Cloudera Director User Guide

Important Notice

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.
1001 Page Mill Road, Bldg 3
Palo Alto, CA 94304
info@cloudera.com
US: 1-888-789-1488
Intl: 1-650-362-0488
www.cloudera.com

Release Information

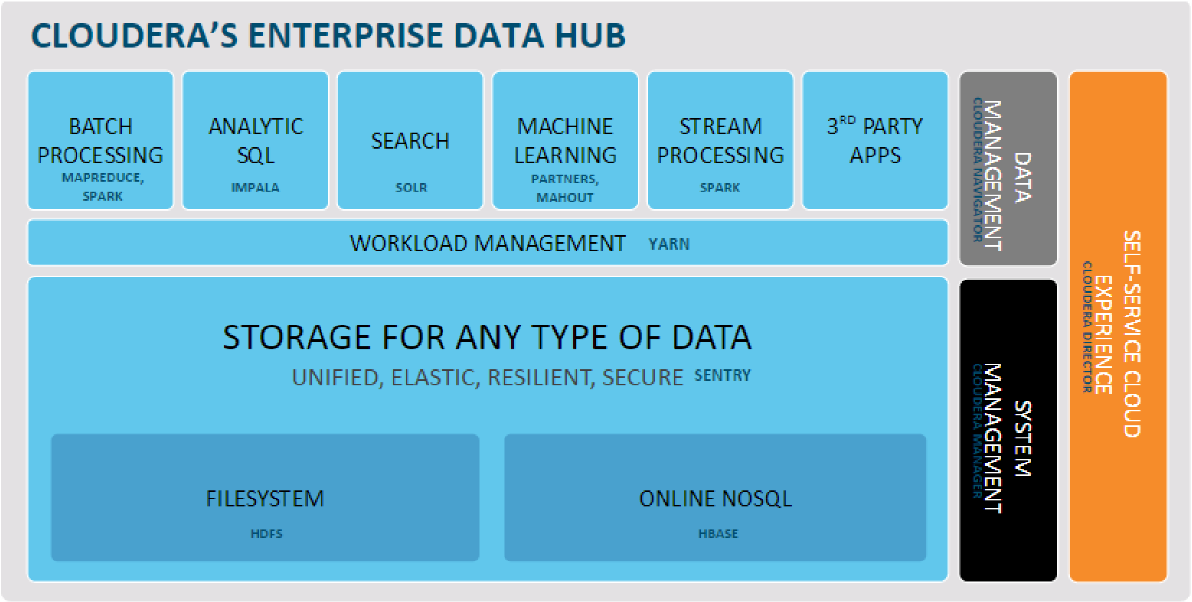
Version: Cloudera Director 2.0.x
Date: June 21, 2016

Table of Contents

Introduction

Cloudera Director enables reliable self-service for using CDH and Cloudera Enterprise Data Hub in the cloud.

Cloudera Director provides a single-pane-of-glass administration experience for central IT to reduce costs and deliver agility, and for end-users to easily provision and scale clusters. Advanced users can interact with Cloudera Director programmatically through the REST API or the CLI to maximize time-to-value for an enterprise data hub in cloud environments.



Cloudera Director is designed for both long running and ephemeral clusters. With long running clusters, you deploy one or more clusters that you can scale up or down to adjust to demand. With ephemeral clusters, you can launch a cluster, schedule any jobs, and shut the cluster down after the jobs complete.

Running Cloudera in the cloud supports:

- Faster procurement—Deploying servers in the cloud is faster than completing a lengthy hardware acquisition process.
- Easier scaling—To meet changes in cluster demand, it is easier to add and remove new hosts in the cloud than in a bare metal environment.
- Infrastructure migration—Many organizations have already moved to a cloud architecture, while others are in the process of moving.

Cloudera Director Features

Cloudera Director provides a rich set of features for launching and managing clusters in cloud environments. The following table describes the benefits of using Cloudera Director.

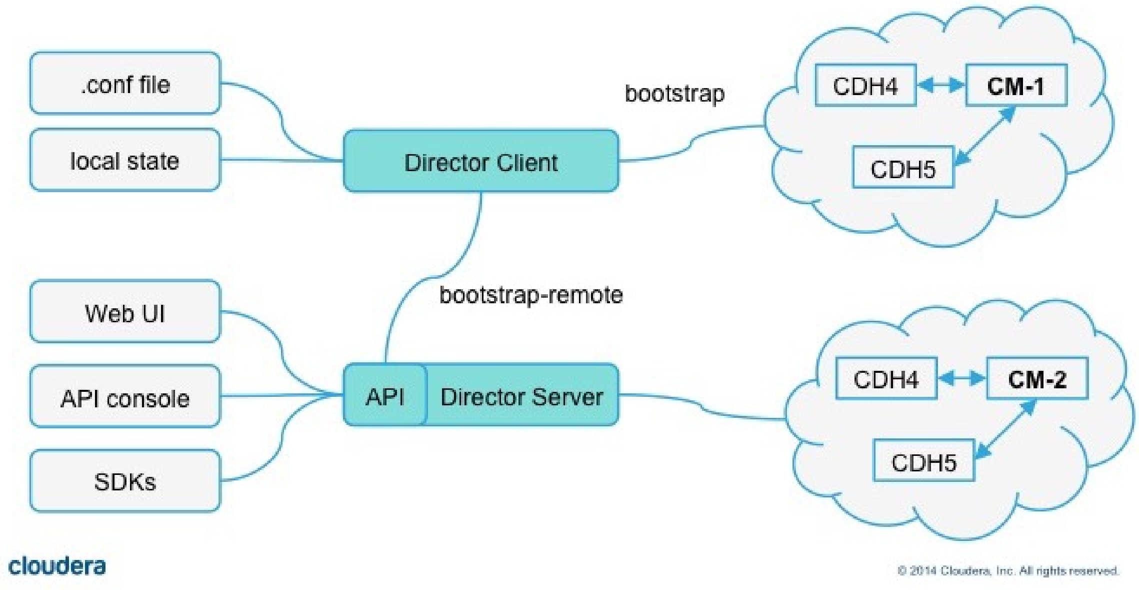
Benefit	Features
Simplified cluster lifecycle management	Simple user interface: • Self-Service spin up and tear down • Dynamic scaling for spiky workloads • Simple cloning of clusters • Cloud blueprints for repeatable deployments

Benefit	Features
Elimination of lock-in	Flexible, open platform: • 100% open source Hadoop distribution • Native support for hybrid deployments • Third-party software deployment in the same workflow • Support for custom, workload-specific deployments
Accelerated time to value	Enterprise-ready security and administration: • Support for complex cluster topologies • Minimum size cluster when capacity constrained • Management tooling • Compliance-ready security and governance • Backup and disaster recovery with an optimized cloud storage connector
Reduced support costs	Monitoring and metering tools: • Multi-cluster health dashboard • Instance tracking for account billing

Cloudera Director Client and Server

Cloudera Director supports cluster deployment through the client or the server.

The diagram below illustrates the components of Cloudera Director. At the center of the diagram are the two main components: the Cloudera Director client and Cloudera Director server.



Cloudera Director Client

The Cloudera Director client is a standalone process, with no UI and no server. It provides the simplest way of using Cloudera Director. You interact at the command line through the host on which the client is installed. You start the client process with the `bootstrap` command, and everything runs in a single process, with all configurations specified in the .conf file. When you are done, issue the terminate command to stop the process. If you want to run Cloudera Director repeatedly with the same settings, your .conf file preserves those settings and can be reused as is or with modifications.

In the diagram above, the lines that extend from the client show that the client stores its state locally. The client uses the .conf file to launch clusters, either directly by using the bootstrap command, or through the server by using the bootstrap-remote command, described below in Using Cloudera Director Server on page 10. For more information about the .conf file, see The Cloudera Director Configuration File on page 59.

Cloudera does not recommend the standalone client mode for production use, but it can be used for development work, proof-of-concept demonstrations, or trying the product.

Cloudera Director Server

The Cloudera Director server is designed for a more centralized environment, managing multiple Cloudera Manager instances and CDH clusters with multiple users and user accounts. You can log into the server UI and launch clusters, or you can send the server a cluster configuration file from the Cloudera Director client using the bootstrap-remote command. Use the server to launch and manage large numbers of clusters in a production environment.

In the diagram above, the lines that extend from the server show three interfaces to the server: the Web UI, the API console, and the SDKs. All three interfaces interact with the server through the API, represented in this diagram as part of the Cloudera Director server component. The line to the right indicates that the server, like the client, can launch Cloudera Manager instances and CDH clusters. The processes that interact with the cloud infrastructure run on the server, and the server owns the state for the clusters it has launched.

Using Cloudera Director Server

You can interact with Cloudera Director server in several ways. The best way for you depends on your purposes and whether you want to use the Cloudera-recommended default configurations, or if instead you require customized configurations for a particular use case or environment.

With the User Interface (UI) Only

The UI provides a view of the components present in your setup, including the clusters and processes that are running; the health of cluster components; and easy access to error logs. You can also use the UI for initial setups, and interact with Cloudera Director server through the UI only, without using the APIs or the .conf file.

This mode can be used in production setups, but Cloudera recommends that it be used for simple setups offered through the guided setup wizard that the UI provides, and not for customized setups. Although many advanced features are available using only the UI, it is not ideal for experimenting with configurations because your configuration settings are not preserved, making iteration difficult. For ease of iteration with customized setups, choose a mode that uses the .conf file, where your settings will be preserved, or the API, where your settings can be saved, for example, in a Python script.

With the Command Line Interface (CLI)

With the CLI, if the server is running, you can set up a cluster using the bootstrap-remote command with the .conf file. In this mode, the UI can be used to get information about the clusters and services that have been set up and the processes running on different clusters.

When used to send the .conf file to a server through bootstrap-remote, the Cloudera Director client provides full access to all advanced features, such as custom configurations of Cloudera Manager services and hosts. If you use this mode and want to iterate with the same settings, you can use the .conf file as a record of the settings. You can set up new clusters later by simply using the UI mode and entering the settings preserved in this .conf file.

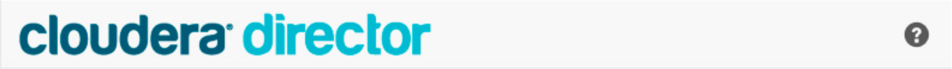
For maximum flexibility and power—for example, if you want to experiment with custom configurations and custom role assignments—using bootstrap-remote with the .conf file is a good choice.

With the API

For programmatic interaction with the server, Cloudera Director includes SDKs for Python and Java, and an API console. You can use the API to access advanced Cloudera Director features, including custom configuration settings. As with the `.conf` file, using the API supports iteration because your settings can be saved in a Python script or Java file.

Displaying Cloudera Director Documentation

To display Cloudera Director documentation for any page in the server UI, click the question mark icon in the upper-right corner at the top of the page:



The latest help files are hosted on the Cloudera web site, but help files are also embedded in the product for users who do not have Internet access. By default, the help files displayed when you click the question mark icon are those hosted on the Cloudera web site because these include the latest updates. You can configure Cloudera Director to open either the latest help from the Cloudera web site or locally installed help by toggling the value of `lp.webapp.documentationType` to `ONLINE` or `EMBEDDED` in the server `application.properties` configuration file.

Cloudera Director Release Notes

These release notes provide information on new features and known issues and limitations for Cloudera Director.

For information about supported operating systems, and other requirements for using Cloudera Director, see Requirements and Supported Versions.

New Features and Changes in Cloudera Director

New Features and Changes in Cloudera Director 2

The following sections describe what's new and changed in each Cloudera Director 2 release.

What's New in Cloudera Director 2.0.0

- AWS Spot Instances and Google Cloud Platform Preemptible Instances are supported.
- Setup of clusters that are highly available and authenticated through Kerberos is automated.
- You can automate submission of jobs to clusters with dynamic creation and termination of clusters.
- You can run custom scripts after cluster setup and before cluster termination.
- The user interface is enhanced, with deeper insights into cluster health.
- Reliability of cluster modifications is increased, including rollback in some failure scenarios.
- RHEL 7.1 is supported.
- A number of issues have been fixed. See Issues Fixed in Cloudera Director 2.0.0 for details.

New Features and Changes in Cloudera Director 1

The following sections describe what's new and changed in each Cloudera Director 1 release.

What's New in Cloudera Director 1.5.2

- Cloudera Director now supports RHEL 6.7.
- A number of issues have been fixed. See Issues Fixed in Cloudera Director 1.5.2 for details.

What's New in Cloudera Director 1.5.1

- A number of issues have been fixed. See Issues Fixed in Cloudera Director 1.5.1 on page 16 for details.

What's New in Cloudera Director 1.5.0

- Cloudera Director now supports multiple cloud providers through an open-source plugin interface, the Cloudera Director Service Provider Interface (Cloudera Director SPI).
- Google Cloud Platform is now supported through an open-source implementation of the Cloudera Director SPI, the Cloudera Director Google Plugin.
- Database servers set up by Cloudera Director can now be managed from the UI.
- You can now specify custom scripts to be run after cluster creation. Example scripts for enabling HDFS high availability and Kerberos are available on the Cloudera GitHub site.
- The Cloudera Director database can now be encrypted. Encryption is enabled by default for new installations.
- Cluster and Cloudera Manager configurations can now be set through the UI.
- A number of issues have been fixed. See Issues Fixed in Cloudera Director 1.5.0 on page 16 for details.

What's New in Cloudera Director 1.1.3

- A number of issues have been fixed. See Issues Fixed in Cloudera Director 1.1.3 on page 17 for details.
- The Cloudera Director disk preparation method now supports RHEL 6.6, which is supported by Cloudera Manager 5.4.

- Custom endpoints for AWS Identity and Access Management (IAM) are now supported.
- To ensure version compatibility between Cloudera Manager and CDH, Cloudera Director now defaults to installing the latest 5.3 version of Cloudera Manager and CDH, rather than installing the latest post-5.3 version.

What's New in Cloudera Director 1.1.2

- A number of issues have been fixed. See Issues Fixed in Cloudera Director 1.1.2 for details.

What's New in Cloudera Director 1.1.1

- A number of issues have been fixed. See Issues Fixed in Cloudera Director 1.1.1 for details.

What's New in Cloudera Director 1.1.0

- Support for demand-based shrinking of clusters
- Integration with Amazon RDS to enable end-to-end setup of clusters as well as related databases
- Native client bindings for Cloudera Director API in Java and Python
- Faster bootstrap of Cloudera Manager and clusters
- Improved User Interface of Cloudera Director server including display of health of clusters and ability to customize cluster setups
- Improvements to usability and documentation

Known Issues and Workarounds in Cloudera Director

The following sections describe the current known issues in Cloudera Director.

Cloudera Director Does Not Recognize Cloudera Manager Password Changes

Cloudera Director does not recognize changes in the `admin` password in Cloudera Manager unless the username associated with the new password is also changed.

Workaround: To update Cloudera Director with a new password for Cloudera Manager, perform the following steps:

1. Change the password for `admin` in Cloudera Manager.
2. Create a new user in Cloudera Manager with Full Administrator privileges.
3. Change Cloudera Director's credentials to this new user, either with the `update-deployment.py` script or with the **Update Cloudera Manager Credentials** command on the **Add Cluster** dropdown menu on the Cloudera Director UI page for the deployment. You can leave Cloudera Director configured to use this new user, or change the Cloudera Director credentials back to `admin` with the new password.

Cloudera Director resize script cannot resize XFS partitions

Cloudera Director is unable to resize XFS partitions, which makes creating an instance that uses the XFS filesystem fail during bootstrap.

Workaround: Use an image with an ext filesystem such as ext2, ext3, or ext4.

Incorrect yum repo definitions for Google Compute Engine RHEL images

The default RHEL 6 image defined in director-google-plugin version 1.0.1 and lower has an incorrect yum repo definition. This causes yum commands to fail after yum caches are cleared. See the Google Compute Engine issue tracker for issue details.

Workaround: Use the image rhel-6-20160119 or higher.

Cloudera Director does not set up external databases for Sqoop2

Cloudera Director cannot set up external databases for Sqoop2.

Workaround: Set up databases for this service as described in Cloudera Manager and Managed Service Databases.

Long version string required for Kafka

Kafka requires a nonintuitive version string to be specified in the configuration file or UI.

Workaround: Use the following format to specify a version string in the cluster configuration section of the configuration file or UI. For example, to deploy Kafka 1.4 in a cluster, specify `0.8.2.0-1.kafka1.4` or `0.8`, instead of `1.4`.

Metrics not displayed for clusters deployed in Cloudera Manager 5.4 and earlier clusters

Clusters deployed in Cloudera Manager version 5.4 and lower might not have metrics displayed in the UI if these clusters share the same name as previously deleted clusters.

Workaround: Use Cloudera Manager 5.5 and higher.

Modifying a cluster can leave some roles marked as stale in Cloudera Manager

When growing or shrinking a cluster, you have the option of restarting the cluster. The restart operation should only restart roles that are marked stale by Cloudera Manager—that is, roles that need to be restarted. This prevents unnecessary cluster downtime. However, with Cloudera Manager 5.5.x and lower, some stale roles might not be restarted, even if you select the **Restart Cluster** option.

Workaround: Go to Cloudera Manager, select the roles marked as stale, and restart them. This will be fixed in a future release.

Validation error after initial setup with high availability

When you set up HDFS high availability using Cloudera Director, the secondary NameNode is not configured, because it is not required for high availability. Because of a Cloudera Manager bug, the absence of a secondary NameNode causes an erroneous validation error to appear in Cloudera Manager in **HDFS** > **Configuration** > **HDFS Checkpoint Directories**.

Workaround: Update the field with a value for the checkpoint directory—for example, `/data/dfs/snn` (the value isn't important, because it is not used)—and save.

Default memory autoconfiguration for monitoring services may be suboptimal

Depending on the size of your cluster and your instance types, you may need to manually increase the memory limits for the Host Monitor and Service Monitor. Cloudera Manager displays a configuration validation warning or error if the memory limits are insufficient.

Workaround: Override firehose_heapsize for HOSTMONITOR and SERVICES with a different value in bytes (for example, 536900000 for ~512 MB). Cloudera also recommends using instances with a minimum of 15 GB of memory for management roles (30 GB recommended).

Changes to Cloudera Manager username and password must also be made in Cloudera Director

If the Cloudera Manager username and password are changed directly in Cloudera Manager, Cloudera Director can no longer add new instances or authenticate with Cloudera Manager. Username and password changes must be implemented in Cloudera Director as well.

Workaround: Use the Cloudera Director UI to update the Cloudera Manager username and password.

Cloudera Director does not sync with cluster changes made in Cloudera Manager

Modifying a cluster in Cloudera Manager after it is bootstrapped does not cause the cluster state to be synchronized with Cloudera Director. Services that have been added or removed in Cloudera Manager do not show up in Cloudera Director when growing the cluster.

Workaround: None.

Cloudera Director may use AWS credentials from instance of Cloudera Director Server

Cloudera Director Server uses the AWS credentials from a configured Environment, as defined in a client configuration file or through the Cloudera Director UI. If the Environment is not configured with credentials in Cloudera Director, the Cloudera Director server instead uses the AWS credentials that are configured on the instance on which the Cloudera Director server is running. When those credentials differ from the intended ones, EC2 instances may be allocated under unexpected accounts. Ensure that the Cloudera Director server instance is not configured with AWS credentials.

Severity: Medium

Workaround: Ensure that the Cloudera Director Environment has correct values for the keys. Alternatively, use IAM profiles for the Cloudera Director server instance.

Root partition resize fails on CentOS 6.5 (HVM)

Cloudera Director cannot resize the root partition on Centos 6.5 HVM AMIs. This is caused by a bug in the AMIs. For more information, see the CentOS Bug Tracker.

Workaround: None.

Terminating clusters that are bootstrapping must be terminated twice for the instances to be terminated

Terminating a cluster that is bootstrapping stops ongoing processes but keeps the cluster in the bootstrapping phase.

Severity: Low

Workaround: To transition the cluster to the **Terminated** phase, terminate the cluster again.

When using RDS and MySQL, Hive Metastore canary may fail in Cloudera Manager

If you include Hive in your clusters and configure the Hive metastore to be installed on MySQL, Cloudera Manager may report, "The Hive Metastore canary failed to create a database." This is caused by a MySQL bug in MySQL 5.6.5 or higher that is exposed when used with the MySQL JDBC driver (used by Cloudera Director) version 5.1.19 or lower. For information on the MySQL bug, see the MySQL bug description.

Workaround: Depending on the driver version installed by Cloudera Director from your platform's software repositories, select an older MySQL version that does not have this bug.

Issues Fixed in Cloudera Director

The following sections describe fixed issues in each Cloudera Director 1 release.

Issues Fixed in Cloudera Director 2.0.0

Cloning and growing a Kerberos-enabled cluster fails

Cloning of a cluster that uses Kerberos authentication fails, whether it is cloned manually or by using the `kerberize-cluster.py` script. Growing a cluster that uses Kerberos authentication fails.

Kafka with Cloudera Manager 5.4 and lower causes failure

Kafka installed with Cloudera Manager 5.4 and lower causes the Cloudera Manager installation wizard, and therefore the bootstrap process, to fail, unless you override the configuration setting `broker_max_heap_size`.

Cloudera Director does not set up external databases for Oozie and Hue

Cloudera Director cannot set up external databases for Oozie and Hue.

Issues Fixed in Cloudera Director 1.5.2

Apache Commons Collections deserialization vulnerability

Cloudera has learned of a potential security vulnerability in a third-party library called the Apache Commons Collections. This library is used in products distributed and supported by Cloudera ("Cloudera Products"), including Cloudera Director. At this time, no specific attack vector for this vulnerability has been identified as present in Cloudera Products.

The Apache Commons Collections potential security vulnerability is titled "Arbitrary remote code execution with InvokerTransformer" and is tracked by COLLECTIONS-580. MITRE has not issued a CVE, but related CVE-2015-4852 has been filed for the vulnerability. CERT has issued Vulnerability Note #576313 for this issue.

Releases affected: Cloudera Director 1.5.1 and lower, CDH 5.5.0, CDH 5.4.8 and lower, Cloudera Manager 5.5.0, Cloudera Manager 5.4.8 and lower, Cloudera Navigator 2.4.0, and Cloudera Navigator 2.3.8 and lower

Users affected: All

Severity (Low/Medium/High): High

Impact: This potential vulnerability may enable an attacker to run arbitrary code from a remote machine without requiring authentication.

Immediate action required: Upgrade to Cloudera Director 1.5.2, Cloudera Manager 5.5.1, and CDH 5.5.1.

Serialization for complex nested types in Python API client

Serialization for complex nested types has been fixed in the Python API client.

Issues Fixed in Cloudera Director 1.5.1

Support for configuration keys containing special characters

Configuration file parsing has been updated to correctly support quoted configuration keys containing special characters such as colons and periods. This enables the usage of special characters in service and role type configurations, and in instance tag keys.

Issues Fixed in Cloudera Director 1.5.0

Growing clusters may fail when using a repository URL that only specifies major and minor versions

When using a Cloudera Manager package repository or CDH/parcel repository URL that only specifies the major or minor versions, Cloudera Director may incorrectly use the latest available version when trying to grow a cluster.

For Cloudera Manager: `http://archive.cloudera.com/cm5/redhat/6/x86_64/cm/5.3.3/`

For CDH: `http://archive.cloudera.com/cdh5/parcels/5.3.3/`

Flume does not start automatically after first run

Although you can deploy Flume through Cloudera Director, you must start it manually using Cloudera Manager after Cloudera Director bootstraps the cluster.

Impala daemons attempt to connect over IPv6

Impala daemons attempt to connect over IPv6.

DNS queries occasionally time out with AWS VPN

DNS queries occasionally time out with AWS VPN.

Issues Fixed in Cloudera Director 1.1.3

Ensure accurate time on startup

Instance normalization has been improved to ensure that time is synchronized by Network Time Protocol (NTP) before bootstrapping, which improves cluster reliability and consistency.

Speed up ephemeral drive preparation

Instance drive preparation during the bootstrapping process was slow, especially for instances with many large ephemeral drives. Time required for this process has been reduced.

Fix typographical error in the virtualizationmappings.properties file

The d2 instance type `d2.4xlarge` was incorrectly entered into Cloudera Director as `d3.4xlarge` in `virtualizationmappings.properties`. This has been corrected.

Avoid upgrading preinstalled Cloudera Manager packages

Cloudera Director no longer upgrades preinstalled Cloudera Manager packages.

Issues Fixed in Cloudera Director 1.1.2

Parcel validation fails when using HTTP proxy

Parcel validation now works when configuring an HTTP proxy for Cloudera Director server, allowing correctly configured parcel repository URLs to be used as expected.

Unable to grow a cluster after upgrading Cloudera Director 1.0 to 1.1.0 or 1.1.1

Cloudera Director now sets up parcel repository URLs correctly when a cluster is modified.

Add support for d2 and c4 AWS instance types

Cloudera Director now includes support for new AWS instance types d2 and c4. Cloudera Director can be configured to use additional instance types at any point as they become available in AWS.

Issues Fixed in Cloudera Director 1.1.1

Service-level custom configurations are ignored

Restored the ability to have service-level custom configurations. Due to internal refactoring changes, it was no longer possible to override service-level configs.

The property customBannerText is ignored and not handled as a deprecated property

Restored the customBannerText configuration file property, which was removed during the internal refactoring work.

Fixed progress bar issues when a job fails

The UI showed a progress bar even when a job had failed.

Updated IAM Help text on Add Environment page

The help text on the Add Environment page for Role-based keys should refer to AWS Identity and Access Management (IAM), not to AMI.

Add eu-central-1 to the region dropdown

The eu-central-1 region has been added to the region dropdown on the Add Environment page.

Gateway roles should assign YARN, HDFS, and Spark gateway roles

All available gateway roles, including YARN, HDFS, and Spark, should be deployed by default on the instance.

Spark on YARN should be shown on the Modify Cluster page

Spark on YARN did not appear in the list of services on the Modify Cluster page.

Requirements and Supported Versions

The following sections describe the requirements and supported operating systems, databases, and browsers for Cloudera Director.

Cloud Providers

Cloudera Director has native support for Amazon Web Services (AWS) and Google Cloud Platform.

Each Cloudera Director release embeds the current plug-in for supported cloud providers, but a newer plug-in may have been posted on the Cloudera GitHub site subsequent to the Cloudera Director release. To check for the latest version, click the appropriate link:

- AWS cloud provider plug-in
- Google Cloud Platform cloud provider plug-in

Cloudera Director Service Provider Interface (SPI)

The Cloudera Director SPI defines an open source Java interface that plug-ins implement to add support for additional cloud providers to Cloudera Director. For more information, see the README.md file in the SPI Cloudera Director GitHub repository.

Supported Software and Distributions

The table below lists software requirements, recommendations, and supported versions for resources used with Cloudera Director.

	Cloudera Director	**Cloudera Manager and CDH**
Operating Systems (64-bit only)	RHEL and CentOS 6.5, 6.7, 7.1, and 7.2 Ubuntu 14.04	RHEL and CentOS 6.5, 6.7, 7.1, and 7.2 **Note:** RHEL 7.2 is supported only for Cloudera Manager and CDH 5.7 and higher, not for lower versions of Cloudera Manager and CDH. **Note:** To use Amazon EC2 D2 instances, you must run a minimum version of RHEL 6.7 or CentOS 6.7. Earlier versions of RHEL and CentOS do not support these instance types.

	Cloudera Director	Cloudera Manager and CDH
Oracle Java SE Development Kit (JDK)	Oracle JDK version 7 or 8 **Note:** For download and installation information, see Java SE Downloads.	Oracle JDK version 7 or 8
Default Database	Embedded H2 database	Embedded PostgreSQL Database
Supported Databases	MySQL 5.5, 5.6 MariaDB 5.5	MySQL 5.5, 5.6 MariaDB 5.5

Note: By default, Cloudera Director stores its environment and cluster data in an embedded H2 database located at `/var/lib/cloudera-director-server/state.h2.db`. Back up this file to avoid losing the data. For information on using an external MySQL database in place of the H2 embedded database, see Using MySQL for Cloudera Director Server on page 62. Cloudera recommends using an external database for both Cloudera Director and Cloudera Manager for production environments.

Resource Requirements

The table below lists requirements for resources used with Cloudera Director.

	Cloudera Director	Cloudera Manager and CDH
CPU	2	4
RAM	3.75 GB	64 GB
Disk	8 GB	500 GB
Recommended AWS instance	c3.large or c4.large	Cloudera Manager: m4.xlarge or m4.4xlarge **Note:** The recommended instance for Cloudera Manager depends on the workload. Contact your Cloudera account representative for more information.
Recommended Google Cloud Platform instance	n1-standard-2	n1-highmem-4 or n1-highmem-8

Supported Cloudera Manager and CDH Versions

Cloudera Director 2.0 can install any version of Cloudera Manager 5 with any CDH 5 parcels. Use of CDH packages is not supported.

Networking and Security Requirements

Cloudera Director requires the following inbound ports to be open:

- **TCP ports 22:** These ports allow SSH to Cloudera Director instance.
- **All traffic across all ports within the security group:** This rule allows connectivity with all the components within the Hadoop cluster. This rule avoids numerous individual ports to be opened in the security group.

Type	Protocol	Port Range	Source
SSH (22)	TCP (6)	22	0.0.0.0/0
ALL Traffic	ALL	ALL	*security_group_id* See note paragraph below.

Note: The **All traffic** rule above requires the security group ID. If you create a security group from scratch, create the security group with the SSH rule and then go back and edit the security group to allow all traffic within the security group.

To connect to the AWS network, Cloudera recommends that you open only these ports and set up a SOCKS proxy. Unless your network has direct connection to AWS, you must set this up to access the Cloudera Director instance. This is done in a later step.

Ports Used by Cloudera Director

Cloudera Director uses the ports listed in the table below.

Component	Service	Port	Access Requirement	Configuration	Comment
Cloudera Director server	HTTP/HTTPS port for Cloudera Director UI and API	7189	Must be accessible from outside the cluster	`server.port` in application.properties	Web UI and API
Cloudera Director internal shell	CRaSH shell port	2000	localhost only	`shell.ssh.port` in application.properties	Used with the ssh client

Cloudera Director also uses the following ports in a typical deployment:

- 80 and 443: to connect to external services for validation and tracking.
- 7180: to talk to the Cloudera Manager API
- 22: to connect to new instances over SSH
- 123: to configure NTP within the cluster.

Supported Browsers

Cloudera Director supports the following browsers:

- Mozilla Firefox 11 and higher
- Google Chrome
- Internet Explorer 9 and higher
- Safari 5 and higher

Getting Started with Cloudera Director

This section explains how to get Cloudera Director up and running on Amazon Web Services (AWS) and Google Cloud Platform.

Getting Started on Amazon Web Services (AWS)

To use Cloudera Director on AWS, you create an environment in Amazon Virtual Private Cloud (Amazon VPC), start an instance in AWS to run Cloudera Director, and create a secure connection. This section describes the steps for each of these tasks.

Important:

Cloudera Director supports Spot instances. Spot instances are virtual machines that have a lower cost but are subject to reclamation at any time by AWS. Because of the possibility of interruption, Cloudera recommends that you use Spot instances only for worker roles in a cluster, not for master or gateway roles. Cloudera Director only supports Spot instances for CentOS.

For more information about using Spot instances with Cloudera Director, see Using Spot Instances on page 104.

Setting up the AWS Environment

You must set up a VPC and create an SSH key pair in the AWS environment before deploying Cloudera Director.

Setting Up a VPC

Cloudera Director requires an Amazon Virtual Private Cloud (Amazon VPC) to implement its virtual environment. The Amazon VPC must be set up for forward and reverse hostname resolution.

To set up a new VPC, follow the steps below. Skip these steps if you are using an existing VPC.

1. Log in to the AWS Management Console and make sure you are in the desired region. The current region is displayed in the upper-right corner of the AWS Management Console. Click the region name to change your region.
2. In the AWS Management Console, select **VPC** in the Networking section.
3. Click **Start VPC Wizard**. (Click VPC Dashboard in the left side pane if the **Start VPC Wizard** button is not displayed.)
4. Select the desired VPC configuration. For the easiest way to get started, select **VPC with a Single Public Subnet**.
5. Complete the VPC wizard and then click **Create VPC**.

Configuring your Security Group

Cloudera Director requires the following inbound ports to be open:

Type	Protocol	Port Range	Source
`ALL Traffic`	`ALL`	`ALL`	*security_group_id*
`SSH (22)`	`TCP (6)`	22	0.0.0.0/0

Note: By default, Cloudera Director requires unrestricted outbound connectivity. You can configure Cloudera Director to use proxy servers or a local mirror of all the relevant repositories if required.

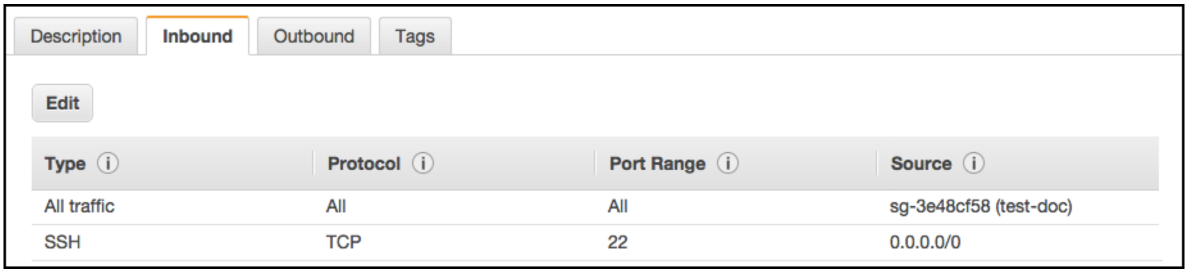
Creating a New Security Group

1. In the left pane, click **Security Groups**.

2. Click **Create Security Group**.
3. Enter a name and description. Make sure to select the VPC you created from the VPC list box.
4. Click **Yes, Create**.

Select the newly created security group and add inbound rules as detailed in the table above.

The configured security group should look similar to the following, but with your own values in the Source column.



Type	Protocol	Port Range	Source
All traffic	All	All	sg-3e48cf58 (test-doc)
SSH	TCP	22	0.0.0.0/0

For more information about security groups in AWS, see Security Groups for Your VPC.

Creating an SSH Key Pair

To interact with the cluster launcher and other instances, you must create an SSH key pair or use an existing EC2 key pair. If you do not have a key pair, follow these steps:

> **Note:** For information on importing an existing key pair, see Amazon EC2 Key Pairs in the AWS documentation.

1. Select **EC2** from the **Services** navigation list box.
2. In the left pane, click **Key Pairs**.
3. Click **Create Key Pair**. In the Create Key Pair dialog box, enter a name for the key pair and click **Create**.
4. Note the key pair name. Move the automatically downloaded private key file (with the .pem extension) to a secure location and note the location.

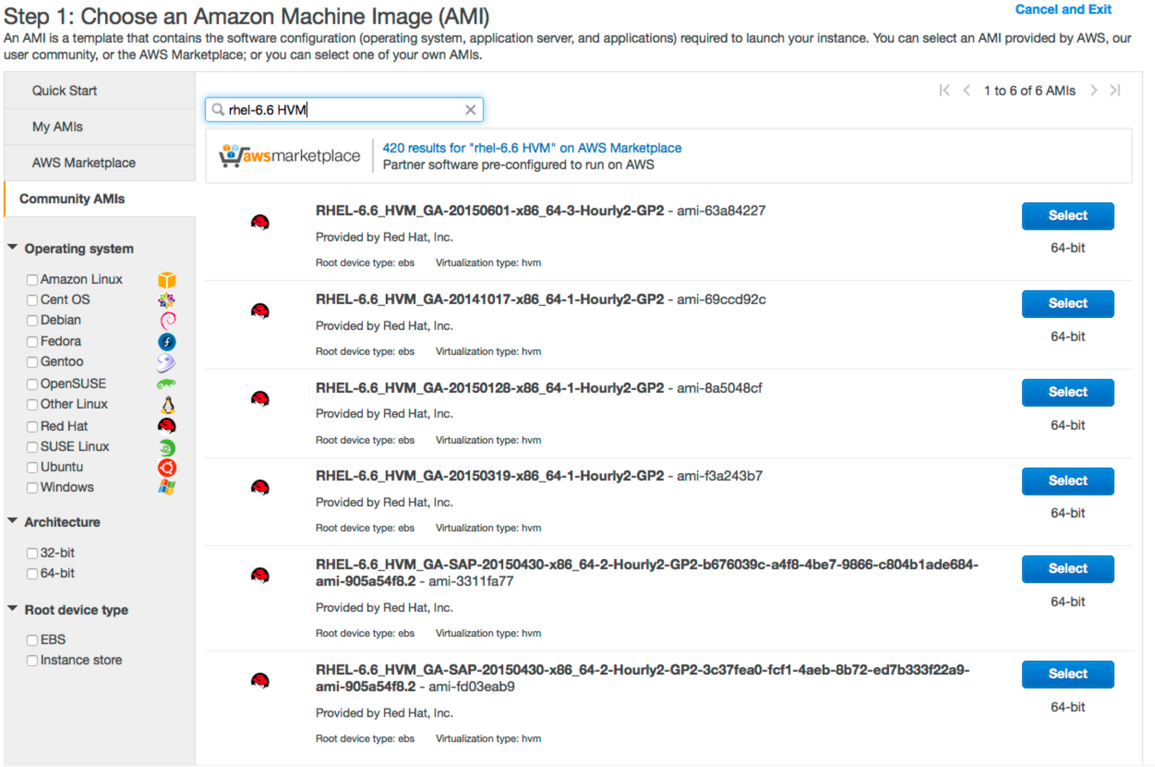
You are now ready to launch an EC2 instance.

Launching an EC2 Instance for Cloudera Director

On AWS, Cloudera Director requires a dedicated Amazon EC2 instance in the same subnet that can access new instances on the private network.

To create the instance, follow these steps:

1. In the AWS Management Console, select **EC2** from the **Services** navigation list box in the desired region.
2. Click the **Launch Instance** button in the Create Instance section of the EC2 dashboard.
3. Select the AMI for your Cloudera Director instance. Cloudera recommends that you choose from the Community AMIs list and the latest release of the desired supported distribution. See Supported Software and Distributions on page 19.
   a. Select **Community AMIs** in the left pane.
   b. In the search box, type the desired operating system. For example, if you type `rhel-6.6 HVM`, the search results show the versions of RHEL v6.6 that support HVM. Select the highest GA number to use the latest release of RHEL v6.6 supporting HVM.



Step 1: Choose an Amazon Machine Image (AMI)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

rhel-6.6 HVM

420 results for "rhel-6.6 HVM" on AWS Marketplace
Partner software pre-configured to run on AWS

RHEL-6.6_HVM_GA-20150601-x86_64-3-Hourly2-GP2 - ami-63a84227
Provided by Red Hat, Inc.
Root device type: ebs Virtualization type: hvm
64-bit

RHEL-6.6_HVM_GA-20141017-x86_64-1-Hourly2-GP2 - ami-69ccd92c
Provided by Red Hat, Inc.
Root device type: ebs Virtualization type: hvm
64-bit

RHEL-6.6_HVM_GA-20150128-x86_64-1-Hourly2-GP2 - ami-8a5048cf
Provided by Red Hat, Inc.
Root device type: ebs Virtualization type: hvm
64-bit

RHEL-6.6_HVM_GA-20150319-x86_64-1-Hourly2-GP2 - ami-f3a243b7
Provided by Red Hat, Inc.
Root device type: ebs Virtualization type: hvm
64-bit

RHEL-6.6_HVM_GA-SAP-20150430-x86_64-2-Hourly2-GP2-b676039c-a4f8-4be7-9866-c804b1ade684-ami-905a54f8.2 - ami-3311fa77
Provided by Red Hat, Inc.
Root device type: ebs Virtualization type: hvm
64-bit

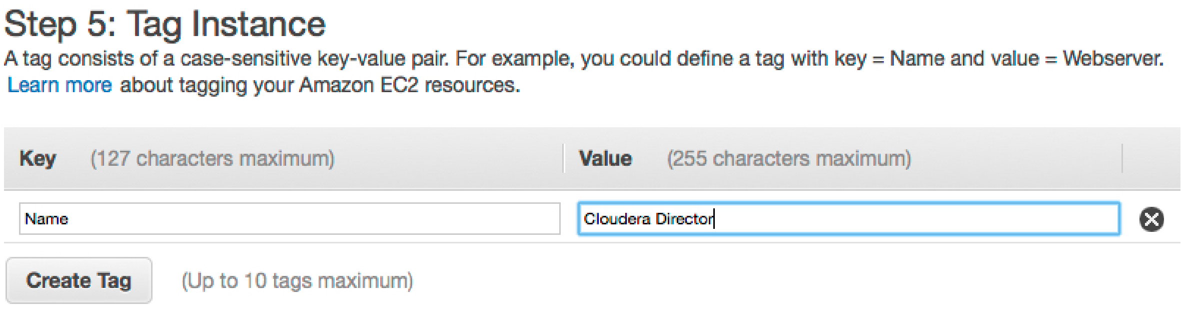
RHEL-6.6_HVM_GA-SAP-20150430-x86_64-2-Hourly2-GP2-3c37fea0-fcf1-4aeb-8b72-ed7b333f22a9-ami-905a54f8.2 - ami-fd03eab9
Provided by Red Hat, Inc.
Root device type: ebs Virtualization type: hvm
64-bit

c. Click **Select** for the AMI version you choose.

4. Select the instance type for Cloudera Director. Cloudera recommends using c3.large or c4.large instances.
5. Click **Next: Configure Instance Details**.

    a. Select the correct VPC and subnet.
    b. The cluster launcher requires Internet access; from the **Auto-assign Public IP** list box, select **Enable**.
    c. Use the default shutdown behavior, **Stop**.
    d. Click the **Protect against accidental termination** checkbox.
    e. (Optional) Click the IAM role drop-down list and select an IAM role.

6. Click **Next: Add Storage**. Cloudera Director requires a minimum of 8 GB.
7. Click **Next: Tag Instance**. For the **Name** key, enter a name for the instance in the **Value** field. Optionally, click **Create Tag** to create additional tags for the instance (up to a maximum of 10 tags).



Step 5: Tag Instance

A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver. Learn more about tagging your Amazon EC2 resources.

Key (127 characters maximum)	Value (255 characters maximum)
Name	Cloudera Director

8. Click **Next: Configure Security Group**.
9. On the **Configure Security Group** page, create a new security group or add ports to an existing group. (If you already have a security group with the required ports for Cloudera Director, you can skip this step.)

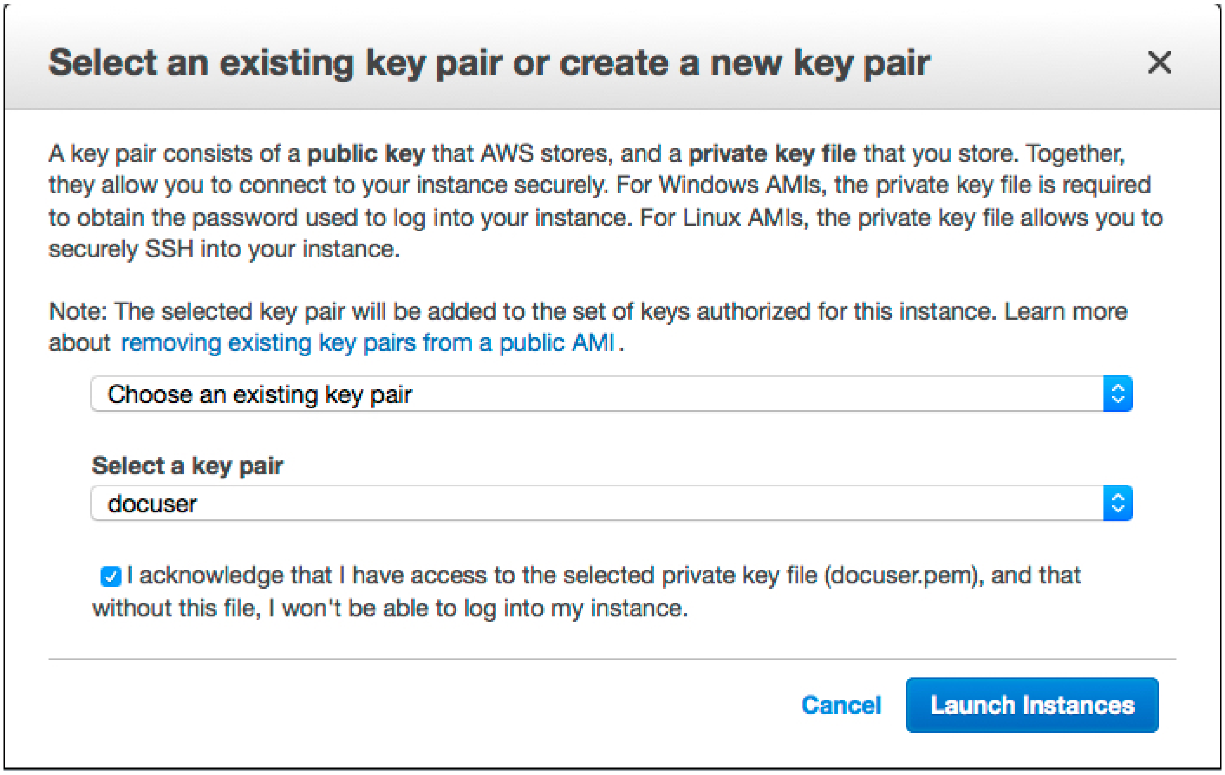
    a. Select either **Create a new security group** or **Select an existing security group**. If you create a new group, enter a **Security group name** and **Description**. To edit an existing group, select the group you want to edit.

b. Click the **Type** drop-down list, and select a protocol type. Type the port number in the **Port Range** field.
c. For each additional port needed, click the **Add Rule** button. Then click the **Type** drop-down list, select a protocol type, and type the port number in the **Port Range** field.

   The following ports must be open for the Cloudera Director EC2 instance:

Type	Protocol	Port Range	Source
SSH (22)	TCP (6)	22	0.0.0.0/0
ALL Traffic	ALL	ALL	*security_group_id*

10. Click **Review and Launch**. Scroll down to review the AMI details, instance type, and security group information, and then click **Launch**.
11. At the prompt for a key pair:
    a. Select **Choose an existing key pair** and select the key pair you created in [Setting up the AWS Environment](#) on page 22.
    b. Click the check box to acknowledge that you have access to the private key.



Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about removing existing key pairs from a public AMI.

Choose an existing key pair

Select a key pair
docuser

☑ I acknowledge that I have access to the selected private key file (docuser.pem), and that without this file, I won't be able to log into my instance.

12. Click **Launch Instances**.
13. After the instance is created, note its public and private IP addresses.

You are now ready to [configure a SOCKS proxy](#).

Installing Cloudera Director Server and Client on the EC2 Instance

Cloudera recommends that you install Cloudera Director server and client on your cloud provider in the subnet where you create CDH clusters, because Cloudera Director requires access to the private IP addresses of the instances that it creates. To install Cloudera Director, perform the following tasks. You must be either running as root or using sudo to perform these tasks.

RHEL 7 and CentOS 7

1. SSH as `ec2-user` into the EC2 instance you created for Cloudera Director. If you have VPN or AWS Direct Connect, SSH to your private IP address. Otherwise, use your public IP address.

```
ssh -i your_file.pem ec2-user@private_IP_address
```

2. Install a supported version of the Oracle Java Development Kit (JDK) on the Cloudera Director host. Currently, Cloudera Director supports JDK versions 7 and 8. For download and installation information, see Java SE Downloads. After downloading the RPM file to the EC2 instance, install the JDK:

```
sudo yum localinstall jdk-version-linux-x64.rpm
```

3. Some RHEL 7 AMIs do not include `wget` by default. If your RHEL AMI does not, install it now:

```
sudo yum install wget
```

4. Add the Cloudera Director repository to the package manager:

```
cd /etc/yum.repos.d/
sudo wget "http://archive.cloudera.com/director/redhat/7/x86_64/director/cloudera-director.repo"
```

5. Install Cloudera Director server and client by running the following command:

```
sudo yum install cloudera-director-server cloudera-director-client
```

6. Start the Cloudera Director server by running the following command:

```
sudo service cloudera-director-server start
```

7. If the RHEL 7 or CentOS firewall is running on the EC2 instance where you have installed Cloudera Director, disable and stop the firewall with the following commands:

```
# include the next line if firewalld is not yet installed
sudo yum install firewalld
sudo systemctl disable firewalld
sudo systemctl stop firewalld
```

You are now ready to deploy Cloudera Manager and CDH on the Cloudera Director server.

RHEL 6 and CentOS 6

1. SSH as `ec2-user` into the EC2 instance you created for Cloudera Director. If you have VPN or AWS Direct Connect, SSH to your private IP address. Otherwise, use your public IP address.

```
ssh -i your_file.pem ec2-user@private_IP_address
```

2. Install a supported version of the Oracle Java Development Kit (JDK) on the Cloudera Director host. Currently, Cloudera Director supports JDK versions 7 and 8. For download and installation information, see Java SE Downloads. After downloading the RPM file to the EC2 instance, install the JDK:

```
sudo yum localinstall jdk-version-linux-x64.rpm
```

3. Add the Cloudera Director repository to the package manager:

```
cd /etc/yum.repos.d/
sudo wget "http://archive.cloudera.com/director/redhat/6/x86_64/director/cloudera-director.repo"
```

4. Install Cloudera Director server and client by running the following command:

```
sudo yum install cloudera-director-server cloudera-director-client
```

5. Start the Cloudera Director server by running the following command:

```
sudo service cloudera-director-server start
```

6. Save the existing iptables rule set and disable the firewall:

```
sudo service iptables save
sudo chkconfig iptables off
sudo service iptables stop
```

You are now ready to deploy Cloudera Manager and CDH on the Cloudera Director server.

Ubuntu

1. SSH as `ubuntu` into the EC2 instance you created for Cloudera Director. If you have VPN or AWS Direct Connect, SSH to your private IP address. Otherwise use your public IP address.

```
ssh -i your_file.pem ubuntu@private_IP_address
```

2. Install a supported version of the Oracle Java Development Kit (JDK) on the Cloudera Director host. Currently, Cloudera Director supports JDK versions 7 and 8. For download and installation information, see Java SE Downloads. After downloading the installation file to the EC2 instance, install the JDK. The following example installs JDK version 7:

```
sudo apt-get update
sudo apt-get install oracle-j2sdk1.7
```

3. Add the Cloudera Director repository to the package manager:

```
cd /etc/apt/sources.list.d/
sudo curl "http://archive.cloudera.com/director/ubuntu/trusty/amd64/director/cloudera-director.list" -O
```

4. Add the signing key:

```
sudo curl -s "http://archive.cloudera.com/director/ubuntu/trusty/amd64/director/archive.key" | sudo apt-key add
-
```

5. Install Cloudera Director server by running the following command:

```
sudo apt-get update
sudo apt-get install cloudera-director-server cloudera-director-client
```

6. Start the Cloudera Director server by running the following command:

```
sudo service cloudera-director-server start
```

7. Save the existing firewall rules and disable the firewall:

```
sudo iptables-save > ~/firewall.rules
sudo service ufw stop
```

Installing Only the Client or the Server

The installation instructions above will install both the server and client, providing the full functionality of Cloudera Director. Optionally, you can install just the client, but this will only enable you to use the client in standalone mode. Similarly, you can install just the server, but then you will be unable to launch a cluster at the command line with a customized configuration file. For more information on the Cloudera Director client and server, and how they work together, see Cloudera Director Client and Server on page 9.

To install only Cloudera Director client, run one of the following installation commands in place of the command given above:

- For RHEL and CentoOS, run the command `sudo yum install cloudera-director-client` instead of `sudo yum install cloudera-director-server cloudera-director-client`.
- For Ubuntu: run the command `sudo apt-get install cloudera-director-client` instead of `sudo apt-get install cloudera-director-server cloudera-director-client`.

To install only Cloudera Director server, run one of the following installation commands in place of the command given above:

- For RHEL and CentoOS, run the command `sudo yum install cloudera-director-server` instead of `sudo yum install cloudera-director-server cloudera-director-client`.
- For Ubuntu: run the command `sudo apt-get install cloudera-director-server` instead of `sudo apt-get install cloudera-director-server cloudera-director-client`.

Configuring a SOCKS Proxy for Amazon EC2

For security purposes, Cloudera recommends that you connect to your cluster using a SOCKS proxy. A SOCKS proxy allows a client (your computer, for example) to connect directly and securely to a server (the Director instance).

To set up a SOCKS proxy for your Google Chrome web browser, follow the steps below.

Step 1: Create a Proxy Autoconfig File

The proxy autoconfig (PAC) file contains the rules required for Cloudera Director. To create a PAC file, perform the following tasks:

1. Open a text editor and enter the following text:

```
function regExpMatch(url, pattern) {
  try { return new RegExp(pattern).test(url); } catch(ex) { return false; }
}

function FindProxyForURL(url, host) {
    // Important: replace 172.31 below with the proper prefix for your VPC subnet
    if (shExpMatch(url, "*172.31.*")) return "SOCKS5 localhost:8157";
    if (shExpMatch(url, "*ec2*.amazonaws.com*")) return 'SOCKS5 localhost:8157';
    if (shExpMatch(url, "*.compute.internal*") || shExpMatch(url,
"*://compute.internal*")) return 'SOCKS5 localhost:8157';
    if (shExpMatch(url, "*ec2.internal*")) return 'SOCKS5 localhost:8157';
    return 'DIRECT';
}
```

2. Save the file.

Step 2: Set Up SwitchySharp

1. Open Google Chrome and go to Chrome Extensions.
2. Search for **Proxy SwitchySharp** and add to it Chrome.
3. In the **SwitchySharp Options** screen, click the **Proxy Profiles** tab and do the following:

   a. In the **Profile Name** field, enter `AWS-Cloudera`.

   b. Click **Automatic Configuration**.

   c. Click **Import PAC File** and import your PAC file.

   d. Click **Save**.

4. Click the **General** tab and do the following:

   a. Click **Quick Switch**.

   b. Drag **[Direct Connection]** and **AWS-Cloudera** to the **Cycled Profiles** area.

c. Set **Startup Profile** to **[Direct Connection]**.
d. Click **Save**.

Step 3: Set Up a SOCKS Proxy with SSH

Set up a SOCKS proxy to access the EC2 instance running Cloudera Director. For example, in RHEL, run the following command (with your instance information):

```
ssh -i "your-key-file.pem" -CND 8157 ec2-user@instance_running_director_server
```

where

- C sets up compression.
- N suppresses any command execution once established.
- D 8157 sets up the SOCKS 5 proxy on the port.

> **Important:** If you are using a PAC file, you must use port 8157.

You are now ready to install Cloudera Director.

Deploying Cloudera Manager and CDH on AWS

To deploy Cloudera Manager and CDH on an AWS EC2 instance, begin by creating an environment. The environment defines common settings, like region and key pair, that Cloudera Director uses with AWS. While creating an environment, you are also prompted to deploy its first cluster.

> **Note:** The lifecycle of instances and clusters depends on the availability of external repositories (for example, the Cloudera Manager repository). If these repositories are unreachable during this lifecycle, Cloudera Director cannot grow the cluster, and a grow operation results in a `Modify failed` state until the repository is available again. To ensure that there is no point of failure during cluster growth, you can preload the AMIs you use with Cloudera Manager and CDH.

To create an environment:

1. Open a web browser and go to the private IP address of the instance you created in Launching an EC2 Instance for Cloudera Director on page 23. Include port 7189 in the address. For example:

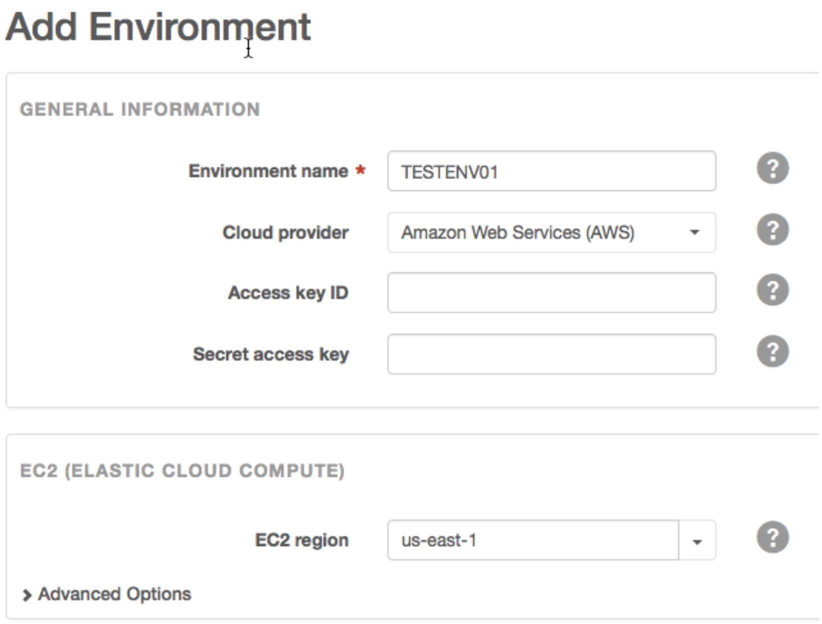
```
http://192.0.2.0:7189
```

2. In the **Cloudera Director** login screen, enter `admin` in both the **Username** and the **Password** fields.
3. In the Cloudera Director **Welcome** screen, click **Let's get started**.

   This opens a wizard for adding an environment, Cloudera Manager, and a CDH cluster.

4. In the **Add Environment** screen:

   a. Enter a name in the **Environment Name** field.
   b. Select **Amazon Web Services (AWS)** from the **Cloud provider** field.
   c. Enter your AWS credentials in the **Access key ID** and **Secret access key** fields.
   d. In the **EC2 region** field, select the same region in which your Cloudera Director instance was created.



Add Environment

GENERAL INFORMATION

Environment name * TESTENV01

Cloud provider Amazon Web Services (AWS)

Access key ID

Secret access key

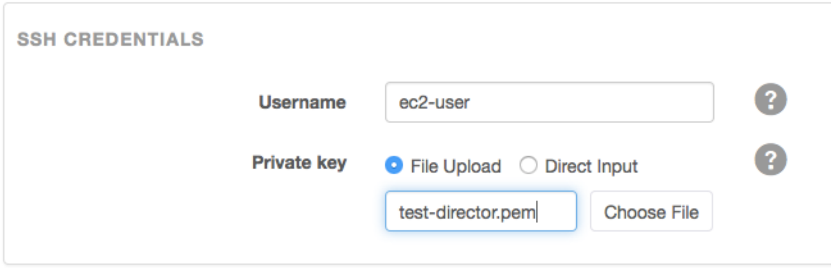
EC2 (ELASTIC CLOUD COMPUTE)

EC2 region us-east-1

› Advanced Options

e. In the **SSH Credentials** section:

   a. Enter **ec2-user** in the **Username** field.
   b. Copy the SSH private key you created in Launching an EC2 Instance for Cloudera Director on page 23 in the **Private key** field.



SSH CREDENTIALS

Username ec2-user

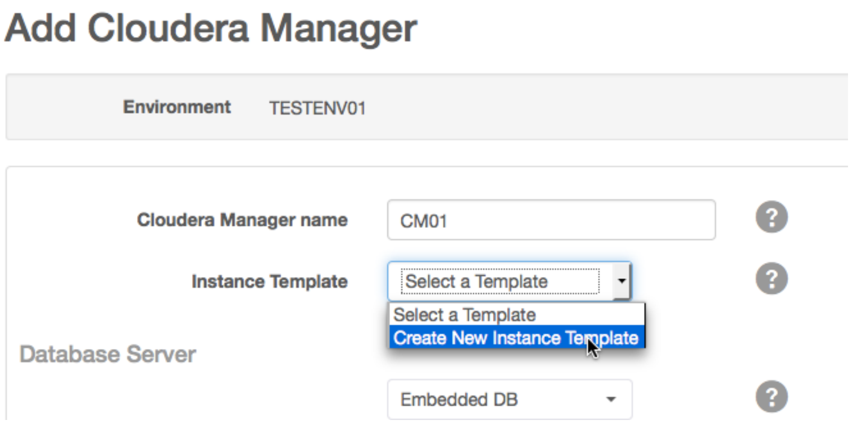
Private key ● File Upload ○ Direct Input

test-director.pem Choose File

5. Click **Continue** to add Cloudera Manager.
6. In the **Add Cloudera Manager** screen:

   a. Enter a name for this deployment of Cloudera Manager in the **Cloudera Manager name** field.
   b. In the **Instance Template** field, click **Select a Template** if you already have one that you want to use, otherwise, click **Create New Instance Template**.

   The **Create New Instance Template** modal screen displays.



Add Cloudera Manager

Environment TESTENV01

Cloudera Manager name CM01

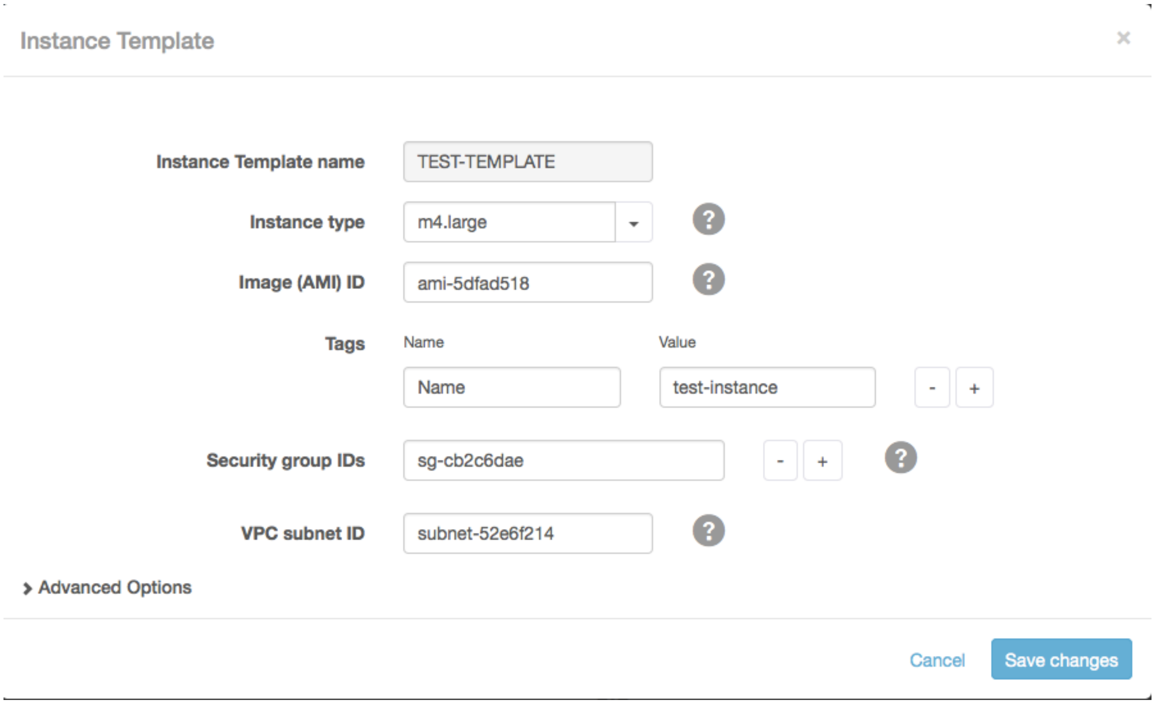
Instance Template Select a Template

Select a Template
Create New Instance Template

Database Server

Embedded DB

7. In the **Create New Instance Template** modal screen:
    a. In the **Instance Template name** field, enter a name for the template.
    b. In the **Instance type** field, select **m4.large** or **m4.xlarge**.
    c. In the **Image (AMI) ID** field, enter the ID for the Amazon machine image (AMI) you chose in Launching an EC2 Instance for Cloudera Director on page 23, or find another AMI with a supported operating system.
    d. In the **Tags** field, add one or more tags to associate with the instance.
    e. In the **Security group IDs** field, enter the security group ID you set up in Creating a New Security Group on page 22.
    f. In the **VPC subnet ID** field, enter the ID of the VPC subnet that was created during VPC setup.
    g. Click **Save changes**.



Instance Template

Instance Template name: TEST-TEMPLATE

Instance type: m4.large

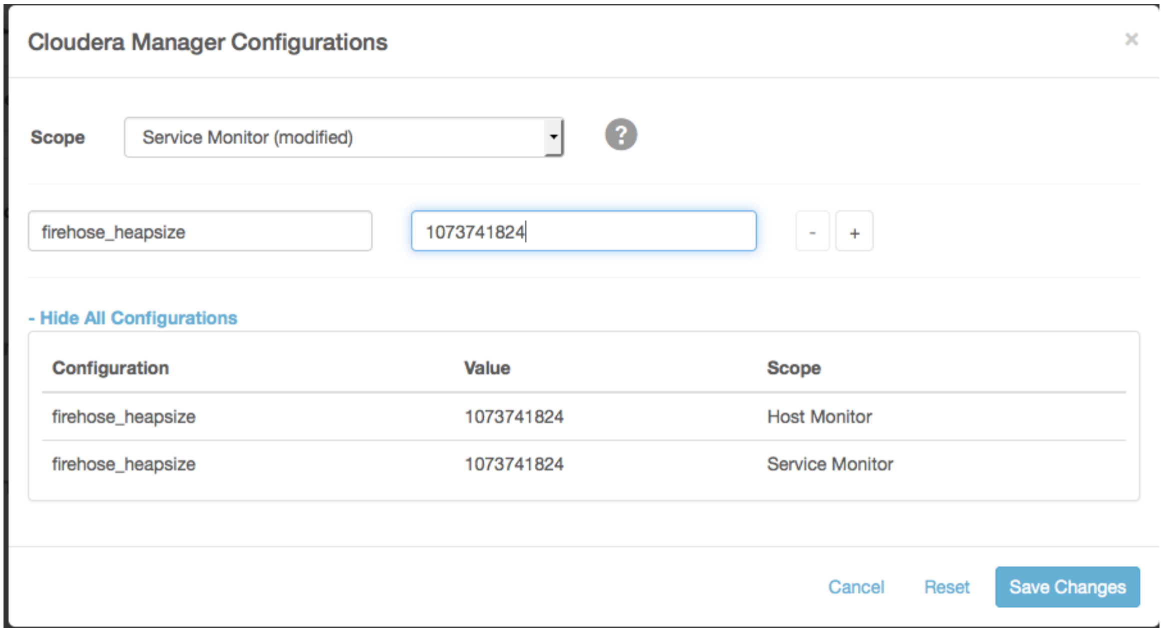
Image (AMI) ID: ami-5dfad518

Tags — Name: Name; Value: test-instance

Security group IDs: sg-cb2c6dae

VPC subnet ID: subnet-52e6f214

8. In the **Add Cloudera Manager** screen, click **Cloudera Manager Configurations**.
9. In the **Cloudera Manager Configurations** modal screen, set the heap size:
    a. In the **Scope** field, select **Host Monitor** and add `firehose_heapsize` and `1073741824` in the respective **Name** and **Value** fields.
    b. Click **+**.
    c. In the **Scope** field, select **Service Monitor** and add `firehose_heapsize` and `1073741824` in the respective **Name** and **Value** fields.
    d. Click **Save Changes**.


Cloudera Manager Configurations
×
Scope
Service Monitor (modified)
?
-
+
- Hide All Configurations
Cancel
Reset
Save Changes

firehose_heapsize 1073741824

Configuration	Value	Scope
firehose_heapsize	1073741824	Host Monitor
firehose_heapsize	1073741824	Service Monitor

10. By default, the version of Cloudera Manager installed depends on the version of Cloudera Director you are using:

    - If you are using Cloudera Director 2.0, the latest released version of Cloudera Manager 5.5 is installed by default.

    To install a version of Cloudera Manager different than the default version, perform the following steps:

    a. In the **Configurations** section, check **Override default Cloudera Manager repository**.
    b. In the **Repository URL** field, enter the repository URL for the version of Cloudera Manager you want to install. Repository URLs for versions of Cloudera Manager 5 have the form http://archive.cloudera.com/cm5/ followed by the operating system, operating system major version, processor architecture, cm (for Cloudera Manager), and the Cloudera Manager major, minor, and (if applicable) maintenance release number. For example, for Cloudera Manager 5.5.4, the repository URL is http://archive.cloudera.com/cm5/redhat/7/x86_64/cm/5.5.4/.

    > **Note:** The Cloudera Manager minor version must be the same as or higher than the CDH minor version. For example, Cloudera Manager 5.5 cannot be used to launch or manage a CDH 5.7 cluster, but Cloudera Manager 5.7 can be used with a CDH 5.7 (or lower) cluster.

    c. In the **Repository Key URL** field, enter the URL for the repository key. Repository key URLs have the same form as repository URLs except they end with the name of the key file instead of the Cloudera Manager version. For example, the repository key URL for any version of Cloudera Manager 5 on any supported version of Red Hat 7 is http://archive.cloudera.com/cm5/redhat/7/x86_64/cm/RPM-GPG-KEY-cloudera.

11. In the **Add Cloudera Manager** screen, click **Continue**.
12. At the **Confirmation** prompt, click **OK** to begin adding a cluster.
13. On the **Add Cluster** screen:

    a. Enter a name for the cluster in the **Cluster name** field.
    b. Select the version of CDH to deploy in the **Version** field.
    c. Enter the version of CDH to deploy in the **Version** field or leave the default value. By default, the version of CDH installed depends on the version of Cloudera Director you are using:

        - If you are using Cloudera Director 2.0, the latest released version of CDH 5.5 is installed by default.

        To install a version of CDH different than the default version, perform the following steps:

        a. Enter the desired CDH version in the **Version** field of the **Products** section. For example, for CDH 5.4.8 enter `5.4.8`.
        b. Scroll down to **Configurations (optional)** and expand the section.

c. Click **Override default parcel repositories**.

d. Enter the repository parcel URL for the version of CDH you want to install. Parcel URLs for versions of CDH 5 have the form http://archive.cloudera.com/cdh5/parcels/, followed by the major, minor, and (if applicable) maintenance release number. For example, the URL for CDH 5.4.8 is http://archive.cloudera.com/cdh5/parcels/5.4.8.

> **Note:** The CDH minor version must not be greater than the Cloudera Manager minor version. For example, CDH 5.7 will not work with Cloudera Manager 5.5, but CDH 5.7 (or lower) will work with Cloudera Manager 5.7.

d. In the **Services** section, select the services you want to install.

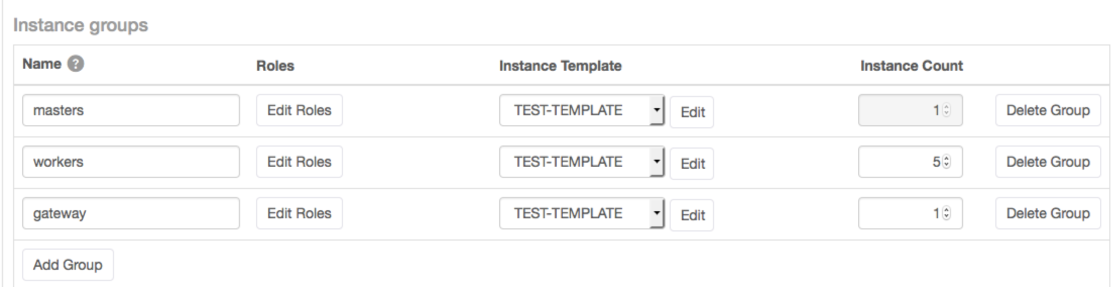
e. In the **Instance groups** area, create a new template for the groups or for each group and the number of instances you want. If you want to use Spot instances for your **workers** group:

a. In the **Create New Instance Template** modal screen, click **Advanced Options**.

b. In the **Spot bid (USD/hr)** field, enter your Spot bid price.

c. Click the **Use Spot instances** checkbox.

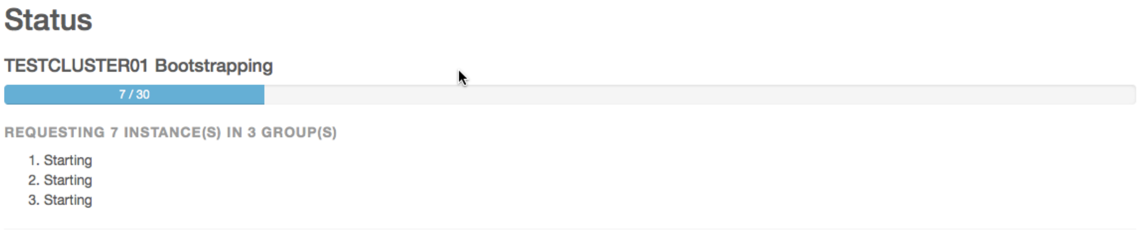
d. Click **Save Changes**.



Instance groups

Name	Roles	Instance Template		Instance Count	
masters	Edit Roles	TEST-TEMPLATE	Edit	1	Delete Group
workers	Edit Roles	TEST-TEMPLATE	Edit	5	Delete Group
gateway	Edit Roles	TEST-TEMPLATE	Edit	1	Delete Group

14. Click **Continue**.
15. At the **Confirmation** prompt, click **OK** to deploy the cluster. Cloudera Director displays a status screen.



Status

TESTCLUSTER01 Bootstrapping

REQUESTING 7 INSTANCE(S) IN 3 GROUP(S)

1. Starting
2. Starting
3. Starting

16. When the cluster is ready, click **Continue**.

    You are finished with the deployment tasks.

Cleaning Up Your AWS Deployment

When you are done testing or using Cloudera Director, terminate your instances to stop incurring charges to your AWS account.

1. In Cloudera Director, terminate each instance in your clusters.

   a. Click an environment name.

   b. In the **Actions** column, select **Terminate Cluster**.

   c. Repeat for each environment you configured.

2. If you want to save anything in Cloudera Director (the configuration file or database, for example), back it up.

3. In the AWS Management Console, terminate the Cloudera Director instance and any other instance Cloudera Director was unable to terminate.
4. If applicable, terminate any external database you configured Cloudera Director to use.

Getting Started on Google Cloud Platform

To use Cloudera Director on Google Cloud Platform, you create a project, start an instance in Google Compute to run Cloudera Director, and create a secure connection. This section details steps for each of these tasks.

> **Important:** Cloudera Director supports preemptible virtual machines. Preemptible virtual machines are short-lived instances that have a lower cost but are subject to reclamation at any time by Google Compute Engine. Because of the possibility of interruption, we recommend that you use preemptible virtual machines only for worker roles in a cluster, not for master or gateway roles. For more information, see the Google Cloud Platform's Preemptible Virtual Machines page.

Creating a Google Cloud Platform Project

To run Cloudera Director on Google Cloud Platform, begin by creating a project:

1. Go to the Google Cloud Platform web site.
2. Click **My console** in the upper-right corner of the screen.
3. Select your Google account, and sign in.

   Your screen is redirected to the **Google Developers Console**.
4. In the **Google Developers Console**, click **Select a project** > **Create a project**.
5. In the **New Project** form, enter a project name, click that you agree to the terms of service, and click **Create**.

> **Note:** To create a project in Google Cloud Platform, first create a billing account or a free trial account, or sign into an existing billing account. To create an account, click **Create new billing account** in the Google Developers Console.

You are ready to configure tools for your project.

Configuring Tools for Your Google Cloud Platform Account

Before installing Cloudera Director, Cloudera recommends that you configure some tools for your Google Cloud Platform account.

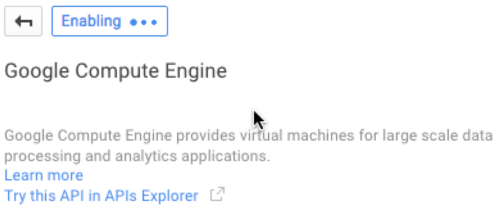
1. Create a service account for Cloudera Director.
2. Create an SSH key.
3. Set up gcloud compute.

Creating a Service Account for Cloudera Director

A service account enables Cloudera Director to authenticate to various Google Cloud Platform services, such as Google Cloud Storage. To create a service account, perform the following steps:

1. Ensure that the Google Compute Engine API is enabled. In the Google Cloud Platform console for your project, click **API Manager**.
2. Click **Compute Engine API** (under **Google Cloud APIs**).
3. If not already enabled, click **Enable API**.
4. At the prompt, click **Enable Billing**.
5. At the prompt, select the billing account and click **Set account**.

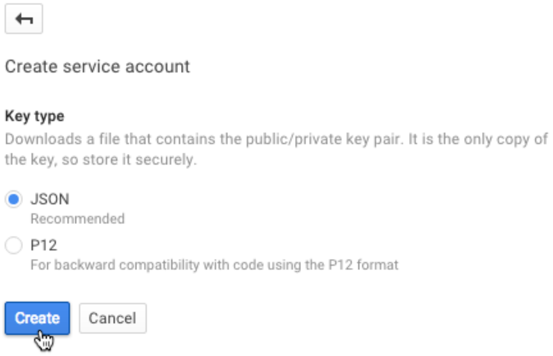
   A status displays, showing that the Google Compute Engine API is enabling.



Google Compute Engine

Google Compute Engine provides virtual machines for large scale data processing and analytics applications.

6. Click **API Manager.**.
7. In the **API Manager** menu, click **Credentials**.
8. In the **Credentials** screen, click **New credentials** > **Service account key**.
9. In the **Create service account key** screen, click **JSON** and click **Create**.



Create service account

Key type

Downloads a file that contains the public/private key pair. It is the only copy of the key, so store it securely.

● JSON
Recommended

○ P12
For backward compatibility with code using the P12 format

You are prompted to save the JSON file to your local machine. Note the location where you download this file. You will be prompted to select this file later, when you create an environment in Cloudera Director.

Creating and Uploading an SSH Key

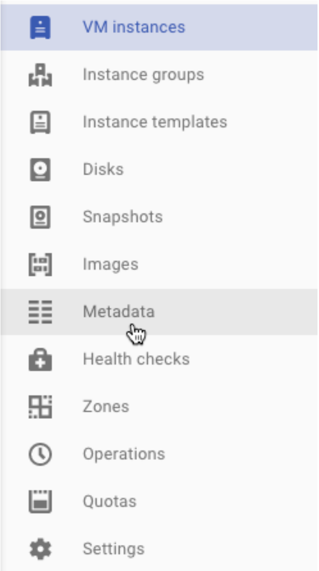
To SSH into an instance using your own terminal (as opposed to the Google Cloud Platform console), you must generate and upload an SSH key.

1. Generate an SSH key using the following command:

```
$ ssh-keygen -f ~/.ssh/my_gcp_keyname -t rsa
```

This generates a public/private key pair.

2. In the **Compute Engine** menu, click **Metadata**.



3. Click the **SSH Keys** tab and click **Add SSH Keys**.
4. Copy your key data into the input box in the following format:

```
protocol public-key-data username@example.com
```

5. Click **Save**. Your public key is now available to all instances in the project.

Installing gcloud compute

Cloudera recommends installing the `gcloud compute` command-line tool because it allows you to manage your Google Compute Engine resources more easily. To install and configure `gcloud compute`, follow the instructions at gcloud compute.

You are ready to create a new VM instance within your project.

Creating a Google Compute Engine VM Instance

Once you have created or selected a project in the Google Developers Console, you can create a new VM instance in your project.

1. In the left side menu of the Google Developers Console, click **Compute** > **Compute Engine** > **VM instances**.
2. Click **Create Instance**.
3. Provide the following values to define your VM instance:

Table 1: VM Instance Values

Name	Description	Details/Restrictions
Name	Name of the instance.	The name must start with a lowercase letter followed by up to 62 lowercase letters, numbers, or hyphens. The name cannot end with a hyphen.
Zone	Where your data is stored.	Some resources can only be used by other resources in the same zone or region. For example, to attach a disk to a VM instance, both resources must reside in the same zone. For more information, see Regions and Zones in the Google GPC documentation.

Name	Description	Details/Restrictions
Machine type	The number of CPUs and amount of memory for your instance.	Cloudera recommends a machine type of at least n1-standard-1 for this Quick Start instance. **Note:** For a production instance, Cloudera recommends at least an n1-standard-2 instance for running Cloudera Director and an n1-highmem-8 instance for running Cloudera Manager and CDH.
Boot disk	The disk to boot from.	Select a preconfigured image with a version of Linux supported for Cloudera Director. For more information about supported Linux versions, see Supported Software and Distributions on page 19.
Boot disk type	The type of boot disk.	For this Quick Start, choose standard persistent disk for less expensive storage space. A solid-state persistent disk (SSD) is better suited to handling high rates of random I/O operations per second (IOPS) or streaming throughput with low latency.
Firewall	Traffic to block.	Leave both HTTP and HTTPS traffic unchecked.
Project access	Access to Google Cloud services.	Leave this unchecked (disabled). These services are not used in this QuickStart.
Management, disk, networking, access & security options	Additional options available when you click the double arrows.	Use the default values for all of these settings.

You are now ready to configure a SOCKS proxy for your instances.

Configuring a SOCKS Proxy for Google Compute Engine

For security purposes, Cloudera recommends that you connect to your cluster using a SOCKS proxy. A SOCKS proxy allows a client to connect directly and securely to a server (the Cloudera Director instance).

To set up a SOCKS proxy, follow the steps in the Google Compute Engine documentation, Securely Connecting to VM Instances, and follow the instructions for setting up a SOCKS proxy over SSH.

Once you have set up a SOCKS proxy, you can install Cloudera Director on your instance.

Installing Cloudera Director Server and Client on Google Compute Engine

Cloudera recommends that you install Cloudera Director server on your cloud provider in the subnet where you will create CDH clusters, because Cloudera Director must have access to the private IP addresses of the instances that it creates. To install Cloudera Director server, perform the following tasks.

Note: You must be either running as root or using sudo to perform these tasks.

RHEL 6 and CentOS 6

1. In the **Compute Engine** > **VM instances** screen, click the **SSH** link next to your instance name.

   This opens a new window.

> **Note:** Alternatively, you can connect to your instance using:
>
> - SSH in a terminal using the following command:
>
> ```
> ssh -i your_key_file -o UserKnownHostsFile=/dev/null \
>  -o CheckHostIP=no -o StrictHostKeyChecking=no user@ip_address
> ```
>
> - The `gcloud compute ssh` command. When you connect to your instance for the first time using the gcloud compute command-line tool, gcloud automatically creates an SSH key and inserts it into the instance.

2. Install a supported version of the Oracle Java Development Kit (JDK) on the Cloudera Director host. Currently, Cloudera Director supports JDK versions 7 and 8. For installation information, see [Java SE Downloads](#).
3. Download Cloudera Director by running the following commands:

```
cd /etc/yum.repos.d/
sudo wget "http://archive.cloudera.com/director/redhat/6/x86_64/director/cloudera-director.repo"
```

4. Install Cloudera Director server by running the following command:

```
sudo yum install cloudera-director-server cloudera-director-client
```

5. Start the Cloudera Director server by running the following command:

```
sudo service cloudera-director-server start
```

6. Save the existing iptables rule set and disable the firewall:

```
sudo service iptables save
sudo chkconfig iptables off
sudo service iptables stop
```

You are now ready to [deploy Cloudera Manager and CDH](#) on the Cloudera Director server.

RHEL 7 and CentOS 7

1. In the **Compute Engine** > **VM instances** screen, click the **SSH** link next to your instance name.

   This opens a new window.

> **Note:** Alternatively, you can connect to your instance using:
>
> - SSH in a terminal using the following command:
>
> ```
> ssh -i your_key_file -o UserKnownHostsFile=/dev/null \
>  -o CheckHostIP=no -o StrictHostKeyChecking=no user@ip_address
> ```
>
> - The `gcloud compute ssh` command. When you connect to your instance for the first time using the gcloud compute command-line tool, gcloud automatically creates an ssh key and inserts it into the instance.

2. Install a supported version of the Oracle Java Development Kit (JDK) on the Cloudera Director host. Currently, Cloudera Director supports JDK versions 7 and 8. For installation information, see [Java SE Downloads](#).
3. Download Cloudera Director by running the following commands:

```
cd /etc/yum.repos.d/
sudo wget "http://archive.cloudera.com/director/redhat/7/x86_64/director/cloudera-director.repo"
```

4. Install Cloudera Director server by running the following command:

```
sudo yum install cloudera-director-server
```

5. Start the Cloudera Director server by running the following command:

```
sudo service cloudera-director-server start
```

6. Disable and stop the firewall with the following commands:

```
sudo systemctl disable firewalld
sudo systemctl stop firewalld
```

You are now ready to deploy Cloudera Manager and CDH on the Cloudera Director server.

Ubuntu

1. In the **Compute Engine** > **VM instances** screen, click the **SSH** link next to your instance name.

   This opens a new window.

   > **Note:** Alternatively, you can connect to your instance using:
   >
   > - SSH in a terminal using the following command:
   >
   > ```
   > ssh -i your_key_file -o UserKnownHostsFile=/dev/null \
   >  -o CheckHostIP=no -o StrictHostKeyChecking=no user@ip_address
   > ```
   >
   > - The `gcloud compute ssh` command. When you connect to your instance for the first time using the gcloud compute command-line tool, gcloud automatically creates an SSH key and inserts it into the instance.

2. Install a supported version of the Oracle Java Development Kit (JDK) on the Cloudera Director host. Currently, Cloudera Director supports JDK versions 7 and 8. For installation information, see Java SE Downloads.
3. Download Cloudera Director by running the following commands:

```
cd /etc/apt/sources.list.d/
sudo wget "http://archive.cloudera.com/director/ubuntu/trusty/amd64/director/cloudera-director.list"
```

4. Add the signing key by running the following command:

```
curl -s "http://archive.cloudera.com/director/ubuntu/trusty/amd64/director/archive.key"
 | sudo apt-key add -
```

5. Install Cloudera Director server by running the following command:

```
apt-get update
apt-get install cloudera-director-server
apt-get install oracle-j2sdk1.7
```

6. Start the Cloudera Director server by running the following command:

```
sudo service cloudera-director-server start
```

7. Save the existing firewall rules and disable the firewall:

```
iptables-save > ~/firewall.rules
sudo service ufw stop
```

You are now ready to deploy Cloudera Manager and CDH on the Cloudera Director server.

Deploying Cloudera Manager and CDH on Google Compute Engine

To deploy Cloudera Manager and CDH on an Google Compute VM instance, begin by creating an environment. The environment defines common settings, like region and key pair, that Cloudera Director uses with Google Cloud Platform. While creating an environment, you are also prompted to deploy its first cluster.

To create an environment:

1. Open a web browser and go to the private IP address of the instance you created in Creating a Google Compute Engine VM Instance on page 36. Include port 7189 in the address. For example:

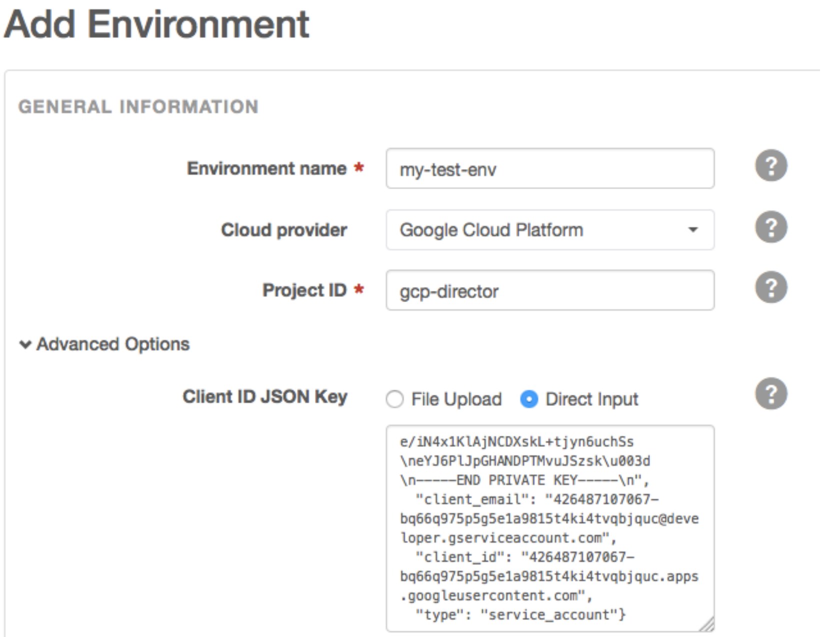
```
http://192.0.2.0:7189
```

2. In the **Cloudera Director** login screen, enter `admin` in both the **Username** and the **Password** fields.
3. In the Cloudera Director **Welcome** screen, click **Let's get started**.

   This opens a wizard for adding an environment, adding Cloudera Manager, and adding a CDH cluster.

4. In the **Add Environment** screen:

   a. Enter a name in the **Environment Name** field.
   b. In the **Cloud provider** field, select **Google Cloud Provider**.
   c. In the **Project ID** field, enter the ID for the project you created in Creating a Google Cloud Platform Project on page 34.
   d. In the **Advanced Options** area, upload or copy the JSON key to the **Client ID JSON Key** field. You created this key in Configuring Tools for Your Google Cloud Platform Account on page 34.



Add Environment

GENERAL INFORMATION

Environment name * my-test-env

Cloud provider: Google Cloud Platform

Project ID * gcp-director

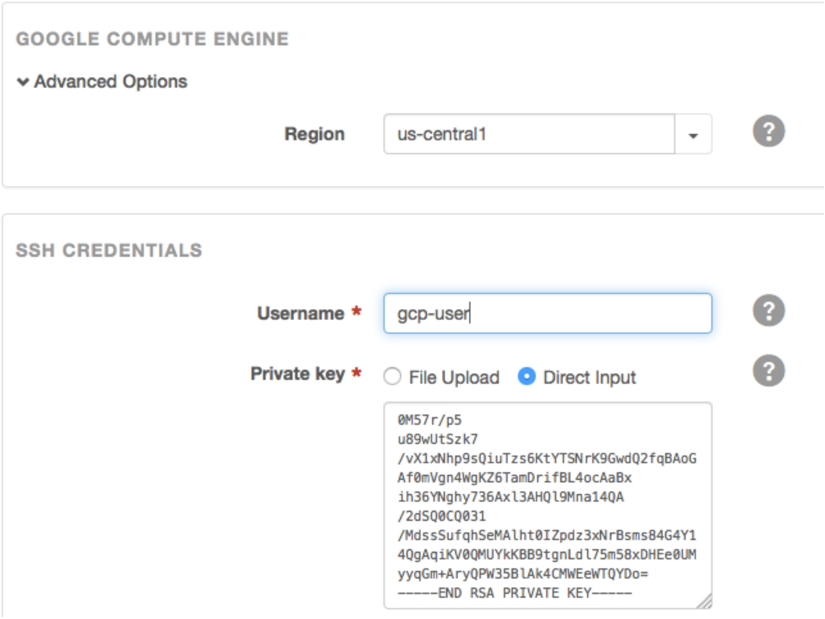
Advanced Options

Client ID JSON Key ○ File Upload ● Direct Input

```
e/iN4x1KlAjNCDXskL+tjyn6uchSs
\neYJ6PlJpGHANDPTMvuJSzsk\u003d
\n-----END PRIVATE KEY-----\n",
  "client_email": "426487107067-
bq66q975p5g5e1a9815t4ki4tvqbjquc@deve
loper.gserviceaccount.com",
  "client_id": "426487107067-
bq66q975p5g5e1a9815t4ki4tvqbjquc.apps
.googleusercontent.com",
  "type": "service_account"}
```

   e. In the **Advanced Options** section, enter the same **region** that your Cloudera Director instance was created in.
   f. In the **SSH Credentials** section:

      - Enter a username in the **Username** field. Google Compute will create the user specified here.
      - Copy the SSH private key you created in Creating and Uploading an SSH Key on page 35 in the **Private key** field.



GOOGLE COMPUTE ENGINE

Advanced Options

Region: us-central1

SSH CREDENTIALS

Username *: gcp-user

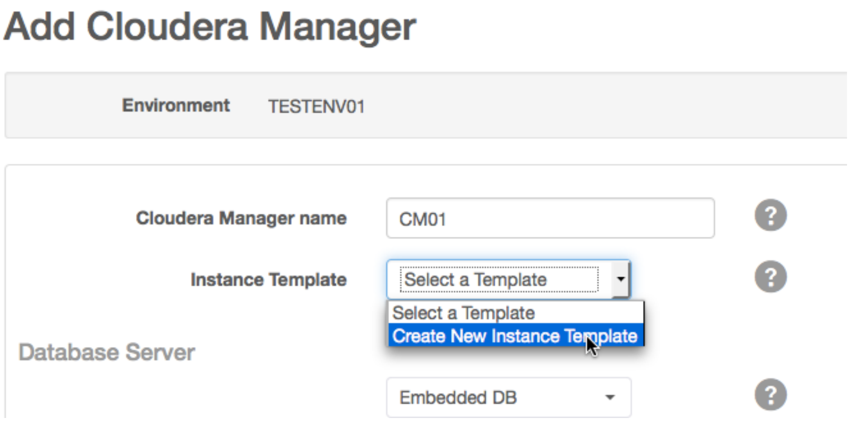
Private key *: ○ File Upload ● Direct Input

```
0M57r/p5
u89wUtSzk7
/vX1xNhp9sQiuTzs6KtYTSNrK9GwdQ2fqBAoG
Af0mVgn4WgKZ6TamDrifBL4ocAaBx
ih36YNghy736Axl3AHQl9Mna14QA
/2dSQ0CQ031
/MdssSufqhSeMAlht0IZpdz3xNrBsms84G4Y1
4QgAqiKV0QMUYkKBB9tgnLdl75m58xDHEe0UM
yyqGm+AryQPW35BlAk4CMWEeWTQYDo=
-----END RSA PRIVATE KEY-----
```

5. Click **Continue** to add Cloudera Manager.
6. In the **Add Cloudera Manager** screen:

    - Enter a name for this deployment of Cloudera Manager in the **Cloudera Manager name** field.
    - In the **Instance Template** field, select **Create New Instance Template**.

      The **Instance Template** modal screen displays.



Add Cloudera Manager

Environment: TESTENV01

Cloudera Manager name: CM01

Instance Template: Select a Template

Database Server: Embedded DB

7. In the **Instance Template** modal screen, do the following:

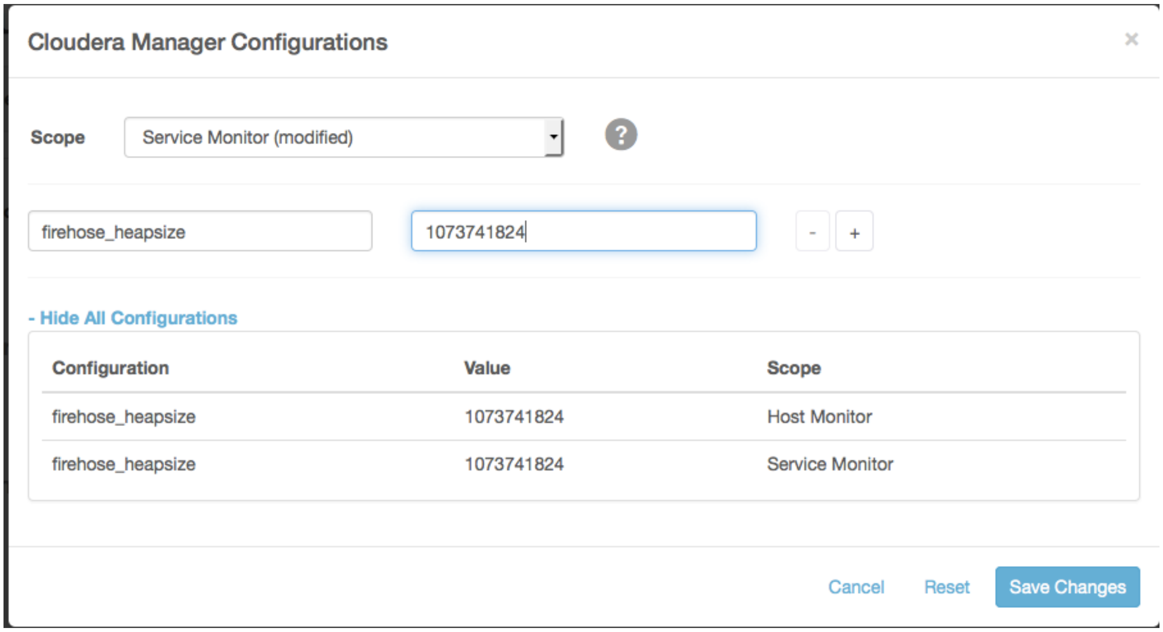
    - In the **Instance Template name** field, enter a name for the template.
    - In the **Instance type** field, select **n1-highmem-4** or **n1-highmem-8**.
    - In the **Machine type** field, enter the machine type you chose in Creating a Google Compute Engine VM Instance on page 36.
    - In the **Tags** field, add one or more tags to associate with the instance.
    - Click **Save changes**.

8. In the **Add Cloudera Manager** screen, click **Cloudera Manager Configurations**.

   The **Cloudera Manager Configurations** modal screen displays.

9. In the **Cloudera Manager Configurations** modal screen, set the heap size:

- In the **Scope** field, select **Host Monitor** and add `firehose_heapsize` and `1073741824` in the respective **Name** and **Value** fields.
- Click **+**.
- In the **Scope** field, select **Service Monitor** and add `firehose_heapsize` and `1073741824` in the respective **Name** and **Value** fields.
- Click **Save Changes**.



Cloudera Manager Configurations

Scope: Service Monitor (modified)

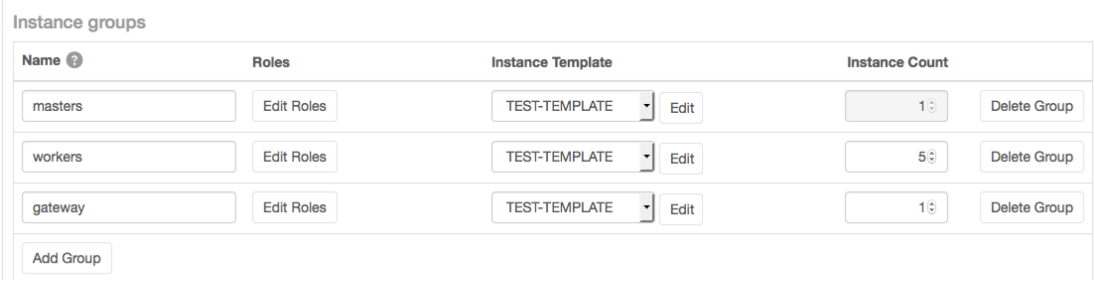
firehose_heapsize | 1073741824

- Hide All Configurations

Configuration	Value	Scope
firehose_heapsize	1073741824	Host Monitor
firehose_heapsize	1073741824	Service Monitor

10. In the **Add Cloudera Manager** screen, click **Continue**.
11. At the **Confirmation** prompt, click **OK** to begin adding a cluster.
12. On the **Add Cluster** screen:

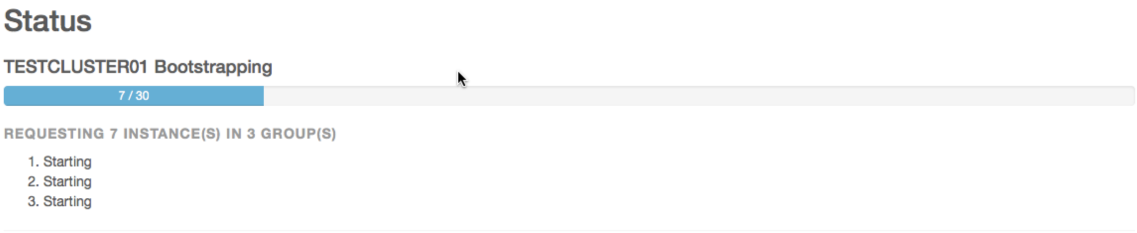
- Enter a name for the cluster in the **Cluster name** field.
- Select the version of CDH to deploy in the **Version** field.
- In the **Services** section, select the services you want to install.
- In the **Instance groups** area, create a new template for the groups or for each group and the number of instances you want.



Instance groups

Name	Roles	Instance Template		Instance Count	
masters	Edit Roles	TEST-TEMPLATE	Edit	1	Delete Group
workers	Edit Roles	TEST-TEMPLATE	Edit	5	Delete Group
gateway	Edit Roles	TEST-TEMPLATE	Edit	1	Delete Group

13. Click **Continue**.
14. At the **Confirmation** prompt, click **OK** to deploy the cluster. Cloudera Director displays a status screen.



Status

TESTCLUSTER01 Bootstrapping

7 / 30

REQUESTING 7 INSTANCE(S) IN 3 GROUP(S)

1. Starting
2. Starting
3. Starting

15. When the cluster is ready, click **Continue**.

    You are finished with the deployment tasks.

Cleaning Up Your Google Cloud Platform Deployment

When you are done testing or using Cloudera Director, terminate your instances to stop incurring charges to your Google Cloud Platform account.

1. In Cloudera Director, terminate each instance in your clusters.
   - Click an environment name.
   - In the **Actions** column, select **Terminate Cluster**.
   - Repeat for each environment you configured.
2. If you want to save anything in Cloudera Director (the configuration file or database, for example), back it up.
3. In the Google Compute Console, delete the Cloudera Director instance and any other instance Cloudera Director was unable to delete.
4. If applicable, delete any external database you configured Cloudera Director to use.

Cloudera Director Client

The Cloudera Director client works well for proof-of-concept demonstrations, development work, and infrequent usage. Deployment through the Cloudera Director client involves installing on an instance, editing a configuration file, and running Cloudera Director from the command line. Cloudera Director client installation, configuration, and use are described in the following topics.

Installing Cloudera Director Client

To install Cloudera Director client in standalone mode, without Cloudera Director server, perform the tasks below. You must be either running as root or using sudo to perform these tasks.

For instructions on installing Cloudera Director client together with Cloudera Director server, see the following:

- For AWS, see Installing Cloudera Director Server and Client on the EC2 Instance on page 25.
- For Google Cloud Platform, see Installing Cloudera Director Server and Client on Google Compute Engine on page 37.

> **Important:** Cloudera Director requires a JDK. For more information, see Supported Software and Distributions on page 19.

1. Install a supported version of the Oracle Java Development Kit (JDK) on the Cloudera Director host. Currently, Cloudera Director supports JDK versions 7 and 8. For installation information, see Java SE Downloads.
2. Download Cloudera Director by running the correct commands for your distribution.

   - For RHEL 6 and CentOS 6:

```
cd /etc/yum.repos.d/
sudo wget "http://archive.cloudera.com/director/redhat/6/x86_64/director/cloudera-director.repo"
```

   - For RHEL 7 and CentOS 7:

```
cd /etc/yum.repos.d/
sudo wget "http://archive.cloudera.com/director/redhat/7/x86_64/director/cloudera-director.repo"
```

   - For Ubuntu 14.04 (Trusty Tahr):

```
cd /etc/apt/sources.list.d
sudo wget "http://archive.cloudera.com/director/ubuntu/trusty/amd64/director/cloudera-director.list"
```

3. Add the signing key.

   - For RHEL 6, CentOS 6 this step is not required. Continue to the next step.
   - For RHEL 7, CentOS 7 this step is not required. Continue to the next step.
   - For Ubuntu 14.04 (Trusty Tahr), run the following command:

```
curl -s "http://archive.cloudera.com/director/ubuntu/trusty/amd64/director/archive.key"
 | sudo apt-key add -
```

4. Install Cloudera Director client by running the correct command for your distribution.

   - For RHEL 6 and CentOS 6:

```
yum install cloudera-director-client
```

- For RHEL 7 and CentOS 7:

```
yum install cloudera-director-client
```

- For Ubuntu 14.04 (Trusty Tahr):

```
apt-get install cloudera-director-client
```

Provisioning a Cluster on AWS

The configuration file contains information Cloudera Director needs to operate and settings that define your cluster.

Sample configuration files are found either in `/usr/lib64/cloudera-director/client` or `/usr/lib/cloudera-director/client`, depending on the operating system you are using. Copy the sample files to your home directory before editing them.

To modify the configuration file:

1. Rename the `aws.simple.conf` file to `cluster.conf`. For advanced cluster configuration, use `aws.reference.conf`.

   > **Note:** The configuration file must use the `.conf` file extension.

2. Open `cluster.conf` with a text editor.
3. Configure the basic settings:
   - **name** - change to something that makes the cluster easy to identify.
   - **id** - leave this set to aws.
   - **accessKeyId** - AWS access key ID. Make sure the value is enclosed in double quotes.
   - **secretAccessKey** - AWS secret access key. Make sure the value is enclosed in double quotes.
   - **region** - specify the region (for example, us-west-2).
   - **keyName** - specify the name of the key pair used to start the cluster launcher. Key pairs are region-specific. For example, if you create a key pair (or import one you have created) in US-West-2, it will not be available in US-West-1. For information on creating key pairs in Amazon EC2 or importing existing key pairs, see Amazon EC2 Key Pairs.
   - **subnetId** - ID of the subnet that you noted earlier.
   - **securityGroupsIds** - ID of the security group that you noted earlier. Use the ID of the group, not the name (for example, sg-b139d3d3, not default).
   - **instanceNamePrefix** - enter the prefix to prepend to each instance's name.
   - **image** - specifies the AMI to use. Cloudera recommends Red Hat Enterprise Linux 6.4 (64bit). To find the correct AMI for the selected region, visit the Red Hat AWS Partner page.

   > **Note:** If you use your own AMI, make sure to disable any software that prevents the instance from rebooting during the deployment of the cluster.

4. Configure the following cluster settings:
   a. You can only use Cloudera Manager 5. No changes are needed for repository and repository key URLs and you must set the parcel repositories to match the CDH and Impala versions you plan to install.
   b. Specify services to start on the cluster. For a complete list of allowed values, see the Cloudera Manager API Service Types.

> **Note:** Include Flume in the list of services only when customizing role assignments. See the configuration file (`aws.reference.conf`) included in the Cloudera Director download for examples on how to configure customized role assignments. If Flume is required, it should be excluded from the list of services in the configuration file and added as a service using Cloudera Manager UI or API after the cluster is deployed. When adding Flume as a service, you must assign Flume agents (which Cloudera Manager does not do automatically).

c. Specify the number of instances in the cluster.

5. Save the file and exit.

> **Note:** If your root disk drive is larger than all the other drives on the machine, Cloudera Manager automatically installs HDFS on the root drive. You can change this behavior with an explicit override in the configs {} block within the cluster {} section of the configuration file.

Running Cloudera Director Client

After you modify the configuration file, you can run Cloudera Director client. There are two ways of running the Cloudera Director client:

- In standalone mode, using the `bootstrap` command. Clusters created using the `bootstrap` command cannot be managed using the Cloudera Director UI. The information below on this page concerns running the client in standalone mode.
- If you already have a server, you can run the client against the server using the commands `bootstrap-remote` and `terminate-remote`. Only clusters created with the `bootstrap-remote` command can be managed using the Cloudera Director UI. For more information on using the client to deploy clusters on the server, see Submitting a Cluster Configuration File.

> **Note:** If you are restarting Cloudera Director client, you are prompted to resume from where the client stopped or start over. If you made changes to the configuration file between deployments, or if you need to start the run from scratch, you should start over.

1. From the cluster launcher, enter the following:

```
[ec2-user@ip-10-1-1-18]$ cloudera-director bootstrap cluster.conf
```

Cloudera Director displays output similar to the following:

```
Installing Cloudera Manager ...
* Starting ... done
* Requesting an instance for Cloudera Manager .................. done
* Inspecting capabilities of 10.1.1.194 .............. done
* Normalizing 10.1.1.194 ................... done
* Installing python (1/4) .... done
* Installing ntp (2/4) .... done
* Installing curl (3/4) .... done
* Installing wget (4/4) ................ done
* Installing repositories for Cloudera Manager ............... done
* Installing jdk (1/5) ..... done
* Installing cloudera-manager-daemons (2/5) ..... done
* Installing cloudera-manager-server (3/5) ..... done
* Installing cloudera-manager-server-db-2 (4/5) ..... done
* Installing cloudera-manager-agent (5/5) .... done
* Starting embedded PostgreSQL database ..... done
* Starting Cloudera Manager server ...... done
* Waiting for Cloudera Manager server to start .... done
* Configuring Cloudera Manager ..... done
* Starting Cloudera Management Services ...... done
* Inspecting capabilities of 10.1.1.194 ......... done
* Done ...
Cloudera Manager ready.
Creating cluster C5-Sandbox-AWS ...
* Starting ... done
* Requesting 3 instance(s) .......... done
* Inspecting capabilities of new instance(s) ....... done
* Running basic normalization scripts ......... done
* Registering instance(s) with Cloudera Manager .... done
* Waiting for Cloudera Manager to deploy agents on instances ... done
* Creating CDH5 cluster using the new nodes ...... done
* Downloading CDH-5.4.0-1.cdh5.4.0.p0.26 parcel ..... done
* Distributing CDH-5.4.0-1.cdh5.4.0.p0.26 parcel ... done
* Activating CDH-5.4.0-1.cdh5.4.0.p0.26 parcel ...... done
* Done ...
Cluster ready.
```

> **Note:** If you have a large root disk partition or if you are using a hardware virtual machine (HVM) AMI, the instances can take a long time to reboot. Cloudera Manager can take 20-25 minutes to become available.

2. To monitor Cloudera Director, log in to the cluster launcher and view the application log:

```
 $ ssh ec2-user@54.186.148.151
Last login: Tue Mar 18 20:33:38 2014 from 65.50.196.130
[ec2-user@ip-10-1-1-18]$ tail -f ~/.cloudera-director/logs/application.log
[...]
```

> **Note:** If you have deployment issues and need help troubleshooting, be careful when distributing the state.h2.db or application.log files. They contain sensitive information, such as your AWS keys and SSH keys.

Using the Command Line Interface

The command-line interface (CLI) includes commands and options for running Cloudera Director locally or for bootstrapping or terminating Cloudera Director on a remote server.

Local commands

The commands in this table can be used when running Cloudera Manager locally (in standalone mode):

Command	Description	Options
bootstrap	Bootstraps an environment, deployment, and cluster locally (in standalone mode).	lp.bootstrap.resume.policy=interactive|resume|restart • If progress was already made bootstrapping, this option determines if the process will automatically resume (resume), start over from scratch (restart), or ask the user (interactive). The default value is interactive.
status	Reports status on various entities, including deployment, cluster,	

Command	Description	Options
	Cloudera Manager instance, and cluster instances.	
`terminate`	Terminates a cluster and deployment locally (in standalone mode).	`lp.terminate.assumeYes=true|false` - This property determines if the user must explicitly confirm termination (`false`) or if confirmation is assumed (`true`). The default value is `false`.
`update`	Updates an environment, deployment, and cluster locally (in standalone mode).	
`validate`	Validates a configuration locally (in standalone mode).	`lp.validate.dumpTemplates=true|false` - If `true`, prints out parsed configuration information. The default value is `false`.

Remote commands

The following CLI commands can be used to bootstrap or terminate Cloudera Director on a remote server:

Command	Description	Option

`bootstrap-remote`	Bootstraps an environment, deployment, and cluster on a remote server.	`lp.remote.hostAndPort=`*`host[:port]`*
		- Default value: `localhost:7189`
		`lp.remote.username=`*`Cloudera Director server username`*
		`lp.remote.password=`*`Cloudera Director server password`*

`terminate-remote`	Terminates a cluster and deployment on a remote server.	`lp.remote.hostAndPort=`*`host[:port]`*
		- Default value: `localhost:7189`
		`lp.remote.username=`*`Cloudera Director server username`*
		`lp.remote.password=`*`Cloudera Director server password`*
		`lp.remote.terminate.assumeYes=true|false`
		- This property determines if the user must explicitly confirm termination (`false`) or if confirmation is assumed (`true`). The default value is `false`.

Connecting to Cloudera Manager with Cloudera Director Client

After the cluster is ready, log in to Cloudera Manager and access the cluster.

To access Cloudera Manager:

1. Use the status command to get the host IP address of Cloudera Manager:

```
$ cloudera-director status cluster.conf
```

Cloudera Director displays output similar to the following:

```
Cloudera Director 2.0.0 initializing ...

Cloudera Manager:
* Instance: 10.0.0.110 Owner=wintermute,Group=manager
* Shell: ssh -i /root/.ssh/launchpad root@10.0.0.110

Cluster Instances:
* Instance 1: 10.0.0.39 Owner=wintermute,Group=master
* Shell 1: ssh -i /root/.ssh/launchpad root@10.0.0.39

* Instance 2: 10.0.0.148 Owner=wintermute,Group=slave
* Shell 2: ssh -i /root/.ssh/launchpad root@10.0.0.148

* Instance 3: 10.0.0.150 Owner=wintermute,Group=slave
* Shell 3: ssh -i /root/.ssh/launchpad root@10.0.0.150

* Instance 4: 10.0.0.147 Owner=wintermute,Group=slave
* Shell 4: ssh -i /root/.ssh/launchpad root@10.0.0.147

* Instance 5: 10.0.0.149 Owner=wintermute,Group=slave
* Shell 5: ssh -i /root/.ssh/launchpad root@10.0.0.149

* Instance 6: 10.0.0.151 Owner=wintermute,Group=slave
* Shell 6: ssh -i /root/.ssh/launchpad root@10.0.0.151

* Instance 7: 10.0.0.254 Owner=wintermute,Group=gateway
* Shell 7: ssh -i /root/.ssh/launchpad root@10.0.0.254

* Instance 8: 10.0.0.32 Owner=wintermute,Group=master
* Shell 8: ssh -i /root/.ssh/launchpad root@10.0.0.32

* Instance 9: 10.0.0.22 Owner=wintermute,Group=master
* Shell 9: ssh -i /root/.ssh/launchpad root@10.0.0.22

Launchpad Gateway:
* Gateway Shell: ssh -i /path/to/launchpad/host/keyName.pem -L 7180:10.0.0.110:7180 -L
 7187:10.0.0.110:7187 root@ec2-54-77-57-3.eu-west-1.compute.amazonaws.com

Cluster Consoles:
* Cloudera Manager: http://localhost:7180
* Cloudera Navigator: http://localhost:7187
```

In this example, the host IP address is 10.0.0.110.

2. Change to the directory where your *`keyfile`*`.`pem file is located. Then, route the connection over SSH:

```
$ ssh -L 7180:cm-host-private-ip:7180 ec2-user@cm-host-public-ip
# go to http://localhost:7180 in your browser and login with admin/admin
```

> **Note:** If you get a permission error, add the `.`pem file from the command line:
>
> ```
> $ ssh -i <keyfile.pem> -L 7180:cm-host-private-ip:7180
> ec2-user@cm-host-public-ip
> ```

3. Open a web browser and enter `http://localhost:7180` to connect to Cloudera Manager. Use admin as both the username and password.
4. Add any additional services to the cluster. The CDH 5 parcel was already distributed by Cloudera Director.

Modifying a Cluster with the Configuration File

This section describes how to make changes to the cluster through Cloudera Director, using the client and the configuration file.

Growing or Shrinking a Cluster with the Configuration File

After launching a cluster, you can add or remove instances:

1. Open the `cluster.conf` file that you used to launch the cluster.
2. Change the value for the type of instance you want to change. For example, the following increases the number of workers to 15:

```
workers {
      count: 15
      minCount: 5

      instance: ${instances.hs18} {
        tags {
          group: worker
        }
      }
```

3. Enter the following command:

```
cloudera-director update cluster.conf
```

   Cloudera Director increases the number of worker instances.

4. Assign roles to the new master instances through Cloudera Manager. Cloudera Director does not automatically assign roles.

Rebalancing the Cluster After Adding or Removing Hosts

After hosts have been added to or removed from a cluster, HDFS data is likely to be distributed unevenly across DataNodes. Cloudera Director does not rebalance HDFS when you add hosts or remove them from the cluster, so after growing or shrinking the cluster, you must perform manual rebalances in Cloudera Manager, as described in the Cloudera Manager documentation, HDFS Balancers.

The need for rebalancing depends on the amount of data in HDFS and the number of hosts added or removed during the cluster. Cloudera Director decommissions hosts before removing them from the cluster during a shrink operation. As part of decommissioning a DataNode, Cloudera Manager will move all the blocks from that host to other hosts so that the replication factor will be maintained even after the hosts are decommissioned. So there is no risk of data loss if the cluster is shrunk by more than two instances at a time. Rebalancing is necessary so that the blocks are placed in an optimal manner and is not required when a small number of hosts have been removed from a cluster, but only when there has been a large movement of data.

Managing Cloudera Manager Instances with Cloudera Director Server

The Cloudera Director server is designed to run in a centralized setup, managing multiple Cloudera Manager instances and CDH clusters, with multiple users and user accounts. The server works well for launching and managing large numbers of clusters in a production environment. Cloudera Director server configuration and use are described in the following topics.

Submitting a Cluster Configuration File

In Cloudera Director, you can deploy clusters in two ways:

- Through the Cloudera Director server UI.
- Through the Cloudera Director client, which you can use to send a configuration file that the server uses for cluster deployment. The configuration file provides advanced options not currently available in the server UI.

This section describes the second of these ways, using the Cloudera Director client to submit a configuration file. The configuration file will be applied to the cluster and managed by the Cloudera Director server.

When you submit a cluster configuration from a Cloudera Director client to the Cloudera Director server, all communications are transmitted in the clear (including the AWS credentials). If the client and server communicate over the Internet, use a VPN for security.

> **Note:** If you create tags in the configuration file for AWS or Google Cloud Platform instance metadata or for service or role configurations, special characters, such as periods and colons, must be enclosed in double quotes. This includes some characters required by the HOCON format. For example, a tag value that would require quoting is `"company:department:team"`. See the AWS and Google Cloud Platform documentation for information about which special characters are supported on these cloud platforms in instance metadata tags.

To submit a cluster configuration file to the Cloudera Director server, follow these steps:

1. Create a configuration file. See Provisioning a Cluster on AWS on page 45.
2. Install the latest version of the Cloudera Director client from the Cloudera Director Download Page.
3. Enter the following command:

```
cloudera-director bootstrap-remote myconfig.conf --lp.remote.username=admin
--lp.remote.password=admin --lp.remote.hostAndPort=host:port
```

   myconfig.conf is the name of your configuration file, *admin* is the default value for both the username and password for the Admin account (enter your actual values), *host* is the hostname or IP address of the instance on which Cloudera Director server is running, and *port* is the port on which it is listening.

   Both the Cloudera Director client (in the terminal where the `bootstrap-remote` command was issued) and the Cloudera Director server UI display the status throughout the deployment process.

Deploying Clusters in an Existing Environment

If you already configured an environment, you can easily deploy a new cluster:

1. Log in to Cloudera Director. For example, http://example.com:7189.
2. Click **Add Cluster**, and then select an environment from the **Environment** list box. .
3. Select a Cloudera Manager from the **Cloudera Manager** list box.
4. To clone an existing cluster, select **Clone from existing** and select a cluster. To specify cluster settings, select **Create from scratch**.

5. Enter a name for the cluster in the **Cluster name** field.
6. Enter the version of CDH to deploy in the **Version** field or leave the default value. By default, the version of CDH that will be installed depends on the version of Cloudera Director you are using:

    - If you are using Cloudera Director 2.0, the latest released version of Cloudera Manager/CDH 5.5 will be installed by default.

    To install an earlier or later version of CDH than the default version, perform the following steps:

    a. Enter the desired CDH version in the **Version** field of the **Products** section. For example, for CDH 5.4.8 enter `5.4.8`.
    b. Scroll down to **Configurations (optional)** and expand the section.
    c. Click **Override default parcel repositories**.
    d. Enter the repository parcel URL for the version of CDH you want to install. Parcel URLs for versions of CDH 5 take the form http://archive.cloudera.com/cdh5/parcels/, followed by the major, minor, and (if applicable) dot release number. For example, the URL for CDH 5.4.8 is http://archive.cloudera.com/cdh5/parcels/5.4.8.

    > **Note:** The CDH minor version must not be greater than the Cloudera Manager minor version. For example, CDH 5.7 will not work with Cloudera Manager 5.5, but CDH 5.7 (or lower) will work with Cloudera Manager 5.7.

7. Select the type of cluster to deploy from **Services**.
8. Select the numbers of masters, workers, and gateways to deploy. Then, select an instance template for each or create one or more new templates.
9. When you are finished, click **Continue**. When prompted for confirmation, click **OK** to confirm.

    Cloudera Director begins deploying the cluster.

    > **Note:** If your root disk drive is larger than all the other drives on the machine, Cloudera Manager automatically installs HDFS on the root drive.

Cloudera Manager Health Information

The following Cloudera Manager health information is available through Cloudera Director server:

- Host health
- Service health
- Cluster health

The health value is displayed in the **Status** column for each entity, when health information is available. Possible health values are:

- **Disabled** - Health collection has been disabled on Cloudera Manager.
- **Not Available** - Cloudera Director does not currently have health information, or a health has "expired."
- **Bad** - Cloudera Manager reports the health as bad.
- **Concerning** - Cloudera Manager reports the health as concerning.
- **Good** - Cloudera Manager reports the health as good.

You can configure the health cache with the following settings in the `application.properties` file:

- **`lp.cache.health.pollingRateInMilliseconds`** - How often the Cloudera Director server polls Cloudera Manager for health information. The default value is 30,000 ms (30 seconds). To disable health collection, set `lp.cache.health.pollingRateInMilliseconds` to 0.
- **`lp.cache.health.numberOfHealthCacheExecutorThreads`** - The number of threads used to simultaneously request health information from Cloudera Manager. the default value is 5.

- **`lp.cache.health.expirationMultiplier`** - Used to determine if a health value is stale. If the health value has not been updated in `pollingRateInMilliseconds * expirationMultiplier` milliseconds, then the health value is considered stale and is reported to the UI as NOT_AVAILABLE. Using the default settings, for example, if health has not been reported in 2 * 30,000 milliseconds = 60 seconds, it becomes stale. The default value is 2.

Note: Cloudera Manager health is collected by Cloudera Director server only, not by Cloudera Director client.

Opening Cloudera Manager

After deploying a cluster, you can manage it using Cloudera Manager:

1. Log in to Cloudera Director. For example, http://example.com:7189.

   Cloudera Director opens with a list of clusters.

2. Locate the cluster to manage and click its Cloudera Manager. The link is available when Cloudera Manager is ready.
3. On the Cloudera Manager Login page, enter your credentials and click **Login**.

   Cloudera Manager opens.

Creating and Modifying Clusters with the Cloudera Director UI

Before initially launching a CDH cluster, you can use the Cloudera Director UI to add, delete, or modify the default roles and instance groups. You can also add, remove, or repair instances in an existing cluster.

Configuring Instance Groups During Cluster Creation

An *instance group* is a collection of roles that are installed together on one or more instances. When Cloudera Director creates a Cloudera Manager cluster, it includes three default instance groups: masters, workers, and gateway. Each of these instance groups contains roles of the type represented by that instance group, for the CDH services selected for the cluster. For example, if your cluster includes HDFS and YARN, the masters instance group includes the following roles:

- For HDFS - NameNode, SecondaryNameNode, Balancer
- For YARN - ResourceManager, JobHistory Server

The workers instance group will include the following roles:

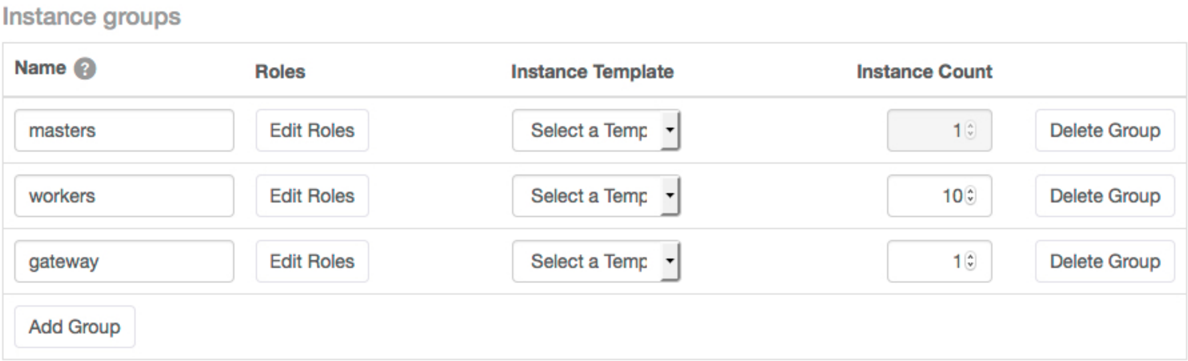
- For HDFS - DataNode
- For YARN - NodeManager

The gateway instance group includes a gateway role for HDFS and another for YARN.

For an introduction to master, worker, and gateway roles, see the Cloudera Manager 5 Overview .

Although the default instance groups are automatically configured with roles of a given type (masters, workers, or gateway), you can add any kind of role to any instance group.

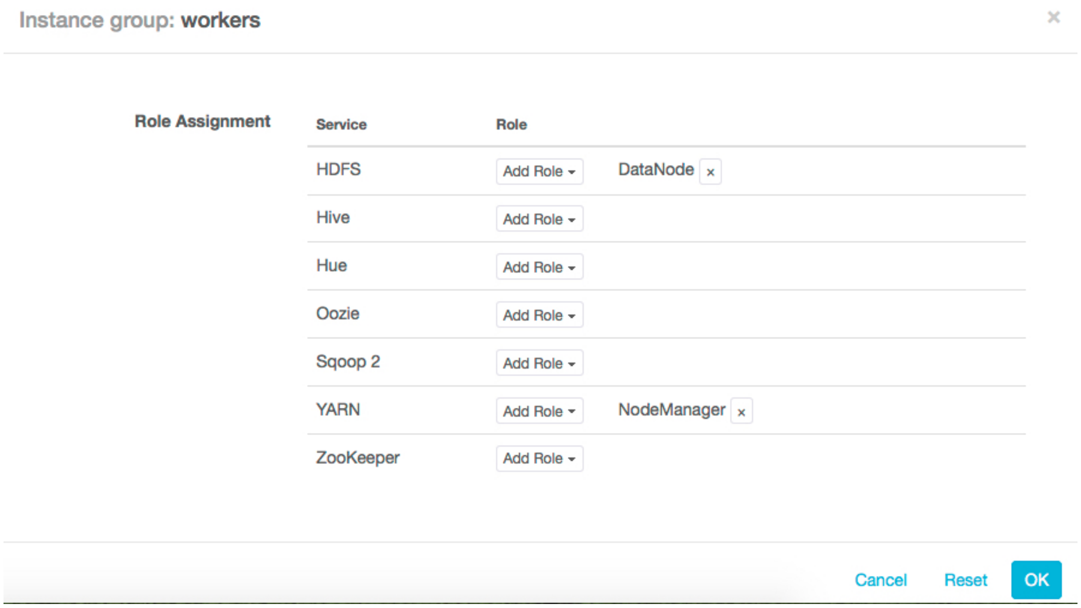
When you create a cluster with Cloudera Director, a default set of instance groups and roles, based on the CDH services you include, is displayed in the Instance Groups section of the Add Cluster page:



Instance groups

Name	Roles	Instance Template	Instance Count	
masters	Edit Roles	Select a Temp	1	Delete Group
workers	Edit Roles	Select a Temp	10	Delete Group
gateway	Edit Roles	Select a Temp	1	Delete Group

By clicking **Edit Roles**, you can see the roles included in each instance group. These roles will be installed on each instance running that instance group. In this example, by clicking **Edit Roles** for the workers instance group above, you can see that each of the 10 instances that will be installed for the workers instance group will include two roles, an HDFS DataNode and a YARN NodeManager:



Instance group: workers

Role Assignment

Service	Role	
HDFS	Add Role	DataNode ×
Hive	Add Role	
Hue	Add Role	
Oozie	Add Role	
Sqoop 2	Add Role	
YARN	Add Role	NodeManager ×
ZooKeeper	Add Role	

You can modify the default configuration of instance groups during cluster creation by doing the following:

- Change the number of instances for an instance group by clicking the up or down arrows.
- Delete an instance group by clicking **Delete Group** at the right end of the row for that instance group.
- Add roles to an existing instance group by clicking **Edit Roles** and then **Add Role**. Available roles for the services in the cluster are displayed. Click a role to add it to the instance group.
- Add another instance group to the cluster by clicking **Add Group**, entering a name for the instance group and assigning roles to it, selecting an instance template, and clicking the up or down arrows to choose the number of instances to install.

Modifying the Number of Instances in an Existing Cluster

Cloudera Director can grow or shrink the size of an existing cluster by adding or removing instances.

Adding Instances to a Cluster

1. Log in to Cloudera Director at `http://director-server-hostname:7189`. Cloudera Director opens on the All Environments page, which displays the current environments, deployments, and clusters. Click the cluster you want to modify.

2. Click **Modify Cluster** to the right of the cluster name. The Modify Cluster page displays the gateway, masters, and workers instance groups and any additional instance groups that have been added to the cluster, with the current number of instances in each instance group.
3. You can add instances to an existing instance group or create a new instance group and add roles to it.
   - To add instances to an existing instance group, click **Edit** to the right of the instance group and click the up or down arrows in the **Add Instances** section to increase the number of workers and gateways to the desired size. Each new instance will contain the same roles as the existing instances of that group.
   - To create a new instance group, click **Add Group**, enter a name for the instance group, assign roles to it, select an instance template, and click the up or down arrows to choose the desired number of instances of that group to add.

> **Note:** Cloudera recommends rebalancing the cluster through Cloudera Manager if you increase the number of HDFS DataNodes by 30% or more. For more information, see Rebalancing the Cluster After Adding or Removing Instances on page 56.

Removing Instances from a Cluster

1. Log in to Cloudera Director at `http://director-server-hostname:7189`.

   Cloudera Director opens on the All Environments page, which displays the current environments, deployments, and clusters. Click the cluster you want to modify.

2. Click **Modify Cluster** to the right of the cluster name. The Modify Cluster page displays the gateway, masters, and workers instance groups and any additional instance groups that have been added to the cluster, with the current number of instances in each instance group.
3. You can remove an entire instance group, including all of its instances, or remove individual instances from an instance group:
   - To remove an entire instance group, click **Delete Group** at the right end of the row for that instance group.
   - To remove individual instances from an instance group, click **Edit** near the right end of the row for the instance group. Click the checkbox for each instance you want to remove, and click the **Delete** button. The instances you select display an action status of **To be deleted**.
4. Click **OK** to continue, **Reset** to unselect the selected instances and make a new selection, or **Cancel** to stop without making any changes.
5. Click **Continue** to confirm and delete the selected instances.

> **Note:**
>
> - It is important to maintain the number of HDFS DataNode role instances at or above the HDFS replication factor configured for the cluster. By default, Cloudera recommends a replication factor of three.
> - Cloudera Director decommissions instances before removing them from the cluster. When decommissioning an HDFS DataNode, Cloudera Manager moves all the blocks from that instance to other instances so that the replication factor is maintained, and there is no risk of data loss.
> - You cannot delete an instance with an HDFS DataNode if the number of DataNodes equals the replication factor (which by default is three) of any file stored in HDFS. For example, if the replication factor of any file is three, and you have three DataNodes, you cannot delete an instance with a DataNode.
> - Cloudera recommends rebalancing the cluster through Cloudera Manager if you reduce the number of HDFS DataNodes by 30% or more. For more information, see Rebalancing the Cluster After Adding or Removing Instances on page 56.

Rebalancing the Cluster After Adding or Removing Instances

After you add or remove instances from a cluster, HDFS data is likely to be distributed unevenly across DataNodes. Cloudera Director does not rebalance HDFS when you add instances or remove them from the cluster. If you need to rebalance the cluster, you must do so manually as described in HDFS Balancers in the Cloudera Manager documentation.

The need for rebalancing depends on the amount of data in HDFS and the number of instances added or removed during the cluster. Rebalancing is required only when there is a large movement of data. Cloudera recommends rebalancing the cluster through Cloudera Manager if you increase or reduce the number of DataNodes by 30% or more.

Repairing Worker and Gateway Instances in a Cluster

1. Log in to Cloudera Director at `http://director-server-hostname:7189`

   Cloudera Director opens on the All Environments page, which displays the current environments, deployments, and clusters. Click the cluster you want to modify.

2. Click **Modify Cluster** to the right of the cluster name. The Modify Cluster page displays the gateway, masters, and workers instance groups and any additional instance groups that have been added to the cluster, with the current number of instances in each instance group.
3. Click **Edit** next to the instance count for workers or gateways to repair, and select the instances to repair.
4. Click the **Repair** button above the list of instances. The instances you selected display an action status of **To be repaired**.
5. Click **OK** to continue, **Reset** to unselect the selected instances and make a new selection, or **Cancel** to stop without making any changes.
6. Click **Continue** to confirm and repair the selected instances.

Note: The above procedure is for worker and gateway roles, not for master roles. Because master roles have state, repairing them requires migrating the roles from one host to another. For information on migrating HDFS master roles, see Using Role Migration to Repair HDFS Master Role Instances on page 101.

Terminating a Cluster

You can terminate a cluster at any time using either the UI or the CLI.

Terminating a Cluster with the UI

To terminate a cluster with the UI:

1. Log in to Cloudera Director. For example, `http://cloudera_director_host:7189`.

   Cloudera Director opens with a list of clusters.

2. Click the Actions dropdown arrow for the cluster you want to terminate and click **Terminate**.

3. In the confirmation dialog box, click **Terminate** to terminate the cluster.

Terminating a Cluster with the CLI

For information on terminating a cluster with the CLI, see the section on the `terminate-remote` command in Using the Command Line Interface.

Starting and Stopping the Cloudera Director Server

Although you can stop and start Cloudera Director at any time, you should terminate any running clusters first.

To start or stop the server, enter the following:

```
$ sudo service cloudera-director-server [start | stop]
```

User Management

User roles control the actions a user can perform. There are currently two user roles:

- **Admin** - For administrative access. Has full access to Cloudera Director functionality, and can perform the following actions:
  - Add environments, Cloudera Manager instances, and clusters
  - Delete environments
  - Terminate Cloudera Manager and cluster instances
  - Review environments, Cloudera Manager instances, and clusters
  - Grow and shrink clusters
  - Add and delete users
  - Change user roles
  - Change passwords, including own password
- **Guest** - For read-only access.

On installation, the Cloudera Director server component includes one of each of the two kinds of user accounts:

- **admin** - Default password: `admin`
- **guest** - Default password: `guest`

Cloudera recommends that you change the passwords for these accounts after installing the server. User accounts can be created, deleted, enabled, or disabled. A disabled user account cannot log in or perform any Cloudera Director actions.

User account data is kept in the Cloudera Director database. You can define new user accounts for Cloudera Director with either the server UI or the API.

Managing Users with the Cloudera Director Web UI

You can perform the following user management operations through the Cloudera Director Web UI:

Create a User Account

To create a new user account, perform the following steps:

1. On the Cloudera Director home screen, click the dropdown menu in the upper right and click **Manage Users**.
2. Click the **Add User** button.
3. Enter a username and password for the new user, and select a role (Admin or Guest).
4. Click **Add User**.

Disable a User Account

To disable an existing user account, perform the following steps:

1. On the Cloudera Director home screen, click the dropdown menu in the upper right and click **Manage Users.**
2. Click the checkbox next to the user account you want to disable.
3. Click the dropdown menu for the user account in the **Actions** column and click **Disable User**.
4. Confirm that user you have disabled now appears as unavailable on the Manage Users screen.

You can use the same procedure to enable a user account that is currently disabled. The Actions dropdown list displays the item **Enable User** for a user account that is currently disabled.

Change User Account Passwords

Users with the admin role can change any user's password. Guest users can change only their own password.

To change your own password, perform the following steps:

1. On the Cloudera Director home screen, click the dropdown menu in the upper right and click **Change password**.
2. Enter your current password, a new password, and the new password again to confirm.
3. Click **Save changes**.

To change another user's password, perform the following steps (using the required Admin role):

1. On the Cloudera Director home screen, click the dropdown menu in the upper right and click **Manage Users**.
2. Click the checkbox next to the user whose password you want to change.
3. Click the dropdown menu for the user account in the **Actions** column and click **Change password**.
4. Enter a new password and enter the password again to confirm.
5. Click **Save changes**.

Change a User's Role

An Admin user can change another user's role by performing the following steps:

1. On the Cloudera Director home screen, click the dropdown menu in the upper right and click **Manage Users**.
2. Click the checkbox next to the user whose role you want to change.
3. Click the dropdown menu for the user in the **Actions** column and click **Change role**.
4. Select the new role in the **Role** dropdown menu.
5. Click **Save changes**.

Delete a User Account

An Admin can delete a user account by performing the following steps:

1. On the Cloudera Director home screen, click the dropdown menu in the upper right and click **Manage Users**.
2. Click the checkbox next to the user account you want to delete.
3. Click the dropdown menu for the user account in the **Actions** column and click **Delete**.
4. Click **Delete** to confirm.

Managing Users with the Cloudera Director API

Cloudera Director server has a REST service endpoint for user management, at *director-server-hostname*:7189/api/v2/users. You can perform the following user-management operations with the Cloudera Director API. They all use JSON for input data and response data.

REST method	Description
`GET /api/v2/users`	Lists all usernames.
`POST /api/v2/users`	Creates a new user account (Admin role required).
`GET /api/v2/users/current`	Gets account information on the currently logged-in user.
`GET /api/v2/users/{username}`	Gets account information on a user.
`PUT /api/v2/users/{username}`	Changes account information on a user.
`DELETE /api/v2/users/{username}`	Deletes an account (Admin role required)
`PUT /api/v2/users/{username}/password`	Changes an account password for Guests; old password required, and Guests can only change their own account.

For information on managing users with the Cloudera Director API, see the server API documentation at *director-server-hostname*:7189/api-console. Expand the section labeled **users**.

Customization and Advanced Configuration

The topics in this section explain how to use some of the advanced features of Cloudera Director.

The Cloudera Director Configuration File

The Cloudera Director configuration file is used to launch a cluster through Cloudera Director client with the `bootstrap` command, or through the Cloudera Director server with the `bootstrap-remote` command.

Location of Sample Configuration Files

Sample configuration files are found either in `/usr/lib64/cloudera-director/client` or `/usr/lib/cloudera-director/client`, depending on the operating system you are using. Copy the sample files to your home directory before editing them.

Customizing the Configuration File

Copy the sample files to your home directory before editing them. Rename the *cloud_provider*.simple.conf file to cluster.conf. For advanced cluster configuration, use *cloud_provider*.reference.conf. The configuration file must use the .conf file extension. Open cluster.conf with a text editor.

The `*cloud_provider*.reference.conf` version of the configuration file includes advanced settings that are documented in comments within the file itself. Details on the specific settings in the file are not duplicated in this document.

Valid Role Types for Use in Configuration Files

For a list of valid roles for Cloudera Manager and CDH services that you can use in a Cloudera Director configuration file, see the Cloudera Manager API page on [Available Role Types].

Creating a Cloudera Manager and CDH AMI

You can reduce instance start times, and thereby cluster bootstrap times, by preloading the AMI with Cloudera Manager packages and CDH parcel files. For information on creating AMIs preloaded with Cloudera Manager packages and CDH parcels for use by Cloudera Director see [Cloudera Director preload creation script] on GitHub.

> **Note:** If you are using an AMI that already has Cloudera Manager or CDH pre-loaded on it, you must override the repository in Cloudera Director by specifying a custom repository URL in the custom repository field. The version you specify in this URL override must match what is on your AMI, down to the three digits of the maintenance release. For example, if you have CDH 5.5.1 on the AMI, the repository you specify should be `/5.5.1` and not `/5.5` or `/5`.

Choosing an AMI

An Amazon Machine Image (AMI) specifies the operating system, architecture (32-bit or 64-bit), AWS Region, and virtualization type (Paravirtualization or HVM) for a virtual machine (also known as an instance) that you launch in AWS.

> **Important:** Cloudera Director, CDH, and Cloudera Manager support only 64-bit Linux. For CDH and Cloudera Manager on Amazon EC2, Cloudera Director only supports RHEL and CentOS.

The virtualization type depends on the instance type that you use. After selecting an instance type based on the expected storage and computational load, check the supported virtualization types. Then, identify the correct AMI based on architecture, AWS Region, and virtualization type.

> **Important:** Cloudera Director supports only MBR and GPT partitions for AMIs that have a single partition on the root block device. AMIs with multiple partitions are not supported.

Finding Available AMIs

There are two ways of finding available AMIs:

- Using the AWS Management Console.
- By generating a list of AMIs using the AWS CLI.

  To generate a list of RHEL 64-bit AMIs using the AWS CLI, perform the following steps:

  1. Install the AWS CLI.

```
$ sudo pip install awscli
```

  2. Configure the AWS CLI.

```
$ aws configure
```

     Follow the prompts. Choose any output format. The following example command defines "table" as the format.

  3. Run the following query:

```
aws ec2 describe-images \
  --output table \
  --query 'Images[*].[VirtualizationType,Name,ImageId]' \
  --owners 309956199498 \
  --filters \
    Name=root-device-type,Values=ebs \
    Name=image-type,Values=machine \
    Name=is-public,Values=true \
    Name=hypervisor,Values=xen \
    Name=architecture,Values=x86_64
```

     AWS returns a table of available images in the region you configured.

Creating AWS Identity and Access Management (IAM) Policies

In AWS, IAM files are used to create policies that control access to resources in a VPC. IAM roles allow EC2 instances to make API requests without the need to use or distribute AWS credentials (accessKey and secretAccessKey). For more information about IAM, see the AWS Identity and Access Management User Guide in the AWS documentation. For instructions on how to create an IAM role, see Creating a Role to Delegate Permissions to an AWS Service in the AWS documentation.

Use the AWS Policy Generator to create the IAM file, keeping in mind the following requirements:

- For EC2, Cloudera Director requires permissions for the following methods:
  - CreateTags
  - DescribeAvailabilityZones
  - DescribeImages
  - DescribeInstanceStatus
  - DescribeInstances

  - DescribeKeyPairs
  - DescribePlacementGroups
  - DescribeRegions
  - DescribeSecurityGroups
  - DescribeSubnets
  - RunInstances
  - TerminateInstances

- To validate the templates used for EC2 instance creation, Cloudera Director requires permissions for the following IAM methods:

  - GetInstanceProfile
  - PassRole

- To create RDS database servers for persistence on demand, Cloudera Director requires permissions for the following methods:

  - CreateDBInstance
  - DeleteDBInstance
  - DescribeDBInstances

- With Cloudera Director 1.5 and higher, Cloudera Director requires permissions for the following method:

  - DescribeDBSecurityGroups

  This permission is required because, beginning with version 1.5, Cloudera Director includes early validation of RDS credentials at the time of creating or updating an environment, whether or not RDS database servers will be used.

Example IAM Policy

The following example IAM policy shows the format to use with Cloudera Director. Your Amazon Resource Name (ARN) will be different. For more information on ARNs, see Amazon Resource Names (ARNs) and AWS Service Namespaces in the AWS documentation.

> **Note:** If Cloudera Director does not have the complete set of permissions it needs, an authorization failure may occur. In that event, AWS will return an authorization failure message, which may help with troubleshooting by providing details about the authorization failure. Authorization failure messages are normally encoded for security purposes. The permission shown in the last section of the example IAM policy below (beginning `"Sid": "directorSts"`) enables Cloudera Director to decode authorization failure messages. Before adding this permission, make certain that decoding of authorization messages does not violate your organization's security policies. Cloudera Director should work without this permission if your IAM policy includes the required permissions specified above.

```
{
  "Statement": [
    {
      "Sid": "directorEc2",
      "Effect": "Allow",
      "Action": [
        "ec2:CreateTags",
        "ec2:DescribeAvailabilityZones",
        "ec2:DescribeImages",
        "ec2:DescribeInstanceStatus",
        "ec2:DescribeInstances",
        "ec2:DescribeKeyPairs",
        "ec2:DescribePlacementGroups",
        "ec2:DescribeRegions",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSubnets",
        "ec2:RunInstances",
        "ec2:TerminateInstances"
      ],
      "Resource": "*"
    },
    {
      "Sid": "directorIam",
      "Effect": "Allow",
      "Action": [
        "iam:GetInstanceProfile",
        "iam:PassRole"
      ],
      "Resource": "*"
    },
    {
      "Sid": "directorRds",
      "Effect": "Allow",
      "Action": [
        "rds:CreateDBInstance",
        "rds:DeleteDBInstance",
        "rds:DescribeDBInstances",
        "rds:DescribeDBSecurityGroups"
      ],
      "Resource": "*"
    },
    {
      "Sid": "directorSts",
      "Action": [
        "sts:DecodeAuthorizationMessage"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

Using MySQL for Cloudera Director Server

> **Note:** This section is about the data Cloudera Director server stores for its own use. You can also use external databases for Cloudera Manager and cluster services. For more information, see Using an External Database for Cloudera Manager and CDH on page 73.

Cloudera Director stores various kinds of data, including information about deployments, database servers, users, CDH clusters, and Cloudera Manager instances. By default, this data is stored in an embedded H2 database stored on the filesystem where the server is running at the following location:

```
/var/lib/cloudera-director-server/state.h2.db
```

Alternatively, you can use a MySQL database instead of the embedded H2 database, as described below.

Installing the MySQL Server

> **Note:**
>
> - If you already have a MySQL database set up, you can skip to Configuring and Starting the MySQL Server on page 63 to verify that your MySQL configuration meets the requirements for Cloudera Director.
> - The `datadir` directory (`/var/lib/mysql` by default) must be located on a partition that has sufficient free space.

1. Install the MySQL database.

OS	Command
RHEL	`$ sudo yum install mysql-server`

SLES	`$ sudo zypper install mysql`
	`$ sudo zypper install libmysqlclient_r15`
	Note: Some SLES systems encounter errors with the `zypper install` command. For more information, see the Novell Knowledgebase topic, error running chkconfig.

Ubuntu and Debian	`$ sudo apt-get install mysql-server`

After issuing the command, you may need to confirm that you want to complete the installation.

Configuring and Starting the MySQL Server

1. Determine the version of MySQL.
2. Stop the MySQL server if it is running.

OS	Command
RHEL	`$ sudo service mysqld stop`
SLES, Ubuntu, and Debian	`$ sudo service mysql stop`

3. Move old InnoDB log files `/var/lib/mysql/ib_logfile0` and `/var/lib/mysql/ib_logfile1` from `/var/lib/mysql/` to a backup location.
4. Determine the location of the option file, `my.cnf`, and update it as follows::

   - To prevent deadlocks, set the isolation level to read committed.
   - Configure MySQL to use the `InnoDB` engine, rather than `MyISAM`. (The default storage engine for MySQL is `MyISAM`.) To check which engine your tables are using, run the following command from the MySQL shell:

```
mysql> show table status;
```

   - To configure MySQL to use the `InnoDB` storage engine, add the following line to the `[mysqld]` section of the `my.cnf` option file:

```
[mysqld]
default-storage-engine = innodb
```

   - Binary logging is not a requirement for Cloudera Director installations. Binary logging provides benefits such as MySQL replication or point-in-time incremental recovery after database restore. Examples of this configuration follow. For more information, see The Binary Log.

   Following is a typical option file:

```
[mysqld]
default-storage-engine = innodb
transaction-isolation = READ-COMMITTED
# Disabling symbolic-links is recommended to prevent assorted security risks;
# to do so, uncomment this line:
# symbolic-links = 0

key_buffer = 16M
key_buffer_size = 32M
max_allowed_packet = 32M
thread_stack = 256K
thread_cache_size = 64
query_cache_limit = 8M
query_cache_size = 64M
```

```
query_cache_type = 1

max_connections = 550

#log_bin should be on a disk with enough free space. Replace
'/var/lib/mysql/mysql_binary_log' with an appropriate path for your system.
#log_bin=/var/lib/mysql/mysql_binary_log
#expire_logs_days = 10
#max_binlog_size = 100M

# For MySQL version 5.1.8 or higher. Comment out binlog_format for lower versions.
binlog_format = mixed

read_buffer_size = 2M
read_rnd_buffer_size = 16M
sort_buffer_size = 8M
join_buffer_size = 8M

# InnoDB settings
innodb_file_per_table = 1
innodb_flush_log_at_trx_commit  = 2
innodb_log_buffer_size = 64M
innodb_buffer_pool_size = 4G
innodb_thread_concurrency = 8
innodb_flush_method = O_DIRECT
innodb_log_file_size = 512M

[mysqld_safe]
log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid
```

5. If AppArmor is running on the host where MySQL is installed, you might need to configure AppArmor to allow MySQL to write to the binary.
6. Ensure that the MySQL server starts at boot.

OS	Command

RHEL	`$ sudo /sbin/chkconfig mysqld on`
	`$ sudo /sbin/chkconfig --list mysqld`
	`mysqld          0:off   1:off   2:on    3:on    4:on    5:on`
	`  6:off`

SLES	`$ sudo chkconfig --add mysql`
Ubuntu and Debian	`$ sudo chkconfig mysql on` **Note:** `chkconfig` may not be available on recent Ubuntu releases. You may need to use Upstart to configure MySQL to start automatically when the system boots. For more information, see the Ubuntu documentation or the Upstart Cookbook.

7. Start the MySQL server:

OS	Command
RHEL	`$ sudo service mysqld start`
SLES, Ubuntu, and Debian	`$ sudo service mysql start`

8. Set the MySQL root password. In the following example, the current `root` password is blank. Press the **Enter** key when you're prompted for the root password.

```
$ sudo /usr/bin/mysql_secure_installation
[...]
Enter current password for root (enter for none):
OK, successfully used password, moving on...
```

```
[...]
Set root password? [Y/n] y
New password:
Re-enter new password:
Remove anonymous users? [Y/n] Y
[...]
Disallow root login remotely? [Y/n] N
[...]
Remove test database and access to it [Y/n] Y
[...]
Reload privilege tables now? [Y/n] Y
All done!
```

Installing the MySQL JDBC Driver

Install the MySQL JDBC driver for the Linux distribution you are using.

OS	Command

RHEL 5 or 6	**1.** Download the MySQL JDBC driver from the Download Connector/J page of the MySQL web site.
	2. Extract the JDBC driver JAR file from the downloaded file. For example:
	`tar zxvf mysql-connector-java-version.tar.gz`
	3. Add the JDBC driver, renamed, to the relevant server. For example:
	`$ sudo cp mysql-connector-java-version/mysql-connector-java-version-bin.jar /usr/share/java/mysql-connector-java.jar`
	If the target directory does not yet exist on this host, you can create it before copying the JAR file. For example:
	`$ sudo mkdir -p /usr/share/java/`
	`$ sudo cp mysql-connector-java-version/mysql-connector-java-version-bin.jar /usr/share/java/mysql-connector-java.jar`
	Note: Do not use the `yum install` command to install the MySQL connector package, because it installs the openJDK, and then uses the Linux `alternatives` command to set the system JDK to be the openJDK.

SLES	`$ sudo zypper install mysql-connector-java`
Ubuntu or Debian	`$ sudo apt-get install libmysql-java`

Creating a Database for Cloudera Director Server

You can create the database on the host where the Cloudera Director server will run, or on another host that is accessible by the Cloudera Director server. The database must be configured to support UTF-8 character set encoding.

Record the values you enter for database names, usernames, and passwords. Cloudera Director requires this information to connect to the database.

1. Log into MySQL as the root user:

```
$ mysql -u root -p
Enter password:
```

2. Create a database for Cloudera Director server:

```
mysql> create database database DEFAULT CHARACTER SET utf8;
Query OK, 1 row affected (0.00 sec)

mysql > grant all on database.* TO 'user'@'%' IDENTIFIED BY 'password';
Query OK, 0 rows affected (0.00 sec)
```

database, *user*, and *password* can be any value. The examples match the names you provide in the Cloudera Director configuration settings described below in [Configure Cloudera Director Server to use the MySQL Database](#).

Backing Up MySQL Databases

To back up the MySQL database, run the `mysqldump` command on the MySQL host, as follows:

```
$ mysqldump -hhostname -uusername -ppassword database > /tmp/database-backup.sql
```

Configuring Cloudera Director Server to use the MySQL Database

Before starting the Cloudera Director server, edit the "Configurations for database connectivity" section of `/etc/cloudera-director-server/application.properties`.

> **Note:** If the Cloudera Director server is already running, it must be restarted after configuring MySQL access. The server will not load configuration updates while running.

.

```
#
# Configurations for database connectivity.
#

# Optional database type (h2 or mysql) (defaults to h2)
#lp.database.type: mysql

# Optional database username (defaults to "director")
#lp.database.username:

# Optional database password (defaults to "password")
#lp.database.password:

# Optional database host (defaults to "localhost")
#lp.database.host:

# Optional database port (defaults to 3306)
#lp.database.port:

# Optional database (schema) name (defaults to "director")
#lp.database.name:
```

Using MariaDB for Cloudera Director Server

> **Note:** This section is about the data Cloudera Director server stores for its own use. You can also use external databases for Cloudera Manager and cluster services. For more information, see [Using an External Database for Cloudera Manager and CDH](#) on page 73.

Cloudera Director stores various kinds of data, including information about deployments, database servers, users, CDH clusters, and Cloudera Manager instances. By default, this data is stored in an embedded H2 database stored on the filesystem where the server is running at the following location:

```
/var/lib/cloudera-director-server/state.h2.db
```

Alternatively, you can use a MariaDB database instead of the embedded H2 database, as described below.

Installing the MariaDB Server

> **Note:**
>
> - If you already have a MariaDB database set up, you can skip to Configuring and Starting the MariaDB Server on page 67 to verify that your MariaDB configuration meets the requirements for Cloudera Director.
> - The `datadir` directory (`/var/lib/mysql` by default) must be located on a partition that has sufficient free space.

1. Install the MariaDB database.

```
$ sudo yum install mysql-server
```

After issuing the command, you might need to confirm that you want to complete the installation.

Configuring and Starting the MariaDB Server

1. Stop the MariaDB server if it is running.

   - For RHEL 6:

```
$ sudo service mysqld stop
```

   - For RHEL 7:

```
$ sudo systemctl mariadb stop
```

2. Move old InnoDB log files `/var/lib/mysql/ib_logfile0` and `/var/lib/mysql/ib_logfile1` from `/var/lib/mysql/` to a backup location.
3. Determine the location of the option file, `my.cnf`, and update it as follows::

   - To prevent deadlocks, set the isolation level to read committed.
   - Configure MariaDB to use the `InnoDB` engine, rather than `MyISAM`. (The default storage engine for MariaDB is `MyISAM`.) To check which engine your tables are using, run the following command from the MariaDB shell:

```
mysql> show table status;
```

   - To configure MariaDB to use the `InnoDB` storage engine, add the following line to the `[mysqld]` section of the `my.cnf` option file:

```
[mysqld]
default-storage-engine = innodb
```

   - Binary logging is not a requirement for Cloudera Director installations. Binary logging provides benefits such as MariaDB replication or point-in-time incremental recovery after database restore. Examples of this configuration follow. For more information, see The Binary Log.

   Following is a typical option file:

```
[mysqld]
default-storage-engine = innodb
transaction-isolation = READ-COMMITTED
# Disabling symbolic-links is recommended to prevent assorted security risks;
# to do so, uncomment this line:
# symbolic-links = 0
```

```
key_buffer = 16M
key_buffer_size = 32M
max_allowed_packet = 32M
thread_stack = 256K
thread_cache_size = 64
query_cache_limit = 8M
query_cache_size = 64M
query_cache_type = 1

max_connections = 550

#log_bin should be on a disk with enough free space. Replace
'/var/lib/mysql/mysql_binary_log' with an appropriate path for your system.
#log_bin=/var/lib/mysql/mysql_binary_log
#expire_logs_days = 10
#max_binlog_size = 100M

# For MySQL version 5.1.8 or later. Comment out binlog_format for older versions.
binlog_format = mixed

read_buffer_size = 2M
read_rnd_buffer_size = 16M
sort_buffer_size = 8M
join_buffer_size = 8M

# InnoDB settings
innodb_file_per_table = 1
innodb_flush_log_at_trx_commit  = 2
innodb_log_buffer_size = 64M
innodb_buffer_pool_size = 4G
innodb_thread_concurrency = 8
innodb_flush_method = O_DIRECT
innodb_log_file_size = 512M

[mysqld_safe]
log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid
```

4. If AppArmor is running on the host where MariaDB is installed, you might need to configure AppArmor to allow MariaDB to write to the binary.
5. Ensure the MariaDB server starts at boot.

    - For RHEL 6:

```
$ sudo chkconfig mysqld on
```

    - For RHEL 7:

```
$ sudo systemctl enable mariadb
```

6. Start the MariaDB server:

    - For RHEL 6:

```
$ sudo service mysqld start
```

    - For RHEL 7:

```
$ sudo systemctl mariadb start
```

7. Set the MariaDB root password. In the following example, the current `root` password is blank. Press the **Enter** key when you're prompted for the root password.

```
$ sudo /usr/bin/mysql_secure_installation
[...]
Enter current password for root (enter for none):
OK, successfully used password, moving on...
[...]
Set root password? [Y/n] y
New password:
Re-enter new password:
Remove anonymous users? [Y/n] Y
[...]
Disallow root login remotely? [Y/n] N
[...]
Remove test database and access to it [Y/n] Y
[...]
Reload privilege tables now? [Y/n] Y
All done!
```

Installing the MariaDB JDBC Driver

Install the MariaDB JDBC driver for the Linux distribution you are using.

1. Download the MySQL JDBC driver from http://www.mysql.com/downloads/connector/j/5.1.html.
2. Extract the JDBC driver JAR file from the downloaded file. For example:

```
tar zxvf mysql-connector-java-5.1.31.tar.gz
```

3. Copy the JDBC driver, renamed, to the relevant host. For example:

```
$ sudo cp mysql-connector-java-5.1.31/mysql-connector-java-5.1.31-bin.jar
/usr/share/java/mysql-connector-java.jar
```

If the target directory does not yet exist on this host, you can create it before copying the JAR file. For example:

```
$ sudo mkdir -p /usr/share/java/
$ sudo cp mysql-connector-java-5.1.31/mysql-connector-java-5.1.31-bin.jar
/usr/share/java/mysql-connector-java.jar
```

Note: Do not use the `yum install` command to install the MySQL driver package, because it installs openJDK, and then uses the Linux `alternatives` command to set the system JDK to be openJDK.

Creating a Database for Cloudera Director Server

You can create the database on the host where the Cloudera Director server will run, or on another host that is accessible by the Cloudera Director server. The database must be configured to support UTF-8 character set encoding.

Record the values you enter for database names, usernames, and passwords. Cloudera Director requires this information to connect to the database.

1. Log into MariaDB as the root user:

```
$ mysql -u root -p
Enter password:
```

2. Create a database for Cloudera Director server:

```
mysql> create database database DEFAULT CHARACTER SET utf8;
Query OK, 1 row affected (0.00 sec)
```

```
mysql > grant all on database.* TO 'user'@'%' IDENTIFIED BY 'password';
Query OK, 0 rows affected (0.00 sec)
```

database, *user*, and *password* can be any value. The examples match the names you provide in the Cloudera Director configuration settings described below in Configure Cloudera Director Server to use the MariaDB Database.

Backing Up MariaDB Databases

To back up the MariaDB database, run the `mysqldump` command on the MariaDB host, as follows:

```
$ mysqldump -hhostname -uusername -ppassword database > /tmp/database-backup.sql
```

Configuring Cloudera Director Server to use the MariaDB Database

Before starting the Cloudera Director server, edit the "Configurations for database connectivity" section of `/etc/cloudera-director-server/application.properties`.

> **Note:** If the Cloudera Director server is already running, it must be restarted after configuring MariaDB access. The server will not load configuration updates while running.

.

```
#
# Configurations for database connectivity.
#

# Optional database type (h2 or mysql) (defaults to h2)
#lp.database.type: mysql

# Optional database username (defaults to "director")
#lp.database.username:

# Optional database password (defaults to "password")
#lp.database.password:

# Optional database host (defaults to "localhost")
#lp.database.host:

# Optional database port (defaults to 3306)
#lp.database.port:

# Optional database (schema) name (defaults to "director")
#lp.database.name:
```

Cloudera Director Database Encryption

The Cloudera Director server stores sensitive data in its database, including SSH credentials and cloud provider keys. You can configure Cloudera Director to encrypt the data stored in the Cloudera Director database.

> **Note:** This section discusses data stored in the Cloudera Director database, not data stored in databases used by Cloudera Manager or CDH cluster services.

Cipher Configuration

Database encryption is configured by setting the two server configuration properties described in the following table.

Table 2: Server Configuration Properties

Property	Description
lp.encryption.twoWayCipher	Cipher used to encrypt data. Possible values: • `desede` - Triple DES (default) • `passthrough` - No encryption • `transitional` - Changing encryption
lp.encryption.twoWayCipherConfig	The configuration string for the chosen cipher.

The format of the configuration string varies with the choice of cipher, as described in the table below:

Table 3: Ciphers and Configuration Strings

Cipher	Configuration String Format
`desede`	24-byte symmetric encryption key, encoded as a string using Base64
`passthrough`	ignored
`transitional`	combination of old cipher and new cipher (see below)

The default value for the configuration string is a fixed 24-byte key for the default triple DES encryption:

```
ZGVmYXVsdGRpcmVjdG9yZGVzZWRla2V5
```

Important: Cloudera highly recommends that you configure a different triple DES key. A warning appears in the server log if the default key is detected.

Starting with Encryption

Cloudera Director's default configuration for database encryption encrypts new data stored in the Cloudera Director database. This default configuration uses triple DES encryption, with a default key, to protect data. In a new installation of Cloudera Director, all data needing protection will be encrypted under the default encryption scheme. In an installation that was previously not configured for encryption, including older releases of Cloudera Director, new data needing protection will be encrypted, but old data needing protection will remain unencrypted until it is updated in the database over time.

If this level of protection is sufficient for your needs, it is not necessary to make any changes to Cloudera Director configuration. While Cloudera Director will function correctly, keep in mind that there are drawbacks: some data needing protection in the database may remain unencrypted indefinitely, and data that is encrypted is effectively only obscured, since the default key is not secret.

Establishing More Secure Encryption for New Installations

For a new installation of Cloudera Director, Cloudera recommends that you generate and configure your own secret encryption key, different from the default key. Create a new key by generating 24 bytes of random data from a cryptographically secure random generator, and encode the bytes using the Base64 encoding algorithm.

Here is an example of generating a new key using Python.

```
python -c 'import base64, os; print base64.b64encode(os.urandom(24))'
```

Set the Cloudera Director configuration property `lp.encryption.twoWayCipherConfig` to the Base64-encoded key string before starting Cloudera Director for the first time. All data needing protection in the database will be encrypted with this key. It is good practice to change the encryption key periodically to protect against unintentional disclosure. See Changing Encryption below for more.

> **Note:** If you configure a new secret key, Cloudera recommends you restrict permissions on the configuration file (`application.properties`) to protect the key from disclosure. Ensure that at least the user running Cloudera Director can still read the file.

Establishing More Secure Encryption for Existing Installations

For an existing installation of Cloudera Director that uses either no encryption at all (including older releases of Cloudera Director) or uses only the default encryption, Cloudera recommends that you use a transitional cipher to change encryption to a more secure state. Not only will changing encryption introduce the use of a non-default and secret key, but it will also forcibly encrypt all data needing protection in the database, whether it was already encrypted or not.

See Changing Encryption below for details on how to configure a transitional cipher to change encryption. When configuring the transitional cipher, you will need to know information about the old cipher that was in effect.

- If the default cipher and key was in use previously, then use "desede" and the default key for the old cipher configuration.
- If no encryption was in place previously, including older releases of Cloudera Director which did not support database encryption, then use "passthrough" (with no configuration string) for the old cipher configuration.

The new cipher should be triple DES ("desede") with a secret key that you generate. See Establishing More Secure Encryption for New Installations above for details on how to generate a good key.

After establishing more secure encryption, it is good practice to change the encryption key periodically to protect against unintentional disclosure. Use the transitional cipher again to change encryption to use a new key.

Changing Encryption

To change the key used for database encryption, or change to a different cipher, you must configure the Cloudera Director server to use a transitional cipher.

> **Note:** Transitional ciphers are supported for Cloudera Director server only, not for Cloudera Director client.

If a transitional cipher is configured, Cloudera Director encrypts all data that needs protection, changing from an old encryption scheme to a new encryption scheme. A transitional cipher can change the encryption in effect, or introduce it when it has not been used before, including under older Cloudera Director releases. It also ensures that all data needing protection becomes encrypted.

To configure a transitional cipher:

1. Stop the server.
2. Configure `lp.encryption.twoWayCipher` with the value `transitional`.
3. Configure `lp.encryption.twoWayCipherConfig` with a configuration string describing both the old cipher and the new cipher.
4. Start the server.

The configuration string for a transitional cipher has the following format:

```
old-cipher;old-configuration-string|new-cipher;new-configuration-string
```

For example, to change the triple DES key, use a configuration string like this:

```
desede;old-key-in-base64|desede;new-key-in-base64
```

To transition from the default triple DES encryption key to a new key, use a configuration string like this:

```
desede;ZGVmYXVsdGRpcmVjdG9yZGVzZWRla2V5|desede;new-key-in-base64
```

To transition from no encryption to triple DES encryption with a new key, use a configuration string like this:

```
passthrough;|desede;new-key-in-base64
```

A transitional cipher cannot be used as the old or new cipher in another transitional cipher.

When the server restarts, it detects that a transitional cipher is configured and updates all relevant data, unencrypted and encrypted, to the new cipher. After this process is complete, the server continues startup as usual. Configuring a transitional cipher ensures that all data needing protection in the database is encrypted.

Wait for the Server to Complete Ongoing Work

Do not try to change encryption while the server is performer ongoing work. If any work is waiting to be resumed by the server on startup (for example, bootstrapping a new cluster), then the server will refuse to change encryption and will stop. If this happens, you must configure the server for its old cipher, start it, and wait for that work to resume and be completed.

Changing from a Transitional Cipher to a Normal Cipher

After encryption has been changed using a transitional cipher, you can configure the server to use the new cipher normally.

Example: Assume the configuration string for the transitional cipher was as follows:

```
desede;old-key-in-base64|desede;new-key-in-base64
```

One restart of the server will suffice to pick up this change, and then the following configuration string for a normal cipher can be used:

```
desede;new-key-in-base64
```

Cloudera recommends that the server be left to run with a transitional cipher only until its next restart or upgrade, and then be reconfigured to use a normal cipher. There are two reasons for doing this:

- While configured with a transitional cipher, the server will not restart if work is waiting to be resumed.
- If the server is left configured with a transitional cipher, each time it is restarted the database contents will be re-encrypted using the same key.

Using an External Database for Cloudera Manager and CDH

By default, Cloudera Director configures Cloudera Manager and CDH services, such as Hive, to use the Cloudera Manager embedded PostgreSQL database. You can use Cloudera Director to configure them to use external database servers, instead, which is recommended for production environments. If you have a database server already configured, you can configure Cloudera Manager and CDH services to create or use databases on that server. You can also configure Cloudera Director to use a cloud provider service such as Amazon's Relational Database Service (RDS) to provision new database servers.

How you set up external database servers and databases differs depending on whether you are using Cloudera Director client or Cloudera Director server:

- **Cloudera Director client** - Configure external databases in the `cluster.conf` file and launch Cloudera Director client (standalone) by issuing the `bootstrap` command.
- **Cloudera Director server** - Configure external databases for Cloudera Director server in one of the following ways:
  - Using the Cloudera Director UI
  - Using the Cloudera Director REST API

- By editing the `cluster.conf` file and launching the Cloudera Director server with the `bootstrap-remote` command

The topics in this section describe how to use Cloudera Director to define external database servers and external databases.

Defining External Database Servers

Cloudera Director needs information about external database servers before it can use them. This section describes defining database server templates and using Amazon Relational Database Service (RDS) to create new database servers..

The Database Server Template

A database server template can refer to either an existing database server or a server to be created. The following are the basic elements of a database server template:

- **name** - A unique name for the server within the environment
- **type** - The type of database server, such as "MYSQL" or "POSTGRESQL"
- **hostname** - The name of the server host
- **port** - The listening port of the server
- **username** - The name of the administrative account for the server
- **password** - The password for the administrative account

The hostname and port are optional in a template. If they are not present, Cloudera Director assumes that the template refers to a server that does not yet exist and must be created.

A database server template also supports a table of key-value pairs of configuration information, which Cloudera Director may require when creating a new server. A template also supports a second table of tag data, which Cloudera Director can employ for certain cloud providers, including Amazon Web Services.

Note: A single database server is scoped to an environment, so only deployments and clusters in that environment recognize it.

Defining a Database Server Using the API

The Cloudera Director server has a REST service endpoint for managing external database server definitions. The operations supported by the endpoint are described in the table below.

- Each service URI begins with "`/api/v2/environments/{environment}`", where "`{environment}`" is the name of the environment within which the database server definition is scoped.
- They all use JSON for input data and response data.

Operation	Description	Notes
POST /databaseServers/	Define a new database.	Admin required.
GET /databaseServers/	List all database servers.	
DELETE /databaseServers/{name}	Delete a database server definition.	Admin required.
PUT /databaseServers/{name}	Update a database server definition.	Admin required.
GET /databaseServers/{name}	Get a database server definition.	
GET /databaseServers/{name}/status	Get the status of a database server.	
GET /databaseServers/{name}/template	Get the template from which a database server was defined.	

If a database server template without a host and port is posted to Cloudera Director, Cloudera Director will asynchronously begin the process of creating the server on a cloud provider. The provider is selected based on the environment.

Similarly, if a database server definition is deleted, and the server was originally created by Cloudera Director, Cloudera Director will begin the process of deleting the database from the cloud provider. Before deleting a server definition, be sure to make any backups of the server that you need.

The status of a database server indicates its current position in the server lifecycle. The following values can be returned by the GET database server status operation:

Status	Description
BOOTSTRAPPING	Cloudera Director is in the process of creating the server.
BOOTSTRAP_FAILED	Cloudera Director failed to create the server.
READY	The server is available for use.
TERMINATING	Cloudera Director is in the process of destroying the server.
TERMINATE_FAILED	Cloudera Director failed to terminate the server.
TERMINATED	The server has been destroyed.

Defining a Database Server Using the Client Configuration File

Database server templates can be provided in the configuration file passed to the Cloudera Director standalone client. Define external database servers in the `databaseServers` section of a configuration file.

See the API section above for a description of the different parts of a template. The following example defines two existing database servers.

```
databaseServers {
   mysql1 {
      type: mysql
      host: 1.2.3.4
      port: 3306
      user: root
      password: password
   }
   postgres1 {
      type: postgresql
      host: 1.2.3.4
      port: 5432
      user: postgres
      password: password
   }
}
```

The following example defines a server that Cloudera Director must create using RDS.

```
databaseServers {
   mysqlt1 {
      type: mysql
      user: root
      password: password
      instanceClass: db.m3.medium
      engineVersion: 5.5.40b
      dbSubnetGroupName: default
      vpcSecurityGroupIds: sg-abcd1234
      allocatedStorage: 10
      tags {
         owner: jsmith
      }
   }
}
```

You cannot include both existing servers and servers that Cloudera Director must create, in the same configuration file. You can create new database servers separately in a cloud provider and then define them as existing servers in the configuration file.

Using Amazon RDS for External Databases

Cloudera Director can use Amazon Relational Database Service (RDS) to create new database servers. These servers can be used to host external databases for Cloudera Manager and CDH cluster services.

Note:

- At this time, only MySQL 5.5 and 5.6 RDS instances are supported.
- RDS works through both `bootstrap-remote` and standalone `bootstrap` on the client, as well as through the UI and the server API.
- The database server must be in the same AWS region as Cloudera Director.

Creating a Template to Use Amazon RDS as an External Database

An external database server to be created on RDS is defined by a template just like any other server, except that the host and port are not specified; these are determined as the server is being created.

- **name** - A unique name for the server within the environment
- **type** - The type of database server, such as “MYSQL”
- **username** - The name of the administrative account for the server
- **password** - The password for the administrative account

The key-value configuration information in the template for an RDS server must include information required by RDS to create a new instance. Cloudera recommends that you specify the engine version in a template. If you do not specify the version, RDS defaults to a recent version, which can change over time.

Note: If you are including Hive in your clusters, and you configure the Hive metastore to be installed on MySQL through RDS, Cloudera Manager may report that "The Hive Metastore canary failed to create a database." This is caused by a MySQL bug that is exposed through using MySQL 5.6.5 or higher with the MySQL JDBC driver (used by Cloudera Director) version 5.1.19 or lower. Cloudera recommends that you use a MySQL version that avoids revealing this bug for the driver version installed by Cloudera Director from your platform software repositories.

key	description	example
`instanceClass`	Instance type for database server instance	`db.m3.medium`
`dbSubnetGroupName`	Name of the DB subnet group which the instance spans	`default`
`engineVersion`	(optional) Version of database engine	5.5.40b
`vpcSecurityGroupIds`	Comma-separated list of security groups for the new instance	`sg-abc123,sg-def456`
`allocatedStorage`	Storage in gigabytes for new server	`10`
`availabilityZone`	(optional) Preferred availability zone for the new server	`us-east-1d`

Note:

- Cloudera Director does not currently support creating multi-AZ instances.
- The template can also specify tags for the new instance.

Defining a Database Server in AWS Using RDS: UI

You can define an RDS database in AWS using the Cloudera Director UI when you create a CM instance. In the Database Server section near the top of the Add Cloudera Manager wizard, click the dropdown list and select either **Create Database Server Instance** or **Register Existing Database Server**.

Select **Create Database Server Instance** to create a new MySQL database server with RDS. In the **Create Database Server Instance** window, enter credentials and configuration values for the database server.

For more information about configuring a database in Amazon RDS see the Amazon Relational Database Service Documentation.

Note: Cloudera Director also supports PostgreSQL database servers for Cloudera Manager and CDH, but they must be created outside of Cloudera Director and then treated as existing databases by selecting **Register Existing Database Server**.

Select **Register Exiting Database Server** to use an existing MySQL or PostgreSQL database server. In the **Register Exiting Database Server** window, enter information and credentials about your existing database server.

Defining a Database Server in AWS Using RDS: API

Use the previously described REST service endpoint for external database server definitions to create and destroy external database servers using RDS. The environment in which servers are defined must already be configured to use AWS, and your account must have permission to create and delete RDS instances.

When an external database server template is submitted through POST to the endpoint, and the template lacks a host and port, Cloudera Director accepts the definition for the server and asynchronously begins the process of creating the new server. The complete existing server definition, including the host and port, will eventually be available through `GET`.

Likewise, when the definition is deleted using `DELETE`, Cloudera Director begins destroying the server.

While a new server is being created on RDS, you may begin the process of bootstrapping new deployments and new clusters whose external database templates refer to the server. The bootstrap process will proceed in tandem with the server creation, and pause when necessary to wait for the new RDS instance to be available for use.

When a deployment or cluster is terminated, Cloudera Director leaves RDS instances alone. This makes it possible for multiple deployments and clusters to share the same external database servers that Cloudera Director creates on RDS.

Defining a Database Server in AWS Using RDS: Client Configuration File

The following example defines a server that Cloudera Director must create using RDS:

```
databaseServers {
   mysqlt1 {
      type: mysql
      user: root
      password: password
      instanceClass: db.m3.medium
      engineVersion: 5.5.40b
      dbSubnetGroupName: default
      vpcSecurityGroupIds: sg-abcd1234
      allocatedStorage: 10
      tags {
         owner: jsmith
      }
   }
}
```

The following example of an external database template uses the new server that Cloudera Director needs to create. The `databaseServerName` item matches the name of the new server:

```
cluster {
   #... databaseTemplates: {
   HIVE {
        name: hivetemplate
        databaseServerName: mysqlt1
        databaseNamePrefix: hivemetastore
        usernamePrefix: hive
   }
}
```

Defining External Databases

After external database servers are defined, the databases on them can be defined. Cloudera Director can use databases that already exist on those servers, or it can create them while bootstrapping new Cloudera Manager instances or CDH clusters.

The following parts of an existing database must be defined:

- **type** - The type of database, “MYSQL” or “POSTGRESQL.”
- **hostname** - The name of the server host.
- **port** - The listening port of the server.
- **name** - The name of the database on the server.
- **username** - The name of the user account having full access to the database.
- **password** - The password for the user account.

The parts of an external database template are:

- **name** - A unique name for the template within the deployment or cluster template.
- **databaseServerName** - The name of the external database server where the new database is to reside.
- **databaseNamePrefix** - The string prefix for the name of the new database server.
- **usernamePrefix** - The string prefix for the name of the new user account that will have full access to the database.

The database server name in a database server template must refer to an external database server that is already defined.

When Cloudera Director creates the new database, it names the database by starting with the prefix in the template and then appends a random string. This prevents name duplication issues when sharing a database server across many deployments and clusters. Likewise, Cloudera Director creates new user accounts by starting with the prefix in the template and appending a random string.

Important: If you are using a MySQL database, the `usernamePrefix` you define should be no more than seven characters long. This keeps usernames generated by Cloudera Director within the MySQL limit of sixteen characters for usernames.

If Cloudera Director creates new external databases during the bootstrap of a deployment or cluster, then it also drops them, and their associated user accounts, when terminating the deployment or cluster. Be sure to back up those databases before beginning termination.

Note: Cloudera Director cannot create databases on remote database servers that Cloudera Director (or code that it runs) is unable to reach. For example, Cloudera Director cannot work with a database server that only allows local access, unless that server happens to be on the same machine as Cloudera Director. Use the following workarounds:

- Reconfigure the database server, and any security measures that apply to it, to allow Cloudera Director access during the bootstrap and termination processes.
- Open an SSH tunnel for database server access.
- Create the databases manually and configure them using normal Cloudera Director support for external databases.

API

Define external databases in the templates for new Cloudera Manager installations ("deployments") or new clusters. You cannot define both existing databases, and new databases that need to be created, in the same template.

Defining External Databases in the Configuration File

External Databases for Cloudera Manager

Define external databases used by Cloudera Manager in the `cloudera-manager` section of a configuration file. The following example defines existing external databases, indicated by the fact that it includes values for the hostnames or IP addresses and the ports.

```
cloudera-manager {
   # ...
   databases {
      CLOUDERA_MANAGER {
         name: scm1
         type: mysql
         host: 1.2.3.4
         port: 3306
         user: scmuser
         password: scmpassword
      }
      ACTIVITYMONITOR {
         name: am1
         type: mysql
         host: 1.2.3.4
         port: 3306
         user: amuser
         password: ampassword
      }
      REPORTSMANAGER {
         name: rm1
         type: mysql
         host: 1.2.3.4
         port: 3306
         user: rmuser
         password: rmpassword
      }
      NAVIGATOR {
         name: nav1
         type: mysql
         host: 1.2.3.4
         port: 3306
         user: navuser
         password: navpassword
      }
      NAVIGATORMETASERVER {
         name: navmeta1
         type: mysql
         host: 1.2.3.4
         port: 3306
         user: navmetauser
         password: navmetapassword
      }
   }
```

The following example, which does not include hostnames or IP addresses and ports, defines new external databases that Cloudera Director must create while bootstrapping the deployment.

```
cloudera-manager {
   # ...
   databaseTemplates {
      CLOUDERA_MANAGER {
         name: cmtemplate
         databaseServerName: mysql1
         databaseNamePrefix: scm
         usernamePrefix: cmadmin
      }
      ACTIVITYMONITOR {
         name: cmamtemplate
         databaseServerName: mysql1
         databaseNamePrefix: am
         usernamePrefix: cmamadmin
      }
      REPORTSMANAGER {
         name: cmrmtemplate
         databaseServerName: mysql1
         databaseNamePrefix: rm
         usernamePrefix: cmrmadmin
      }
      NAVIGATOR {
         name: cmnavtemplate
         databaseServerName: mysql1
         databaseNamePrefix: nav
         user: cmnavadmin
      }
      NAVIGATORMETASERVER {
         name: cmnavmetatemplate
         databaseServerName: mysql1
         databaseNamePrefix: navmeta
         usernamePrefix: cmnavmetaadmin
      }
   }
```

Each template must refer to a database server defined elsewhere in the configuration file. The database server template can be for a server that does not yet exist; in that case, Cloudera Director starts creating the server, and then waits while bootstrapping the deployment until the server is available.

A deployment must use either all existing databases or all non-existing databases for the different Cloudera Manager components; they cannot be mixed.

For CDH Services

Define external databases used by cluster services such as Hive in the `cluster` section of a configuration file. The following example defines existing external databases.

```
cluster {
   #...
   databaseTemplates: {
      HIVE {
         name: hive1
         type: mysql
         host: 1.2.3.4
         port: 3306
         user: hiveuser
         password: hivepassword
      }
   }
```

The following example defines new external databases that Cloudera Director must create while bootstrapping the cluster.

```
cluster {
   #...
   databaseTemplates: {
      HIVE {
      name: hivetemplate
      databaseServerName: mysql1
      databaseNamePrefix: hivemetastore
      usernamePrefix: hive
   }
}
```

Each template must refer to a database server defined elsewhere in the configuration file. The database server template can be for a server that does not yet exist; in that case, Cloudera Director starts creating the server, and then waits while bootstrapping the cluster until the server is available.

A deployment must use either all existing databases or all non-existing databases for the different cluster services; they cannot be mixed.

Setting Cloudera Director Properties

This topic lists the configuration properties recognized by Cloudera Director. Upon installation, these properties are pre-configured with reasonable default values, and you can run either client or server versions without specifying any of them. However, you might want to customize one or more properties, depending on your environment and the Cloudera Director features you want to use.

Setting Configuration Properties

The Cloudera Director command line provides the simplest way to specify a configuration property. For example:

```
./bin/cloudera-director bootstrap aws.simple.conf \
--lp.pipeline.retry.maxWaitBetweenAttempts=60
```

```
./bin/cloudera-director-server --lp.security.disabled=false
```

Tip: If you want to configure many properties, add them to the `etc/application.properties` file in the Cloudera Director installation. The properties in this file take effect automatically. To override these properties, set new values in the command line.

For users upgrading Cloudera Director

If you modified the `application.properties` file in Cloudera Director, the result of an upgrade depends on the version of Linux you are using:

- **RHEL and CentOS** - When new properties are introduced in Cloudera Director, they are added to `application.properties.rpmnew`. The original `application.properties` file functions as before and is not overwritten with the new Cloudera Director version properties. You do not need to copy the new properties from `application.properties.rpmnew` to the old `application.properties` file.
- **Ubuntu** - The modified Cloudera Director `application.properties` file is backed up to a file named `application.properties.dpkg-old`. The original `application.properties` file is then overwritten by the new `application.properties` file containing new Cloudera Director properties. After upgrading, copy your changes from `application.properties.dpkg-old` to the new `application.properties` file.

All the new properties are commented, and they all use valid defaults, so you do not necessarily need to merge the two properties files. But you must merge the two files if you want to modify one of the newly introduced properties.

See

Property Types

Type	Description
boolean	Either true or false
char	Single character
directory	Valid directory path
enum	Fixed set of string values; a list of each enumeration's values is provided following the main property table below
enum list	Comma-separated list of enums
file	Valid file path
int	Integer (32-bit)
long	Long integer (64-bit)
string	Ordinary character string
time unit	Enumeration of time units: DAYS, HOURS, MICROSECONDS, MILLISECONDS, MINUTES, NANOSECONDS, SECONDS

Properties

Property	Description
`lp.access.logging.config.file`	File for Cloudera Director server access log. Type: string Default: none; must be set if `lp.access.logging.enabled` is `true`.
`lp.access.logging.enabled`	Enable Cloudera Director server access logging. Type: boolean Default: false
`lp.bootstrap.agents.maxNumberOfInstallAttempts`	Maximum number of times to retry installing Cloudera Manager agent. Use -1 for unlimited. Type: int Default: -1
`lp.bootstrap.parallelBatchSize`	Parallelism for allocating and setting up cluster instances when bootstrapping a cluster. Type: int Default: 20
`lp.bootstrap.parcels.distributeMaxConcurrentUploads`	Maximum concurrent uploads of parcels across cluster. Type: int Default: 5
`lp.bootstrap.parcels.distributeRateLimitKBs`	Maximum rate of parcel upload, in KB/s. Type: int Default: 256000

Property	Description
`lp.bootstrap.resume.policy`	Action to take when resuming a previous bootstrap. Use RESTART to start from scratch. Use RESUME to resume from last known state. Use INTERACTIVE to prompt to ask. Type: enum Valid values: RESTART | RESUME | INTERACTIVE Default: INTERACTIVE
`lp.cache.health.expirationMultiplier`	Multiplier applied to polling rate to find health cache expiration duration; negative = disable health polling. Type: int Default: 2
`lp.cache.health.numberOfCacheExecutionThreads`	Number of threads used to poll for service and cluster health. Type: int Default: 5
`lp.cache.health.pollingRateInMilliseconds`	Rate at which service and cluster health is polled, in milliseconds. Type: long Default: 30000
`lp.cleanup.databases.intervalBetweenAttemptsInMs`	Wait time between attempts to destroy external databases, in milliseconds. Type: long Default: 60000
`lp.cleanup.databases.maxNumberOfDeleteAttempts`	Maximum number of times to retry destroying external databases; -1 = unlimited. Type: int Default: 5
`lp.cloud.databaseServers.allocate.timeoutInMinutes`	Time to wait for allocated database server instances to begin running to have ports available. Type: int Default: 20
`lp.cloud.databaseServers.destroy.timeoutInMinutes`	Time to wait for terminated database server instances to stop running to have ports no longer available. Type: int Default: 20
`lp.cloud.instances.allocate.numberOfRetriesOnConnectionError`	Number of times to retry connecting to newly allocated instances over SSH. Type: int Default: 3

Property	Description
`lp.cloud.instances.allocate.parallelBatchSize`	Parallelism for waiting for SSH to become available on newly allocated instances. Type: int Default: 20
`lp.cloud.instances.allocate.timeBetweenConnectionRetriesInSeconds`	Time to wait between attempts to connect to newly allocated instances over SSH. Type: int Default: 1
`lp.cloud.instances.allocate.timeoutInMinutes`	Time to wait for allocated instances to begin running to have SSH ports available. Type: int Default: 20
`lp.cloud.instances.terminate.timeoutInMinutes`	Time to wait for terminated instances to stop running. Type: int Default: 20
`lp.debug.collectDiagnosticDataOnFailure`	Collect Cloudera Manager diagnostic data on unrecoverable bootstrap failure. Type: boolean Default: true
`lp.debug.createDiagnosticDataDownloadDirectory`	Create the download directory for Cloudera Manager diagnostic data if it does not already exist. Type: boolean Default: true
`lp.debug.diagnosticDataDownloadDirectory`	Destination directory for downloaded Cloudera Manager diagnostic data. Type: string Default: /tmp
`lp.debug.downloadDiagnosticData`	Download Cloudera Manager diagnostic data once it has been collected. Type: boolean Default: true
`lp.debug.dumpClouderaManagerLogsOnFailure`	Dump Cloudera Manager log entries into the Director logs on unrecoverable bootstrap failure. Type: boolean Default: false

Property	Description
`lp.debug.dumpClusterLogsOnFailure`	Dump cluster service logs, standard output, or standard error into the Cloudera Director logs on unrecoverable bootstrap failure. Type: boolean Default: false
`lp.encryption.twoWayCipher`	Cipher used to encrypt data. Possible values: • `desede` - Triple DES • `passthrough` - No encryption • `transitional` - Changing encryption Type: string Default: desede
`lp.encryption.twoWayCipherConfig`	The configuration string for the chosen cipher. Type: string Default: `ZGVmYXVsdGRpcmVjdG9yZGVzZWRla2V5` **Important:** Cloudera recommends that you configure a different triple DES key. A warning appears in the server log if the default key is detected.
`lp.metrics.durationUnits`	Time units for reporting durations in metrics. Type: time unit Valid values: DAYS | HOURS | MICROSECONDS | MILLISECONDS | MINUTES | NANOSECONDS | SECONDS Default: MILLISECONDS
`lp.metrics.enabled`	Enable metrics gathering Type: boolean Default: false
`lp.metrics.location`	Directory for storing metrics reports. Type: directory Default: `$LOG_DIR/metrics`
`lp.metrics.rateUnits`	Time units for reporting rates in metrics. Type: time unit Valid values: DAYS | HOURS | MICROSECONDS | MILLISECONDS | MINUTES | NANOSECONDS | SECONDS Default: SECONDS
`lp.metrics.reportingRate`	Frequency of metrics reporting, in minutes. Type: long Default: 1

Property	Description
`lp.pipeline.retry.maxNumberOfAttempts`	Maximum number of times to retry failed pipeline jobs; -1 = unlimited. Type: int Default: -1 for client, 16 for server
`lp.pipeline.retry.maxWaitBetweenAttempts`	Maximum wait time between pipeline retry attempts, in seconds. Type: int Default: 45
`lp.proxy.http.domain`	NT domain for HTTP proxy authentication; none = no domain. Type: string Default: none
`lp.proxy.http.host`	HTTP proxy host; none = no proxy. Type: string Default: none
`lp.proxy.http.password`	HTTP proxy password; none = no password. Type: string Default: none
`lp.proxy.http.port`	HTTP proxy port; -1 = no proxy. Type: int Default: -1
`lp.proxy.http.preemptiveBasicProxyAuth`	Whether to preemptively authenticate to HTTP proxy. Type: boolean Default: false
`lp.proxy.http.username`	HTTP proxy username; none = no username. Type: string Default: none
`lp.proxy.http.workstation`	Originating workstation in NT domain for HTTP proxy authentication; none = no workstation. Type: string Default: none
`lp.remote.hostAndPort`	Host and port of remote Cloudera Director server. Type: string Default: localhost:7189
`lp.remote.password`	Remote Cloudera Director server password (client only). Type: string

Property	Description
	Default:
`lp.remote.username`	Remote Cloudera Director server username (client only). Type: string Default: none
`lp.remote.terminate.assumeYes`	Whether to skip prompting user to confirm termination for client terminate-remote command. Type: boolean Default: false
`lp.security.enabled`	Whether to enable Cloudera Director server security (server only). Type: boolean Default: true
`lp.security.userSource`	Source for user account information (server only). Type: enum Default: internal
`lp.ssh.connectTimeoutInSeconds`	SSH connection timeout. Type: int Default: 30
`lp.ssh.heartbeatIntervalInSeconds`	SSH heartbeat interval. Type: int Default: 45
`lp.ssh.readTimeoutInSeconds`	SSH read timeout. Type: int Default: 30
`lp.task.evictionRate`	Rate of execution of database eviction, in milliseconds. Type: long Default: 600000
`lp.terminate.assumeYes`	Whether to skip prompting user to confirm termination for client terminate command. Type: boolean Default: false
`lp.terminate.deployment.` `clouderaManagerServerStopWaitTimeInMs`	Time to wait for Cloudera Manager to stop when terminating a deployment, in milliseconds. Type: long Default: 300000

Property	Description
`lp.terminate.deployment.timeBetweenConnectionRetriesInMs`	Time to wait between checks for whether Cloudera Manager has been terminated. Type: int Default: 10000
`lp.update.parallelBatchSize`	Parallelism for allocating and setting up cluster instances when bootstrapping a cluster. Type: int Default: 20
`lp.update.redeployClientConfigs.numberOfRetries`	Maximum number of times to retry deploying Cloudera Manager client configurations; -1 = unlimited. Type: int Default: 5
`lp.update.redeployClientConfigs.sleepAfterFailureInSeconds`	Wait time between attempts to deploy Cloudera Manager client configurations, in seconds. Type: int Default: 10
`lp.update.restartCluster.numberOfRetries`	Maximum number of times to retry a Cloudera Manager rolling restart; -1 = unlimited. Type: int Default: 5
`lp.update.restartCluster.rollingRestartSlaveBatchSize`	Number of instances with Cloudera Manager worker roles to restart at a time. Type: int Default: 20
`lp.update.restartCluster.rollingRestartSlaveFailCountThreshold`	Threshold for number of worker host batches that are allowed to fail to restart before the entire command is considered failed (advanced use only). Type: int Default: 0
`lp.update.restartCluster.rollingRestartSleepSeconds`	Number of seconds to sleep between restarts of Cloudera Manager worker host batches. Type: int Default: 0
`lp.update.restartCluster.sleepAfterFailureInSeconds`	Wait time between attempts to perform a Cloudera Manager rolling restart, in seconds. Type: int Default: 10

Property	Description
`lp.validate.dumpTemplates`	Whether to output validated configuration data as JSON. Type: boolean Default: false
`lp.webapp.anonymousUsageDataAllowed`	Allow Cloudera Director to send anonymous usage information to help Cloudera improve the product. Type: boolean Default: true
`lp.webapp.documentationType`	Whether Cloudera Director opens the latest help from the Cloudera web site (online) or locally installed help (embedded). Type: enumerated string {ONLINE, EMBEDDED} Default: ONLINE
`port`	Cloudera Director server port (server only). Type: int Default: 7189
`server.sessionTimeout`	Cloudera Director server session timeout (server only). Type: int Default: 18000

Setting Cloudera Manager Configurations

You can use Cloudera Director to set configurations for the various Cloudera Manager entities that it deploys:

- Cloudera Manager
- Cloudera Management Service
- The various CDH components, such as HDFS, Hive, and HBase
- Role types, such as NameNode, ResourceManager, and Impala Daemon

This functionality is available for both Cloudera Director client and Cloudera Director server:

- **Client** - Using the configuration file.
- **Server** - Using the Cloudera Director UI or APIs (Java, REST, or Python).
  - To use the REST API, you can submit JSON documents to the REST service endpoint, or access the API console at `http://`*`director-server-hostname`*`:7189/api-console`.
  - You can find information about the Cloudera Director Java and Python APIs on the director-sdk GitHub page.
  - In the UI, you can specify custom values for Cloudera Manager configurations when adding an environment or creating a Cloudera Manager cluster.

Note: Cloudera Manager configuration properties are case-sensitive. To verify the correct way to specify Cloudera Manager configuration properties in Cloudera Director API calls and in the configuration name fields of the Cloudera Director UI, see Cloudera Manager Configuration Properties in the Cloudera Manager documentation. By expanding this heading, you see topics such as the following:

- CDH 5.4.0 Properties
- Host Configuration Properties
- Cloudera Manager Server Properties
- Cloudera Management Service

These pages include tables of configuration properties. Locate the property whose value you want to customize, and use the name in the column **API Name**.

Cloudera Director enables you to customize deployment and cluster setup, and configurations are applied on top of Cloudera Manager default and automatic host-based configuration of services and roles. Set configurations either in the deployment template or in the cluster template.

Cluster Configuration Using Cloudera Manager

Some configuration changes can safely be made to Cloudera Director-managed clusters using Cloudera Manager directly. For these use cases, Cloudera Director will sync up automatically with changes made in Cloudera Manager. Other configuration changes cannot be safely made using Cloudera Manager directly because Cloudera Director will not become aware of the change, resulting in failures when a user later tries to expand or otherwise modify the cluster.

For information on configuration changes and other changes to clusters that can and cannot be safely made directly through Cloudera Manager, see Modifying or Updating Clusters Using Cloudera Manager on page 94.

Setting up a Cloudera Manager License

There are three ways to set up a Cloudera Manager license using Cloudera Director, each corresponding to a field within the `Licensing configuration` section of the `aws.conf` configuration file. The three are mutually exclusive.

- **license field** - You can embed license text in the `license` field of the configuration file. (Cloudera recommends using triple quotes (""") for including multi-line text strings, as shown in the commented-out lines of the configuration file.) To embed a license in the `license` field, find the `Licensing configuration` section of the configuration file and enter the appropriate values.
- **licensePath field** - The `licensePath` field can be used to specify the path to a file containing the license.
- **enableEnterpriseTrial field** - The `enableEnterpriseTrial` flag indicates whether the 60-Day Cloudera Enterprise Trial should be activated when no license is present. This must *not* be set to `true` if a license is included using either `license` or `licensePath`.

The `License configuration` section of the configuration file is shown below:

```
#
# Embed a license for Cloudera Manager
#

# license: """
# -----BEGIN PGP SIGNED MESSAGE-----
# Hash: SHA1
#
# {
# "version" : 1,
# "name" : "License Owner",
# "uuid" : "license id",
# "expirationDate" : 0,
# "features" : [ "FEATURE1", "FEATURE2" ]
# }
# -----BEGIN PGP SIGNATURE-----
# Version: GnuPG v1.4.11 (GNU/Linux)
#
```

```
# PGP SIGNATURE
# -----END PGP SIGNATURE-----
# """

#
# Include a license for Cloudera Manager from an external file
#
# licensePath: "/path/to/license.txt.asc"

#
# Activate 60-Day Cloudera Enterprise Trial
#
enableEnterpriseTrial: true
```

For more information about Cloudera Manager licenses, see Managing Licenses in the Cloudera Manager documentation.

Deployment Template Configuration

This section shows the structure of the Cloudera Manager deployment configuration settings in both the configuration file and the API.

Configuration File

Using the configuration file, the `configs` section in the deployment template has the following structure:

```
cloudera-manager {
   ...
   configs {
     # CLOUDERA_MANAGER corresponds to the Cloudera Manager Server configuration options

      CLOUDERA_MANAGER {
         enable_api_debug: false
      }

     # CLOUDERA_MANAGEMENT_SERVICE corresponds to the Service-Wide configuration options

      CLOUDERA_MANAGEMENT_SERVICE {
         enable_alerts : false
         enable_config_alerts : false
      }

      ACTIVITYMONITOR { ... }

      REPORTSMANAGER { ... }

      NAVIGATOR { ... }

      # Added in Cloudera Manager 5.2+
      NAVIGATORMETASERVER { ... }

      # Configuration properties for all hosts
      HOSTS { ... }
   }
   ...
}
```

API

Using the API, the `configs` section for deployment templates has the following structure:

```
{
   "configs":  {
      "CLOUDERA_MANAGER": {
         "enable_api_debug": "true"
      },
      "CLOUDERA_MANAGEMENT_SERVICE": {
         "enable_alerts": "false"
      }
   }
}
```

Cluster Template Service-wide Configuration

This section shows the structure of the Cloudera Manager service-wide configuration settings in both the configuration file and the API.

Configuration File

Using the configuration file, the `configs` section for service-wide configurations in the cluster template has the following structure:

```
cluster {
   ...
   configs {
      HDFS {
         dfs_block_size: 1342177280
      }
      MAPREDUCE {
         mapred_system_dir: /user/home
         mr_user_to_impersonate: mapred1
      }
   }
   ...
}
```

API

Using the API, the service-wide configurations block in the `ClusterTemplate` is labelled `servicesConfigs`, and has the following structure:

```
{
   "servicesConfigs": {
      "HDFS": {
         "dfs_block_size": 1342177280
      },
      "MAPREDUCE": {
         "mapred_system_dir": "/user/home",
         "mr_user_to_impersonate": "mapred1"
      }
   }
}
```

Cluster Template Roletype Configurations

This section shows the structure of the Cloudera Manager roletype configuration settings in both the configuration file and the API.

Configuration File

Using the configuration file, roletype configurations in the cluster template are specified per instance group:

```
cluster {
   ...
   masters {
      ...
      # Optional custom role configurations
      configs {
         HDFS {
            NAMENODE {
               dfs_name_dir_list: /data/nn
               namenode_port: 1234
            }
         }
      }
      ...
   }
   ...
}
```

API

Using the API, roletype configurations in the cluster template are specified per instance group:

```
{
   "virtualInstanceGroups" : {
      "configs": {
         "HDFS": {
            "NAMENODE": {
                "dfs_name_dir_list": "/data/nn",
                "namenode_port": "1234"
            }
         }
      }
   }
}
```

Configuring Cloudera Director for a New AWS Instance Type

Amazon Web Services occasionally introduces new instance types with improved specifications. Cloudera Director ships with the functionality needed to support all of the instance types available at the time of release, but customers can augment that to allow it to support new types that are introduced after release.

Updated Virtualization Mappings

Each Linux Amazon Machine Image (AMI) uses one of two types of virtualization, paravirtual or HVM. Cloudera Director ensures that the instance type of an instance that is to host an AMI supports the AMI's virtualization type. The knowledge of which instance types support which virtualizations resides in a virtualization mappings file.

The AWS plugin included with Cloudera Director ships with an internal mappings file for all instance types that are available at the time of release. You can add new mappings, or override existing mappings, by creating another custom mappings file. Only new or changed mappings need to be included in the custom mappings file.

The standard location for the custom mappings file is `etc/ec2.virtualizationmappings.properties` under the AWS plugin directory. An example file is provided in the `etc` directory as a basis for customization. You can provide a different location to Cloudera Director by setting the configuration property `lp.ec2.virtualization.customMappingsPath` in one of the usual ways (in `application.properties` or on the command line). If the property is a relative path, it is based on the etc directory under the AWS plugin directory.

Here is an example of a custom mappings file that adds the new "d2" instance types introduced in AWS at the end of March 2015. These new instance types only support HVM virtualization. To keep the example short, many instance types are omitted; in an actual custom mappings file, each property value must provide the full list of instance types that support the property key and virtualizaton type.

```
hvm=m3.medium,\
 m3.large,\
 m3.xlarge,\
 m3.2xlarge,\
 ...
 d2.xlarge,\
 d2.2xlarge,\
 d2.4xlarge,\
 d2.8xlarge
```

To learn more about virtualization types, see Linux AMI Virtualization Types in the AWS documentation.

Updated Ephemeral Device Mappings

Each AWS instance type provides zero or more instance store volumes, also known as ephemeral storage. These volumes are distinct from EBS-backed storage volumes; some instance types include no ephemeral storage. Cloudera Director specifies naming for each ephemeral volume, and keeps a list of the number of such volumes supported per instance type in an ephemeral device mappings file.

The AWS plugin included with Cloudera Director ships with an internal mappings file for all instance types that are available at the time of release. You can add new mappings, or override existing mappings, by creating another custom mappings file. Only new or changed mappings need to be included in the custom mappings file.

The standard location for the custom mappings file is `etc/ec2.ephemeraldevicemappings.properties` under the AWS plugin directory. An example file is provided in the `etc` directory as a basis for customization. You can provide a different location to Cloudera Director by setting the configuration property `lp.ec2.ephemeral.customMappingsPath` in one of the usual ways (in `application.properties` or on the command line). If the property is a relative path, it is based on the etc directory under the AWS plugin directory.

Here is an example of a custom mappings file that describes the new “d2” instance types introduced at the end of March 2015. These new instance types each support a different number of instance store volumes.

```
d2.xlarge=3
d2.2xlarge=6
d2.4xlarge=12
d2.8xlarge=24
```

To learn more about ephemeral storage, including the counts for each instance type, see Instance Stores Available on Instance Types in the AWS documentation.

Using the New Mappings

Once the custom mappings files have been created, restart the Cloudera Director server so that they are detected and overlaid on the built-in mappings.

New instance types do not automatically appear in drop-down menus in the Cloudera Director web interface. However, the selected values for these menus may be edited by hand to specify a new instance type.

Modifying or Updating Clusters Using Cloudera Manager

Some modifications or updates to a Cloudera Director-managed cluster can be made directly in Cloudera Manager. Other changes cannot be safely made directly in Cloudera Manager because Cloudera Director will not become aware of the change, resulting in failures when a user later tries to expand or otherwise modify the cluster.

The following table shows changes that are safe and unsafe to make directly in Cloudera Manager.

Description	Safe Changes	Unsafe Changes
Cloudera Manager or CDH version upgrade	Maintenance upgrades of Cloudera Manager or CDH, where only the 3rd digit of the Cloudera Manager or CDH version changes (for example, 5.4.0 to 5.4.3) are supported. The upgrade must be done using Cloudera Manager; it cannot be done from within Cloudera Director.	Minor or major version upgrades of Cloudera Manager or CDH (for example, 5.4 to 5.5 or 5 to 6) are not supported, even if done outside Cloudera Director.
Configuration changes	Configuration changes made on the Cloudera Manager Server or the Cloudera Management Service are supported if the configurations will not be affected by the addition of new	

Description	Safe Changes	Unsafe Changes
	hosts to a cluster, that is, the configurations will not need to be set up on the new host. Configuration changes made to hosts, to CDH services and roles, or to the Cloudera Manager Server in areas that *will* impact future hosts (for example, parcels or agent-related configurations), are supported, but only in the present state of the deployment and cluster. You can use the cluster with these changes, but future modifications to the cluster from Cloudera Director will not propagate the changes to new hosts or roles you add to the cluster, since Cloudera Director will not be aware of the changes.	
Role assignment		Adding new roles to hosts directly in Cloudera Manager is not supported.
Role migration		Migrating roles from one host to another within Cloudera Manager is not supported.
Decommissioning hosts outside Cloudera Director		Cloudera recommends that you do not decommission hosts using Cloudera Manager directly. Doing so puts Cloudera Director and Cloudera Manager out of sync with one another and can negatively impact future operations on the cluster.
Adding new hosts or clusters outside of Cloudera Director		Adding new hosts to existing clusters or adding a new cluster to Cloudera Manager, done directly in Cloudera Manager is not supported. Both these should be done from within Cloudera Director.
Changing Cloudera Manager username and password	This can be done in Cloudera Manager, but Cloudera Director has to be updated to be made aware of the new values.	
A Cloudera Manager cluster that was set up using Cloudera Manager, rather than through Cloudera Director		Not supported. Cloudera Director is not aware of clusters set up directly in Cloudera Manager.
Enabling Kerberos outside of Cloudera Director		Do not enable Kerberos through Cloudera Manager directly. Enable Kerberos using Cloudera Director 2.0 or higher. Use the configuration file, not the Cloudera Director UI, to enable Kerberos.

Description	Safe Changes	Unsafe Changes
Modifying a cluster after enabling high availability outside Cloudera Director	If HDFS high availability has been enabled outside Cloudera Director, using Cloudera Manager directly, then you can run future **Modify** operations on the cluster, but only on those instance groups that do not contain highly available master roles on Cloudera Manager.	Do not enable high availability through Cloudera Manager directly, do so using Cloudera Director 2.0. Use the configuration file, not the Cloudera Director UI, to enable high availability.
Upgrading your Cloudera Manager license	Upgrading a license using Cloudera Manager, for example, from Cloudera Express to Cloudera Enterprise, is supported.	

Post-Creation Scripts

Post-creation scripts are run after a Cloudera Manager cluster has been created. The scripts are run sequentially on a randomly selected cluster host. The scripts can be written in any scripting language that can be interpreted on the system where it runs.

Configuring the Scripts

Post-creation scripts are available only through the client configuration file or the Cloudera Director API.

You can supply post-creation scripts in the client configuration file in two ways:

- Use the `postCreateScripts` directive inside of the cluster {} configuration block. This block can take an array of scripts, similar to the `bootstrapScript` that can be placed inside the instance {} configuration block.
- Use the `postCreateScriptsPaths` directive inside of the cluster {} configuration block. It can take an array of paths to arbitrary files on the local filesystem. This is similar to the `bootstrapScriptPath` directive. Cloudera Director reads the files from the filesystem and uses their contents as post-creation scripts.

Unlike `bootstrapScript` and `bootstrapScriptPath`, both post-creation scripting methods can be used simultaneously. For example, `postCreateScripts` can be used for setup (package installation, light system configuration), and `postCreateScriptsPaths` can be used to refer to more complex scripts that may depend on the configuration that was performed in `postCreateScripts`. Everything in the `postCreateScripts` block is run first, sequentially, and then everything in `postCreateScriptsPaths` is run sequentially.

```
cluster {
   ....
   postCreateScripts: [#!/usr/bin/python]
   print 'Hello World Again!'

   #!/bin/bash
   echo 'Hello World!',

   postCreateScriptsPaths: ["/tmp/script1.py", "/tmp/script2.sh"]
```

Predefined Environment Variables

Post-creation scripts have access to several environment variables defined by Cloudera Director. Use these variables in your scripts to communicate with Cloudera Manager and configure it after Cloudera Director has completed its tasks.

Variable Name	Example	Description
`DEPLOYMENT_HOST_PORT`	192.168.1.100:7180	The host and port used to connect to the Cloudera Manager deployment that this cluster belongs to.
`ENVIRONMENT_NAME`	Cloudera Director Environment	The name of the environment that this cluster belongs to.
`DEPLOYMENT_NAME`	Cloudera Director Deployment	The name of the Cloudera Manager deployment that this cluster belongs to.
`CLUSTER_NAME`	Cloudera Director Cluster	The name of the cluster. The Cloudera Manager API needs this to specify which cluster on a Cloudera Manager server to operate on.
`CM_USERNAME`	admin	The username needed to connect to the Cloudera Manager deployment.
`CM_PASSWORD`	admin	The password needed to connect to the Cloudera Manager deployment.

Creating Kerberized Clusters With Cloudera Director

Using Cloudera Director 2.0 and higher with Cloudera Manager 5.5.0 and higher, you can create and configure Kerberized Cloudera Manager clusters. To launch a Kerberized cluster, edit the configuration file as described below and launch the cluster with Cloudera Director client, using the `bootstrap-remote` command to send the configuration file to a running Cloudera Director server.

Note: You must have an existing Kerberos Key Distribution Center (KDC) set up, and it must be reachable by the instance where Cloudera Director server is running and the instances where your Cloudera Manager cluster will be deployed. You must also set up a Kerberos realm for the cluster and a principal in that realm.

Important: Do not use Cloudera Manager to enable Kerberos on an existing cluster that is managed by Cloudera Director. Kerberos must be enabled through Cloudera Director using the configuration file.

Creating a Kerberized Cluster with the Cloudera Director Configuration File

A sample configuration file for creating Kerberized Cloudera Manager clusters is available on the Cloudera GitHub site: director-scripts/kerberos/aws.kerberos.sample.conf.

The settings for enabling Kerberos are in the Cloudera Manager section of the configuration file. Provide values for the following configuration settings:

Configuration setting	Description
krbAdminUsername	An administrative Kerberos account with permissions that allow the creation of principals on the KDC that Cloudera Manager will be using. This is typically in the format *principal@your.KDC.realm*
krbAdminPassword	The password for the administrative Kerberos account.

Configuration setting	Description
KDC_TYPE	The type of KDC Cloudera Manager will use. Valid values are "MIT KDC" and "Active Directory".
KDC_HOST	The hostname or IP address of the KDC.
SECURITY_REALM	The security realm that the KDC uses.
AD_KDC_DOMAIN	The Active Directory KDC domain in the format of an X.500 Directory Specification (`DC=domain,DC=example,DC=com`). This setting is for Active Directory KDCs only.
KRB_MANAGE_KRB5_CONF	Set this to `true`. This allows Cloudera Manager to deploy Kerberos configurations to cluster instances. The value `false` is not supported for this configuration setting.
KRB_ENC_TYPES	The encryption types your KDC supports. Some of encryption types listed in the sample configuration file require the unlimited strength JCE policy files.

Other Kerberos configuration options are available to Cloudera Manager. For more information, see Configuring Authentication in the Cloudera Security guide.

The following example shows the cloudera-manager section of a configuration file with MIT KDC Kerberos enabled:

```
cloudera-manager {
   instance: ${instances.cm-image} {
      tags {
         application: "Cloudera Manager 5"
      }
   }

#
# Automatically activate 60-Day Cloudera Enterprise Trial
#
   enableEnterpriseTrial: true

   unlimitedJce: true
# Kerberos principal and password for use by Cloudera Director
   krbAdminUsername: "principal@my.kdc.realm"
   krbAdminPassword: "password"

# Cloudera Manager configuration values
   configs {
      CLOUDERA_MANAGER {
         KDC_TYPE: "MIT KDC"
         KDC_HOST: "KDC_host_ip_address"
         SECURITY_REALM: "my_security_realm"
         KRB_MANAGE_KRB5_CONF: true
        KRB_ENC_TYPES: "aes256-cts aes128-cts des3-hmac-sha1 arcfour-hmac des-hmac-sha1
 des-cbc-md5 des-cbc-crc"
      }
   }
}
```

Creating Highly Available Clusters With Cloudera Director

Using Cloudera Director 2.0 or higher and Cloudera Manager 5.5 or higher, you can launch highly available clusters for HDFS, YARN, ZooKeeper, HBase, Hive, Hue, and Oozie. The services are highly available on cluster launch with no additional setup. To enable high availability, edit the Cloudera Director configuration file as described in this topic and launch the cluster with the Cloudera Director client and the `bootstrap-remote` command, which sends the configuration file to a running Cloudera Director server.

Note: With Cloudera Director 1.5 and Cloudera Manager 5.4, you can set up a highly available cluster by running a script after the cluster is launched. For more information, see the high-availability scripts and the README file on the Cloudera Director GitHub site.

Limitations and Restrictions

The following limitations and restrictions apply to creating highly available clusters with Cloudera Director:

- The procedure described in this section works with Cloudera Director 2.0 or higher and Cloudera Manager 5.5 or higher.
- Cloudera Director does not support migrating a cluster from a non-high availability setup to a high availability setup.
- Cloudera recommends sizing the master nodes large enough to support the desired final cluster size.
- Settings must comply with the configuration requirements described below and in the `aws.ha.reference.conf` file. Incorrect configurations can result in failures during initial bootstrap.

Editing the Configuration File to Launch a Highly Available Cluster

Follow these steps to create a configuration file for launching a highly available cluster.

1. Download the sample configuration file `aws.ha.reference.conf` from the Cloudera GitHub site. The cluster section of the file shows the role assignments and required configurations for the services where high availability is supported. The file includes comments that explain the configurations and requirements.
2. Copy the sample file to your home directory before editing it. Rename the `aws.ha.reference.conf` file, for example, to `ha.cluster.conf`. The configuration file must use the `.conf` file extension. Open the configuration file with a text editor.

   Note: The sample configuration file includes configuration specific to Amazon Web Services, such as the section for cloud provider credentials. The file can be modified for other cloud providers by copying sections from the other cloud provider-specific sample files, for example, gcp.simple.conf.

3. Edit the file to supply your cloud provider credentials and other details about the cluster. A highly available cluster has additional requirements, as seen in the sample `aws.ha.reference.conf` file. These requirements include duplicating the master roles for highly available services.

The sample configuration file includes a set of instance groups for the services where high availability is supported. An instance group specifies the set of roles that are installed together on an instance in the cluster. The master roles in the sample `aws.ha.reference.conf` file are included in four instance groups, each containing particular roles. The names of the instance groups are arbitrary, but the names used in the sample file are hdfsmasters-1, hdfsmasters-2, masters-1, and masters-2. You can create multiple instances in the cluster by setting the value of the `count` field for the instance group. The sample file is configured for two hdfsmasters-1 instances, one hdfsmasters-2 instance, two masters-1 instances, and one masters-2 instance.

The cluster services for which high availability is supported are listed below, with the minimum number of roles required and other requirements.

- HDFS
  - Two NAMENODE roles.
  - Three JOURNALNODE roles.
  - Two FAILOVERCONTROLLER roles, each colocated to run on the same host as one of the NAMENODE roles (that is, included in the same instance group).
  - One HTTPFS role if the cluster contains a Hue service.
  - The NAMENODE nameservice, autofailover, and quorum journal name must be configured for high availability exactly as shown in the sample `aws.ha.reference.conf` file.

  - Set the HDFS service-level configuration for fencing as shown in the sample `aws.ha.reference.conf` file:

```
configs {
                # HDFS fencing should be set to true for HA configurations
                HDFS {
                dfs_ha_fencing_methods: "shell(true)"
                }
```

  - Three role instances are required for the HDFS JOURNALNODE role. This ensures a quorum for determining which is the active node and which are standbys.

  For more information, see HDFS High Availability in the Cloudera Administration documentation.

- YARN
  - Two RESOURCEMANAGER roles.
  - One JOBHISTORY role.

  For more information, see YARN (MRv2) ResourceManager High Availability in the Cloudera Administration documentation.

- ZooKeeper
  - Three SERVER roles (recommended). There must be an odd number, but one will not provide high availability
  - Three role instances are required for the ZooKeeper SERVER role. This ensures a quorum for determining which is the active node and which are standbys.

- HBase
  - Two MASTER roles.
  - Two HBASETHRIFTSERVER roles (needed for Hue).

  For more information, see HBase High Availability in the Cloudera Administration documentation.

- Hive
  - Two HIVESERVER2 roles.
  - Two HIVEMETASTORE roles.

  For more information, see Hive Metastore High Availability in the Cloudera Administration documentation.

- Hue
  - Two HUESERVER roles.
  - One HTTPFS role for the HDFS service.

  For more information, see Hue High Availability in the Cloudera Administration documentation.

- Oozie
  - Two SERVER roles.
  - Oozie plug-ins must be configured for high availability exactly as shown in the sample `aws.ha.reference.conf` file. In addition to the required Oozie plug-ins, other Oozie plug-ins can be enabled. All Oozie plug-ins must be configured for high availability.
  - Oozie requires a load balancer for high availability. Cloudera Director does not create or manage the load balancer. The load balancer must be configured with the IP addresses of the Oozie servers after the cluster completes bootstrapping.

  For more information, see Oozie High Availability in the Cloudera Administration documentation.

- The following requirements apply to databases for your cluster:
  - You can configure external databases for use by the services in your cluster and for Cloudera Director. If no databases are specified in the configuration file, an embedded PostgreSQL database is used.

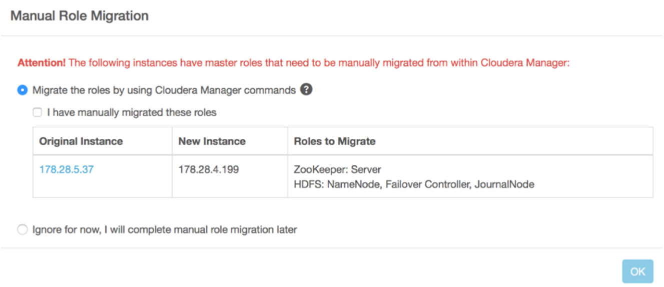
- External databases can be set up by Cloudera Director, or you can configure preexisting external databases to be used. Databases set up by Cloudera Director are specified in the `databaseTemplates` block of the configuration file. Preexisting databases are specified in the `databases` block of the configuration file. External databases for the cluster must be either all preexisting databases or all databases set up by Cloudera Director; a combination of these is not supported.
- Hue, Oozie, and the Hive metastore each require a database.
- Databases for highly available Hue, Oozie, and Hive services must themselves be highly available. An Amazon RDS MySQL Multi-AZ deployment, whether preexisting or configured to be created by Cloudera Director, satisfies this requirement.

Using Role Migration to Repair HDFS Master Role Instances

Cloudera Director supports exact one-for-one host replacement for HDFS master role instances. This is a partially manual process that requires migration of the roles in Cloudera Manager. If a host running HDFS master roles (NameNode, Failover Controller, and JournalNode) fails in a highly available cluster, you can use Cloudera Director and the Cloudera Manager Role Migration wizard to move the roles to another host without losing the role states, if any. The previously standby instance of each migrated role runs as the active instance. When the migration is completed, the role that runs on the new host becomes the standby instance.

Keep in mind the following when performing HDFS role migration:

- Do not modify any instance groups on the cluster during the repair and role migration process.
- Do not clone the cluster during the repair and role migration process.
- To complete the migration (Step 3 below), click a checkbox to indicate that the migration is done, after which the old instance is terminated. Check Cloudera Manager to ensure that the old host has no roles or data on it before performing this step in Cloudera Director. Once the old instance is terminated, any information or state it contained is lost.
- If you have completed Step 1 (on Cloudera Director) and intend to complete Step 2 (on Cloudera Manager) at a later time, you can confirm which IP address to migrate from or to by going to the cluster status page in Cloudera Director and clicking either the link for migration in the upper left, or the **Modify Cluster** button on the right. A popup displays the hosts to migrate from and to:



Manual Role Migration

Attention! The following instances have master roles that need to be manually migrated from within Cloudera Manager:

● Migrate the roles by using Cloudera Manager commands ?

□ I have manually migrated these roles

Original Instance	New Instance	Roles to Migrate
178.28.5.37	178.28.4.199	ZooKeeper: Server HDFS: NameNode, Failover Controller, JournalNode

○ Ignore for now, I will complete manual role migration later

- You do not need to check the boxes to restart and deploy client configuration at the start of the repair process. You restart and deploy the client configuration manually after role migration is complete.
- Do not attempt repair for non-highly available master roles. The Cloudera Manager Role Migration wizard only works for high availability HDFS roles.

Step 1: In Cloudera Director, Create a New Instance

1. In Cloudera Director, click the cluster name and click **Modify Cluster**.
2. Click the checkbox next to the IP address of the failed instance (containing the HDFS NameNode and colocated Failover Controller, and possibly a JournalNode). Click **Repair**.
3. Click **OK**. You do not need to select **Restart Cluster** at this time, because you will restart the cluster after migrating the HDFS master roles.

Cloudera Director creates a new instance on a new host, installs the Cloudera Manager agent on the instance, and copies the Cloudera Manager parcels to it.

Step 2: In Cloudera Manager, Migrate Roles and Data

Open the cluster in Cloudera Manager. On the **Hosts** tab, you see a new instance with no roles. The cluster is in an intermediate state, containing the new host to which the roles will be migrated and the old host from which the roles will be migrated.

Use the Cloudera Manager **Migrate Roles** wizard to move the roles.

See Moving Highly Available NameNode, Failover Controller, and JournalNode Roles Using the Migrate Roles Wizard in the Cloudera Administration guide.

Step 3: In Cloudera Director, Delete the Old Instance

1. Return to the cluster in Cloudera Director.
2. Click **Details**. The message "Attention: Cluster requires manual role migration" is displayed. Click **More Details**.
3. Check the box labeled, "I have manually migrated these roles."
4. Click **OK**.

The failed instance is deleted from the cluster.

Enabling Sentry Service Authorization

This topic describes how to enable the Sentry service with Cloudera Director.

Prerequisites

- Cloudera Director 1.1.x
- CDH 5.1.x (or higher) managed by Cloudera Manager 5.1.x (or higher).
- Kerberos authentication implemented on your cluster.

Setting Up the Sentry Service Using the Cloudera Director CLI

This method requires you to send configuration files that the Cloudera Director server can use to deploy clusters. See Submitting a Cluster Configuration File for more details. Make sure you add SENTRY to the array of `services` to be launched. This is specified in the configuration file as:

```
services: [HDFS, YARN, ZOOKEEPER, HIVE, OOZIE, HUE, IMPALA, SENTRY]
```

To specify a database, use the `databases` setting as follows:

```
cluster {
...
  databases {
      SENTRY: {
        type: mysql
        host: sentry.db.example.com
        port: 3306
        user: <database_username>
        password: <database_password>
        name: <database_name>
      }
  }
}
```

The Sentry service also requires the following custom configuration for the MapReduce, YARN, HDFS, Hive, and Impala Services.

- **MapReduce:** Set the **Minimum User ID for Job Submission** property to zero (the default is 1000) for *every* TaskTracker role group that is associated with Hive.

```
MAPREDUCE {
    TASKTRACKER {
        taskcontroller_min_user_id: 0
    }
}
```

- **YARN:** Ensure that the **Allowed System Users** property, for *every* NodeManager role group that is associated with Hive, includes the hive user.

```
YARN {
    NODEMANAGER {
        container_executor_allowed_system_users: hive, impala, hue
    }
}
```

- **HDFS:** Enable HDFS extended ACLs.

```
HDFS {
    dfs_permissions: true
    dfs_namenode_acls_enabled: true
}
```

  With Cloudera Manager 5.3 and CDH 5.3, you can enable synchronization of HDFS and Sentry permissions for HDFS files that are part of Hive tables. For details on enabling this feature using Cloudera Manager, see Synchronizing HDFS ACLs and Sentry Permissions.

- **Hive:** Make sure Sentry policy file authorization has been disabled for Hive.

```
HIVE {
    sentry_enabled: false
}
```

- **Impala:** Make sure Sentry policy file authorization has been disabled for Impala.

```
IMPALA {
    sentry_enabled: false
}
```

Set Permissions on the Hive Warehouse

Once setup is complete, configure the following permissions on the Hive warehouse. For Sentry authorization to work correctly, the Hive warehouse directory (/user/hive/warehouse or any path you specify as hive.metastore.warehouse.dir in your hive-site.xml) must be owned by the Hive user and group.

- Permissions on the warehouse directory must be set as follows:
  - **771** on the directory itself (for example, /user/hive/warehouse)
  - **771** on all subdirectories (for example, /user/hive/warehouse/mysubdir)
  - All files and subdirectories must be owned by hive:hive

  For example:

```
$ sudo -u hdfs hdfs dfs -chmod -R 771 /user/hive/warehouse
$ sudo -u hdfs hdfs dfs -chown -R hive:hive /user/hive/warehouse
```

Setting up the Sentry Service Using the Cloudera Director API

You can use the Cloudera Director API to set up Sentry. Define the ClusterTemplate to include Sentry as a service, along with the configurations specified above, but in JSON format.

Set permissions on the Hive warehouse as described above.

Related Links

For detailed instructions on adding and configuring the Sentry service, see Installing and Upgrading the Sentry Service and Configuring the Sentry Service.

Examples on using Grant/Revoke statements to enforce permissions using Sentry are available at Hive SQL Syntax.

Using Spot Instances

To help manage cloud resource costs, Cloudera supports Spot instances. Spot instances are Amazon EC2 instances that you can bid on. Unlike On-Demand Amazon EC2 instances, Spot instances only run as long as the price you bid exceeds the current Spot price. This allows you to add capacity to your workload at a low price.

Spot instances run just like On-Demand instances, except that they are not provisioned until the instance price falls below your bid. They also terminate automatically when the instance price exceeds or equals your bid price.

For more information about using Spot instances, see the Amazon EC2 documentation. For help with bidding on Spot instances, see the Spot Bid Advisor.

Planning for Spot Instances

It is normal for Spot instances on a cluster to disappear over time. However, Cloudera Manager does not see that these instances are terminated. If you restart a cluster that contains a Spot instance group, and the Spot instances have terminated, the restart fails. If you are modifying any group in the cluster that has lost Spot instances, do not select the **Restart** checkbox.

If your bid price is so low that you do not obtain an instance when the group is created, you will have 0 instances in your group. If this happens, you can:

- Delete the entire group.
- Add more instances to the group.
- Delete unprovisioned instances from the group (only as part of adding more instances to the group).
- Retry (repair) existing instances.

You cannot do the following:

- Change the bid price
- Delete all instances without adding more

The bid price for Spot instances is set in an instance template. This template is associated with a group. Although you can modify the group, you cannot change the bid price. Therefore, if you set the bid price too low for successful provisioning, you must delete the group where that price is set and create a new group with the higher bid price. You must also delete the current group and create a new one if you want to drop the bid price.

Specifying Spot Instances

To specify Spot instances, create a new instance template and use this template for your group. For more information, see the steps for adding a cluster in the Deploying Cloudera Manager and CDH on AWS topic.

Best Practices for Using Spot Instances

- Use a Spot instance worker group in conjunction with an On-Demand worker group. This ensures that the cluster can redo computational tasks run on Spot instances that could be terminated before the tasks are finished.
- Use Spot instances only in contexts where the loss of the instance can be tolerated, as in a worker group. Do not use Spot instances for master nodes or for data storage.
- Use a minimum count of `0` for Spot instance groups. If you use a number above 0, the cluster will likely enter a failed state. If the cluster fails, contact Cloudera support for help.

Upgrading Cloudera Director

This section contains notes and procedures for upgrading Cloudera Director.

Before Upgrading Cloudera Director

Follow these steps before upgrading Cloudera Director.

1. Let running operations finish.

   For example, if Cloudera Director is setting up a Cloudera Manager or CDH cluster (indicated by a progress bar in the UI), an upgrade will not complete successfully. An error in the log file instructs you to use the old version of Cloudera Director until all running operations are completed, and then perform the upgrade.

2. Back up the Cloudera Director database that stores state information.

   By default, this is the embedded H2 database at `/var/lib/cloudera-director-server/state.h2.db`.

   If you are using a MySQL database to store the Cloudera Director state, use MySQL backup procedures to back up the Cloudera Director database. The following example shows how to do this using the `mysqldump` utility:

```
mysqldump --all-databases --single-transaction --user=root --password > backup.sql
```

   For more information on using `mysqldump`, see the MySQL documentation.

3. Change your default encryption key.

   After an upgrade from Cloudera Director 1.1 to a higher version, any new data that Cloudera Director persists in its database is encrypted with a default encryption key. For increased security, Cloudera recommends that you change your encryption key in the `application.properties` file after performing an upgrade from 1.1 to a higher version. The file is located at `/etc/cloudera-director-server/application.properties`.

   For more information about encryption and Cloudera Director data, see Cloudera Director Database Encryption on page 70.

Changes to the `application.properties` File

If you modified your existing `application.properties` file, the result of upgrading depends on which version of Linux you are using:

- **RHEL and CentOS** - When new properties are introduced in Cloudera Director, they are added to `application.properties.rpmnew`. The original `application.properties` file functions as before and is not overwritten with the new Cloudera Director version properties. You do not need to copy the new properties from `application.properties.rpmnew` to the old `application.properties` file.
- **Ubuntu** - The modified Cloudera Director `application.properties` file is backed up to a file named `application.properties.dpkg-old`. The original `application.properties` file is then overwritten by the new `application.properties` file containing new Cloudera Director properties. After upgrading, copy your changes from `application.properties.dpkg-old` to the new `application.properties` file.

Supported Operating Systems for Cloudera Director

Cloudera Director 2.0.0 and higher support the following Linux operating systems:

- RHEL and CentOS 6.5, 6.7, and 7.1
- Ubuntu 14.04

If you are running a lower version of Cloudera Director on an operating system that is not supported for Cloudera Director 2.0, you cannot upgrade to version Cloudera Director 2.0.

Handling Modified Plug-in Configuration Files

Cloudera Director includes plug-in configuration files that enable you to configure how the plug-ins work. The following plug-in files are located in directories in `/var/lib/cloudera-director-server/plugins/`:

- aws-provider-*version*
- byon-provider-example-*version*
- google-provider-*version*
- sandbox-provider-*version*

You do not normally need to modify these files, but if you have modified any of them, back up the modified files to another location before running the upgrade command. Then, restore your backed-up copies after the upgrade. Both of these steps are included in the upgrade procedures below.

> **Note:** The location for plug-in configuration files has changed starting with Cloudera Director 2.0. In Cloudera Director 1.5.x and lower, they are located at `/var/lib/cloudera-director-server/plugins`. In Cloudera Director 2.0 and higher, they are located at `/var/lib/cloudera-director-plugins/`.

Changes in Cloudera Director 2.0

- Cloudera Director now requires Oracle JDK (Oracle Java SE Development Kit) version 7 or 8. Java 6 is not supported.
- Cloudera Director 2.0 can install any version of Cloudera Manager 5 with any CDH 5 parcels. Cloudera Manager 4 and CDH 4 are not supported. Use of CDH packages is not supported.
- Improved validation of `create` and `update` endpoints allows Cloudera Director to fail faster in many cases.
- General API changes:
  - Listing the external database servers for a nonexistent environment now returns `404 Not Found`.
  - The response codes for the v4 endpoints has been changed from `500 Internal Server Error` to `204 No Content` under the following conditions:
    - Attempting to GET a `Deployment` when the deployment is in a failed state.
    - Attempting to GET an `ExternalDatabaseServer` when the external database server is in a failed state.
    - Attempting to GET the `Cluster` when the cluster is in a failed state.

Upgrading Cloudera Director

The following sections describe steps for upgrading Cloudera Director on supported Linux operating systems.

RHEL and CentOS

1. Stop the Cloudera Director server service by issuing the following command:

```
sudo service cloudera-director-server stop
```

2. Cloudera Director 2.0.x requires Java 7 or 8. If you must upgrade your version of the Java SDK to meet this requirement, do so now.
3. Update your Cloudera Director `.repo` file (the yum repository configuration file) to point to the version of Cloudera Director you are upgrading to by doing one of the following:
   - Open `/etc/yum.repos.d/cloudera-director.repo`. The `baseurl` value in this file now points to your current version of Cloudera Director, such as `/1/` (and may include a specific minor or maintenance release version, such as `/1.1/` or `/1.1.3/`). Update the `baseurl` value to point to the new version, `/2/`.

- Instead of editing your existing `.repo` file, you can download a new Cloudera Director `.repo` file, which will point to the latest version of Cloudera Director:

```
cd /etc/yum.repos.d/
sudo wget "http://archive.cloudera.com/director/redhat/7/x86_64/director/cloudera-director.repo"
```

  To upgrade to a version of Cloudera Director other than the latest version, you can edit the newly downloaded `.repo` file as described in the previous bullet point.

4. If you have not modified the plug-in configuration files, skip to the next step. If you modified the plug-in configuration files in /var/lib/cloudera-director-server/plugins/*plug-in_name-version*/etc, back them up to another location before running the upgrade command.
5. Issue the following commands:

```
sudo yum clean all
sudo yum update cloudera-director-server cloudera-director-client
```

6. If you have not modified the plug-in configuration files, skip to the next step. If you modified the plug-in configuration files in /var/lib/cloudera-director-server/plugins/*plug-in_name-version*/etc, restore your backed up files now to the new location, /var/lib/cloudera-director-plugins/*plug-in_name-new_version*/etc, before restarting the Cloudera Director server.
7. Restart the Cloudera Director server:

```
sudo service cloudera-director-server start
```

> **Note:** Installing the Cloudera Director server and client packages will automatically install the required plug-in package.

Ubuntu

1. Stop the Cloudera Director server service by issuing the following command:

```
sudo service cloudera-director-server stop
```

2. Cloudera Director 2.0.x requires Java 7 or 8. If you must upgrade your version of the Java SDK to meet this requirement, do so now.
3. Update your Cloudera Director `cloudera-director.list` file (the repository configuration file) to point to the version of Cloudera Director you are upgrading to by doing one of the following:
   - Open `/etc/apt/sources.list.d/cloudera-director.list`. The `baseurl` value in this file now points to your current version of Cloudera Director, such as `trusty-director1` (and may include a specific minor or maintenance release version, such as `trusty-director1.1` or `trusty-director1.1.3`). Update the `baseurl` value to point to the new version, `trusty-director2`.
   - Instead of editing your existing `cloudera-director.list` file, you can download a new Cloudera Director `cloudera-director.list` file, which will point to the latest version of Cloudera Director:

```
cd /etc/apt/sources.list.d/
sudo curl "http://archive.cloudera.com/director/ubuntu/trusty/amd64/director/cloudera-director.list"
```

  To upgrade to a version of Cloudera Director other than the latest version, you can edit the newly downloaded `cloudera-director.list` file as described in the previous bullet point.

4. If you have not modified the plug-in configuration files, skip to the next step. If you modified the plug-in configuration files in /var/lib/cloudera-director-server/plugins/*plug-in_name-version*/etc, back them up to another location before running the upgrade command.

5. Issue the following commands:

```
sudo apt-get clean
sudo apt-get update
sudo apt-get dist-upgrade
sudo apt-get install cloudera-director-server cloudera-director-client
```

6. If your original Cloudera Director `application.properties` file has not been modified, proceed to the next step. If your `application.properties` file was modified, the original properties file will be overwritten by the new properties file containing new Cloudera Director properties, as described above in [Changes to the application.properties File]() on page 105. Copy your changes from `application.properties.dpkg-old` to the new `application.properties` file before restarting the server.
7. If you have not modified the plug-in configuration files, skip to the next step. If you modified the plug-in configuration files in `/var/lib/cloudera-director-server/plugins/`*`plug-in_name-version`*`/etc`, restore your backed up files now to the new location, `/var/lib/cloudera-director-plugins/`*`plug-in_name-new_version`*`/etc`, before restarting the Cloudera Director server.
8. Restart the Cloudera Director server:

```
sudo service cloudera-director-server start
```

> **Note:** Installing the Cloudera Director server and client packages will automatically install the required plug-in package.

Using IAM Policies with Cloudera Director 1.5 and Higher

In AWS, if you are using an IAM policy to control access to resources in the VPC, Cloudera Director 1.5 and higher requires permission for the method `DescribeDBSecurityGroups`. To give Cloudera Director permission for this method, add these values to your policy:

```
{
  "Action": [ "rds:DescribeDBSecurityGroups" ],
  "Effect": "Allow",
  "Resource": ["*"]
}
```

This permission is required because Cloudera Director 1.5 and higher includes early validation of RDS credentials when creating or updating an environment, whether or not RDS database servers are used.

For a sample IAM policy that includes this permission, see [Example IAM Policy]() on page 61. For more information on AWS IAM, see the [IAM User Guide]() in the AWS documentation.

Troubleshooting Cloudera Director

This topic contains information on issues, causes, and solutions for problems you might face when setting up, configuring, or using Cloudera Director.

Viewing Cloudera Director Logs

To help you troubleshoot problems, you can view the Cloudera Director logs. Log files can be found in the following locations:

- Cloudera Director Client
  - One shared log file per user account:

```
$HOME/.cloudera-director/logs/application.log
```

- Cloudera Director Server
  - One file for all clusters:

```
/var/log/cloudera-director-server/application.log
```

Backing Up the H2 Embedded Database

By default, Cloudera Director uses an H2 embedded database to store environment and cluster data. The H2 embedded database file is located at:

```
/var/lib/cloudera-director-server/state.h2.db
```

Back up the `state.h2.db` file to avoid losing environment and cluster data. To ensure that your backup copy can be restored, you should use the H2 backup tools and rather than simply copying the file. For more information, see the H2 Tutorial.

Cloudera Director Cannot Manage a Cluster That Was Kerberized Through Cloudera Manager

Symptom

Cloudera Director cannot manage a cluster after Cloudera Manager is used to enable Kerberos on the cluster.

Cause

Once a cluster is deployed via Cloudera Director, some changes to the cluster that are made using Cloudera Manager cause Cloudera Director to be out of sync, and hence unable to manage the cluster. See Modifying or Updating Clusters Using Cloudera Manager.

Solution

Deploy a new kerberized cluster, use `distcp` to transfer data from the old cluster to the new one, and then destroy the old cluster.

New Cluster Fails to Start Because of Missing Roles

Symptom

A new cluster will not start because roles are missing.

Cause

Cloudera Director does not validate that all necessary roles are assigned when provisioning a cluster. This can lead to failures during the intial run of a new cluster. For example, if the gateway instance group was removed but the Flume Agent and Kafka Broker were assigned to roles in that group, the cluster will fail to start.

Solution

Ensure that all required role types for the CDH services included in the cluster are assigned to instances before starting the cluster.

Cloudera Director Server Will Not Start with Unsupported Java Version

Symptom

Cloudera Director server will not start, and `/var/log/cloudera-director-server/cloudera-director-server.out` has the following error:

```
Exception in thread "main" java.lang.UnsupportedClassVersionError:
com/cloudera/launchpad/Server : Unsupported major.minor version 51.0
```

Cause

You are running Cloudera Director server against an older, unsupported version of the Oracle Java SE Development Kit (JDK).

Solution

Update to Oracle JDK version 7 or 8.

Error Occurs if Tags Contain Unquoted Special Characters

Symptom

When using the configuration file with the `bootstrap` command to start Cloudera Director client, or using the `bootstrap-remote` command to set up a cluster with Cloudera Director server, an error message is displayed. This applies to HOCON characters, and includes periods. If the added configuration is in the form x.y, for example, the following error message may be displayed: "`com.typesafe.config.ConfigException$WrongType: ... <x> has type OBJECT rather than STRING`". This means that x.y must be in quotes, as in "`x.y`".

```
com.typesafe.config.ConfigException$WrongType: ... <x> has type OBJECT rather than STRING
```

Cause

Cloudera Director validation checks to ensure that special characters in configurations are enclosed in double quotes.

Solution

Use double quotes for special characters in configurations. An example of a configuration that would require double quotes is `"log.dirs"` in Kafka.

DNS Issues

Symptom

Director fails to bootstrap a cluster with a DNS error.

Cause

This can be caused by a couple of things:

- The **Edit DNS Hostnames** is not set to **Yes** the VPC settings.
- The Amazon Virtual Private Cloud (VPC) is not set up for forward and reverse hostname resolution. Functional forward and reverse DNS resolution is a key requirement for many components of the Cloudera EDH platform, including Cloudera Director.

Solutions

In the AWS Management Console, go to **Services** > **Networking** and click **VPC**. In the VPC Dashboard, select your VPC and click **Action**. In the shortcut menu, click **Edit DNS Hostnames** and click **Yes**. If this does not fix the issue, continue with the instructions that follow to configure forward and reverse hostname resolution.

Configure the VPC for forward and reverse hostname resolution. You can verify if DNS is working as expected on a host by issuing the following one-line Python command:

```
python -c "import socket; print socket.getfqdn(); print
socket.gethostbyname(socket.getfqdn())"
```

For more information on DNS and Amazon VPCs, see DHCP Options Sets in the Amazon VPC documentation.

If you are using Amazon-provided DNS, perform these steps to configure DHCP options:

1. Log in to the AWS Management Console.
2. Select **VPC** from the **Services** navigation list box.
3. In the left pane, click **Your VPCs**. A list of currently configured **VPCs** appears.
4. Select the **VPC** you are using and note the **DHCP options set ID**.
5. In the left pane, click **DHCP Option Sets**. A list of currently configured DHCP Option Sets appears.
6. Select the option set used by the VPC.
7. Check for an entry similar to the following and make sure the domain-name is specified. For example:

```
domain-name = ec2.internal
domain-name-servers = AmazonProvidedDNS
```

> **Note:** If you're using AmazonProvidedDNS in `us-east-1`, specify `ec2.internal`. If you're using AmazonProvidedDNS in another region, specify *region*.compute.internal (for example, `ap-northeast-1.compute.internal`).

8. If it is not configured correctly, create a new DHCP option set for the specified region and assign it to the VPC. For information on how to specify the correct domain name, see the AWS Documentation.

Server Does Not Start

Symptom

The Cloudera Director server does not start or quickly exits with an Out of Memory exception.

Cause

The Cloudera Director server is running on a machine with insufficient memory.

Solution

Run Cloudera Director on an instance that has at least 1GB of free memory. See Resource Requirements on page 20 for more details on Cloudera Director hardware requirements.

Problem When Removing Hosts from a Cluster

Symptom

A **Modify Cluster** operation fails to complete.

Cause

You are trying to shrink the cluster below the HDFS replication factor. See Removing Instances from a Cluster on page 55 (Note paragraph) for more information about replication factors.

Solution

Do not attempt to shrink a cluster below the HDFS replication factor. Doing so can result in a loss of data.

Problems Connecting to Cloudera Director Server

Symptom

You are unable to connect to the Cloudera Director server.

Cause

Configuration of security group and iptables settings. For more information about configuring security groups, see Setting up the AWS Environment on page 22. For commands to turn off iptables, see either Installing Cloudera Director Server and Client on the EC2 Instance on page 25 or Installing Cloudera Director Server and Client on Google Compute Engine on page 37. Some operating systems have IP tables turned on by default, and they must be turned off.

Solution

Check security group and iptables settings and reconfigure if necessary.

Frequently Asked Questions

This page answers frequently asked questions about Cloudera Director.

General Questions

How can I reduce the time required for cluster deployment?

You can reduce cluster deployment time by using an Amazon Machine Image (AMI). For information on creating an AMI, see Creating a Cloudera Manager and CDH AMI on page 59.

How can I make Cloudera Director highly available?

Cloudera Director can set up highly available clusters in a Cloudera Manager deployment, but does not support a high availability setup for itself. You can make Cloudera Director more robust by configuring it to use a backed-up, robust MySQL database server (one that is hosted, for example, on AWS RDS) for its database instead of Cloudera Director's default H2 database. Then, if the Director instance goes down, another instance can be spun up that references the same database. In this case, Cloudera Director has the ability to resume interrupted work.

For information on setting up highly available clusters in a Cloudera Manager deployment using Cloudera Director, see Creating Highly Available Clusters With Cloudera Director on page 98.

How can I find a list of available AMIs?

Perform the following steps to generate a list of RHEL 64-bit images:

1. Install the AWS CLI.

```
$ sudo pip install awscli
```

2. Configure the AWS CLI.

```
$ aws configure
```

   Follow the prompts. Choose any output format. The following example command defines *table* as the format.

3. Run the following query:

```
aws ec2 describe-images \
  --output table \
  --query 'Images[*].[VirtualizationType,Name,ImageId]' \
  --owners 309956199498 \
  --filters \
    Name=root-device-type,Values=ebs \
    Name=image-type,Values=machine \
    Name=is-public,Values=true \
    Name=hypervisor,Values=xen \
    Name=architecture,Values=x86_64
```

   AWS returns a table of available images in the region you configured.

Cloudera Director Glossary

availability zone

A distinct location in the region that is insulated from failures in other availability zones. For a list of regions and availability zones, see Regions and Availability Zones in the AWS documentation.

Cloudera Director

An application for deploying and managing CDH clusters using configuration template files.

Cloudera Manager

An end-to-end management application for CDH clusters. Cloudera Manager enables administrators to easily and effectively provision, monitor, and manage Hadoop clusters and CDH installations.

cluster

A set of computers that contains an HDFS file system and other CDH components.

cluster launcher

An instance that launches a cluster using Cloudera Director and the configuration file.

configuration file

A template file used by Cloudera Director that you modify to launch a CDH cluster.

deployment

See cluster. Additionally, deployment refers to the process of launching a cluster.

environment

The region, account credentials, and other information used to deploy clusters in a cloud infrastructure provider.

ephemeral cluster

A short lived cluster that launches, processes a set of data, and terminates. Ephemeral clusters are ideal for periodic jobs.

instance

One virtual server running in a cloud environment, such as AWS.

instance group

A specification that includes general instance settings (such as the instance type and role settings), which you can use to launch instances without specifying settings for each individual instance.

instance type

A specification that defines the memory, CPU, storage capacity, and hourly cost for an instance.

keys

The combination of your AWS access key ID and secret access key used to sign AWS requests.

long-lived cluster

A cluster that remains running and available.

provider

A company that offers a cloud infrastructure which includes computing, storage, and platform services. Providers include AWS, Rackspace, and HP Public Cloud.

region

A distinct geographical AWS data center location. Each region contains at least two availability zones. For a list of regions and availability zones, see http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.html.

tags

Metadata (name/value pairs) that you can define and assign to instances. Tags make is easier to find instances using environment management tools. For example, AWS provides the AWS Management Console.

template

A template file that contains settings that you use to launch clusters.