Big Data Wrangling With Google Books Ngrams

Analysis with Hadoop, Spark, and AWS *S3*

By

Anish Karnik

August 24, 2025

Q1) Spinning up a new EMR cluster on AWS for using Spark and EMR notebooks. (Screenshots Q1)

1. Create cluster

First, we must create our EMR cluster. Open the AWS console and navigate to the EMR panel through the search bar or homepage and click 'Create Cluster':

2. Cluster name and version

Give your EMR cluster a name, in this case it is '**Big Data Assignment'**. From the drop-down menu, **select release emr-6.10.0**, and select the 'Core Hadoop' application bundle. This will ensure that all the software we need for today comes pre-installed on the cluster.

3. Cluster hardware configuration

Scroll down to the 'Cluster Configuration' block. Leave the default setting ('Instance groups') ticked, leave the 'Primary' and 'Core' groups to **'m5.xlarge'**.

Remove the 'Task' instance group. Next, we can select how many instances we want in our cluster. We will have only one 'Primary' node, but let's give ourselves two 'Core' nodes. **Set the 'Size' of the 'Core' instance group to 2.**

4. Cluster Termination Settings

Scroll down to 'Cluster termination'. These settings determine whether our cluster should shut itself down automatically if it has sat idle for a set amount of time.Make sure that 'Terminate cluster after idle time' is selected and set the idle time to 4h.Untick the 'Use termination protection' box.

5. Security, Identity and Access Management

Select the EC2 key pair that in my case **"aws_cloud"**.For the 'Service role', select 'Choose an existing service role', and choose **'EMR_DefaultRole'** from the drop-down menu. For the 'Instance profile', select 'Choose an existing instance profile', and choose **'EMR_EC2_DefaultRole'** from the drop-down menu.

6. Create Cluster

Click **'Create Cluster'** on the right side of the screen. The cluster creation process will start. It may take some time for the EC2 instances to spin up and all the cluster software to be installed and configured. Once the cluster has been created it will enter a 'Waiting' state.

Q2) Connect to the head node of the cluster using SSH.(Screenshots Q2)

Now we can connect to the head node via SSH. Return to the cluster's overview screen and click the 'Connect to the Primary Node using SSH' hyperlink.

PuTTY Connection for Windows-EMR

- Spin up a Putty instance and navigate to Session, under Host Name enter this address:**hadoop@ec2-3-144-144-99.us-east-2.compute.amazonaws.com**
- Then Open up **Authenticate>Credentials** and under Private key file for authentication, select the appropriate **AWS access key.**
- Select Open then, If the connection is successful, we should be greeted by the **EMR ASCII** banner.

Q3) Copy the data folder from the S3 bucket directly into a directory on the Hadoop File System (HDFS) named /user/hadoop/eng_1M_1gram(Screenshots-Q3)

Under the EMR Cluster banner, run this command:

hadoop distcp s3://brainstation-dsft/eng_1M_1gram.csv /user/Hadoop/eng_1M_1gram/

After the folder is copied, check if it is in the HDFS, by running the following command:

hadoop fs -ls /user/Hadoop/eng_1M_1gram/

Q4) Using pyspark, read the data you copied into HDFS in Step 3. You may either use Jupyterhub on EMR (the default user and password are jovyan and jupyter) or work from pyspark in the terminal if you prefer. Once you have created a pyspark DataFrame, complete the following steps below: (Screenshots-Q4)

PuTTY Connection for Windows-Spark Notebook

- Spin up a new Putty instance and navigate to Session, under Host Name enter the same primary node SSH address:**hadoop@ec2-3-144-144-99.us-east-2.compute.amazonaws.com**

- Then Open up **Authenticate>Credentials** and under **Private key** file for authentication, select the appropriate AWS access key.

- Navigate to Tunnels, in the Source port enter **9995**, under destination port enter **localhost:9443.** Click on Add and hit Open, If the connection is successful, we should be greeted by the EMR ASCII banner.Now we have created a tunnel to connect to Spark notebooks

- We can now access JupyterHub in our browser at **https://localhost:9995.**If successfully connected, we will be dropped into the JupyterHub login screen as below. The login defaults are username of **jovyan** with password **jupyter**
- Check " **Big Data Wrangling With Google Books Ngrams_Q4.ipynb"** for Q4 analysis
- To check contents in Hadoop after Q4 part c, run the following command:

  hadoop fs -ls /user/hadoop/eng_1M_1gram/

Q5) Collect the contents of the directory into a single file on the local drive of the head node using getmerge and move this file into a S3 bucket in your account.(Screenshot-Q5)

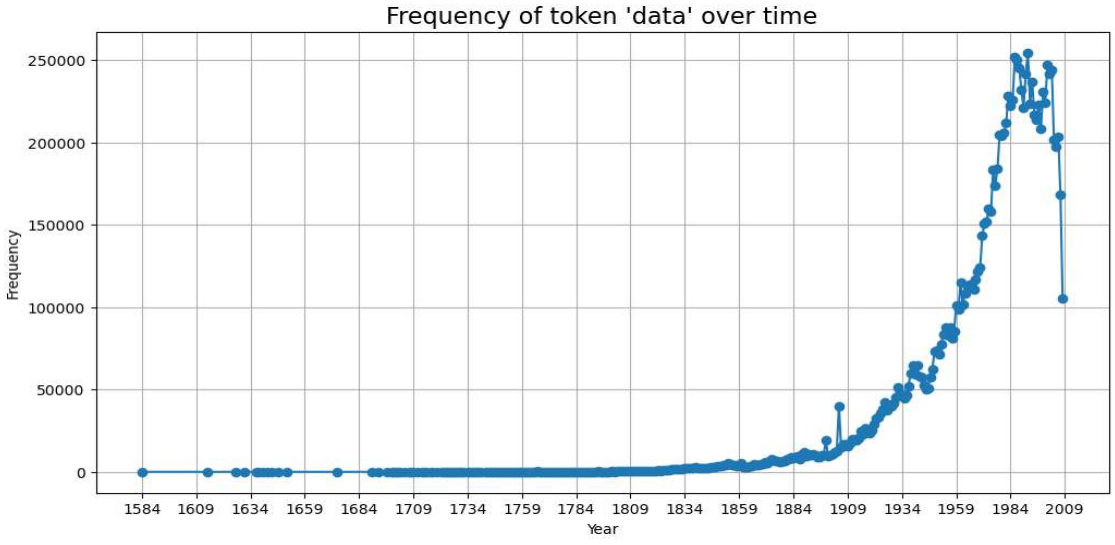
- To merge the conents into 1 file run this command:
  hdfs dfs -getmerge -nl /user/Hadoop/Bigdata_token ~/Bigdata_token.csv
- To move this file to S3 bucket on AWS, run the following command:
  aws s3 cp ~/Bigdata_token.csv s3 //akarnik-bstn-bucket/Bigdata_token.csv
- We can verify the location of the file moved by loging into our AWS console and searching on the task bar for S3, under buckets, we have **akarnik-bstn-bucket,** which should contain the file **Bigdata_token.csv**

Q6) On your local machine (or on AWS outside of Spark) in python, read the CSV data from the S3 folder into a pandas DataFrame (You will have to research how to read data into pandas from S3 buckets). Note You must have first authenticated on your machine using aws configure on the command line to complete this step)(Screenshot-Q6)

- To authenticate local for AWS, we first need to have AWS commad Line interface. We can download and following the instruction for default instructions for setup on Windows here: https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-getting-started.html
- We open with git bash and run the following command: **aws configure.** We are asked to later provide our AWS Acess Key, along with the Secret Access Key and select the default region which is **"us-east-2"** along with default output format **"json".** We can then run this command to see all our s3 buckets on aws: **aws s3 ls**

Q7.) Plot the number of occurrences of the token (the *frequency* column) of data over the years using matplotlib.

- Check Notebook # 6 for the code for this plot below:



Historical Usage of the Token 'Data'

The following analysis summarizes the frequency of the token 'data' in published works over time, based on the provided plot. The trends highlight the evolution of the term from medieval times to central importance in the digital era.

Key Observations

1. Early Usage (Before 1800s):
- The word 'data' appeared very rarely before the 19th century.
- This suggests it was not a common term in published texts.

2. Steady Growth (1800–1900):
- Usage increased slowly during the 19th century.
- This aligns with the growth of science, statistics, and industrialization.

3. Rapid Acceleration (1900–1950):
- Sharp increase in the early 20th century.
- Likely linked to developments in mathematics, physics, and early computing.

4. Exponential Rise (1950–2000):
- A huge surge post–World War II.
- Connected to the information age: digital computers, databases, and telecommunications.

5. Peak and Fluctuations (After 1980s):
- Usage peaked between the 1980s and early 2000s.
- Reflects the rise of computing, the internet, and data-driven research.
- Later fluctuations may reflect digitization biases or alternative terminology like 'big data' or 'information'.

Key Takeaway

The token 'data' evolved from a little-used academic term to one of the most central concepts in modern science and technology, with explosive growth during the digital revolution.

Q8) Compare Hadoop and Spark as distributed file systems.

a. What are the advantages/ differences between Hadoop and Spark? List two advantages for each.

b. Explain how the HDFS stores the data.

A. Hadoop vs Spark

It is important to note that Hadoop is not just a file system, but an ecosystem with two key components: the Hadoop Distributed File System (HDFS) for storage, and MapReduce for processing. Spark, on the

other hand, is not a file system at all, but a distributed data processing engine that can run on top of HDFS, Amazon S3, or other storage systems.

Differences Between Hadoop (MapReduce) and Spark

Feature	Hadoop (MapReduce)	Spark
Processing	Disk-based (reads/writes to disk between operations)	In-memory (keeps data in RAM, much faster)
Speed	Slower (I/O heavy)	Up to 100x faster for iterative tasks
Ease of Use	Low-level Java APIs (harder to code)	High-level APIs in Python, Scala, SQL
Use Cases	Batch processing (ETL, log analysis)	Batch + Streaming + ML + Graph processing

Advantages of Hadoop

- Scalability & Storage: Handles petabytes of data reliably using HDFS, with built-in fault tolerance.
- Cost-effective: Runs on commodity hardware without requiring expensive servers.

Advantages of Spark

- Speed: In-memory computation makes Spark much faster than Hadoop MapReduce, especially for iterative tasks like machine learning.
- Flexibility: Provides rich APIs and libraries such as Spark SQL, MLlib for machine learning, GraphX for graph processing, and Structured Streaming for real-time data.

B. How HDFS Stores Data

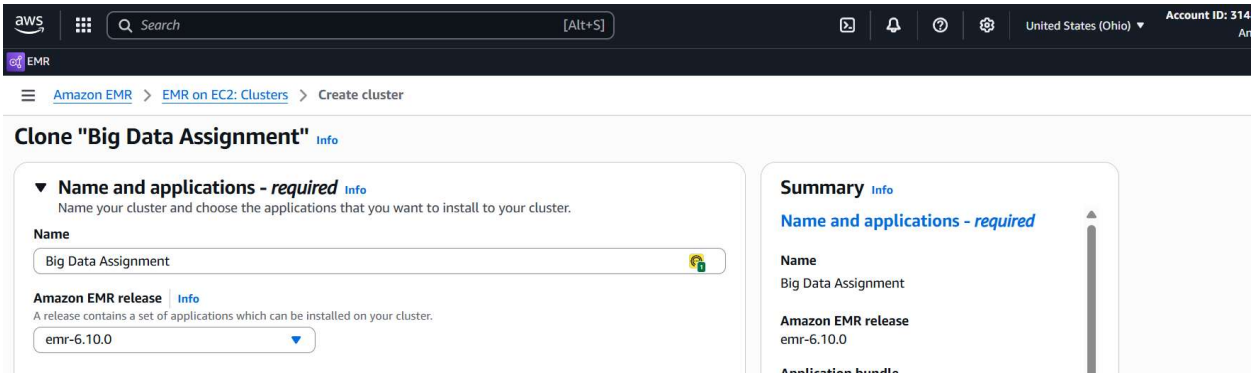
The Hadoop Distributed File System (HDFS) is the storage layer of the Hadoop ecosystem. It is designed for high-throughput access to large datasets and is optimized for streaming reads and writes.

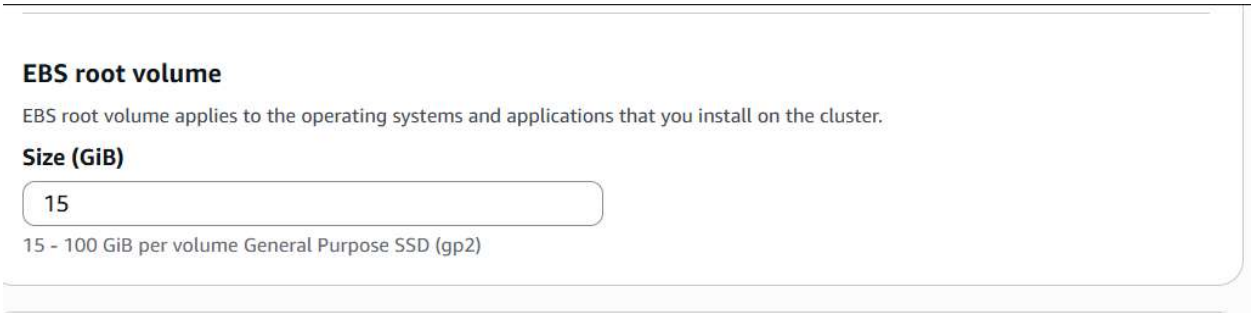
• Data Block Storage: Files are split into large fixed-size blocks (default size = 128 MB or 256 MB).
• Replication: Each block is replicated across multiple DataNodes (default replication factor = 3) to ensure fault tolerance.
• NameNode: The master node that stores metadata such as file names, block locations, and directory structure.
• DataNodes: Worker nodes that actually store the file blocks.
• High Throughput: Optimized for handling very large files rather than low-latency access.
• Fault Tolerance: If a DataNode fails, HDFS automatically uses replicas from other nodes.

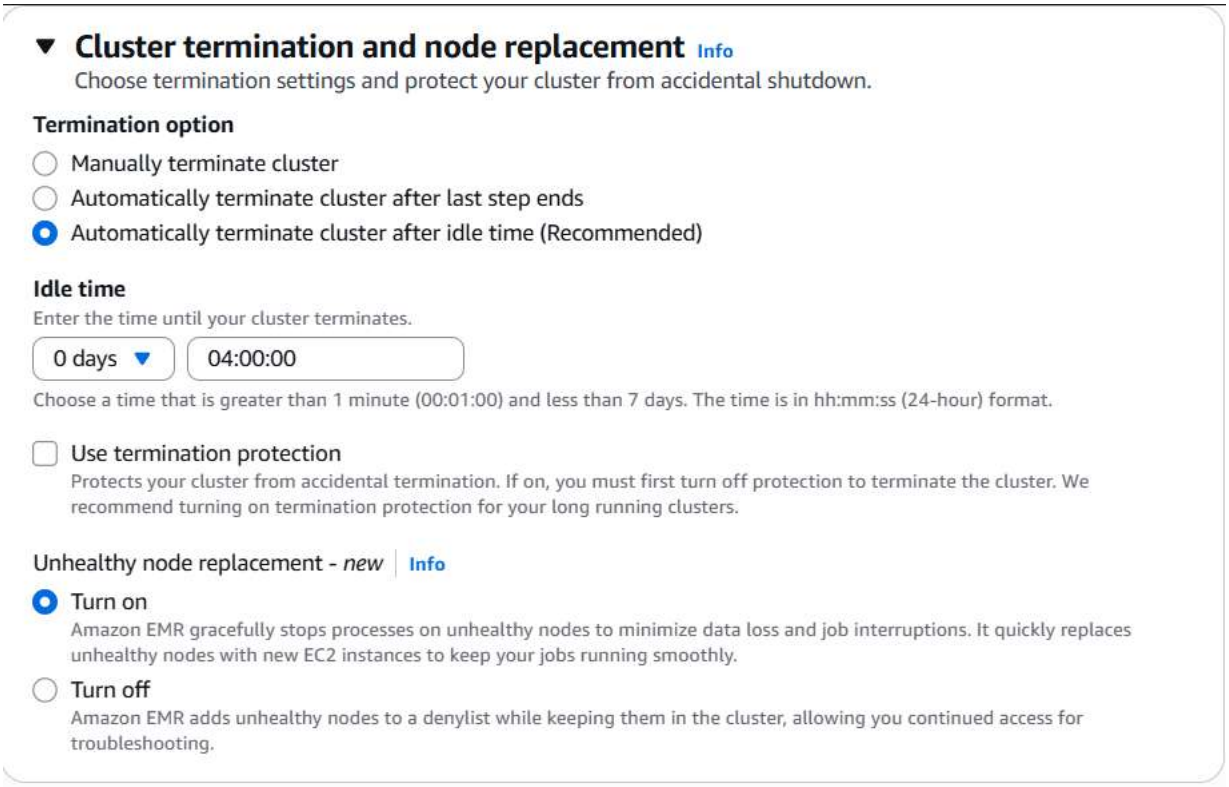
Example: If a 512 MB file is stored in HDFS with a block size of 128 MB, it will be split into 4 blocks. Each block will be replicated 3 times across different DataNodes, while the NameNode maintains metadata about block locations.

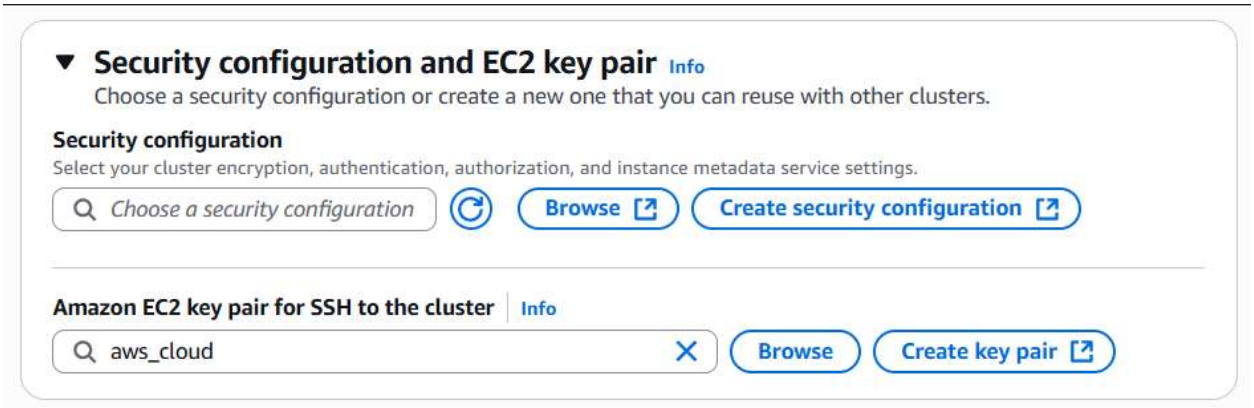
Appendix

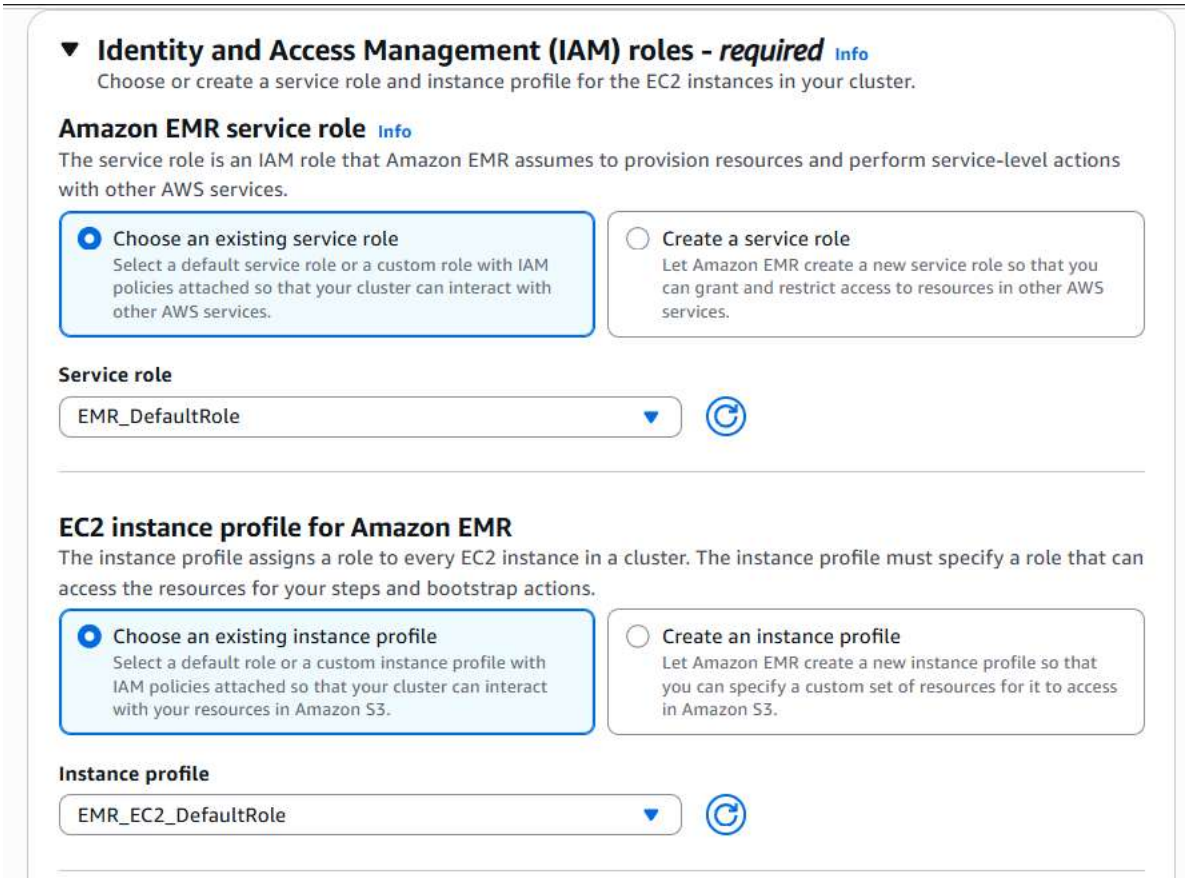
Q1-Screenshots



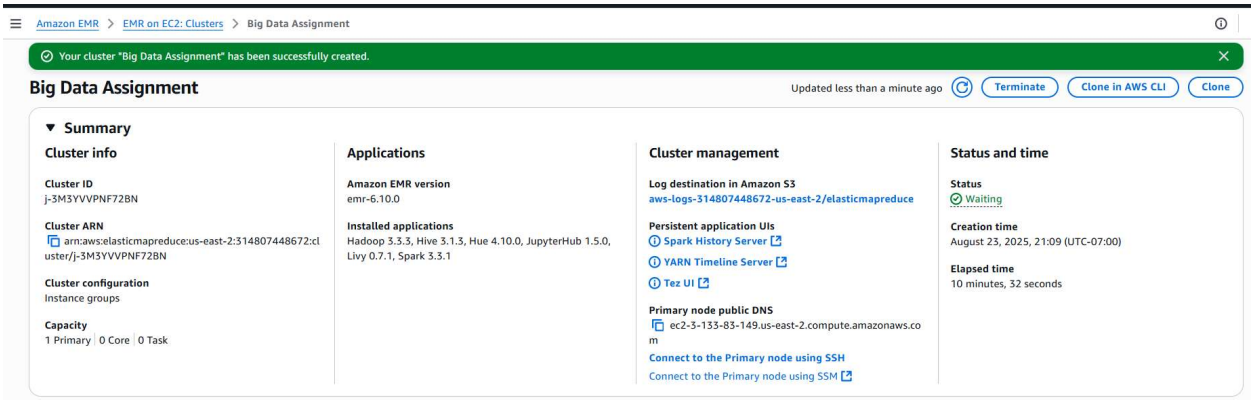



Cluster termination and node replacement Info
Choose termination settings and protect your cluster from accidental shutdown.
Termination option
Manually terminate cluster
Automatically terminate cluster after last step ends
Automatically terminate cluster after idle time (Recommended)
Idle time
Enter the time until your cluster terminates.
0 days
04:00:00
Choose a time that is greater than 1 minute (00:01:00) and less than 7 days. The time is in hh:mm:ss (24-hour) format.
Use termination protection
Protects your cluster from accidental termination. If on, you must first turn off protection to terminate the cluster. We recommend turning on termination protection for your long running clusters.
Unhealthy node replacement - new Info
Turn on
Amazon EMR gracefully stops processes on unhealthy nodes to minimize data loss and job interruptions. It quickly replaces unhealthy nodes with new EC2 instances to keep your jobs running smoothly.
Turn off
Amazon EMR adds unhealthy nodes to a denylist while keeping them in the cluster, allowing you continued access for troubleshooting.


Security configuration and EC2 key pair Info
Choose a security configuration or create a new one that you can reuse with other clusters.
Security configuration
Select your cluster encryption, authentication, authorization, and instance metadata service settings.
Choose a security configuration
Browse
Create security configuration
Amazon EC2 key pair for SSH to the cluster Info
aws_cloud
Browse
Create key pair



Q2-Screenshots



Connect to the primary node using SSH

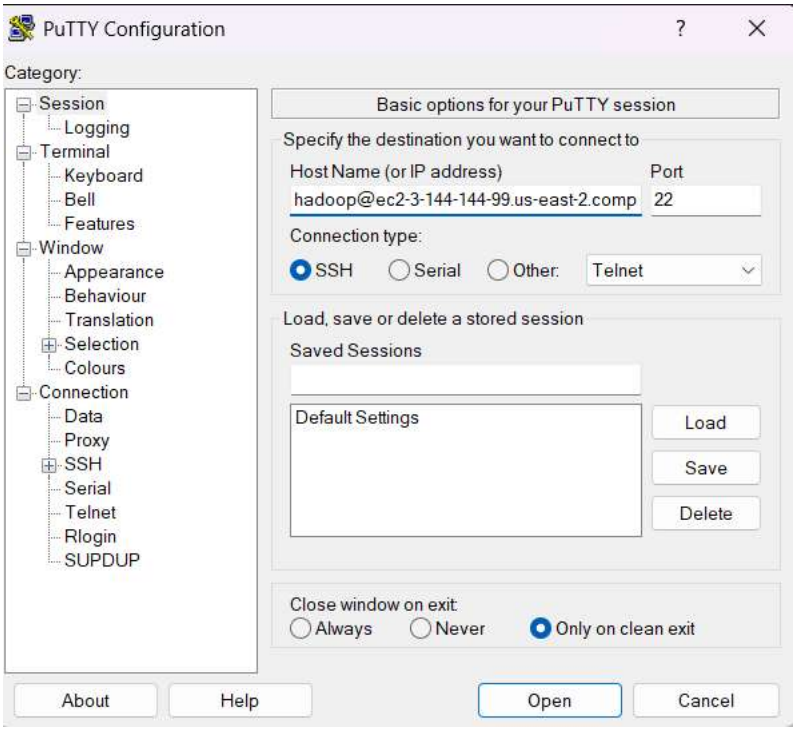
You can connect to the Amazon EMR primary node using SSH to perform actions like running interactive queries, examining log files, submit Linux commands, and view web interfaces hosted on Amazon EMR clusters. Learn more

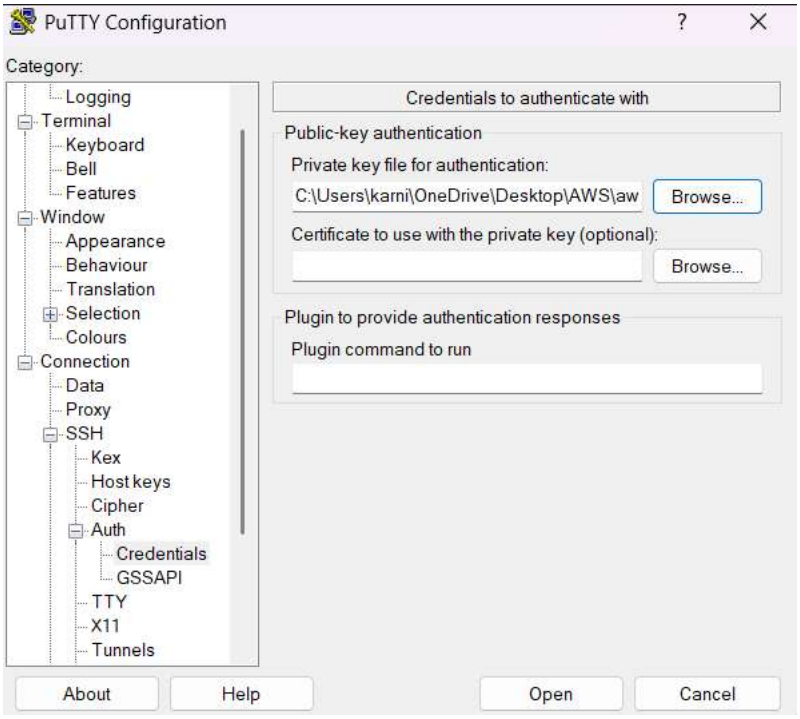
Windows | Mac/Linux

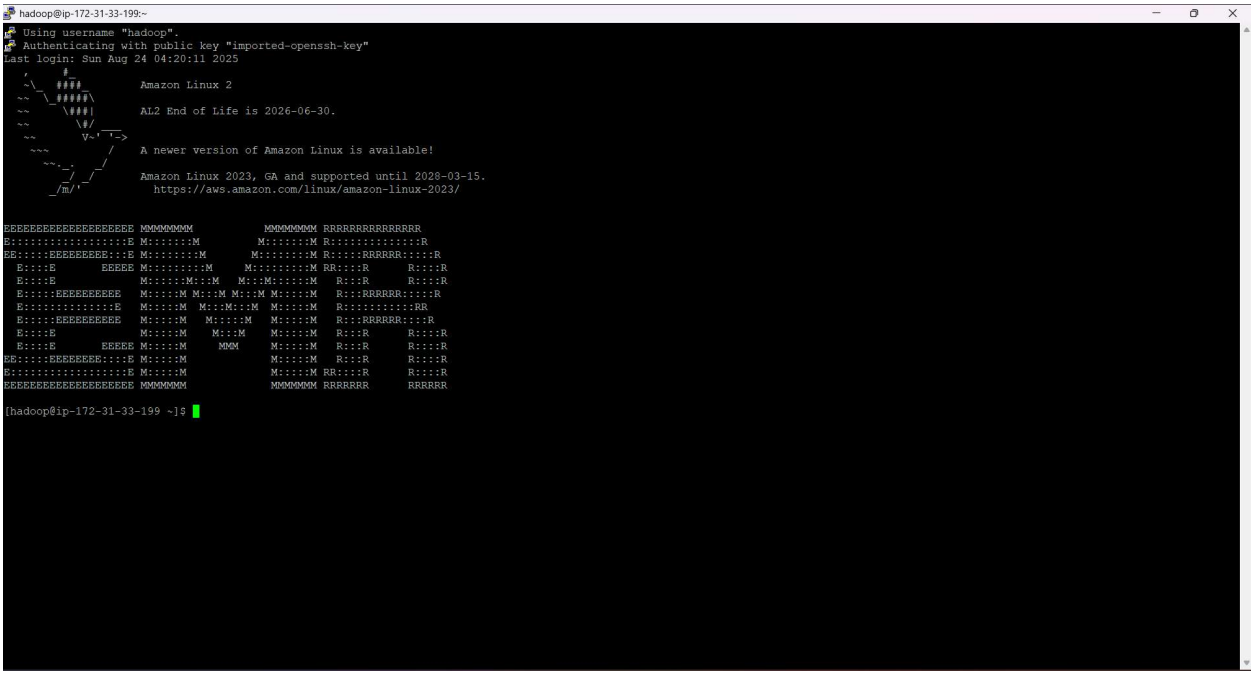
1. Download PuTTY.exe to your computer from:
   https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html
2. Start PuTTY.
3. In the Category list, choose Session.
4. In the Host Name field, enter `hadoop@ec2-3-144-144-99.us-east-2.compute.amazonaws.com`
5. In the Category list, expand Connection > SSH, and then choose Auth.
6. For Private key file for authentication, choose Browse and select the private key file (`aws_cloud.ppk`) that you used to launch the cluster.
7. Select Open.
8. Select Yes to dismiss the security alert.

View web interfaces hosted on Amazon EMR clusters

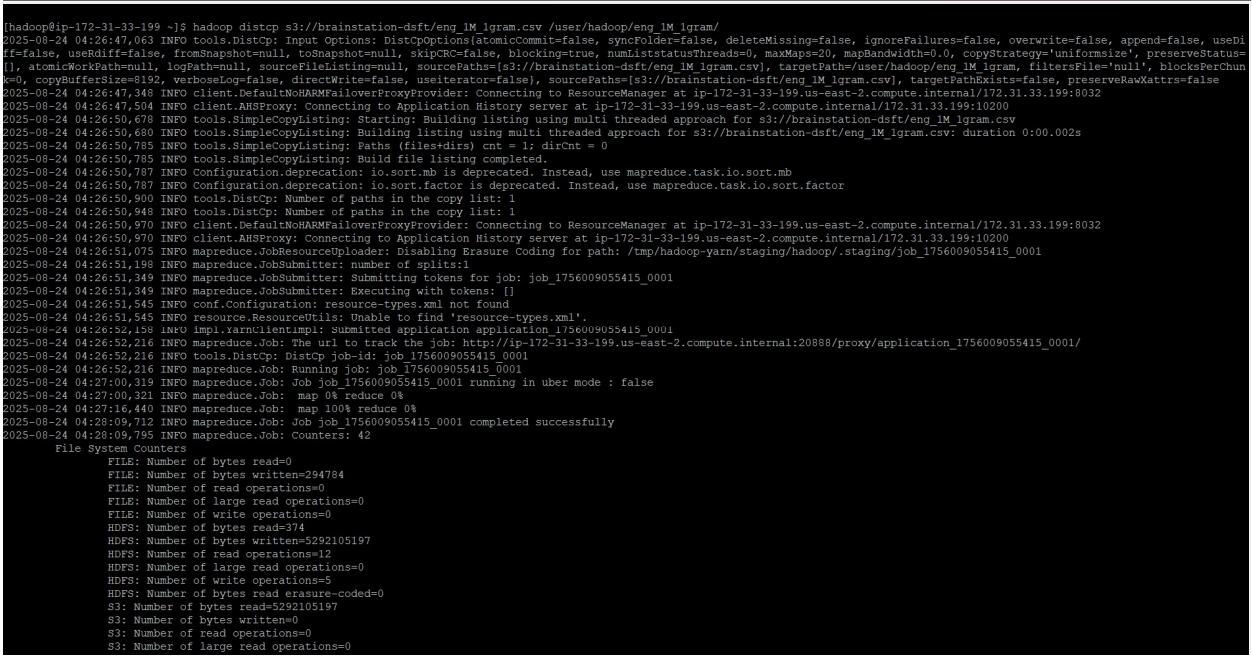
Close




PuTTY Configuration
Category:
Logging
Terminal
Keyboard
Bell
Features
Window
Appearance
Behaviour
Translation
Selection
Colours
Connection
Data
Proxy
SSH
Kex
Host keys
Cipher
Auth
Credentials
GSSAPI
TTY
X11
Tunnels
Credentials to authenticate with
Public-key authentication
Private key file for authentication:
C:\Users\karni\OneDrive\Desktop\AWS\aw
Browse...
Certificate to use with the private key (optional):
Browse...
Plugin to provide authentication responses
Plugin command to run
About
Help
Open
Cancel


hadoop@ip-172-31-33-199:~
Using username "hadoop".
Authenticating with public key "imported-openssh-key"
Last login: Sun Aug 24 04:20:11 2025
Amazon Linux 2
AL2 End of Life is 2026-06-30.
A newer version of Amazon Linux is available!
Amazon Linux 2023, GA and supported until 2028-03-15.
https://aws.amazon.com/linux/amazon-linux-2023/
[hadoop@ip-172-31-33-199 ~]$

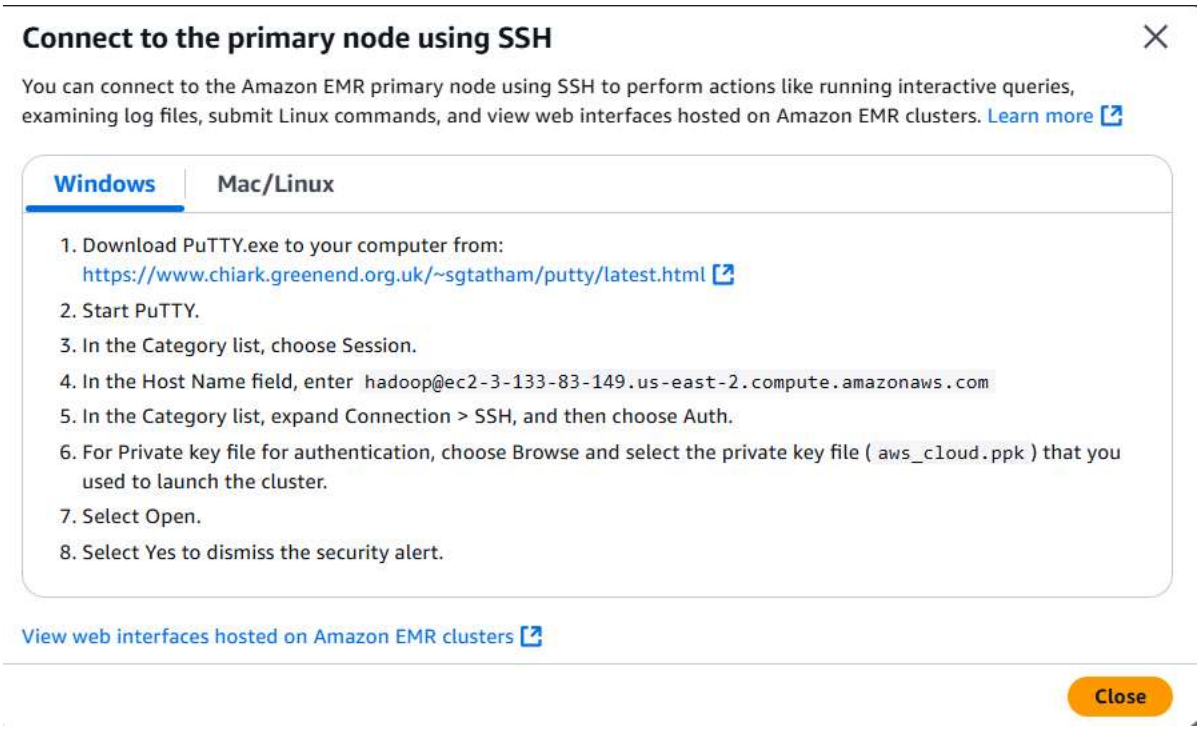
Q3-Screenshots

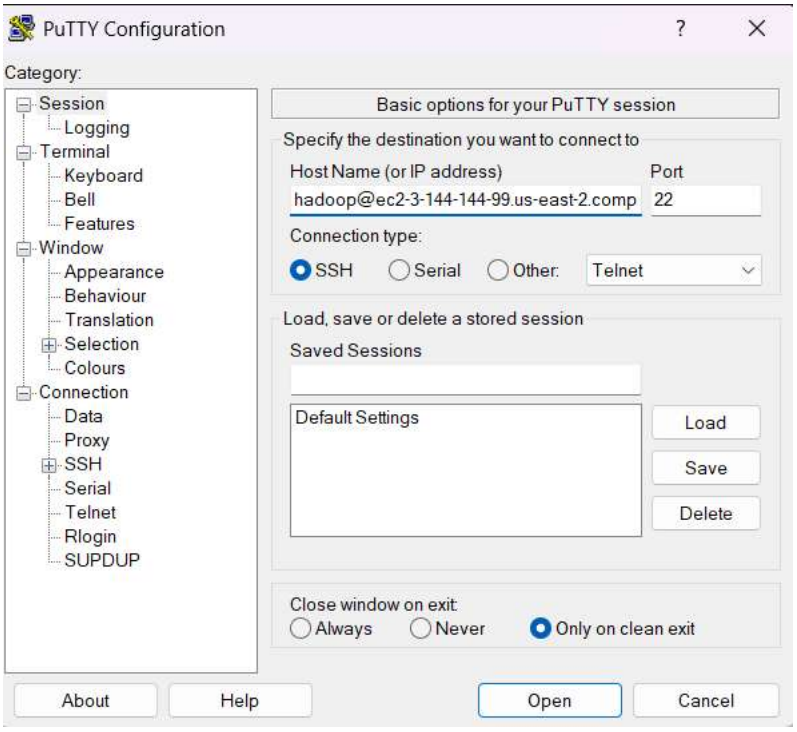


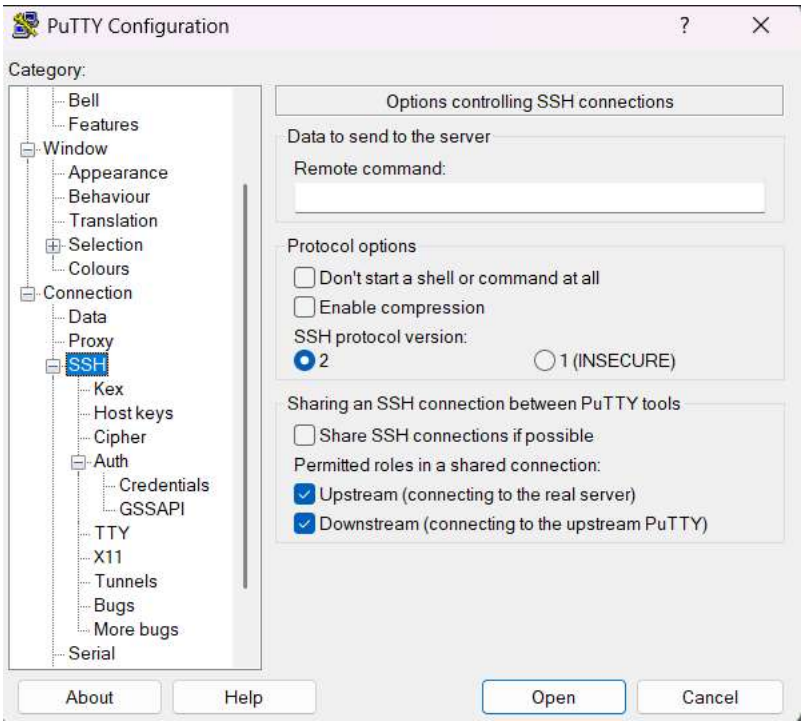
```
        Job Counters
                Launched map tasks=1
                Other local map tasks=1
                Total time spent by all maps in occupied slots (ms)=202681344
                Total time spent by all reduces in occupied slots (ms)=0
                Total time spent by all map tasks (ms)=65977
                Total vcore-milliseconds taken by all map tasks=65977
                Total megabyte-milliseconds taken by all map tasks=202681344
        Map-Reduce Framework
                Map input records=1
                Map output records=0
                Input split bytes=135
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=393
                CPU time spent (ms)=61870
                Physical memory (bytes) snapshot=971833344
                Virtual memory (bytes) snapshot=4428816384
                Total committed heap usage (bytes)=927465472
                Peak Map Physical memory (bytes)=971833344
                Peak Map Virtual memory (bytes)=4445511680
        File Input Format Counters
                Bytes Read=239
        File Output Format Counters
                Bytes Written=0
        DistCp Counters
                Bandwidth in Bytes=84001669
                Bytes Copied=5292105197
                Bytes Expected=5292105197
                Files Copied=1
[hadoop@ip-172-31-33-199 ~]$
```

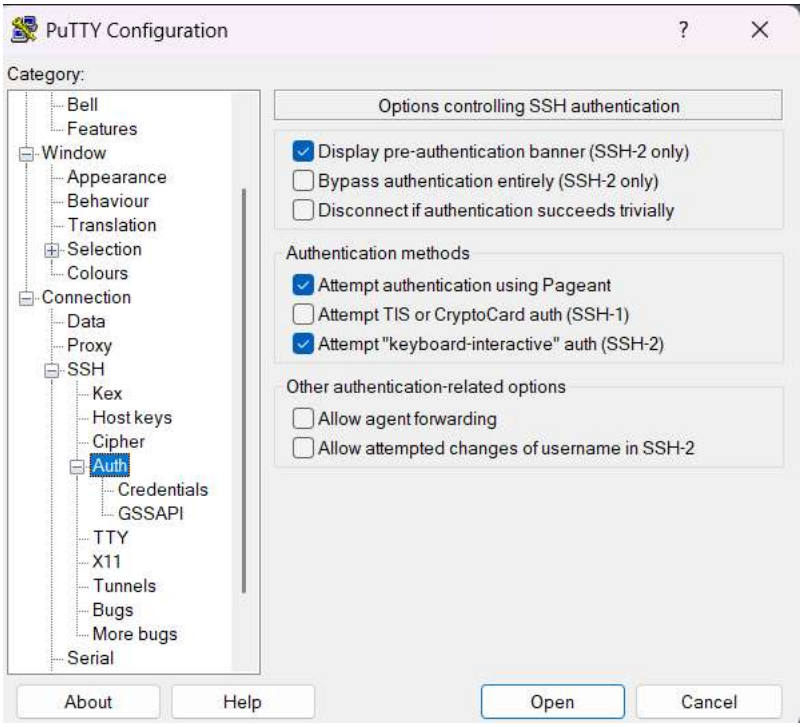
```
[hadoop@ip-172-31-33-199 ~]$ hadoop fs -ls /user/hadoop/eng_1M_1gram/
-rw-r--r--   1 hadoop hdfsadmingroup 5292105197 2025-08-24 04:28 /user/hadoop/eng_1M_1gram
[hadoop@ip-172-31-33-199 ~]$
```

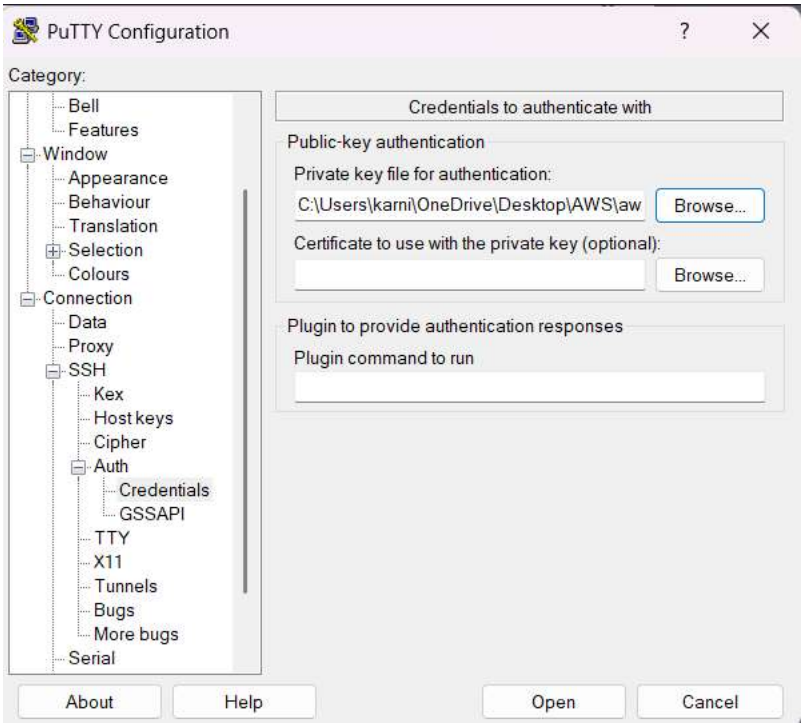
Q4-Screenshots

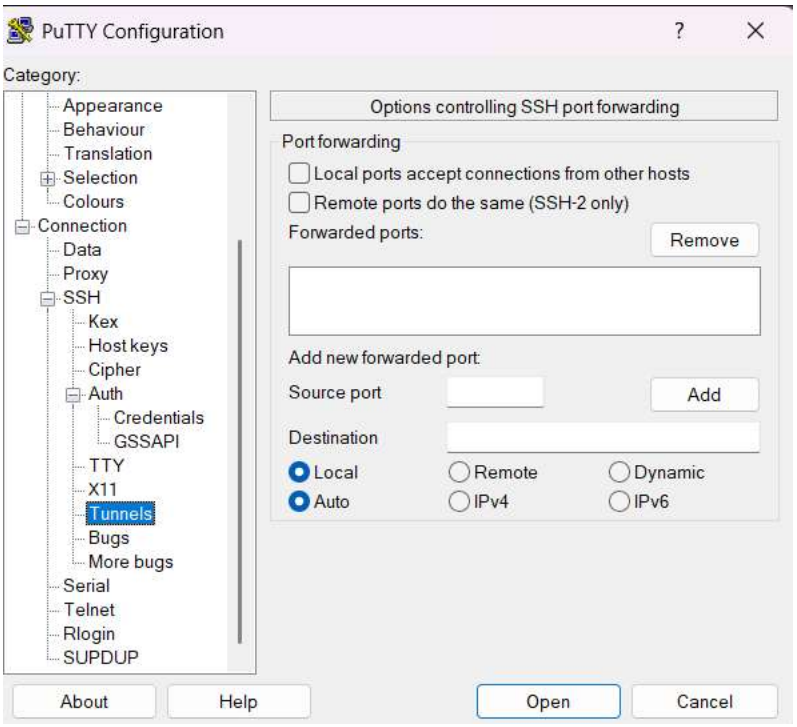


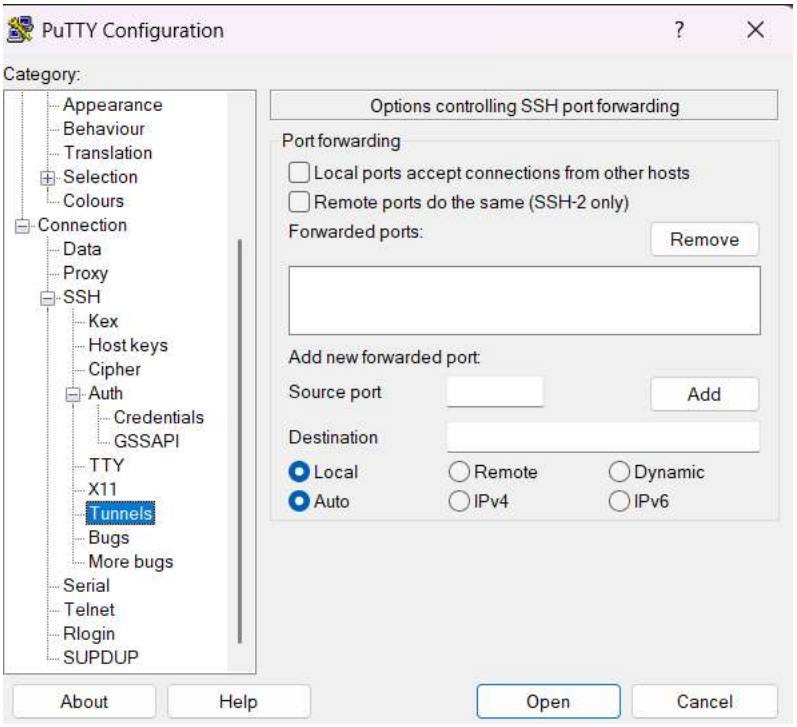

PuTTY Configuration
Category:
Session
Logging
Terminal
Keyboard
Bell
Features
Window
Appearance
Behaviour
Translation
Selection
Colours
Connection
Data
Proxy
SSH
Serial
Telnet
Rlogin
SUPDUP
Basic options for your PuTTY session
Specify the destination you want to connect to
Host Name (or IP address)
Port
hadoop@ec2-3-144-144-99.us-east-2.comp
22
Connection type:
SSH
Serial
Other:
Telnet
Load, save or delete a stored session
Saved Sessions
Default Settings
Load
Save
Delete
Close window on exit:
Always
Never
Only on clean exit
About
Help
Open
Cancel

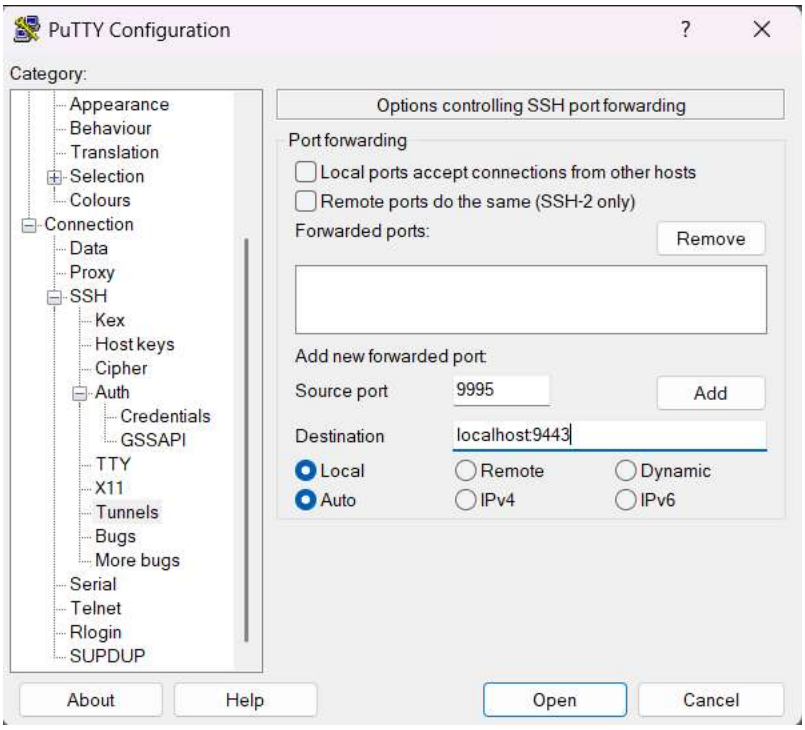

PuTTY Configuration
Category:
Bell
Features
Window
Appearance
Behaviour
Translation
Selection
Colours
Connection
Data
Proxy
SSH
Kex
Host keys
Cipher
Auth
Credentials
GSSAPI
TTY
X11
Tunnels
Bugs
More bugs
Serial
Options controlling SSH connections
Data to send to the server
Remote command:
Protocol options
Don't start a shell or command at all
Enable compression
SSH protocol version:
2
1 (INSECURE)
Sharing an SSH connection between PuTTY tools
Share SSH connections if possible
Permitted roles in a shared connection:
Upstream (connecting to the real server)
Downstream (connecting to the upstream PuTTY)
About
Help
Open
Cancel

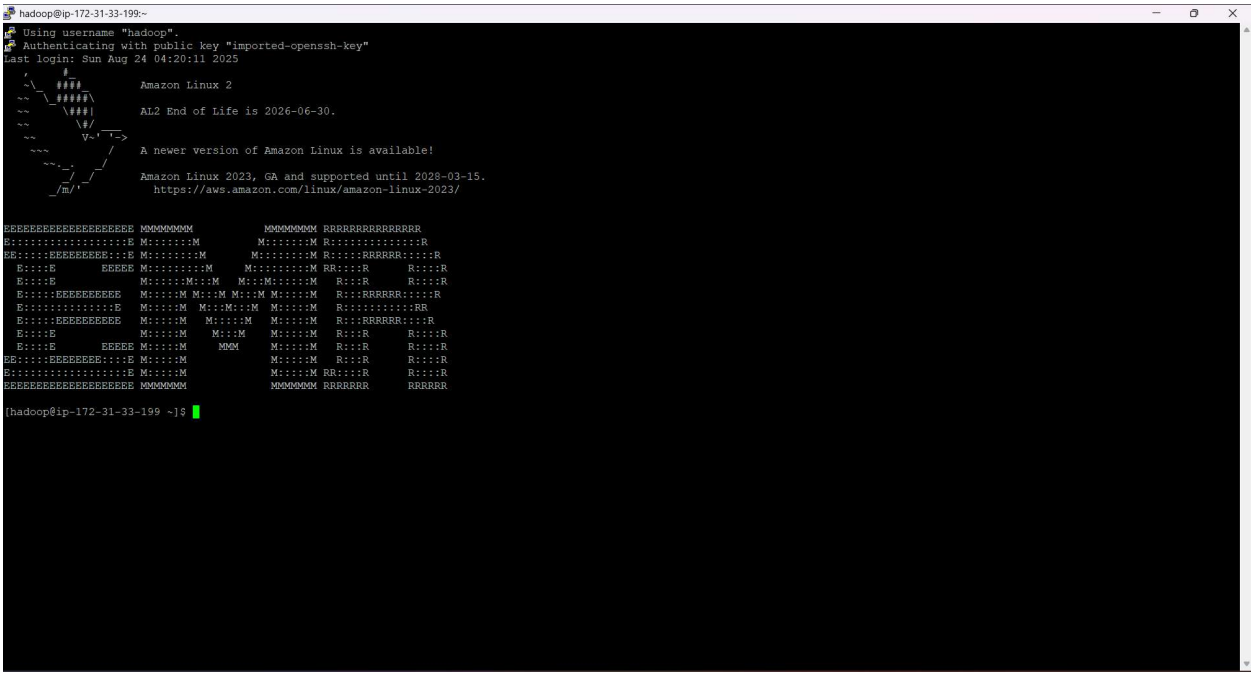

PuTTY Configuration
Category:
Bell
Features
Window
Appearance
Behaviour
Translation
Selection
Colours
Connection
Data
Proxy
SSH
Kex
Host keys
Cipher
Auth
Credentials
GSSAPI
TTY
X11
Tunnels
Bugs
More bugs
Serial
Options controlling SSH authentication
Display pre-authentication banner (SSH-2 only)
Bypass authentication entirely (SSH-2 only)
Disconnect if authentication succeeds trivially
Authentication methods
Attempt authentication using Pageant
Attempt TIS or CryptoCard auth (SSH-1)
Attempt "keyboard-interactive" auth (SSH-2)
Other authentication-related options
Allow agent forwarding
Allow attempted changes of username in SSH-2
About
Help
Open
Cancel

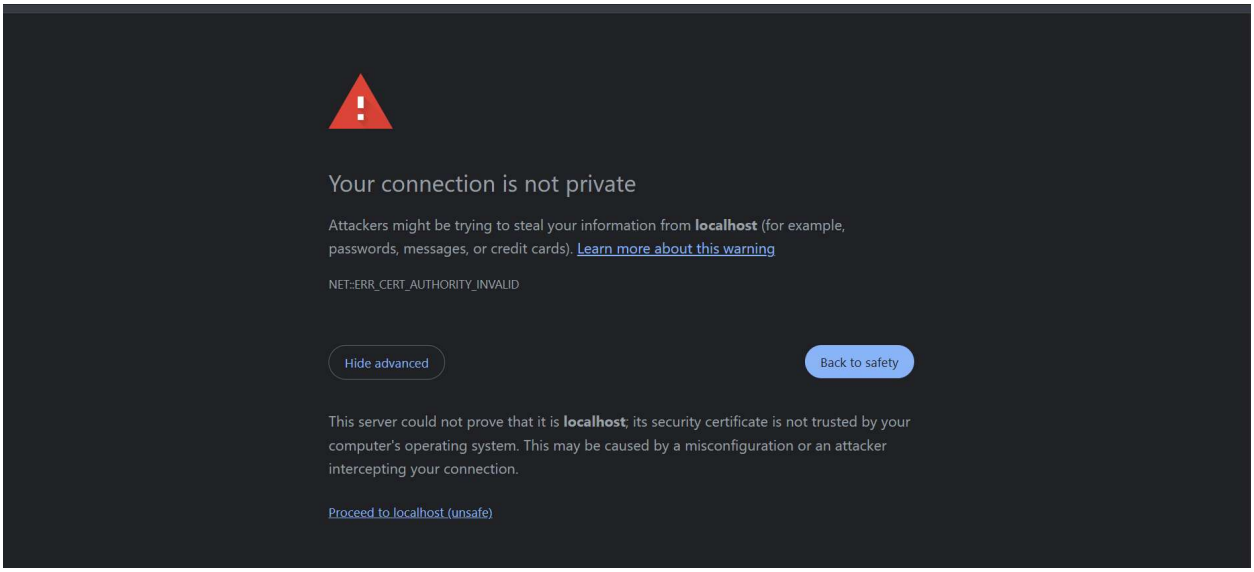

PuTTY Configuration
Category:
Bell
Features
Window
Appearance
Behaviour
Translation
Selection
Colours
Connection
Data
Proxy
SSH
Kex
Host keys
Cipher
Auth
Credentials
GSSAPI
TTY
X11
Tunnels
Bugs
More bugs
Serial
Credentials to authenticate with
Public-key authentication
Private key file for authentication:
C:\Users\karni\OneDrive\Desktop\AWS\aw
Browse...
Certificate to use with the private key (optional):
Browse...
Plugin to provide authentication responses
Plugin command to run
About
Help
Open
Cancel

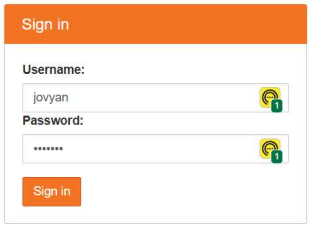

PuTTY Configuration
Category:
Appearance
Behaviour
Translation
Selection
Colours
Connection
Data
Proxy
SSH
Kex
Host keys
Cipher
Auth
Credentials
GSSAPI
TTY
X11
Tunnels
Bugs
More bugs
Serial
Telnet
Rlogin
SUPDUP
Options controlling SSH port forwarding
Port forwarding
Local ports accept connections from other hosts
Remote ports do the same (SSH-2 only)
Forwarded ports:
Remove
Add new forwarded port:
Source port
Add
Destination
Local
Remote
Dynamic
Auto
IPv4
IPv6
About
Help
Open
Cancel


PuTTY Configuration
Category:
Appearance
Behaviour
Translation
Selection
Colours
Connection
Data
Proxy
SSH
Kex
Host keys
Cipher
Auth
Credentials
GSSAPI
TTY
X11
Tunnels
Bugs
More bugs
Serial
Telnet
Rlogin
SUPDUP
Options controlling SSH port forwarding
Port forwarding
Local ports accept connections from other hosts
Remote ports do the same (SSH-2 only)
Forwarded ports:
Remove
Add new forwarded port:
Source port
Add
Destination
Local
Remote
Dynamic
Auto
IPv4
IPv6
About
Help
Open
Cancel


PuTTY Configuration
Category:
Appearance
Behaviour
Translation
Selection
Colours
Connection
Data
Proxy
SSH
Kex
Host keys
Cipher
Auth
Credentials
GSSAPI
TTY
X11
Tunnels
Bugs
More bugs
Serial
Telnet
Rlogin
SUPDUP
Options controlling SSH port forwarding
Port forwarding
Local ports accept connections from other hosts
Remote ports do the same (SSH-2 only)
Forwarded ports:
Remove
Add new forwarded port:
Source port
9995
Add
Destination
localhost:9443
Local
Remote
Dynamic
Auto
IPv4
IPv6
About
Help
Open
Cancel


hadoop@ip-172-31-33-199:~
Using username "hadoop".
Authenticating with public key "imported-openssh-key"
Last login: Sun Aug 24 04:20:11 2025
Amazon Linux 2
AL2 End of Life is 2026-06-30.
A newer version of Amazon Linux is available!
Amazon Linux 2023, GA and supported until 2028-03-15.
https://aws.amazon.com/linux/amazon-linux-2023/
[hadoop@ip-172-31-33-199 ~]$



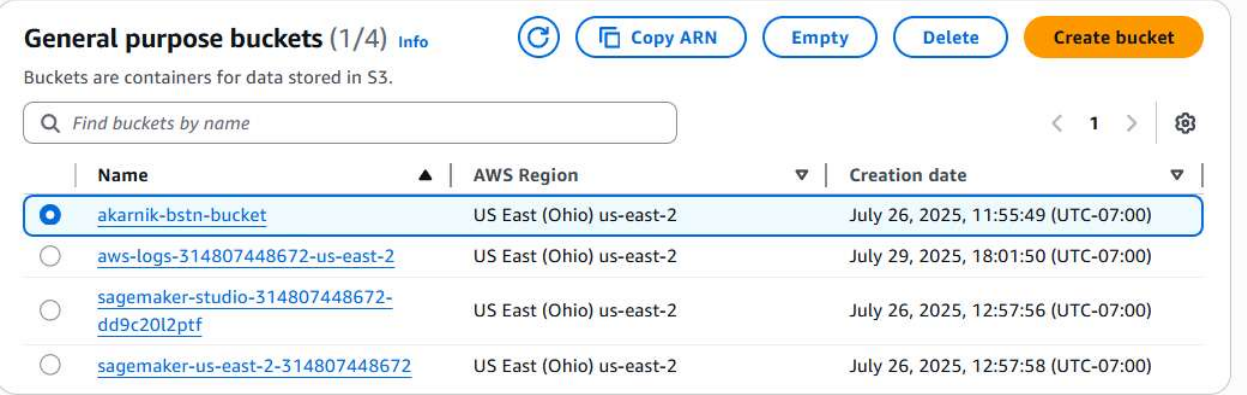


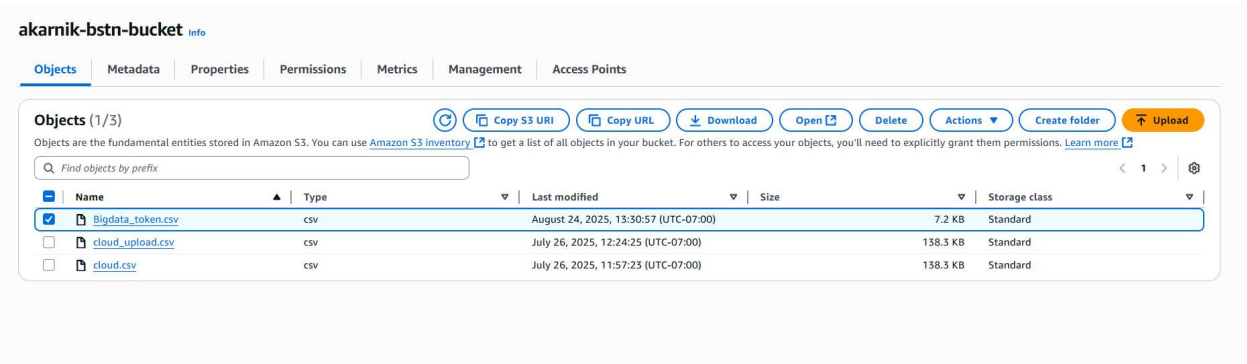

```
Found 3 items
-rw-r--r--   1 livy hdfsadmingroup          0 2025-08-24 20:18 /user/hadoop/Bigdata_token/_SUCCESS
-rw-r--r--   1 livy hdfsadmingroup         33 2025-08-24 20:16 /user/hadoop/Bigdata_token/part-00000-f4a5bc09-2f86-4471-85d6-de68dd058565-c000.csv
-rw-r--r--   1 livy hdfsadmingroup       7305 2025-08-24 20:17 /user/hadoop/Bigdata_token/part-00023-f4a5bc09-2f86-4471-85d6-de68dd058565-c000.csv
[hadoop@ip-172-31-32-150 ~]$
```

Q5-Screenshots

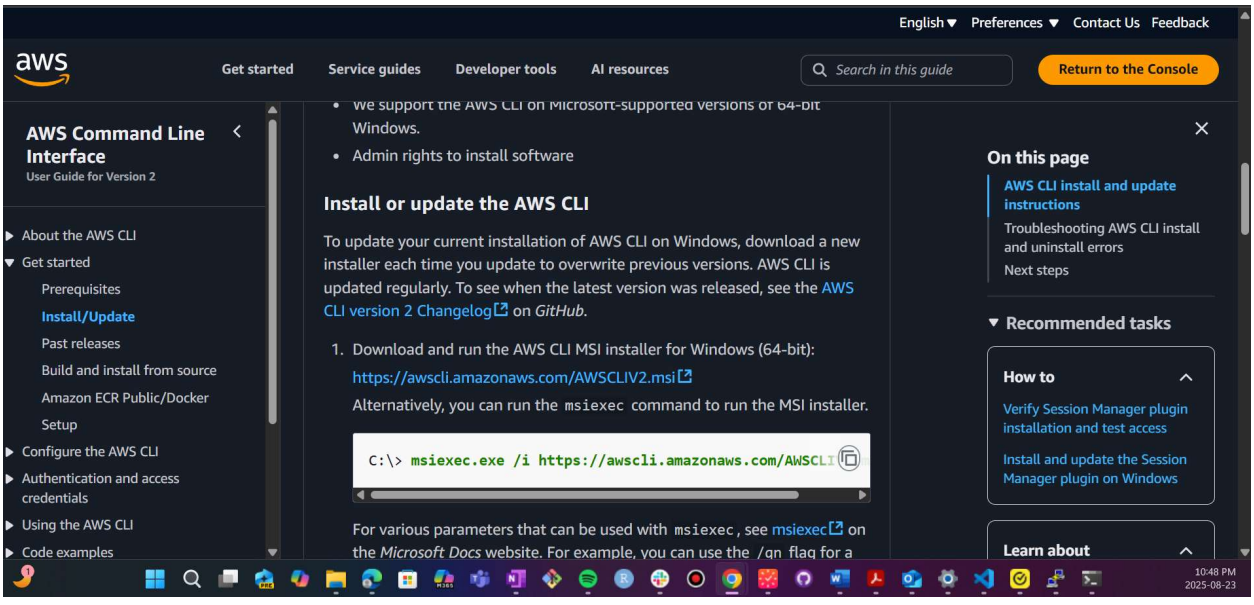
```
[hadoop@ip-172-31-32-150 ~]$ # merge to a single csv on the head node
[hadoop@ip-172-31-32-150 ~]$ hdfs dfs -getmerge -nl /user/hadoop/Bigdata_token ~/Bigdata_token.csv
[hadoop@ip-172-31-32-150 ~]$
```

```
[hadoop@ip-172-31-32-150 ~]$ # merge to a single csv on the head node
[hadoop@ip-172-31-32-150 ~]$ hdfs dfs -getmerge -nl /user/hadoop/Bigdata_token ~/Bigdata_token.csv
[hadoop@ip-172-31-32-150 ~]$ aws s3 cp ~/Bigdata_token.csv s3://akarnik-bstn-bucket/Bigdata_token.csv
upload: ./Bigdata_token.csv to s3://akarnik-bstn-bucket/Bigdata_token.csv
[hadoop@ip-172-31-32-150 ~]$
```





Q6-Screenshots



```
karni@Ark CLANGARM64 ~ (master)
$ aws configure
AWS Access Key ID [****************4JA4]: AKIAUSS74ORQL4N3RU2J
AWS Secret Access Key [****************i3BB]: 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
Default region name [us-east-2]:
Default output format [json]:
(base)
karni@Ark CLANGARM64 ~ (master)
$ aws s3 ls
2025-07-26 11:55:49 akarnik-bstn-bucket
2025-07-29 18:01:50 aws-logs-314807448672-us-east-2
2025-07-26 12:57:57 sagemaker-studio-314807448672-dd9c20l2ptf
2025-07-26 12:58:00 sagemaker-us-east-2-314807448672
(base)
karni@Ark CLANGARM64 ~ (master)
$
```