## Install Hive

<https://phoenixnap.com/kb/install-hive-on-ubuntu>

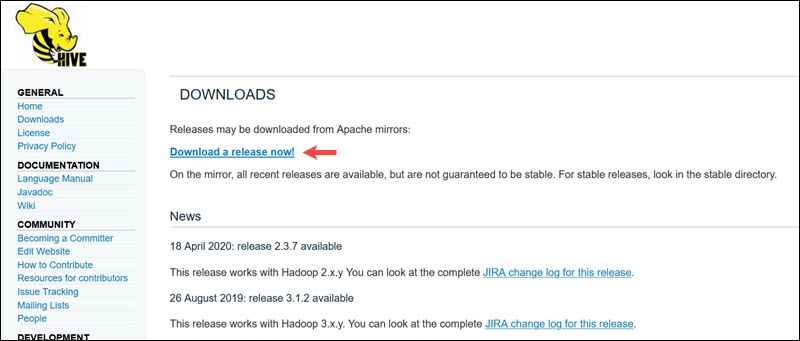
<https://stackoverflow.com/questions/50500718/underlying-cause-java-lang-classnotfoundexception-com-mysql-jdbc-driver-use>

To configure Apache Hive, first you need to download and unzip Hive. Then you need to customize the following files and settings:

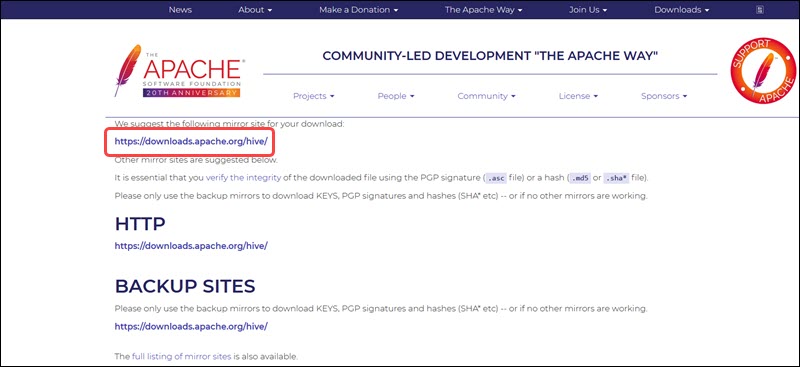
* + Edit .bashrc file
  + Edit hive-config.sh file
  + Create Hive directories in HDFS
  + Configure hive-site.xml file
  + Initiate Derby database // In our case, we will be using MySQL

### Step 1: Download and Untar Hive

Visit the [Apache Hive official download page](https://hive.apache.org/downloads.html) and determine which Hive version is best suited for your Hadoop edition. Once you establish which version you need, select the Download a Release Now! option.



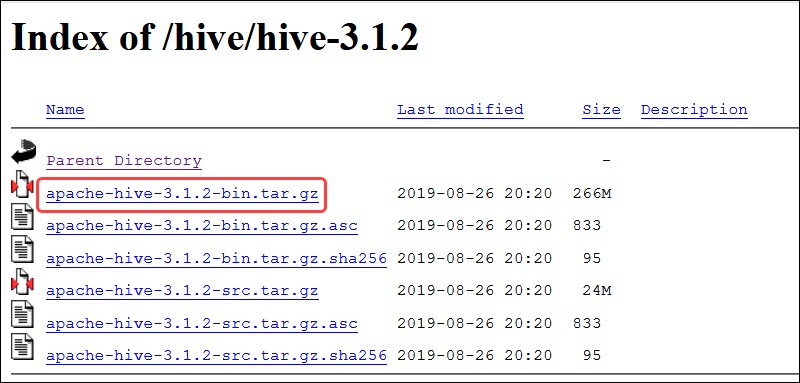
The mirror link on the subsequent page leads to the directories containing available Hive tar packages. This page also provides useful instructions on how to validate the integrity of files retrieved from mirror sites.



The Ubuntu system presented in this guide already has Hadoop 3.2.1 installed. This Hadoop version is compatible with the Hive 3.1.2 release.



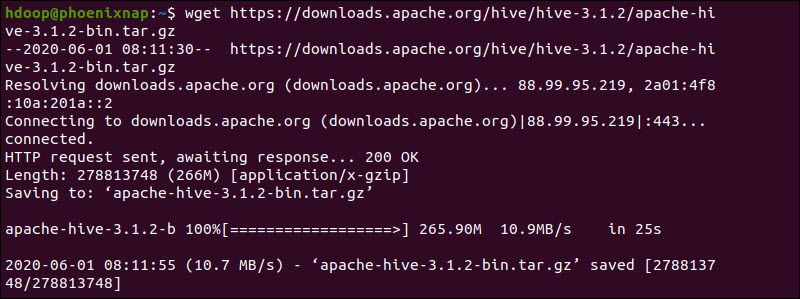
Select the apache-hive-3.1.2-bin.tar.gz file to begin the download process.



#### Download Using wget

Alternatively, access your Ubuntu command line and download the compressed Hive files using and the **wget** command followed by the download path:

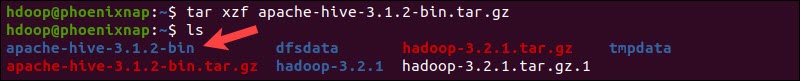
wget https://dlcdn.apache.org/hive/hive-3.1.2/apache-hive-3.1.2-bin.tar.gz



Once the download process is complete, untar the compressed Hive package:

tar xzf apache-hive-3.1.2-bin.tar.gz

The Hive binary files are now located in the **apache-hive-3.1.2-bin** directory.



### Step 2: Configure Hive Environment Variables (bashrc)

The $HIVE\_HOME environment variable needs to direct the client shell to the **apache-hive-3.1.2-bin** directory. Edit the **.bashrc** shell configuration file using a text editor of your choice (we will be using nano):

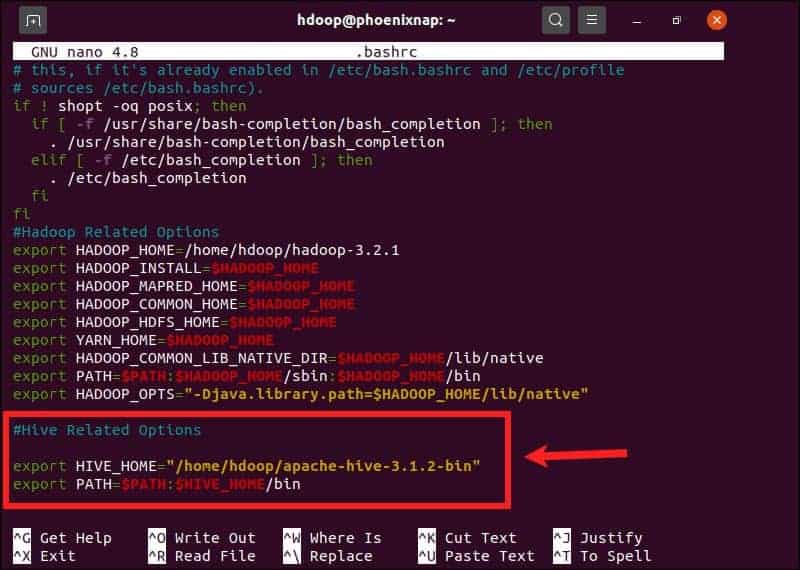
sudo nano .bashrc

Append the following Hive environment variables to the **.bashrc** file:

export HIVE\_HOME=~/apache-hive-3.1.2-bin

export PATH=$PATH:$HIVE\_HOME/bin

The Hadoop environment variables are located within the same file.



Save and exit the **.bashrc** file once you add the Hive variables. Apply the changes to the current environment with the following command:

source ~/.bashrc

### Step 3: Edit hive-config.sh file

Apache Hive needs to be able to interact with the Hadoop Distributed File System. Access the **hive-config.sh** file using the previously created $HIVE\_HOME variable:

sudo nano $HIVE\_HOME/bin/hive-config.sh

Note: The **hive-config.sh** file is in the **bin** directory within your Hive installation directory.

Add the HADOOP\_HOME variable and the full path to your Hadoop directory:

export HADOOP\_HOME=~/hadoop/hadoop-3.3.3



Save the edits and exit the **hive-config.sh** file.

### Step 4: Create Hive Directories in HDFS

Create two separate directories to store data in the HDFS layer:

* The temporary, **tmp** directory is going to store the intermediate results of Hive processes.
* The **warehouse** directory is going to store the [Hive related tables](https://phoenixnap.com/kb/hive-create-table).

#### Create tmp Directory

Create a **tmp** directory within the HDFS storage layer. This directory is going to store the intermediary data Hive sends to the HDFS:

hdfs dfs -mkdir /tmp

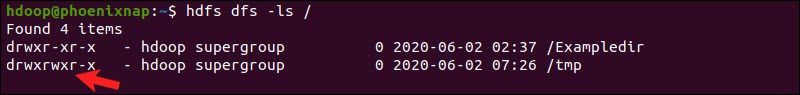
Add write and execute permissions to tmp group members:

hdfs dfs -chmod g+w /tmp

Check if the permissions were added correctly:

hdfs dfs -ls /

The output confirms that users now have write and execute permissions.



#### Create warehouse Directory

Create the **warehouse** directory within the **/user/hive/** parent directory:

hdfs dfs -mkdir -p /user/hive/warehouse

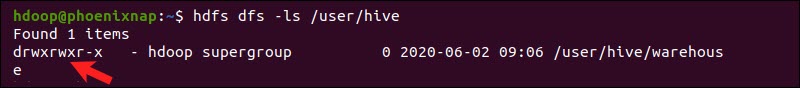
Add write and execute permissions to **warehouse** group members:

hdfs dfs -chmod g+w /user/hive/warehouse

Check if the permissions were added correctly:

hdfs dfs -ls /user/hive

The output confirms that users now have write and execute permissions.



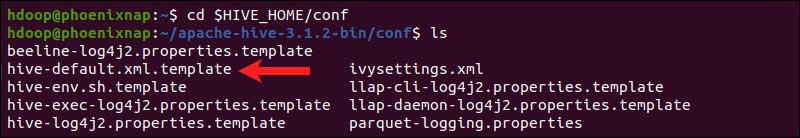
### Step 5: Configure hive-site.xml File (Optional)

Apache Hive distributions contain template configuration files by default. The template files are located within the Hive **conf** directory and outline default Hive settings.

Use the following command to locate the correct file:

cd $HIVE\_HOME/conf

List the files contained in the folder using the ls command.



Use the **hive-default.xml.template** to create the **hive-site.xml** file:

cp hive-default.xml.template hive-site.xml

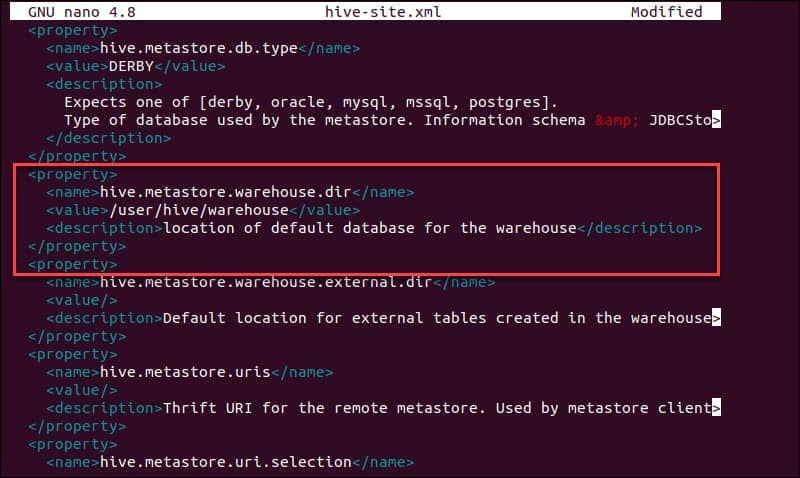
cp hive-env.sh.template hive-env.sh

Access the **hive-site.xml** file using the nano text editor:

sudo nano hive-site.xml

Note: The hive-site.xml file controls every aspect of Hive operations. The number of available advanced settings can be overwhelming and highly specific. Consult the [official Hive Configuration Documentation](https://cwiki.apache.org/confluence/display/Hive/AdminManual+Configuration#AdminManualConfiguration-HiveConfigurationVariables) regularly when customizing Hive and Hive Metastore settings.

Using Hive in a stand-alone mode rather than in a real-life Apache Hadoop cluster is a safe option for newcomers. You can configure the system to use your local storage rather than the HDFS layer by setting the **hive.metastore.warehouse.dir** parameter value to the location of your Hive **warehouse** directory.



Then, search for “&#8” in the hive-site.xml file and remove it. If it is not removed, it gives an error when initializing Hive.

Step 6(a): Install MySQL Server

Install mysql server and make sure it is running.

sudo apt-get install mysql-server

sudo service mysql start

sudo mysql\_secure\_installation utility

### Step 6: Configure the MySQL JDBC Connector

Determine the MySQL version as explained [here](#_Install). In my case it is 8.0.27. Download the tar zip in the Downloads folder, unzip the file and copy it in the $HIVE\_HOME/lib folder:

wget <https://dev.mysql.com/get/Downloads/Connector-J/mysql-connector-java-8.0.30.tar.gz>

cp mysql-connector-java-8.0.30.tar.gz $HIVE\_HOME/lib

tar xzf mysql-connector-java-8.0.30.tar.gz

cd mysql-connector-java-8.0.30

mv mysql-connector-java-8.0.30.jar ./..

### Step 7: Create the Hive user in MySQL

Start mysql and run the following commands:

$ sudo mysql -u root -p

CREATE USER 'hive'@'localhost' IDENTIFIED BY 'Password@123';

GRANT ALL PRIVILEGES ON \*.\* to 'hive'@'localhost' WITH GRANT OPTION;

flush privileges;

IF the above commands does not work, try this:

CREATE USER 'hive'@'%' IDENTIFIED BY 'Password@123';

GRANT ALL PRIViLEGES ON \*.\* to 'hive'@'%' WITH GRANT OPTION;

flush privileges;

If the metastore\_db database exists, delete it. It will be created when we start Hive.

Show databases;

Drop database metastore\_db;

Exit out of mysql.

### Step 8: Apply Settings in hive-site.xml and hive-env.sh

Open the hive-site.xml file and apply the following changes. Search for the “name” entries and make changes accordingly.

sudo nano $HIVE\_HOME/conf/hive-site.xml

Apply these changes:

<property>

<name>javax.jdo.option.ConnectionURL</name>

<value> jdbc:mysql://localhost:3306/metastore\_db?createDatabaseIfNotExist=true</value>

<description>JDBC connection string used by Hive Metastore</description>

</property>

<property>

<name>javax.jdo.option.ConnectionDriverName</name>

<value>com.mysql.jdbc.Driver</value>

<description>JDBC Driver class</description>

</property>

<property>

<name>javax.jdo.option.ConnectionUserName</name>

<value>hive</value>

<description>Metastore database user name</description>

</property>

<property>

<name>javax.jdo.option.ConnectionPassword</name>

<value>Password@123</value>

<description>Metastore database password</description>

</property>

Save the file and exit the editor.

Open the hive-env.sh file (May not exist! If so, create it and insert line):

sudo nano $HIVE\_HOME/conf/hive-env.sh

Add this at the end:

export HADOOP\_HOME=~/hadoop/hadoop-3.3.3

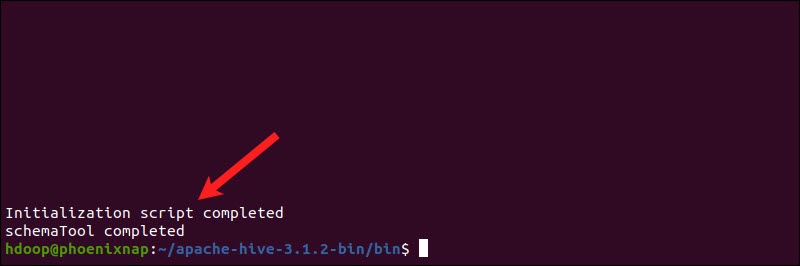
Save the file and exit the editor.

### Step 9: Initiate MySQL Database

Apache Hive uses the MySQL database to store metadata. Initiate the database, from the Hive **bin** directory using the schematool command:

$HIVE\_HOME/bin/schematool -initSchema -dbType mysql

The process can take a few moments to complete. *If you get an error on the “guava” incompatibility, refer the next sub-section on how to fix it*.



Derby is the default metadata store for Hive. If you plan to use a different database solution, such as [MySQL or PostgreSQL](https://phoenixnap.com/kb/postgres-vs-mysql), you can specify a database type in the **hive-site.xml** file.

#### How to Fix guava Incompatibility Error in Hive

If the Derby database does not successfully initiate,  you might receive an error with the following content:

**“Exception in thread “main” java.lang.NoSuchMethodError: com.google.common.base.Preconditions.checkArgument(ZLjava/lang/String;Ljava/lang/Object;)V”**

This error indicates that there is most likely an incompatibility issue between Hadoop and Hive **guava** versions.

Locate the guava jar file in the Hive **lib** directory:

ls $HIVE\_HOME/lib



Locate the guava jar file in the Hadoop **lib** directory as well:

ls $HADOOP\_HOME/share/hadoop/hdfs/lib



The two listed versions are not compatible and are causing the error. Remove the existing guava file from the Hive **lib** directory:

mv $HIVE\_HOME/lib/guava-19.0.jar $HIVE\_HOME/lib/guava-19.0.jar.old

Copy the guava file from the Hadoop **lib** directory to the Hive **lib** directory:

cp $HADOOP\_HOME/share/hadoop/hdfs/lib/guava-27.0-jre.jar $HIVE\_HOME/lib/

Use the schematool command once again to initiate the Derby database:

$HIVE\_HOME/bin/schematool -initSchema -dbType mysql

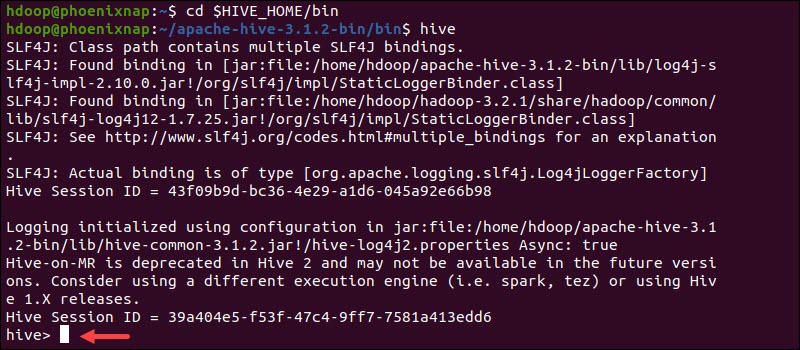
### Step 10: Launch Hive Client Shell on Ubuntu

Start the Hive command-line interface using the following commands:

cd $HIVE\_HOME/bin

hive

You are now able to issue SQL-like commands and directly interact with HDFS.



## ERRATA

If you get an “[java.net.URISyntaxException when starting HIVE](https://stackoverflow.com/questions/27099898/java-net-urisyntaxexception-when-starting-hive)” exception, do the following:

Put the following at the beginning of hive-site.xml

<property>

<name>system:java.io.tmpdir</name>

<value>/tmp/hive/java</value>

</property>

<property>

<name>system:user.name</name>

<value>${user.name}</value>

</property>

# Change the following values too:

<name>hive.exec.scratchdir</name>

<value>/tmp/hive-${user.name}</value>

<name>hive.exec.local.scratchdir</name>

<value>/tmp/${user.name}</value>

<name>hive.downloaded.resources.dir</name>

<value>/tmp/${user.name}\_resources</value>

<name>hive.scratch.dir.permission</name>

<value>733</value>