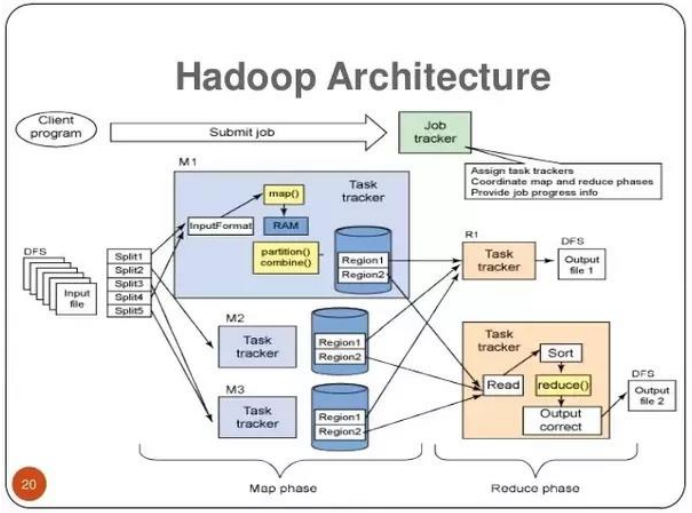
DB Management Systems
Distributed: Hadoop

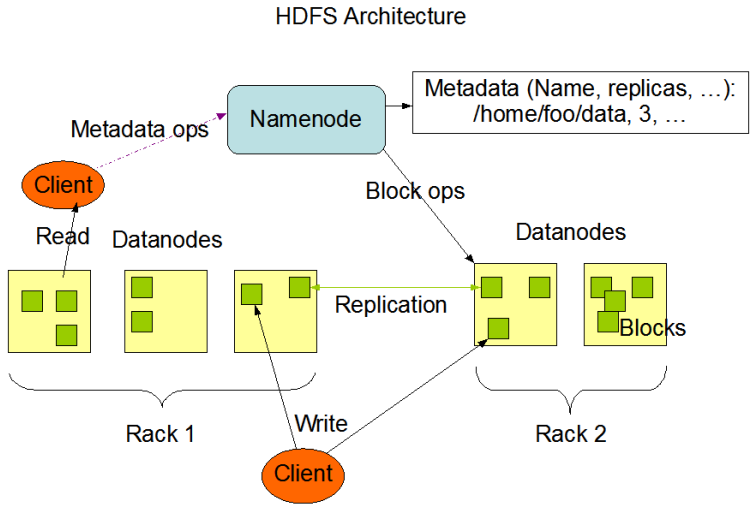
Joel Klein – jdk514@gwmail.gwu.edu

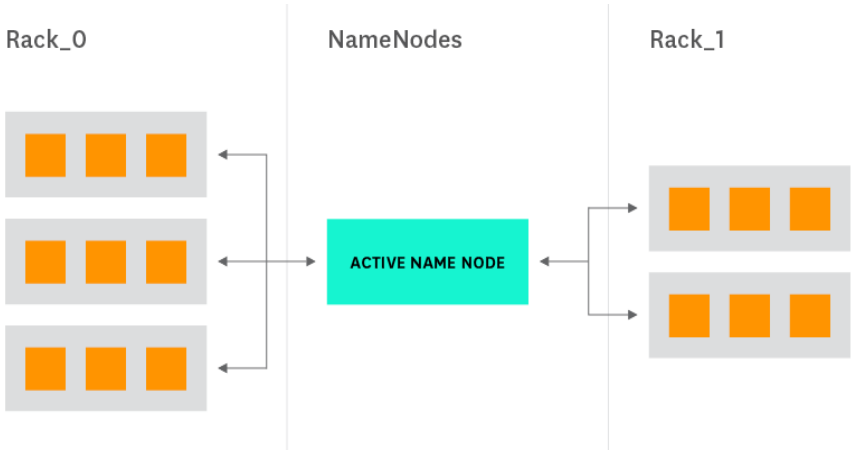
Hadoop Architecture

Hadoop Architectural Diagram

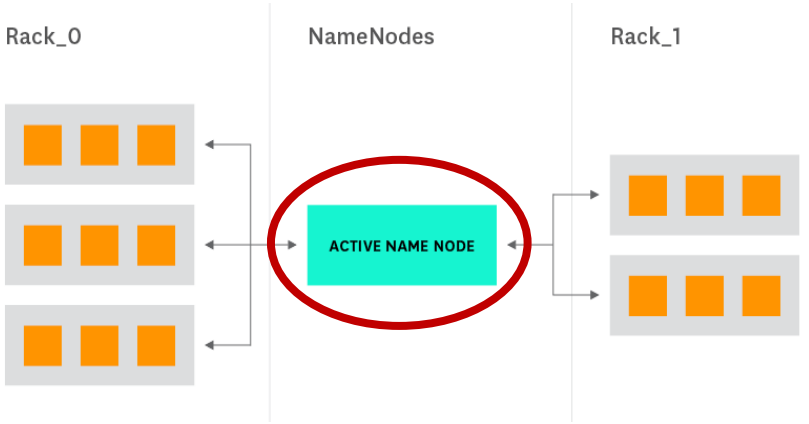
Complex → **Simple**





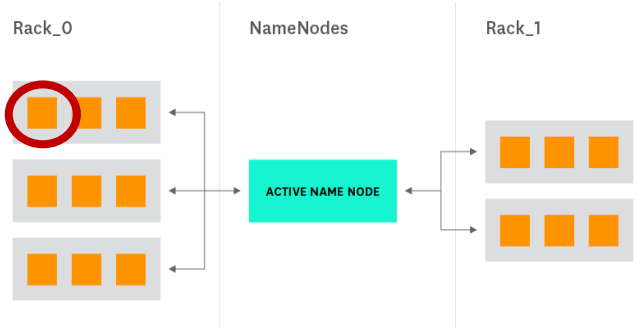


NameNode



- Think of this as the master node.
- It is the node a user actually interfaces with to input commands/configuration
- It doesn’t need to necessarily be a powerful machine (specs wise) as it doesn’t do any of the heavy lifting

DataNode



- DataNode's represent the true brute force behind the Hadoop framework.
- They are the nodes that both store and process the data used by Hadoop's MapReduce processes
  - DataNodes typically consume/store data through HDFS, but there are other options:
    - S3, FTP, local filesystem, Azure, Swift(??)
- These machines are usually beefier than the NameNode, as they need to be able to process large amounts of data quickly

MapReduce

- MapReduce isn't really a "component" of Hadoop, but rather the algorithm that makes Hadoop work.
- MapReduce works under two basic principles
  - Map a function across data on the datanodes
  - Reduce the results to create a composite answer that can be returned
- This is critical to distributed computing, as it removes the dependency on locality of the data being processed.

Uploading Data

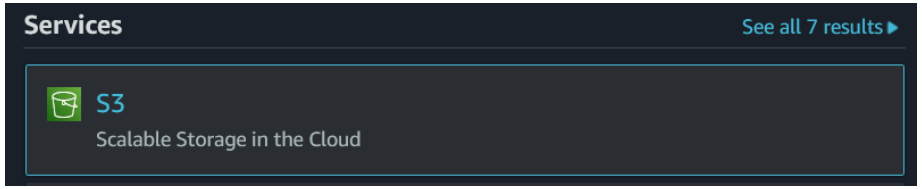
S3 (Simple Storage Service)

- For us to explore anything within the Hadoop framework, we'll need some data to work with.
  - In Blackboard there is a **ratebeer.zip** archive with the files we'll be uploading

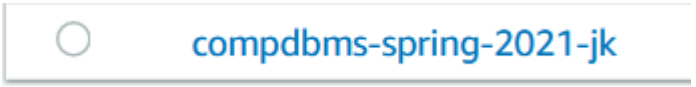
- S3 is effectively a cloud-based file directory. It allows us to store and access files for use within a range of AWS services.

- To create the correct environment, we'll need to setup a directory for use.

Initializing Our Bucket



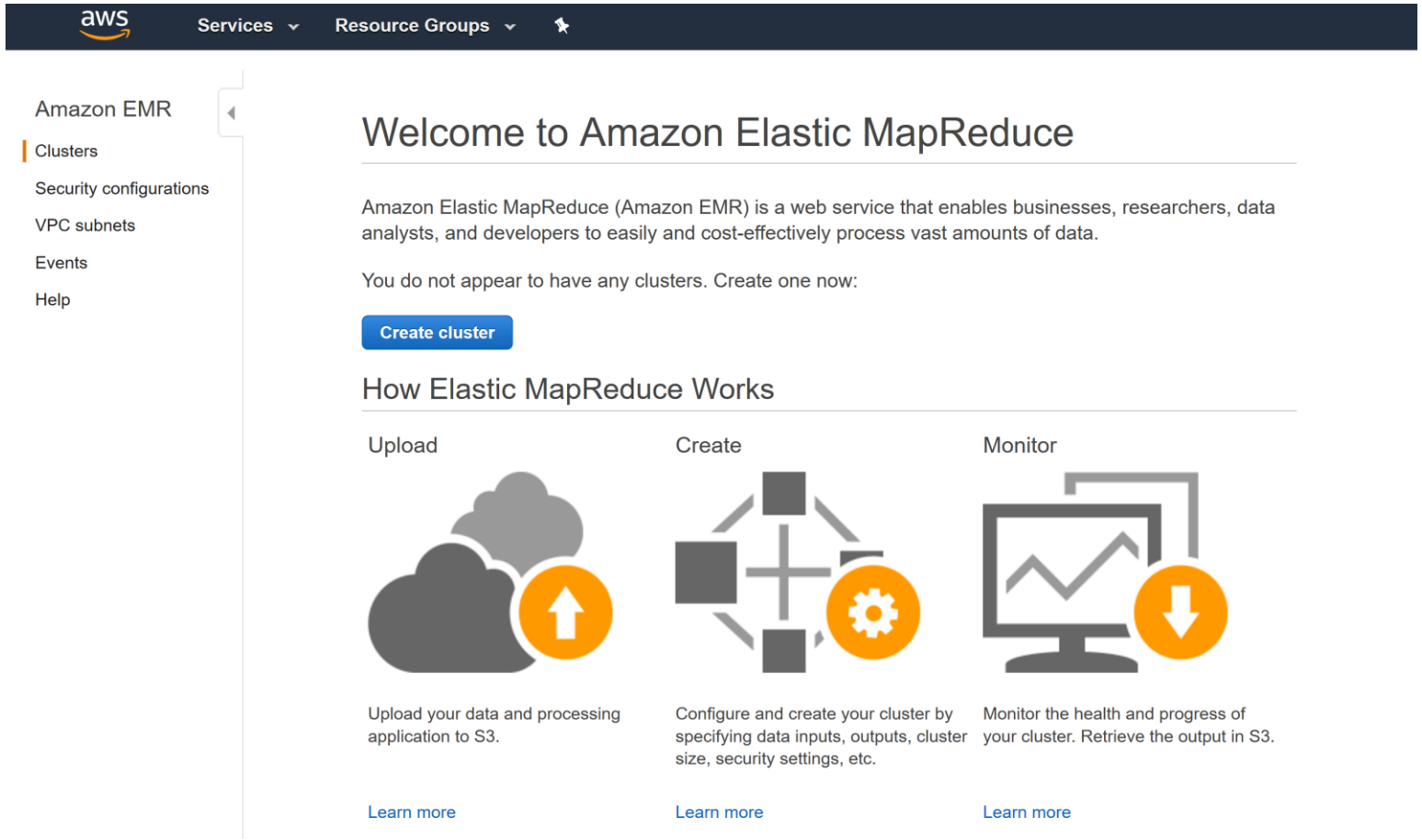
1. Navigate to the S3 service's landing page
2. Click **Create bucket**
3. Provide a Bucket Name
    1. Note: Bucket names must be AWS unique

4. Click **Create bucket** at the bottom of the page
5. Click on your bucket – 

6. Click **Upload**
7. Drag and drop your files

Hadoop on AWS

EMR (Elastic Map Reduce)



EMR cont.

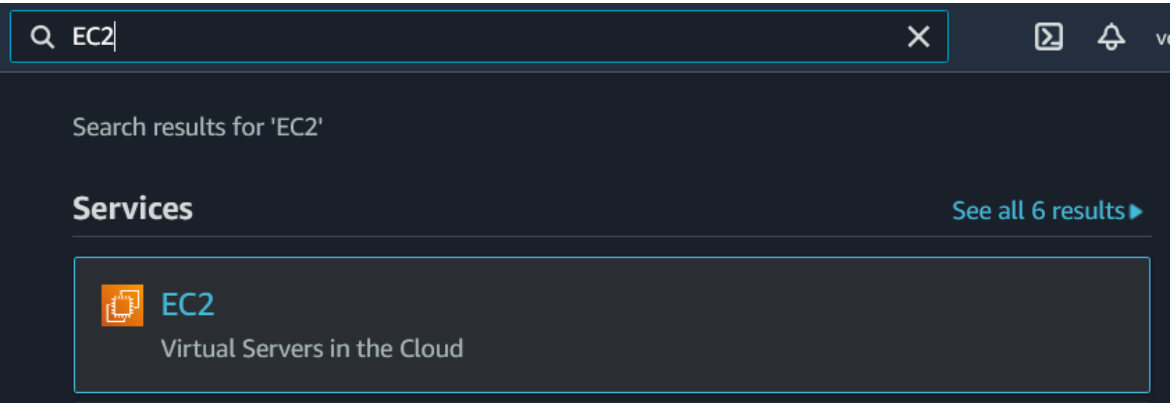
- EMR is amazon's answer to easily scalable and demandable Hadoop clusters
- Since Hadoop clusters can easily leverage AWS S3 storage, they focus on providing bursts of massive processing rather than more permanent infrastructure
- Thus, EMR relies heavily on creation scripts and automatic processing of clusters to make demand when needed and remove it when it is not.

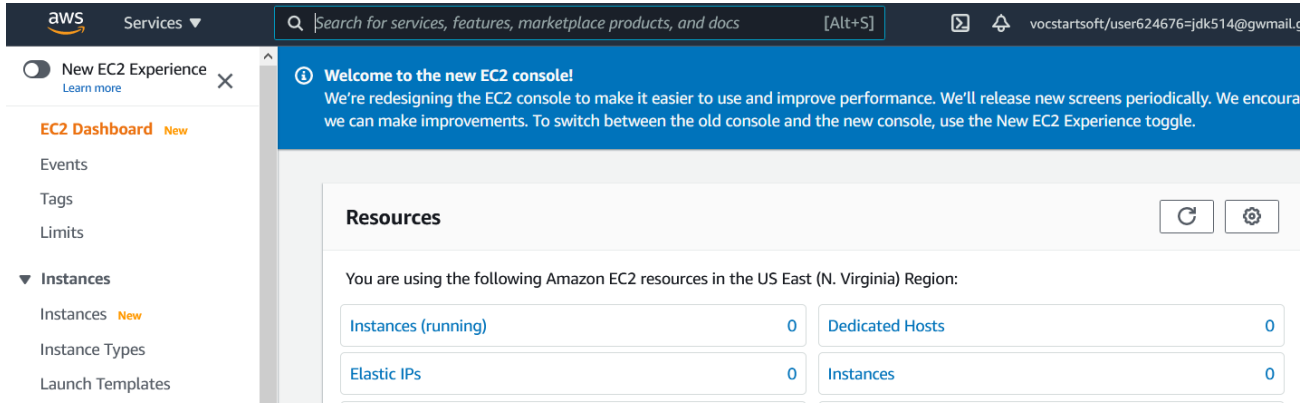
Security for EMR

Key-Pair

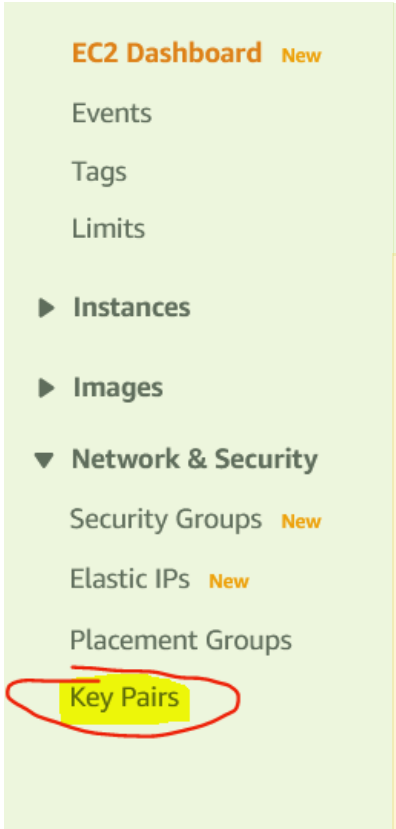
- To connect and work with our EMR cluster we'll need to SSH into the namenode
  - To accomplish this, we'll need to create a key-pair to authenticate our connection

- To start, we'll need to navigate to the EC2 service within our AWS console (accessed through the AWS Educate classroom).



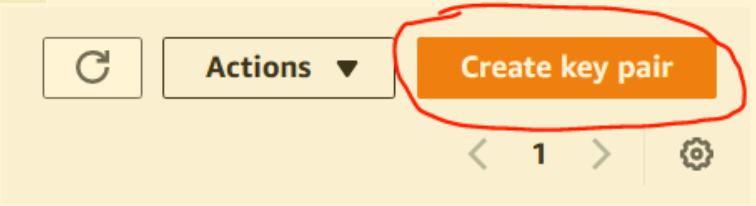


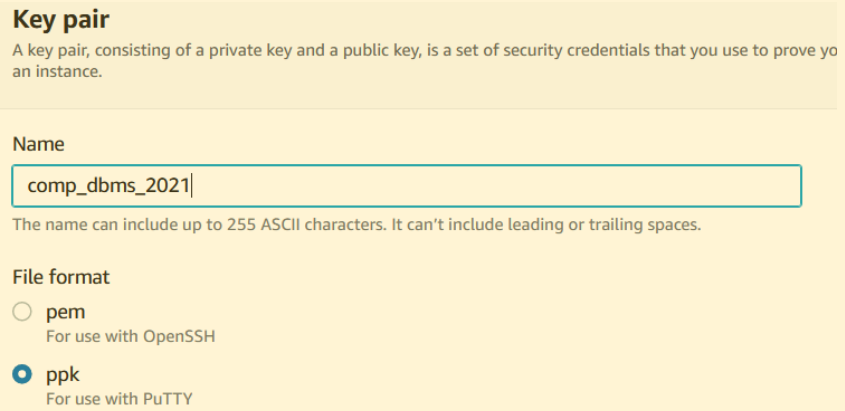
Creating Our Key-Pair



1. On the EC2 page, there is an option for **Key Pairs** on the left-hand side

2. Within the Key Pair page, click **Create key pair**
   1. Give the key pair a name
   2. Select your format (ppk for windows, pem for Mac/Linux)



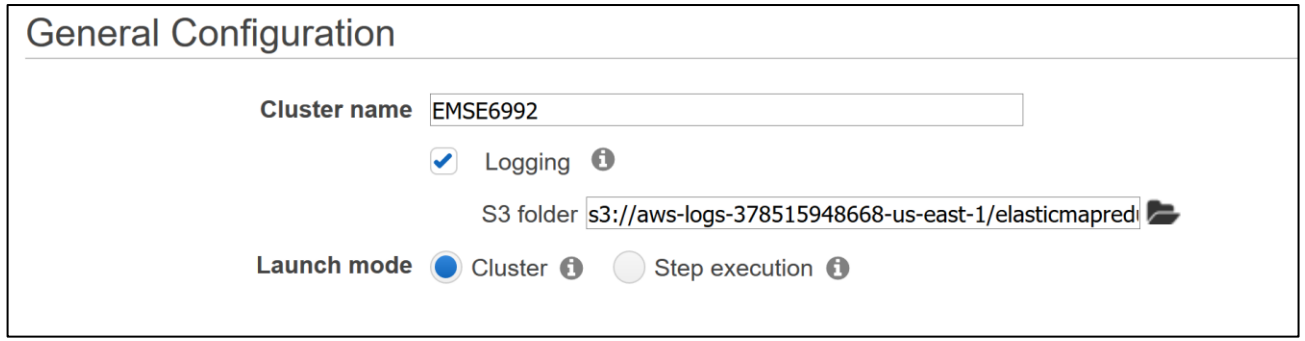


Creating Our EMR Cluster

Creating an EMR Cluster

1. Start creating a cluster:

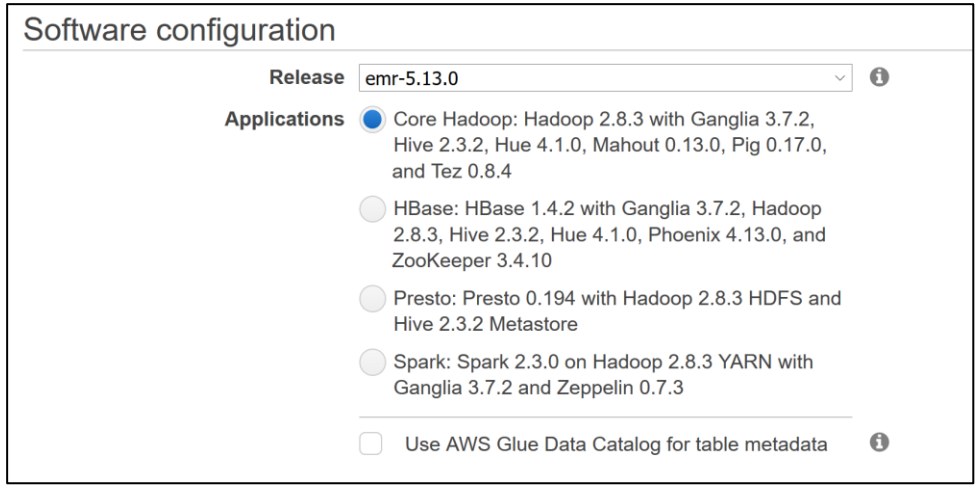
2. Define Cluster:



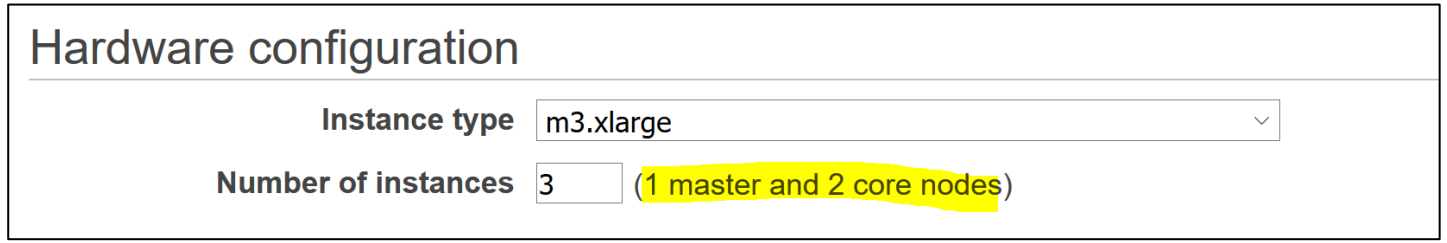
- NOTE: “Launch mode” defines how we initialize our cluster. “Step execution” enables us to define certain configuration or setup processes to run while creating our cluster (not necessary for our purposes).

Creating an EMR Cluster cont.

3. Defining Software:

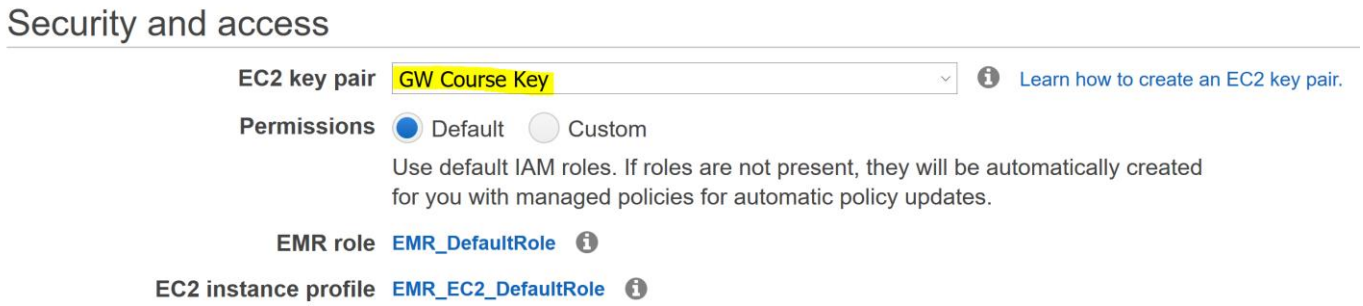


4. Defining the Nodes:

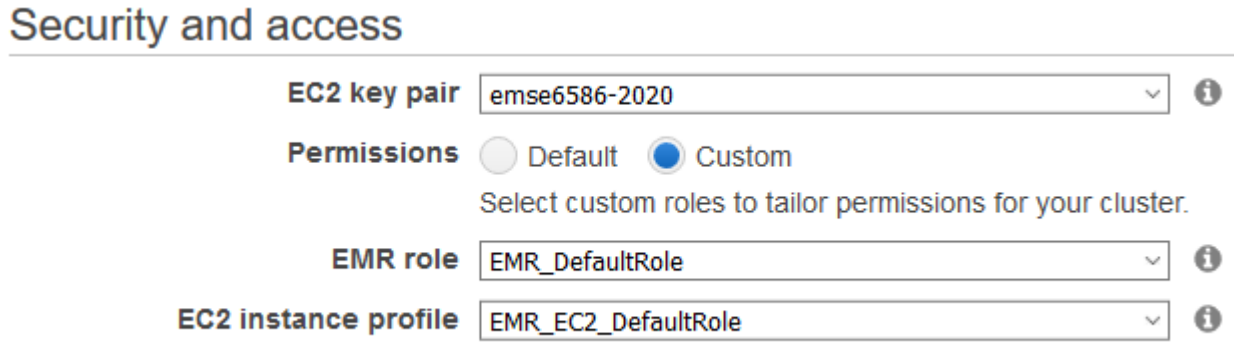


Creating an EMR Cluster cont.

5. Defining Security:
   - This defines the credentials to access the cluster



6. Defining Roles:
   - Select **Custom** permissions and provide the roles we created earlier



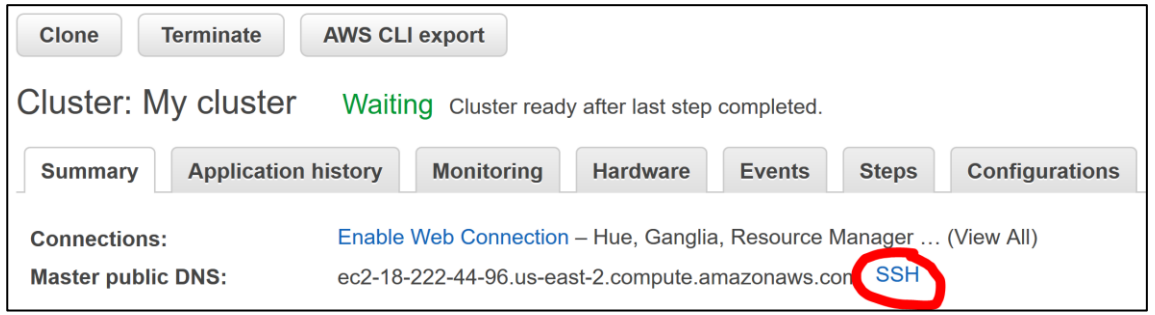
Accessing Our EMR

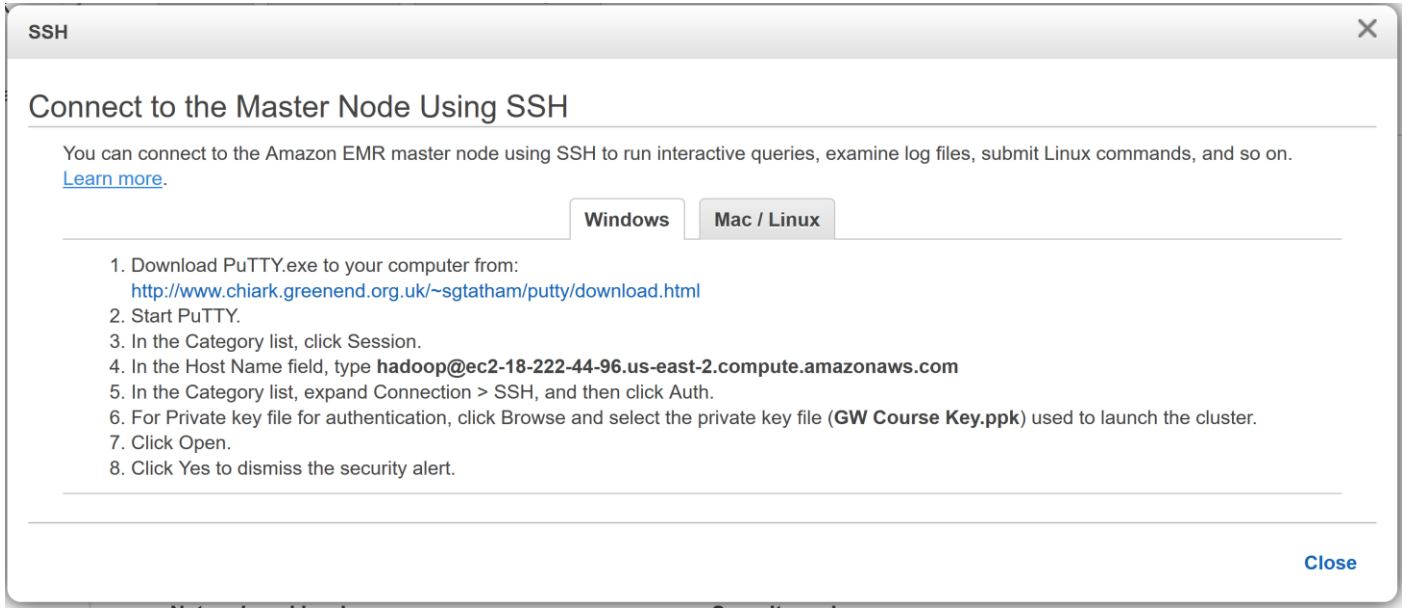
Access Through Shell/SSH

- AWS launches an EC2 (Virtual Machine) instance for each node, but we typically only connect to the namenode.
  - We connect to the namenode, as this is where commands are executed
- The EC2 instance does not have a graphical interface and thus we connect via SSH (directly into the terminal)
- The endpoint and security keys are designated during the creation of the cluster and the EC2 pair key

SSH Endpoint

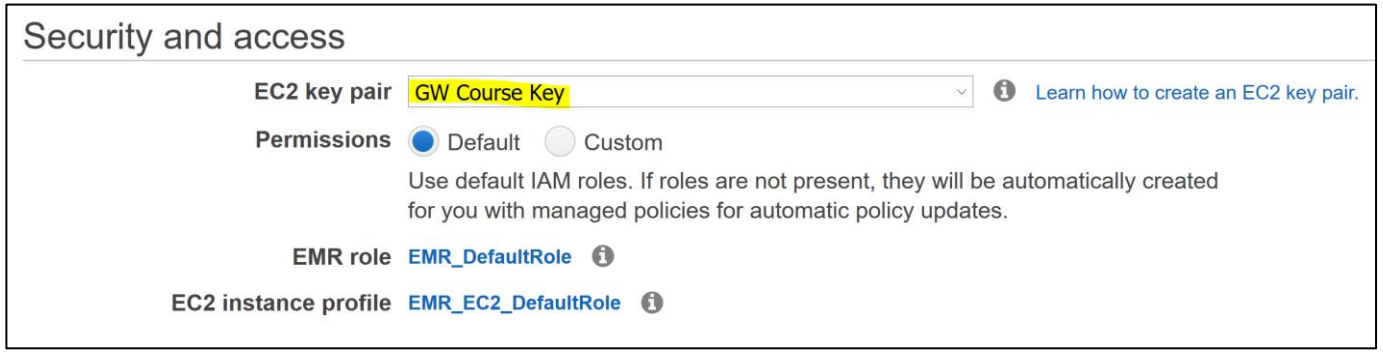
- Finding our endpoint:



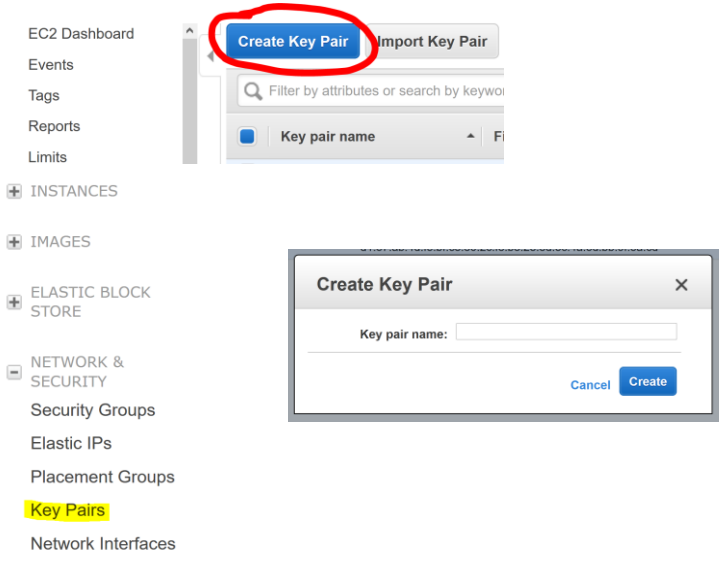


SSH Security Key

- Finding our security key:



- Note:
  - The EC2 key pair is something defined at the EC2 level. It defines an RSA key that can be used as credentials for connecting to EC2 instances.



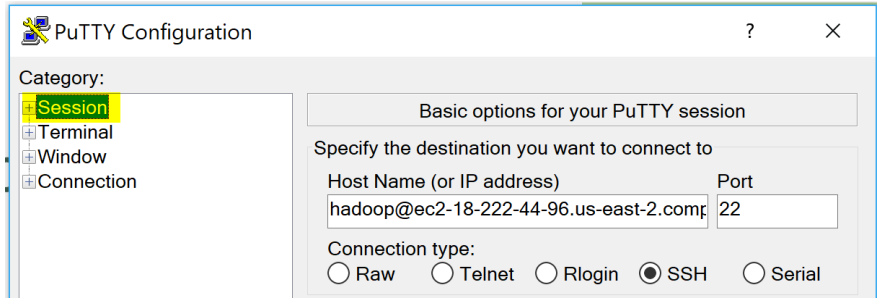
Using the Endpoint and Security Key to SSH

- Ultimately the “how” on the SSH side is platform dependent
  - Macs and Linux machines have SSH built-in
  - Windows typically use Putty

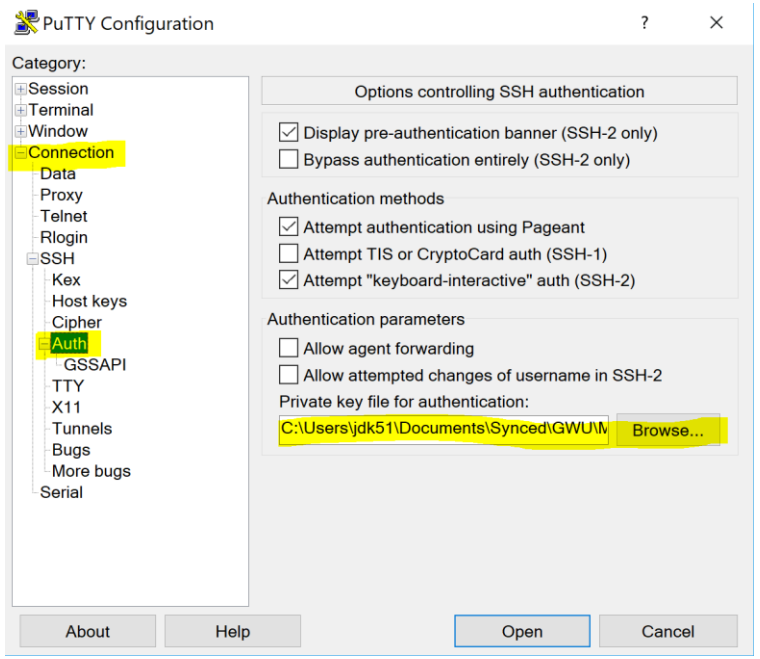
- When connecting via SSH there are a couple key important steps
  1. IP Address: hadoop@{endpoint}
  2. Port: 22 (unless defined otherwise)
  3. Key: EC2 key-pair (if defined during cluster creation)

Putty Example

1. IP and Port:



2. EC2 Key-Pair:



Mac/Linux Example

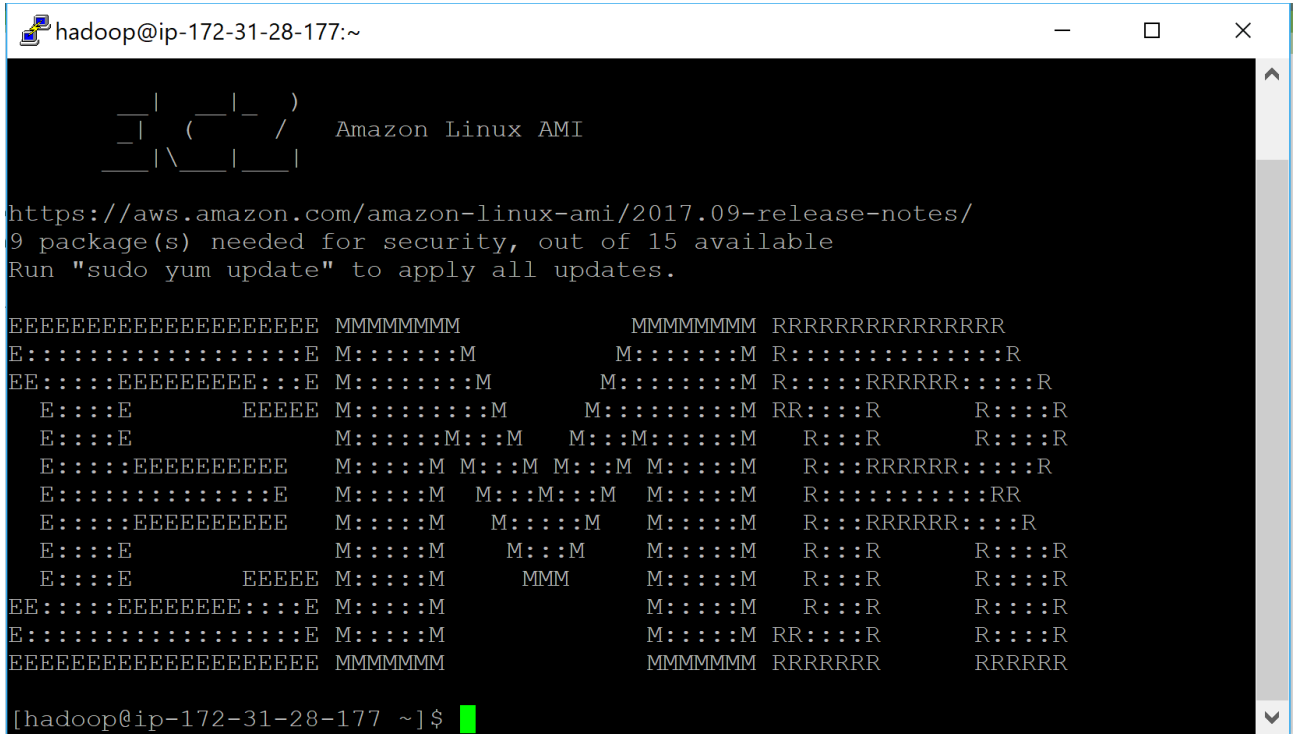
ssh –i **/path/to/key-pair.pem** hadoop@namenode_address

- Here we see another example of a flag in a command
  - The –i flag states the identifier file used to authenticate

Hadoop Terminal

Once Connected Via SSH

- If everything was done correctly, we should be met with the following:

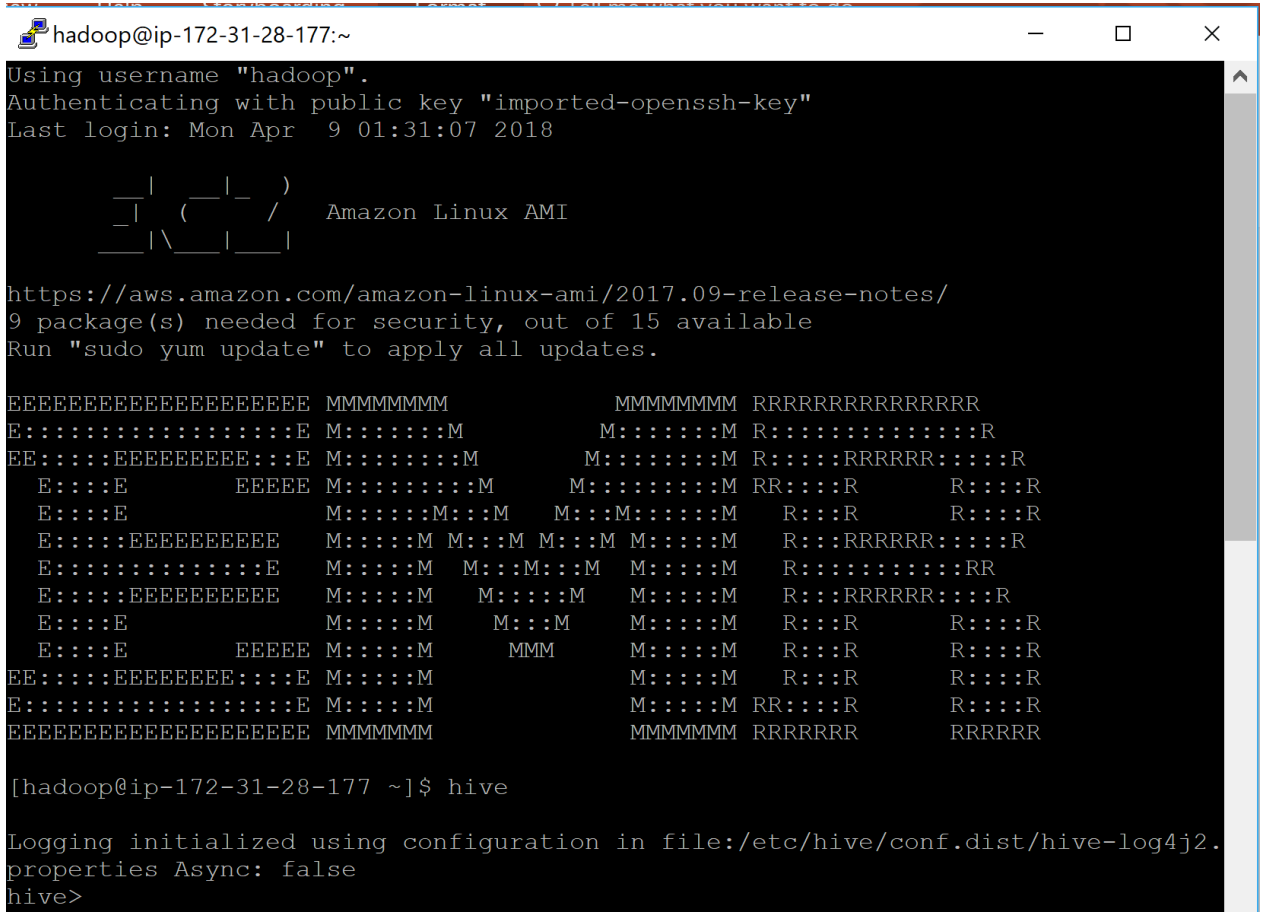


Hive

- Sadly, the terminal isn't very useful to us
- Instead to leverage Hadoop/our cluster, we need to leverage the installed software.
- Our focus today will be on Hive
  - Hive is essentially a SQL interface for working with data stored in Hadoop

Connecting to Hive

- From the terminal, we are can easily connect to hive
  - We simply run: $ hive
  - This should produce the hive interface: hive >



Hive

What is Hive?

- As mentioned earlier, Hive is essentially a SQL interface for working with Hadoop.
  - This may raise the question of why?
- Why Hive is so useful is that it allows us to create a SQL-like representation of our data in Hadoop.
  - Remember that Hadoop is for storing/processing terabytes of data per node (in the more extreme cases).
- Thus, Hive enables us to run standard queries against data that cannot fit in a normal relational database (MySQL, SQL Server, SQLite, etc)

SQL Interface?

- When I say SQL interface, it means that Hive doesn't actually run SQL
- Hive only enforces schema on read
  - This means that Hive doesn't load all the data in Hadoop into tables for querying. Instead, it reads the data and parses it as if it was in the table structure
- This means that Hive can be pointed at any data source that we can coerce into a table-like structure

Creating a Hive Table

Defining a Hive Table

- When defining a Hive table we need to know a couple of things:
  - Data Source (S3, HDFS, Azure, etc)
  - Format of the Data (CSV, JSON, TXT, etc)
    - This includes data types (string, int, double, etc)
  - External vs Internal (Does Hive own the data?)

- With the answers to these questions we can now run:
  - CREATE {External/Internal} Table {Table Name} (
    {col name} {col type},
) LOCATION {url to data};

Beer Data

- I’ve loaded a number of records into S3 based on beer ratings
  - These ratings are in JSON format

- Using this data, I can create an External Hive Table with the following:
  - CREATE EXTERNAL TABLE beer (`beer/beerId` string, `beer/brewerId` string, `beer/ABV` double) ROW FORMAT SERDE 'org.apache.hive.hcatalog.data.JsonSerDe' LOCATION 's3n://compdbms-spring-2021-jk/ratebeer/';

  - Note: {ROW FORMAT SERDE 'org.apache.hive.hcatalog.data.JsonSerDe’} is stating that I will be using the apache hive Json serializer/deserializer for processing the files

Querying Hive

Querying in Hive

- One of the primary benefits of Hive is that it imitates SQL.
  - This means everything we know about SQL is fairly applicable to Hive

- The main reason for the imitation is the fact that SQL is still used very extensively for data processing and analysis, thus integrating with SQL makes Hadoop integratable with other processes

Proof of Parity

- Lets run a quick example:
  - “SELECT * FROM beer LIMIT 10;
- For an outside perspective:
  - https://hortonworks.com/blog/hive-cheat-sheet-for-sql-users/

Some Queries to Run

- Select sum(cast(`beer/beerid` as int)) From beer;
- Select sum(distinct(cast(`beer/beerid` as int))) From beer;
- Select avg(split(`review/overall`, '/')[0]) From beer;
- SELECT avg(length(`review/text`)) as avg_len, std(length(`review/text`)) as std_len From beer;
- SELECT `review/text` as rev, length(`review/text`) as len from beer where length(`review/text`) > (218.16 + 343.28) limit 2;

DELETE YOUR EMR CLUSTER!!!

EMR clusters are running multiple expensive EC2 clusters – so we don’t want to leave it running or you’ll have no money left ☹

End Slide

EMSE 6992 – DBMS for Data Analytics