

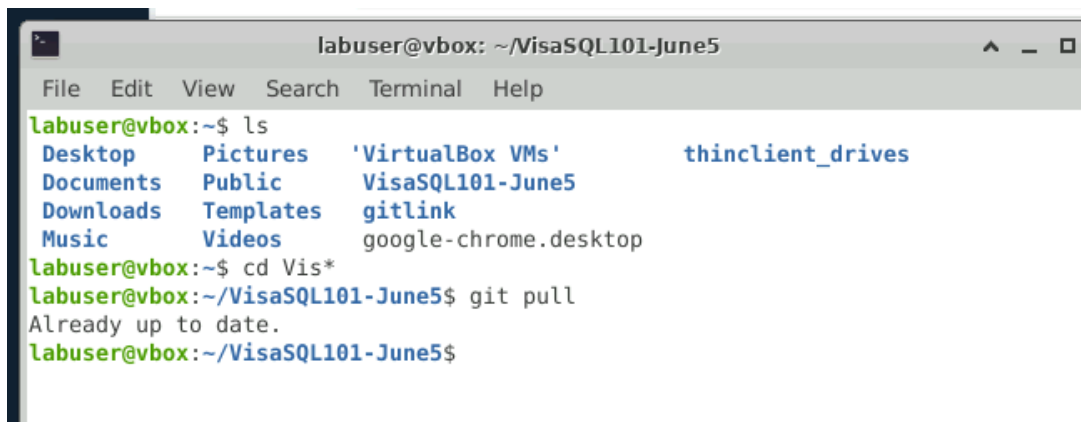
Lab 1: Hive Basics

These labs assume you are using the class repository cloned into the Nuvelab VM. You can also clone the repo to your HDP VM but that variation is not covered in the lab instructions.

All of the hive commands in this document are listed in text form you can cut and paste – these can be found in the accompanying document named “commands.txt” in each lab directory.

Importing Data

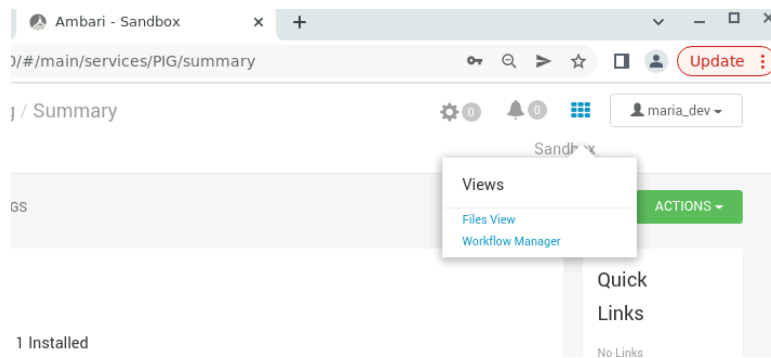
1. Make sure you have the latest version of the repository in the Nuvelab repository. At the command line, go into the repository directory and issue the “git pull” command. This assumes you have already installed the repo according to the instructions in the Lab setup section



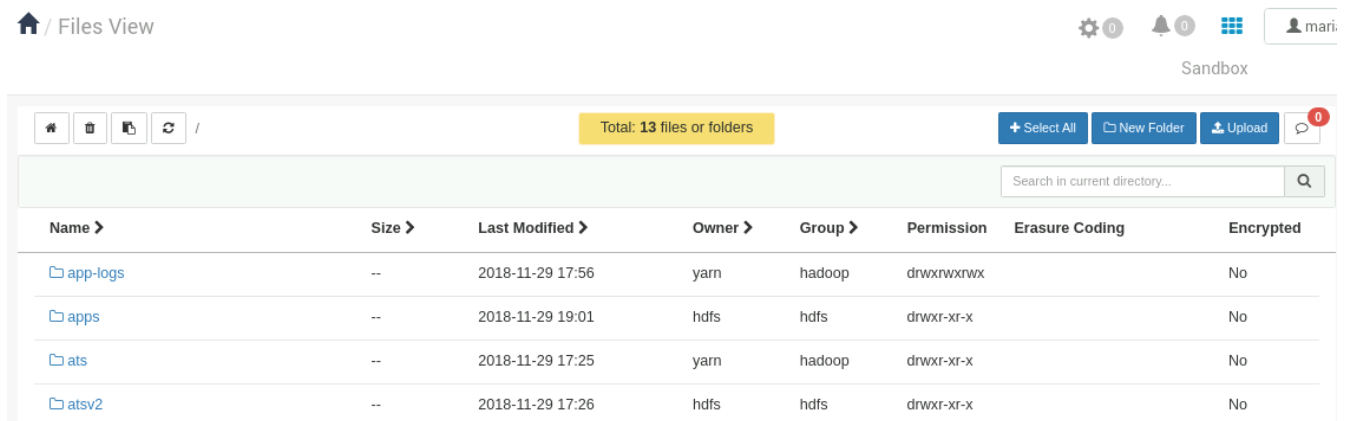
```
labuser@vbox: ~/VisaSQL101-June5
File Edit View Search Terminal Help
labuser@vbox:~$ ls
Desktop    Pictures  'VirtualBox VMs'  thinclient_drives
Documents  Public   VisaSQL101-June5
Downloads  Templates gitlink
Music      Videos  google-chrome.desktop
labuser@vbox:~$ cd Vis*
labuser@vbox:~/VisaSQL101-June5$ git pull
Already up to date.
labuser@vbox:~/VisaSQL101-June5$
```

2. Ensure that your HDP VM is up and running and all the services ready.
3. Go to localhost:8080 or go to localhost:1080 and select the Ambari link. Once this opens, login as maria_dev (password maria_dev) and select the “Files View” option from the blue boxes next to the login.

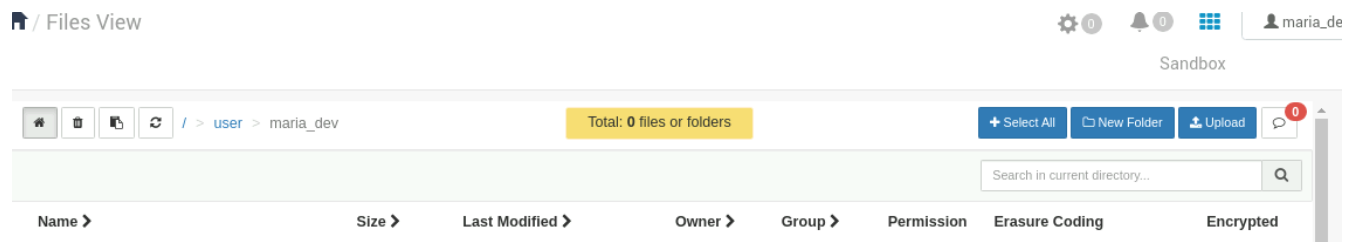
Lab 1: Hive Basics



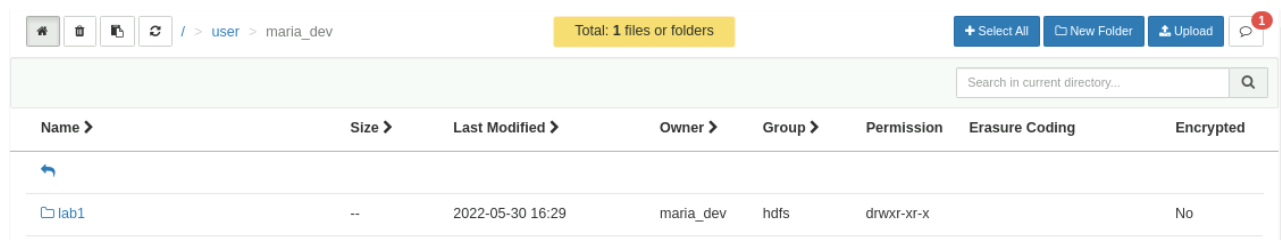
- This may take a few minutes to load, but what you will see is a GUI of the Hadoop file system. This is not the linux file system but the HDFS.



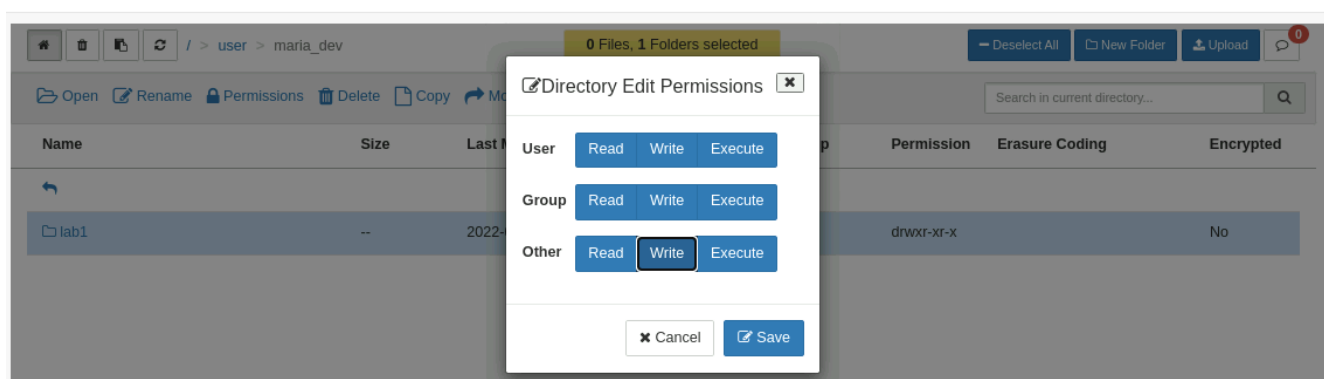
- Use the interface to navigate to the folder “/user/maria_dev” which should be empty.



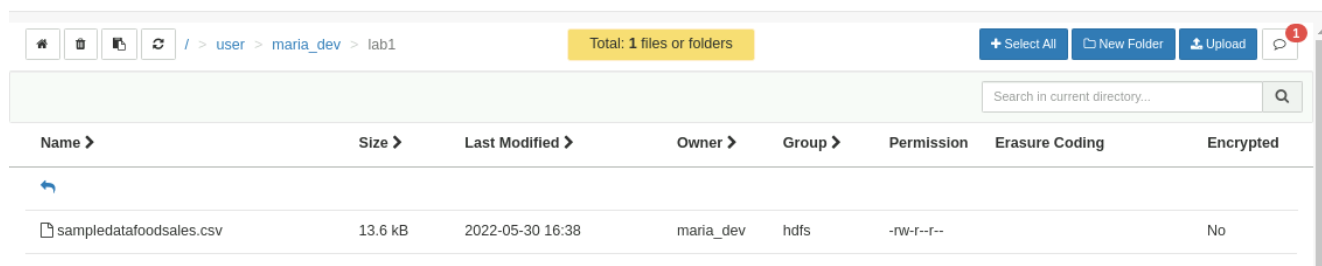
6. Create a folder called lab1 using the “New Folder” button on the right.



7. To ensure we don't get any odd permission errors, use the “Permissions” option to make the folder writeable by everyone (including Hive). This is NOT recommended for regular operations but will just make the labs a bit more streamlined.




8. Go into the lab1 directory and select the “Upload” blue button. This will allow you to upload files from your Nuvelab VM directly into the HDP VM Hadoop file system. Just follow the prompts and upload “sampledatafoodsals.csv” from the repo.



9. Just to make things simpler for the rest of the lab, add write permissions for everyone to the csv file the same way you did for the folder. If you don't see the changed permis-

sions after you get the green pop-up telling you the were successfully changed, just refresh Ambari – sometimes the display lags a bit.

 sampledatafoodsals.csv	13.6 kB	2022-05-30 16:38	maria_dev	hdfs	-rw-rw-rw-
--	---------	------------------	-----------	------	------------

Creating a Database

1. Log into the webshell (localhost:4200) as maria_dev and start hive.
2. At the hive prompt, enter the following command to see a list of the current databases.
`show databases;`
3. Create a new database for the lab using:
`create database food;`
4. Repeat the command in step 2 to ensure the database has been created. To get more information about the database, use the describe command:
`describe database food;`
5. Notice that if you try to execute step 3 again, you will be an error that the database already exists
6. Now drop the database like this:
`drop database food;`
7. Run the show databases command again to see that it no longer exists. Once you have confirmed that, then recreate it using the command in step 3.
8. To use the new food database, execute the command:
`use food;`

Creating an Internal Table

1. This part assumes you have created and are using a database called “food”.
2. The schema for the data that will be loaded into the table is:

```
orderdate  string;
region     string;
city       string;
category   string;
product    string;
```

```
quantity    int;
unitprice   double;
total       double;
```

3. In the Nuvelab VM, open the csv file in libreoffice so you can see what the data looks like. Notice that the first row are the column headers. To open up the file, just double click on it.
4. To see that there are no tables in this database yet, use the command

```
show tables;
```

5. Back in the HPD VM, create a table using the following command (you can type it in as one line, the formatting here is just for readability):

```
create table foodbase(
    orderdate string,
    region    string,
    city      string,
    category  string,
    product   string,
    quantity  int,
    unitprice double,
    total     double)
row format delimited fields terminated by ','
lines terminated by '\n' stored as textfile;
```

6. To see that the table has been created correctly, use the following commands

```
show tables;
describe foodbase;
```

7. We don't want the first line of the CSV file to be read as data so we alter the table properties so that it skips the first row.

```
alter table foodbase SET TBLPROPERTIES ("skip.header.line.count"="1");
```

8. Now read in the data from the CSV file.

```
load data inpath '/user/maria_dev/lab1/sampledatafoodsales.csv'
into table foodbase;
```

9. Confirm the table load by running:

```
describe extended foodbase;
```

10. Inspect the data by running:

```
select * from foodbase limit 20;
```

Locating the Table

Because this is a managed table, it is stored in the directory

```
/warehouse/tablespace/managed/hive/foodbase
```

Use the file explorer to examine the contents of this directory

Dropping the Table

1. The table can be deleted with the command:

```
drop table foodbase;
```

2. Once you execute this command, confirm the table no longer exists by running:

```
show tables;
```

3. Confirm using the file explorer that it also has been removed from the location it was being stored

```
/warehouse/tablespace/managed/hive/foodbase
```

4. Now go to the /user/maria_dev/lab1 directory and confirm that the data is no longer there. This is because the table you created was a Hive managed table.

Creating an External Table

1. Repeat the steps to reload the data into the /user/maria_dev/lab1 directory that you did in part one of this lab.
2. Create a fully writeable directory, like you did for the directory lab1, to store the external table called:

```
/user/maria_dev/data
```

3. Now recreate the table, but this time add the external keyword and a location.

```
create external table foodbase(
  orderdate string,
  region    string,
  city      string,
  category  string,
  product   string,
  quantity  int,
  unitprice double,
  total     double)
row format delimited fields terminated by ','
lines terminated by '\n' stored as textfile
location '/user/maria_dev/data/foods';
```

If you get permission errors when trying to create the table, it is because the Hive user is trying to create the table in a directory that it might not have permission to write to. Ensuring that the /user/maria_dev/data directory is fully writeable for all should solve this problem.

4. Confirm that the table is created by running. If you get perm

```
describe foodbase;
```

5. Load the table as before and run a select query to see that the data is there as before.

```
load data inpath '/user/maria_dev/lab1/sampledatafoodsales.csv'
into table foodbase;

select * from foodbase limit 20;
```

6. Go into the data directory you created and examine the contents. How many copies of the original CSV file now exist and where are they?
7. The table can be dropped, just like before with the command:

```
drop table foodbase;
```

8. Once you execute this command, confirm the table no longer exists by running:

```
show tables;
```

9. Go back and examine the locations you already looked at previously and see if there are any copies of the data.

End of Lab

This will not generally be repeated at the end of the labs, but it is a good idea to delete all the lab resources you created at the end of each lab.

Drop any tables you might have and then switch to the default database and delete the food database;

```
use default;
drop database food;
```