Ryan Britton & Ashan Deen

ECE 4150

Professor Vijay Madisetti

Monday, April 15th

Project Report: Apache Spark

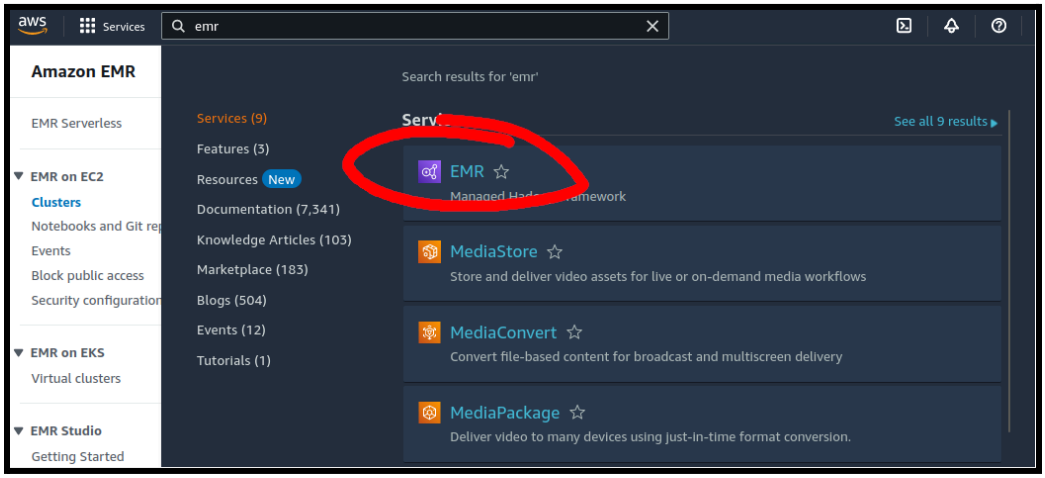
1. Introduction to Apache Spark

Apache Spark is a framework developed by contributors to the Apache Foundation designed for processing data at great scales. Some example use cases for Apache Spark could be as simple as dataset processing to complex as machine learning pipelines. Other notable use cases are for analytics and forecasting, such as for financial models, as well as image processing. Spark also supports the streaming of real time data, with integrations for Apache Kafka, TCP sockets, and more.

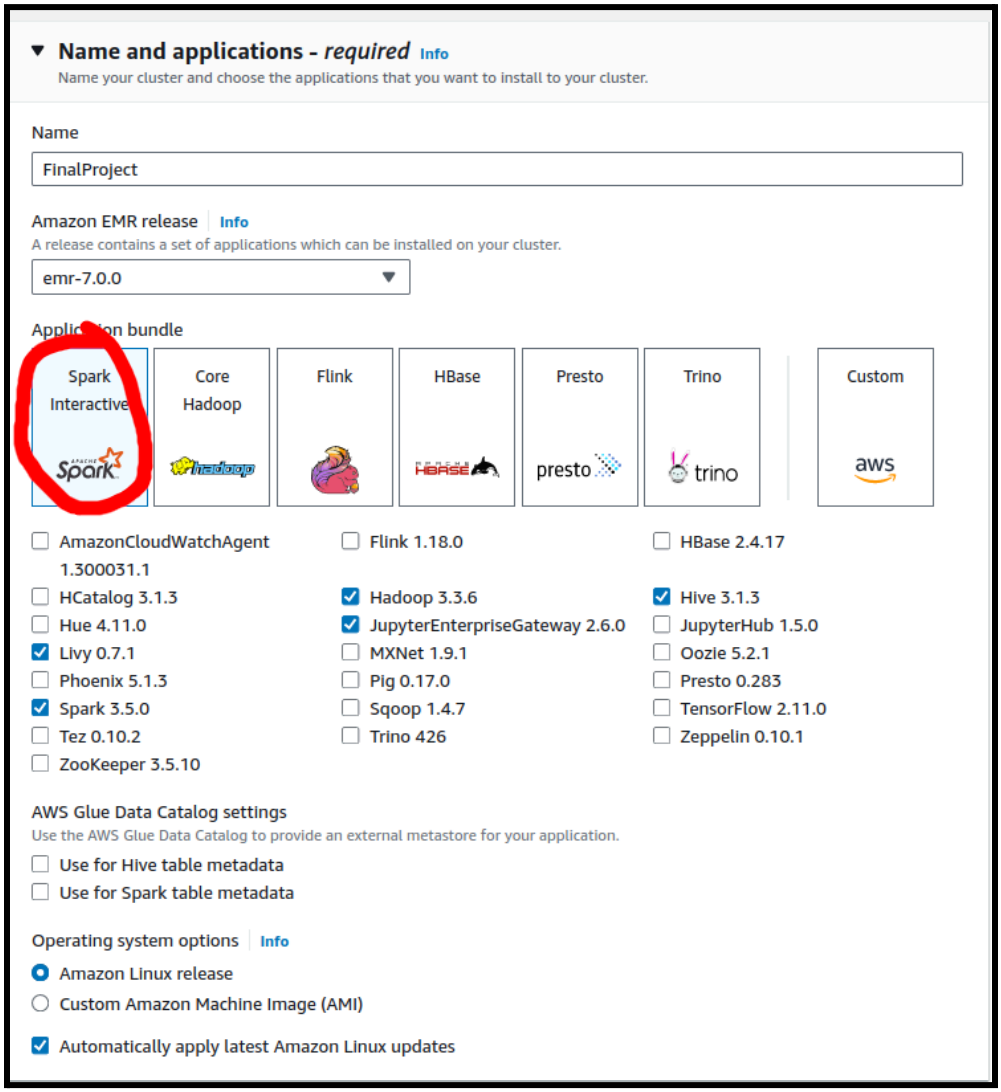
In this project, we will be performing data processing with Apache Spark by reading from a local file in our EMR cluster. Coincidentally, we will be counting the number of words in the book "The Adventures of Sherlock Holmes' ' by Arthur Conan Doyle, as well as counting the occurrences of letters of the alphabet.

2. Setting Up Apache Spark

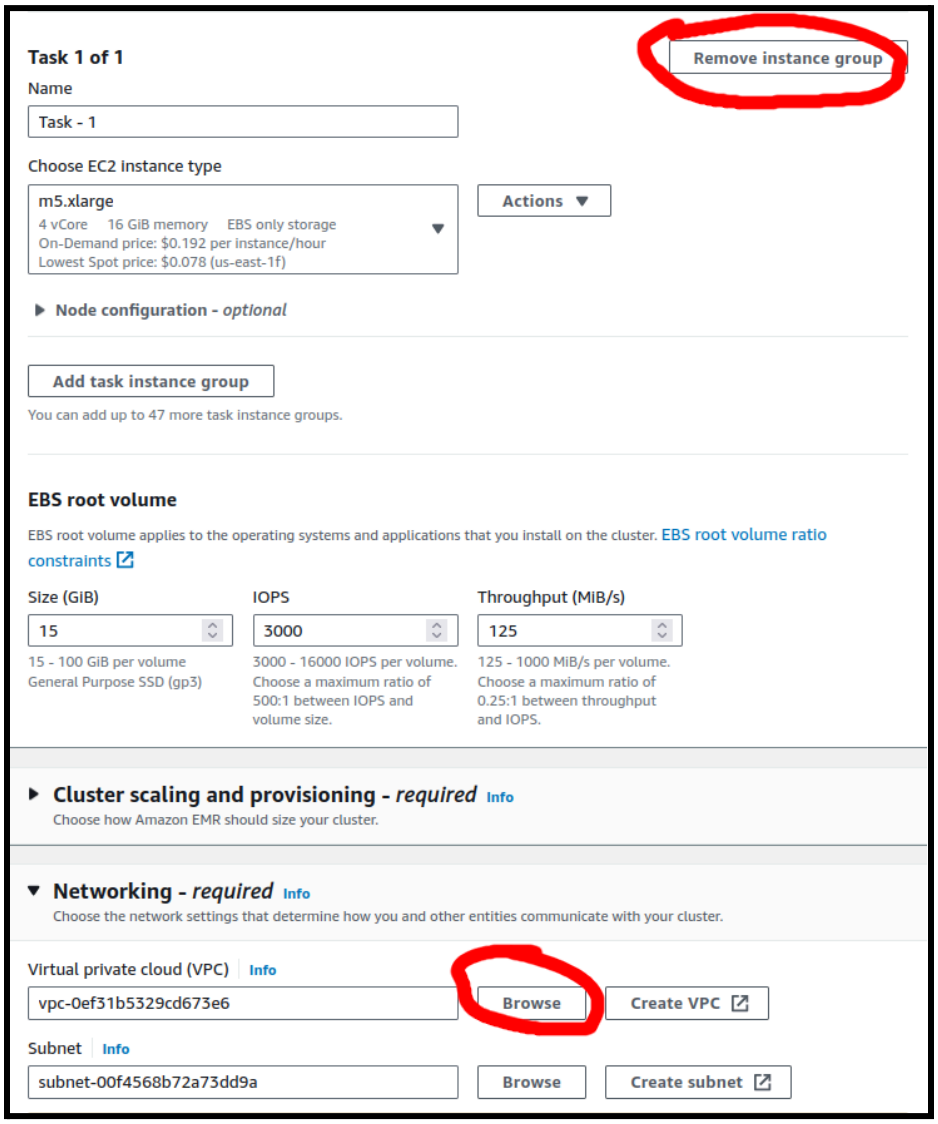
Beginning in the AWS console, navigate to Amazon EMR.



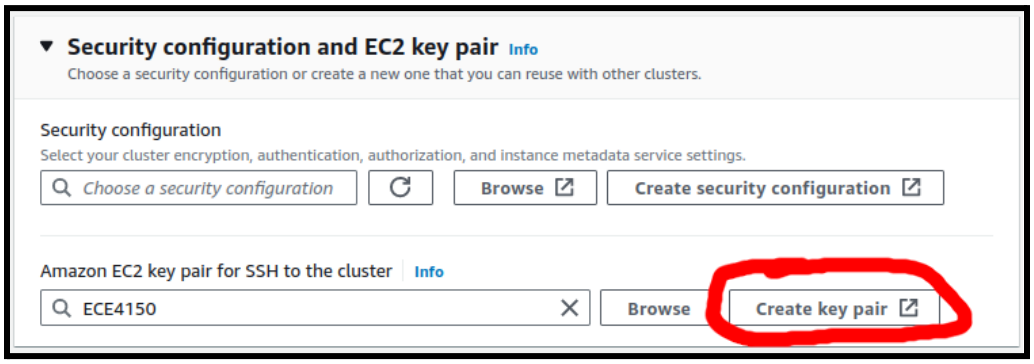
Next you should create a new cluster. Be sure that "Spark Interactive" is selected for the Application Bundle.



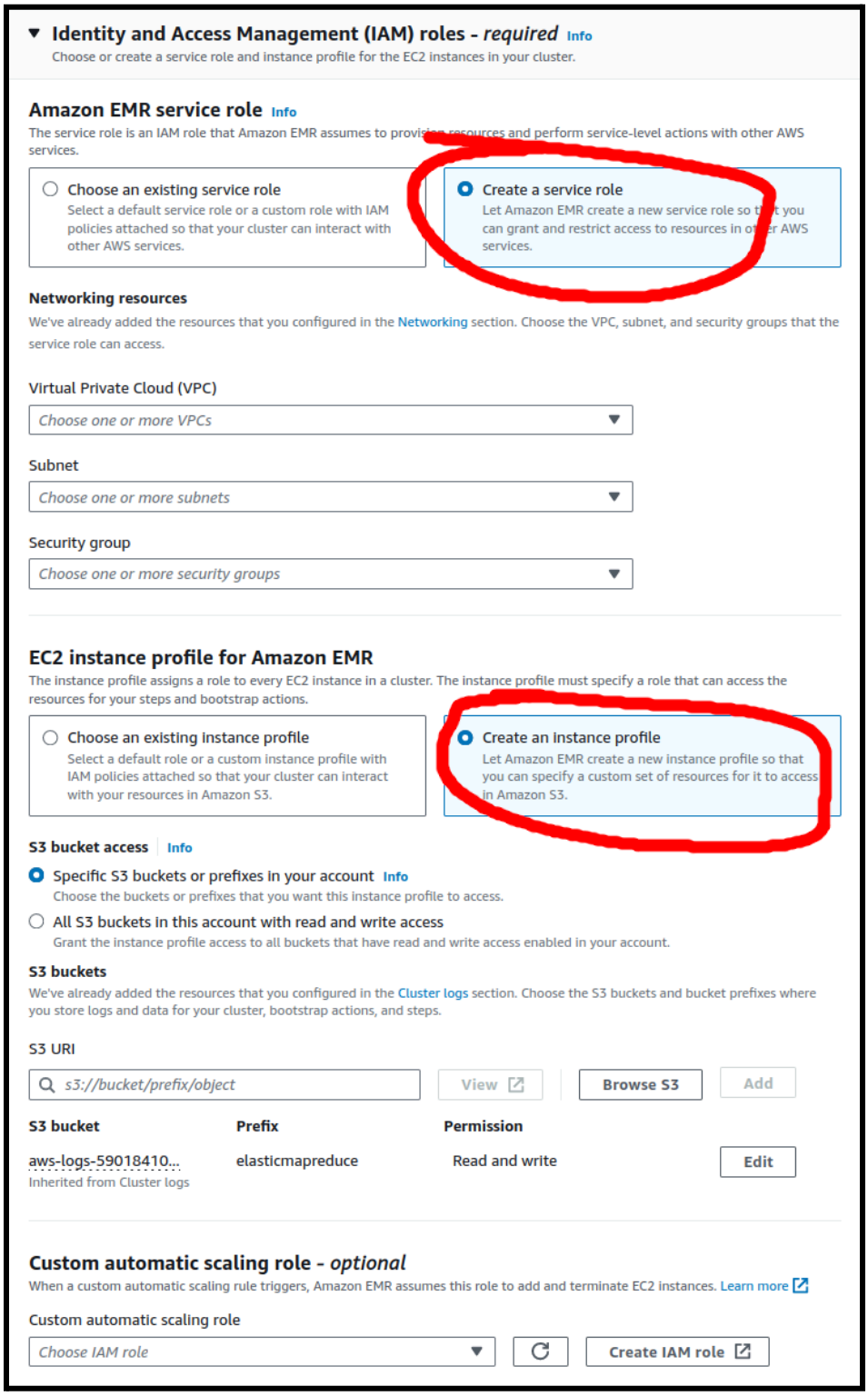
For the remaining options, you may delete "Task 1 of 1", and assign or create a new VPC. Be sure the subnet is public.



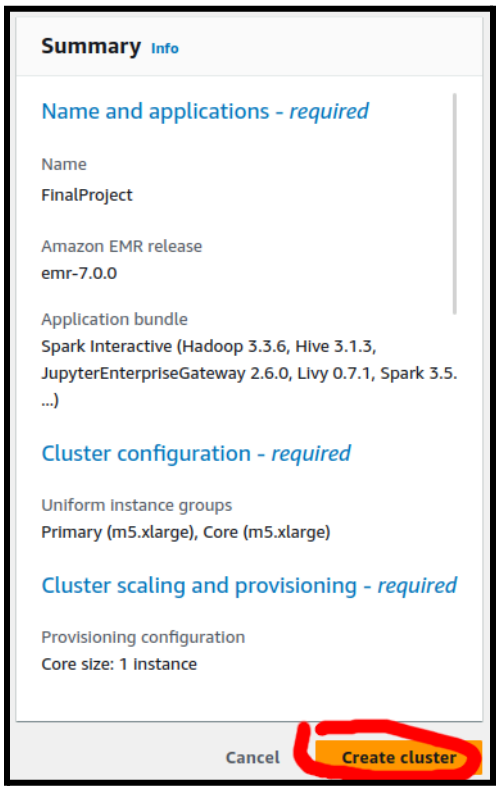
Next you must create or reuse an SSH key to access the cluster. Download this key to your local machine.



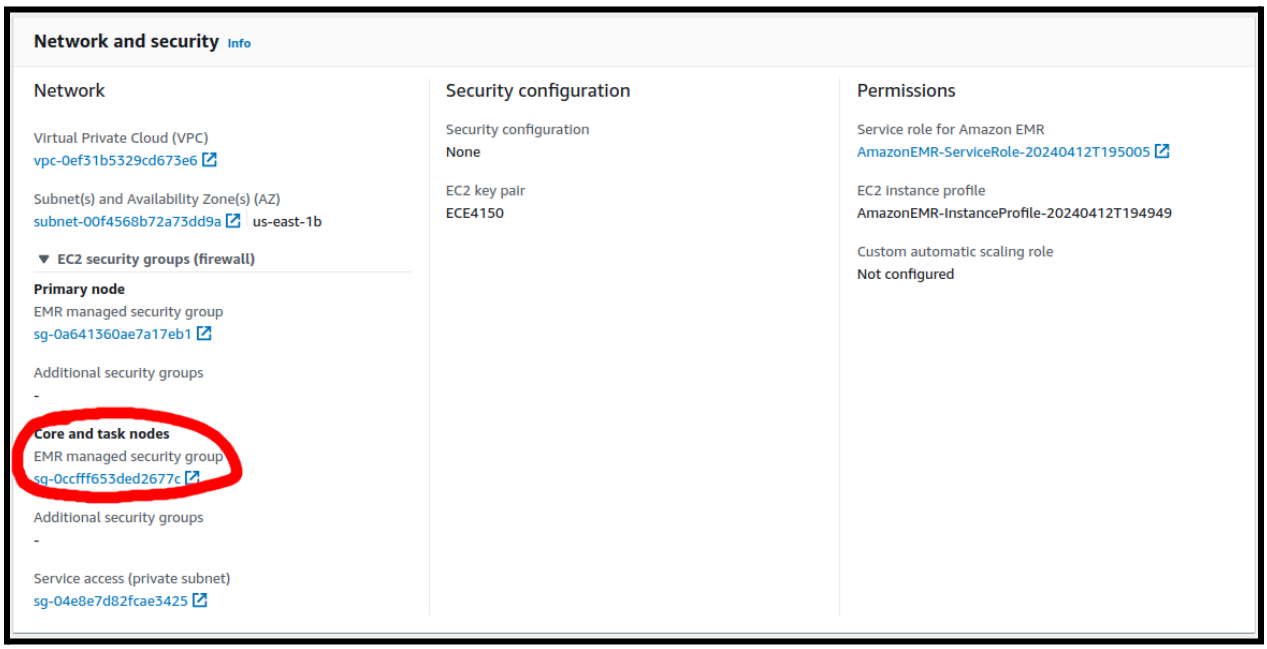
Finally, select "Create a service role" and "Create an instance profile"



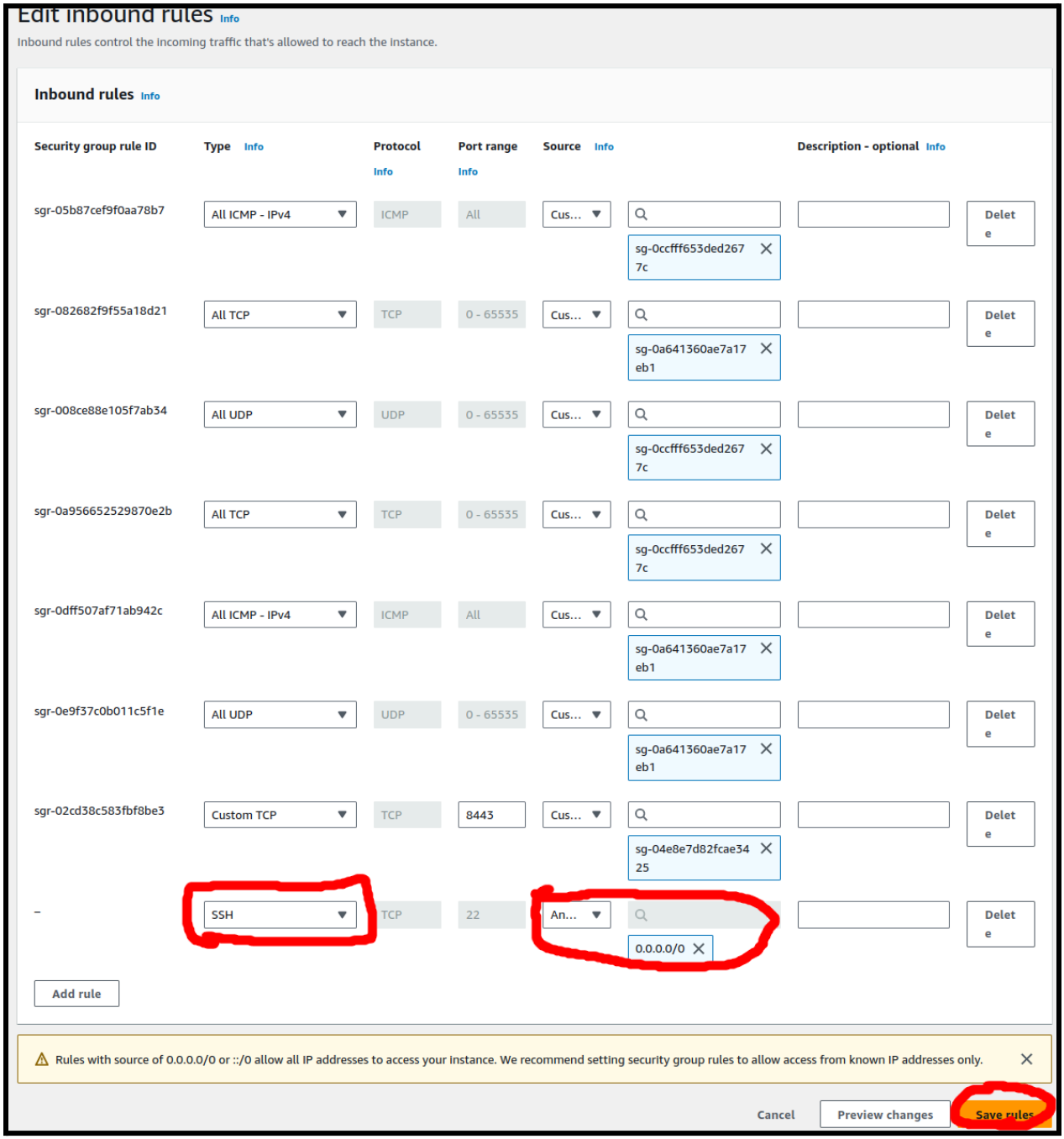
Now you can create the cluster. Ensure the options here match.



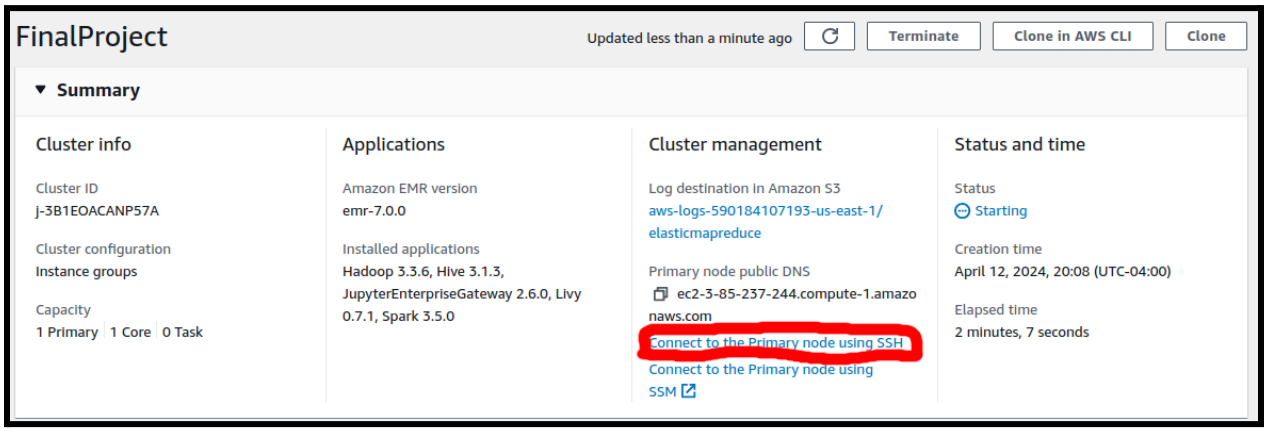
Once the EMR cluster is created, you must allow access to the from your IP address. Locate the "Core and task nodes - EMR managed security group" on the EMR cluster summary page.



For this security group, edit the inbound rules to create a new rule that allows SSH access from any IP.



Now that SSH access is permitted, connect to the primary node using the steps listed on the EMR cluster summary page.



3. Task 1: Counting Unique Words

For counting the words in the book, we will use a modified version of a sample python script provided by Apache in the Spark source code GitHub page: (https://github.com/apache/spark)

Once you have confirmed you can SSH into the cluster, use scp to copy the "book.txt" and "wordcount.py" from the task1 folder to the cluster.

```
((.venv) ) ~/git/ece4150-labs/finalproject $ scp -i ECE4150.pem -r resources/task1 hadoop@ec2-3-85-237-244.compute-1.amazonaws.com:~
book.txt                                                                    100%  581KB   2.5MB/s   00:00
wordcount.py                                                                100% 1418    47.0KB/s   00:00
((.venv) ) ~/git/ece4150-labs/finalproject $
```

Next ssh into the cluster and copy the dataset to hadoop, as that is how Spark in the EMR cluster has been configured to stream data from.

```
[hadoop@ip-10-0-8-250 task1]$ hadoop fs -copyFromLocal book.txt /user/hadoop/book.txt
```

Finally, submit the job to Apache Spark for processing

```
[hadoop@ip-10-0-8-250 task1]$ spark-submit wordcount.py book.txt
```

If that is successful, you may copy the results.txt file back to your computer for submission for task 1.

```
$ scp -i ECE4150.pem -r hadoop@ec2-3-85-237-244.compute-1.amazonaws.com:~/task1/results.txt resources/task1/
```

For completion, we've included results.txt, which should match the output results.txt.

4. Task 2: Counting Character Occurrences

For the second task, if actually a task, we would then ask the students to modify the wordcount.py script to count the occurrences of characters in the text. For the sake of simplicity and completion, we've provided characteroccurrances.py files in task2/, as well as the expected result. The file may be modified in EMR or modified locally and SCP'd later. We will use the same book.txt from the previous task. The output should be sorted by character, with the format of "Character: '{}', Count: {}" for each line.

```
scp -i ECE4150.pem -r resources/task2/ hadoop@ec2-3-85-237-244.compute-1.amazonaws.com:~
```

For completion, we've included results.txt, which should match the output results.txt.