# Table of Contents

**Hue Versions**

**Hue Reference Architecture**

**General Guidelines**

**Hue Installation & Upgrade**

**Using Hue**

- Getting Started with Hue
- How to Enable SQL Editor Autocompleter in Hue
- How to Use Governance-Based Data Discovery
- Using Governance-based Data Discovery
- How to Use S3 as Source or Sink in Hue
- Populate S3 Bucket
- Create Table with S3 File
- Export Query Results to S3
- Troubleshoot Errors

**Hue Administration**

- Configuring Hue
- Hue Configuration Files and Safety Valves
- Hue Logs and Paths
- Customize the Hue Web UI
- Supported Browsers
- Customizing the Hue Web UI
- How to Enable Governance-Based Data Discovery
- Administrator Setup Tasks
- How to Enable Usage-Based Query Assistance for Hue
- Requirements
- Enabling Navigator Optimizer for Hue
- How to Enable S3 Cloud Storage in Hue
- Enable S3 in Hue with the S3 Connector Service
- Enable S3 in Hue with Safety Valves
- Generate Access Keys in AWS
- IAM Permissions Needed for Hue S3 Browser
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>How to Run Hue Shell Commands</td>
<td>42</td>
</tr>
<tr>
<td>Hue Custom Databases</td>
<td>43</td>
</tr>
<tr>
<td>Connect Hue to an External Database</td>
<td>43</td>
</tr>
<tr>
<td>Custom Database Concepts</td>
<td>44</td>
</tr>
<tr>
<td>Connect Hue to MySQL or MariaDB</td>
<td>44</td>
</tr>
<tr>
<td>Connect Hue to PostgreSQL</td>
<td>49</td>
</tr>
<tr>
<td>Connect Hue to Oracle with Client Parcel</td>
<td>54</td>
</tr>
<tr>
<td>Connect Hue to Oracle with Client Package</td>
<td>62</td>
</tr>
<tr>
<td>Hue Custom Database Tutorial</td>
<td>70</td>
</tr>
<tr>
<td>Migrate the Hue Database</td>
<td>73</td>
</tr>
<tr>
<td>Dump Database</td>
<td>73</td>
</tr>
<tr>
<td>Connect New Database</td>
<td>74</td>
</tr>
<tr>
<td>Synchronize and Load</td>
<td>74</td>
</tr>
<tr>
<td>How to Populate the Hue Database</td>
<td>77</td>
</tr>
<tr>
<td><strong>Tuning Hue Performance</strong></td>
<td>80</td>
</tr>
<tr>
<td>How to Add a Hue Load Balancer</td>
<td>80</td>
</tr>
<tr>
<td>Configure Hue for High Availability</td>
<td>80</td>
</tr>
<tr>
<td>Configure Hue for High Availability</td>
<td>80</td>
</tr>
<tr>
<td>Configure Hive and Impala for High Availability</td>
<td>81</td>
</tr>
<tr>
<td>Hue and HDFS High Availability</td>
<td>83</td>
</tr>
<tr>
<td>Configuring Hue to Work with HDFS HA Using Cloudera Manager</td>
<td>83</td>
</tr>
<tr>
<td><strong>Hue Security</strong></td>
<td>84</td>
</tr>
<tr>
<td>Hue User Permissions</td>
<td>84</td>
</tr>
<tr>
<td>Hue Superusers</td>
<td>84</td>
</tr>
<tr>
<td>Hue Applications and Permissions</td>
<td>85</td>
</tr>
<tr>
<td>Secure Hue Passwords with Scripts</td>
<td>86</td>
</tr>
<tr>
<td>Authenticate Hue Users with LDAP</td>
<td>87</td>
</tr>
<tr>
<td>Authenticate Hue Users and Groups with LDAP</td>
<td>87</td>
</tr>
<tr>
<td>Table of Hue LDAP Properties</td>
<td>92</td>
</tr>
<tr>
<td>Synchronize Hue with LDAP Server</td>
<td>94</td>
</tr>
<tr>
<td>Synchronize Hue Users and Groups with LDAP</td>
<td>94</td>
</tr>
<tr>
<td>Restrict Group Permissions</td>
<td>96</td>
</tr>
<tr>
<td>Authenticate Hue Users with SAML</td>
<td>96</td>
</tr>
<tr>
<td>Configure Hue for SAML Authentication</td>
<td>96</td>
</tr>
<tr>
<td>SAML Properties in hue.ini</td>
<td>99</td>
</tr>
<tr>
<td>Troubleshooting</td>
<td>101</td>
</tr>
<tr>
<td>Authorize Hue User Groups with Sentry</td>
<td>101</td>
</tr>
<tr>
<td>Prerequisites</td>
<td>101</td>
</tr>
<tr>
<td>Create Hue Users and Groups</td>
<td>102</td>
</tr>
<tr>
<td>Enable Sentry in Hue Security Browser</td>
<td>102</td>
</tr>
</tbody>
</table>
Hue Versions

Hue is released upstream, and is also packaged with CDH.
Hue packaged with CDH is tightly coupled and cannot be installed or upgraded separately.

Note: Hue package names = `<hue version>++<cdh version>++<changes.log>`. In CDH 5.12.0, the package name is `hue-3.9.0+cdh5.11.0+6396` because there are 6396 records in the corresponding `changes.log`.

### Table 1: Hue Version in CDH

<table>
<thead>
<tr>
<th>CDH Version</th>
<th>Hue Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.2</td>
<td>4.3</td>
</tr>
<tr>
<td>6.1</td>
<td>4.3</td>
</tr>
<tr>
<td>6.0</td>
<td>4.2</td>
</tr>
<tr>
<td>5.16</td>
<td>4.2</td>
</tr>
<tr>
<td>5.15</td>
<td>4.2</td>
</tr>
<tr>
<td>5.14</td>
<td>4.1</td>
</tr>
<tr>
<td>5.13</td>
<td>4.0</td>
</tr>
<tr>
<td>5.12</td>
<td>4.0</td>
</tr>
<tr>
<td>5.11</td>
<td>3.12</td>
</tr>
<tr>
<td>5.10</td>
<td>3.11</td>
</tr>
<tr>
<td>5.9</td>
<td>3.11</td>
</tr>
<tr>
<td>5.8</td>
<td>3.10</td>
</tr>
<tr>
<td>5.7</td>
<td>3.9</td>
</tr>
<tr>
<td>5.6</td>
<td>3.9</td>
</tr>
<tr>
<td>5.5</td>
<td>3.9</td>
</tr>
<tr>
<td>5.4</td>
<td>3.7</td>
</tr>
<tr>
<td>5.3</td>
<td>3.7</td>
</tr>
<tr>
<td>5.2</td>
<td>3.6</td>
</tr>
<tr>
<td>5.1</td>
<td>3.6</td>
</tr>
<tr>
<td>5.0</td>
<td>3.5</td>
</tr>
</tbody>
</table>

Links:
- Hue versions for each CDH 5.x.x release: [CDH 5 Packaging Information](https://github.com/cloudera/hue)
- GitHub repository: [https://github.com/cloudera/hue](https://github.com/cloudera/hue)
Hue Reference Architecture

This document provides a reference architecture for deploying Hue. It is a guide to assist with deployment and sizing options.

In practice, each Hue server can support approximately 25 concurrent users, depending on what tasks the users are performing. Most scaling issues occur as a result of users performing resource-intensive operations and not from the number of users. For example, large downloads of query results can impact resource availability for the other users who are using the same Hue instance during the download operation. During that time, the users can experience slow performance. Another common cause of noticeable performance changes are slow RPC calls between Hue and another service. When this happens, queries may appear to suddenly "hang" after they are submitted.

As a guide, 2 Hue servers can support up to:

- 100 unique users per week
- 50 users per hour at peak times executing up to 100 queries

A typical setup is 2 Hue servers.

General Guidelines

- Deploy a load balancer in front of Hue.
- Use a production-quality database. For more information, see Hue Custom Databases on page 43.
- Ensure that other services, such as Impala, Hive, and Oozie, are healthy and not impacted by too few resources. If these services are hanging, it adversely affects Hue performance.
- Consider moving workloads that are subject to SLAs (service-level agreements) or considered "noisy neighbors" to their own compute cluster. Noisy neighbors are workloads that use the majority of available resources and cause performance issues. For more information about separating compute and storage, see Virtual Private Clusters and Cloudera SDX.
- Limit the number of rows that are returned for queries.

One way to limit the number of rows returned is to specify a value for the download_row_limit configuration property for the Hue Beeswax application. This property can be set in the Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini property in Cloudera Manager:

1. In Cloudera Manager, click Hue > Configuration, and enter Hue Service Advanced Configuration Snippet in the search text box.
2. In the text box for the Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini, add the following configuration information:

   [beeswax]
   download_row_limit=number_of_rows

3. Click Save Changes and click the restart icon at the top of the page to restart the Hue service:
• Upgrade to CDH 5.15 or later, which includes Hue version 4.2. In Hue 4.2 and later, there are better query submission controls on the backend and you also gain the ability to visualize queued queries.
Hue Installation & Upgrade

Hue is included in Cloudera CDH, which you can install using one of the following methods:

- **Production installation** — Installs Cloudera Manager using system packages and installs CDH using either packages or parcels.
- **Non-production installation** — Installs Cloudera Manager and CDH using an automated installer and is intended only for non-production use. The installer configures an embedded PostgreSQL database for use with Hue, which is not suitable for production use.

See [Cloudera Installation Guide](#).

The Hue Server is a container web application that sits between your CDH installation and the browser. The Hue server hosts a suite of Hue applications and communicates with CDH component servers.
Using Hue

This section provides information about how to use Hue for the following tasks:
Try the following tutorial to get started using Hue immediately.

Getting Started with Hue

Analyze and visualize your data with Impala, a high-speed, low-latency SQL query engine.

Tip: If you don’t have your own Hue instance, try running sample queries on http://demo.gethue.com/.

1. Download and unzip one year of bike trips from the Bay Area Bike Share program. This file is about 80 MB in size.
2. Create a table from the ~/babs_open_data_year_1/201402_babs_open_data/201402_trip_data.csv file found in the unzipped babs_open_data_year_1.zip file:
   a. In the Cloudera Manager Admin Console, select Hue > WebUI > Hue Load Balanced to launch Hue.
   b. In the left panel of Hue, make sure the default database is selected, and click the plus sign to create a table:
   c. In the center panel Importer UI, set Type to File.
   d. Drag the 201402_trip_data.csv file to the Path field:
   e. Set the formats as follows:
      - Field Separator = Comma (,)

[Diagram of Hue interface showing table creation process]

If the default database is not selected, click the "less than" icon that is next to the database icon in the left panel. This enables you to select the default database.

[Diagram of Hue interface showing database selection process]
• Record Separator = New line
• Quote Character = Double Quote

Then click **Next** at the bottom of the page.

f. Set the properties **Format** = **Text**.
g. Edit the **FIELDS** as follows:
   • Rename Bike # to Bike ID
   • Change the data type of ZipCode to string.
   • Remove all of the spaces in the Name fields.

Then click **Submit** at the bottom of the page.

3. Click **Query** at the top of the page and select **Editor** > **Hive** to open the HiveQL editor and then create a query.
   • Enter the following query into the editor window:

   ```sql
   SELECT * FROM default.201402_trip_data
   LIMIT 10;
   ```

   • Click the execute icon to run the query. The following rows are returned:

4. Click **Query** at the top of the page and select **Editor** > **Impala** to open the Impala SQL editor and then create a query.
   a. In the left panel, click the refresh icon and select **Perform incremental metadata update** to make the new table visible to Impala:
b. Enter the following query into the editor window:

```
select 'startstation', 'endstation', count(*) as trips from default.'201402_trip_data'
group by 'startstation', 'endstation' order by trips desc;
```

c. Click the down arrow just under the execution icon and select **Format**:

This reformats the query:
d. Click the save icon, enter a query name, and click Save.

e. Click the execute icon to run the query.

5. Create a bar chart that is based on the query results:

a. Click the chart icon and then select Bars.

b. Set the bar chart elements as follows:
   - **X-AXIS** = startstation
   - **Y-AXIS** = trips
   - **LIMIT** = 10
6. Create a pie chart by clicking the chart icon again and then select Pie.

7. Download the query results by clicking the download icon and selecting in what format you want to download, copy, or export the results.

How to Enable SQL Editor Autocompleter in Hue

Autocompleter provides finely tuned SQL suggestions for Hive and Impala dialects while you enter queries into the editor window. See Brand new Autocompleter for Hive and Impala in the Hue blog.

Autocompleter is enabled by default. To manually enable or disable it, open the editor configuration panel and edit settings as follows:

1. Log in to Hue and go to either the Hive or Impala editor.
2. Place your cursor in the editor window and then use one of the following keyboard shortcuts to open the editor configuration panel:
   - For Macs, use the Command key followed by a hyphen and then a comma: Command-,
   - For Windows, use the Ctrl key followed by a hyphen and then a comma: Ctrl-

   Tip: Type a question mark (?) anywhere but in the active editor window to open a menu of editor keyboard shortcuts.
3. To enable autocompletion, check the box adjacent to Enable Autocompleter. When you check Enable Autocompleter, Enable Live Autocompletion is automatically enabled as well. Place your cursor in the editor window to close the configuration panel.
4. To disable:
   - Uncheck **Enable Live Autocompletion** but leave **Enable Autocompleter** checked, and then place your cursor in the editor window to close the configuration panel. This disables live autocompletion, but if you want to use autocompletion while building your queries in the editor, enter the following key stroke sequence to activate autocompletion: 
     Ctrl + Space Key
   - Uncheck both **Enable Autocompleter** and **Enable Live Autocompletion**, and then click in the editor to close the configuration panel. This disables all autocompletion functionality.

### How to Use Governance-Based Data Discovery

Hue can use the metadata tagging, indexing, and search features available in Cloudera Navigator data management. After integrating Hue with Cloudera Navigator tags and indexed entities can be accessed and viewed in Hue. You can also tag entities using Hue. Managed metadata and custom metadata tags that are created or applied using Hue can then stored in a Cloudera Navigator instance. This topic shows you how to use metadata tags in Hue.

Integration between Hue and Cloudera Navigator is enabled by default, but if your administrator has disabled it, it must be re-enabled before you can use governance-based data discovery as described below. To enable integration between Hue and Cloudera Navigator, see [How to Enable Governance-Based Data Discovery](#) on page 34.

### Using Governance-based Data Discovery

This short tutorial shows you how to create new metadata tags in Hue and search for them:

#### Defining Metadata Tags

**Prepare Tables for Tagging**

As the Hue administrator, install sample tables, and then refresh Impala metadata:

1. In the Cloudera Manager Admin Console, launch Hue by selecting **Clusters > Hue > Hue Web UI > Hue Load Balanced**.
2. In the top right corner of the page, select `<user_name>` > **Hue Administration > Step 2: Examples**
3. Install the sample tables for Hive and Impala by clicking the download icons.

4. At the top of the page, click **Query** and then select **Editor > Impala** to go to the Impala SQL editor, click the refresh icon, select **Perform incremental metadata update**, and then click **Refresh**.

5. In the upper left corner of the page, select the menu icon and then select **Browsers > Tables** to load the Table Browser.

6. In the Table Browser, click **Refresh** on the right side of the page. This ensures that the sample tables are available to work with in Hue.

**Tag a Database, Table, and Field**

1. At the top of the Table Browser page, click the table icon to display the Databases page, and then click **default** or some other database. This loads a detail view of the database.

2. To add a database tag, click **Add tags**, enter a tag name in the text box, click **Add <tag_name>**, and then click the check on the right to save the tag:
3. To add a table tag, click the a table name, for example customers, which loads a detail view of the table where you can define a tag:
   a. Click Add tags...:

   b. Enter a tag name in the text box, click Add <tag_name>..., and then click the check on the right to save the tag. This is the same process you performed for tagging the database in Step 2.

4. To add a field tag:
   a. Click the information icon by a column name:
b. In the dialog box that opens, click Add tags..., enter a tag name in the text box, click Add <tag_name>..., and then click the check on the right to save the tag.

Now, you have added tags to a database, a table, and a field (column) and you can search using these tag names as described in the next section.

Searching Metadata Tags

The SQL Editor in Hue provides a search text box where you can search on the metadata tags that are associated with your databases, tables, and columns.

**Note:** On clusters that use Sentry for role-based access control, the Search mechanism does not display counts of popular values. Sentry ensures that Hue users can view only entities to which their user role (as configured and managed by Sentry) has been granted specific permissions.

You can search for tags in either the Hive or the Impala editors:

1. Go to Query Editor > Impala.
2. To locate the tags you created, in the metadata search box located just to the right of the Query drop-down menu, type a tag name:
After you type the tag name in the search box, the `<database><table>` where the tag is found is returned. Click the `<database><table>` to view the tags that have been defined for it.

**How to Use S3 as Source or Sink in Hue**

On this page, we demonstrate how to write to, and read from, an S3 bucket in Hue.

**Populate S3 Bucket**

In this section, we use open data from the [U.S. Geological Survey](https://www.usgs.gov).

1. Download 30 days of earthquake data (all_month.csv) from the [USGS](https://www.usgs.gov) (~2 MB).
2. Log on to the Hue Web UI from Cloudera Manager.
3. Select File Browser > S3 Browser.
4. Click New > Bucket, name it "quakes_<any unique id>" and click Create.
   Tip: Unique bucket names are important per S3 bucket naming conventions.
5. Navigate into the bucket by clicking the bucket name.
6. Click New > Directory, name it "input" and click Create.
7. Navigate into the directory by clicking the directory name.
8. Click Upload and select, or drag, all_month.csv. The path is s3a://quakes/input/all_month.csv.

   ![](Important: Do not add anything else to the "input" directory—no extra files, no directories.

**Create Table with S3 File**

1. Go to the Metastore Manager by clicking Data Browsers > Metastore Tables.
2. Create a new table from a file by clicking .
3. Enter a Table Name such as "earthquakes".
4. Browse for the Input Directory, s3a://quakes/input/, and click Select this folder.
5. Select **Create External Table** from the Load Data menu and click **Next**.

6. Delimit by Comma(,) and click **Next**.

7. Click **Create Table**.

8. Click **Browse Data** to automatically generate a `SELECT` query in the **Hive** editor:

```
SELECT * FROM `default`.`earthquakes` LIMIT 10000;
```

### Export Query Results to S3

1. **Run and Export Results in Hive**
   a. Run the query by clicking **Execute**.
   b. Click **Get Results**.
   c. Select **Export** to open the **Save query result** dialog.

2. **Save Results as Custom File**
   a. Select **In store (max 10000000 cells)** and open the **Path to CSV file** dialog.
   b. Navigate into the bucket, `s3a://quakes`.
   c. Create folder named, "output."
   d. Navigate into the **output** directory and click **Select this folder**.
e. Append a file name to the path, such as `quakes.cvs`.
f. Click Save. The results are saved as `s3a:///quakes/output/quakes.csv`.

3. Save Results as MapReduce files
   a. Select In store (large result) and open the Path to empty directory dialog.
   b. Navigate into the bucket, `s3a:///quakes`.
   c. If you have not done so, create a folder named, "output."
   d. Navigate into the output directory and click Select this folder.
   e. Click Save. A MapReduce job is run and results are stored in `s3a:///quakes/output/`.

4. Save Results as Table
   a. Run a query for "moment" earthquakes and export:
      ```sql
      SELECT time, latitude, longitude, mag
      FROM `default`.'earthquakes'
      WHERE magtype IN ('mw', 'mwb', 'mwc', 'mwr', 'mww');
      ```
   b. Select A new table and input `<database>.<new table name>.
   c. Click Save.
   d. Click Browse Data to view the new table.

Troubleshoot Errors

This section addresses some error messages you may encounter when attempting to use Hue with S3.

Tip: Restart the Hue service to view buckets, directories, and files added to your upstream S3 account.

- Failed to access path
  Failed to access path: "s3a:///quakes". Check that you have access to read this bucket and that the region is correct.
  Possible solution: Check your bucket region:
  1. Log on to your AWS account and navigate to the S3 service.
  2. Select your bucket, for example "quakes", and click Properties.
  3. Find your region. If it says US Standard, then region=us-east-1.
  4. Update your configuration in Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety Valve.ini.
  5. Save your changes and restart Hue.

- The table could not be created
  Possible solution: Set your S3 credentials in Hive core-site.xml:
  1. In Cloudera Manager, go to Hive > Configuration.
2. Filter by **Category > Advanced.**
3. Set your credentials in **Hive Service Advanced Configuration Snippet (Safety Valve) for core-site.xml.**
   a. Click the ⚜️ button and input Name and Value for `fs.s3a.AccessKeyId`.
   b. Click the ⚜️ button and input Name and Value for `fs.s3a.SecretAccessKey`.
4. Save your changes and restart Hive.

- **The target path is a directory**
  Possible solution: Remove any directories or files that may have been added to `s3a://quakes/input/` (so that `all_month.csv` is alone).

- **Bad status for request TFetchResultsReq ... Not a file**
  ```plaintext
  Bad status for request TFetchResultsReq(...):
  TFetchResultsResp(status=TStatus(errorCode=0, errorMessage='java.io.IOException: java.io.IOException: Not a file: s3a://Not a file: s3a://quakes/input/output' ...
  ```
  Possible solution: Remove any directories or files that may have been added to `s3a://quakes/input/` (so that `all_month.csv` is alone). Here, Hive cannot successfully query the `earthquakes` table (based on `all_month.csv`) due to the directory, `s3a://quakes/input/output`.

  **Tip:** Run `tail -f` against the Hive server log in `/var/log/hive/`. 
Hue Administration

This section of the guide includes the following topics:

In addition to the topics contained in the Hue Guide, the following additional administration and configuration topics related to Hue are located in other sections of the Cloudera documentation set.

- Hue Browser Requirements
- Adding a Hue Service and Role Instance
- Enabling Hue Applications Using Cloudera Manager
- Managing Hue Analytics Data Collection
- Using Hue with Cloudera Search

Configuring Hue

This topic includes the following information about configuring Hue:

Hue Configuration Files and Safety Valves

All Hue configuration properties are stored in hue.ini but only a subset are exposed in Cloudera Manager. Properties that are not exposed (such as the Authentication Backend for SAML) can be configured in Cloudera Manager with "safety-valves."

Configuration Files

Hue roles are configured with three .ini files that are read in order (with the last one, hue_safety_valve_server.ini, taking precedence):

- hue.ini
- hue_safety_valve.ini
- hue_safety_valve_server.ini.

Important: Do not edit the .ini files at the command line because they are stored in dynamic directories named by process ID and populated from the Cloudera Manager database.

```bash
ls -ltr /var/run/cloudera-scm-agent/process/* | grep HUE_SERVER | tail -1 | awk '{print $9}''
```
Tip: The process directory for any given role is mirrored in Cloudera Manager. Go to Hue > Instances, select a role such as Hue Server, and then click the Processes tab.

Advanced Safety-valves

In Cloudera Manager, all Hue properties and safety-valves can be found on the tab, Hue > Configuration.

The field for hue_safety_valve.ini is service-wide and affects all Hue roles listed on the Instances tab. The field for hue_safety_valve_server.ini only affects Hue Server role instances.
**Environment Safety-valves**

*Environment* safety-valves let you configure environment variables across the service or for specific role instances. These safety-valves lack headers.
This section explains the logs generated by Cloudera Manager about each Hue role instance and those generated by the Hue service itself.

**Hue Logs**

There are two sets of Hue log files:

- Standard stream logs for the starting/stopping of each role supervised by Cloudera Manager:

  ```
  [root@hue4-cdh512-1 logs]$ pwd
  /var/run/cloudera-scm-agent/process/259-hue-HUE_LOAD_BALANCER/logs
  
  [root@hue4-cdh512-1 logs]$ ll
  total 16
  -rw-r--r-- 1 root root 11148 Aug 6 14:33 stderr.log
  -rw-r--r-- 1 root root 447 Aug 6 14:33 stdout.log
  ```

- Hue log4j logs when the service is running:

  ```
  [root@hue4-cdh512-1 logs]$ ls -ltr /var/log/hue
  total 3116
  -rw-r--r-- 1 hue hue 8040 Jul 23 09:26 ldaptest.log
  drwx------ 2 hue hue 89 Aug 4 09:17 audit
  -rw-r--r-- 1 hue hue 25735 Aug 4 09:17 syncdb.log
  -rw-r--r-- 1 hue hue 3275 Aug 4 09:17 migrate.log
  -rw-r--r-- 1 hue hue 70366 Aug 6 10:53 error.log
  -rw-r--r-- 1 hue hue 141048 Aug 6 11:19 hive_renamer.log
  -rw-r--r-- 1 hue hue 479464 Aug 6 12:08 runcpserver.log
  -rw-r--r-- 1 hue hue 2395951 Aug 6 12:08 access.log
  drwxr-xr-x 2 hue hue 24 Aug 6 12:08 metrics-hue_server
  ```
Standard Stream Logs

Cloudera Manager logs the starting and stopping of each supervised Hue process in standard stream logs (`stdout.log`, `stderr.log`).

When the Hue service restarts, Cloudera Manager generates a new directory for each supervised process (of a Hue role instance):

```
ls -vrl /var/run/cloudera-scm-agent/process | grep HUE
```

It writes to a nested `logs` directory for each running instance:

```
ls -vrl /var/run/cloudera-scm-agent/process | grep HUE
```

Configuration errors are written here because they prevent Hue servers and load balancers from starting properly.

**Tip:** `Test LDAP Configuration` also writes to standard stream logs: `ls -vrl /var/run/cloudera-scm-agent/process | grep ldaptest`

**The Supervisor**

The supervisor is a watchdog process and `supervisor.conf` manages all Hue processes; its only purpose is to spawn and monitor other processes. A standard Hue installation starts and monitors the `runcpserver` process, which provides the core web functionality for Hue.

**Note:** To see active supervisor processes, run: `ps -f -u hue`.

For each Hue role, Cloudera Manager looks to the appropriate `supervisor.conf` for instructions on how to start the server.

```
# Hue Server Process Directory
cd /var/run/cloudera-scm-agent/process/`ls -valrt /var/run/cloudera-scm-agent/process | grep HUE_SERVER | tail -1 | awk '{print $9}'`
cat supervisor.conf
```

```
[program:288-hue-HUE_SERVER]
command=cmf-redactor "'/usr/lib64/cmf/service/hue/hue.sh" "runcpserver"
autostart=true
directory=/run/cloudera-scm-agent/process/288-hue-HUE_SERVER
stdout_logfile=/run/cloudera-scm-agent/process/288-hue-HUE_SERVER/logs/stdout.log
stdout_logfile_maxbytes=10MB
stdout_logfile_backups=10
stderr_logfile=/run/cloudera-scm-agent/process/288-hue-HUE_SERVER/logs/stderr.log
stderr_logfile_maxbytes=10MB
```
stderr_logfile_backups=10
environment= ...

# Hue Load Balancer Process Directory
cd /var/run/cloudera-scm-agent/process/`ls -valrt /var/run/cloudera-scm-agent/process | grep HUE_LOAD | tail -1 | awk '{print $9}'`
cat supervisor.conf

[program:258-hue-HUE_LOAD_BALANCER]
command=cmf-redactor "'/usr/lib64/cmf/service/hue/httpd.sh"
...

Note: Currently, maxbytes=10MB, is hard-coded and cannot be changed for stdout or stderr.

If you installed other applications into your Hue instance, you may see other daemons running under the supervisor as well. Supervisor automatically restarts these processes if they fail for any reason. If they fail repeatedly in a short period of time, the supervisor itself shuts down.

Hue Service Django Logs

Once the Hue service is up and running, Hue generates logs in /var/log/hue using log4j. Load balancer logs are in /var/run/httpd.

Tip: You can view these logs in Hue at http://hueserver:port/logs

Table 2: Hue Service Logs

<table>
<thead>
<tr>
<th>Log Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>access.log</td>
<td>Filtered list of successful attempts to access Hue Web UI</td>
</tr>
<tr>
<td>audit/hue_server_audit_wal.log</td>
<td>Audit log visible in Cloudera Navigator</td>
</tr>
<tr>
<td>error.log</td>
<td>Filtered list of all nontrivial errors</td>
</tr>
<tr>
<td>kt_renewer.log</td>
<td>Kerberos ticket renews</td>
</tr>
<tr>
<td>metrics-hue_server/metrics.log</td>
<td>Populates charts in Cloudera Manager</td>
</tr>
<tr>
<td>migrate.log</td>
<td>Database and table migrations + First Run of Hue server</td>
</tr>
<tr>
<td>runcpserver.log</td>
<td>Hue (CherryPy) web server info (CP server runs Django core)</td>
</tr>
<tr>
<td>hue_install.log</td>
<td>Contains the log produced during installation</td>
</tr>
</tbody>
</table>

Enable DEBUG

DEBUG is available for the Hue Django logs in /var/log/hue.

By default, the Hue service writes INFO level messages and keeps a small buffer of log messages at all levels in memory.

There are two ways to enable DEBUG messages for all the logs in /var/log/hue:

- Cloudera Manager: Go to Hue > Configuration, check Enable Django Debug Mode, and Save Changes > Restart.
- Hue Web UI: Go to the Home page, select Server Logs, and check Force Debug Level. Debug is enabled on-the-fly.

Hue Paths

This section is a cheat sheet for navigating a managed deployment of Hue.

$HUE_HOME

$HUE_HOME is a reference to the location of your Hue installation.
• For package installs, this is usually /usr/lib/hue/
• For parcel installs, this is usually, /opt/cloudera/parcels/CDH/lib/hue/

The CDH directory is actually a symbolic link to the directory with the latest parcels:

```bash
ls -l /opt/cloudera/parcels
```

You can Run Hue Shell Commands from this location.

### Hue Role Instance Process Directories

Cloudera Manager creates dynamic process directories to manage each Hue role instance:

```bash
cd /var/run/cloudera-scm-agent/process/`ls -alrt /var/run/cloudera-scm-agent/process | grep HUE_SERVER | tail -1 | awk '{print $9}'`
cd /var/run/cloudera-scm-agent/process/`ls -alrt /var/run/cloudera-scm-agent/process | grep HUE_LOAD | tail -1 | awk '{print $9}'`
```

### Hue Server Configuration Files

Hue Configuration Files and Safety Valves on page 23 (hue.ini) and corresponding safety valve files, are in each Hue Server process directory:

```bash
ls -l /var/run/cloudera-scm-agent/process/`ls -alrt /var/run/cloudera-scm-agent/process | grep HUE_SERVER | tail -1 | awk '{print $9}'`
```

### Customize the Hue Web UI

This topic describes which browsers to use to get the best experience with the Hue Web UI and how you can customize the UI in the following sections:

**Important:** Hue requires that the machines in your cluster connect to each other freely over TCP. Machines not in the cluster must be able to open TCP port 8888 on the Hue Server (or the configured Hue web HTTP port) to interact with the system.

### Supported Browsers

Hue works with the two most recent LTS (long term support) or ESR (extended support release) browsers. Cookies and JavaScript must be on. The following lists the tested common browsers:

- **Chrome:** ([Version history](#))
- **Firefox:** ([Version history](#))
- **Safari** (Mac only): ([Version history](#))
- **Microsoft Edge:** ([Version history](#))

Hue can display in other browsers and in older versions of the common browsers, but you might not have access to all features.

### Customizing the Hue Web UI

To customize the Hue Web UI, you can add configuration properties in Cloudera Manager. You can customize the following Hue Web UI objects:
Banner

You can add a custom banner to the Hue Web UI by applying HTML directly to the property, `banner_top_html`. For example:

```
banner_top_html=<H4>My company's custom Hue Web UI banner</H4>
```

To configure a custom banner:

1. In the Cloudera Manager Admin Console, select **Clusters > Hue > Configuration** to navigate to the configuration page for Hue.
2. In the Search text box, type `top banner` to locate the **Top Banner Custom HTML banner_top_html** configuration parameter.
3. Add your custom HTML to the text box for the configuration parameter.
4. Click **Save Changes** at the bottom of the page to save the configuration change.
5. Refresh the browser page and click the restart icon at the top of the page so the new configuration changes can be read by the server:

6. In the Hue configuration page of Cloudera Manager, select **Web UI > Hue Load Balanced** to load Hue and view your custom banner.

Page Logo

You can also replace the Hue logo with a customized logo that is created with SVG code. You can add any type of logo you want, but your custom logo should be designed to fit into a 160 x 40 pixel space.

For example, here is the Hue logo shown below:

You can change this Hue logo by adding the appropriate SVG code to the `logo_svg` property under `desktop` `[custom]` in the Hue Service Advanced Configuration Snippet (Safety Valve) for `hue_safety_valve.ini` configuration parameter in Cloudera Manager as follows:

To replace the Hue logo with a custom logo:
1. In the Cloudera Manager Admin Console, select Clusters > Hue > Configuration to navigate to the configuration page for Hue.

2. In the Search text box type hue_safety_valve.ini to locate the configuration parameter:

3. Add the following parameters with your custom logo SVG code to the Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini configuration parameter:

   ```
   [desktop]
   [[custom]]
   logo_svg='<custom_svg_code_for_logo>
   ```

   For example, the following SVG code replaces the Hue logo with a red heart:

   ```
   [desktop]
   [[custom]]
   logo_svg='&lt;g&gt;&lt;path stroke="null" id="svg_1" d="m44.41215,11.43463c-4.05017,-10.71473
   -17.19753,-5.90773 -18.41353,-0.5567c-1.672,-5.70253 -14.497,-9.95663
   -18.411,0.5643c-4.35797,11.71793 16.891,22.23443 18.4163,23.95773c1.5181,-1.36927
   22.7696,-12.43803
   18.4129,-23.96533z" fill="#ffffff"/&gt;&lt;/g&gt;
   ```

4. Click Save Changes at the bottom of the page to save the configuration change.

5. Refresh the browser page and click the restart icon at the top of the page so the new configuration changes can be read by the server.

6. In the Hue configuration page of Cloudera Manager, select Web UI > Hue Load Balanced to load Hue and view your custom logo.
If you added the sample SVG code that defines a red heart as the logo, your Hue web UI looks like this:

Splash Screen
You can customize a splash screen on the login page by specifying HTML for the login_splash_html property under [desktop] [[custom]] in the Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini configuration parameter in Cloudera Manager. For example, you could add the following warning message to the splash screen by specifying the following for this property:

```
[desktop]
[[custom]]
login_splash_html=WARNING: You are required to have authorization before you proceed.
```

The above configuration adds the following message to the Hue login screen:

To configure a splash screen in Cloudera Manager:
1. In the Cloudera Manager Admin Console, select Clusters > Hue > Configuration to navigate to the configuration page for Hue.

2. In the Search text box, type `hue_safety_valve.ini` to locate the Hue Service Advanced Configuration Snippet (Safety Valve) for `hue_safety_valve.ini` configuration parameter.

3. Add the following parameters with your custom splash screen message to the Hue Service Advanced Configuration Snippet (Safety Valve) for `hue_safety_valve.ini` configuration parameter:

```ini
[desktop]
[[custom]]
login_splash_html=<custom_message>
```

4. Click Save Changes at the bottom of the page to save the configuration change.

5. Refresh the browser page and click the restart icon at the top of the page so the new configuration changes can be read by the server.

6. In the Hue configuration page of Cloudera Manager, select Web UI > Hue Load Balanced to load Hue and view your custom splash screen message.

**Cache Timeout**

Enable Hue UI caching by setting a timeout in milliseconds. The default is 86400000 milliseconds or one day. Set the timeout to 0 to disable caching.

You can set the cache timeout using the `cacheable_ttl` property under `[desktop][[custom]]` in the Hue Service Advanced Configuration Snippet (Safety Valve) for `hue_safety_valve.ini` configuration property in Cloudera Manager as follows.

To set the cache timeout value:

1. In the Cloudera Manager Admin Console, select Clusters > Hue > Configuration to navigate to the configuration page for Hue.

2. In the Search text box, type `hue_safety_valve.ini` to locate the Hue Service Advanced Configuration Snippet (Safety Valve) for `hue_safety_valve.ini` configuration parameter.

3. Add the following parameters with the cache timeout value to the Hue Service Advanced Configuration Snippet (Safety Valve) for `hue_safety_valve.ini` configuration parameter:

```ini
[desktop]
[[custom]]
cacheable_ttl=<value_in_milliseconds>
```

For example, the following configuration sets the cache timeout to the default value of 86400000 milliseconds:

```ini
[desktop]
[[custom]]
cacheable_ttl=86400000
```

4. Click Save Changes at the bottom of the page to save the configuration change.

5. Refresh the browser page and click the restart icon at the top of the page so the new configuration changes can be read by the server and the new cache timeout limit takes effect.

**Anonymous Usage Data Collection**

Hue tracks anonymized pages and application versions to gather information about application usage levels. The data collected does not include hostnames or IDs.
You can restrict or enable data collection by setting the `collect_usage` parameter to `false` or `true` under the `[desktop]` configuration section in the Hue Service Advanced Configuration Snippet (Safety Valve) for `hue_safety_valve.ini` configuration property in Cloudera Manager.

To enable or disable anonymous usage data collection:

1. In the Cloudera Manager Admin Console, select Clusters > Hue > Configuration to navigate to the configuration page for Hue.
2. In the Search text box, type `hue_safety_valve.ini` to locate the Hue Service Advanced Configuration Snippet (Safety Valve) for `hue_safety_valve.ini` configuration parameter.
3. Add the following parameter to the Hue Service Advanced Configuration Snippet (Safety Valve) for `hue_safety_valve.ini` configuration parameter:

```ini
[desktop]
...#
# Help improve Hue with anonymous usage analytics.
# Use Google Analytics to see how many times an application or specific section of an
# application is used, nothing more.
## collect_usage=<false | true>
```

Where `false` disables data collection and `true` enables it.

4. Click Save Changes at the bottom of the page to save the configuration change.
5. Refresh the browser page and click the restart icon at the top of the page so the new configuration changes can be read by the server and the new data collection setting takes effect.

How to Enable Governance-Based Data Discovery

Hue can use the metadata tagging, indexing, and search features available with Cloudera Navigator data management. After integrating Hue with Cloudera Navigator, existing Cloudera Navigator tags and indexed entities can be accessed and viewed in Hue. You can also tag entities using Hue. Managed metadata and custom metadata tags that are created or applied using Hue are then stored in a Cloudera Navigator instance. This topic shows administrators how to enable this capability with Cloudera Manager.

Administrator Setup Tasks

Enabling Cloudera Navigator for Hue

To use Hue with Cloudera Navigator, you must give the Hue server access to the Navigator Administrator account and enable the integration by configuring some properties using the Cloudera Manager Admin Console.

**Note:** After enabling the integration on clusters that use Cloudera Sentry role-based access control, different Hue users can view only those entities to which their respective user roles have been granted permission.

Requirements

Follow the steps below to integrate Cloudera Navigator with the Hue server. These steps require Cloudera Navigator to already be installed, configured, and running in the context of a Cloudera Manager cluster. See Cloudera Data Management Guide for more information about Cloudera Navigator.

The administrator performing the configuration tasks must have the Cloudera Manager user role of Navigator Administrator or Full Administrator. Use the same account that was used to set up authentication for Cloudera Navigator users and groups.

**Enabling the Integration and Configuring Authentication**

1. Log in to the Cloudera Manager Admin Console.
2. Select Clusters > Hue.
3. Click the Configuration tab.
4. Select Service-wide from the Scope filter.
5. Select Cloudera Navigator from the Category filter. The properties for Cloudera Navigator configuration for Hue display:
   a. Select the Enable Navigator Metadata Server Integration check box.
   b. Select one of the below authentication mechanisms for Navigator Metadata Server Auth used by the Cloudera Navigator instance. This selection must match the configuration for Navigator Metadata Server:
      - Cloudera Manager
      - LDAP (Active Directory, OpenLDAP)
      - SAML (for SSO support)
   c. Select the Enable Audit Collection check box.
6. Click Save Changes at the bottom of the page to save the configuration change.
7. Refresh the browser page and click the restart icon at the top of the page so the new configuration changes can be read by the server.
8. Log in to Hue by selecting Web UI > Hue Load Balanced.

How to Enable Usage-Based Query Assistance for Hue

Hue can leverage usage statistics for recommendations and risk alerts during query development for Hive and Impala. This includes autocomplete suggestions based on most used (or “popular”) values from SELECT, FROM, JOIN, WHERE, GROUP BY, and ORDER BY queries, so SQL users can easily identify tables and other items of interest. This is particularly useful to discover and prioritize when there are thousands of tables and columns available. It also helps for guessing join conditions based on this usage.

Before queries are submitted, Hue also leverages risk evaluations to help protect against inefficient or resource-heavy queries. Risk alerts and query suggestions are provided within the interface during development.

Both usage statistics and risk evaluations are gathered by Cloudera Navigator Optimizer. Organizations can gain the benefit of statistically derived optimizations by integrating Cloudera Navigator Optimizer with Hue as detailed below.

For more information about Hue Query Assistant with Navigator Optimizer Integration, see the Cloudera Engineering Blog post New in Cloudera Enterprise 5.12: Hue 4 Interface and Query Assistant.

Requirements

Navigator Optimizer is a cloud-based component available only with select Cloudera Enterprise editions. It requires a Cloudera Altus account. To request an Altus account, see the Altus registration page and complete the necessary Altus setup and any other steps needed to obtain an Altus access key and private key. The steps below assume that these have already been obtained and the administrator integrating Hue and Navigator Optimizer has them available during the process.

In addition to the required Altus account, the administrator performing the tasks below must be a Cloudera Manager administrator, that is, have an account that has the Cloudera Manager Full Administrator user role.

The Hue application must also have access to the Navigator Optimizer public API, which gives Hue access to the Navigator Optimizer service. The Navigator Optimizer public API is available at the following URL:

https://navoptapi.us-west-1.optimizer.altus.cloudera.com:443

Enabling Navigator Optimizer for Hue

To enable Navigator Optimizer for use by the Hue server running on your Cloudera Manager cluster:

1. Log in to the Cloudera Manager Admin Console.
2. Select **Administration > Altus Credentials**. The Add Access Key Authentication page displays.

   a. Enter a meaningful name for the Altus credential.

   b. Enter the access key ID and the private key. For example:

   ![Add Access Key Authentication](image)

   c. Click **Add** to save the credentials and dismiss the dialog.

   The credential are now listed on the Altus Credentials page:

   ![Altus Credentials](image)

   You can now configure Hue to use the credentials.

3. Select **Clusters > Hue**.

4. Click the **Configuration** tab.

5. Click **Cloudera Navigator Optimizer** under the **Category** filter.

6. In the **Cloudera Altus Access Key** drop-down, select your key.

7. Check **Enable Cloudera Navigator Optimizer integration with Hue**.
8. Click **Save Changes**.
9. **Restart** Hue.

**How to Enable S3 Cloud Storage in Hue**

**Cloudera S3 Connector** in Cloudera Manager securely connects your CDH cluster to Amazon S3.

**Note:**
- C5.11 adds **S3 Guard** for list consistency and support for IAM roles in Cloudera Manager.
- C5.10 connects Hue, Impala, and Navigator securely with the Cloudera S3 Connector Service.
- C5.9 adds support for Amazon S3 with plain-text credentials using Cloudera Manager safety valves.

**Enable S3 in Hue with the S3 Connector Service**

For a secure and fine-grained connection to Amazon S3 (for Hue, Impala, and Navigator), Cloudera recommends its S3 Connector service in Secure Mode with encrypted access keys and Kerberos and Sentry installed.

<table>
<thead>
<tr>
<th>Method</th>
<th>Security</th>
<th>Required</th>
<th>Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>Secure Mode</td>
<td>High</td>
<td>Kerberos, Sentry</td>
<td>Hue, Impala, Navigator, Hive</td>
</tr>
<tr>
<td>Unsecure Mode</td>
<td>Medium</td>
<td></td>
<td>Hue, Impala, Navigator, Hive</td>
</tr>
</tbody>
</table>

1. Log on to Cloudera Manager.
2. Select **Administration > External Accounts**.
3. Click **Add Access Key Credentials** or **Add IAM Role-based Authentication**.

**Important:** IAM Role-based Authentication is not fine-grained authentication. Also, to use it with Hue, configure the region in *hue_safety_valve.ini*—see step step 11.

4. Add any **Name** and enter your S3 credentials:
   - **a.** To connect your AWS *root user*, add the **Access Key ID** and **Secret Access Key** for your *root account*. 
b. To connect an IAM user, add the Access Key ID and Secret Access Key for a read-only IAM account.

5. If you have an Amazon DynamoDB database, check Enable S3Guard for consistent read operations.

![Warning] Components writing data to S3 are constrained by the inherent Amazon S3 limitation known as "eventual consistency." This can lead to data loss when a Spark or Hive job writes output directly to S3. Cloudera recommends that you use S3 Guard or write to HDFS and distcp to S3.

6. Click Enable for <cluster name> to give Hue access to S3 and S3-backed tables. Impala must have permissions defined in Sentry.

7. If using access keys, select Secure or Unsecure mode.

8. Click Continue (at Step 1) if your cluster passes validation. You are automatically taken to step 5.

9. Click Continue (at Step 5) to restart Hive, Impala, Oozie, and Hue.

10. When finished, click Home to see the S3 Connector.

![Note] A gray status icon means the S3 Connector service was successfully added.

11. If using IAM roles, set the region to us-east-1 (N. Virginia) in hue_safety_valve.ini. If not, ignore this step.

![Note] Configuring hue_safety_valve.ini is a temporary Hue workaround for CDH 5.10.

a. Select Configuration > Advanced Configuration Snippets.

b. Filter by Scope > Hue.

c. Set Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini with the following:

```ini
[aws]
[[aws_accounts]]
[[[default]]]
region=us-east-1
```

d. Click Save Changes.

e. Restart Hue: select Cluster > Hue and Actions > Restart.

![Note] The S3 Connector service is not added when you use IAM roles.

Related topics: How to Configure AWS Credentials and Configuring the Amazon S3 Connector.
Enable S3 in Hue with Safety Valves

This section assumes an AWS account with access keys, but not necessarily a Kerberized cluster.

You can connect to S3 using three safety valves (also known as Advanced Configuration Snippets):

- **Hue Service** Advanced Configuration Snippet (Safety Valve) for `hue_safety_valve.ini`
- **Cluster-wide** Advanced Configuration Snippet (Safety Valve) for `core-site.xml`
- **Hive Service** Advanced Configuration Snippet (Safety Valve) for `core-site.xml`.

1. Log on to Cloudera Manager and select **Clusters** > **your cluster**.
2. Select **Configuration** > **Advanced Configuration Snippets**.
3. Filter by **Scope** > **Hue**.
4. Set your S3 credentials in **Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini**:

   Note: Store your credentials in a script that outputs to stdout. A **security_token** is optional.

   ```
   [aws]
   [[aws_accounts]]
   [[[default]]]
   access_key_id_script=/path/to/access_key_script
   secret_access_key_script=/path/to/secret_key_script
   #security_token=<your AWS security token>
   allow_environment_credentials=false
   region=<your region, such as us-east-1>
   
   For a proof-of-concept installation, you can add the IDs directly.
   ```

   ```
   access_key_id=<your_access_key_id>
   secret_access_key=<your_secret_access_key>
   ```

5. Clear the scope filters and search on "core-site.xml".
6. To enable the S3 Browser, set your S3 credentials in Cluster-wide Advanced Configuration Snippet (Safety Valve) for core-site.xml:

```xml
<property>
  <name>fs.s3a.access.key</name>
  <value>AWS access key ID</value>
</property>
<property>
  <name>fs.s3a.secret.key</name>
  <value>AWS secret key</value>
</property>
```

7. To enable Hive with S3, set your S3 credentials in Hive Service Advanced Configuration Snippet (Safety Valve) for core-site.xml.

8. Click Save Changes.

9. Restart Hue: select Cluster > Hue and Actions > Restart.


Related topics: Amazon Web Services (AWS) Security.

Generate Access Keys in AWS

To integrate Hue with S3, you must have an Amazon Web Services (AWS) account, with access keys for either your root user or a read-only IAM user.

**Root Account**

1. Create an AWS account and sign in to the AWS Console.
2. Create access keys for this AWS root account:
   a. Expand the drop-down menu under your account name and select My Security Credentials.
   b. Click Continue to Security Credentials.
   d. Click Create New Access Key.
   e. Click Show Access Key or Download Key File. These are your AWS root credentials.

**IAM Account**

1. Create two IAM groups (AWS admin and S3 Read-only):
   
   **Important:** AWS requires that your first IAM group and associated user has administrator access.

   a. Go to the IAM service.
   b. Click Groups and Create New Group.
   c. Enter a name and click Next Step.
   d. Filter on "admin" and select the AdministratorAccess policy.
   e. Click Next Step and Create Group.
   f. Create a second group with AmazonS3ReadOnlyAccess.

2. Create two IAM users and assign one to the admin policy and one to the S3 read policy.
   a. Click Users and Add User.
   b. Enter a name, and at a minimum, select Programmatic access.
   c. Click Next: Permissions.
   d. Select the group with administrator permissions.
   e. Click Next: Review and Create User.
   f. Create a second user and assign the group with S3 read-only access.
3. Create access keys for your read-only IAM user:
   a. Click the name of your read-only IAM user.
   b. Click the Security Credentials tab.
   c. Click Create Access Key.
   d. Click Show Access Key or Download Key File. These are your IAM user credentials.

IAM Permissions Needed for Hue S3 Browser

In AWS, IAM files are used to create policies that control access to resources in a VPC. You can give IAM roles and permissions to your Hue servers to allow the Hue S3 browser to make API requests without the need to use or distribute AWS credentials (accessKey and secretAccessKey). For more information about IAM, see the AWS Identity and Access Management User Guide in the AWS documentation. For instructions on how to create an IAM role, see Creating a Role to Delegate Permissions to an AWS Service in the AWS documentation. For information about granting permission to Amazon S3 resources, see Managing Access Permissions to Your Amazon S3 Resources in the AWS documentation.

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

- Only the Hue servers need to have an IAM role applied to them to access S3 with the browser.
- The Hue S3 browser does not become available until the Amazon S3 connector service is added to the cluster.
- ListBucket on the Amazon S3 resource is necessary to drill down into that bucket, along with ListAllMyBuckets on all resources.
- These permissions do not give access to other private buckets in that AWS account, although public buckets are accessible.
- For the Hue S3 browser, your Hue servers require permissions for the following methods:
  - s3:ListBucket
  - s3:PutObject
  - s3:GetObject
  - s3:DeleteObject
  - s3:PutObjectAcl
  - s3:ListAllMyBuckets

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

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "S3BucketPermissions",
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::<S3BucketARN>"
      ]
    },
    {
      "Sid": "S3ObjectPermissions",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:DeleteObject",
        "s3:PutObjectAcl",
        "s3:ListAllMyBuckets"
      ],
      "Resource": [
        "arn:aws:s3:::<S3BucketARN>/*"
      ]
    }
  ]
}
```
How to Run Hue Shell Commands

You may need to administer Hue programmatically, for example, to reset the superuser password or assign an LDAP user superuser permissions. You must be logged in as root on a host with a Hue server to do so.

1. Set HUE_CONF_DIR to the latest Hue process directory:

```bash
export HUE_CONF_DIR="/var/run/cloudera-scm-agent/process/`ls -alrt
/var/run/cloudera-scm-agent/process | grep HUE_SERVER | tail -1 | awk '{print $9}'``”
echo $HUE_CONF_DIR
```

2. Set environment variables used to run the Hue webserver:

   - CentOS/RHEL:

     ```bash
     for line in `strings /proc/$(lsof -i :8888|grep -m1 python|awk '{ print $2 }')/environ|egrep -v ^HOME=|^TERM=|^PWD=`;do export $line;done
     ```

   - Ubuntu:

     ```bash
     for line in `strings /proc/$(lsof -i :8888|grep -m1 hue|awk '{ print $2 }')/environ|egrep -v ^HOME=|^TERM=|^PWD=`;do export $line;done
     ```

3. Run shell subcommands

   When true, HUE_IGNORE_PASSWORD_SCRIPT_ERRORS runs the Hue shell even if hue.ini contains passwords generated by Cloudera Manager (such as bind_password and ssl_password).

   **Note:** Do not export HUE_IGNORE_PASSWORD_SCRIPT_ERRORS or HUE_DATABASE_PASSWORD to ensure that they are not stored and only apply to this command.

For CDH parcel deployments:

   - Run the interactive Hue Python shell (Ctrl+D to quit)

```bash
HUE_IGNORE_PASSWORD_SCRIPT_ERRORS=1 /opt/cloudera/parcels/CDH/lib/hue/build/env/bin/hue shell
```

Or with the database password:

```bash
HUE_IGNORE_PASSWORD_SCRIPT_ERRORS=1 HUE_DATABASE_PASSWORD=<your db password>
```

```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
• Change a user password

```bash
HUE_IGNORE_PASSWORD_SCRIPT_ERRORS=1 /opt/cloudera/parcels/CDH/lib/hue/build/env/bin/hue
changepassword admin
```

• Promote Hue user to superuser

```python
from django.contrib.auth.models import User
a = User.objects.get(username='gwen')
a.is_superuser = True
a.save()
```

• Count all of the documents of a certain user:

```python
from django.contrib.auth.models import User
from desktop.models import Document2
user=User.objects.get(username='demo')
Document2.objects.documents(user=user).count()
```

Out[1]: 1167

• List available subcommands

```bash
HUE_IGNORE_PASSWORD_SCRIPT_ERRORS=1 /opt/cloudera/parcels/CDH/lib/hue/build/env/bin/hue
```

For CDH package deployments:

```bash
HUE_IGNORE_PASSWORD_SCRIPT_ERRORS=1 /usr/lib/hue/build/env/bin/hue shell
```

For unmanaged and lower CDH versions, see:

- Execute some builtin or shell commands
- Storing passwords in file script
- How to change or reset a forgotten password?

---

**Hue Custom Databases**

Hue needs its own database for such things as user account information, job submissions, and SQL queries.

Hue is packaged with a lightweight **embedded database** (PostgreSQL) for proof-of-concept deployments with one Hue server. Hue also supports connections to a custom **external database**, local or remote.

**Important:** Cloudera recommends an external database in production environments.

**Connect Hue to an External Database**

- [Connect Hue to MySQL or MariaDB](#) on page 44
- [Connect Hue to PostgreSQL](#) on page 49
- [Connect Hue to Oracle with Client Parcel](#)
Custom Database Concepts

- **There are two ways to connect** Hue to an external database:
  - During a new CDH installation with the Cloudera Manager Installation Wizard at Database Setup. The external (or custom) database must be installed, configured, and running.
  - After CDH is installed with Cloudera Manager on the Hue > Configuration tab. You can migrate and connect, or simply connect to the new database without saving the data in the old database.

- **Migrate to a new database only if** you want to save data in your current database. Otherwise, simply connect to your new database and restart Hue.

- **An external database can be remote**—it does not need to be on the same host as the Hue server. Ensure the database server is properly configured (particularly the bind or listen address).

- **Managed CDH deployments** must use Cloudera Manager to configure an external database:
  1. In the Cloudera Manager Admin Console, select Clusters > Hue > Configuration to navigate to the configuration page for Hue.
  2. In the Search text box, type database to locate the database configuration parameters for Hue:
     - **Hue Database Type**: Select what type of database you want to use.
     - **Hue Database Directory**: Type the path to the database location.
     - **Hue Database Hostname**: Type the name of the host where the Hue database is running.
     - **Hue Database Port**: Type of number of the port on the host where the Hue database is listening.
     - **Hue Database Username**: Type of name of the user who is used to log on to the Hue database.
     - **Hue Database Password**: Type the password for the user who is specified in the Hue Database Username property.
     - **Hue Database Name**: Type the name of the Hue database.
     - **Database Dump File**: Type the path to the file where the database is dumped to or loaded from.
  3. Click **Save Changes** at the bottom of the page to save the configuration change.
  4. Refresh the browser page and click the restart icon at the top of the page so the new configuration changes can be read by the server.

Connect Hue to MySQL or MariaDB

- If you have an external database installed, review [MySQL/MariaDB Troubleshooting](#) on page 44 before creating a database for Hue.

Install and Configure MySQL or MariaDB Server

MariaDB is a fork of the MySQL relational database. Refer to the MariaDB documentation or MySQL documentation for more help on how to install a MariaDB or MySQL database.

MySQL/MariaDB Troubleshooting

Pay close attention to these areas and revisit when troubleshooting:

- **Remote connections**:
  - The bind or address should be set to 0.0.0.0 so it can listen to multiple hosts.
  - Grant wildcard (%) permissions to the Hue database user so it can connect from any host.
  - Install a JDBC connector if necessary, for example, if your CDH version does not include it.

- **Security**: Delete anonymous users because they are able to log on without a password.

- **Storage engine**: Use InnoDB (the default engine in version 5.5.5 and higher: `mysql -V`).

- **Data validation**: Use `sql_mode=STRICT_ALL_TABLES` to prevent columns being truncated during migration.
Install MySQL or MariaDB Server

1. Install MariaDB or MySQL. The table lists the maximum version of each supported distribution for this CDH release, and corresponding default database versions.

### Table 3: Install Commands for Supported OS Versions

<table>
<thead>
<tr>
<th>OS</th>
<th>OS Ver</th>
<th>DB Ver</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS / RHEL</td>
<td>7.3</td>
<td>5.1</td>
<td><code>sudo yum install mariadb-server</code></td>
</tr>
<tr>
<td></td>
<td>6.8</td>
<td></td>
<td><code>sudo yum install mysql-server</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>No package mariadb-server available.</strong></td>
</tr>
<tr>
<td>SLES</td>
<td>12.2</td>
<td>10.0</td>
<td><code>sudo zypper install mariadb</code></td>
</tr>
<tr>
<td></td>
<td>11.4</td>
<td>5.5</td>
<td><code>sudo zypper install mysql</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>'mariadb' not found in package names.</strong></td>
</tr>
<tr>
<td>Ubuntu</td>
<td>16.04</td>
<td>5.7</td>
<td><code>sudo apt-get install mysql-server</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>10.0</td>
<td><code>sudo apt-get install mariadb-server</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>#set root psswd when prompted</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>5.5</td>
<td><code>sudo apt-get install mysql-server</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>#set root psswd when prompted</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>5.5</td>
<td><code>sudo apt-get install mariadb-server</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>#set root psswd when prompted</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>12.04</td>
<td><code>sudo apt-get install mysql-server</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>#set root psswd when prompted</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>5.5</td>
<td><code>sudo apt-get install mariadb-server</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>#set root psswd when prompted</code></td>
</tr>
<tr>
<td>Debian</td>
<td>8.4</td>
<td>5.5</td>
<td><code>sudo apt-get install mysql-server</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>10.0</td>
<td><code>sudo apt-get install mariadb-server</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>#set root psswd when prompted</code></td>
</tr>
<tr>
<td></td>
<td>7.8</td>
<td>5.5</td>
<td><code>sudo apt-get install mysql-server</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>#set root psswd when prompted</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Package 'mariadb-server' has no installation candidate</strong></td>
</tr>
</tbody>
</table>

2. Start the database server as necessary (some are automatically started):
Table 4: Start Commands

<table>
<thead>
<tr>
<th>OS</th>
<th>OS Ver</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS / RHEL</td>
<td>7.3</td>
<td>sudo systemctl start mariadb</td>
</tr>
<tr>
<td></td>
<td>6.8</td>
<td>sudo service mysql start</td>
</tr>
<tr>
<td>SLES</td>
<td>11.4, 12.1, 12.2</td>
<td>sudo rcmysql start</td>
</tr>
<tr>
<td>Ubuntu</td>
<td>12.04, 14.04, 16.04</td>
<td>sudo service mysql start</td>
</tr>
<tr>
<td>Debian</td>
<td>7.8, 8.4</td>
<td>sudo service mysql start</td>
</tr>
</tbody>
</table>

3. Secure your installation. If you make a mistake, simply rerun:

```bash
sudo /usr/bin/mysql_secure_installation
```

Enter current password for root (enter for none): [If unset, press Enter.]
OK, successfully used password, moving on...
Set root password? [Y/n] Y [Enter n if password is set.]
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

Configure MySQL or MariaDB Server

1. Configure my.cnf (only as necessary).
   - Ensure `bind-address=0.0.0.0` (or is commented out if the default).
   - Ensure `default-storage-engine=innodb` (which is the default in 5.5 and higher: `mysql -V`).
   - Ensure `sql_mode=STRICT_ALL_TABLES` to avoid columns being truncated during migration.

```
[mysqld]
...
bind-address=0.0.0.0
default-storage-engine=innodb
sql_mode=STRICT_ALL_TABLES
```

   - CentOS/RHEL/SLES: /etc/my.cnf
   - Ubuntu/Debian: /etc/mysql/my.cnf

2. Restart the database server.

   **Note:** See the Table 4: Start Commands on page 46 table above and replace with "restart".

3. Enable the server to automatically start on boot:
Table 5: Enable Automatic Start

<table>
<thead>
<tr>
<th>OS</th>
<th>OS Ver</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS / RHEL</td>
<td>7.3</td>
<td><code>sudo systemctl enable mariadb</code></td>
</tr>
<tr>
<td></td>
<td>6.8</td>
<td><code>sudo chkconfig mysqld on</code></td>
</tr>
<tr>
<td>SLES</td>
<td>11.4, 12.1, 12.2</td>
<td><code>sudo chkconfig mysql on</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>sudo rcmysql status</code></td>
</tr>
<tr>
<td>Ubuntu</td>
<td>12.04, 14.04, 16.04</td>
<td><code># preconfigured to start at boot</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>sudo service mysql status</code></td>
</tr>
<tr>
<td>Debian</td>
<td>7.8, 8.4</td>
<td><code># preconfigured to start at boot</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>sudo service mysql status</code></td>
</tr>
</tbody>
</table>

Create Hue Database

1. Log on to MySQL or MariaDB with your root password:

   ```bash
   mysql -u root -p
   Enter password: <root password>
   ```

2. Create a database for Hue (we call it “hue” but any name works) with UTF8 collation and grant user privileges:

   ```sql
   MariaDB [(none)]:> CREATE DATABASE hue DEFAULT CHARACTER SET utf8 DEFAULT COLLATE utf8_general_ci;
   Query OK, 1 row affected (0.00 sec)
   MariaDB [(none)]:> GRANT ALL ON hue.* TO 'hue'@'%' IDENTIFIED BY '<password>';
   Query OK, 0 rows affected (0.00 sec)
   ```

3. Verify the connection to the Hue database:

   ```bash
   mysql -u hue -p
   Enter password: <password>
   quit
   ```

Note:

Ensure Hue uses UTF8 collation and character set. Some commands:

```
# To create (use utf8_general_ci or utf8mb4_general_ci):
CREATE DATABASE hue COLLATE = 'utf8_general_ci';

# To view default_character_set_name and default_collation_name
SELECT * FROM INFORMATION_SCHEMA.SCHEMATA;

# To alter if not created with UTF8 collation
ALTER DATABASE hue COLLATE = 'utf8_general_ci';
```

See [Setting Character Sets and Collations](#).
Connect Hue Service to MySQL

**Tip:** To save the data in your current database (embedded or external), you must migrate (dump, synch, load) before connecting to the new database. Otherwise, skip those steps.

1. **Stop Hue Service**
   a. In Cloudera Manager, navigate to **Cluster > Hue**.
   b. Select **Actions > Stop**.

   ![Note](image)
   
   **Note:** Refresh the page if the Hue service does not look stopped: 🔄.

2. **[migration only] Dump Current Database**
   a. Select **Actions > Dump Database**.
   b. Click **Dump Database**. The file is written to `/tmp/hue_database_dump.json` on the host of the Hue server.
   c. Log on to the host of the Hue server in a command-line terminal.
   d. Edit `/tmp/hue_database_dump.json` by removing all objects with `useradmin.userprofile` in the `model` field. For example:

   ```
   # Count number of objects
   grep -c useradmin.userprofile /tmp/hue_database_dump.json

   vi /tmp/hue_database_dump.json
   ```

   ```json
   ```

3. **Connect to New Database**
   a. Go to **Hue > Configuration**.
   b. Filter by category, **Database**.
   c. Set the following database parameters:
      - **Hue Database Type**: MySQL
      - **Hue Database Hostname**: FQDN of host running MySQL server
      - **Hue Database Port**: 3306, 5432, or 1521
      - **Hue Database Username**: username
      - **Hue Database Password**: password
      - **Hue Database Name**: Hue database name or SID
d. Click Save Changes.

4. [migration only] Synchronize New Database
   a. Select Actions > Synchronize Database
   b. Click Synchronize Database.

5. [migration only] Load Data from Old Database
   a. Log on to the host of the MySQL server in a command-line terminal.
      
      ```
      mysql -u root -p
      Enter password: <root password>
      ```
   b. Drop the foreign key constraint from the auth_permission table in the hue database.
      
      ```
      SHOW CREATE table hue.auth_permission;
      ALTER TABLE hue.auth_permission DROP FOREIGN KEY content_type_id_refs_id_id value;
      ```
   c. Clean the table, django_content_type.
      
      ```
      DELETE FROM hue.django_content_type;
      ```
   d. In Cloudera Manager, load the JSON file: select Actions > Load Database and click Load Database.
   e. Add the foreign key back:
      
      ```
      ALTER TABLE hue.auth_permission ADD FOREIGN KEY (content_type_id) REFERENCES django_content_type (id);
      ```

6. Start Hue service
   a. Navigate to Cluster > Hue, if not already there.
   b. Select Actions > Start.
   c. Click Start.
   d. Click Hue Web UI to log on to Hue with a custom MySQL database.

Connect Hue to PostgreSQL

If you have an external database installed, review PostgreSQL Troubleshooting on page 49 before creating a database for Hue.

Install and Configure PostgreSQL Server

Refer to the PostgreSQL documentation for more help on how to install a PostgreSQL database.

PostgreSQL Troubleshooting

Pay close attention to these areas and revisit when troubleshooting:

- **Python**: Some Linux distributions need python-psycopg2 (for PostgreSQL). See the community thread.
- **Security**: Delete anonymous users because they are able to log on without a password.
Remote connections: The listen address should be set to 0.0.0.0 so it can listen to multiple hosts.

Authentication: Configure `pg_hba.conf` as follows (and change database/user as appropriate):

<table>
<thead>
<tr>
<th>#</th>
<th>TYPE</th>
<th>DATABASE</th>
<th>USER</th>
<th>CIDR-ADDRESS</th>
<th>METHOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>local</td>
<td>all</td>
<td>all</td>
<td>all</td>
<td>127.0.0.1/32</td>
<td>trust</td>
</tr>
<tr>
<td>host</td>
<td>all</td>
<td>all</td>
<td>all</td>
<td>::1/128</td>
<td>password</td>
</tr>
<tr>
<td>host</td>
<td>hue_d</td>
<td>hue_u</td>
<td>all</td>
<td>0.0.0.0/0</td>
<td>md5</td>
</tr>
</tbody>
</table>

Schemas: For private schemas, configure Django with the schema owner to DROP objects.

Install PostgreSQL Server

1. Install and initialize the PostgreSQL server. The table lists the max version of each supported distribution for this CDH release, and corresponding default database database versions.

Table 6: Install Commands

<table>
<thead>
<tr>
<th>OS</th>
<th>OS Ver</th>
<th>DB Ver</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS / RHEL</td>
<td>7.3</td>
<td>9.2</td>
<td><code>sudo yum install postgresql-server</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>sudo postgresql-setup initdb</code></td>
</tr>
<tr>
<td></td>
<td>6.8</td>
<td>8.4</td>
<td><code>sudo yum install postgresql-server</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>sudo service postgresql initdb</code></td>
</tr>
<tr>
<td>SLES</td>
<td>12.1, 12.2</td>
<td>9.4</td>
<td><code>zypper install postgresql</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>postgresql-server</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>systemctl start postgresql</code></td>
</tr>
<tr>
<td></td>
<td>11.4</td>
<td>8.4</td>
<td><code># Refresh repo for python-psycopg2</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>server:database:postgresql.repo</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>zypper refresh</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>zypper install postgresql</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>postgresql-server</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>rcpostgresql start</code></td>
</tr>
<tr>
<td>Ubuntu</td>
<td>16.04</td>
<td>9.5</td>
<td><code>sudo apt-get install postgresql</code></td>
</tr>
<tr>
<td></td>
<td>14.04</td>
<td>9.3</td>
<td><code>sudo apt-get install postgresql</code></td>
</tr>
<tr>
<td></td>
<td>12.04</td>
<td>9.1</td>
<td><code>sudo apt-get install postgresql</code></td>
</tr>
<tr>
<td>Debian</td>
<td>8.4</td>
<td>9.4</td>
<td><code>sudo apt-get install postgresql</code></td>
</tr>
<tr>
<td></td>
<td>7.8</td>
<td>9.1</td>
<td><code>sudo apt-get install postgresql</code></td>
</tr>
</tbody>
</table>

Tip: If you need to start over, you can reinitialize:

```
rm -rf /var/lib/pgsql/*
<reinitialize per your os>
```
Configure PostgreSQL Server

1. Configure `pg_hba.conf` to set authentication methods:

```
# TYPE  DATABASE    USER        CIDR-ADDRESS          METHOD
local   all         all
host    all         all         127.0.0.1/32          password  # IPv4
host    all         all         ::1/128               password  # IPv6
host    hue_d       hue_u       0.0.0.0/0             md5
```

- **CentOS/RHEL/SLES**:
  ```bash
  vi /var/lib/pgsql/data/pg_hba.conf
  ```

- **Ubuntu/Debian**:
  ```bash
  vi /etc/postgresql/<pgres version>/main/pg_hba.conf
  ```

2. Configure `postgresql.conf` to listen to all available addresses:

```
listen_addresses = '0.0.0.0'
```

- **CentOS/RHEL/SLES**:
  ```bash
  vi /var/lib/pgsql/data/postgresql.conf
  ```

- **Ubuntu/Debian**:
  ```bash
  vi /etc/postgresql/<version>/main/postgresql.conf
  ```

3. Start (or restart) the database and enable automatic start on boot if necessary.
### Table 7: Restart Commands

<table>
<thead>
<tr>
<th>OS</th>
<th>OS Ver</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS / RHEL</td>
<td>7.3</td>
<td><code>sudo systemctl restart postgresql</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>sudo systemctl enable postgresql</code></td>
</tr>
<tr>
<td></td>
<td>6.8</td>
<td><code>sudo service postgresql restart</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>sudo chkconfig postgresql on</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>sudo chkconfig postgresql --list</code></td>
</tr>
<tr>
<td>SLES</td>
<td>12.1, 12.2</td>
<td><code>systemctl restart postgresql</code></td>
</tr>
<tr>
<td></td>
<td>11.4</td>
<td><code>rcpostgresql restart</code></td>
</tr>
<tr>
<td>Ubuntu</td>
<td>12.04, 14.04, 16.04</td>
<td><code>sudo /etc/init.d/postgresql restart</code></td>
</tr>
<tr>
<td>Debian</td>
<td>7.8, 8.4</td>
<td><code>sudo /etc/init.d/postgresql restart</code></td>
</tr>
</tbody>
</table>

### Create Hue Database

![Important: If you use a private schema, you must configure Django to use the schema owner (which can be a user or group) to DROP objects, because DROP is not a grantable permission in PostgreSQL.

1. Create the `hue_d` database and grant privileges to the `hue_u` user:

   ```
   sudo -u postgres psql
   postgres=# create database hue_d with lc_collate='en_US.UTF-8';
   CREATE DATABASE
   postgres=# create user hue_u with password 'huepassword';
   CREATE ROLE
   postgres=# grant all privileges on database hue_d to hue_u;
   GRANT
   ```

   ![Note: You can name the Hue database and user anything you like.]

2. Verify the connection to the `hue_d` database.

   ```
   psql -h localhost -U hue_u -d hue_d
   Password for user hue_u:
   hue=> \q
   ```

   ![Note: If you cannot connect, try typing the command manually. The hyphens may become corrupted when copied.]

### Connect Hue Service to PostgreSQL

**Tip:** To save the data in your current database (embedded or external), you must migrate (dump, synch, load) before connecting to the new database. Otherwise, skip those steps.

1. **Stop Hue Service**
In Cloudera Manager, navigate to **Cluster > Hue**.

a. Select **Actions > Stop**.

Note: If necessary, refresh the page to ensure the Hue service is stopped.

2. [migration only] **Dump Current Database**

   a. Select **Actions > Dump Database**.
   
   b. Click **Dump Database**. The file is written to /tmp/hue_database_dump.json on the host of the Hue server.
   
   c. Log on to the **host of the Hue server** in a command-line terminal.
   
   d. Edit /tmp/hue_database_dump.json by removing all objects with useradmin.userprofile in the **model** field. For example:

   ```
   # Count number of objects
   grep -c useradmin.userprofile /tmp/hue_database_dump.json
   
   vi /tmp/hue_database_dump.json
   
   { 
     "pk": 1,
     "model": "useradmin.userprofile",
     "fields": { 
       "last_activity": "2016-10-03T10:06:13",
       "creation_method": "HUE",
       "first_login": false,
       "user": 1,
       "home_directory": "/user/admin"
     }
   },
   { 
     "pk": 2,
     "model": "useradmin.userprofile",
     "fields": { 
       "last_activity": "2016-10-03T10:27:10",
       "creation_method": "HUE",
       "first_login": false,
       "user": 2,
       "home_directory": "/user/alice"
     }
   },
   ```

3. **Connect to New Database**

   a. Go to **Hue > Configuration**.
   
   b. Filter by category, **Database**.
   
   c. Set the following database parameters:

   ```
   DB Hostname  = <fqdn of host with postgres server>:5432
   DB Type      = <PostgreSQL>
   DB Name      = hue_d
   Username     = hue_u
   Password     = <hue database password set when granting hue permissions>
   ```
   
   d. Click **Save Changes**.

4. [migration only] **Synchronize New Database**

   a. Select **Actions > Synchronize Database**
   
   b. Click **Synchronize Database**.

5. [migration only] **Load Data from Old Database**
a. Log on to the *host of the PostgreSQL server* in a command-line terminal.

```
psql -h localhost -U hue_u -d hue_d
Password for user hue_u: <hue user password>
```

b. Drop the foreign key constraint from the `auth_permission` table in the hue database.

```
hue=# \d auth_permission;
hue=# ALTER TABLE auth_permission DROP CONSTRAINT content_type_id_refs_id_id value;
```

c. Clean the table, `django_content_type`.

```
hue=# TRUNCATE django_content_type CASCADE;
```

d. In Cloudera Manager, load the JSON file: select *Actions > Load Database* and click *Load Database*.  
Tip: If you are blocked by a duplicate key value such as this:

```
django.db.utils.IntegrityError: Problem installing fixture '/tmp/hue_database_dump.json': Could not load desktop.DocumentTag(pk=1): duplicate key value violates unique constraint "desktop_documenttag_owner_id_1d5f76680ee9998b_uniq" 
DETAIL: Key (owner_id, tag)=(1100713, default) already exists.
```

Delete that value and try loading again, for example:

```
DELETE FROM desktop_documenttag WHERE owner_id = '1100713' and tag = 'default';
```

e. Add the foreign key back (still logged on to the Hue database):

```
ALTER TABLE auth_permission ADD FOREIGN KEY (content_type_id) REFERENCES django_content_type (id);
```

6. Start Hue service

   a. Navigate to *Cluster > Hue*, if not already there.
   b. Select *Actions > Start*.
   c. Click *Start*.
   d. Click *Hue Web UI* to log on to Hue with a custom PostgreSQL database.

**Connect Hue to Oracle with Client Parcel**

To connect to an Oracle database, Hue needs Oracle client libraries (Basic and SDK). These are available from Oracle as packages (zip files) or from Cloudera as a parcel (for CDH parcel deployments).

This page covers connecting with the Oracle client parcel.
Important: Currently, Cloudera only provides a parcel for the Oracle 11 client (which works with the Oracle 12 server). For the Oracle 12 client package (which can be used for either CDH parcel or package deployments), see Connect Hue to Oracle with Client Package on page 62.

Install and Configure Oracle Server

Refer to the Oracle documentation for help on how to install an Oracle database.

Tip: Daniel Westermann has a helpful blog post: a simple script to automate the oracle 12c setup.

Set Environment Variables

1. Set all necessary Oracle environment variables. For example:

```bash
# Example Environment Variables
VERSION=12.1.0.2
ORACLE_HOSTNAME=<your hostname>
ORACLE_BASE=/ora01/app/oracle/product/base
ORACLE_HOME=${ORACLE_BASE}/${VERSION}
ORACLE_SID=orcl
ORACLE_OWNER_BIN=/home/oracle/bin
LD_LIBRARY_PATH=${ORACLE_HOME}/lib:${LD_LIBRARY_PATH}
```

2. Ensure that your shell .profile resembles:

```bash
# Example from /home/oracle/.bash_profile
TMP=/tmp
ORACLE_HOSTNAME=<your hostname>
ORACLE_BASE=/ora01/app/oracle/product/base
ORACLE_HOME=/ora01/app/oracle/product/base/12.1.0.2
ORACLE_SID=orcl
ORACLE_OWNER_BIN=/home/oracle/bin
LD_LIBRARY_PATH=${ORACLE_HOME}/lib:${LD_LIBRARY_PATH}
PATH=${ORACLE_HOME}/bin:${ORACLE_OWNER_BIN}:${PATH}
CLASSPATH=${ORACLE_HOME}/jlib:${ORACLE_HOME}/rdbms/jlib;
export ORACLE_HOSTNAME ORACLE_BASE ORACLE_HOME ORACLE_SID LD_LIBRARY_PATH PATH CLASSPATH TMP
```

Configure Character Set

1. Log on as the oracle user:

```bash
su - oracle
```

2. Start the listener control (as user oracle):

```bash
$ORACLE_HOME/bin/lsnrctl start
```

3. Log on to SQL*Plus:

```bash
sqlplus / as sysdba
```

4. Ensure character set is AL32UTF8 and national character set is UTF8:

```sql
SELECT * FROM v$sessions WHERE SESSION_USER = 'ADMIN'
```

To update, quit the shell and run these commands in a SQL*Plus script:

```bash
vi alter_charset.ddl
```

```bash
# Save in alter_charset.ddl (script takes 2-3 minutes)
CONNECT / as sysdba
SHUTDOWN immediate
```
STARTUP mount
ALTER SYSTEM ENABLE RESTRICTED_SESSION;
ALTER SYSTEM SET JOB_QUEUE_PROCESSES=0 SCOPE = MEMORY;
ALTER SYSTEM SET AQ_TM_PROCESSES=0 SCOPE = MEMORY;
ALTER DATABASE OPEN;
ALTER DATABASE CHARACTER SET AL32UTF8;
ALTER DATABASE NATIONAL CHARACTER SET INTERNAL_USE UTF8;
SHUTDOWN immediate
STARTUP

sqlplus /nolog < alter_charset.ddl

Create Hue Database

1. Create the hue schema, set quotas, and grant select permissions (do not grant all):
   Tip: Oracle 12 users must ALTER session set to avoid creating a common user with prefix, c##.

vi create_hue_database.ddl

## Save in create_hue_database.ddl
## Change huepassword to something more secure
CONNECT / as sysdba
ALTER session set "_ORACLE_SCRIPT"=true;
DROP user hue cascade;
CREATE user hue identified by huepassword;
ALTER user hue quota 1000m on users;
ALTER user hue quota 100m on system;
GRANT create sequence to hue;
GRANT create session to hue;
GRANT create table to hue;
GRANT create view to hue;
GRANT create procedure to hue;
GRANT create trigger to hue;
GRANT execute on sys.dbms_crypto to hue;
GRANT execute on sys.dbms_lob to hue;

sqlplus /nolog < create_hue_database.ddl

2. Verify that you can connect to hue:

sqlplus hue/<your hue password>

3. Clean all hue user tables. Create a script to spool delete statements into a new file, delete_from_tables.ddl:

vi spool_statements.ddl

## Save in spool_statements.ddl (which generates delete_from_tables.ddl)
spool delete_from_tables.ddl
set pagesize 100;
SELECT 'DELETE FROM ' || table_name || ';'; FROM user_tables;
commit;
spool off
quit

## Create delete_from_tables.ddl
sqlplus hue/<your hue password> < spool_statements.ddl

## Run delete_from_tables.ddl
sqlplus hue/<your hue password> < delete_from_tables.ddl
Create Oracle Client Parcel Repository

Cloudera provides the Oracle Instant Client for Hue (11.2 only) as a parcel for CDH parcel deployments.

**Important:** The Oracle 11 client works with the Oracle 12 server, but if you prefer the Oracle 12 client, see [Connect Hue to Oracle with Client Package](#) on page 62.

**Important:** There is currently no parcel support for Ubuntu 16 (xenial).
Oracle Instant Client for Hue

The Oracle Instant Client parcel for Hue enables Hue to be quickly and seamlessly deployed by Cloudera Manager with Oracle as its external database. For customers who have standardized on Oracle, this eliminates extra steps in installing or moving a Hue deployment on Oracle and allows for automated deployment of Hue on Oracle via the Cloudera Manager API.

Use of this software requires acceptance of the Cloudera Redistribution License Agreement for Oracle Instant Client. Please review the documentation for more information.

Download and Stage Oracle Instant Client Parcel

2. Select your OS and click Get It Now!
3. Check the box to accept Cloudera's Standard Licence Agreement and click Submit.
4. Download the parcel: ORACLE_INSTANT_CLIENT-11.2-1.oracleinstantclient1.0.0.p0.130-<your linux distro>.parcel.
5. Download the manifest for the mirrored repository.
6. Upload the parcel and manifest to the host with Cloudera Manager server, for example:

   scp ORACLE_INSTANT_CLIENT-11.2-1* manifest.json root@<Cloudera Manager server hostname>:

Install Asynchronous I/O Library

1. Log on to the host of Cloudera Manager server.
2. Install the Asynchronous I/O library, libaio/libaio1:

   ## CentOS/RHEL (yum), SLES (zypper), Ubuntu/Debian (apt-get)
   sudo yum install -y libaio
   #sudo zypper install -y libaio
   #sudo apt-get install -y libaio1

Create Mirrored Parcel Repository

When manually adding parcels it is best to use mirrored repository as it preserves the metadata that enforces relation constraints.

1. Create a temporary repository, for example:

   mkdir -pm 755 /var/www/html/cdh511
   mv ~/ORACLE_INSTANT_CLIENT-11.2-1* ~/manifest.json /var/www/html/cdh511

2. Start a web server with any available port, for example:

   cd /var/www/html/cdh511/
   python -m SimpleHTTPServer 8900
3. Test the repository in a browser:

   http://<server hostname>:8900/

   Directory listing for /

   • manifest.json
   • ORACLE_INSTANT_CLIENT-11.2-1.oracleinstantclient1.0.0.p0.130-el7.parcel

[Optional]

In fact, the Oracle parcel does not have any constraints, but using a repository allows you to more easily connect to an Oracle database during a new CDH installation if necessary. It is also a best practice and not more work.

However, if you have an existing CDH installation, you can simply copy the parcel (in this case) and add a corresponding SHA-1 file to /opt/cloudera/parcel-repo.

You must have CDH installed because the directory, parcel-repo, is created during step 6 of a CDH parcel installation.

```
sha1sum ORACLE_INSTANT_CLIENT-11.2-1.oracleinstantclient1.0.0.p0.130-<your linux distro>.parcel | awk '{ print $1 }' > ORACLE_INSTANT_CLIENT-11.2-1.oracleinstantclient1.0.0.p0.130-<your linux distro>.parcel.sha1
mv ORACLE_INSTANT_CLIENT* /opt/cloudera/parcel-repo/
```

Connect Hue Service to Oracle

You can connect Hue to your Oracle database while installing CDH (and Hue) or with an existing installation. With existing CDH installations, you can connect and restart Hue, without saving the data in your current database, or you can migrate the old data into Oracle.

New CDH Installation

See [Cloudera Installation Guide](http://<cm server hostname>:7180) to install Cloudera Manager (and its Installation Wizard), which you will use here to install CDH and the Oracle client.

**Install CDH and Oracle Parcel**

1. Open the Cloudera Manager Admin Console and run the [Cloudera Manager Installation Wizard](http://<cm server hostname>:7180) to install CDH (and Hue). The URL for Cloudera Manager is: http://<cm server hostname>:7180

2. Stop at **Select Repository** to add the Oracle client parcel repository (**Cluster Installation**, step 1):
   a. Choose Method **Use Parcels** and click **More Options**.
   b. and add the URL for your Oracle **Remote Parcel Repository**:

   ![Remote Parcel Repository](https://archive.cloudera.com/cdh5/parcels/latest_supported/)

   ![Remote Parcel Repository](http://<test1-cent73-cdh510-ord11-1.gce.cloudera.com:8900/)

   c. Click **Save Changes**.
   d. Select the newly added radio button by **ORACLE_INSTANT_CLIENT** and click **Continue**.
The Oracle parcel is downloaded, distributed, and activated at Cluster Installation, step 6 (Installing Selected Parcels).

### Cluster Installation

*Installing Selected Parcels*

The selected parcels are being downloaded and installed on all the hosts in the cluster.

<table>
<thead>
<tr>
<th>Parcel</th>
<th>Downloaded</th>
<th>Distributed</th>
<th>Unpacked</th>
<th>Actuated</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDH 5.10.0-1.cdh5.10.0.143</td>
<td>100%</td>
<td>4/4 (2.4 MB/s)</td>
<td>4/4</td>
<td>4/4</td>
</tr>
<tr>
<td>ORACLE_INSTANT_CLIENT-11.2-1.oraclenetworkclient-1.0.0.0.130</td>
<td>100%</td>
<td>4/4 (54 MB/s)</td>
<td>4/4</td>
<td>4/4</td>
</tr>
</tbody>
</table>

---

### Connect Hue to Oracle

*Continuing with Cloudera Manager Installation Wizard …*

1. Stop at Database Setup to set connection properties (Cluster Setup, step 3).
   a. Select Use Custom Database.
   b. Under Hue, set the connection properties to the Oracle database.

   **Note:** Copy and store the password for the Hue embedded database (just in case).

   
   Database Hostname (and port): <fqdn of host with Oracle server>:1521
   Database Type (or engine): Oracle
   Database SID (or name): orcl
   Database Username: hue
   Database Password: <hue database password>

   c. Click Test Connection and click Continue when successful.

2. Continue with the installation and click Finish to complete.
3. Add support for a multi-threaded environment:
   a. Go to Clusters > Hue > Configuration.
b. Filter by Category, Hue-service and Scope, Advanced.
c. Add support for a multi-threaded environment by setting Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini:

```plaintext
desktop
[database]
options={"threaded":true}
```
d. Click Save Changes.

4. Restart the Hue service: select Actions > Restart and click Restart.
5. Log on to Hue by clicking Hue Web UI.

Existing CDH Installation

Activate Oracle Client Parcel

1. Log on to Cloudera Manager.
2. Go to the Parcels page by clicking Hosts > Parcels (or clicking the parcels icon 🌐).
3. Click the Configuration > Check for New Parcels.
4. Find ORACLE_INSTANT_CLIENT and click Download, Distribute, and Activate.

Connect Hue to Oracle

If you are not migrating the current (or old) database, simply connect to your new Oracle database and restart Hue. See Connect Hue to an External Database.

1. [migration only] Stop Hue Service
   a. In Cloudera Manager, navigate to Cluster > Hue.
   b. Select Actions > Stop.

   Note: If necessary, refresh the page to ensure the Hue service is stopped.

2. [migration only] Dump Current Database
   a. Select Actions > Dump Database.
   b. Click Dump Database. The file is written to /tmp/hue_database_dump.json on the host of the Hue server.
   c. Log on to the host of the Hue server in a command-line terminal.
   d. Edit /tmp/hue_database_dump.json by removing all objects with useradmin.userprofile in the model field. For example:

   ```bash
   # Count number of objects
   grep -c useradmin.userprofile /tmp/hue_database_dump.json
   
   vi /tmp/hue_database_dump.json
   ```
   ```json
   {"pk": 1,
   "model": "useradmin.userprofile",
   "fields": {
   "last_activity": "2016-10-03T10:06:13",
   "creation_method": "HUE",
   "first_login": false,
   "user": 1,
   "home_directory": "/user/admin"
   }}
   ```
3. Connect to New Database
   a. Configure Database connections:
      • Go to Hue > Configuration and filter by category, Database.
      • Set database properties and click Save Changes:

      Hue Database Type (or engine): Oracle
      Hue Database Hostname: <fqdn of host with Oracle server>
      Hue Database Port: 1521
      Hue Database Username: hue
      Hue Database Password: <hue database password>
      Hue Database Name (or SID): orcl

   b. Add support for a multi-threaded environment:
      • Filter by Category, Hue-service and Scope, Advanced.
      • Set Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini and click Save Changes:

      [desktop]
      [[database]]
      options={"threaded":true}

4. [migration only] Synchronize New Database
   a. Select Actions > Synchronize Database
   b. Click Synchronize Database.

5. [migration only] Load Data from Old Database
   Important: All user tables in the Hue database must be empty. You cleaned them at step 3 on page 64 of Create Hue Database on page 64. Ensure they are still clean.

   sqlplus hue/<your hue password> < delete_from_tables.ddl

6. Re/Start Hue service
   a. Navigate to Cluster > Hue.
   b. Select Actions > Start, and click Start.
   c. Click Hue Web UI to log on to Hue with a custom Oracle database.

Connect Hue to Oracle with Client Package

To connect to an Oracle database, Hue needs Oracle client libraries (Basic and SDK). These are available from Oracle as packages (zip files) or from Cloudera as a parcel (for CDH parcel deployments).

This page covers connecting with Oracle client packages.
Install and Configure Oracle Server

Refer to the Oracle documentation for help on how to install an Oracle database.

Tip: Daniel Westermann has a helpful blog post: a simple script to automate the oracle 12c setup.

Set Environment Variables

1. Set all necessary Oracle environment variables. For example:

```bash
## Example Environment Variables
VERSION=12.1.0.2
ORACLE_HOSTNAME=<your hostname>
ORACLE_BASE=/ora01/app/oracle/product/base
ORACLE_HOME=${ORACLE_BASE}/${VERSION}
ORACLE_SID=orcl
ORAOWNER_BIN=/home/oracle/bin
LD_LIBRARY_PATH=${ORACLE_HOME}/lib:${LD_LIBRARY_PATH}
```

2. Ensure that your shell .profile resembles:

```bash
## Example from /home/oracle/.bash_profile
TMP=/tmp
ORACLE_HOSTNAME=<your hostname>
ORACLE_BASE=/ora01/app/oracle/product/base
ORACLE_HOME=/ora01/app/oracle/product/base/12.1.0.2
ORACLE_SID=orcl
ORAOWNER_BIN=/home/oracle/bin
LD_LIBRARY_PATH=${ORACLE_HOME}/lib:${LD_LIBRARY_PATH}
PATH=${ORACLE_HOME}/bin:${ORAOWNER_BIN}:${PATH}
CLASSPATH=${ORACLE_HOME}/jlib:${ORACLE_HOME}/rdbms/jlib;
export ORACLE_HOSTNAME ORACLE_BASE ORACLE_HOME ORACLE_SID LD_LIBRARY_PATH PATH CLASSPATH TMP
```

Configure Character Set

1. Log on as the oracle user:

```bash
su - oracle
```

2. Start the listener control (as user oracle):

```bash
$ORACLE_HOME/bin/lsnrctl start
```

3. Log on to SQL*Plus:

```bash
sqlplus / as sysdba
```

4. Ensure character set is AL32UTF8 and national character set is UTF8:

```sql
SELECT * FROM v$nls_parameters where parameter like '%CHARACTERSET';
```

To update, quit the shell and run these commands in a SQL*Plus script:

```bash
vi alter_charset.ddl
```

```bash
## Save in alter_charset.ddl (script takes 2-3 minutes)
CONNECT / as sysdba
SHUTDOWM immediate
STARTUP mount
ALTER SYSTEM ENABLE RESTRICTED SESSION;
ALTER SYSTEM SET JOB_QUEUE_PROCESSES=0 SCOPE = MEMORY;
ALTER SYSTEM SET AQ_TM_PROCESSES=0 SCOPE = MEMORY;
ALTER DATABASE OPEN;
ALTER DATABASE CHARACTER SET AL32UTF8;
ALTER DATABASE NATIONAL CHARACTER SET INTERNAL_USE UTF8;
```
create hue database

1. Create the hue schema, set quotas, and grant select permissions (do not grant all):
   
   Tip: Oracle 12 users must ALTER session set to avoid creating a common user with prefix, c##.

   vi create_hue_database.ddl

   ## Save in create_hue_database.ddl
   ## Change huepassword to something more secure
   ALTER session set "_ORACLE_SCRIPT"=true;

   DROP user hue cascade;
   CREATE user hue identified by huepassword;
   ALTER user hue quota 1000m on users;
   ALTER user hue quota 100m on system;
   GRANT create sequence to hue;
   GRANT create session to hue;
   GRANT create table to hue;
   GRANT create view to hue;
   GRANT create procedure to hue;
   GRANT create trigger to hue;
   GRANT execute on sys.dbms_crypto to hue;
   GRANT execute on sys.dbms_lob to hue;

   sqlplus /nolog < create_hue_database.ddl

2. Verify that you can connect to hue:

   sqlplus hue/<your hue password>

3. Clean all hue user tables. Create a script to spool delete statements into a new file, delete_from_tables.ddl:

   vi spool_statements.ddl

   ## Save in spool_statements.ddl (which generates delete_from_tables.ddl)
   spool delete_from_tables.ddl
   set pagesize 100;
   SELECT 'DELETE FROM ' || table_name || ';' FROM user_tables;
   commit;
   spool off
   quit

   ## Create delete_from_tables.ddl
   sqlplus hue/<your hue password> < spool_statements.ddl

   ## Run delete_from_tables.ddl
   sqlplus hue/<your hue password> < delete_from_tables.ddl
Install Oracle Client Package

Cloudera Manager requires the Oracle instant client libraries to be in /usr/share/oracle/instantclient/lib/. The following commands arrange the files as such.

⚠️ Important: You must add client libraries to each machine that hosts a Hue server.

Install Asynchronous I/O Library

1. Log on to the host of Cloudera Manager server.
2. Install the Asynchronous I/O library, libaio/libaio1:

```bash
# CentOS/RHEL (yum), SLES (zypper), Ubuntu/Debian (apt-get)
sudo yum install -y libaio
#sudo zypper install -y libaio
#sudo apt-get install -y libaio1
```

Install Oracle Client

1. Download zip files for Instant Client Package, Basic and SDK (with headers).
2. For this step, switch to the host with the downloaded files and upload zip to the Cloudera Manager server host:

```bash
cp instantclient-*.zip root@<CM server hostname>:
```
3. Arrange the client libraries to mirror the tree structure in the image. Here is one way to do this:

```bash
# Create nested directories: /usr/share/oracle/instantclient/lib/
mkdir -pm 755 /usr/share/oracle/instantclient/lib

# Unzip. The files expand into /usr/share/oracle/instantclient/instantclient_<ver>/
unzip '*.zip' -d /usr/share/oracle/instantclient/

# Move lib files from instantclient_<ver> to /usr/share/oracle/instantclient/lib/
mv /usr/share/oracle/instantclient/`ls -l /usr/share/oracle/instantclient/ | grep instantclient_ | awk '{print $9}'`/lib* /usr/share/oracle/instantclient/lib/

# Move rest of the files to /usr/share/oracle/instantclient/
mv /usr/share/oracle/instantclient/`ls -l /usr/share/oracle/instantclient/ | grep instantclient_ | awk '{print $9}'`/* /usr/share/oracle/instantclient/

# Create symbolic links. Remember to edit version numbers as necessary

```
```

4. Set $ORACLE_HOME and $LD_LIBRARY_PATH:

```bash
export ORACLE_HOME=/usr/share/oracle/instantclient
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ORACLE_HOME
```

**Note:** If using the Oracle 11 instant client you are ready to Connect Hue to Oracle. Else if using the Oracle 12 instant client, upgrade the Python module, `cx_Oracle`.

Apply Temporary Workaround for Oracle 12 Client

Update the `cx_Oracle` package in your native Python environment and copy it to Hue's Python environment.

1. Install `gcc` and Python development tools:

```bash
# CentOS/RHEL (yum), SLES (zypper), Ubuntu/Debian (apt-get)
yum install -y python-setuptools python-devel gcc
#zypper install -y python-setuptools python-devel gcc
#apt-get install -y python-setuptools python-dev gcc
```
2. Install pip:

```bash
easy_install pip
```

3. Install `cx_Oracle`. Ensure that `ORACLE_HOME` and `$LD_LIBRARY_PATH` are properly set so that `pip` knows which version to install.

```bash
echo $ORACLE_HOME $LD_LIBRARY_PATH
```
```
pip install cx_Oracle
```

**Tip:** You can also `wget` the proper `cx_Oracle` file yourself: [https://pypi.python.org/pypi/cx_Oracle/](https://pypi.python.org/pypi/cx_Oracle/).

4. Get the version of the new `cx_Oracle` package:
   - **CentOS/RHEL and SLES:**
     ```bash
     ls /usr/lib64/python2.7/site-packages/cx_Oracle*
     ```
   - **Ubuntu/Debian:**
     ```bash
     ls /usr/local/lib/python2.7/dist-packages/cx_Oracle*
     ```

5. If this is a **New CDH Installation** on page 68, stop here to run the first 5 or 6 steps of the Cloudera Manager Installation Wizard (packages=5, parcels=6). Do not go past **Cluster Installation**.

   - **CDH Parcel installation:**
     ```bash
     cd /opt/cloudera/parcels/`ls -l /opt/cloudera/parcels | grep CDH | tail -1 | awk '{print $9}'`/lib/hue/build/env/lib/python2.7/site-packages
     ```
   - **CDH package installation:**
     ```bash
     cd /usr/lib/hue/build/env/lib/python2.7/site-packages
     ```

   **Important:** The parcel path is created during step 5 or 6 of **Cluster Installation**, so you must have completed this to continue.

7. Move the existing `cx_Oracle` file:

```bash
mv cx_Oracle-5.2.1-py2.7-linux-x86_64.egg cxfoo
```

8. Copy the new `cx_Oracle` module to Hue's python environment. The version can change:
   - **CentOS/RHEL and SLES:**
     ```bash
     cp -a /usr/lib64/python2.7/site-packages/cx_Oracle-5.3-py2.7.egg-info
     ```
   - **Ubuntu/Debian**
     ```bash
     cp -a /usr/local/lib/python2.7/dist-packages/cx_Oracle-5.3.egg-info
     ```
**Connect Hue Service to Oracle**

You can connect Hue to your Oracle database while installing CDH (and Hue) or with an existing installation. With existing CDH installations, you can connect and restart Hue, without saving the data in your current database, or you can migrate the old data into Oracle.

**New CDH Installation**

See [Cloudera Installation Guide](#) to install Cloudera Manager (and its Installation Wizard), which you will use here to install CDH and the Oracle client.

1. Open the Cloudera Manager Admin Console and run the Cloudera Manager Installation Wizard to install CDH (and Hue). The URL for Cloudera Manager is: http://<cm server hostname>:7180
2. Stop at the end of Cluster Installation to copy the latest cx_Oracle package into Hue’s Python environment.
3. Stop at Database Setup to set connection properties (Cluster Setup, step 3).
   a. Select Use Custom Database.
   b. Under Hue, set the connection properties to the Oracle database.

   **Note:** Copy and store the password for the Hue embedded database (just in case).

   - Database Hostname (and port): <fqdn of host with Oracle server>:1521
   - Database Type (or engine): Oracle
   - Database SID (or name): orcl
   - Database Username: hue
   - Database Password: <hue database password>

c. Click Test Connection and click Continue when successful.
4. Continue with the installation and click Finish to complete.
5. Add support for a multi-threaded environment:
   a. Go to Clusters > Hue > Configuration.
   b. Filter by Category, Hue-service and Scope, Advanced.
   c. Add support for a multi-threaded environment by setting Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini:

   ```ini
   [desktop]
   [[database]]
   options={"threaded":true}
   ```

d. Click Save Changes.
6. Restart the Hue service: select **Actions > Restart** and click **Restart**.

7. Log on to Hue by clicking **Hue Web UI**.

**Existing CDH Installation**

If you are not migrating the current (or old) database, simply connect to your new Oracle database and restart Hue. See [Connect Hue to an External Database](#).

1. [migration only] **Stop Hue Service**
   a. In Cloudera Manager, navigate to **Cluster > Hue**.
   b. Select **Actions > Stop**.

   ![Note] If necessary, refresh the page to ensure the Hue service is stopped:

2. [migration only] **Dump Current Database**
   a. Select **Actions > Dump Database**.
   b. Click **Dump Database**. The file is written to `/tmp/hue_database_dump.json` on the host of the Hue server.
   c. Log on to the **host of the Hue server** in a command-line terminal.
   d. Edit `/tmp/hue_database_dump.json` by removing all objects with `useradmin.userprofile` in the `model` field. For example:

   ```bash
   # Count number of objects
   grep -c useradmin.userprofile /tmp/hue_database_dump.json
   vi /tmp/hue_database_dump.json
   ```

3. **Connect to New Database**
   a. Configure Database connections: Go to **Hue > Configuration**, filter by **Database**, set properties, and click **Save Changes**:

   ```
   Hue Database Type (or engine): Oracle
   Hue Database Hostname: <fqdn of host with Oracle server>
   Hue Database Port: 1521
   Hue Database Username: hue
   Hue Database Password: <hue database password>
   Hue Database Name (or SID): orcl
   ```
b. Add support for a multi-threaded environment: Filter by Hue-service, set Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini, and click Save Changes:

```python
[desktop]
[[database]]
options={"threaded":true}
```

4. [migration only] Synchronize New Database
   a. Select Actions > Synchronize Database
   b. Click Synchronize Database.

5. [migration only] Load Data from Old Database

   Important: All user tables in the Hue database must be empty. You cleaned them at step 3 on page 64 of Create Hue Database on page 64. Ensure they are still clean.

   ```sql
   @sqlplus hue/<hue_password> < delete_from_tables.ddl
   ```

6. Re/Start Hue service
   a. Navigate to Cluster > Hue.
   b. Select Actions > Start, and click Start.
   c. Click Hue Web UI to log on to Hue with a custom Oracle database.

Hue Custom Database Tutorial

This page explains how to configure Hue with a custom database from end to end by migrating your existing database and synching to a new custom database. Learn how to switch databases for:

- A new installation of CDH, with the Cloudera Manager Installation Wizard
- An existing installation of CDH, with the Cloudera Manager Admin Console.

Note: On this page we use CentOS 6 with MySQL. For instructions on other platforms and databases, see Hue Databases.

Prepare Hosts

Create, or prepare, five machines, each with CentOS 6 and at least 8 GB of RAM:

2. Create one machine for the database. Name it cdh-db.<your domain>.com.

Separating the database from the CDH cluster is a best practice, but if necessary, you can install it on one of the hosts in the cluster (for example, cdh-cluster-1).

Install Custom Database

Install MySQL on the single machine you designated for this purpose (cdh-db.<your domain>.com).

1. Install MySQL server on cdh-db.<your domain>.com:
   ```bash
   sudo yum install -y mysql-server
   ```
2. Start the server:
   ```bash
   sudo service mysqld start
   ```
3. Secure your installation:

```bash
sudo /usr/bin/mysql_secure_installation
```

Enter current password for root (enter for none): [Press Enter if the password is unset]

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

4. Configure /etc/my.cnf:

```ini
[mysqld]
...
bind-address=0.0.0.0
default-storage-engine=innodb
sql_mode=STRICT_ALL_TABLES
```

5. Restart the server

```bash
sudo service mysqld restart
```

6. Log on with your new root password:

```bash
mysql -u root -p<root password>
```

7. Create the hue database with UTF8 collation and configure the hue user (with your own password):

```sql
create database hue collate = 'utf8_general_ci';
grant all on hue.* to 'hue'@'%' identified by 'huepassword';
quit
```

Install CM and CDH

In this section, we test connecting to a custom database with the installation wizard; then we undo the connection so we can connect with the admin console in Dump, Synchronize, and Load on page 71.

When you run the Cloudera Manager Installation Wizard, stop at the Database Setup page.

See Cloudera Installation Guide.

Populate Database (optional)

Populate the Hue database with user account information, a Hive query, and an Oozie workflow (to ensure that the database migration works).

Dump, Synchronize, and Load

To connect to other supported databases, see Hue Custom Databases.

1. Stop the Hue service: go to Hue and select Actions > Stop.

Note: Refresh the page if the Hue service does not look stopped: 📞
2. Dump the existing database:
   a. Select Actions > Dump Database.
   b. Click Dump Database. The file is written to /tmp/hue_database_dump.json on the host of the Hue server.
   c. Log on to the host of the Hue server in a command-line terminal.
   d. Edit /tmp/hue_database_dump.json by removing all objects with useradmin.userprofile in the model field. For example:

   ```
   # Count number of objects
   grep -c useradmin.userprofile /tmp/hue_database_dump.json

   vi /tmp/hue_database_dump.json
   ```

3. Connect Hue to the new MySQL database:
   a. Go to Hue > Configuration.
   b. Filter by category, Database.
   c. Set the following database parameters:

   ```
   DB Hostname = <fqdn of host with postgres server>:3306
   DB Type = <PostgreSQL>
   DB Name = hue
   Username = hue
   Password = <hue database password set when granting hue permissions>
   ```
   d. Click Save Changes.

4. Synchronize the new database: select Actions > Synchronize Database and click Synchronize Database.

5. Load the database after removing the foreign key constraint:
   a. Log on to the host of the MySQL server in a command-line terminal.
   b. Delete the foreign key constraint and clean the table, django_content_type:

   ```
   mysql -u root -p
   SHOW CREATE table hue.auth_permission;
   ALTER TABLE hue.auth_permission DROP FOREIGN KEY content_type_id_refs_id_<input id>;
   DELETE FROM hue.django_content_type;
   ```
c. In Cloudera Manager, load the JSON file: select **Actions > Load Database** and click **Load Database**.

d. Add the foreign key back:

```sql
ALTER TABLE hue.auth_permission ADD FOREIGN KEY (content_type_id) REFERENCES django_content_type (id);
```

6. Start the Hue service: select **Actions > Start** and click **Start**. If you went through **Use Hue**, ensure your data was migrated properly.

---

**Migrate the Hue Database**

![Hue Custom Databases](image)

Note: **Hue Custom Databases** includes database-specific pages on how to migrate from an old to a new database. This page summarizes across supported database types.

When you change Hue databases, you *can* migrate the existing data to your new database. If the data is dispensable, there is no need to migrate.

The Hue database stores things like user accounts, SQL queries, and Oozie workflows, and you may have accounts, queries, and workflows worth saving. See **How to Populate the Hue Database** on page 77.

Migrating your existing database currently requires some work-arounds (in parentheses):

- Stop the Hue service.
- Dump database (and delete "useradmin.userprofile" objects from .json file).
- Connect to new database.
- Synchronize database (and drop foreign key to clean tables).
- Load database (and add foreign key).
- Start Hue service.

**Dump Database**

1. In the **Hue Web UI**, click the home icon to see what documents you are migrating.
2. In Cloudera Manager, stop the Hue service: go to **Hue** and select **Actions > Stop**.

   ![Note](image)

   Note: Refresh the page to ensure that the Hue service is stopped.

3. Select **Actions > Dump Database** and click **Dump Database**. The file is written to /tmp/hue_database_dump.json on the host of the Hue server.
4. Log on to the host of the **Hue server** in a command-line terminal. You can find the hostname on the Dump Database window and at **Hue > Hosts**.
5. Edit /tmp/hue_database_dump.json by removing all objects with useradmin.userprofile in the model field. For example:

```
# Count number of objects
grep -c useradmin.userprofile /tmp/hue_database_dump.json
```

```
vi /tmp/hue_database_dump.json
```

```
|
{  
  "pk": 1,  
  "model": "useradmin.userprofile",  
  "fields": {   
    "last_activity": "2016-10-03T10:06:13",  
    "creation_method": "HUE",  
    "first_login": false,  
    "user": 1,  
    "home_directory": "/user/admin"  
  }  
},
```

**Connect New Database**

In Cloudera Manager, connect Hue to the new database. See [Hue Custom Databases](#) for help on installing and configuring a custom database.

1. Go to **Hue > Configuration**.
2. Filter by category, **Database**.
3. Set the appropriate database parameters:

   - Hue Database Type: MySQL or PostgreSQL or Oracle
   - Hue Database Hostname: `<fqdn of host with database server>`
   - Hue Database Port: 3306 or 5432 or 1521
   - Hue Database Username: `<hue database username>`
   - Hue Database Password: `<hue database password>`
   - Hue Database Name: `<hue database name or SID>`

4. Click **Save Changes**.

5. **Oracle users only** should add support for a multithreaded environment:

   - a. Filter by Category, **Hue-service** and Scope, **Advanced**.
   - b. Add support for a multithreaded environment by setting **Hue Service Advanced Configuration Snippet (Safety Valve)** for `hue_safety_valve.ini`:

```
[desktop]
    [[[database]]]
        options={"threaded":True}
```

   - c. Click **Save Changes**.

**Synchronize and Load**

1. **Synchronize**: select **Actions > Synchronize Database** and click **Synchronize Database**.
2. Log on to the host of the **database server** in a command-line terminal and clean tables:
   - MySQL and PostgreSQL on page 76 users remove a foreign key from `auth.permission` and clean `django_content_type`.
   - Oracle on page 76 users delete content from all tables.
3. **Load**: select **Actions > Load Database** and click **Load Database**.
4. Return to the host of the database server:
MySQL and PostgreSQL on page 76 users add the foreign key to `auth_permission`.

5. Start: select Actions > Start and click Start.

   ![Note: Refresh the page to ensure that the Hue service is running.]

6. In the Hue Web UI, click the home icon 🏡 to ensure that all documents were migrated.

**MariaDB / MySQL**

1. Synchronize Database in Cloudera Manager.
2. Log on to MySQL:

   ```
   mysql -u root -p
   Enter password: <root password>
   ```

3. Drop the foreign key constraint from the `hue.auth_permission` table:

   - Execute the following statement to find the `content_type_id` in the CONSTRAINT clause of the CREATE TABLE statement for the `hue.auth_permission` table:

   ```
   SHOW CREATE TABLE hue.auth_permission;
   ```

   This SHOW CREATE TABLE statement produces output similar to the following:

   ```
   | auth_permission | CREATE TABLE 'auth_permission' ( 
   |                | 'id' int(11) NOT NULL AUTO_INCREMENT, 
   |                | 'name' varchar(50) NOT NULL, 
   |                | 'content_type_id' int(11) NOT NULL, 
   |                | 'CODENAME' VARCHAR(100) NOT NULL, 
   |                | PRIMARY KEY ('id'), 
   |                | UNIQUE KEY 'content_type_id' ('content_type_id', 'codename'), 
   |                | KEY 'auth_permission_37ef4eb4' ('content_type_id'), 
   |                | CONSTRAINT 'content_type_id_refs_id_d043b34a' FOREIGN KEY ('content_type_id') 
   |                | REFERENCES 'django_content_type' ('id') 
   |                | ) ENGINE=InnoDB AUTO_INCREMENT=229 DEFAULT CHARSET=utf8 |
   ```

   - Then execute the following statement to drop the foreign key constraint:

   ```
   ALTER TABLE hue.auth_permission DROP FOREIGN KEY 
   content_type_id_refs_id_<value>;
   ```

   For example, if you used the above output from the SHOW CREATE TABLE statement, you would use the following ALTER TABLE statement:

   ```
   ALTER TABLE hue.auth_permission DROP FOREIGN KEY 
   content_type_id_refs_id_d043b34a;
   ```

4. Delete the contents of `django_content_type`:

   ```
   DELETE FROM hue.django_content_type;
   ```
5. Load Database in Cloudera Manager.
6. Add the foreign key, content_type_id, to auth_permission:

```
ALTER TABLE hue.auth_permission ADD FOREIGN KEY (content_type_id) REFERENCES
django_content_type (id);
```

7. Start Hue in Cloudera Manager.

**PostgreSQL**

1. Synchronize Database in Cloudera Manager.
2. Log on to PostgreSQL:

```
psql -h localhost -U hue -d hue
Password for user hue:
```

3. Drop the foreign key constraint from auth_permission:

```
\d auth_permission;
ALTER TABLE auth_permission DROP CONSTRAINT content_type_id_refs_id_<id value>;
```

4. Delete the contents of django_content_type:

```
TRUNCATE django_content_type CASCADE;
```

5. Load Database in Cloudera Manager.
6. Add the foreign key, content_type_id, to auth_permission:

```
ALTER TABLE auth_permission ADD FOREIGN KEY (content_type_id) REFERENCES
django_content_type(id) DEFERRABLE INITIALLY DEFERRED;
```

7. Start Hue in Cloudera Manager.

**Oracle**

Oracle users should delete all content from the Oracle tables after synchronizing and before loading:

1. Synchronize Database in Cloudera Manager.
2. Log on to Oracle:

```
su - oracle
sqlplus / as sysdba
```

3. Grant a quota to the tablespace where tables are created (the default is SYSTEM). For example:

```
ALTER USER hue quota 100m on system;
```

4. Log on as the hue:

```
sqlplus hue/<hue password>
```
5. Create a spool script that creates a delete script to clean the content of all tables.

```sql
vi spool_statements.ddl

## Save in spool_statements.ddl (which generates delete_from_tables.ddl)
spool delete_from_tables.ddl
set pagesize 100;
SELECT 'DELETE FROM ' || table_name || ';' FROM user_tables;
commit;
spool off
quit
```

6. Run both scripts:

```sql
## Create delete_from_tables.ddl
sqlplus hue/<your hue password> < spool_statements.ddl
## Run delete_from_tables.ddl
sqlplus hue/<your hue password> < delete_from_tables.ddl
```

7. Load Database in Cloudera Manager.

8. Start Hue in Cloudera Manager.

How to Populate the Hue Database

Not every action in the Hue UI touches the Hue database (embedded or custom). This page explains how to populate the database with user account information, Hive queries, and Oozie workflows. This is useful when testing the migration of a database.

1. Add New User
   a. Log on to Hue as the administrator.
   b. Click the Administration drop-down menu and select Manage Users.
   c. In the Hue Users page, click Add user and specify the following:
      - Type the Username.
      - Type the user’s password in the New Password text box.
      - Re-type the user’s password.
      - Select the Create home directory check box if you want a home directory created for the user.
      - Click Next.
      - Specify the user’s First name, Last name, Email address, and the Groups that the user is a member of.
      - Click Next.
      - Specify whether the user is Active and if you want to grant the user Superuser status.
      - Click Add user.
   d. Log out as the administrator and log in as the new user you just created.

2. Save Hive Query
   a. In the top right corner of the page, select <user_name> > Hue Administration > Step 2: Examples:
b. Click download Hive to install sample databases.

c. In the left panel, click the default database and the customers sample table.

d. Click Query > Editor > Hive to load the Hive SQL editor.

e. Drag the customers table onto the Hive editor and select SELECT FROM ... to automatically generate a SELECT query in the editor.

f. Run the query by placing your cursor in the editor and pressing CTRL + Enter, or by clicking the Run icon ▶.

G. Save the query as customers.sql by clicking the Save icon  

h. View the query on the Saved Queries tab in the Hive editor:

3. Save Oozie Workflow

a. Go to Oozie by selecting, Query > Scheduler > Workflow.
b. Click the Edit icon in the upper right corner of the Oozie Editor page:

![Oozie Editor interface]

<table>
<thead>
<tr>
<th>Tables</th>
<th>Drop your action here</th>
</tr>
</thead>
<tbody>
<tr>
<td>customers</td>
<td>id (id)</td>
</tr>
</tbody>
</table>

You can add a description in the same way if you want to.

c. Click My Workflow to rename it and click the check mark to save it. In this example, we name it Customers Workflow.

d. Drag the action icon for a saved Hive query to the field Drop your action here.

e. Select a saved query from the drop-down list and click Add. In this example, we choose customers.sql.

f. Save the workflow by clicking the Save icon at the top of the page.

g. Submit the workflow by clicking the Run icon and then clicking Submit. You should see the workflow progress bar change to green, which indicates the workflow ran successfully.

h. View the saved workflow by clicking the Workflows tab at the top of the page.
Tuning Hue Performance

This section contains the following topics on Hue performance tuning and high availability:

How to Add a Hue Load Balancer

1. Log on to Cloudera Manager and click Hue.
2. Select Actions > Add Role Instances.
3. Add 1 Load Balancer:
   a. Click Select hosts in the field under Load Balancer.
   b. Select a host and click OK.
4. [Optional] Add 2 additional Hue servers (for a total of 3) to boost performance:
   a. Click Select hosts in the field under Hue Server.
   b. Select a host and click OK > Continue.
5. Check the boxes for the new servers and load balancer.

Note: Hue servers can share hosts with Load Balancers. But Hue servers must be on distinct hosts from other Hue servers, and Load Balancers must be on distinct hosts from other Load Balancers.

7. Click Save Changes and Restart Hue.
8. Click Hue Web UI > Load Balanced Hue Web UI.
9. Log on to Hue and ensure the port is 8889.
   Tip: The Load Balancer instance can always be accessed on the Hue Instances tab.

Configure Hue for High Availability

Configuring Hue for High Availability (HA) means configuring Hue, Hive, and Impala.

Configure Hue for High Availability

Prerequisites

- SSH network access to host machines with an Hue Server/Kerberos Ticket Renewer role.
- External database configured for each Hue Server. See Hue Databases.

Add Hue Roles

Hue HA requires at least two Hue server roles and one Load Balancer role. If the cluster is authenticating with Kerberos, you need one Kerberos Ticket Renewer on each host with a Hue Server.

1. Log on to Cloudera Manager and go to the Hue service.
2. Go to the Hue service and select Actions > Add Role Instances.
3. Click Hue Server, assign to one or more hosts, and click OK > Continue.
4. Click Kerberos Ticket Renewer, assign to each host with a Hue Server, and click OK > Continue.
5. Click Load Balancer, assign to one or more hosts, and click OK > Continue.
6. Check each role and select Actions for Selected > Start and click Start.
Enable TLS for Hue Load Balancer

**Note:** You can configure the Load Balancer for TLS/SSL or each endpoint (H2S, Impalad).

1. Go to Hue > Configuration and search on TLS/SSL.
2. Check Enable TLS/SSL for Hue for the Hue Server Default Group.
3. Set other TLS/SSL properties appropriate for your setup. Some to consider are:
   - **Hue Load Balancer Port** - Apache Load Balancer listens on this port (default is 8889).
   - **Path to TLS/SSL Certificate File** - Must be multi-domain with CN = Load Balancer in PEM format.
   - **Path to TLS/SSL Private Key File** - Must be in PEM format.
4. Click Save Changes and Restart Hue.

Configure Hive and Impala for High Availability

**Note:** Hive must have two or more HS2 roles, and Impala two or more ImpalaD roles.

Prerequisites & Requirements

- **SSH network access** to host machines with a HiveServer2 or Impala Daemon role.
- **External database** configured for each H2S and Impala Daemon.
- **Hue Load Balancer** Hive/Impala Load Balancer configured with Source IP Persistence.

Source IP Persistence

Without IP Persistence, you may encounter the error, "Results have expired, rerun the query if needed.

Hue supports High Availability through a "load balancer" to HiveServer2 and Impala. Because the underlying Hue thrift libraries reuse TCP connections in a pool, a single user session may not have the same TCP connection. If a TCP connection is balanced away from a HiveServer2 or Impalad instance, the user session and its queries (running or returned) can be lost and trigger the "Results have expired" error.

To prevent sessions from being lost, configure the Hive/Impala Load Balancer with **Source IP Persistence** so that each Hue instance sends all traffic to a single HiveServer2/Impala instance. Of course, this is not true load balancing, but a configuration for failover High Availability.

To prevent sessions from timing out while in use, **add more Hue Server instances**, so that each can be pinned to another HiveServer2/Impalad instance. And for both HiveServer2/Impala, **set the affinity timeout** (that is, the timeout to close persisted sessions) to be longer than the impala query and session timeouts.

For the best load distribution, **create multiple profiles** in your load balancer, per port, for both non-Hue clients and Hue clients. Have non-Hue clients distribute loads in a round robin and configure Hue clients with source IP Persistence on dedicated ports, for example, 21000 for impala-shell, 21050 for impala-jdbc, and 21051 for Hue.

Add Hive and Impala Roles

In Cloudera Manager, add roles for HiveServer2 and Impala Daemon (like **Add Hue Roles** on page 80):

1. Configure the cluster with at least two roles for HiveServer2:
   a. Go to the **Hive** service and select Actions > Add Role Instances.
   b. Click **HiveServer2**, assign one or more hosts, and click OK > Continue.
   c. Check each role and select Actions for Selected > Start and click Start.

2. Configure the cluster with at least two roles for Impala Daemon:
   a. Go to the **Impala** service and select Actions > Add Role Instances.
Install Proxy Service

This is an example of how to add a proxy server for each HiveServer2 and Impala Daemon with multiple profiles.

1. Install haproxy (for either RHEL / Ubuntu / SLES):

   ```
   yum install haproxy
   ```

   ```
   apt-get install haproxy
   ```

   ```
   zypper addrepo
   zypper refresh
   zypper install haproxy
   ```

2. Configure haproxy for each role, for example:

   ```
   vi /etc/haproxy/haproxy.cfg
   ```

   ```
   listen impala-shell
   bind :21001
   mode tcp
   option tcplog
   balance roundrobin
   stick-table type ip size 20k expire 5m
   server impala_0 shortname-2.domain:21000 check
   server impala_1 shortname-3.domain:21000 check
   
   listen impala-jdbc
   bind :21051
   mode tcp
   option tcplog
   balance roundrobin
   stick-table type ip size 20k expire 5m
   server impala_0 shortname-2.domain:21050 check
   server impala_1 shortname-3.domain:21050 check
   
   listen impala-hue
   bind :21052
   mode tcp
   option tcplog
   balance source
   server impala_0 shortname-2.domain:21050 check
   server impala_1 shortname-3.domain:21050 check
   
   listen hiveserver2-jdbc
   bind :10001
   mode tcp
   option tcplog
   balance roundrobin
   stick-table type ip size 20k expire 5m
   server hiveserver2_0 shortname-1.domain:10000 check
   server hiveserver2_1 shortname-2.domain:10000 check
   
   listen hiveserver2-hue
   bind :10002
   mode http
   option tcplog
   balance source
   server hiveserver2_0 host shortname-1.domain:10000 check
   server hiveserver2_1 host shortname-2.domain:10000 check
   ```
Replace shortname-#.domain with those in your environment:

```bash
sed -i "s/host shortname/your host shortname/g" /etc/haproxy/haproxy.cfg
sed -i "s/domain/your domain/g" /etc/haproxy/haproxy.cfg
```

3. Restart haproxy:

```bash
service haproxy restart
```

4. Run `netstat` to ensure your proxies are running:

```bash
netstat | grep LISTEN
```

5. For information about using HUE with the configured load balancer for either Impala or Hive, see one of the following references:

- Using Impala through a Proxy for High Availability
- Configuring HiveServer2 High Availability in CDH

Hue and HDFS High Availability

You can use Cloudera Manager to configure Hue to use HDFS high availability NameNodes.

Configuring Hue to Work with HDFS HA Using Cloudera Manager

1. Add the HttpFS role.
2. After the command has completed, go to the Hue service.
3. Click the Configuration tab.
4. Locate the HDFS Web Interface Role property or search for it by typing its name in the Search box.
5. Select the HttpFS role you just created instead of the NameNode role, and save your changes.
6. Restart the Hue service.
Hue Security

The following security topics are included in this Hue Guide:

In addition, the following Hue security documents are located in other parts of the Cloudera documentation set:

- **Hue Authentication**
- **Configuring TLS/SSL for Hue**
- **Configuring Other CDH Components to Use HDFS HA**

Hue User Permissions

Hue is a gateway to CDH cluster services (see Table 8: Hue Applications on page 85) and both have completely separate permissions. Being a Hue superuser means nothing to HDFS, Hive, and so on.

**Important:** Hue and the underlying cluster services have completely separate permissions!

Users who log on to the Hue UI must have permission to use Hue, *and also,* each CDH service accessible within Hue.

A common configuration is for **Hue users** to be authenticated with an LDAP server and **CDH users** with Kerberos. These users can differ. For example, CDH services do not authenticate each user who logs on to Hue. Rather, they authenticate **Hue** and trust that Hue has authenticated *its* users.

Once Hue is authenticated by a service (for example, Hive), **Hue impersonates the user** (*doAs*) requesting use of that service (for example, to create a table). The service uses **Sentry** (essentially a `chmod` tool) to ensure the group to which that user belongs is authorized for that action.

Hue user permissions are at the application level only. For example, a Hue superuser can filter Hue user access to a CDH service but cannot authorize the use of its features. Again, Sentry does that. Learn how to **Authorize Hue User Groups with Sentry** on page 101.

Hue Superusers

The Hue User Admin application provides two levels of privileges: users and superusers.

**Users** can change their name, email address, and password. They can login to Hue and run Hue applications according to their group permissions.

**Superusers** can perform administrative functions such as:

- Add and delete users and groups
- Import and sync users and groups from an LDAP server
- Assign group permissions
- Promote users to superusers and vice versa.

Hue superusers have no special privileges to the underlying CDH cluster services. Sentry is used to add those privileges.

**Important:** The first user to log on to Hue (without LDAP authentication) becomes the first superuser.

How to Assign Superuser Status to an LDAP User

In a non-secure cluster, the first user to log on to Hue is designated a superuser. In a secure cluster with LDAP, there are three ways to assign superuser status:

1. **Important:** Hue and the underlying cluster services have completely separate permissions!
1. With the `AllowAllBackend` temporarily enabled, assign superuser status and synchronize one user.
2. With the `LdapBackend` enabled, run a Hue shell command to apply superuser status.
3. Enable multiple backends so that the first user to log on still works when integrated with LDAP.

Hue Applications and Permissions

Hue is a gateway to (and web-based UI for) the following CDH cluster services.

**Hue Applications**

These CDH services are available in Hue. Currently, Spark is only available upstream.

**Table 8: Hue Applications**

<table>
<thead>
<tr>
<th>Hue App</th>
<th>Sentry</th>
<th>App Dependencies</th>
</tr>
</thead>
<tbody>
<tr>
<td>HBase</td>
<td></td>
<td>HBase Browser</td>
</tr>
<tr>
<td>HDFS</td>
<td></td>
<td>Core, File Browser</td>
</tr>
<tr>
<td>Hive</td>
<td></td>
<td>Metastore Tables, Hive Editor</td>
</tr>
<tr>
<td>Impala</td>
<td>✔️</td>
<td>Metastore Tables, Impala Editor</td>
</tr>
<tr>
<td>MapRed / YARN</td>
<td></td>
<td>Job Browser, Job Designer, Oozie, Hive Editor, Pig, Sqoop</td>
</tr>
<tr>
<td>Oozie</td>
<td></td>
<td>Job Designer, Oozie Editor/Dashboard</td>
</tr>
<tr>
<td>Pig</td>
<td></td>
<td>Pig Editor, Oozie</td>
</tr>
<tr>
<td>Sentry</td>
<td></td>
<td>Solr Search</td>
</tr>
<tr>
<td>Solr (Search)</td>
<td>✔️</td>
<td>Hadoop Security</td>
</tr>
<tr>
<td>Spark</td>
<td></td>
<td>Spark</td>
</tr>
<tr>
<td>Sqoop 2</td>
<td></td>
<td>Sqoop Transfer</td>
</tr>
</tbody>
</table>

**Hue Permissions**

Hue application permissions are composed of `name.permission:action`.

For example, `filebrowser.access:Launch this application(3)`:

- Hue application name = `filebrowser`
- Permissions = `access` (as in, execute)
- Action = Launch this application (the HDFS filebrowser).
- Process ID in Hue database (3).

**Table 9: Hue Application Permissions**

<table>
<thead>
<tr>
<th>Hue App</th>
<th>Permission</th>
<th>rwx</th>
<th>Action Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>about</td>
<td>access</td>
<td>-x</td>
<td>Launch this application</td>
</tr>
<tr>
<td>beeswax</td>
<td>access</td>
<td>-x</td>
<td>Launch this application</td>
</tr>
<tr>
<td>dashboard</td>
<td>access</td>
<td>-x</td>
<td>Launch this application</td>
</tr>
<tr>
<td>filebrowser</td>
<td>access</td>
<td>-x</td>
<td>Launch this application</td>
</tr>
<tr>
<td>filebrowser</td>
<td>s3_access</td>
<td>-x</td>
<td>Access to S3 from filebrowser and filepicker</td>
</tr>
<tr>
<td>help</td>
<td>access</td>
<td>-x</td>
<td>Launch this application</td>
</tr>
<tr>
<td>impala</td>
<td>access</td>
<td>-x</td>
<td>Launch this application</td>
</tr>
</tbody>
</table>
Secure Hue Passwords with Scripts

Hue lets you secure passwords in one consolidated script, or multiple individual scripts. Hue runs each password script at startup and extracts passwords from stdout.

Store scripts in a directory that only Hue can read, write, and execute. You can choose password script names but you cannot change hue.ini property names to which you assign those scripts.

1. At the command line, create one or more password scripts. For example, create a consolidated script named my_passwords_script.sh:

```bash
#!/bin/bash
SERVICE=$1
if [[ $(SERVICE) == "ldap_password" ]]
    then
    echo "your_ldap_password"
    fi
if [[ $(SERVICE) == "ssl_password" ]]
    then
    echo "your_ssl_password"
    fi
```

Hue Security

<table>
<thead>
<tr>
<th>Hue App</th>
<th>Permission</th>
<th>rwX</th>
<th>Action Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>indexer</td>
<td>access</td>
<td>--x</td>
<td>Launch this application</td>
</tr>
<tr>
<td>jobbrowser</td>
<td>access</td>
<td>--x</td>
<td>Launch this application</td>
</tr>
<tr>
<td>jobsub</td>
<td>access</td>
<td>--x</td>
<td>Launch this application</td>
</tr>
<tr>
<td>metadata</td>
<td>access</td>
<td>--x</td>
<td>Launch this application</td>
</tr>
<tr>
<td>metadata</td>
<td>write</td>
<td>--w-</td>
<td>Allow edition of metadata like tags</td>
</tr>
<tr>
<td>metastore</td>
<td>access</td>
<td>--x</td>
<td>Launch this application</td>
</tr>
<tr>
<td>metastore</td>
<td>write</td>
<td>--w-</td>
<td>Allow DDL operations. Need the app access too</td>
</tr>
<tr>
<td>notebook</td>
<td>access</td>
<td>--x</td>
<td>Launch this application</td>
</tr>
<tr>
<td>oozie</td>
<td>access</td>
<td>--x</td>
<td>Launch this application</td>
</tr>
<tr>
<td>oozie</td>
<td>dashboard_jobs_access</td>
<td>--x</td>
<td>Oozie Dashboard read-only user for all jobs</td>
</tr>
<tr>
<td>oozie</td>
<td>disable_editor_access</td>
<td>--x</td>
<td>Disable Oozie Editor access</td>
</tr>
<tr>
<td>pig</td>
<td>access</td>
<td>--x</td>
<td>Launch this application</td>
</tr>
<tr>
<td>proxy</td>
<td>access</td>
<td>--x</td>
<td>Launch this application</td>
</tr>
<tr>
<td>rdbms</td>
<td>access</td>
<td>--x</td>
<td>Launch this application</td>
</tr>
<tr>
<td>search</td>
<td>access</td>
<td>--x</td>
<td>Launch this application</td>
</tr>
<tr>
<td>security</td>
<td>access</td>
<td>--x</td>
<td>Launch this application</td>
</tr>
<tr>
<td>security</td>
<td>impersonate</td>
<td></td>
<td>Let a user impersonate another user when listing objects like files or tables</td>
</tr>
<tr>
<td>sqoop</td>
<td>access</td>
<td>--x</td>
<td>Launch this application</td>
</tr>
<tr>
<td>useradmin</td>
<td>access_view:useradmin:edit_user</td>
<td>rwx</td>
<td>Access to profile page on User Admin</td>
</tr>
<tr>
<td>useradmin</td>
<td>access_view:useradmin:view_user</td>
<td>rwx</td>
<td>Access to any profile page on User Admin</td>
</tr>
<tr>
<td>useradmin</td>
<td>access</td>
<td>--x</td>
<td>Launch this application</td>
</tr>
</tbody>
</table>
fi
if [[ ${SERVICE} == "bind_password" ]]
then
echo "your_bind_password"
fi

if [[ ${SERVICE} == "db_password" ]]
then
echo "your_database_password"
fi

2. Log on to Cloudera Manager and go to **Hue > Configuration**.
3. Search on **Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini**: 
4. Add script properties, for example:

```sh
[desktop]
ldap_username=hueservice
ldap_password_script="/var/lib/hue/password_script.sh ldap_password"
ssl_password_script="/var/lib/hue/password_script.sh ssl_password"

[[ldap]]
bind_password_script="/var/lib/hue/password_script.sh bind_password"

[[database]]
db_password_script="/var/lib/hue/password_script.sh db_password"
```

5. Click **Save Changes** and **Restart Hue**.

**Authenticate Hue Users with LDAP**

Configuring Hue for Lightweight Directory Access Protocol (LDAP) lets you **import** users and groups from a directory service, **synchronize** group membership manually or at automatically login, and **authenticate** with LDAP.

This page explains how to configure Hue for LDAP authentication. To import users and group from LDAP, see **Synchronize Hue with LDAP Server** on page 94.

**Authenticate Hue Users and Groups with LDAP**

Hue supports **Active Directory** (AD) and open standard LDAP such as **OpenLDAP** and **OpenDJ**.

There are two ways to bind Hue with an LDAP directory service:

- **Search Bind**: Hue searches for user credentials with search base (and attribute and filter).
- **Direct Bind**: Hue authenticates (without searching) in one of two ways:
  - **NT Domain**: Bind to Microsoft Active Directory with username@domain (the UPN)
  - **Username Pattern**: Bind to open standard LDAP with full path of directory information tree (DIT).

**Note**: Username pattern does not work with AD because AD inserts spaces into the UID which Hue cannot process.

**Encryption**: To prevent credentials from transmitting in the clear, encrypt with LDAP over SSL, using the LDAPS protocol on the LDAPS port (636 by default); or encrypt with the **StartTLS** extension using the standard LDAP protocol and port (389 by default). Cloudera recommends LDAPS. You must have a CA Certificate in either case.

**Table 10: Hue Supported LDAP Authentication and Encryption Methods**

<table>
<thead>
<tr>
<th>LDAP Auth Action</th>
<th>Encrypted (LDAPS)</th>
<th>Encrypted (LDAPS+TLS)</th>
<th>Not Encrypted (LDAP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search Bind</td>
<td>AD, LDAP</td>
<td>AD, LDAP</td>
<td>AD, LDAP</td>
</tr>
</tbody>
</table>
Hue Security

<table>
<thead>
<tr>
<th>LDAP Auth Action</th>
<th>Encrypted (LDAPS)</th>
<th>Encrypted (LDAP+TLS)</th>
<th>Not Encrypted (LDAP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct Bind - NT Domain</td>
<td>AD</td>
<td>AD</td>
<td>AD</td>
</tr>
<tr>
<td>Direct Bind - User Pattern</td>
<td>LDAP</td>
<td>LDAP</td>
<td>LDAP</td>
</tr>
</tbody>
</table>

**Prerequisites**

To authenticate Hue with LDAP, you must have:

- LDAP server
- Bind account (or support for anonymous binds)
- Cloudera Manager account with Full Administrator permissions
- [optional] LDAP server with LDAPS or StartTLS encryption.

**Important:** To authenticate securely, configure your LDAP server with either LDAP over SSL (LDAPS) or StartTLS encryption. Both methods require a Certificate Authority (CA) chain in a .pem file.

**Search Bind**

Search bind authentication does an ldapsearch against one or more directory services and binds with the found distinguished name (DN) and password. Hue searches the subtree from the base distinguished name. If LDAP Username Attribute is set, Hue looks for an entry whose attribute has the same value as the short name given at login.

**Important:** Search binding works with all directory service types. It is also the only method that allows synchronizing groups at login (set with sync_groups_on_login in a safety-valve).

**Video:** Authenticate Hue with LDAP and Search Bind

1. Log on to Cloudera Manager and click Hue.
2. Click the Configuration tab and filter by scope=Service-wide and category=Security.
3. Set the following required properties:

<table>
<thead>
<tr>
<th>Authentication Backend</th>
<th>desktop.auth.backend.LdapBackend</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDAP URL</td>
<td>ldaps://&lt;ldap_server&gt;:636 (or ldap://&lt;ldap_server&gt;:389)</td>
</tr>
<tr>
<td>LDAP Server CA Certificate</td>
<td>/path_to_certificate/cert.pem</td>
</tr>
<tr>
<td>LDAP Search Base</td>
<td>DC=mycompany,DC=com</td>
</tr>
<tr>
<td>LDAP Bind User Distinguished Name</td>
<td>username@domain</td>
</tr>
<tr>
<td>LDAP Bind Password</td>
<td>bind_user_password</td>
</tr>
<tr>
<td>Use Search Bind Authentication</td>
<td>TRUE</td>
</tr>
<tr>
<td>Enable LDAP TLS</td>
<td>FALSE if using LDAPS or not encrypting</td>
</tr>
<tr>
<td>Create LDAP users on login</td>
<td>TRUE</td>
</tr>
</tbody>
</table>

**Note:** To encrypt with TLS, set LDAP URL to ldap://<ldap_server>:389 and check Enable LDAP TLS. For a proof of concept without encryption, use ldaps://<ldap_server>:389, remove the value for LDAP Server CA Certificate, and uncheck Enable LDAP TLS.

4. You can optionally improve search performance with attributes and filters.

   | LDAP User Filter                          | objectclass=user (default = *) |
### LDAP Username Attribute
- sAMAccountName (AD default), uid (LDAP default)

### LDAP Group Filter
- objectclass=group (default = *)

### LDAP Group Name Attribute
- cn (default)

### LDAP Group Membership Attribute
- member (default)

---

**Note:** With the user settings in the table above, the LDAP search filter has the form:

```
(& (objectClass=user) (sAMAccountName=<user entered username>))
```

---

5. Add any valid user and/or valid group to quickly test your LDAP configuration.

<table>
<thead>
<tr>
<th>LDAP Username Attribute for Test LDAP Configuration</th>
<th>Any valid user</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDAP Group Name Attribute for Test LDAP Configuration</td>
<td>Any valid group</td>
</tr>
</tbody>
</table>

6. Click **Save Changes**.
7. **Test your LDAP configuration**, and when successful, **Restart Hue**.

---

**Note:** The syntax of Bind Distinguished Name differs per bind method:
- Search Bind: username@domain
- Direct Bind with NT Domain: username
- Direct Bind with Username Pattern: DN string (full DIT path)

Do not use if anonymous binding is supported.

---

```
## You can test ldapsearch at the command line as follows:
LDAP_TLS_CACERT=/path_to_cert/ca_certificate:
ldapsearch -H ldaps://ldap_server:636 -D "<bind_dn>" -w <bind_password> -b <base_dn> "samaccountname=<user>
```

**Note:** To run `ldapsearch` with a CA certificate, you may need to install `ldap_utils` on Debian/Ubuntu and `openldap-clients` on RHEL/CentOS.

---

**Direct Bind**

To authenticate with direct binding, Hue needs either the User Principal Name (UPN) for Active Directory, or the full path to the LDAP user in the Directory Information Tree (DIT) for open standard LDAP.

**Important:** Direct binding only works with one domain. For multiple directories, use **Search Bind** on page 88.

---

Video: [Authenticate Hue with LDAP and Direct Bind](#)

---

To directly bind to an Active Directory/LDAP server with NT domain:

1. Log on to Cloudera Manager and click **Hue**.
2. Click the **Configuration** tab and filter by `scope=Service-wide` and `category=Security`.
3. Set LDAP properties exactly like Search Bind with these exceptions:

<table>
<thead>
<tr>
<th>Active Directory Domain</th>
<th>&lt;your NT domain&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDAP Bind User Distinguished Name</td>
<td>&lt;username only&gt; (not username@domain)</td>
</tr>
<tr>
<td>Use Search Bind Authentication</td>
<td>FALSE</td>
</tr>
</tbody>
</table>
4. Click **Save Changes**.
5. **Test your LDAP configuration**, and when successful, **Restart Hue**.

To directly bind to an open standard LDAP server with a username pattern:

1. Remove the value for Active Directory Domain.
2. Set both **LDAP Username Pattern** and **LDAP Bind User Distinguished Name** to a DN string that represents the full path of the directory information tree, from UID to top level domain.

   **Note:** When using direct bind, set **LDAP Search Base**, not for authentication (you can log on to Hue without it), but to **Synchronize Hue with LDAP Server** on page 94.

---

**Test Hue LDAP Configuration On-the-Fly**

You can test your LDAP settings **without** restarting the Hue service—simply input values and save changes.

1. Configure Hue LDAP **Search Bind** on page 88 or **Direct Bind** on page 89.
2. Add a user and group name for **Test LDAP Configuration**.
3. Click **Save Changes**.
4. Select Actions > **Test LDAP Configuration**.
5. Click **Test LDAP Configuration**.
6. **Restart Hue** when the test succeeds and log on to the Hue Web UI.
Unmanaged Clusters

Consumers with unmanaged clusters (that is, without Cloudera Manager) must manually set properties in `hue.ini`. Consumers with managed clusters must use Cloudera Manager to set properties in `hue.ini`.

Example of a Search Bind configuration encrypted with LDAPS:

```ini
[[custom]]
[[auth]]
backend=desktop.auth.backend.LdapBackend

[[ldap]]
ldap_url=ldaps://w2k8-1.ad.sec.cloudera.com:636
search_bind_authentication=true
ldap_cert=./<path_to_cacert>/w2k8-1-root.pem
use_start_tls=false
create_users_on_login=true
base_dn="DC=ad,DC=sec,DC=cloudera,DC=com"
bind_dn="<username>@ad.sec.cloudera.com"
bind_password_script=./<path_to_password_script>/<script.sh>
test_ldap_user="testuser1"
test_ldap_group="testgroup1"

[[[users]]]
user_filter="objectclass=user"
user_name_attr="sAMAccountName"

[[[groups]]]
group_filter="objectclass=group"
group_name_attr="cn"
group_member_attr="member"
```

Example of a Direct Bind configuration for Active Directory encrypted with LDAPS:

```ini
[[[ldap]]
ldap_url=ldaps://w2k8-1.ad.sec.cloudera.com:636
search_bind_authentication=false
nt_domain=ad.sec.cloudera.com
ldap_cert=./<path_to_cacert>/w2k8-1-root.pem
use_start_tls=false
create_users_on_login=true
base_dn="DC=ad,DC=sec,DC=cloudera,DC=com"
bind_dn="<username>"
```
Example of a Direct Bind configuration for Active Directory encrypted with StartTLS:

```bash
[[ldap]]
ldap_url=ldap://w2k8-1.ad.sec.cloudera.com:389
search_bind_authentication=false
nt_domain=ad.sec.cloudera.com
ldap_cert=/opt/cloudera/security/cacerts/w2k8-1-root.pem
use_start_tls=true
create_users_on_login=true
base_dn="DC=ad,DC=sec,DC=cloudera,DC=com"
bind_dn="cconnor"
bind_password_script=<path_to_password_script>/<script.sh>
```

### Table of Hue LDAP Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description and Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>General Hue LDAP Properties</strong></td>
<td></td>
</tr>
<tr>
<td>Authentication Mode. Select</td>
<td>desktop.auth.backend.LdapBackend. Multiple backends are allowed. Create a list and add it to the Hue safety-valve.</td>
</tr>
<tr>
<td>backend</td>
<td></td>
</tr>
<tr>
<td>LDAP URL</td>
<td>URL for the LDAP server.</td>
</tr>
<tr>
<td>ldap_url</td>
<td>Syntax: ldaps://&lt;ldap_server&gt;:&lt;636&gt; or ldap://&lt;ldap_server&gt;:&lt;389&gt;</td>
</tr>
<tr>
<td></td>
<td><strong>Important:</strong> To prevent usernames and passwords from transmitting in the clear, use ldaps:// or ldap:// + &quot;Enable LDAP TLS&quot;.</td>
</tr>
<tr>
<td>Create LDAP users on login</td>
<td>Flag to create new LDAP users at Hue login.</td>
</tr>
<tr>
<td>create_users_on_login</td>
<td>If true, any user who logs into Hue is automatically created. If false, only users that exist in useradmin can log in.</td>
</tr>
<tr>
<td><strong>Direct Bind Properties</strong></td>
<td></td>
</tr>
<tr>
<td>Active Directory Domain</td>
<td>For direct binding with Active Directory only. Typically maps to the user email address or ID in conjunction with the domain. Allows Hue to authenticate without having to follow LDAP references to other partitions. Hue binds with <em>User Principal Names</em> (UPNs) if provided.</td>
</tr>
<tr>
<td>nt_domain</td>
<td>Example: ad.&lt;mycompany&gt;.com. <strong>Important:</strong> Do not use nt_domain when binding with a username pattern or if using search bind.</td>
</tr>
<tr>
<td>LDAP Username Pattern</td>
<td>For direct binding with LDAP (non-Active Directory) only (because AD uses UPNs which have a space in them). Username Pattern finds the user attempting to login into LDAP by adding the username to a predefined DN string. Use <em>&lt;username&gt;</em> to reference the user logging in. An example is &quot;uid=&lt;username&gt;,ou=people,dc=mycompany,dc=com&quot;.</td>
</tr>
<tr>
<td>ldap_username_pattern</td>
<td></td>
</tr>
<tr>
<td><strong>Search Bind Properties</strong></td>
<td></td>
</tr>
<tr>
<td>Use Search Bind Authentication</td>
<td>Flag to enable/disable search binding.</td>
</tr>
<tr>
<td>Property Name</td>
<td>Description and Syntax</td>
</tr>
<tr>
<td>---------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>search_bind_authentication</td>
<td>Distinguished name to use as a search base for finding users and groups.</td>
</tr>
<tr>
<td>base_dn</td>
<td>Syntax: dc=ad, dc=sec, dc=mycompany, dc=com</td>
</tr>
<tr>
<td><strong>Encryption Properties</strong></td>
<td></td>
</tr>
<tr>
<td>LDAP Server CA Certificate</td>
<td>Full path to .pem file with Certificate Authority (CA) chain used to sign the LDAP server certificate. If left blank, all certificates are trusted and otherwise encrypted usernames and passwords are vulnerable to attack.</td>
</tr>
<tr>
<td>ldap_cert</td>
<td></td>
</tr>
<tr>
<td>use_start_tls</td>
<td>Flag to enable/disable encryption with StartTLS.</td>
</tr>
<tr>
<td><strong>Import / Sync Properties</strong></td>
<td></td>
</tr>
<tr>
<td>LDAP Bind User Distinguished Name</td>
<td>Bind user. Only use if LDAP/AD does not support anonymous binds. (Typically, LDAP supports anonymous binds and AD does not.) Bind User differs per auth type:</td>
</tr>
<tr>
<td>bind_dn</td>
<td>- Search Bind: username@domain&lt;br&gt;- Direct Bind with NT Domain: username&lt;br&gt;- Direct Bind with Username Pattern: DN string (and same as LDAP Username Pattern)</td>
</tr>
<tr>
<td>ldap_bind_password</td>
<td>Bind user password.</td>
</tr>
<tr>
<td><strong>Filter Properties</strong></td>
<td></td>
</tr>
<tr>
<td>LDAP User Filter</td>
<td>General LDAP filter to restrict search of valid users. Only used by Search Bind authentication and LDAP Sync.</td>
</tr>
<tr>
<td>user_filter</td>
<td>The default is objectclass=* but can differ. For example, some LDAP environments support Posix objects for *nix authentication and the user filter might need to be objectclass=posixAccount.</td>
</tr>
<tr>
<td>LDAP Username Attribute</td>
<td>Username to search against (the attribute in LDAP that contains the username).</td>
</tr>
<tr>
<td>user_name_attr</td>
<td>Typical attributes include sAMAccountName (default for AD/LDAP) and uid (LDAP default). Maintain case sensitivity when setting attributes for AD/LDAP.</td>
</tr>
<tr>
<td>LDAP Group Filter</td>
<td>General LDAP filter to restrict search of valid groups. Only used by LDAP Sync (not authentication). If left blank, no filtering is used and all groups in LDAP are synced. The default is objectclass=* but can differ. For example, some LDAP environments support Posix objects for *nix authentication and the user filter might need to be objectclass=posixGroup.</td>
</tr>
<tr>
<td>group_filter</td>
<td></td>
</tr>
<tr>
<td>LDAP Group Name Attribute</td>
<td>Group name to search against (the attribute in LDAP that contains the groupname).</td>
</tr>
</tbody>
</table>
Synchronize Hue with LDAP Server

Configuring Hue for Lightweight Directory Access Protocol (LDAP) lets you import users and groups from a directory service, synchronize group membership manually or at automatically login, and authenticate with LDAP.

This page explains how to import and synchronize Hue users and groups with the LDAP server. See Authenticate Hue Users with LDAP on page 87 to ensure you are configured properly.

Tip: After you import and synchronize, learn how to Restrict Group Permissions on page 96.

Synchronize Hue Users and Groups with LDAP

There are four LDAP import and sync options in Hue:

<table>
<thead>
<tr>
<th>LDAP Sync Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add/Sync LDAP user</td>
<td>Import and synchronize one user at a time</td>
</tr>
<tr>
<td>Sync LDAP users/groups</td>
<td>Synchronize user memberships in all groups</td>
</tr>
<tr>
<td>Add/Sync LDAP group</td>
<td>Import and synchronize all users in one group</td>
</tr>
<tr>
<td>sync_groups_at_login</td>
<td>Automatically synchronize group membership at login</td>
</tr>
</tbody>
</table>

Note: Hue does not support importing all groups at once.

Prerequisites

To synchronize your Hue users and groups with your LDAP server:

- Hue must be configured to authenticate with LDAP. See Authenticate Hue Users with LDAP on page 87.
- The logged in user must have Hue superuser permissions.
Users

Import and Synchronize One User

To import and synchronize one LDAP user in Hue:

1. Log on to the Hue UI as a superuser.
2. Go to User Admin > Users.
3. Click Add/Sync LDAP user.
4. Add a username, check Create home directory, and click Add/Sync user.

Synchronize All User Memberships

To synchronize group memberships (for already imported users) to the current state of the LDAP server:

1. Log on to the Hue UI as a superuser.
2. Go to User Admin > Users.
3. Click Sync LDAP users/groups.
4. Check Create home directories, and click Sync.

Groups

Import and Synchronize One Group (with one or more users)

To import and synchronize a group (and its multiple users):

1. Log on to the Hue UI as a superuser.
2. Go to User Admin > Groups.
3. Click Add/Sync LDAP group.
4. Check Create home directories, and click Sync.

Synchronize Groups (and User Membership) at Login

Note: LDAP sync_groups_at_login only works with Search Bind on page 88.

To configure Hue to automatically synchronize users at the Hue login:
Hue Security

1. Log on to Cloudera Manager and click **Hue**.
2. Click the **Configuration** tab and filter by scope=**Service-wide** and category=**Advanced**.
3. Configure **Hue Service Advanced Configuration Snippet (Safety Valve)** for hue_safety_valve.ini:

```ini
[desktop]
[ldap]
sync_groups_on_login=true
```

4. Click **Save Changes** and **Restart Hue**.

Restrict Group Permissions

You can configure user permissions on the **Groups** tab.

1. Log on to the Hue UI as a superuser.
2. Go to **User Admin > Groups**.
3. Click the name of the group you want to alter.
4. Deselect any users that you do not want to change (all users in the group are selected by default).
5. Select or deselect the permissions you want to apply or remove.
6. Click **Update Group**.

**Note:** A best practice is to remove all permissions from the default group and assign permissions as appropriate to your own groups.

Authenticate Hue Users with SAML

Hue supports SAML (Security Assertion Markup Language) for Single Sign-on (SSO) authentication.

The **SAML 2.0 Web Browser SSO** profile has three components:

- **User Agent** - Browser that represents you, the user, seeking resources.
- **Service Provider** (SP) - Service (Hue) that sends authentication requests to SAML.
- **Identity Provider** (IdP) - SAML service that authenticates users.

When a user requests access to an application, the Service Provider (Hue) sends an authentication request from the User Agent (browser) to the Identity Provider. The Identity Provider authenticates the user, sends a response, and redirects the browser back to Hue (see graphic).

This page explains how to configure Hue, the Service Provider, and gives guidance on how to configure the Identity Provider, which differs per product.

Configure Hue for SAML Authentication

The Service Provider (Hue) and the Identity Provider use a metadata file to confirm each other’s identity. Hue stores metadata from the SAML server, and the IdP stores metadata from Hue server.
Install and Configure IdP SAML Server

The instructions on this page assume that you have an Identity Provider set up and running. You can use any Identity Provider—Okta, Ping, OpenAM, and so on.

In Configure Hue at the Command Line on page 97 (step 3 on page 97), you must copy the metadata from your IdP’s SAML server and store it in an XML file on every host with a Hue server.

**Important:** Read the documentation of your Identity Provider for details on how to procure the XML of the SAML server metadata.

Configure Hue at the Command Line

**Important:** You may need to disable cipher algorithms. See SAML SSL Error on page 101 in Troubleshooting below.

1. Install the following libraries **on all hosts** in your cluster:

   ```
   ## RHEL/CentOS
   yum install git gcc python-devel swig openssl
   
   ## Ubuntu/Debian
   apt-get install git gcc python-dev swig openssl
   
   ## SLES
   zypper install git gcc python-devel swig openssl make libxml2-devel libxml2-devel
   ```

2. Install `xmlsec1` and `xmlsec1-openssl` **on all hosts** in the cluster:

   **Important:** Ensure that the `xmlsec1` package is executable by the user, `hue`.

   ```
   ## RHEL/CentOS
   yum install xmlsec1 xmlsec1-openssl
   ```

   **Note:** If `xmlsec` libraries are not available, use the appropriate `epel repository`:

   ```
   ## For RHEL/CentOS 7
   wget http://dl.fedoraproject.org/pub/epel/7/x86_64/e/epel-release-7-6.noarch.rpm
   rpm -ivh epel-release-7-6.noarch.rpm
   
   ## Ubuntu/Debian
   apt-get install xmlsec1 libxmlsec1-openssl
   
   ## SLES (get latest version)
   wget http://www.aleksey.com/xmlsec/download/xmlsec1-1.2.24.tar.gz
   tar -xvzf xmlsec1-1.2.24.tar.gz
   cd xmlsec1-1.2.24
   ./configure &&
   make
   ```

3. Copy metadata from your IdP’s SAML server and save it as an XML file **on every host with a Hue server**.
For example, if your Identity Provider is Shibboleth, visit https://<idp_host>:8443/idp/shibboleth, copy the metadata content, and paste it into an .xml file.

**Note:** You may have to edit the copied metadata; for example, the IdP’s port number (8443) may be missing from its URL.

```
mkdir -pm 755 /opt/cloudera/security/saml/
cd /opt/cloudera/security/saml/

vim idp-<your idp provider>-metadata.xml
# Paste IdP SAML here and save
```

4. Add key_file and cert_file for encrypted assertions–see Table 11: Table of SAML Parameters on page 99.

![Warning: Add key and cert files even if not encrypting assertions. Hue checks for the existence and validity of these files even if they are not needed! They cannot be empty files. This is a known issue.
If necessary, create "valid" dummy files:](image)

```
openssl genrsa -des3 -out dummy.key 2048
openssl rsa -inform PEM -outform PEM -in dummy.key -pubout -out
dummy-nopass.pem
```

Configure Hue in Cloudera Manager

Currently, all hue.ini properties for SAML must be added to Hue Service safety-valve in Cloudera Manager.

1. Log on to Cloudera Manager and go to Hue > Configuration.
2. Configure Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini with:

```
## Example Settings using Open AM:
[desktop]
redirect_whitelist="^/.*$,^http:\/\/clr.sec.cloudera.com:8080\/.*$"
[[auth]]
backend=libsaml.backend.SAML2Backend
[libsaml]
xmlsec_binary=/usr/bin/xmlsec1
metadata_file=/opt/cloudera/security/saml/idp-openam-metadata.xml
key_file=/opt/cloudera/security/saml/host.key
cert_file=/opt/cloudera/security/saml/host.pem
username_source=nameid
name_id_format="urn:oasis:names:tc:SAML:2.0:nameid-format:transient"
entity_id=<host base name>
logout_enabled=false
```
3. Click Save Changes, then select, Actions > Restart Hue.

Integrate IdP SAML Server with Hue

After Hue is configured and restarted, copy the metadata generated by Hue server and send it to your Identity Provider so they can configure the SAML server.

1. Ensure Hue is configured, restarted, and running.
3. Copy the metadata and send it to your Identity Provider.
4. Ensure that your Identity Provider configures the SAML server with the Hue metadata (just as you configured the Hue server with SAML metadata).

SAML Properties in hue.ini

Table 11: Table of SAML Parameters

<table>
<thead>
<tr>
<th>SAML Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>authn_requests_signed</td>
<td>Boolean, that when True, signs Hue-initiated authentication requests with X.509 certificate.</td>
</tr>
<tr>
<td>backend</td>
<td>Hard-coded value set to SAML backend library packaged with Hue (libsaml.backend.SAML2Backend).</td>
</tr>
<tr>
<td>base_url</td>
<td>URL that SAML Identity Provider uses for responses. Typically used in Load balanced Hue environments.</td>
</tr>
<tr>
<td>cert_file</td>
<td>Path to X.509 certificate sent with encrypted metadata. File format must be .PEM.</td>
</tr>
<tr>
<td>create_users_on_login</td>
<td>Boolean, that when True, creates users from OpenId, upon successful login.</td>
</tr>
<tr>
<td>entity_id</td>
<td>Service provider ID. Can also accept pattern where '&lt;base_url&gt;' is replaced with server URL base.</td>
</tr>
<tr>
<td>key_file</td>
<td>Path to private key used to encrypt metadata. File format must be .PEM.</td>
</tr>
<tr>
<td>key_file_password</td>
<td>Password used to decrypt the X.509 certificate in memory.</td>
</tr>
<tr>
<td>logout_enabled</td>
<td>Boolean, that when True, enables single logout.</td>
</tr>
<tr>
<td>logout_requests_signed</td>
<td>Boolean, that when True, signs Hue-initiated logout requests with an X.509 certificate.</td>
</tr>
<tr>
<td>metadata_file</td>
<td>Path to readable metadata XML file copied from Identity Provider.</td>
</tr>
<tr>
<td>name_id_format</td>
<td>Format of NameID that Hue requests from SAML server.</td>
</tr>
<tr>
<td>optional_attributes</td>
<td>Comma-separated list of optional attributes that Hue requests from Identity Provider.</td>
</tr>
<tr>
<td>required_attributes</td>
<td>Comma-separated list of required attributes that Hue requests from Identity Provider. For example, uid and email.</td>
</tr>
<tr>
<td>SAML Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>redirect_whitelist</td>
<td>Fully qualified domain name of SAML server: &quot;^/.<em>$&quot;,&quot;https://&lt;SAML_server_FQDN&gt;/.</em>$&quot;.</td>
</tr>
<tr>
<td>user_attribute_mapping</td>
<td>Map of Identity Provider attributes to Hue django user attributes. For example, {'uid':'username', 'email':'email'}.</td>
</tr>
<tr>
<td>username_source</td>
<td>Declares source of username as nameid or attributes.</td>
</tr>
<tr>
<td>xmlsec_binary</td>
<td>Path to xmlsec_binary that signs, verifies, encrypts/decrypts SAML requests and assertions. Must be executable by user, hue.</td>
</tr>
</tbody>
</table>

Description of some properties to be set in hue.ini (via Cloudera Manager):

- **redirect_whitelist** [desktop]
  
  Set to the fully qualified domain name of the SAML server so that Hue can redirect to the SAML server for authentication.

  ```ini
  [desktop]
  redirect_whitelist=^/\.*$","https:\/\/<SAML_server_fully_qualified_domain_name>\/.*$
  ```

  **Note:** Hue uses redirect_whitelist to protect itself from redirecting to unapproved URLs.

- **backend** [desktop][[auth]]
  
  Point to the SAML backend (packaged with Hue):

  ```ini
  backend=libsaml.backend.SAML2Backend
  ```

- **xmlsec_binary** [libsaml]
  
  Point to the xmlsec1 library path:

  ```ini
  xmlsec_binary=/usr/bin/xmlsec1
  ```

  **Note:** To find the path, run: which xmlsec1

- **metadata_file** [libsaml]
  
  Point to the path of the XML file you created from the IdP’s metadata:

  ```ini
  metadata_file=/path/to/<your_idp_metadata_file>.xml
  ```

- **key_file and cert_file** [libsaml]
  
  To encrypt communication between Hue and the Identity Provider, you need a private key and certificate. The private key signs requests sent to the Identity Provider and the certificate file encrypts and decrypts messages from the Identity Provider.

  Copy these files from the Identity Provider and set key_file and cert_file to their respective paths. Both files are in PEM format and must be named with the .PEM extension.

  **Note:** The key and certificate files specified by the key_file and cert_file parameters in hue.ini must be .PEM files.
Users with **password-protected certificates** can set the property, `key_file_password` in `hue.ini`. Hue uses the password to decrypt the SAML certificate *in memory* and passes it to `xmlsec1` through a named pipe. The decrypted certificate never touches the disk. This only works for POSIX-compatible platforms.

**Troubleshooting**

Remember to [Enable DEBUG](#) on page 28 for logging.

**SAML SSL Error**

OpenSSL might fail in CDH 5.5.x and higher with this message:

```
SSLError: [Errno bad handshake] [SSL routines, SSL3_CHECK_CERT_AND_ALGORITHM, dh key too small]
```

To resolve, append the following code to the file,

```
/usr/java/<your_jdk_version>-cloudera/jre/lib/security/java.security:
```

```
jdk.tls.disabledAlgorithms=MD5, RC4, DH
```

**SAML Decrypt Error**

The following error is an indication that you are using a slightly different SAML protocol from what Hue expects:

```
Error: ('failed to decrypt', -1)
```

To resolve:

1. Download and rename Python script, `fix-xmlsec1.txt`.

   ```
   ```

2. Change permissions as appropriate, for example:

   ```
   chmod 755 fix-xmlsec1.py
   ```


4. Run `fix-xmlsec1.py`.

This script repairs the known issue whereby `xmlsec1` is not compiled with `RetrievalMethod` and cannot find the location of the encrypted key. SAML2 responses would sometimes place `EncryptedKey` outside of the `EncryptedData` tree. This script moves `EncryptedKey` under `EncryptedData`.

**Authorize Hue User Groups with Sentry**

Like Hive and Impala, Hue communicates with Sentry using the thrift protocol; but you can also use the [Security Browser](#) in Hue to grant privileges. See [Apache Sentry Made Easy](#).

**Note:** Granting privileges in the [Hue Security Browser](#) > [Hive Tables > Roles](#) is the same as running `grant role` with [HiveServer2 Beeline](#).

This page demonstrates how to use Sentry in Hue by creating three Hue user groups (`readers, writers, sysadmins`), creating three corresponding Sentry roles (`reader_role, writer_role, sysadmin_role`), and granting privileges to those roles.

**Prerequisites**

To grant privileges with Sentry in Hue:
Hue Security

- CDH services must be authenticated (usually with Kerberos but LDAP is also allowed)
- Hue users and groups must be mapped to the OS with Hadoop User Group Mapping.

Hue users and groups do not need to be authenticated; but in production, LDAP authentication is recommended.

Create Hue Users and Groups

Note: Because CDH requires POSIX compliant user names, Hue should use them too (even though Hue leverages django.contrib.auth.models.User which is less strict).

To demonstrate Sentry grants, we create three groups and corresponding Sentry roles.

1. Create three user groups (or import from LDAP). Log on to Hue as a superuser, expand the user drop down, and select Manage Users.
   - **sysadmins** - user1
   - **writers** - user2
   - **readers** - user3, user4

2. Configure group permissions in Hue as appropriate. See Restrict Group Permissions on page 96.

3. Ensure that users and groups are defined in the OS with Hadoop User Group Mapping. See Hue User Permissions on page 84.

Enable Sentry in Hue Security Browser

This section explains how to configure the Sentry service to work with Hue and CDH services: Hue, Hive, and Impala.

1. Set up an external database for Sentry metadata.
   
   Note: See Hue Custom Databases for guidance and create a table something like this:

   ```
   create database sentry default character set utf8 default collate utf8_general_ci;
   grant all on sentry.* to 'sentry'@'%' identified by 'sentrypassword';
   ```

2. Log on to Cloudera Manager and Add the Sentry Service. (Hue does not need a gateway.)

3. Configure Sentry Admin Groups for applicable services and manually add the Hue user group (in this demo, sysadmins).

   Note: See More on Sentry Admin Groups on page 102 for details on user permissions.

4. Enable Sentry Service for each applicable service installed: Hue, Hive, Impala.
   - Go to Hue > Configuration > Sentry Service, select Sentry radio button, and click Save Changes.
   - Repeat for Hive, Impala.

5. Uncheck Hive > Configuration > HiveServer2 Enable Impersonation.

6. Check HDFS > Configuration > Enable Access Control Lists.

7. Ensure all changes are saved and restart applicable services (or the entire cluster).

More on Sentry Admin Groups

On startup, Hue reads sentry_conf/sentry-site.xml and looks for the property, sentry.service.admin.group.
In this demo, group **sysadmins** can grant Sentry roles within Hue. Members of sysadmins must be defined in the OS and also within Hue via Manage Users.

```xml
<property>
  <name>sentry.service.admin.group</name>
  <value>hue,hive,impala,sysadmins</value>
</property>
```

If user1 in our example is part of sysadmins on the backend, but not in Hue Manage Users > Groups, user1 will **not** be able to grant roles in the Hue UI (only in Beeline). If user1 is a Hue superuser, user1 can view roles in the Security Browser > Hive Tables, but cannot edit them.

Conversely, if we add user2 to sysadmins in Manage Users > Groups, user2 will not be able edit roles in Hue nor in Beeline because user2 is not part of sysadmins in the OS.

**Note:** To view logs, **Enable DEBUG** on page 28 and run:

```
cat /var/log/hue/runcpserver.log | grep Sentry
```

### Create Sentry Roles and Grant Privileges

In this section, configure the **sysadmin_role**, and assign it to the **sysadmins** group, because it requires the most permissions.

**System Administrator Requirements**

To create roles and run grants with Sentry in Hue, system administrators must be configured with:

- User/group membership defined in the OS with **Hadoop User Group Mapping**.
- User/group membership defined in Hue **Manage Users**.
- Superuser access configured in Hue **Manage Users**.
- Sentry **Database privileges** set to ALL (for select, insert, create privileges).
- Sentry **URI privileges** to all user directories in HDFS.

**Note:** By default, every user has access to their own HDFS directory in /user; but permissions through Hive/Impala must be granted with a URI.

- **Default ACL** set for hive with r-w-x permissions so that it can load files into hdfs at /hive/warehouse.
- **ACL** set for the same to ensure recursive attempts are covered.

### Create Roles and Grants

1. Log on to Hue as a user with Sentry Admin and Hue Superuser privileges (in this demo, user1).
2. Go to **Security > HiveTables > Roles**: http://<your_hostname>:8889/hue/security/hive#@roles
3. Click **Add**, enter a role name (**sysadmin_role**), and select a group from the drop down (**sysadmins**).
4. Click the plus icon to begin assigning privileges.
5. Select the database radio button:
   - Enter a database name.
   - Select ALL for create database and table privileges.
   - Check the box, grant permissions to give others permission on this database.
6. Select the URI radio button, and enter the path to which you want hive to have access:
   
   ```
   hdfs://<your hostname>:8020/user/
   ```
7. Go to the tab, File ACLs, in the Security Browser. ACLs give hive r-w-x permissions so that it can load files into /hive/warehouse in hdfs.
8. Add a Default ACL. For individual users:
   - Expand the /user directory and select an individual user directory.
   - Click the plus icon under default ACL.
   - Give hive r-w-x permissions and save. If hive is not in the drop down, manually add it.

   For system administrators, go to the command line of your host and give hive r-w-x privileges on all /user.

   ```
  .Edit location of Java path as necessary
   export JAVA_HOME=/usr/java/jdk1.7.0_67
   kinit hdfs
   hdfs dfs -setfacl -m -R default:user:hive:rwx /user
   hdfs dfs -setfacl -m -R user:hive:rwx /user
   ```
9. Create an identical ACL to cover any recursive cases.
10. Create roles and ACLs for groups writers (INSERT) and readers (SELECT).

   ![Figure 1: Hue Guide](image.png)
Deconstruct Hue Actions

Now that we have our three groups, let us analyze how the services operate when users take actions.

1. When user2, in group writers, creates a hive table from a file:
   - hue asks hive to doas user2 and create table with this hdfs /dir/file
   - hive asks sentry if user2 can create tables in this database (DB)
   - hive asks sentry if user2 has sentry level creds on this /dir/file (URI)
   - table gets created as hive (not user2) in hive metastore
   - hive asks hdfs if it can move file into table as hive, not user2 (ACL)

2. When user2, in group writers, creates, saves, and runs a hive query:
   - hue asks hive to doas user2 and run this query
   - hive asks sentry if user2 has permission to run queries

3. When user2, in group writers, creates an oozie job to run the hive query on a schedule:
   - hue asks oozie to doas user2 and run this job
   - oozie does not authorize and runs job as user2
   - oozie spawns mapred job that runs hive query as user2
   - mapred job asks hive to run this query as user2
   - hive asks sentry if user2 has permission to run queries
Hue Troubleshooting

This section addresses possible obstacles when installing, configuring, and using Hue. Watch this space for more topics!

Potential Misconfiguration Detected

This page covers various configuration errors. The goal is for all configuration checks to pass.

Checking current configuration

Configuration files located in /var/run/cloudera-scm-agent/process/108-hue-HUE_SERVER

All OK. Configuration check passed.

Preferred Storage Engine

PREFERRED_STORAGE_ENGINE: We recommend MySQL InnoDB engine over MyISAM which does not support transactions.

Warning: Talk to your DBA before changing the storage engine for the Hue database tables.

Alter Hue database tables from MyISAM to InnoDB

1. Stop the Hue service in Cloudera Manager: go to Cluster > Hue and select Actions > Stop.
2. Log on to the host of your MySQL server.
3. Look for any MyISAM tables in your Hue server database:

   mysql -u root -p <root password>

   SELECT table_schema, table_name, engine 
   FROM information_schema.tables 
   WHERE engine = 'MyISAM' AND table_schema = '<hue database name>';

   quit
4. Set the engine to InnoDB for all Hue database tables:

```sql
# Create script, /tmp/set_engine_innodb.ddl
mysql -u root -p<root password> -e "SELECT CONCAT('ALTER TABLE ',table_schema,'.',table_name,' engine=InnoDB;') FROM information_schema.tables WHERE engine = 'MyISAM' AND table_schema = '<hue database name>';" | grep "ALTER TABLE <hue database name>" > /tmp/set_engine_innodb.ddl

# Run script
mysql -u root -p<root password> < /tmp/set_engine_innodb.ddl
```

5. Verify that no MyISAM tables exist by rerunning the SELECT statement in step 3 on page 106.
6. Start the Hue service.

MySQL Storage Engine

MYSQL_STORAGE_ENGINE: All tables in the database must be of the same storage engine type (preferably InnoDB).

Follow the instructions in the section, Preferred Storage Engine on page 106, to ensure all Hue tables use InnoDB.
Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License.

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.


Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims
licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution.

You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

1. You must give any other recipients of the Work or Derivative Works a copy of this License; and

2. You must cause any modified files to carry prominent notices stating that You changed the files; and

3. You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

4. If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions.

Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks.

This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty.

Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability.

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability.
While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

   http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.