Getting Started with Apache Pig

About This Lab

Objective: To use Pig to navigate through HDFS and explore a dataset

Successful outcome: You will have a couple of Pig programs that load the White House visitors' data, with and without a schema, and store the output of a relation into a folder in HDFS

File locations: `~/data`

<!–STEP–>

<img src="https://user-images.githubusercontent.com/558905/40613898-7a6c70d6-624e-11e8-9178-7bde851ac7bd.png" align="left" width="50" height="50" title="ToDo Logo">
<h4>1. View the Raw Data</h4>

View the contents of the file `whitehouse_visits.txt`:

```
tail whitehouse_visits.txt
```

> *Note this publicly available data contains records of visitors to the White House in Washington, D.C.*

<!–STEP–>

<img src="https://user-images.githubusercontent.com/558905/40613898-7a6c70d6-624e-11e8-9178-7bde851ac7bd.png" align=“left” width=“50” height=“50” title=“ToDo Logo”>
<h4>2. Load the Data into HDFS</h4>

Start the Grunt shell:

```
pig
```

From the Grunt shell, make a new directory in HDFS named `whitehouse`:

```
grunt> mkdir whitehouse
```

Use the copyFromLocal command in the Grunt shell to copy the `whitehouse_visits.txt` file to the whitehouse folder in HDFS, renaming the file `visits.txt`. (Be sure to enter this command on a single line):

```
grunt> copyFromLocal whitehouse_visits.txt whitehouse/visi
ts.txt
```

Use the ls command to verify the file was uploaded successfully:

```
grunt> ls whitehouse

hdfs://[your ip]:8020/user/[user-name]/whitehouse/visits.t
xt<r 3>183292235
```

<!–STEP–>

<img src="https://user-images.githubusercontent.com/558905/40613898-7a6c70d6-624e-11e8-9178-7bde851ac7bd.png" align=“left” width=“50” height=“50” title=“ToDo Logo”>
<h4>3. Define a Relation</h4>

You will use the TextLoader to load the `visits.txt` file.

> *Note TextLoader simply creates a tuple for each line of text , and it uses a single chararray field that contains the entire line. It allows you to load lines of text and not worry about the format or schema yet.*

Define the following `LOAD` relation:

```
grunt> A = LOAD '/user/[user-name]/whitehouse/' USING Text
Loader();
```

Use `DESCRIBE` to notice that `A` does not have a schema:

```
grunt> DESCRIBE A;
Schema for A unknown.
```

We want to get a sense of what this data looks like. Use the `LIMIT` operator to define a new relation named `A_limit` that is limited to 10 records of `A`.

```
grunt> A_limit = LIMIT a 10;
```

Use the `DUMP` operator to view `A_limit relation`. Each row in the output will look similar to the following and should be 10 arbitrary rows from `visits.txt`:

```
grunt> DUMP A_limit;

...
(WHITLEY,KRISTY,J,U45880,,VA,,,,,10/7/2010 5:51,10/9/2010
10:30,10/9/2010
23:59,,294,B3,WIN,10/7/2010 5:51,B3,OFFICE,VISITORS,WH,RES
,OFFICE,VISITORS,GROUP TOUR ,1/28/2011,,,,,,,,,,,,,,,,,,,,
,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,, ,,,,,,,,,,
,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
```

<!-STEP->

<img src="https://user-

images.githubusercontent.com/558905/40613898-7a6c70d6-624e-11e8-9178-7bde851ac7bd.png" align=“left” width=“50” height=“50” title=“ToDo Logo”>

<h4>4. Define a Schema</h4>

Load the White House data again, but this time use the PigStorage loader and also define a partial schema:

```
grunt> B = LOAD '/user/[user-name]/whitehouse/visits.txt'
USING PigStorage(',') AS (
lname:chararray,
fname:chararray,
mname:chararray,
id:chararray,
status:chararray,
state:chararray,
arrival:chararray );
```

Use the `DESCRIBE` command to view the schema:

```
grunt> describe B;
B: {lname: chararray,fname: chararray,mname: chararray,id:
 chararray,status: chararray,state: chararray,arrival: cha
rarray}
```

<!–STEP–>

<img src="https://user-images.githubusercontent.com/558905/40613898-7a6c70d6-624e-11e8-9178-7bde851ac7bd.png" align="left" width="50" height="50" title="ToDo Logo">

<h4>5. The `STORE` Command</h4>

Enter the following `STORE` command, which stores the `B` relation into a folder named `whouse_tab` and separates the fields of each record with tabs:

```
grunt> store B into 'whouse_tab' using PigStorage('\t');
```

Verify the `whouse_tab` folder was created:

```
grunt> ls whouse_tab
```

You should see two map output files.

View one of the output files to verify they contain the `B` relation in a `tab-delimited` format:

```
grunt> fs -tail whouse_tab/part-m-00000
```

Each record should contain 7 fields.

What happened to the rest of the fields from the raw data that was loaded from `whitehouse/visits.txt`?

<!–STEP–>

<img src="https://user-images.githubusercontent.com/558905/40613898-7a6c70d6-624e-11e8-9178-7bde851ac7bd.png" align=“left” width=“50” height=“50” title=“ToDo Logo”>
<h4>6. Use a Different Storer</h4>

In the previous step, you stored a relation using PigStorage with a tab delimiter. Enter the following command, which stores the same relation but in a JSON format:

```
grunt> store B into 'whouse_json' using JsonStorage();
```

Verify the `whouse_json` folder was created:

```
grunt> ls whouse_json
```

View one of the output files:

```
grunt> fs -tail whouse_json/part-m-00000
```

> *Note that the schema you defined for the B relation was used to create the format of each JSON entry:*

```
{"lname":"MATTHEWMAN","fname":"ROBIN","mname":"H","id":"U8
```

```
1961","status":"735 74","state":"VA","arrival":"2/10/2011
11:14"}
{"lname":"MCALPINEDILEM","fname":"JENNIFER","mname":"J","i
d":"U81961","status ":"78586","state":"VA","arrival":"2/10
/2011 10:49"}
...
```


Result

You have now seen how to execute some basic Pig commands, load data into a relation, and store a relation into a folder in HDFS using different formats.

You are finished!