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

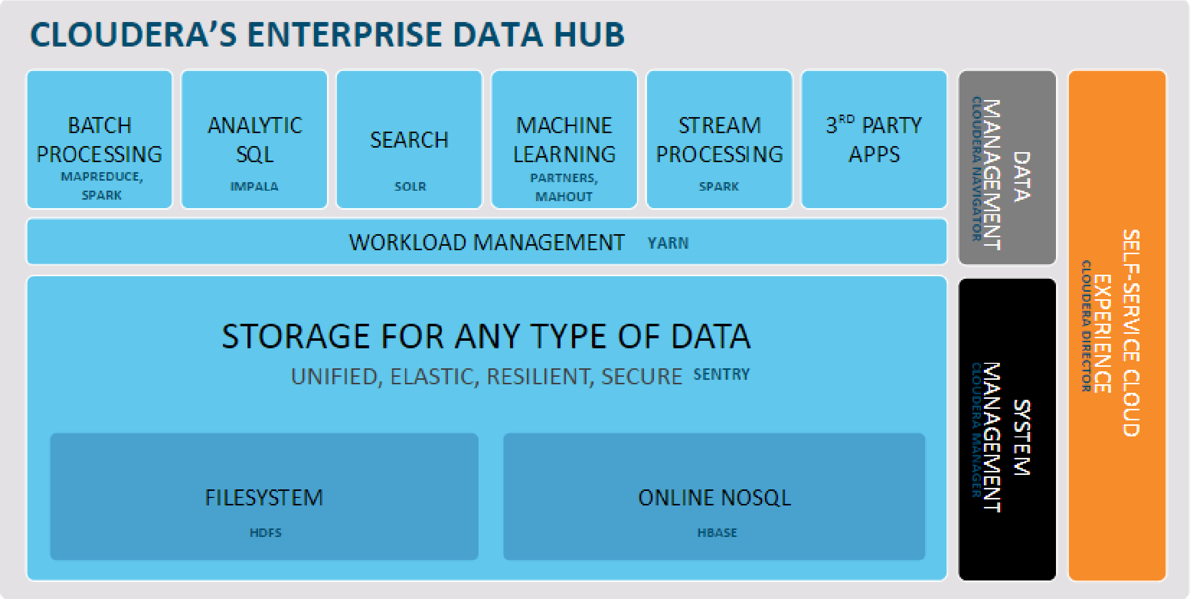
Version: Cloudera Director 1.1.x
Date: June 21, 2016

Table of Contents

Introduction

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

It is designed to provide a single-pane-of-glass administration experience for central IT to reduce costs and deliver agility, and for end-users to self-service provision and elastically 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.

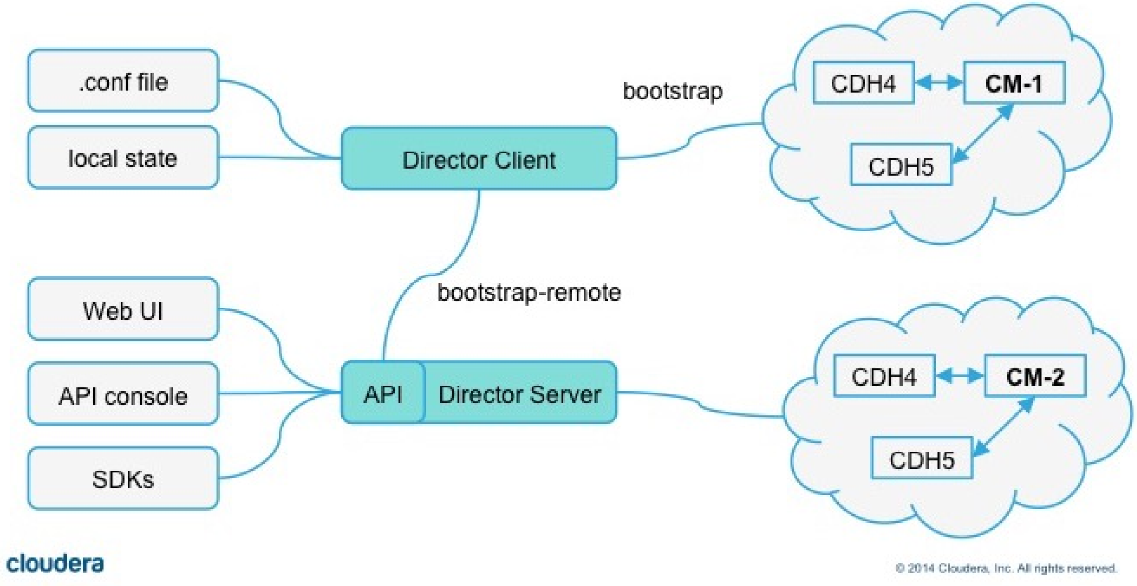
Benefit	Features
Simplified cluster life cycle management	Simple user interface: • Self-Service spin up and tear down • Dynamic scaling for spiky workloads • Simple cloning of clusters • Cloud blueprints for repeatable deployments

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.

- The lines that extend out from the client show that the client stores its state locally, and uses the `.conf` file to launch clusters, either through the server, using the `bootstrap-remote` command, or directly, using the `bootstrap` command.

- The lines that extend out from the server show that there are 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 a piece of the 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.
- **Cloudera Director client** - The client consumes a configuration file that describes the topology of a cluster, and can either create a cluster in standalone mode (without a server) using the `bootstrap` command, or dispatch the configuration file to a server through `bootstrap-remote`.
  - In standalone mode, the Cloudera Director client works well for proof of concept demonstrations, development work, and infrequent usage.
  - When used to dispatch the configuration file to a server through `bootstrap-remote`, the Cloudera Director client gives you full access to all of Cloudera Director's most advanced features, some of which can only be set up using the configuration file.
- **Cloudera Director server** - The server is designed for a more centralized environment than the client, managing multiple Cloudera Manager instances and CDH clusters, with multiple users and user accounts. You can either 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. The server works well for launching and managing large numbers of clusters in a production environment.
- **Cloudera Director client and server together** - To make use of advanced server features, you configure those features in the configuration file and then launch the server using the client, with the `bootstrap-remote` command.

The following summarizes key differences between the Cloudera Director client and server:

- cloudera-director-client
  - provides the ‘cloudera-director’ command
  - consumes a configuration file and can create/grow/terminate a cluster
  - stores state locally in the same folder with the config file
  - can be used to dispatch a config file to a server to create a cluster, giving access to advanced cloudera-director-server features
- cloudera-director-server
  - service cloudera-director-server [start | stop | status] (use the init script installed from the package to register the service with the operating system)
  - API for cluster management
  - has an embedded web interface and an API console
  - runs on port 7189 by default (can do https)
  - uses a database for state

Displaying Cloudera Director Documentation

To display Cloudera Director documentation for any page in the Director Server UI, click the question mark icon in the upper-right corner.

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 the help files embedded in the product are not updated after installation. 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 the new features and known issues and limitations for Cloudera Director 1.

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

New Features and Changes in Cloudera Director

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 1.1.3

- A number of issues have been fixed. See Issues Fixed in Cloudera Director 1.1.3 on page 11 for details.
- Cloudera Director's 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 1

The following sections describe the current known issues in Cloudera Director 1.

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.

Workaround: Use a parcel repository that is specified down to the maintenance version. This will ensure that Cloudera Director finds the correct version during cluster growth.

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/`

AWS credential environment variables for Director server

If the shell environment variables AWS_ACCESS_KEY and AWS_SECRET_KEY are set in the shell within which the Cloudera Director server component runs, then the server may use those credentials in communications with AWS when a Cloudera Director environment does not have them configured properly. When those credentials differ from the intended ones, EC2 instances may be allocated under unexpected accounts. Cloudera recommends ensuring that the environment variables are not set.

Severity: Medium

Workaround: Ensure that the Cloudera Director environment, defined in a client configuration file or in a server, has values for the keys that are present and correct.

Root partition resize fails on CentOS 6.5 (HVM)

Director is unable to resize root partition on Centos 6.5 HVM AMIs. This is due to a bug in the AMI. For more information, see the CentOS Bug Tracker.

Workaround: None.

Flume doesn't start automatically after FirstRun

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

Workaround: Start Flume manually using Cloudera Manager after Cloudera Director bootstraps the cluster.

Database support for Oozie, Hue, and Sqoop2 is incomplete

Cloudera Director cannot setup external databases for Oozie, Hue, and Sqoop2, the way it does with Hive.

Workaround: Set up the databases for these services as described in Cloudera Manager and Managed Service Databases. Provide the database properties such as `host address` and `username` to Cloudera Director in the relevant Oozie service configuration section.

Impala daemons attempt to connect over IPv6

Impala daemons attempt to connect over IPv6.

Workaround: Add the following command part of the instance bootstrap script: `sysctl -w net.ipv6.conf.all.disable_ipv6=1`.

DNS queries occasionally timeout with AWS VPN

DNS queries occasionally timeout with AWS VPN.

Workaround: Cloudera recommends that you install NSCD (name service cache daemon) on all cluster nodes via a bootstrap script. By default Linux does not cache DNS lookups. For more information, see the Linux NSCD man page.

When using RDS and MySQL, Hive metastore canary fails in Cloudera Manager

If you are including 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 due to a MySQL bug that is exposed through using MySQL 5.6.5 or later with the MySQL JDBC driver (used by Cloudera Director) version 5.1.19 or earlier. For information on the MySQL bug, see the MySQL bug description.

Workaround: Select an older MySQL version that avoids this bug, depending on the driver version installed by Cloudera Director from your platform's software repositories.

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

Terminating a cluster that is 'Bootstrapping' stops on-going processes, but keeps the cluster in 'Bootstrapping' phase.

Severity: Low

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

Fixed Issues

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

Issues Fixed in Cloudera Director 1.1.3

Ensure accurate time on startup

Instance normalization has been improved to ensure time is synchronized by Network Time Protocol (NTP) prior to 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 pre-installed Cloudera Manager packages

Cloudera Director no longer upgrades pre-installed 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 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 first-class 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 host.

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.

Supported Operating Systems for Cloudera Director

For installation of Cloudera Director server and client on a host computer or EC2 instance, Cloudera provides packages for the following Linux distributions:

- **RHEL-compatible**
  - Red Hat Enterprise Linux and CentOS
    - 5.7, 64-bit
    - 6.4, 64-bit
    - 6.5, 64-bit
- **SLES** - SUSE Linux Enterprise Server 11, 64-bit
- **Debian** - Wheezy (7.0 and 7.1), 64-bit
- **Ubuntu** - Precise (12.04), Trusty (14.04), 64-bit

Supported Operating Systems for Cloudera Manager and CDH Clusters

For Cloudera Manager and CDH clusters, Cloudera Director can instantiate EC2 instances based on AMIs of the following Linux distributions:

- RHEL6.4
- RHEL6.5
- Centos6.4
- Centos6.5

Use the AWS CLI or the AWS web console to discover available AMIs. See the AWS documentation on Amazon Machine Images for more information.

Supported Browsers

Cloudera Director supports the following browsers:

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

Supported Databases

Cloudera Director can use the following databases:

- H2 embedded database stored on the filesystem (default)
- MySQL - 5.5 and 5.6

Note: Cloudera Director uses the H2 database to store environment and cluster data, which it stores in the `state.h2.db` file. Back up this file to avoid losing this data.

Supported Cloudera Manager and CDH Versions

Cloudera Director can install any version of Cloudera Manager 5 with CDH 4 or CDH 5 parcels. Use of CDH packages is not supported.

Resource Requirements

Cloudera Director requires the following hardware resources, either on a host computer or an EC2 instance:

- **Disk Space** - At least 1 GB. More may be required depending on log retention settings.
- **RAM** - At least 1 GB free memory.
- If you are deploying Cloudera Director on an EC2 instance, Cloudera recommends an instance type of at least c3.large or c4.large.

Software Requirement

Cloudera Director requires Oracle Java SE Development Kit (JDK) version 8, 7, or 6. For download and installation information, see Java SE Downloads.

Networking and Security Requirements

The hosts in a Cloudera Manager deployment must meet the following networking and security requirements:

- The Cloudera Director API and web interface listen on port 7189 by default.
- Cloudera Director must make outbound connections over SSH to Cloudera Manager on port 7180 and to AWS API endpoints over HTTPS.
- Cloudera Director requires a DNS setup that allows forward and reverse name resolution.

Getting Started

Before you can install and use Cloudera Director, you have to 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 details steps for each of these tasks.

Setting up the AWS Environment

Whether you are using the Cloudera Director client or server, you must first set up the environment.

> **Note:** For new AWS users, a quick way to get started with Cloudera Director is by using the AWS Quickstart template.

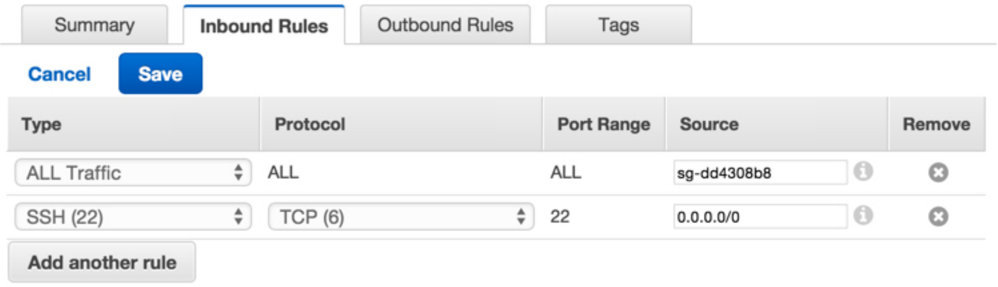
Setting Up VPC

Cloudera Director requires Amazon Virtual Private Cloud (Amazon VPC) to implement its virtual environment. You cannot use EC2-Classic. For more information about the differences between EC2-VPC and EC2-Classic, go to Amazon EC2 and Amazon Virtual Private Cloud.

> **Note:** The AWS VPC must be set up for forward and reverse hostname resolution.

To set up a VPC, follow these steps:

1. Log in to web console at https://aws.amazon.com/console.
2. In the upper right of the AWS Console, select a region.
3. Select **VPC** from the **Services** navigation list box.
4. Click **Start VPC Wizard**.
5. On the Select a VPC Configuration page, specify IP address settings, a VPC name, and any other preferences. The easiest way to get started is to select **VPC with a Single Public Subnet**. For more information, see the VPC documentation.
6. Click **Create VPC**.
7. In the left pane, click **Subnets**.
8. Click **Create Subnet**.
9. Configure a subnet of the VPC you created and click **Yes, Create**.
10. In the left pane, click **Security Groups**.
11. Click **Create Security Group**.
12. Enter a name and description. Make sure to select the VPC you created from the VPC list box.
13. Click **Yes, Create**.
14. Select the newly created security group and add the following rules:
    a. Add the `All traffic, all protocols, all ports`, and *`id of this security group`* inbound rules. Then add the `SSH(22), TCP(6), 22, 0.0.0.0/0` inbound rules. You can secure this further later.



Type	Protocol	Port Range	Source	Remove
ALL Traffic	ALL	ALL	sg-dd4308b8	⊗
SSH (22)	TCP (6)	22	0.0.0.0/0	⊗

b. Add the `All traffic, all protocols, all ports, 0.0.0.0/0` outbound rule.

For more information about security group rules, see the AWS documentation: Security Groups for Your VPC.

Creating a Key Pair

To interact with the cluster launcher and other instances, you must create an SSH key pair.

> **Note:** For information on importing an existing key pair, see the EC2 Documentation.

If you do not have a key pair, follow these steps:

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 keyfile (with .pem extension) to a secure location and note the location.

Creating AWS Identity and Access Management (IAM) Policies

In AWS, you use IAM files to create policies that control access to resources in a VPC. 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 create RDS database servers for persistence on demand, Cloudera Director requires permissions for the following methods:
  - createDBInstance
  - deleteDBInstance
  - describeDBInstances

The following example IAM policy shows the format to use with Cloudera Director. Your Amazon Resource Name (ARN) will be different.

```
{
  "Statement": [
    {
      "Sid": "Stmt1423159499758",
      "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"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:ec2:<region>:<account>:<resourceType>/<resourcePath>"
    },
    {
      "Sid": "Stmt1423159567748",
      "Action": [
        "rds:CreateDBInstance",
        "rds:DeleteDBInstance",
        "rds:DescribeDBInstances"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

Starting an Instance

Cloudera Director requires a dedicated instance in the same subnet that can access new instances on the private network.

To start the instance:

> **Note:** For general information about copying files to and from an instance, see the EC2 Documentation.

1. Make sure you are logged in to web console at http://aws.amazon.com/console/.
2. Select **EC2** from the **Services** navigation list box.
3. Click **Launch Instance**.
4. Get the AMI ID for the instance. Cloudera recommends Red Hat Linux 6.4 (ami-b8a63b88 in US-West-2) with a c3.xlarge instance type. If the AMI does not show up in the list, go to https://aws.amazon.com/partners/redhat/ and select a 64-bit version of Red Hat Linux 6.4 for the region in which you are launching the cluster.
5. Click **Community AMIs** in the left pane.
6. Enter the **AMI ID**. The AMI appears in the list.
7. Click **Select**. The Choose an Instance Type page appears.
8. Click **Next: Configure Instance Details**.

   a. Make sure to select the VPC and subnet that you created or noted earlier.
   b. The cluster launcher requires Internet access; from the Auto-assign Public IP list box, select **Enable**.

9. Click **Next: Add Storage**.

10. Click **Next: Tag Instance** and create any tags to quickly find the instance.
11. Click **Next: Configure Security Group**. Then, select the **Select an existing security group** radio button and select the security group you noted or created earlier.
12. Click **Review and Launch**.
13. Click **Launch**. When prompted, launch the instance with the key pair that you created. If you selected SSD storage, you might be prompted to choose the storage type. The instance does not require SSD storage.
14. Click **Launch Instances**.
15. After the instance is created, note its public and private IP addresses.

Connecting to Your Cluster Using a SOCKS Proxy

For security purposes, we recommend that you connect to your cluster using a SOCKS proxy. This topic shows you how.

Create a Proxy Auto-Config File

To create proxy auto-config (PAC) file, perform the following tasks.

- 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 bellow 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';
}
```

- Save the file.

The PAC file contains the three rules needed for Cloudera Director.

Set Up SwitchySharp

1. Open Chrome and go to Chrome Apps
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:
    - In the **Profile Name** field, enter `AWS-Cloudera`.
    - Click **Automatic Configuration**.
    - Click **Import PAC File** and import your PAC file.
    - Click **Save**.
4. Click the **General** tab and do the following:
    - Click **Quick Switch**.
    - Drag **[Direct Connection]** and **AWS-Cloudera** to the **Cycled Profiles** area.
    - Set **Startup Profile** to **[Direct Connection]**.
    - Click **Save**.

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 <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.

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, perform the following tasks.

> **Important:** Cloudera Director requires a JDK. For more information, see Software Requirement on page 14

1. Download the Cloudera Director by running the correct command for your distribution.

    - For RHEL 6, CentOS 6, and Oracle 6:

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

    - For RHEL 5, CentOS 5, and Oracle 5:

```
wget http://archive.cloudera.com/director/redhat/5/x86_64/director/cloudera-director.repo
 -O /etc/yum.repos.d/cloudera-director.repo
```

    - For Debian:

```
wget
http://archive.cloudera.com/director/debian/wheezy/amd64/director/cloudera-director.list
 -O /etc/apt/sources.list.d/cloudera-director.list
```

    - For SLES:

```
zypper addrepo -f
http://archive.cloudera.com/director/sles/11/x86_64/director/cloudera-director.repo
```

    - For Ubuntu 12.04 (Precise Pangolin):

```
wget
http://archive.cloudera.com/director/ubuntu/precise/amd64/director/cloudera-director.list
 -O /etc/apt/sources.list.d/cloudera-director.list
```

    - For Ubuntu 14.04 (Trusty Tahr):

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

2. Add the signing key.

    - For RHEL 6, CentOS 6, and Oracle 6 this step is not required. Continue to the next step.
    - For RHEL 5, CentOS 5, and Oracle 5 this step is not required. Continue to the next step.

- For Debian, run the following command:

```
curl -s http://archive.cloudera.com/director/debian/wheezy/amd64/director/archive.key
| sudo apt-key add -
```

- For SLES this step is not required. Continue to the next step.
- For Ubuntu 12.04 (Precise Pangolin), run the following command:

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

- 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 -
```

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

   - For RHEL 6, CentOS 6, and Oracle 6:

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

   - For RHEL 5, CentOS 5, and Oracle 5:

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

   - For Debian:

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

   - For SLES:

```
zypper install cloudera-director-client
```

   - For Ubuntu 12.04 (Precise Pangolin):

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

   - For Ubuntu 14.04 (Trusty Tahr):

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

You are now ready to configure the Cloudera Director Client.

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). An AMI is the basis 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.

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.

Provisioning a Cluster

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

To modify the configuration file:

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

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

2. Open `aws.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 nodes 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.

The Cloudera Director Configuration File

This section describes the configuration file used when launching a cluster through Cloudera Director client with the `bootstrap` command, or through the Cloudera Director server with the `bootstrap-remote` command. The configuration file also includes additional advanced settings that are documented in comments within the file.

File Location

To create a configuration file, install the `cloudera-director-client` package, open `aws.simple.conf` or `aws.reference.conf`, and save it as `aws.conf`. The 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.

Environment Settings

This section describes basic settings you must configure before deploying a cluster.

Setting	Type	Required	Description
name	string	yes	Specifies the name of the cluster in Cloudera Manager. Example: C5-Reference-AWS Default: none
provider	container	yes	Container for the cloud infrastructure provider.
id	string	yes	The ID of the cloud infrastructure provider; leave this set to aws. Example: aws Valid Values: aws Default: aws
accessKeyId	string	yes	The access key used to make AWS requests. Make sure the value is enclosed in double quotes. Example: RQU1JC3XKTTYYJTXDR Valid Values: Valid AWS access key. Default: none
secretAccessKey	string	yes	The secret access key used to make AWS requests. Make sure the value is enclosed in double quotes.

Setting	Type	Required	Description
			Example: vWdg/y4ArVuxdlzZoO06139xTSd5V5S8 Valid Values: Valid AWS secret key. Default: none
publishAccessKeys	boolean	no	Specifies whether Cloudera Director automatically publishes your credentials as cluster configurations for Amazon S3 access. Example: true Valid Values: true | false Default: false
region	string	yes	The region in which to launch the cluster. Example: us-west-2 Valid Values: See Availability Zones . Default: none
regionEndpoint	string	no	Specifies the region endpoint for clusters launched in the .gov region. If you are not launching in the .gov region, leave this commented out. Example: ec2.us-gov-west-1.amazonaws.com Valid Values: any valid .gov region. Default: none
keyName	string	yes	The name of the key pair used to start the cluster launcher. Example: my-cloudera-keypair Valid Values: any valid key pair associated with the region. Default: none
subnetId	string	yes	ID of the subnet that you noted earlier. Example: subnet-5b818f1d

Setting	Type	Required	Description
			Valid Values: any valid subnet ID in the region. Default: none
securityGroupsIds	string	yes	ID of the security group that you noted earlier. Use the ID of the group, not the name (for example, sg-b139d3d3, not default). To specify more than one security group, separate them with commas and enclose the string with quotes. Example: sg-b139d3d3 Valid Values: any valid security group ID in the region. Default: none
instanceNamePrefix	string	yes	The prefix used to launch instances. This prefix is part of the instance name which you can use to find instances started by Cloudera Director in the AWS Console. Example: skynet-cluster-1 Valid Values: any string Default: none
rootVolumeSizeGB	integer	yes	Sets the size of the root volume for the cluster launcher. Example: 100 Default: 50
associatePublicIpAddresses	boolean	no	Specifies whether nodes will have public IP addresses. To optimize Amazon S3 data transfer performance, set this to true. Example: true Valid Values: true | false Default: false
image	string	yes	Specifies the AMI to use. Cloudera recommends Red Hat Enterprise Linux 6.4 (64bit). To find the correct

Setting	Type	Required	Description
			AMI for the selected region, visit the Red Hat AWS Partner Page. Example: ami-22558833 Valid Values: Any valid AMI running Enterprise Enterprise Linux 6.4 (64bit) Default: none Note: For more information about AMI selection, see Choosing an AMI on page 21.
ssh	container	yes	Container for SSH settings.
username	string	yes	Specifies the username for SSH access to the instances. Example: ec2-user Default: none
privateKey	string	yes	Specifies the location of the SSH private key. Example: ${?HOME}/.ssh/director_id_rsa Default: none

Instance Settings

This section describes settings that define instances. Once defined, you can launch these instance types for Cloudera Manager and nodes in the cluster.

Setting	Type	Required	Description
instances	container	yes	The container that specifies instance settings.
instance_type	container	yes	A container that specifies settings for a type of instance to launch. You can specify any string value. For example, you can create an instance type called “cm” that uses an m1.large instance and another instance type called “node” that uses an m1.xlarge instance.
type	string	yes	The type of instances to launch. Example: m3.2xlarge

Setting	Type	Required	Description
			Valid Values: Any valid instance name. For a list of valid instance types, go to Instance Types. Default: none
bootstrapScript	string	yes	Linux shell script that executes whenever a cluster instance reboots. After the instance boots, this script automatically runs. This script can contain anything you need for your environment including libraries, monitoring tools, security configurations, and so on. Example: """#!/bin/sh # This is an embedded bootstrap script that runs # as root and can be used to customize # the instances immediately after boot and before # any other Cloudera Director action # If the exit code is not zero Cloudera Director will # automatically retry echo 'Hello World!' exit 0 """ Valid Values: any valid script Default: none
tags	container	yes	Container for any tags to apply to the instances. These tags can be used to find your instances in the AWS Console or on your AWS invoice.
tag	string	yes	Specifies the name and value of the tag. Example: department: “Data Science”

Setting	Type	Required	Description
			Valid Values: Any valid name/value pair. Default: none

Cloudera Manager Settings

This section describes settings for the Cloudera Manager instance.

Setting	Type	Required	Description
cloudera-manager	container	yes	The container for Cloudera Manager settings.
instance	string	yes	Specifies the instance type to use that you defined in Instance Settings. Example: ${instances.cm} Valid Values: any instance type that you defined earlier. Default: none
tags	container	yes	Container for any tags to apply to the Cloudera Manager instance.
tag	string	yes	Specifies the name and value of the tag. Example: application: “Cloudera Manager 5” Valid Values: Any valid name/value pair. Default: none
customBannerText	string	no	Specifies custom banner text to display in Cloudera Manager. Example: “Managed by Cloudera Director” Valid Values: any valid string Default: none
enableEnterpriseTrial	boolean	no	When set to true, automatically enables a 60-day Cloudera Enterprise trial. Example: true Valid Values: true | false Default: false

Database Settings

This section describes settings for configuring external databases. This section is optional. If no settings are specified, Cloudera Director uses the embedded PostgreSQL database.

Setting	Type	Required	Description
databases	container	no	The container for databases.
CLOUDERA_MANAGER	container	no	The container for the database used by Cloudera Manager.
ACTIVITYMONITOR	container	no	The container for the database used by the activity monitor.
REPORTSMANAGER	container	no	The container for the database used by the reports manager.
NAVIGATOR	container	no	The container for the database used by Navigator.
type	string	no	The type of database. Example: postgresql Valid Values: postgresql | mysql. Default: none Note: Cloudera currently provides PostgreSQL drivers. Drivers for other databases must be added with the bootstrap script.
host	string	no	The database host. Example: db.example.com Default: none
port	string	no	The database port. Example: 123 Default: none
user	string	no	A database user. Example: dbuser Default: none
password	string	no	The password of the database user. Example: Pa$$word Default: none
name	string	no	The name of database.

Setting	Type	Required	Description
			Example: cmdb Default: none

Cluster Settings

This section describes products and services to launch on instances in the cluster.

Setting	Type	Required	Description
cluster	container	yes	The container for the cluster.
products	container	yes	The container for products to launch.
CDH	string	no	The version of CDH to launch. Example: 5 Valid Values: 4 | 5 Default: 4
IMPALA	string	yes	The version of Impala to launch. Example: 1.2 Default: none
services	array	yes	An array of services to launch. Options include: Example: [HDFS, YARN, ZOOKEEPER, HBASE, HIVE, HUE, OOZIE] Valid Values: HBASE, HDFS, HIVE, HUE, IMPALA, KS_INDEXER, MAPREDUCE, OOZIE, SOLR, SPARK, SQOOP, YARN, and ZOOKEEPER. Default: none
HIVE	container	no	The container for the database used by Hive. All Hive database settings are commented out by default.
type	string	no	The type of Hive database. Example: postgresql Valid Values: postgresql | mysql. Default: none

Setting	Type	Required	Description
host	string	no	The Hive database host. Example: db.example.com Default: none
port	string	no	The Hive database port. Example: 123 Default: none
user	string	no	A database user for Hive. Example: dbuser Default: none
password	string	no	The password of the database user. Example: Pa$$word Default: none
name	string	no	The name of Hive database. Example: cmdb Default: none
masters	container	yes	The container for service masters.
count	integer	yes	The number of instances to launch.
instance	string	yes	Specifies the instance type to use that you defined in Instance Settings. Example: ${instances.nodes} Valid Values: any instance type that you defined earlier. Default: none
tags	container	yes	Container for any tags to apply to the instances.
tag	string	yes	Specifies the name and value of the tag. Example: group: master Valid Values: Any valid name/value pair. Default: none
roles	container	yes	Container for roles.

Setting	Type	Required	Description

role	string	yes	Specifies the roles to apply to the masters.
			Example:
			HDFS: ${roles.HDFS_MASTERS}
			YARN: ${roles.YARN_MASTERS}
			ZOOKEEPER: ${roles.ZOOKEEPER_MASTERS}
			HBASE: ${roles.HBASE_MASTERS}
			HIVE: ${roles.HIVE_MASTERS}
			HUE: ${roles.HUE_MASTERS}
			OOZIE: ${roles.OOZIE_MASTERS}
			Default: none

workers	container	yes	Container for workers to launch.
count	integer	yes	The number of instances to launch.
instance	string	yes	Specifies the instance type that you defined in Instance Settings. Example: ${instances.nodes} Valid Values: any instance type that you defined earlier. Default: none
tags	container	yes	Container for any tags to apply to the instances.
tag	string	yes	Specifies the name and value of the tag. Example: group: master Valid Values: Any valid name/value pair. Default: none
roles	container	yes	Container for roles.
role	string	yes	Specifies the roles to apply to the masters.

Setting	Type	Required	Description

			Example:
			HDFS: ${roles.HDFS_MASTERS}
			YARN: ${roles.YARN_MASTERS}
			HBASE: ${roles.HBASE_MASTERS}
			Default: none

placementGroup	string	yes	Specifies the placement group in which to launch the instance. For more information, see Placement Groups.
gateways	container	yes	Container for gateways to launch. Note: Although this container is called gateways, containers at this level can use any name to launch a set of instances with shared instance settings and roles.
count	integer	yes	The number of instances to launch.
instance	string	yes	Specifies the instance type that you defined in Instance Settings. Example: ${instances.nodes} Valid Values: any instance type that you defined earlier. Default: none
tags	container	yes	Container for any tags to apply to the instances.
tag	string	yes	Specifies the name and value of the tag. Example: group: master Valid Values: Any valid name/value pair. Default: none
roles	container	yes	Container for roles.
role	string	yes	Specifies the roles to apply to the masters.

Setting	Type	Required	Description
			Example: HIVE: ${roles.HIVE_MASTERS} Default: none

Note: Although you can deploy Flume through Cloudera Director, you must start it manually using Cloudera Manager.

Running Cloudera Director Client

After you modify the configuration file, you can run Cloudera Director client. The client can run in standalone mode, or, if you already have a server, you can run the client against the server using the commands `bootstrap-remote` and `terminate-remote`. For more information on using the client to deploy clusters on the server, see Submitting a Cluster Configuration File on page 45.

Note: If you are restarting Cloudera Director client, it prompts you to resume from where it 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-1.1.0]$ cloudera-director bootstrap aws.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 CDH4 cluster using the new nodes ...... done
* Downloading CDH-4.6.0-1.cdh4.6.0.p0.26 parcel ..... done
* Distributing CDH-4.6.0-1.cdh4.6.0.p0.26 parcel ... done
* Activating CDH-4.6.0-1.cdh4.6.0.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.

Connecting to Cloudera Manager

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 aws.conf
```

Cloudera Director displays output similar to the following:

```
Cloudera Launchpad 1.0.0 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 <your-cert.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 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

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

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

1. Open the `aws.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 aws.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.

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 installation, configuration, and use are described in the following topics.

Installing Cloudera Director Server

We recommend that you install Cloudera Director Server within the subnet where you will create clusters. To install Cloudera Director Server, perform the following tasks.

> **Important:** Cloudera Director requires a JDK. For more information, see Software Requirement on page 14.

1. Download the Cloudera Director by running the correct command for your distribution.

    - For RHEL 6, CentOS 6, and Oracle 6:

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

    - For RHEL 5, CentOS 5, and Oracle 5:

```
wget http://archive.cloudera.com/director/redhat/5/x86_64/director/cloudera-director.repo
 -O /etc/yum.repos.d/cloudera-director.repo
```

    - For Debian:

```
wget
http://archive.cloudera.com/director/debian/wheezy/amd64/director/cloudera-director.list
 -O /etc/apt/sources.list.d/cloudera-director.list
```

    - For SLES:

```
zypper addrepo -f
http://archive.cloudera.com/director/sles/11/x86_64/director/cloudera-director.repo
```

    - For Ubuntu 12.04 (Precise Pangolin):

```
wget
http://archive.cloudera.com/director/ubuntu/precise/amd64/director/cloudera-director.list
 -O /etc/apt/sources.list.d/cloudera-director.list
```

    - For Ubuntu 14.04 (Trusty Tahr):

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

2. Add the signing key.

    - For RHEL 6, CentOS 6, and Oracle 6 this step is not required. Continue to the next step.
    - For RHEL 5, CentOS 5, and Oracle 5 this step is not required. Continue to the next step.

- For Debian, run the following command:

```
curl -s http://archive.cloudera.com/director/debian/wheezy/amd64/director/archive.key
| sudo apt-key add -
```

- For SLES this step is not required. Continue to the next step.
- For Ubuntu 12.04 (Precise Pangolin), run the following command:

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

- 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 -
```

3. Install Cloudera Director Server by running the correct command for your distribution.
   - For RHEL 6, CentOS 6, and Oracle 6:

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

   - For RHEL 5, CentOS 5, and Oracle 5:

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

   - For Debian:

```
apt-get install cloudera-director-server
```

   - For SLES:

```
zypper install cloudera-director-server
```

   - For Ubuntu 12.04 (Precise Pangolin):

```
apt-get install cloudera-director-server
```

   - For Ubuntu 14.04 (Trusty Tahr):

```
apt-get install cloudera-director-server
```

4. Start the Cloudera Director Server by running the following command:

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

5. Turn off IP tables by running the following command:

```
service iptables stop
```

You are now ready to configure the Cloudera Director Server.

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 Clusters on page 51.

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

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 40 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 the InnoDB engine. Cloudera Director will not start if its tables are configured with the MyISAM engine. (Typically, tables revert to MyISAM if the InnoDB engine is misconfigured.) To check which engine your tables are using, run the following command from the MySQL shell:

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

- 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]
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
```

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`

OS	Command
	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 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-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`

OS	Command
	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, user names, 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 user name (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:
```

Configuring an Environment and Deploying a Cluster

The environment defines common settings used with AWS. While creating an environment, you are also prompted to deploy its first cluster.

To add an environment:

1. Open Cloudera Director through a web browser using the public IP address you noted in Starting an Instance on page 17. For example, http://100.100.100.100:7189.

   You are prompted to log in.

2. Enter the username and password (default: admin/admin).
3. Click **Add Environment**.
4. On the Add Environment page, enter a name in the **Environment Name** field.
5. Select a region from the **Region** field.
6. Enter your keys in the **Access key ID** and **Secret access key** fields.
7. To make the keys available to the cluster, select the **Make access keys available to Hadoop** check box.
8. Enter the name of the EC2 key pair in the **EC2 Public key name** field.
9. Enter the name of the SSH user in the **SSH username** field, and copy the SSH private key into the **SSH private key** field.
10. Enter the SSH passphrase in the **SSH passphrase** field. If the SSH key is not encrypted, leave this blank.
11. To assign public IP addresses to instances, select the **Associate public IP addresses with instances** check box, and then click **Continue**.
12. On the Add Cloudera Manager page, enter a name for the Cloudera Manager in the **Cloudera Manager name** field. If you want to enable a trial of Cloudera Enterprise, select the **Enable Cloudera Enterprise trial** check box.
13. Select whether to create a new template or use an existing one from the **Instance Template** list box. If you select **Create New Instance Template**, configure the following options:

    - **Instance Template name** - Enter a name for the template.
    - **Type** - select the instance type.
    - **Amazon Machine Image (AMI)** - enter the AMI ID to use for Cloudera Manager.
    - **Tags** - specify one or more tags to associate with the instance.
    - **Instance Name Prefix** - A prefix that Cloudera Director should use when naming the instances (not part of the hostname).
    - **Root volume size** - select the size of the root volume.
    - **VPC subnet ID** - enter the ID of the VPC subnet in which the instance will be located.
    - **Security group IDs** - enter one or more security group IDs with which the instance will be associated.
    - **Bootstrap script** (optional) - enter a Linux bootstrap script. After the instance boots, this script automatically runs. This script can contain anything you need for your environment including libraries, monitoring tools, security configurations, and so on.

14. Select whether to override the default Cloudera Manager repository, and click **Continue**. You are prompted for confirmation; click **OK**.
15. On the **Add Cluster** page, enter a name for the cluster in the **Cluster name** field, and select the version of CDH to deploy in the **Version** field.

16. Select the type of cluster to deploy from **Services**.
17. Select the numbers of masters, workers, and gateways to deploy. Then, select an instance template for each, or create new templates.

> **Note:** You must deploy at least one master and one worker. Cloudera recommends that you deploy at least three workers.

18. When you are finished, click **Continue**. You are prompted for confirmation; click **OK** to deploy the cluster.

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 configuration files that the server uses for cluster deployment. These configuration files provide advanced options not currently available in the server UI.

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.

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

1. Create a configuration file. See Provisioning a Cluster on page 22.
2. Install the latest version of the Cloudera Director client from the Cloudera Director Download Page.
3. Unzip the Cloudera Director client.
4. Change to the client directory and enter the following:

```
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 name or IP address of the host on which Cloudera Director server is running, and *port* is the port on which it is listening.

   The Cloudera Director client provides deployment status.

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. If you have not created an environment, see Configuring an Environment and Deploying a Cluster on page 44.
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, and select the version of CDH to deploy in the **Version** field.
6. Select the type of cluster to deploy from **Services**.
7. Select the numbers of masters, workers, and gateways to deploy. Then, select an instance template for each or create one or more new templates.
8. 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.

Adding Nodes to a Cluster

You can use Cloudera Director to increase the number of workers or gateways in a cluster:

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

   Cloudera Director opens with a list of clusters.

2. If the cluster has a status of **Ready**, click the **Actions** list box to the right of the target cluster and select **Modify Cluster**.

   The Modify Cluster page appears, displaying the number of gateways, workers, and masters.

3. On the Modify Cluster page, click **Edit** and increase the number of workers and gateways to the desired size.

   > **Note:** Cloudera recommends rebalancing the cluster through Cloudera Manager if you increase the number of data nodes by 30% or more.

Removing or Repairing Nodes in a Cluster

Removing Nodes from a Cluster

Cloudera Director can reduce the size of a cluster by removing nodes that contain worker or gateway roles.

To shrink a cluster:

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

   Cloudera Director opens with a list of clusters.

2. If the cluster has a status of **Ready**, click the **Actions** list box to the right of the cluster to shrink and select **Modify Cluster**.

   The Modify Cluster page appears, displaying the number of gateways, workers, and masters.

3. To remove all instances for a role type:

   - On the Modify Cluster page, click **Delete Group**.

   To remove individual instances (that is, individual nodes for a role instance group):

   - Click **Edit** next to the instance count for workers or gateways, select the nodes you want to remove, and click the **Delete** button above the list of instances. The instances you selected 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 editing the instance group 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.
> - Reducing or repairing the instance count to a running level that is below the replication factor can cause Cloudera Manager to fail to decommission nodes. In this case, Cloudera Director may stop functioning properly. If this happens, abort the decommission command in Cloudera Manager.
> - If the instance count is below the replication factor, reduce the replication factor before attempting a repair.
> - Cloudera recommends rebalancing the cluster through Cloudera Manager if you reduce the number of data nodes by 30% or more.

Repairing Nodes in a Cluster

To repair nodes in a cluster:

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

   Cloudera Director opens with a list of clusters.

2. If the cluster has a status of **Ready**, click the **Actions** list box to the right of the cluster to repair and select **Modify Cluster**.

   The Modify Cluster page appears, displaying the number of gateways, workers, and masters.

3. Click **Edit** next to the instance count for workers or gateways you want to repair, and select the nodes you want to repair.
4. Click the **Repair** button above the list of instances. The nodes 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 editing the instance group without making any changes.
6. Click **Continue** to confirm and repair the selected instances.

Terminating a Cluster

You can terminate a cluster at any time. To terminate a cluster:

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

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 grayed out 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 user 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

This section explains how to use some of the advanced features of Cloudera Director.

Using an External Database for Cloudera Manager and Clusters

By default, Cloudera Director configures Cloudera Manager and cluster services, such as Hive, to use the Cloudera Manager embedded PostgreSQL database. You can also configure Cloudera Director to use external database servers, instead. If you have a database server configured, you can use Cloudera Director to configure Cloudera Manager and cluster services to create or use databases on that server. You can also configure Cloudera Director to use a cloud provider, like 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 `aws.conf` file and launch Cloudera Director client (standalone) by issuing the `bootstrap` command.
- **Cloudera Director server** - Configure external databases with the API. You can also configure external databases by editing the `aws.conf` file and launching Cloudera Director server with the `bootstrap-remote` command.

These procedures are described in the topics in this section.

Defining Database Servers

Cloudera Director must have information about external database servers before it can use them. A single database server is scoped to an environment, so only deployments and clusters in that environment recognize it.

A database server template can refer to either an existing server or a database server to be created. 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, then 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.

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.	

Operation	Description	Notes
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.

Client Configuration File (databaseServers section)

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 RDS instances are supported.
- RDS works through both `bootstrap-remote` and standalone `bootstrap` on the client, as well as the server API. It is not supported through the UI.

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 later with the MySQL JDBC driver (used by Cloudera Director) version 5.1.19 or earlier. 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`

key	description	example
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 may also specify tags for the new instance.

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 via POST to the endpoint, and the template lacks a host and port, 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 via GET.

Likewise, when the definition is deleted via DELETE, 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, Director leaves RDS instances alone. This makes it possible for multiple deployments and clusters to share the same external database servers that Director creates on RDS.

Defining a Database Server 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
   }
}
```

Using 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 installations or CDH clusters.

The following parts of an existing databasemust 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.

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

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.

```
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 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 Cluster 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. You can run either client or server versions without specifying any of these properties. However, you might want to customize one or more properties, depending on your deployment.

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.

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.aws.client.configuration.connectionTimeoutInMilliseconds`	Wait time to establish AWS client connections, in milliseconds. Type: int Default: 10000
`lp.aws.client.configuration.maxErrorRetries`	Maximum number of times to retry failed AWS requests. Type: int Default: 7

Property	Description
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 nodes 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
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

Property	Description
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
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

Property	Description
	Default:true
`lp.debug.diagnosticDataDownloadDirectory`	Destination directory for downloaded Cloudera Manager diagnostic data. Type: string Default: /tmp
`lp.debug.dumpClouderaManagerLogsOnFailure`	Dump Cloudera Manager log entries into the Director logs on unrecoverable bootstrap failure?. Type: boolean Default: false
`lp.debug.dumpClusterLogsOnFailure`	Dump cluster service logs, standard output, or standard error into the Director logs on unrecoverable bootstrap failure?. Type: boolean Default: false
`lp.ec2.ephemeral.customMappingsPath`	File containing custom EC2 ephemeral device mappings. Type: file Default: `./etc/ec2.ephemeraldevicemappings.properties`
`lp.ec2.ephemeral.rangeStart`	First character suffix for ephemeral devices mapped to EC2 instances. Type: char Default: b
`lp.ec2.virtualization.customMappingsPath`	File containing custom EC2 virtualization mappings. Type: file `./etc/ec2.virtualizationmappings.properties`
`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`

Property	Description
`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
`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

Property	Description
`lp.proxy.http.workstation`	Originating workstation in NT domain for HTTP proxy authentication; none = no workstation. Type: string Default: none
`lp.rds.customEndpointsPath`	File containg custom RDS endpoints. Type: file Default: `./etc/rds.endpoints.properties`
`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 Default:
`lp.remote.username`	Remote Cloudera Director server username (client only). Type: string Default:
`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

Property	Description
	Default: 240
`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
`lp.update.parallelBatchSize`	Parallelism for allocating and setting up cluster nodes 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 hosts with Cloudera Manager slave roles to restart at a time. Type: int Default: 20
`lp.update.restartCluster.rollingRestartSlaveFailCountThreshold`	Threshold for number of slave host batches that are allowed to fail to restart before the entire command is considered failed (advanced use only). Type: int Default: 0

Property	Description
`lp.update.restartCluster.rollingRestartSleepSeconds`	Number of seconds to sleep between restarts of Cloudera Manager slave 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
`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
`port`	Director server port (server only). Type: int Default: 7189
`server.sessionTimeout`	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 not available through the Cloudera Director UI, but is available for both Cloudera Director client and Cloudera Director server:

- **Client** - Using the configuration file. For more information on the configuration file, see The Cloudera Director Configuration File.
- **Server** - Submitting JSON documents to the REST service endpoint. The REST service endpoint can also be exercised through the API console at `http://`*`director-server-hostname`*`:7189/api-console`.

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.

Note: You can add Llama with autoconfiguration only if you are using Cloudera Manager 5.2 or later. For Cloudera Manager 5.0 and 5.1, you must set Llama configurations manually, as described in the Cloudera Manager documentation.

Deployment Template Configuration

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

CLI

Using the CLI, 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 CLI and the API.

CLI

Using the CLI, 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 CLI and the API.

CLI

Using the CLI, 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"
            }
         }
      }
   }
}
```

Creating a Cloudera Manager AMI

In a larger production environment, you can optimize instance start times by creating an Amazon machine image (AMI). Create this AMI after you complete the initial installation of Cloudera Manager (for CentOS/RHEL, adding the appropriate repo file to the `/etc/yum.repos.d/` directory, `yum install cloudera-scm-server`).

1. Download the appropriate parcel file. For example: `http://archive.cloudera.com/cdh5/parcels/5.3.0/CDH-5.3.0-1.cdh5.3.0.p0.30-wheezy.parcel`
2. Copy the parcel file, depending on your Cloudera Director installation.
   - For Cloudera Director Server
     - Copy the parcel to the `/opt/cloudera/parcel-repo` directory. Create this directory if it does not exist.
     - Calculate the SHA hash for the AMI and place it in `/opt/cloudera/parcel-repo`. Name the file the same as the parcel, but add ".sha" to the filename.
   - For Cloudera Director Client:
     - Copy the parcel to the `/opt/cloudera/parcel-cache` directory. Create this directory if it does not exist.
3. Make sure that the `/opt/cloudera` directory and its sub-directories are owned by cloudera-scm. Add the cloudera-scm user if it does not already exist.

The following example script completes all the steps.

```
#!/usr/bin/env bash
#
# Copyright (c) 2014 Cloudera, Inc. All rights reserved.
#

# For this script to work properly, you need to supply a URL to a parcel file,
# e.g.
http://archive.cloudera.com/cdh5/parcels/5.3.0/CDH-5.3.0-1.cdh5.3.0.p0.30-wheezy.parcel

# You can do this one of two ways:
# 1. Set a PARCEL_URL environment variable.
# 2. Supply an argument that is a PARCEL_URL.

# This script will have to be re-run for each parcel you want to cache on the
# image that you are building.

if [ -z ${PARCEL_URL+set} ]
then
  if [ "$#" -ne 1 ]
  then
    echo "Usage: $0 <parcel-url>"
    echo ""
    echo "Alternatively, set the environment variable PARCEL_URL prior to"
    echo "running this script."
    exit 1
  else
    PARCEL_URL=$1
  fi
```

```
fi

sudo useradd -r cloudera-scm
sudo mkdir -p /opt/cloudera/parcels /opt/cloudera/parcel-repo /opt/cloudera/parcel-cache

PARCEL_NAME="${PARCEL_URL##*/}"

echo "Downloading parcel from $PARCEL_URL"
sudo curl "${PARCEL_URL}" -o "/opt/cloudera/parcel-repo/$PARCEL_NAME"
sudo curl "${PARCEL_URL}.sha1" -o "/opt/cloudera/parcel-repo/$PARCEL_NAME.sha"

sudo cp /opt/cloudera/parcel-repo/*.parcel /opt/cloudera/parcel-cache

sudo chown -R cloudera-scm.cloudera-scm /opt/cloudera
```

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.

Cloudera Director ships with an internal mappings file for all instance types that are available at the time of release. A customer 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`. An example file is provided in the `etc` directory as a basis for customization. A different location can be provided 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).

Note: Note: If you have installed Cloudera Director from a package, set the configuration property instead of relying on the standard location.

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, consult 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.

Cloudera Director ships with an internal mappings file for all instance types that are available at the time of release. A customer 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`. An example file is provided in the `etc` directory as a basis for customization. A different location can be provided 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).

> **Note:** If you have installed Cloudera Director from a package, set the configuration property instead of relying on the standard location.

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, consult 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.

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 on page 45 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.

Troubleshooting

This section describes common configuration and setup errors.

Configuration Issues

Depending on the size of the cluster, it can take up to 25 minutes or longer to deploy. If issues occur during deployment, check the following:

1. Incorrectly configured credentials
2. Overly specific region (us-west-2a instead of us-west-2)
3. Specifying default as the security group instead of its security group ID
4. Specifying a security group that is not part of the VPC

To troubleshoot configuration issues, view the `application.log` file.

After you correct the issue, delete the state.h2.db file and run Cloudera Director again.

DNS Issues

The AWS VPC must be set up for forward and reverse hostname resolution.

DHCP Issues

Depending on the size of the disk, it can take a while for the Cloudera Manager to become available. If it takes too long, do the following:

1. Log in to web console at https://aws.amazon.com/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 domain-name is specified:

```
domain-name = ec2.internal
domain-name-servers = AmazonProvidedDNS
```

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.

AMI Issues

It can be difficult to find a list of Amazon machine images (AMIs) that you can choose from. The following example shows how 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 http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.html.

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.