Hortonworks Tutorial 2

Working with Pig

In this tutorial we review how to upload files to HDP, and how to load and save data in **Pig** and then use these files to perform a number of basic Pig operations.

Task 1. Using HDP to upload files

We will used the same dataset used in Tutorial 1. If you do not have the files, download the files from this link: `http://opensourcesports.com/files/basketball/BasketballDB-20130121.zip`. We will use the following 3 files: basketball_master.csv, basketball_players.csv and basketball_teams.csv. You also need to upload the files to HDP (Please see Tutorial 1), if you do not have the files in your HDP.

Task 2. Introduction to Pig

1. From the Off-canvas menu at the top (right-hand side) select "Pig View" (see Fig. 1). This will bring up the Ambari Pig User View interface. Your Pig View does not have any scripts to display, so it will look like Fig. 2:

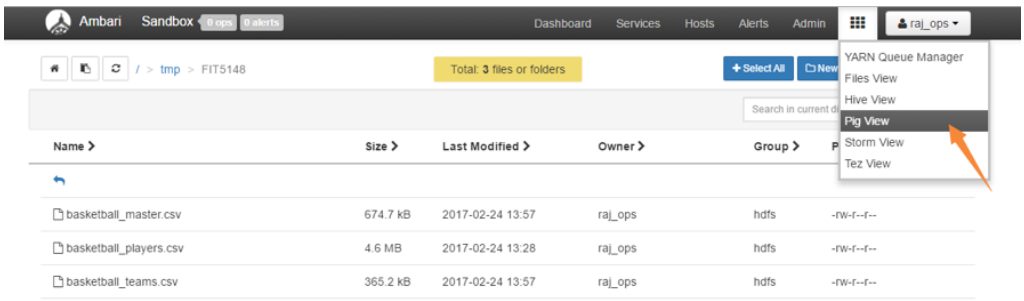

Figure 1: Pig view

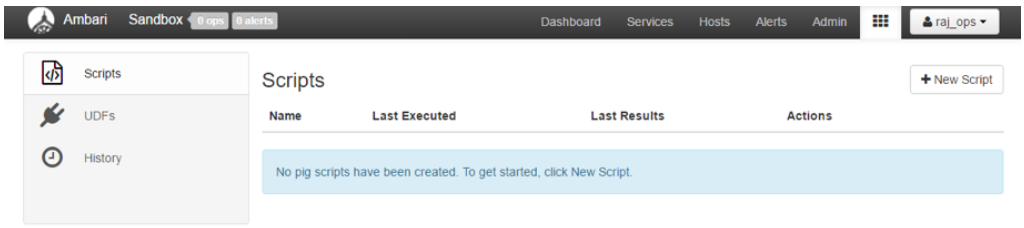

Figure 2: Initial Pig View

2. To get started push the button "New Script" at the top right. Name the script as "t2", then click on the "Create" button. If you leave the gap Script HDFS Location (Optional) empty, it will be filled automatically. Then, you will see the screen as Fig. 3.

   On the left, we can choose different options: our saved "Pig Scripts", "UDFs", and "History of the Pig jobs" executed in the past. The centre is the composition area where we write our script. The top right of the composition area includes buttons to "Execute", "Explain" and perform a "Syntax check" of the current script. At the very bottom we can add "arguments". A special feature of the interface is the "Pig helper" that will provide us with templates for the Functions, Operators, I/O statements, HCatLoader() and Python UDF. The following screenshot shows and describes the various components and features of the Pig User View:

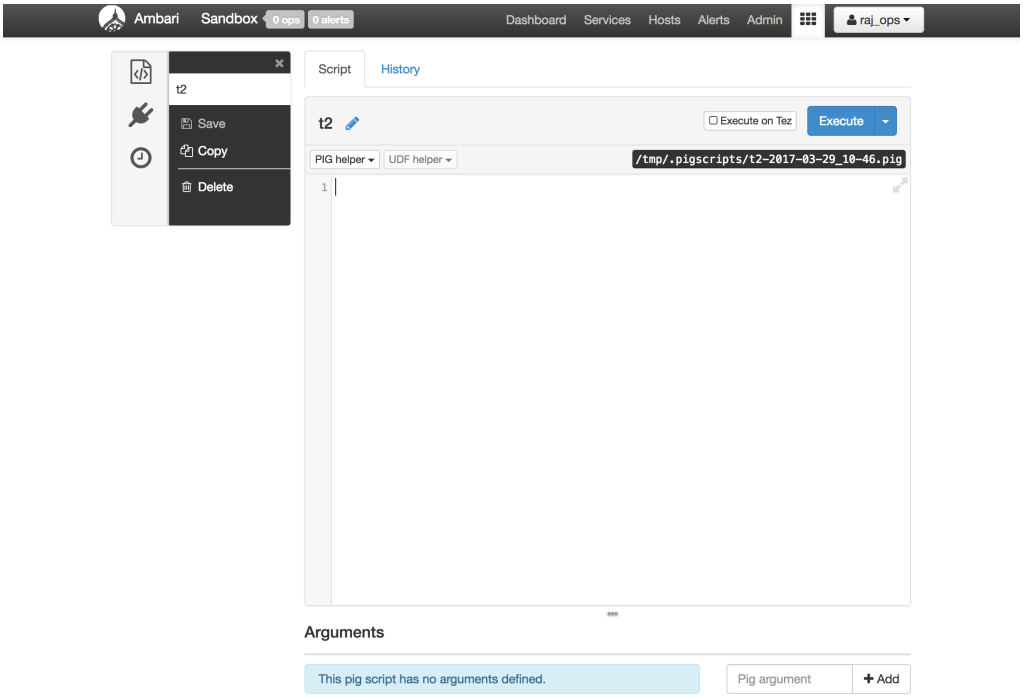

Figure 3: Create a new script in Pig View

Task 3. Loading the Data

1. The first thing we need to do is load the data. We use the load statement for this. The PigStorage function is what does the loading and we pass it a comma as the data delimiter. Our code is:

   ```
   players = LOAD '/tmp/FIT5148/basketball_players.csv' using PigStorage(',');
   ```

   Adding the "players = " before the template saves the results into players. Note the = has to have a space before and after it.

   This statement loads tuples from "basketball_players.csv" in a relation named players without specifying any schema and data types. When we need to refer any field of the players bag, we can use positional notation which starts from 0.

2. The next thing we want to do is to show the results. Use the "DUMP" command to display players. You write the code (DUMP players) or click "PIG helper→I/O→DUMP" template and replace %VAR% with players.

```
players = LOAD '/tmp/FIT5148/basketball_players.csv' using PigStorage(',');
DUMP players;
```

3. One the left side of the screen, click "Save". Then click "Execute" at top right to run the script. It will start the job and show RUNNING as seen Fig 4 - Fig 6. When the job is complete the output of the job is displayed in the "Results" section. This action creates one or more MapReduce jobs. Click on the Logs dropdown menu to see what happened when your script ran. Errors will appear as seen in Fig. 7.

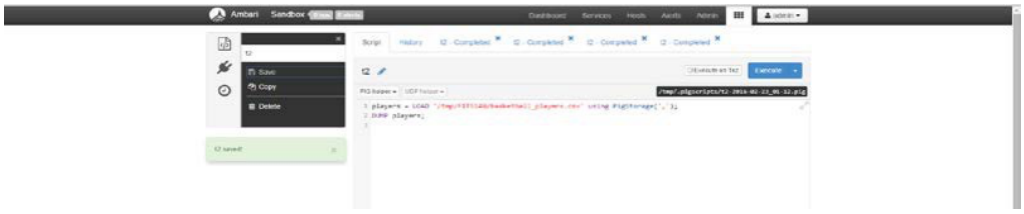

Figure 4: Load Data in Pig View

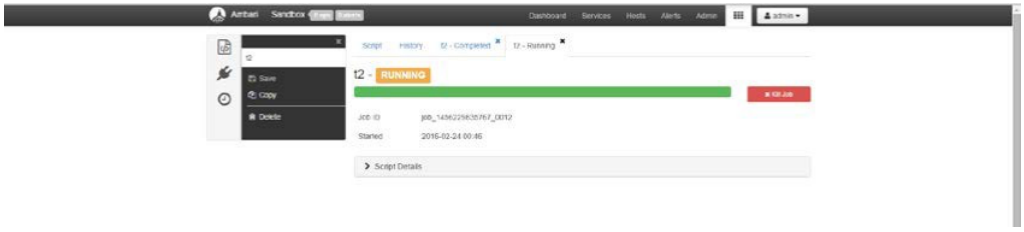

Figure 5: A script runs in Pig View

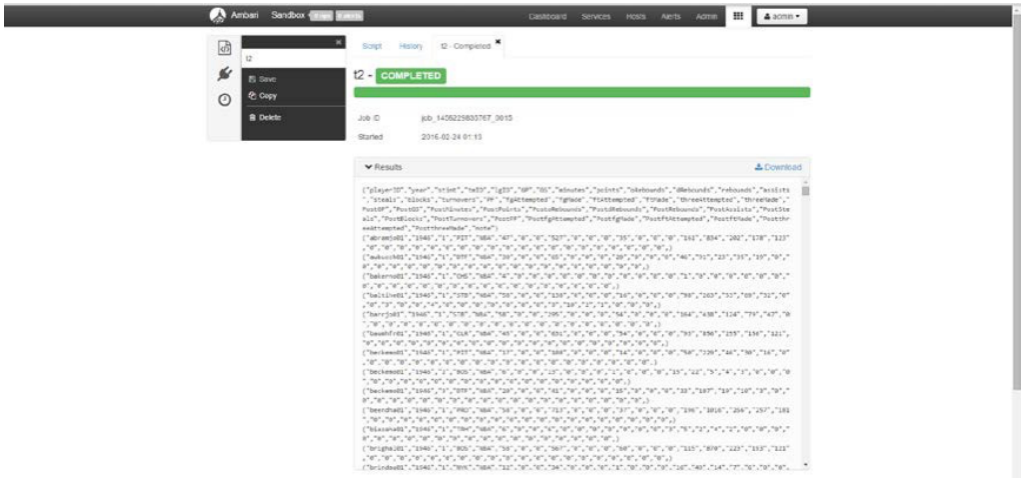

Figure 6: The results of the execution of a script in Pig View

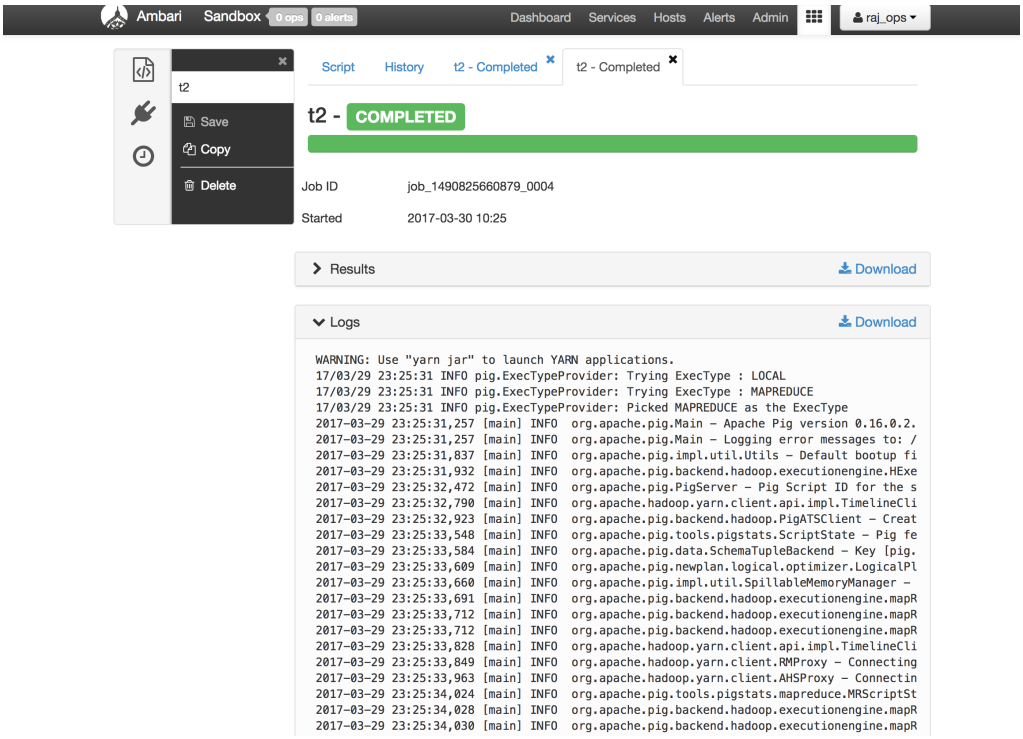

Figure 7: Erros in Logs in Pig View

Task 4. Select specific columns from a relation

One of the key uses of Pig is data transformation. You can define a new relation based on the fields of an existing relation using the ”FOREACH” command.

Define a new relation ”player_points”, which will contain only the playerID, year and points fields from the relation players. We will use a FOREACH statement to iterate through the players data object.

We can use Pig Helper to provide us with a template. Delete the DUMP players command from your Pig script. We will click on Pig Helper, select ”Relational Operators functions→ FOREACH%DATA%GENERATE%NEW_DATA%” template (see Fig. 8)

So the FOREACH statement will iterate through the players data object and generate pulls out selected fields and assigns them names. The new data object we are creating is then named ”player_points”. We will add this code:

```
player_points = FOREACH players GENERATE $0 as playerID, $1 as year,
$8 as points;
```

Task 5. Define a new relation from an existing relation

You can define a new relation based on an existing one. For example, define the following ”players_limit” relation, which is a collection of 100 entries (arbitrarily selected), from the ”player_points” relation.

The Pig ”Limit” operator will limit the number of output tuples. There is no guarantee which tuples will be returned, and the tuples that are returned can change from one run to the next.

Add the following line to the end of your code:

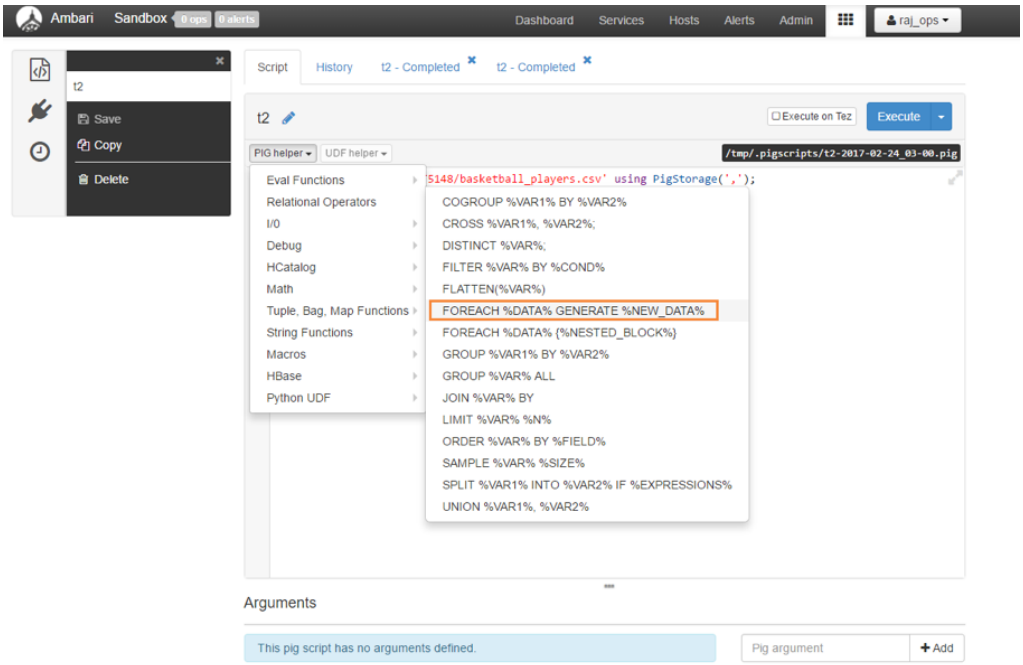

Figure 8: Choose a FOREACH template

```
players_limit = LIMIT player_points 100;
```

Task 6. Sort the data using "ORDER BY"

Use the "ORDER BY" command to sort a relation by one or more of its fields. Add code named "order_players" and enter the following commands to sort the "players_limit" by year in a descending order then "playerID":

```
order_players = ORDER players_limit by year desc, playerID;
```

Task 7. Filter and Group the data

The "GROUP" command allows you to group a relation by one of its fields. Add code named "group_players". Then, enter the following commands, which group the "order_players" relation by year.

```
group_players = GROUP order_players BY year;
```

Task 8. Run the above script

Add another line to dump the result. The completed lines of code will be:

```
players = LOAD '/tmp/FIT5148/basketball_players.csv' using PigStorage(',');
player_points = FOREACH players GENERATE $0 as playerID, $1 as year,
$8 as points;
players_limit = LIMIT player_points 100;
order_players = ORDER players_limit by year desc, playerID;
group_players = GROUP order_players BY year;
DUMP group_players;
```

Save and execute the script again. The result will be as follows:

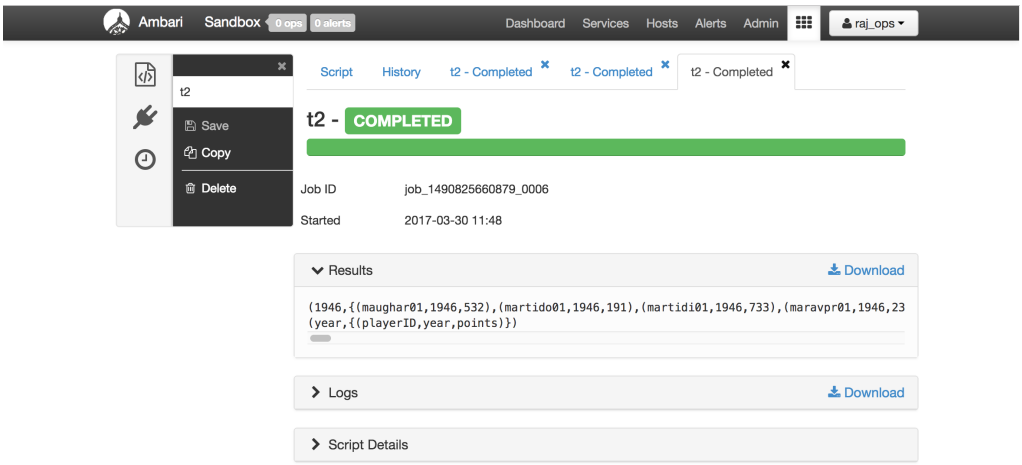

Figure 9: Result of running the script, group_players

Task 9. Store relationship data into a HDFS File

We will use the "STORE" command to output a relation into a new file in HDFS (Replacing Dump with STORE) instead of dump it to the screen. Enter the following command to output the "group_players" relation to a folder named " /output/group_players" (then save and execute):

```
players = LOAD '/tmp/FIT5148/basketball_players.csv' using PigStorage(',');
player_points = FOREACH players GENERATE $0 as playerID, $1 as year,
$8 as points;
players_limit = LIMIT player_points 100;
order_players = ORDER players_limit by year desc, playerID;
group_players = GROUP order_players BY year;
STORE group_players INTO '/tmp/FIT5148/output/group_players';
```

Once this job is finished, go to "Files View" and navigate to "/tmp/FIT5148". Look for a newly created folder called output. You will see an output file named "part-r-00000" under /tmp/FIT5148/output/group_players. Click on the file part-r-00000, and click the "Open" menu. It will show data stored in the file (see Fig 10).

Task 10. Load Data in Pig using Hcatalog

1. We will use "HCatalog" to load data into Pig. HCatalog allows us to share schema across tools and users within our Hadoop environment. It also allows us to factor out schema and location information from our queries and scripts and centralize them in a common repository. Since it is in HCatalog we can use the "HCatLoader()"" function.
2. Pig makes it easy by allowing us to give the table a name or alias and not have to worry about allocating space and defining the structure. We just have to worry about how we are processing the table. Before we can run the Pig code, one of the requirements for the "HCatStorer()"" class is that the table must already exist in "Hive".
3. Create a new script. Then, click on "Pig Helper", then select "HCatalog". We use the "LOAD" template for this (see Fig. 11).
4. The entry "%TABLE%"" is highlighted in red. Replace that with the table "basketball_teams". Add the "teams_raw = " before the LOAD statement. This saves the results into "teams_raw".

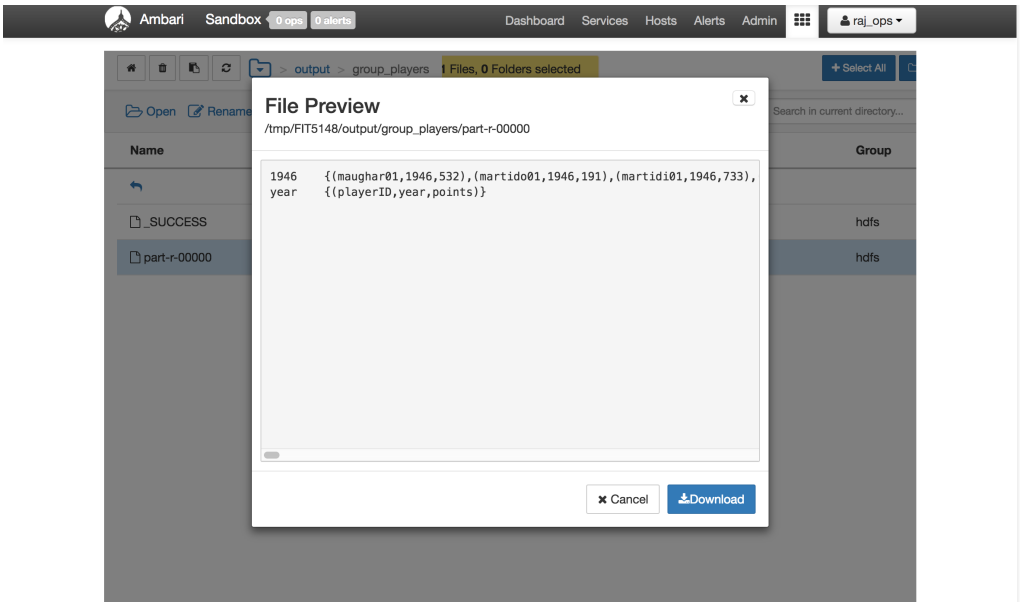

Figure 10: Result of running the script in the output folder

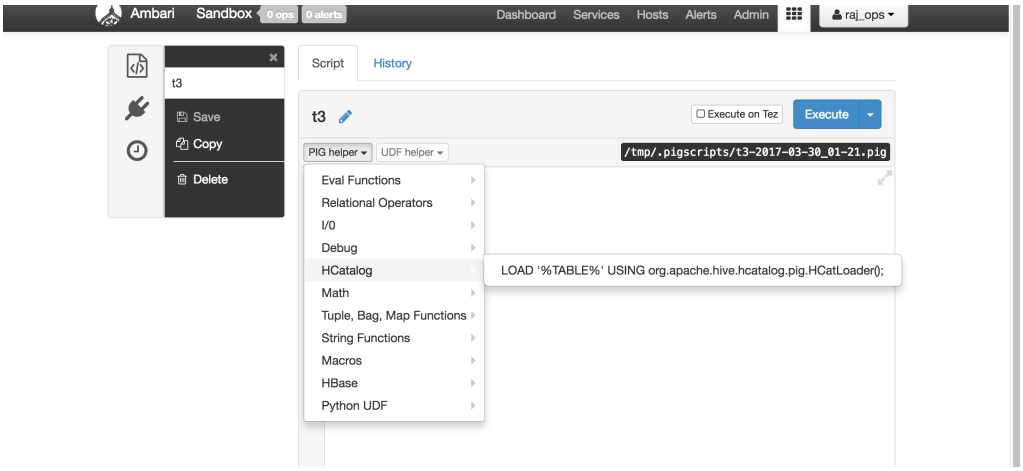

Figure 11: Choose HCatalog→LOAD

You code will be like:

```
teams_raw = LOAD 'basketball_teams'
USING org.apache.hive.hcatalog.pig.HCatLoader();
```

5. To filter out the first row of the data, which includes the table headers, we have to add this line:

```
teams = filter teams_raw by year > 0;
```

6. Describe the relation to see the columns and their data types, which should be consistent with the schema while creating the table "basketball_teams". Our completed line of code will look like:

```
teams_raw = LOAD 'basketball_teams'
USING org.apache.hive.hcatalog.pig.HCatLoader();
```

```
teams = filter teams_raw by year > 0;
DESCRIBE teams;
```

7. You need to configure the Pig Editor to use HCatalog so that the Pig script can load the proper libraries. In the Pig arguments text box, enter "-useHCatalog" and click the "Add" button or press "ENTER": Note that this argument is case sensitive. It should be typed exactly "-useHCatalog". Then it will appear under Arguments (see Fig. 12). Make sure you dont click Add twice (or press Enter twice) which adds an empty argument. Then, execute the script. The result should be seen as Fig. 13.

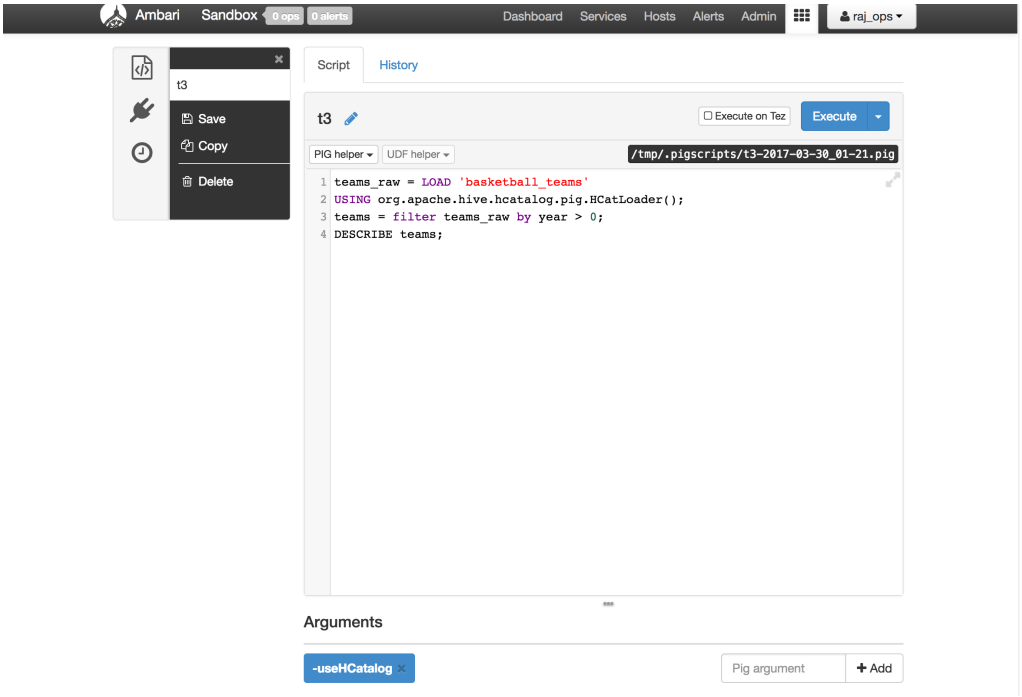

Figure 12: Add "-useHCatalog"

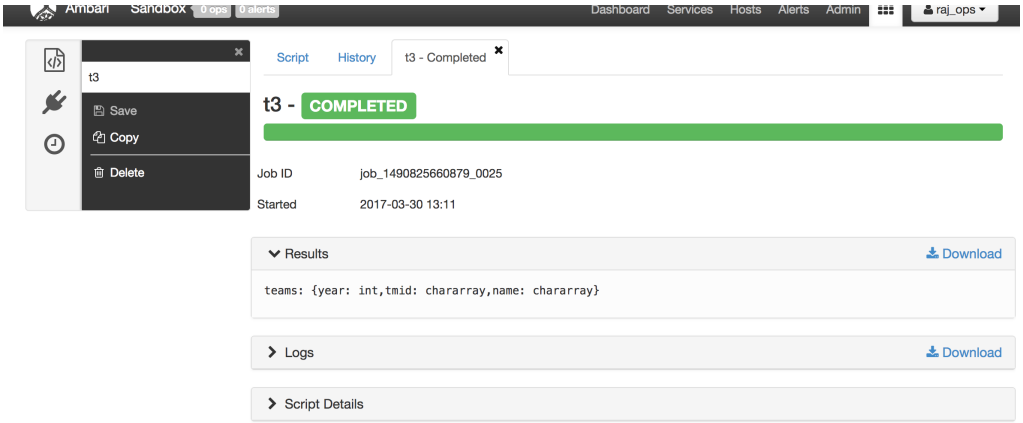

Figure 13: Result of using HCatalog

References If you need more practice or more information, use the following links:

- `http://hortonworks.com/hadoop-tutorial/how-to-use-basic-pig-commands/`
- `http://hortonworks.com/hadoop-tutorial/hello-world-an-introduction-to-hadoop-hcatalog-hive-and-pig/#section_6`
- `http://hortonworks.com/hadoop-tutorial/how-to-process-data-with-apache-pig/`