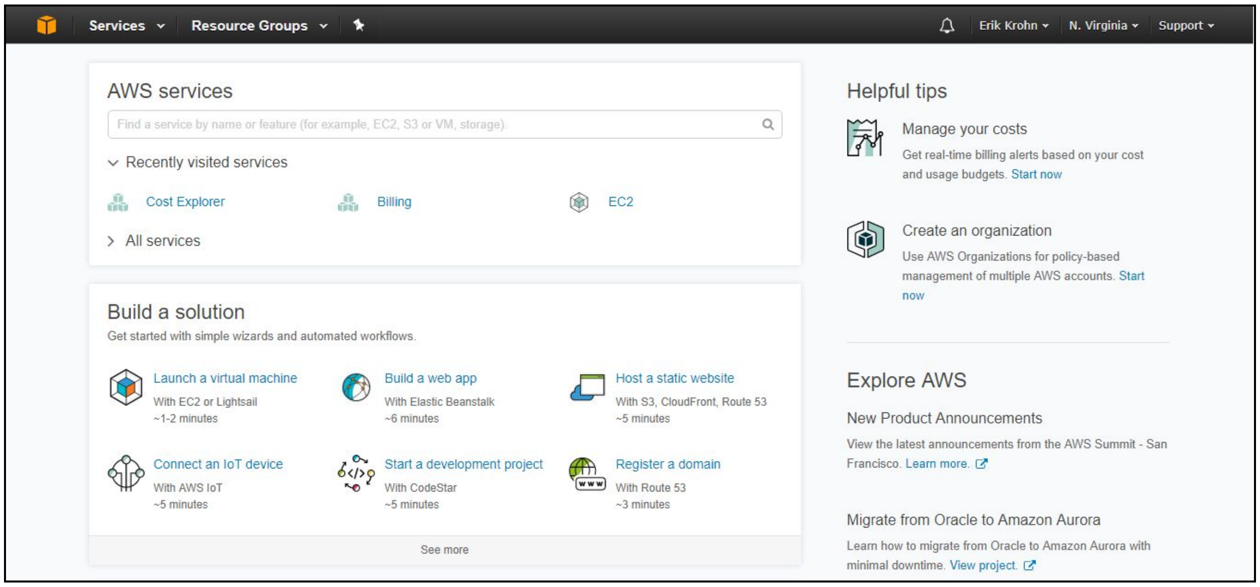
Hadoop and Cloud Computing

DS730

In this activity, you will be running some Python code using Amazon Web Services. You will also go through how to save and load data from the cloud and execute MapReduce in the cloud.

Task 1: Move Files into S3 Bucket

1. Retrieve your word count program that you used on War and Peace from the previous activity. I will assume you have 3 files:
   - mapper.py
   - reducer.py
   - wap.txt
2. Go to http://aws.amazon.com and log in with your username and password.
   - Currently, you hover over **My Account** and then click **AWS Management Console**.
3. Confirm you see something that looks like this:



AWS services

Find a service by name or feature (for example, EC2, S3 or VM, storage).

Recently visited services

Cost Explorer Billing EC2

All services

Build a solution

Get started with simple wizards and automated workflows.

Launch a virtual machine
With EC2 or Lightsail
~1-2 minutes

Build a web app
With Elastic Beanstalk
~6 minutes

Host a static website
With S3, CloudFront, Route 53
~5 minutes

Connect an IoT device
With AWS IoT
~5 minutes

Start a development project
With CodeStar
~5 minutes

Register a domain
With Route 53
~3 minutes

Helpful tips

Manage your costs
Get real-time billing alerts based on your cost and usage budgets. Start now

Create an organization
Use AWS Organizations for policy-based management of multiple AWS accounts. Start now

Explore AWS

New Product Announcements
View the latest announcements from the AWS Summit - San Francisco. Learn more.

Migrate from Oracle to Amazon Aurora
Learn how to migrate from Oracle to Amazon Aurora with minimal downtime. View project.

Our first goal is to get the Python files and our input into an S3 bucket. On our Hadoop setup, we had to send files to the HDFS. In a similar fashion, we must create a “bucket” and upload our information to that bucket.

4. Click **Services** and do a search for **`S3`**
5. Click **S3** when you find it. S3 is scalable storage in the cloud.
6. Click the **Create Bucket** button.
7. When the dialog box appears, give your bucket a name.

Important:

Buckets *must be unique across all users*. Therefore, you cannot create a bucket called **`wordcount`** as someone else has probably taken it. Nor can you call yours **`aws-wordcount-test-ek`** because I am using that one right now. You can call it whatever you want as long as it’s unique.

8. Feel free to leave the region at the default region but be sure to make note of the region for a task later in this activity.
9. Click **Next**.
10. For properties, the default values are fine, click **Next**.
11. For permissions, the default values are fine, click **Next**.
12. Click **Create bucket.**
13. When the bucket has been created, click on it to open it.
14. Click **Create Folder**.
15. Create 3 folders: a **jobs** folder, a **logs** folder and an **input** folder. The default encryption setting of None is fine. Use the **Create Folder** option to make each one.
16. Click the **jobs** folder.

17. Upload your **mapper.py** and **reducer.py** files by clicking the **Upload** button and following the directions on how to upload.
    - You can leave in all of the default options.

18. When finished, click your bucket name near the top (mine was called aws-wordcount-test-ek).

19. Click **input** to upload the **wap.txt** file.

Task 2: Configure AWS Cluster

We are now ready to start our first AWS project.

Important:

Be sure to complete all of the following steps at once. You do not want to start a cluster and leave it running. Make sure you finish all steps and shut down your cluster.

1. Click on **Services** and search for **EMR**.
2. Click **EMR** (Managed Hadoop Framework).
3. Check the region in the upper right hand corner. Make sure that the region is identical to the region in Task 1, step 8.
4. Click the **Create Cluster** button.
5. Here you'll see many options to create the cluster. Most of the defaults are sufficient but I'll explain each here. The **Name** doesn't matter; call it what you want.
6. Keep **Logging** checked.
7. Change the logging folder to be your **logs** folder created in Task 1, Step 15.
8. For the launch mode, use **Cluster**.
9. Everything under **Software configuration** can be kept at the default values.
    - As of this writing, the current version is **emr-5.17.0** and **Core Hadoop**.

10. Everything under **Hardware configuration** can be kept at the default: **m3.xlarge** and **3** instances.

    - My default region is the N. Virginia region. If you are using a different region, you may have a different default configuration. For example, in the Ohio region, the current default is `m4.large` and `3` instances. Whatever the default configuration is, you can leave it at that. It is possible that the instance type configuration is not available/supported in your region. If this is the case, your instance will terminate immediately. If this happens, go back to step 4 and repeat. When you reach this step, use a different instance type until you find one that works.

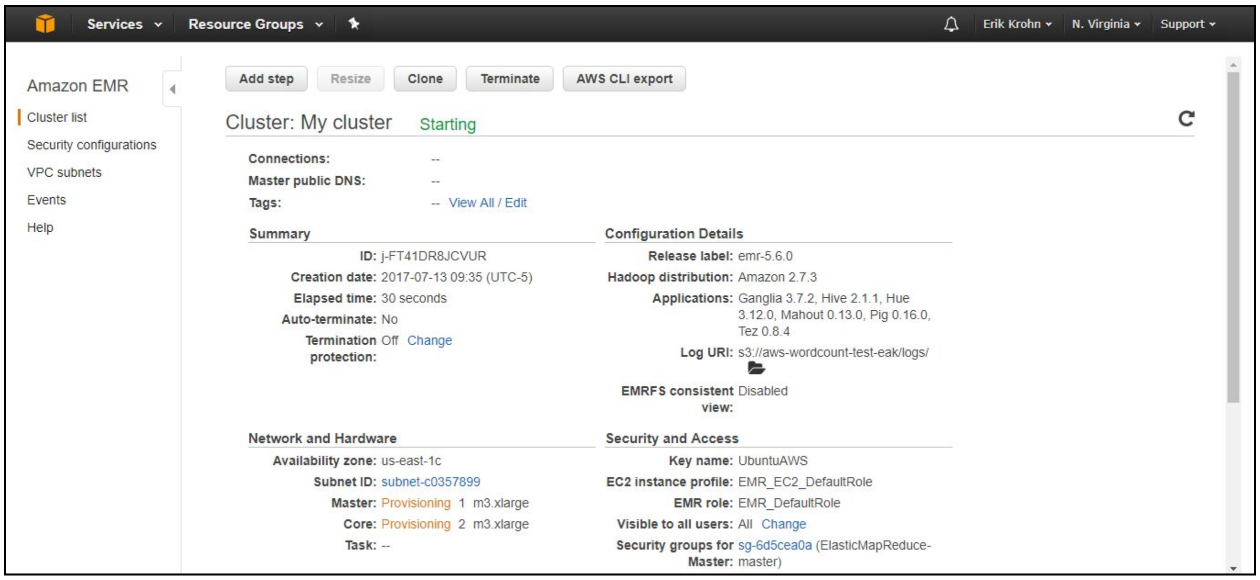
    - You should read about the different instances depending on the problem you are trying to solve. If you use a lot of storage, memory or CPU time, you might want to choose a specific instance for your problem.

11. For the EC2 key pair, choose the key you created in the first activity. It is likely the last option in the dropdown box.

    - If you wanted to connect to this cluster from your local machine, you would need to use this key to connect to it. For our purposes, this part does not matter. I selected **Proceed without an EC2 key pair** as I won't be connecting to this cluster.

12. The other options can be left at their default settings. Click **Create cluster**.

13. Confirm you see something like this:



Cluster: My cluster Starting

Connections: --
Master public DNS: --
Tags: -- View All / Edit

Summary
ID: j-FT41DR8JCVUR
Creation date: 2017-07-13 09:35 (UTC-5)
Elapsed time: 30 seconds
Auto-terminate: No
Termination protection: Off Change

Configuration Details
Release label: emr-5.6.0
Hadoop distribution: Amazon 2.7.3
Applications: Ganglia 3.7.2, Hive 2.1.1, Hue 3.12.0, Mahout 0.13.0, Pig 0.16.0, Tez 0.8.4
Log URI: s3://aws-wordcount-test-eak/logs/
EMRFS consistent view: Disabled

Network and Hardware
Availability zone: us-east-1c
Subnet ID: subnet-c0357899
Master: Provisioning 1 m3.xlarge
Core: Provisioning 2 m3.xlarge
Task: --

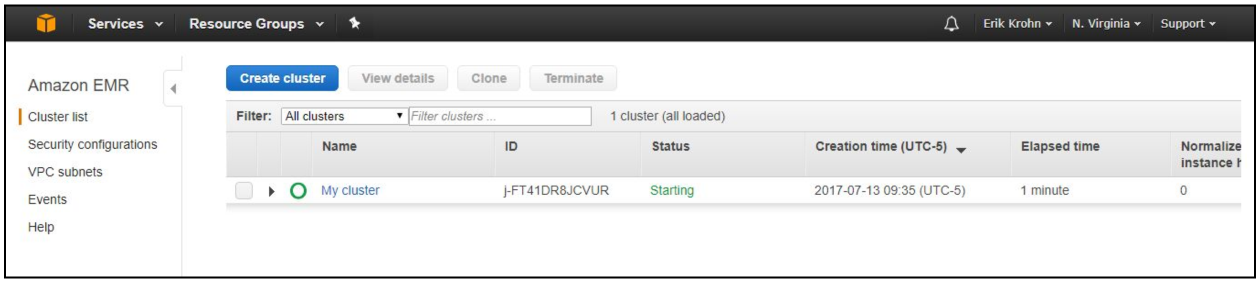
Security and Access
Key name: UbuntuAWS
EC2 instance profile: EMR_EC2_DefaultRole
EMR role: EMR_DefaultRole
Visible to all users: All Change
Security groups for Master: sg-6d5cea0a (ElasticMapReduce-master)

14. Your cluster is starting.

- In testing this during the semester, it sometimes took 45 minutes to an hour for the cluster to start. While I was testing it on a weeknight in the middle of summer, it took a couple of minutes.
- You can click the reload button to the right (↻) to see if anything changed.

15. While you wait, click the **Clusters** link on the left hand side.

16. Confirm you see something like this:



			Name	ID	Status	Creation time (UTC-5)	Elapsed time	Normalized instance h
			My cluster	j-FT41DR8JCVUR	Starting	2017-07-13 09:35 (UTC-5)	1 minute	0

17. Click your cluster name again and you will likely still be in the Starting state.

- Eventually, you will see Running or Waiting in the middle of your screen and you know your cluster is ready to go.

18. Click on the **Steps** tab and then the **Add step** button.

19. For the **Step type**, choose Streaming program.

20. The **Name** can be anything.

- I just called mine **`WordCount`**

21. For **Mapper**, click the folder icon and find your mapper.py file in your S3 folder.

22. Similarly, use the **Reducer** option to find your reducer.py file.

Important:

Make sure your mapper creates (key,value) pairs that are separated by a tab. Other delimiters may work but a tab definitely works. If you do not separate them by a tab, then the sorting part of MapReduce does not know what is the key and what is the value.

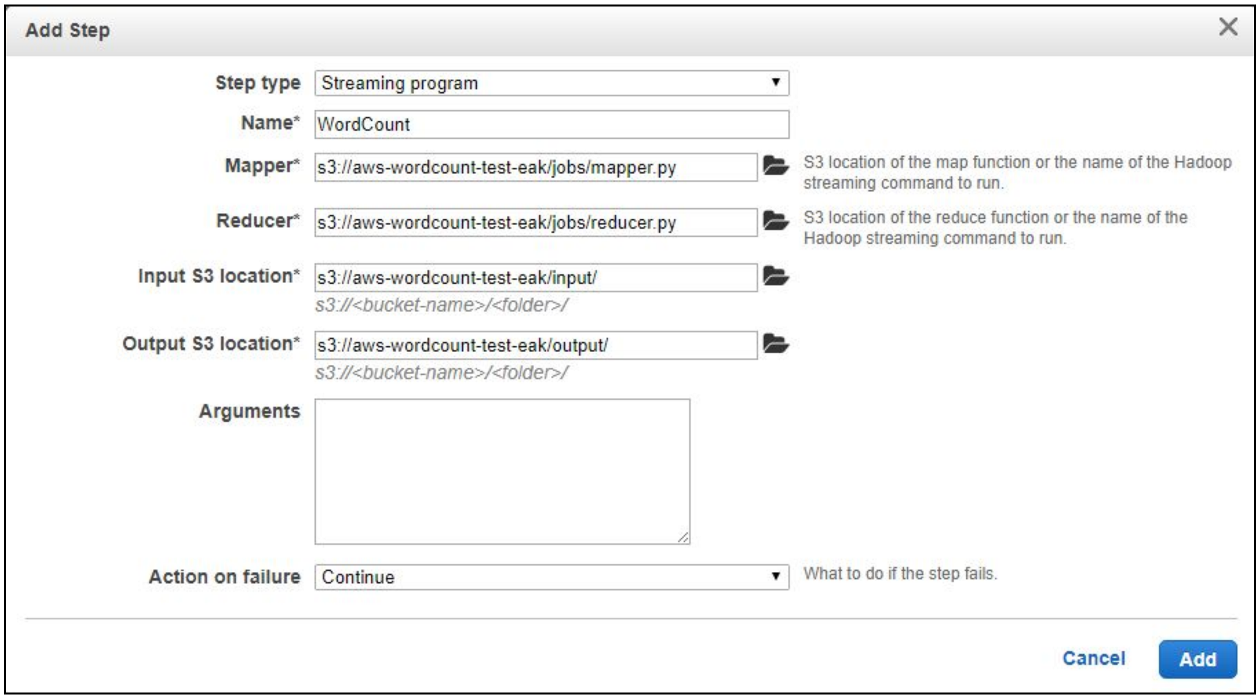
23. For the **Input S3 location**, choose your input S3 folder

- Do not choose the **wap.txt** file; choose *the entire input folder*.

24. For the **Output S3 location**, choose an S3 folder that you haven't created yet!

- The program will fail if you choose an output folder that already exists.

25. The remaining options (**Arguments**, **Action**) can be left alone. Confirm your step looks something like this:

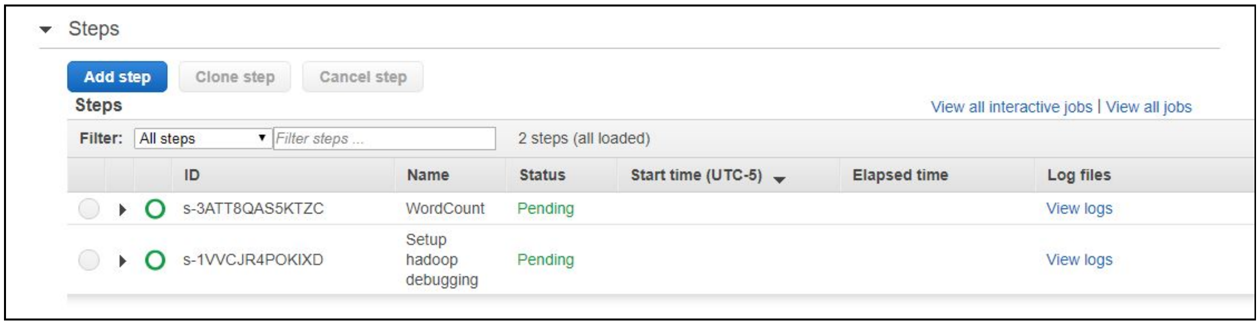


Step type	Streaming program
Name*	WordCount
Mapper*	s3://aws-wordcount-test-eak/jobs/mapper.py — S3 location of the map function or the name of the Hadoop streaming command to run.
Reducer*	s3://aws-wordcount-test-eak/jobs/reducer.py — S3 location of the reduce function or the name of the Hadoop streaming command to run.
Input S3 location*	s3://aws-wordcount-test-eak/input/ (s3://<bucket-name>/<folder>/)
Output S3 location*	s3://aws-wordcount-test-eak/output/ (s3://<bucket-name>/<folder>/)
Arguments	
Action on failure	Continue — What to do if the step fails.

26. Click **Add**.

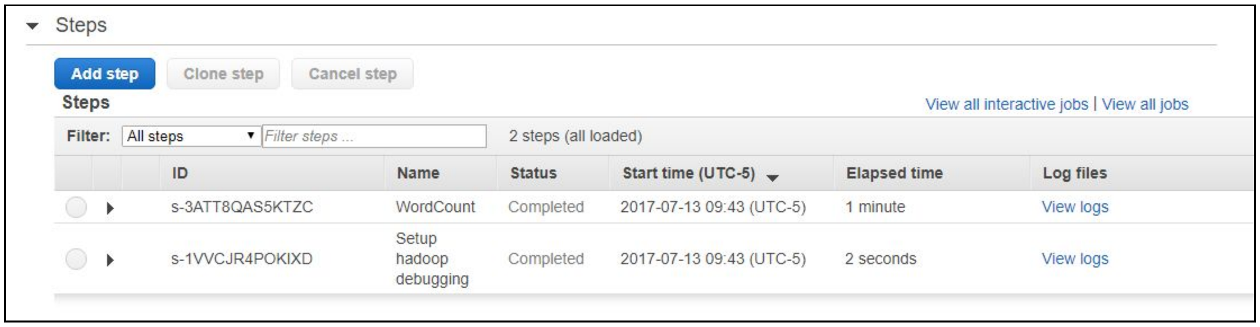
27. In the **Steps** tab, you will see the steps that have run or are going to run.

- Since my instance was still starting up, my step was Pending. It will eventually say Running.



ID	Name	Status	Start time (UTC-5)	Elapsed time	Log files
s-3ATT8QAS5KTZC	WordCount	Pending			View logs
s-1VVCJR4POKIXD	Setup hadoop debugging	Pending			View logs

28. Once the instance has started up, if you click the refresh icon to the right, it will show your program as Running instead of Pending. When the program finishes, confirm you see the **Status** change to Completed (you may have to click the refresh icon):



ID	Name	Status	Start time (UTC-5)	Elapsed time	Log files
s-3ATT8QAS5KTZC	WordCount	Completed	2017-07-13 09:43 (UTC-5)	1 minute	View logs
s-1VVCJR4POKIXD	Setup hadoop debugging	Completed	2017-07-13 09:43 (UTC-5)	2 seconds	View logs

Note:

If your WordCount program failed, then something obviously went wrong. If something went wrong, click the **View logs** to see the log files. The syslog file generally provides an answer on what went wrong. Some common mistakes include:

- the output folder already existed.
- the input folder was not defined correctly.
- the mapper or reducer was not created using Linux newlines and has bad characters in it.
- the mapper or reducer has errors in it, syntax, indentation, etc.

The stderr and stdout might tell you what went wrong so they are worth looking at.

29. Assuming your job has completed successfully, click the **View logs** link if it is there.
    - It may say **controller | syslog …** already depending on when you clicked on certain things.

30. Copy everything in the **controller** file and the **syslog** file and store them into a file called **wapLogs.txt**. You will submit this at the end of the activity.

31. Congratulations, you ran your first MapReduce program in the cloud. The last thing we want to do is shut down our cluster. To do this, scroll up and click the **Terminate** button.

32. It will confirm with you that you want to terminate the cluster. Click the **Terminate** button.

33. Click the **Clusters** link on the left hand side of the screen. Under status, it should say **Terminating User request**.

34. Wait a few minutes and make sure the cluster terminates. Eventually the status will change to Terminated.

35. Click the AWS icon in the upper left hand corner to go back to the main screen.

Task 3: Get Your Output

Use *one* of the following two methods. You don't need to perform both.

Option 1: Get Output Through the Brute Force Method

1. The brute force way of getting your output is to first click the **S3** icon.

2. Click your bucket.

3. Click your output folder.

4. Left click on each *part* file and then click **Download**.

    - You have to do this for each part file. As of this writing, I wish there was a better way to do this with the web interface, currently there is not.

Option 2: Get Output by Transferring Files

A more clever way to transfer files is to sync our S3 folder with the virtual machine.

5. You will start by creating a user who can access our S3 bucket. With AWS open, go up to **Services**.

6. Search for and click **IAM**.

7. On the left, click the **Users** option.

8. Click the **Add User** button.

9. Pick a username. It can be anything, just remember it.

10. Under Access type, choose **Programmatic access**.

11. Click **Next: Permissions**.

12. Click **Attach existing policies directly**.

13. Search for **`AmazonS3FullAccess`**. Check that box, click **Next: Review**.

14. Click the **Create user** button. Save your **Access Key ID** and **Secret Access Key**.

Important:

Do not leave this page without saving those values somewhere as the **Secret Access Key** can never be retrieved again.

15. After you have saved that information, click **Close**.

Now we want to add this user to our Hortonworks system so we can download and upload files automatically.

16. Log into your Hortonworks machine by going to http://localhost:4200 and using the maria_dev username/password, type in this command:
    ```
    sudo yum install awscli
    ```

17. When the system asks you about whether it's ok to install, enter **`Y`**

18. Enter **`aws configure`** and wait for the prompt.

19. For **Access Key ID**, enter the access key ID you just saved and then press **ENTER**.

20. Similarly, enter the **Secret Access Key**.

21. For the **Default region name**, first recall the region name you noted in Task 1, Step 8. Then, look up your region on this page: http://docs.aws.amazon.com/general/latest/gr/rande.html#apigateway_region

Example:

If you chose US West (N. California), the region you would enter for this step would be **us-west-1**.

22. At the **Default output format** prompt, just press **ENTER**.

23. Make a folder called WordCount by entering:
    ```
    mkdir WordCount
    ```

24. Go into that folder by entering:

    ```
    cd WordCount
    ```

25. Once you are in the WordCount folder, you are ready to download from the S3 bucket. Type in the following command changing it to be your bucket name and the folder that you want to download:

    ```
    aws s3 sync s3://name-of-your-bucket-here/folderName .
    ```

    - Do not forget the period at the end of that command. This will download all files from your bucket to the WordCount folder.

Note:

If you wish to upload files, swap the bucket name and period:

```
aws s3 sync . s3://name-of-your-bucket-here/folderName
```

26. Your files are now located in the WordCount folder and you can do with them what you want. Transfer them to your HDFS so that you can easily download them from the Ambary Files View. To transfer them to HDFS, use the following commands:

    ```
    hdfs dfs -mkdir /user/maria_dev/s3Test

    hdfs dfs -copyFromLocal * /user/maria_dev/s3Test
    ```

27. Go to your Ambari File View and your files will be located in /user/maria_dev/s3Test. Download all of the files in that folder and submit them at the end.

Task 4: Combine Part Files

You'll note that there are several output files for this problem. Because we (likely) had multiple reducers running on multiple machines, we ended up with an answer stored in several files. This is not really something we want to change. We could force our MapReduce program to only use one reducer, but this doesn't scale well and it defeats the purpose of distributing our work across multiple machines. It's best to produce multiple output files and deal with them after the fact. Combining a lot of small output files is going to, in general, be much faster than forcing Hadoop to only use one reducer.

Our solution is going to be to read in all values from the output files, sort them and produce a single output file. This is acceptable because our output files are likely considerably smaller than our original input files. Sorting a very small number of output files/data is cheap compared to sorting our original massive input files.

A simple solution to combining all of the part files and sorting them is to use the following command at the command prompt on your Hortonworks command line:

```
cat part* | sort > sorted.txt
```

A similar command in Windows is the following:

```
type part* 2>nul | sort > sorted.txt
```

Submitting Your Work

When you are finished, zip up the following files and upload the resulting `a3.zip` file to the dropbox:

- From Task 2: **wapLogs.txt**
- From Task 3 option 1 or 2: the **part-******* files
- From Task 4: **sorted.txt**