions* that can be scheduled in advance and do not need to be launched manually each time.

Livelog

Cloudera Data Science Workbench allows users to work interactively with R, Python, and Scala from their browser and display results in realtime. This realtime state is stored in an internal database, called Livelog.

master

A typical Cloudera Data Science Workbench deployment consists of 1 master host and zero or more *worker* hosts. The master host keeps track of all critical, persistent, and stateful application data within Cloudera Data Science Workbench.

model

Model is a high level abstract term that is used to describe several possible incarnations of objects created during the model deployment process in Cloudera Data Science Workbench. You should note that 'model' does not always refer to a specific artifact. More precise terms (as defined in the [documentation](#)) should be used whenever possible.

pipeline

A series of *jobs* that depend on each other and must therefore be executed in a specific pre-defined sequence.

project

Projects hold the code, configuration, and libraries needed to reproducibly run data analytics workloads. Each project is independent, ensuring users can work freely without interfering with one another or breaking existing workloads.

session

A session is an interactive environment where you can run exploratory analysis in R, Python, and Scala.

team

A group of [trusted](#) users who are collaborating on a project in Cloudera Data Science Workbench.

terminal

Cloudera Data Science Workbench allows [terminal access](#) to actively running engines. The terminal can be used to move project files around, run Git commands, access the YARN and Hadoop CLIs, or install libraries that cannot be installed directly from the engine.

web application

Refers to the Cloudera Data Science Workbench web application running at `cdsw.<your_domain>.com`.

workbench

The console in the web application that is used to launch interactive sessions and run exploratory data analytic workloads. It consists of two panes, a navigable filesystem and editor on the left, and an interactive command prompt on the right.

worker

Worker hosts are transient hosts that can be added or removed from a Cloudera Data Science Workbench deployment depending on the number of users and workloads you are running.

Related Topics

- [Cloudera Enterprise Glossary](#)
- [Cloudera Director Glossary](#)

Appendix: Apache License, Version 2.0

SPDX short identifier: Apache-2.0

Apache License
Version 2.0, January 2004
<http://www.apache.org/licenses/>

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License.

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License.

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims

licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution.

You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

1. You must give any other recipients of the Work or Derivative Works a copy of this License; and
2. You must cause any modified files to carry prominent notices stating that You changed the files; and
3. You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
4. If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions.

Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks.

This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty.

Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability.

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability.

Appendix: Apache License, Version 2.0

While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

```
Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```