**Bank Loan data analysis**

In this project you are going to analyze the bank loan data set. You need to use HDFS, Hive, Spark and MongoDB.

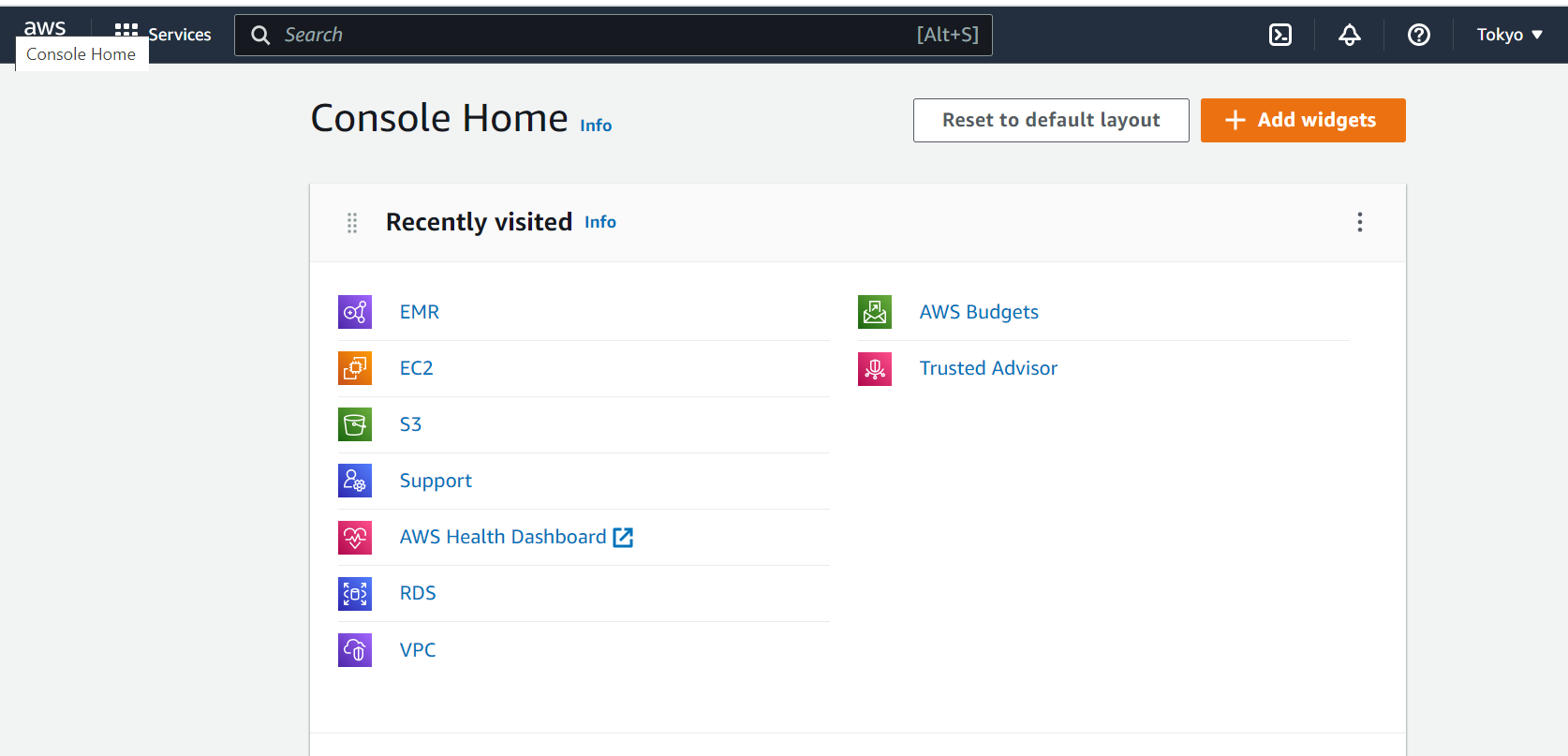
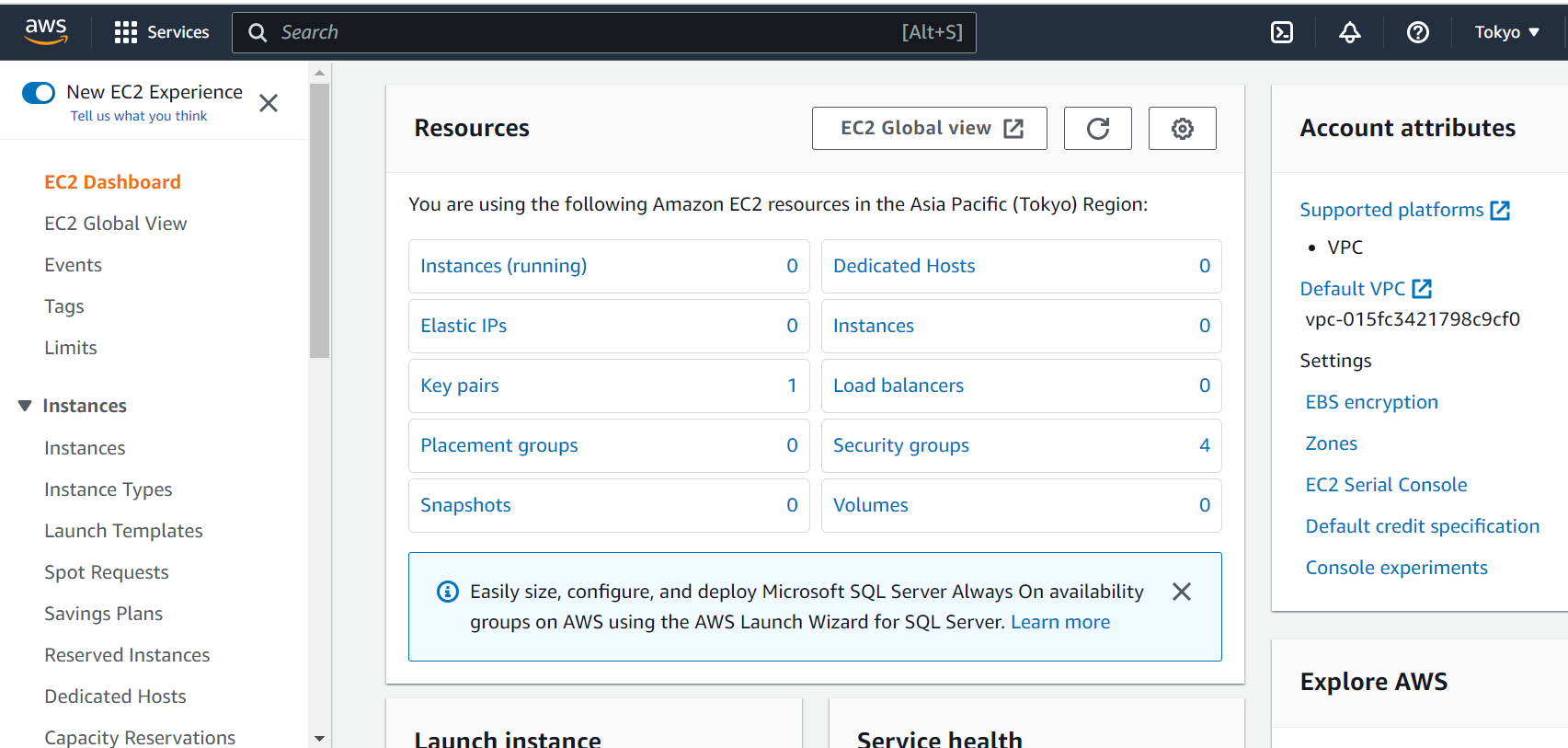
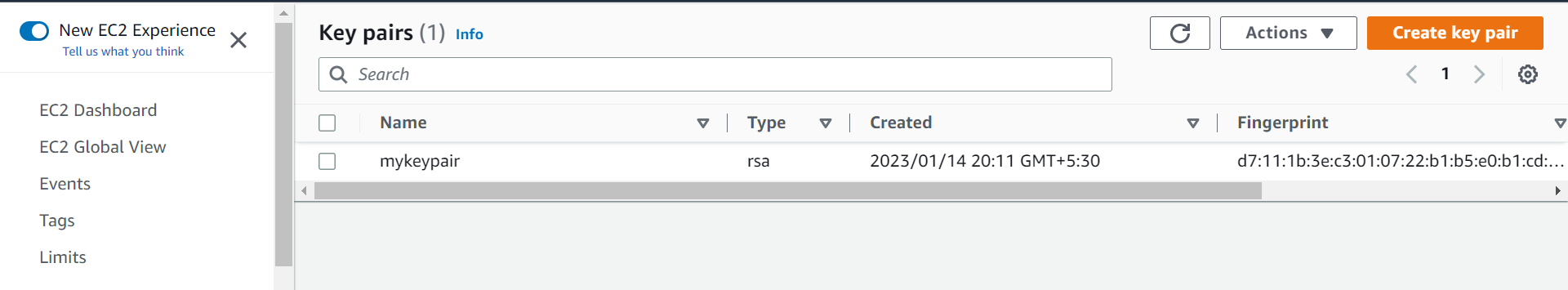
**Part 1:**

Get the data to HDFS and use Hive to create the required database.

**Part 2:**

Read the loan table from Hive and create the Dataframe in PySpark and upload data in MongoDB

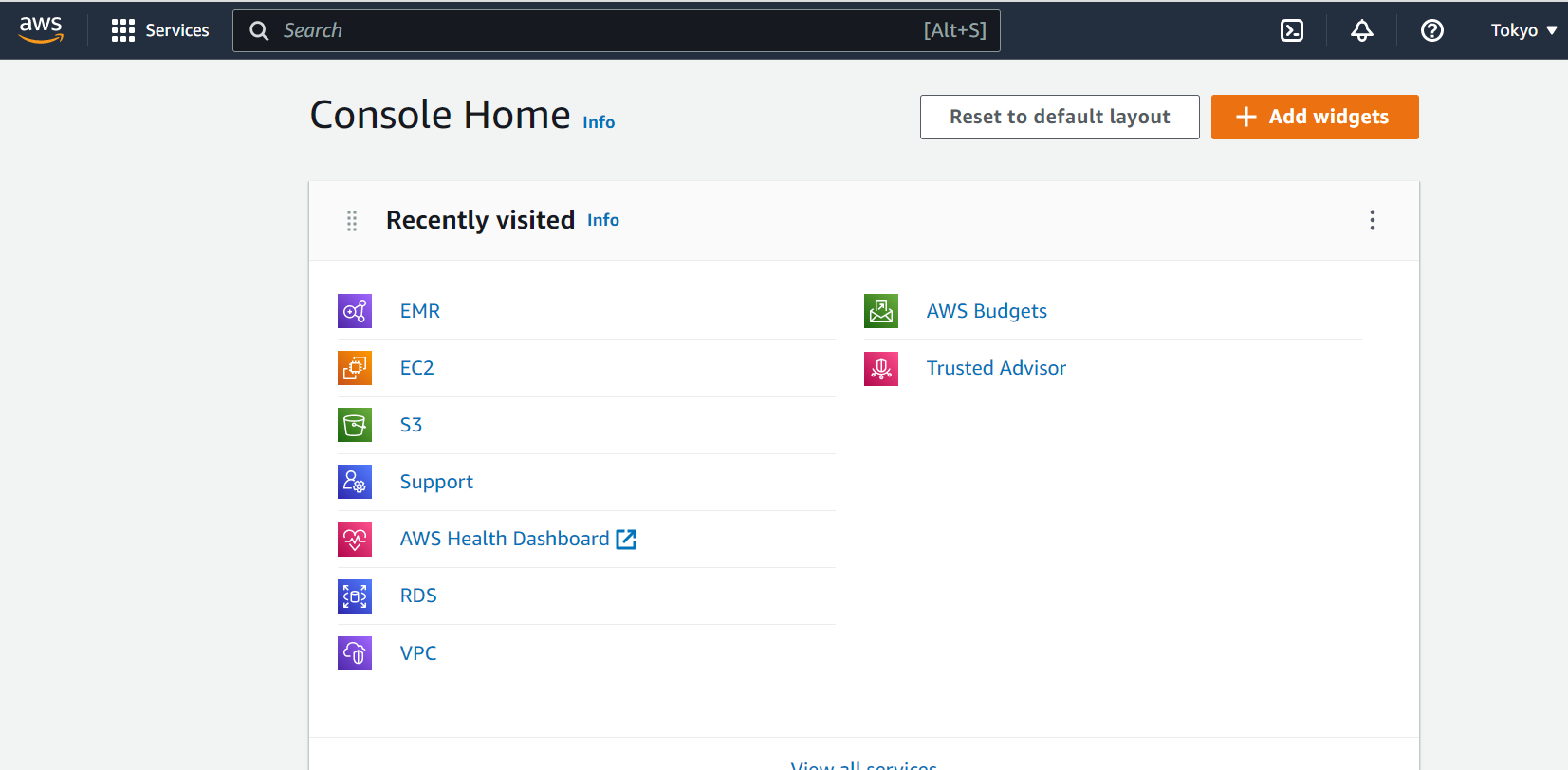
EC 2 Key Pair Setup:

* Click on services
* 
* Click on EC2
* 
* Click on Key Pairs
* 
* Click on Create keypair
* Give Name as you want (I gave project2), Key pair type: RSA, Private key file format: .ppk
* Click on ‘Create key pair’
* After this Key pair will be downloaded automatically

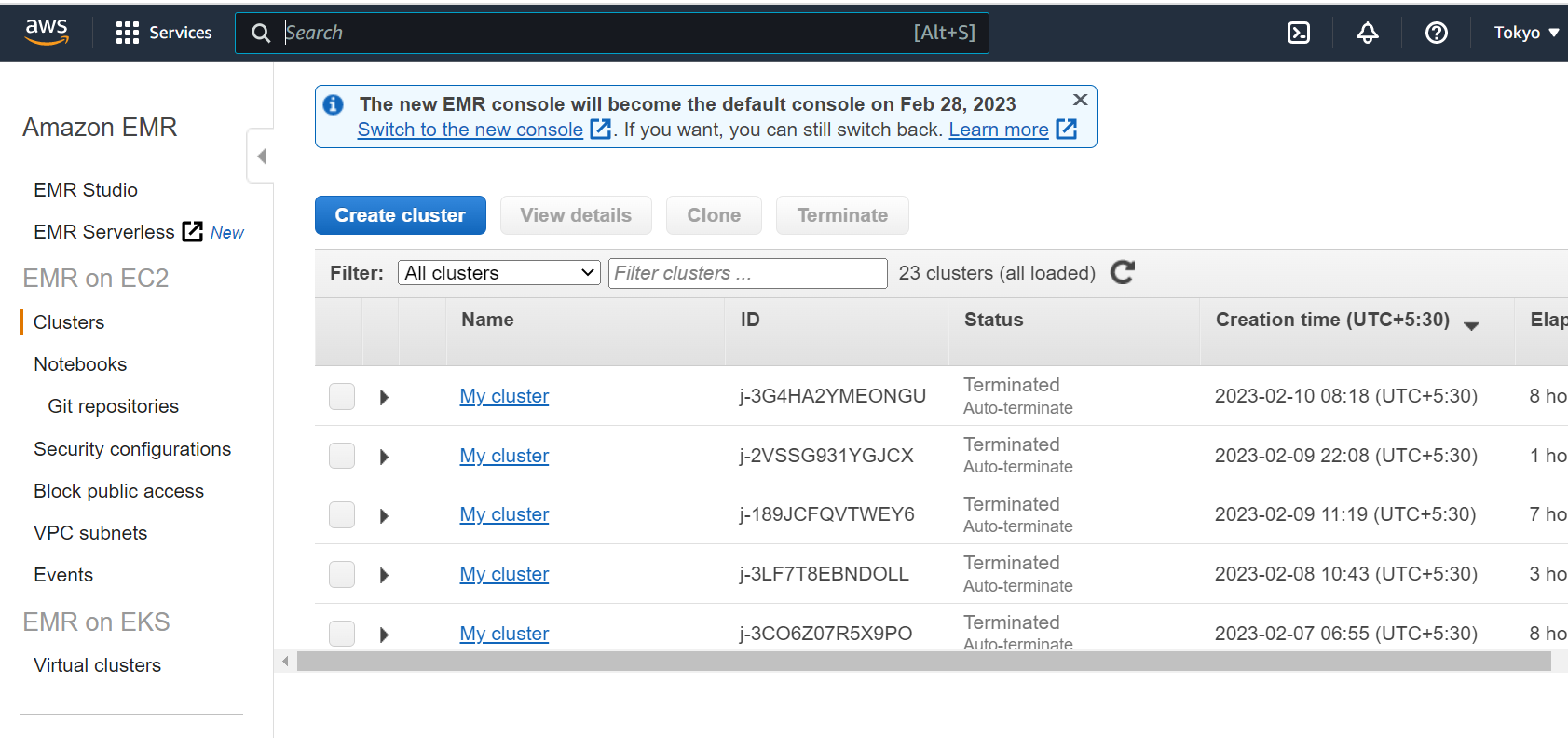
Step1(After Creating Key Pair):

* I am going to use AWS services for implementing HDFS, Hive, PySpark, MongoDB operations
* Login to the AWS, and create EMR cluster
* Creating EMR cluster using following procedure (Must have AWS account):

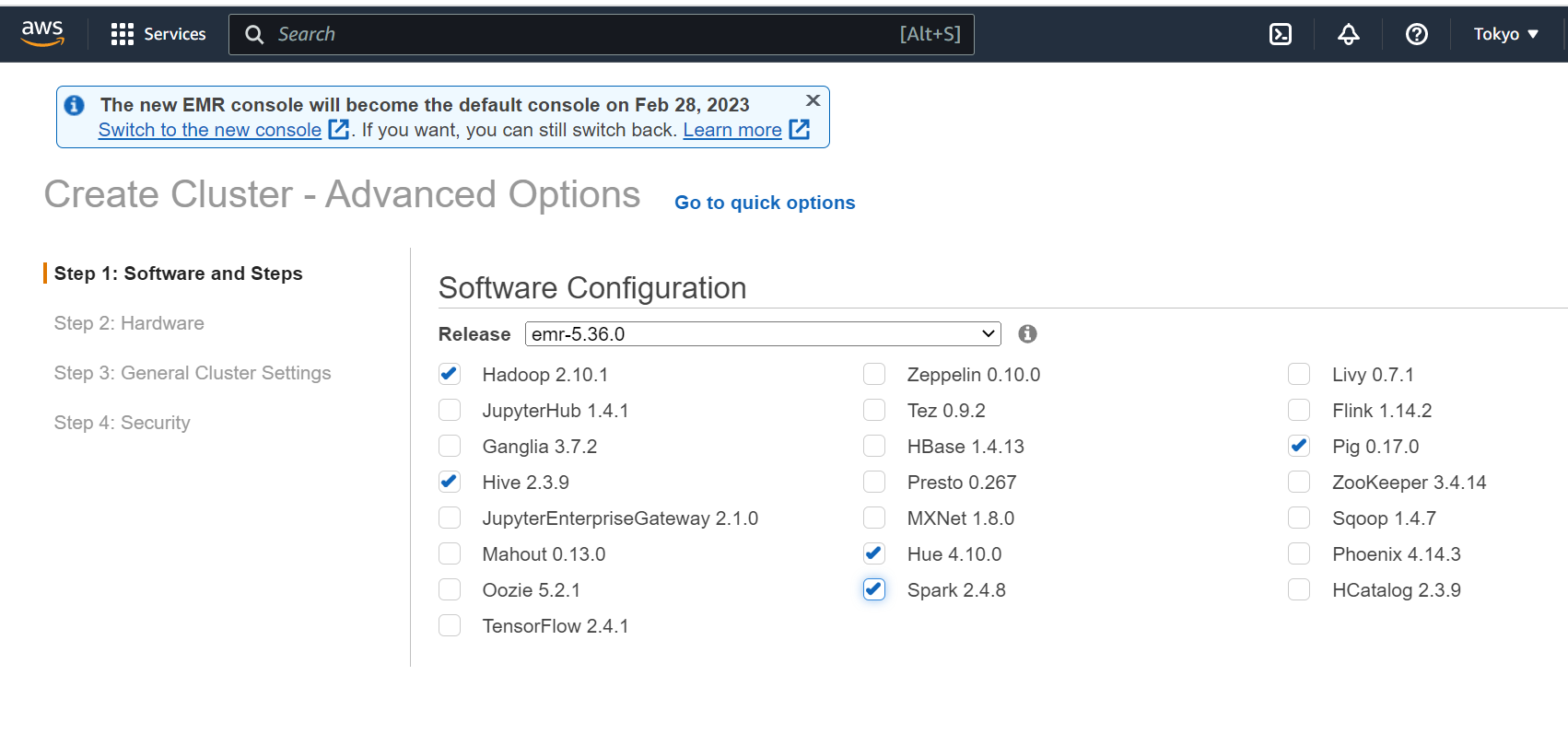
1. Log in to the AWS
2. Click on Services



1. Click on EMR

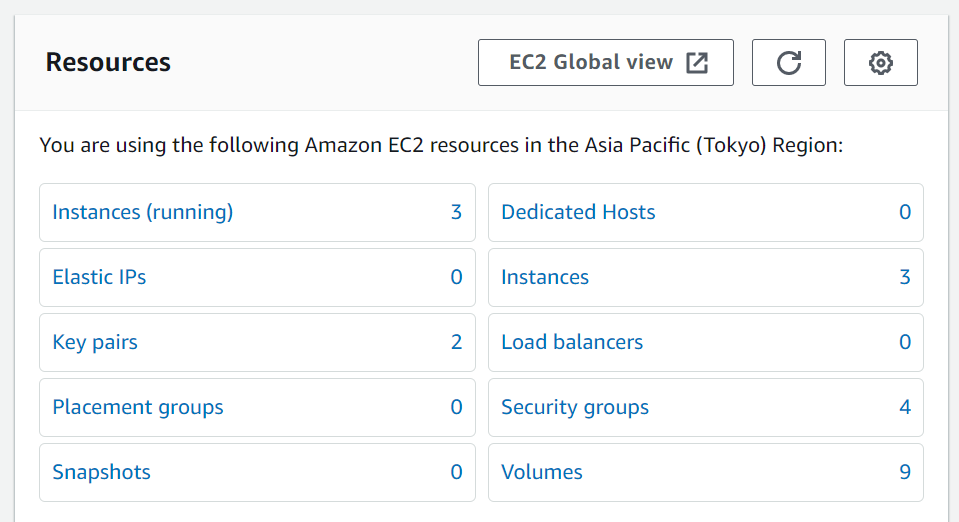
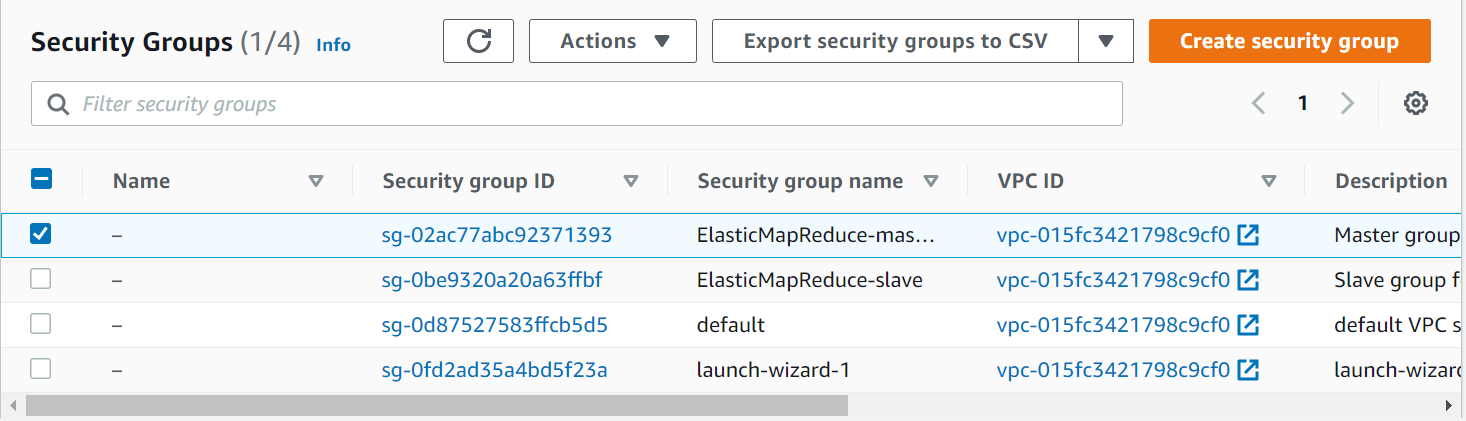
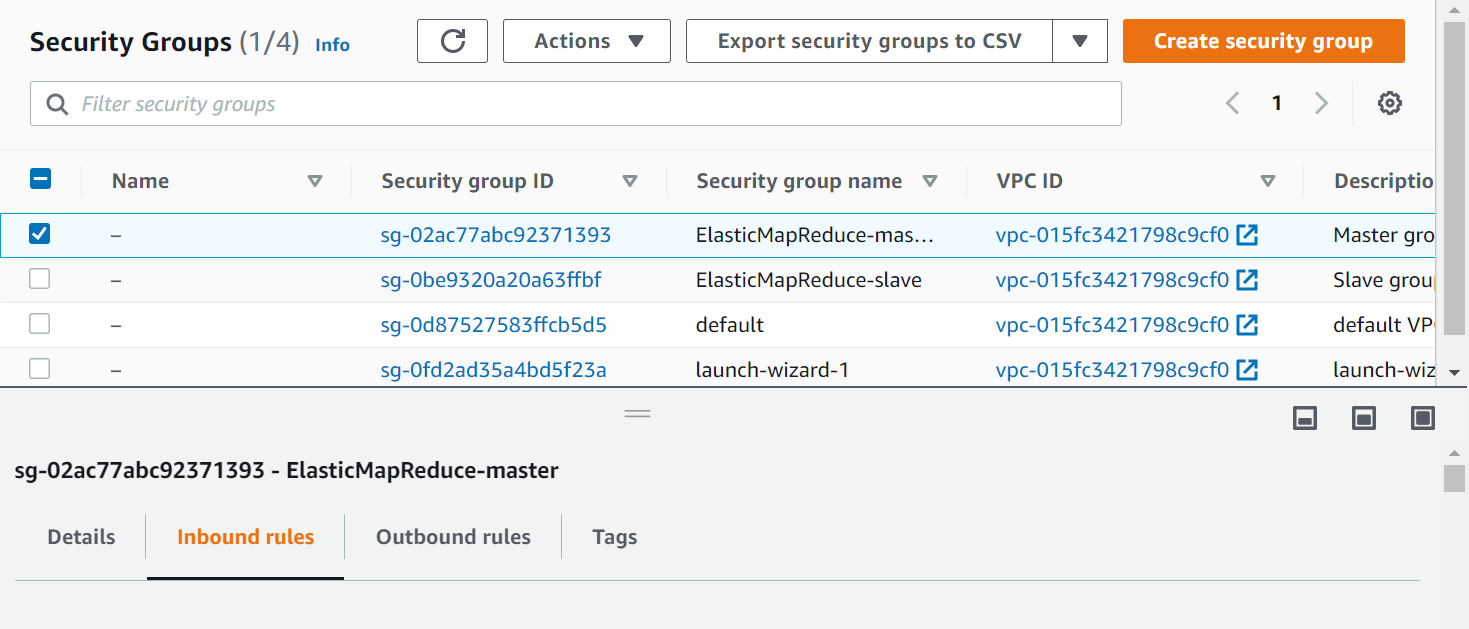
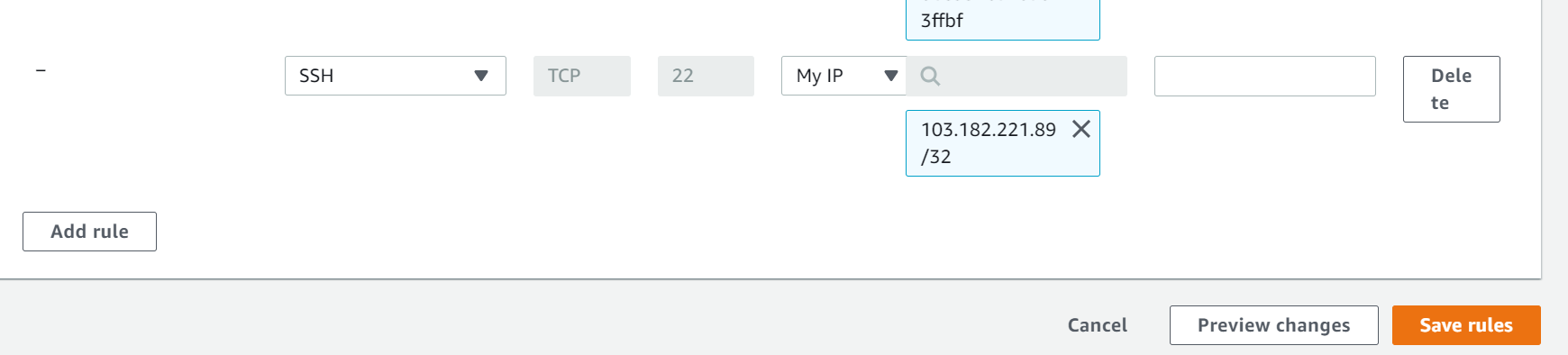


1. Click on Create Cluster
2. After redirecting on to the creating cluster page click on ‘Go advance options’

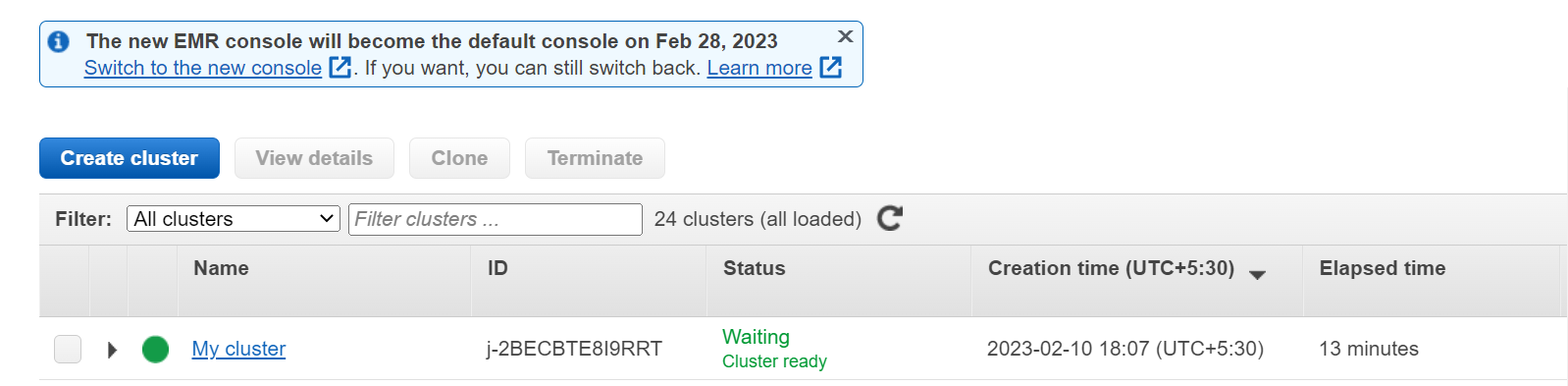
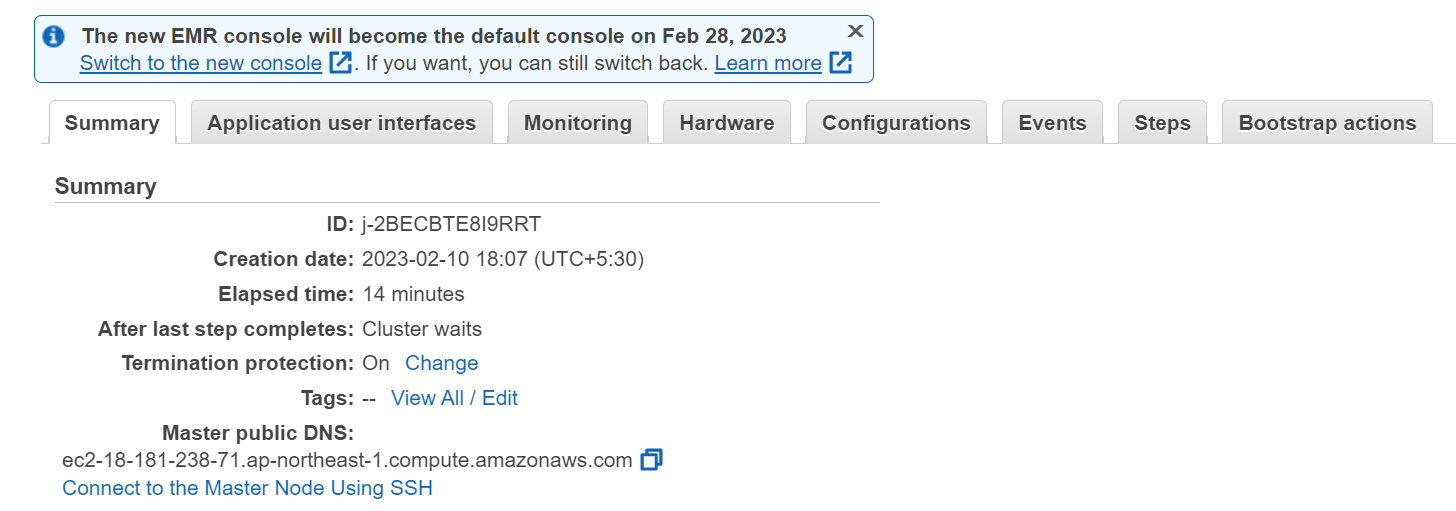
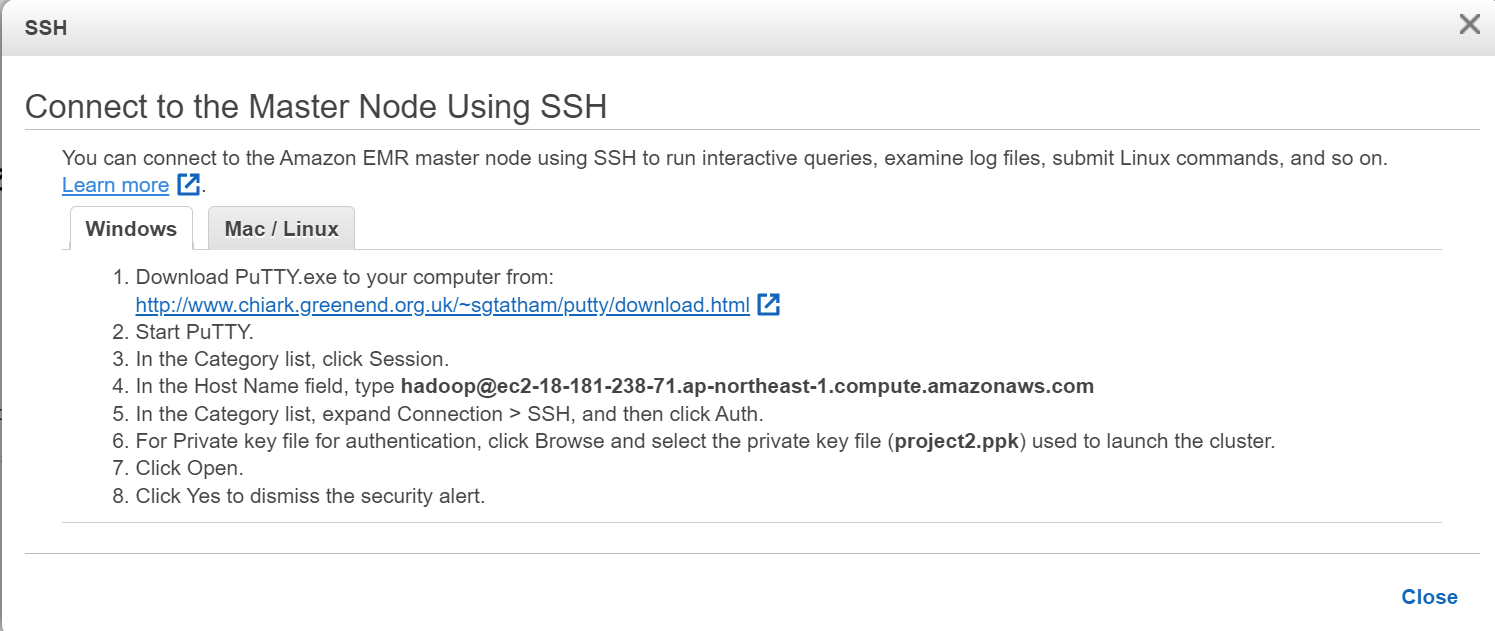
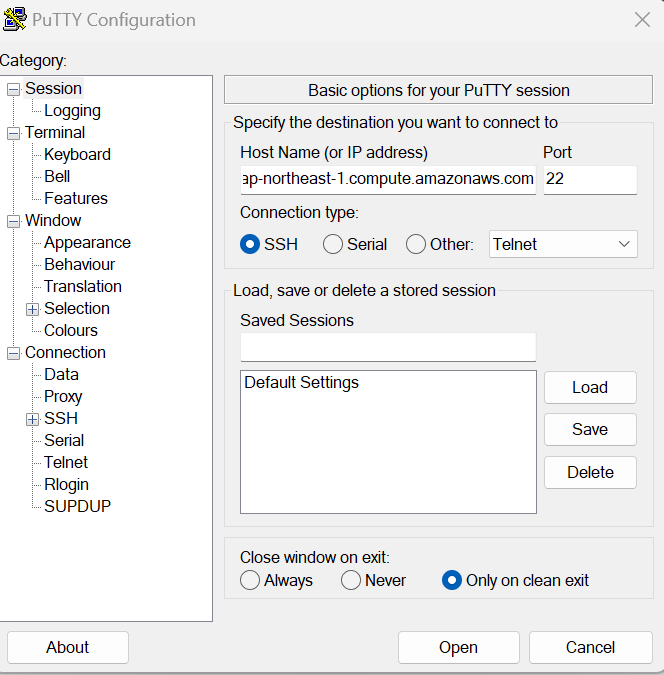
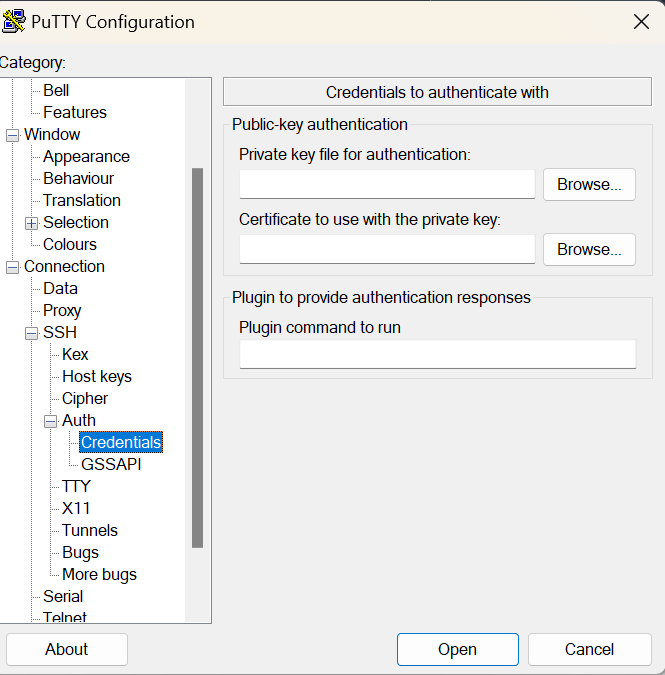
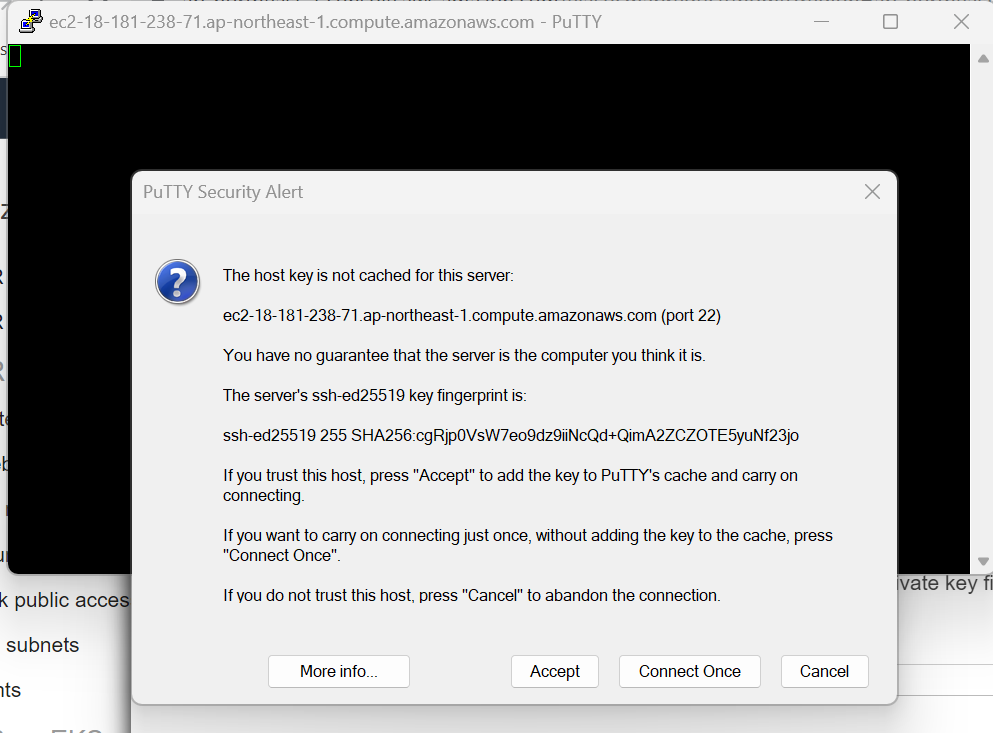


1. After selecting check boxes as given in above image go to next till Security Option
2. And select EC2 key pair as you created earlier. I selected project2

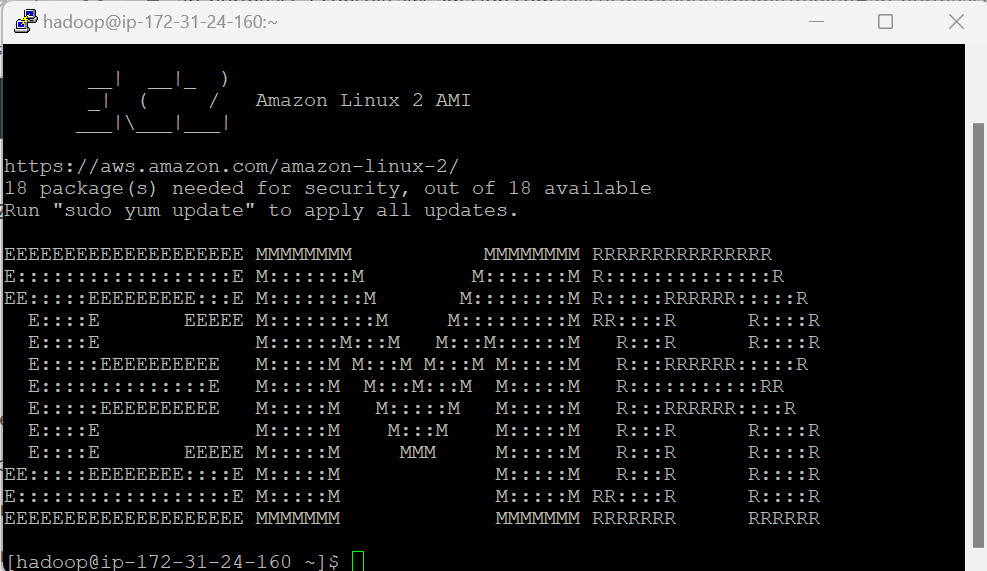
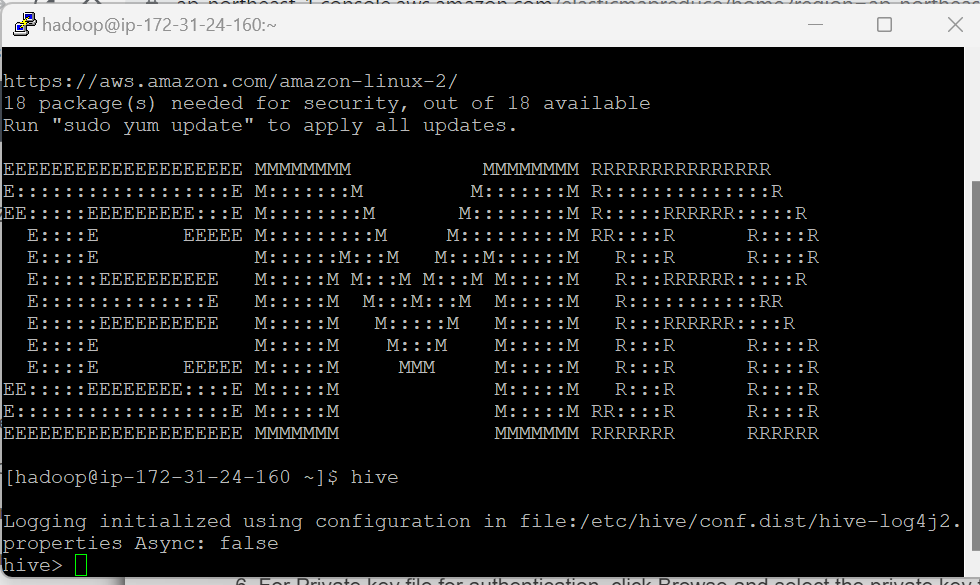
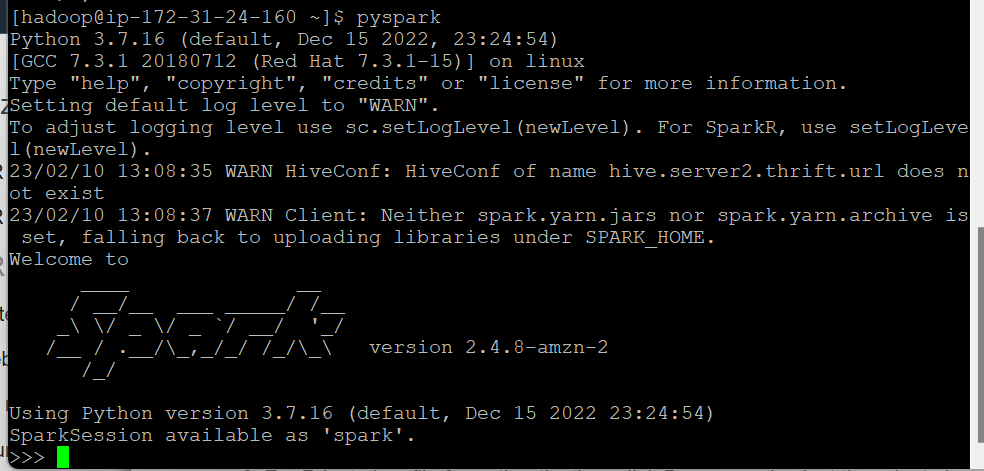
Step2(after creating EMR cluster):

* Need to add some rules in Security Groups to access EMR using SSH.
* Click on Services
* Go to EC2 Services
* 
* Click on Security groups:
* 
* Check MasterGroup for Elastic MapReduce
* Scroll down and click on Inbound rules
* 
* Scroll down and Click on Edit inbound rules
* Scroll down and click on Add rule
* Select SSH in Type and Source is My IP
* 
* Click on Save Rules

Step 3(After adding rule in Security group):

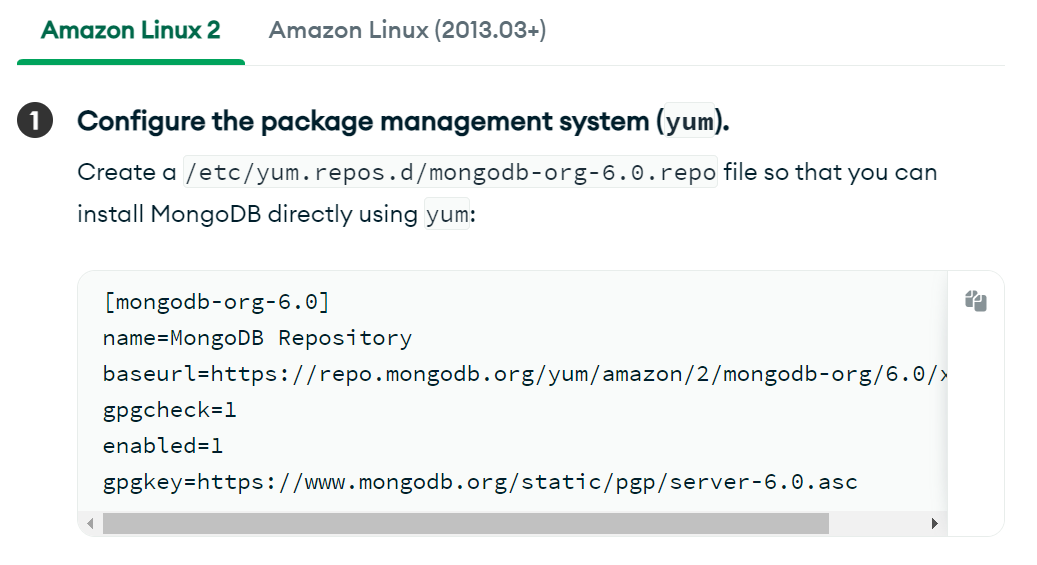
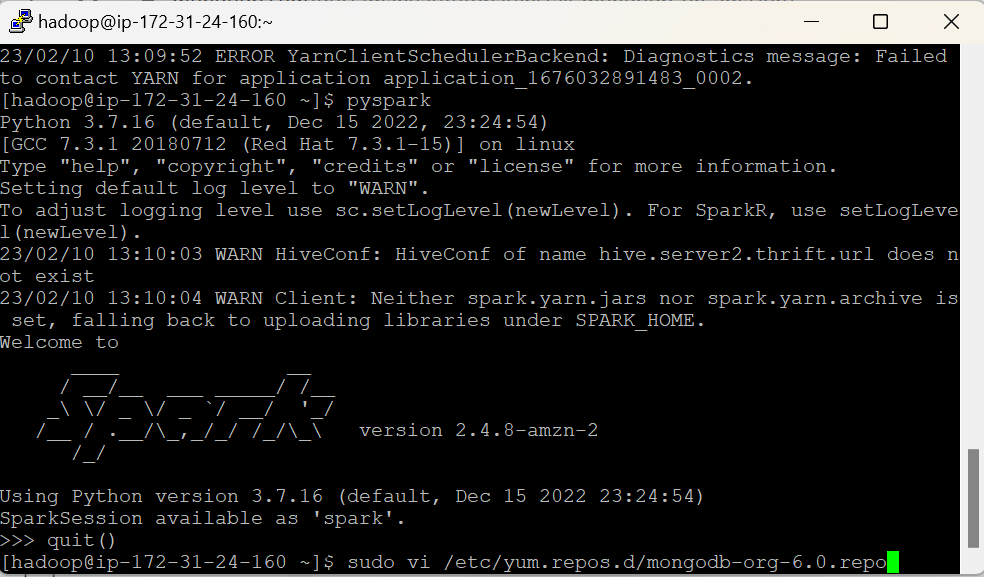
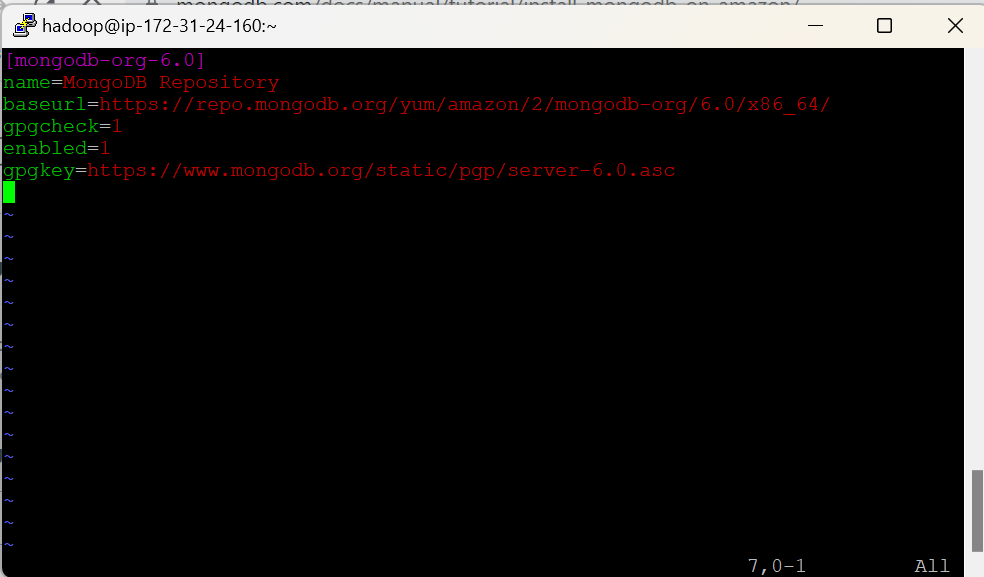
* Click on EMR services
* 
* Click on you created cluster. (My Cluster)
* 
* Click on Connect to the Master Node using SSH
* 
* Download Putty
* And copy 4 statement ([hadoop@ec2-18-181-238-71.ap-northeast-1.compute.amazonaws.com](mailto:hadoop@ec2-18-181-238-71.ap-northeast-1.compute.amazonaws.com))
* And Open PuTTY
* Paste copied data in Host Name
* 
* Click on SSH in Category
* Then Click on Auth
* After Click on Credentials
* 
* In Private key for authentication: Select that .ppk file you created earlier.
* And click on Open
* 
* Click on Accept

Step 4 (Check Hive and PySpark ):

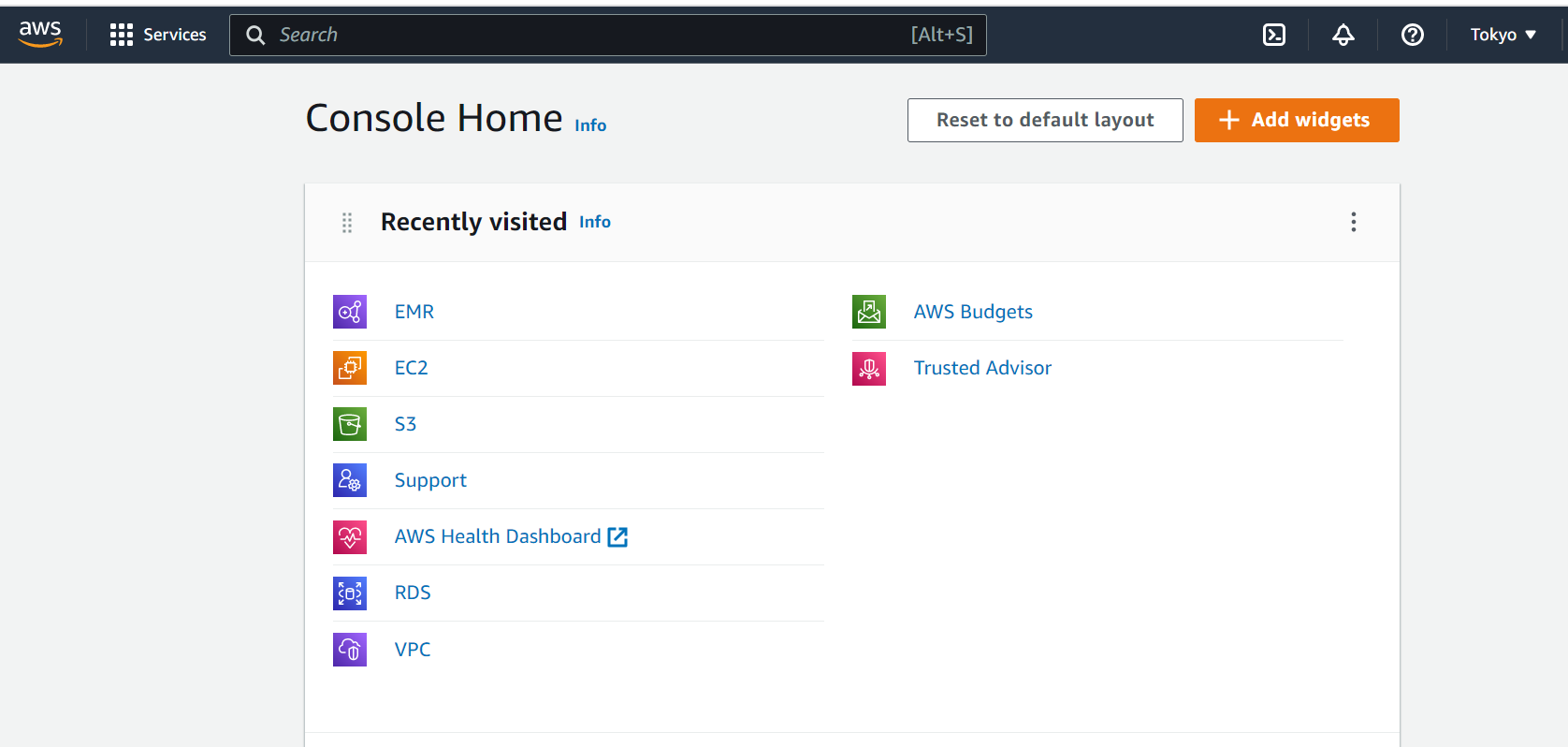
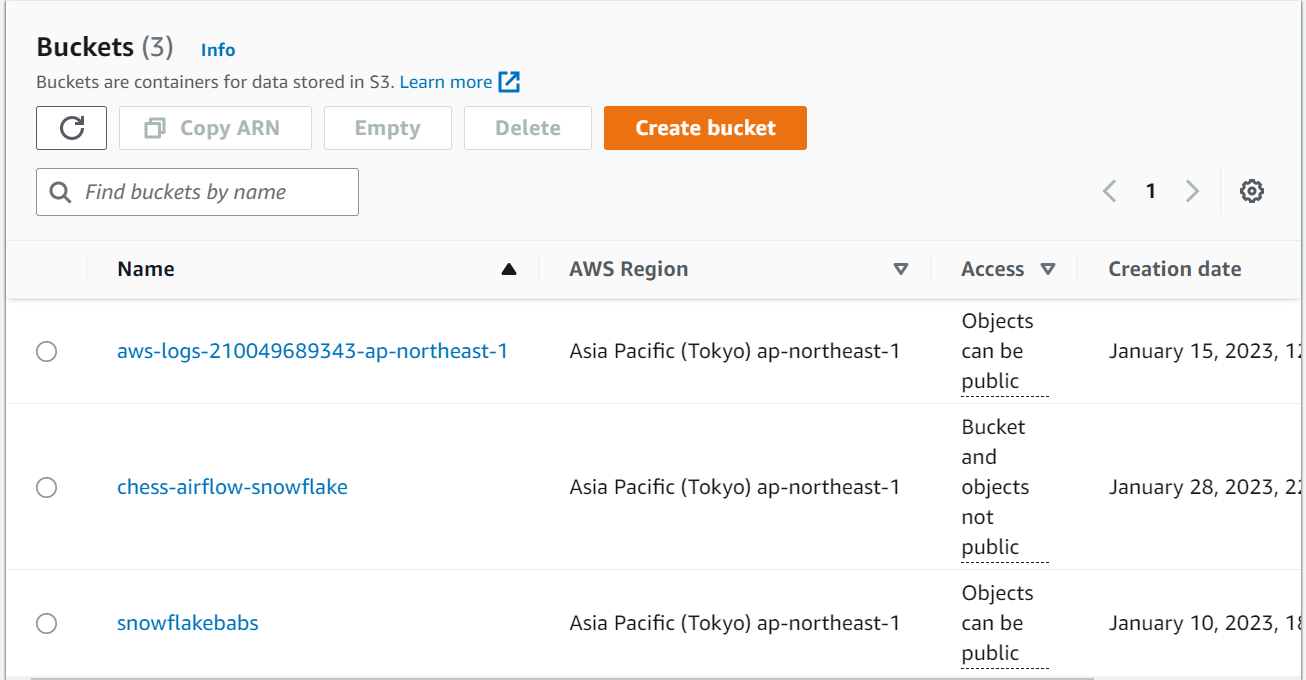
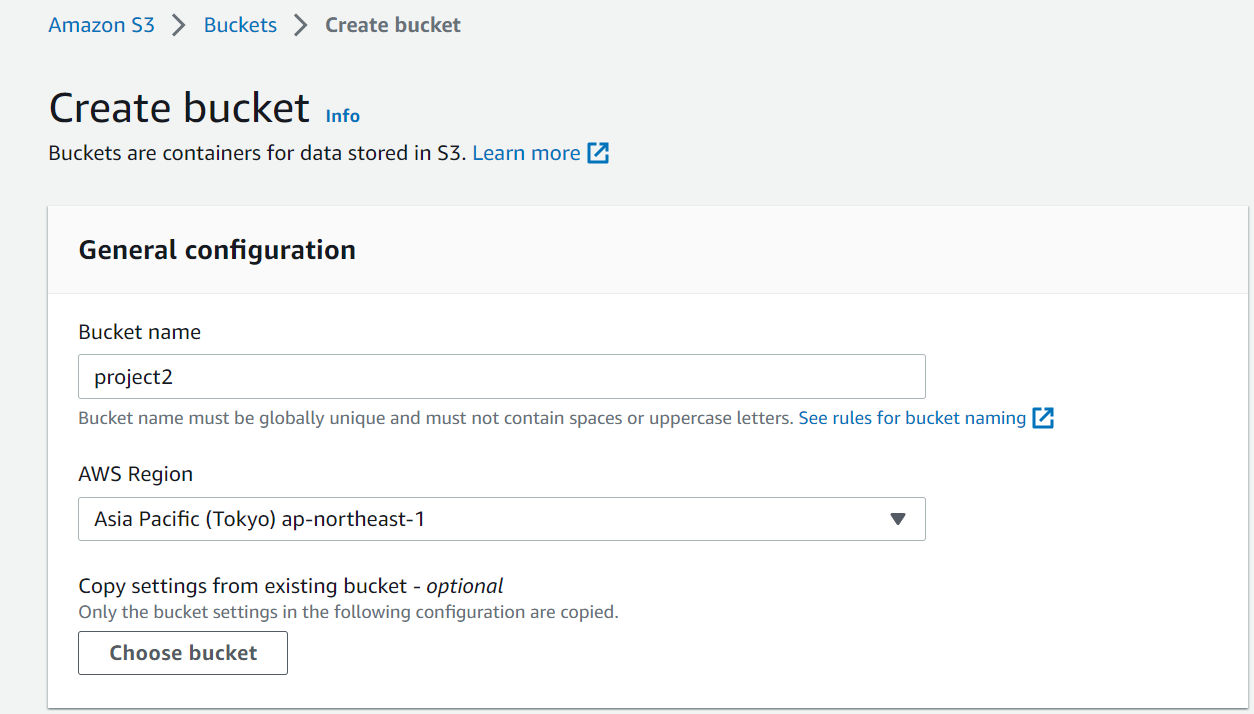
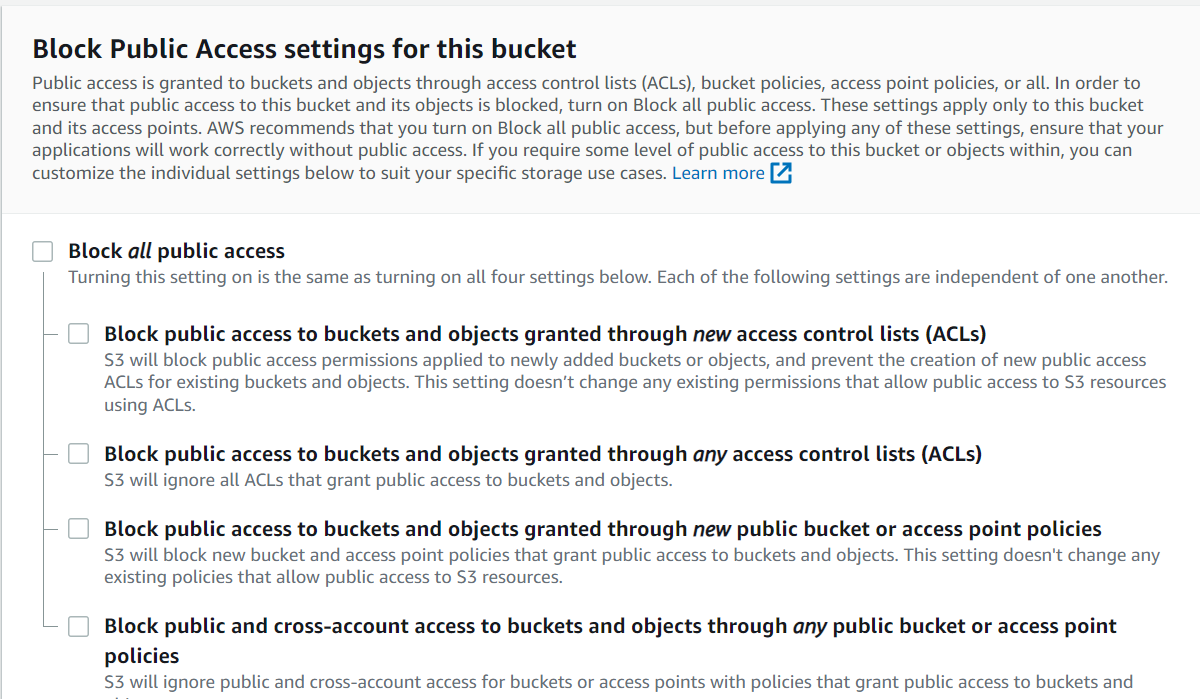
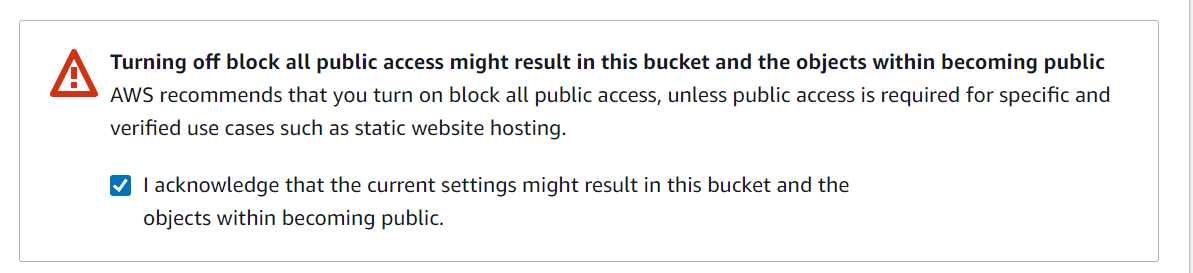
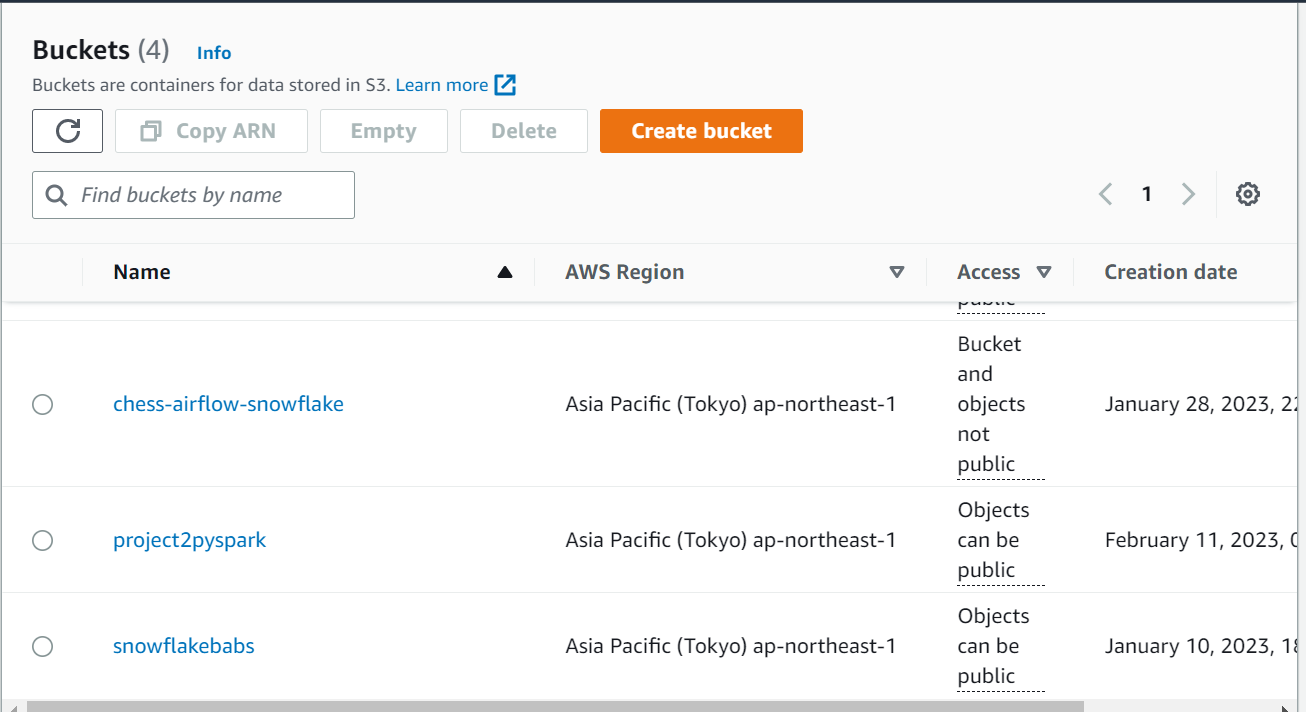
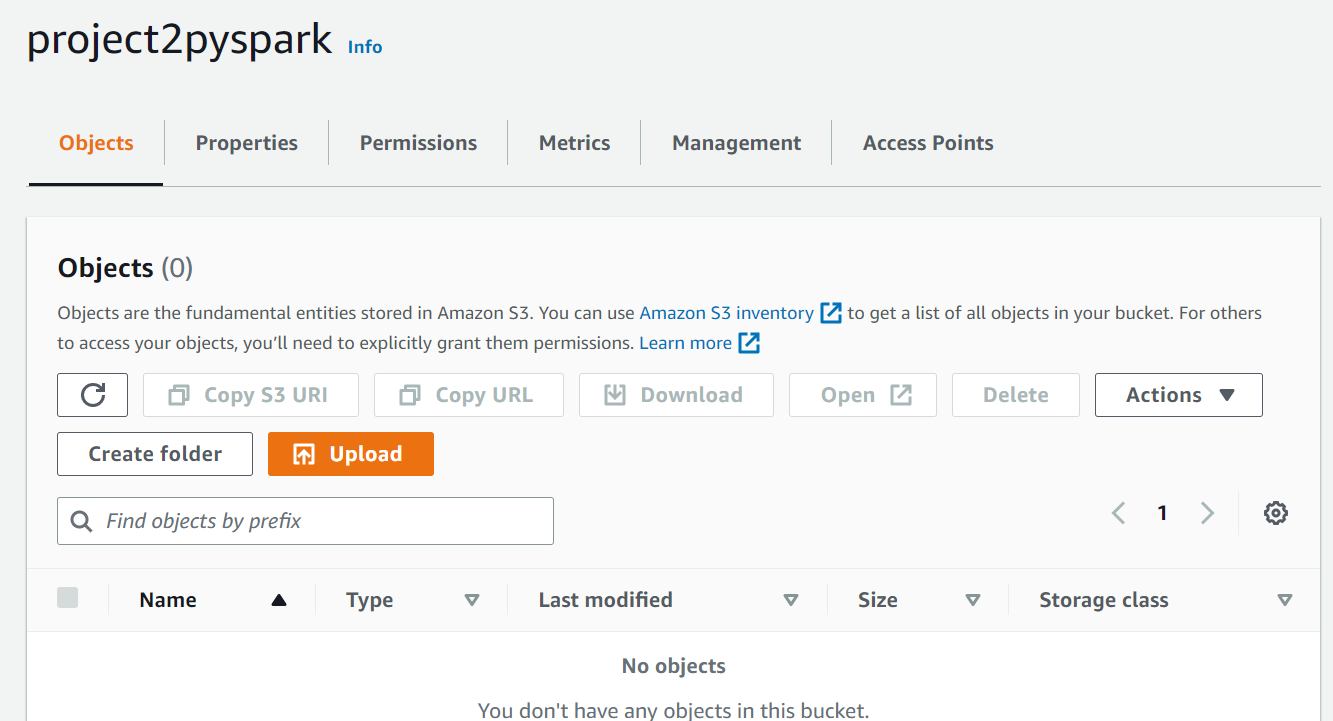
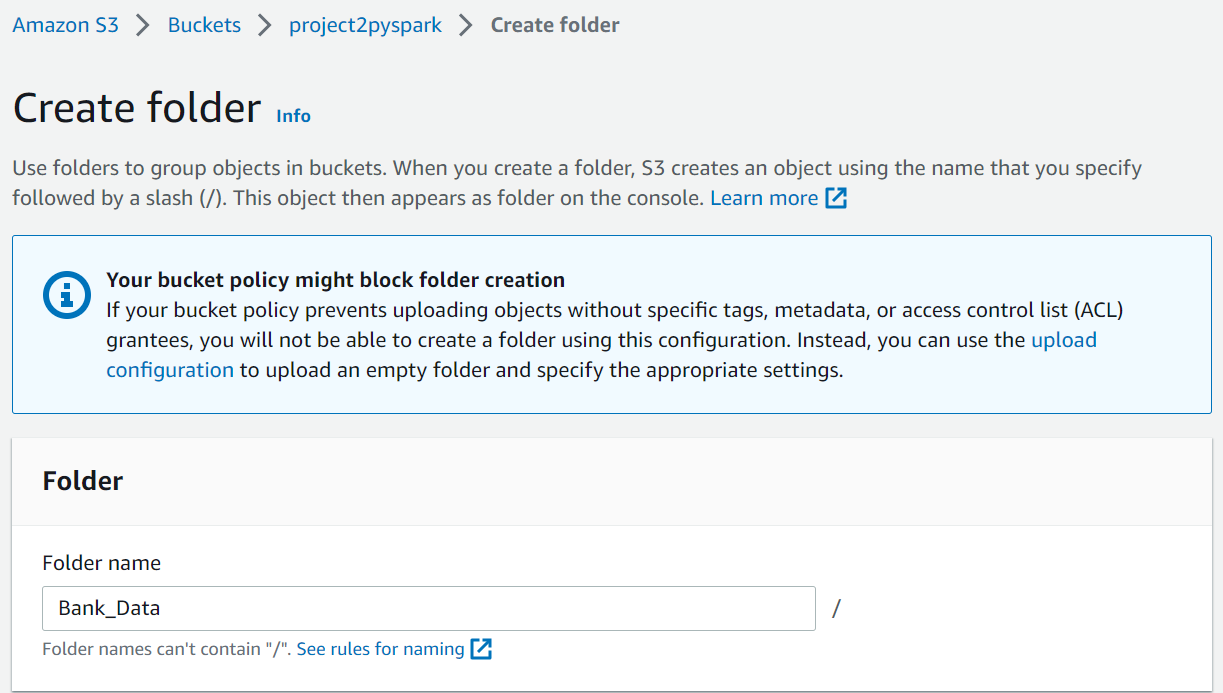
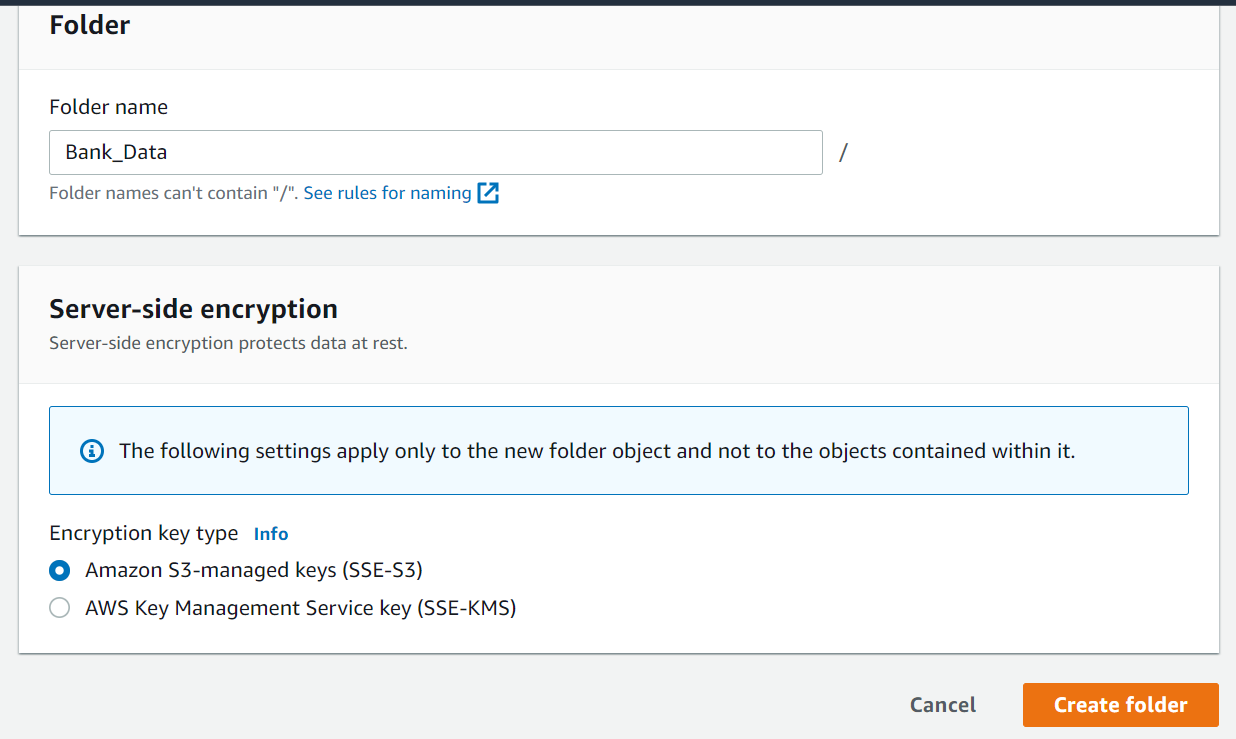
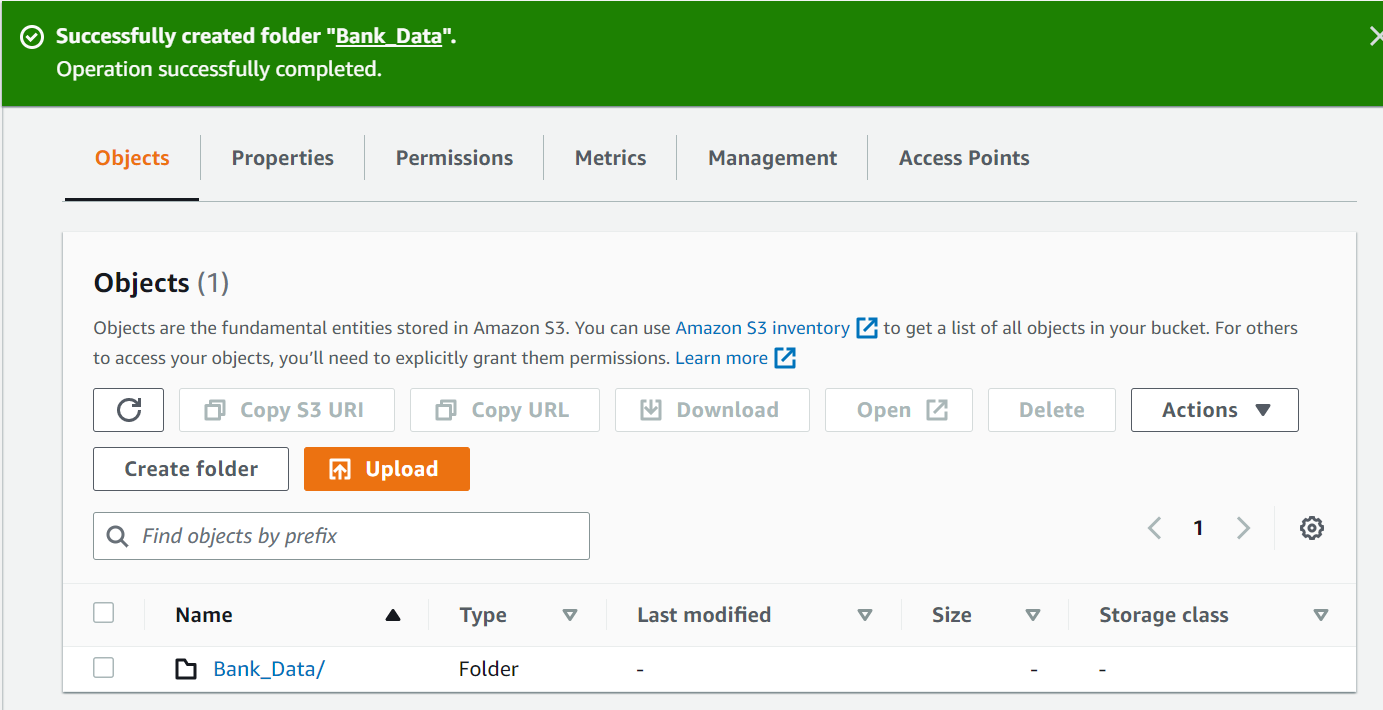
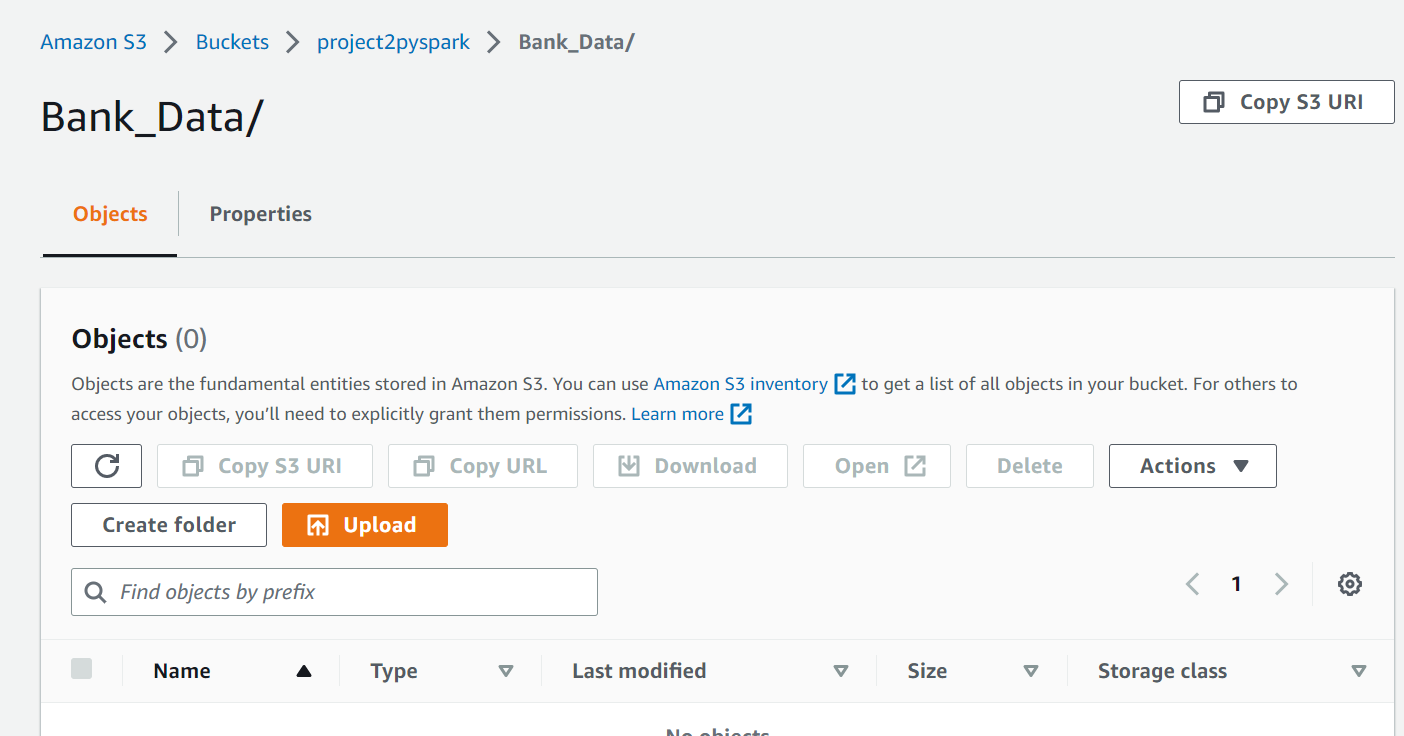
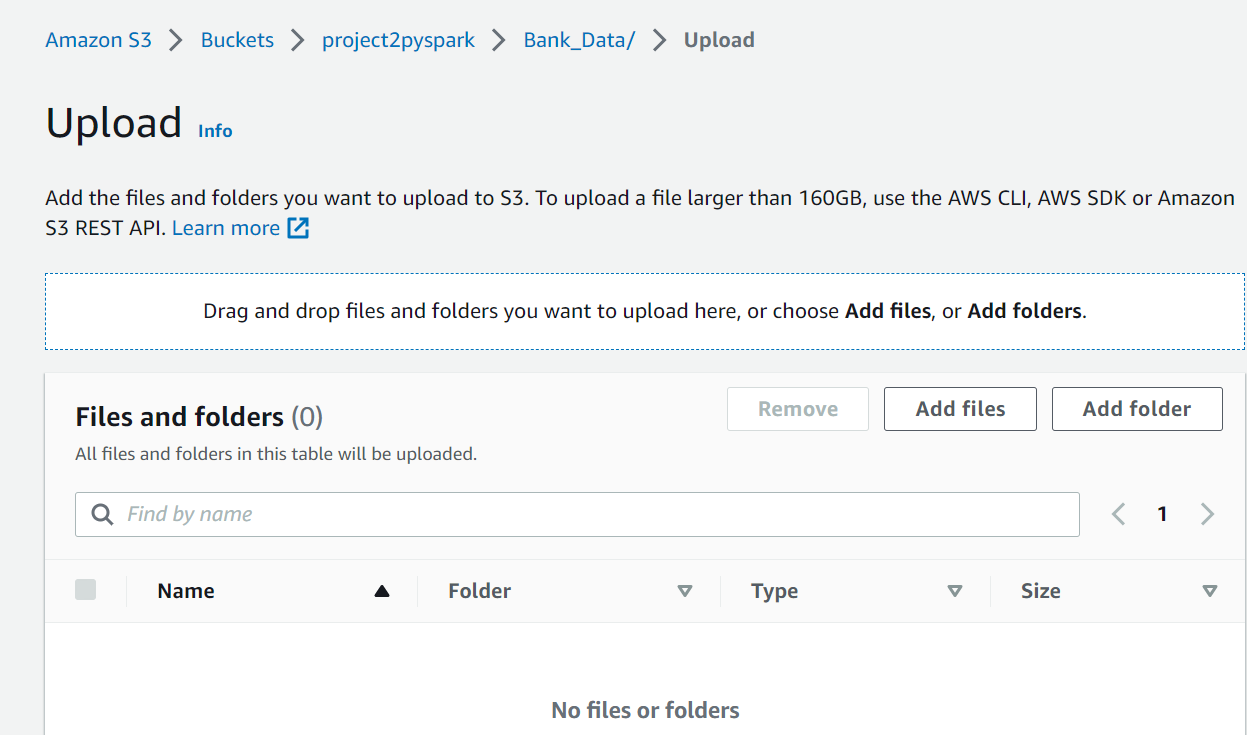
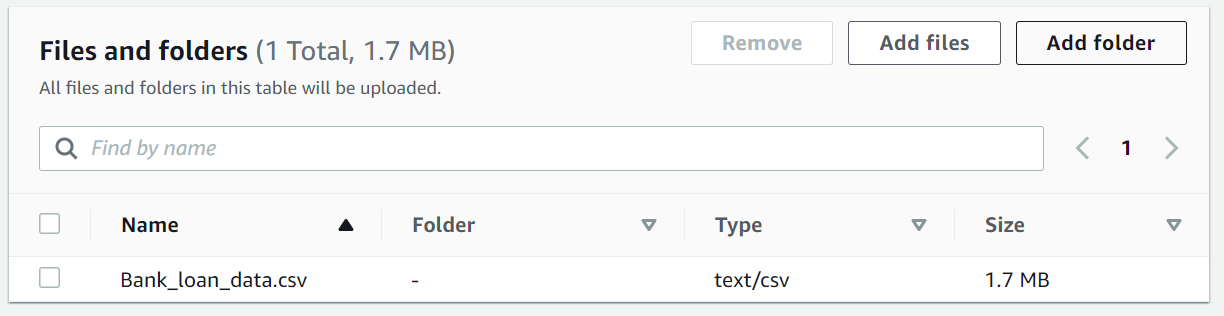
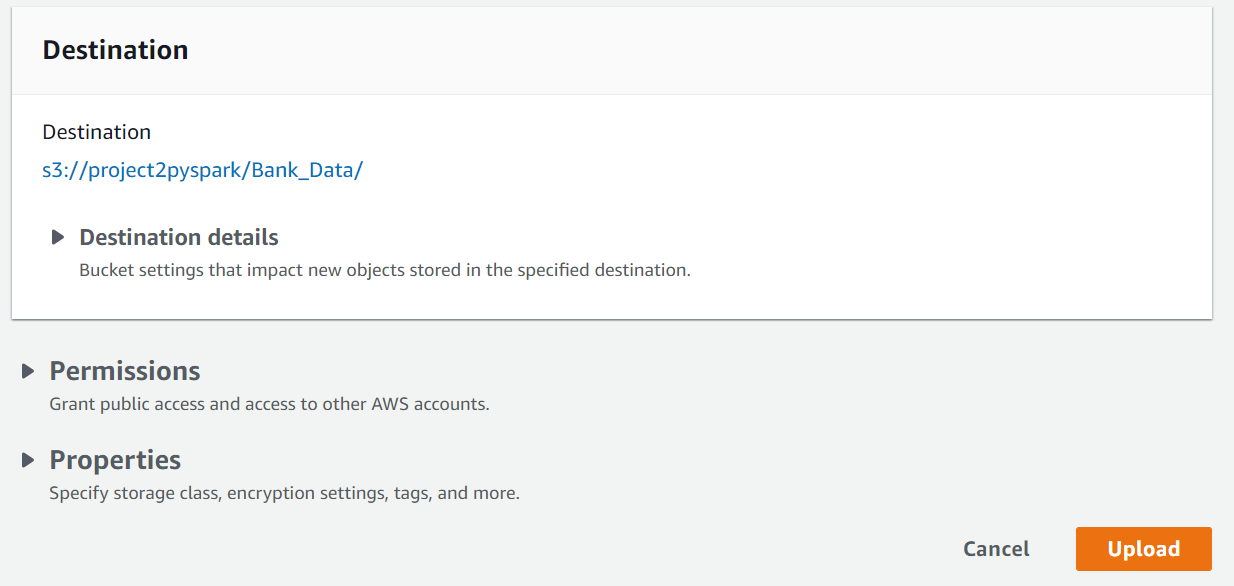
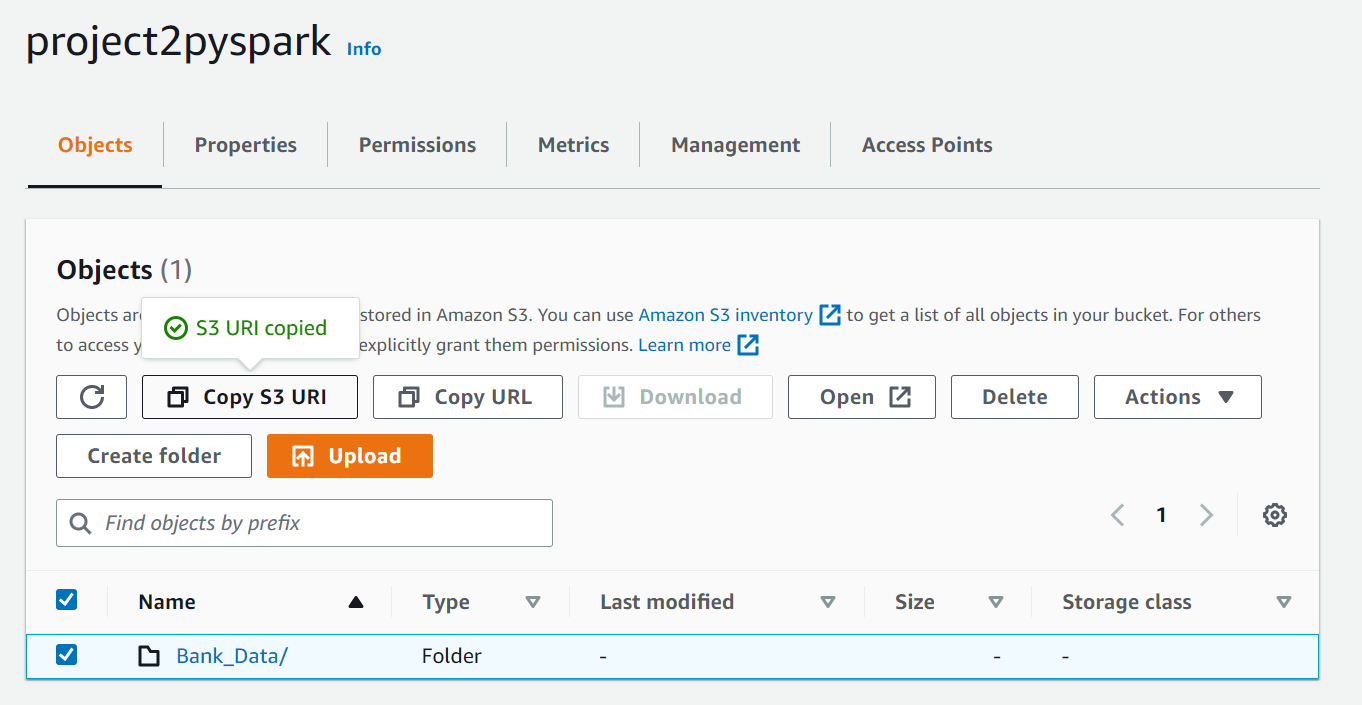
* Once you accepted then your EMR cluster connected through SSH following:
* 
* Type ‘hive ‘ and check and if hive is seen then hive is working
* 
* Click ‘ctrl + c’ or type ‘exit;’ to exit from hive
* And type pyspark to check pyspark is working or not and you will find following result
* 
* Type quit() to exit.

Step 5 (Install MongoDB):

<https://www.mongodb.com/docs/manual/tutorial/install-mongodb-on-amazon/>

* To install MongoDB open above given link
* Follow the steps for Amazon Linux2
* 
* Type following to create file and install MongoDB
* Sudo vi /etc/yum.repos.d/mongodb-org-6.0.repo
* 
* Copy and paste following lines:
* **[mongodb-org-6.0]**
* **name=MongoDB Repository**
* **baseurl=https://repo.mongodb.org/yum/amazon/2/mongodb-org/6.0/x86\_64/**
* **gpgcheck=1**
* **enabled=1**
* **gpgkey=https://www.mongodb.org/static/pgp/server-6.0.asc**
* ****
* Press ‘Esc ‘
* After press ‘ZZ’
* To come out from the editor
* Type to install MongoDB: sudo yum install -y mongodb-org
* Type to start MongoDB : sudo systemctl start mongod
* Type to check status of MongoDB: sudo systemctl status mongod
* Type mongosh to open MongoDB shell

Step 6(Uploading CSV file to S3 bucket):

* Click on S3 services
* 
* Click on Create bucket
* 
* 
* Give unique name(project2 is already exist so I gave project2pyspark)
* 
* Uncheck Block all public access
* 
* Check above public access check box
* Click on Create bucket.
* Now in Buckets you can see your Bucket
* 
* Click on your Bucket
* 
* Click on Create folder
* 
* Give folder name
* Click on Create Folder
* 
* Now click on to your Folder
* 
* Click on Upload
* 
* Click on Add files
* 
* Select csv file from your computer.
* 
* Click on Upload
* 
* Your file get uploaded successfully
* Now in S3 services, Click on your Bucket
* After that check your folder to copy S3 URI
* 
* That copied URI is required while creating table in Hive using AWS EMR

Hive:

* **Create database:**

create database bank\_loan;

use bank\_loan;

* **Create Table:**

create external table loan\_data

(

loan\_id string,

customer\_id string,

current\_loan\_amount decimal(10,2),

term string,

credit\_score decimal(10,2),

annual\_income decimal(10,2),

years\_current\_job string,

home\_ownership string,

purpose string,

monthly\_debt decimal(10,2),

years\_of\_credit\_history decimal(10,2),

months\_since\_last\_delinquent string,

number\_of\_open\_accounts int,

number\_of\_credit\_problems int,

current\_credit\_balance decimal(10,2),

max\_open\_credit decimal(10,2),

bankruptcies int,

tax\_liens int

)

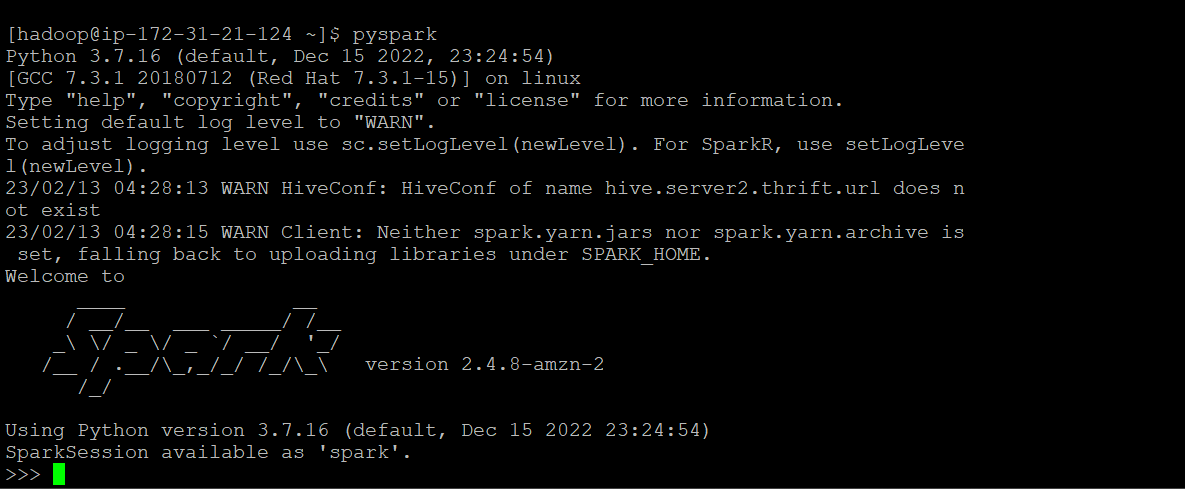
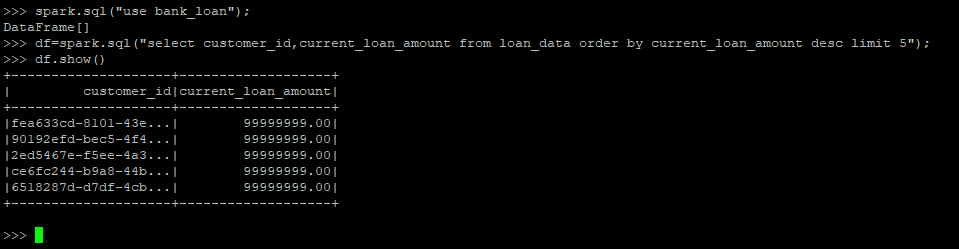
row format delimited fields terminated by ','

location 's3://project2pyspark/Bank\_Data/'

TBLPROPERTIES ("skip.header.line.count"="1");

* **In location put that S3 Bucket’s copied URI**
* I created external table so data is automatically loaded from csv file which was already there in S3 bucket.
* Now We can go to PySpark and access this table.
* Before going into PySpark environment we have to install Pandas,PyMogno

**Pyspark:**

* Type PySpark
* ****
* Using spark.sql we can access hive database in PySpark. As shown in below example.
* ****

**Below are Question-answers of project:**

**1. Create a dataframe with the Customer IDs having top 5 current loan amounts and save the result in MongoDB?**

select \* from loan\_data order by current\_loan\_amount desc limit 5;

spark.sql("use bank\_loan");

df=spark.sql("select customer\_id,current\_loan\_amount from loan\_data order by current\_loan\_amount desc limit 5");

df.show();

import pymongo

from pymongo import MongoClient

client = MongoClient("mongodb://127.0.0.1:27017/",27017)

dbs = client.list\_database\_names()

db = client["bank\_loan"]

coll = db.top\_5\_loan

using pandas

import pandas as pd

pdf = pd.toPandas()

pdf['current\_loan\_amount']=pdf['current\_loan\_amount'].astype('str')

records=pdf.to\_dict(orient='records')

db.coll.insert\_many(records)

cursor = db.coll.find()

for rec in cursor:

print(rec)

**2. Create a dataframe with the Customer IDs having the lowest 5 current loan amounts and save the result in MongoDB?**

spark.sql("use bank\_loan");

df=spark.sql("select customer\_id,current\_loan\_amount from loan\_data order by current\_loan\_amount limit 6");

df.show();

pdf=df.toPandas();

res = pdf.dropna(axis=0);

res['current\_loan\_amount']=res['current\_loan\_amount'].astype('str')

records=res.to\_dict(orient='records')

import pymongo

from pymongo import MongoClient

client = MongoClient("mongodb://127.0.0.1:27017/",27017)

dbs = client.list\_database\_names()

db = client["bank\_loan"]

coll1 = db.low\_5\_loan

db.coll1.insert\_many(records)

cursor = db.coll1.find()

for rec in cursor:

print(rec)

**3. Create a dataframe of Customer IDs who have taken the Short Term Loan and include their total Current Loan Amount and save the result in MongoDB?**

spark.sql("use bank\_loan");

df=spark.sql("select customer\_id,current\_loan\_amount,term from loan\_data where term='Short Term'");

df.show();

pdf=df.toPandas();

res = pdf.dropna(axis=0);

res['current\_loan\_amount']=res['current\_loan\_amount'].astype('str')

records=res.to\_dict(orient='records')

import pymongo

from pymongo import MongoClient

client = MongoClient("mongodb://127.0.0.1:27017/",27017)

dbs = client.list\_database\_names()

db = client["bank\_loan"]

coll\_short\_term = db.short\_term

db.coll\_short\_term.insert\_many(records)

cursor = db.coll\_short\_term.find()

for rec in cursor:

print(rec)

**4. Create a dataframe of Customer IDs who have taken the Long Term Loan and include their total Current Loan Amount and save the result in MongoDB?**

spark.sql("use bank\_loan");

df=spark.sql("select customer\_id,current\_loan\_amount,term from loan\_data where term='Long Term'");

df.show();

pdf=df.toPandas();

res = pdf.dropna(axis=0);

res['current\_loan\_amount']=res['current\_loan\_amount'].astype('str')

records=res.to\_dict(orient='records')

import pymongo

from pymongo import MongoClient

client = MongoClient("mongodb://127.0.0.1:27017/",27017)

dbs = client.list\_database\_names()

db = client["bank\_loan"]

coll\_long\_term = db.long\_term

db.coll\_long\_term.insert\_many(records)

cursor = db.coll\_long\_term.find()

for rec in cursor:

print(rec)

**5. Count how many Bankruptcies are present?**

select count(distinct customer\_id) bankruptcies from loan\_data where bankruptcies>0;

spark.sql("use bank\_loan");

df=spark.sql("select count(distinct customer\_id) bankruptcies from loan\_data where bankruptcies>0");

df.show();

**6. Group the data based on Term and find the average monthly debt.**

select term,avg(monthly\_debt)as avg from loan\_data group by term;

spark.sql("use bank\_loan");

df=spark.sql("select term,avg(monthly\_debt)as avg from loan\_data group by term");

df.show();

import pandas as pd

pdf=df.toPandas()

res = pdf.dropna(axis=0)

res

**7. Create a dataframe of the customers who have 10 + years experience in their current job. Include their Annual Income.**

There are following numbers of categories for experience:

10+ years

3 years

5 years

9 years

< 1 year

n/a

1 year

2 years

4 years

6 years

7 years

8 years

select customer\_id,years\_current\_job,annual\_income from loan\_data;

spark.sql("use bank\_loan");

df=spark.sql("select customer\_id,years\_current\_job,annual\_income from loan\_data");

df.show();

import pandas

pdf = df.toPandas()

final\_pdf = pdf.dropna(axis=0,subset=['years\_current\_job'])

final\_pdf.drop(final\_pdf[(final\_pdf['customer\_id']=='Customer ID')].index , inplace=True)

final\_pdf['years\_current\_job'].replace('10+ years','10+',inplace=True)

res = final\_pdf[final\_pdf['years\_current\_job']=='10+']

**8. Group the data based on Home ownership and Term. Find the aggregated sum of the total current loan.**

select sum(current\_loan\_amount),term,home\_ownership from loan\_data group by term,home\_ownership;

spark.sql("use bank\_loan");

df=spark.sql("select sum(current\_loan\_amount) as sum,term,home\_ownership from loan\_data group by term,home\_ownership");

df.show();

import pandas

pdf = df.toPandas()

final\_pdf = pdf.dropna(axis=0,subset=['sum'])

final\_pdf

**9. Find the highest credit score for short term and long term customers.**

select max(credit\_score), term from loan\_data group by term;

spark.sql("use bank\_loan");

df=spark.sql(" select max(credit\_score) as max, term from loan\_data group by term");

df.show();

import pandas

pdf = df.toPandas()

final\_pdf = pdf.dropna(axis=0,subset=['max'])

final\_pdf

**10. Group the data based on years in current job and Home ownership and find the aggregated sum of credit score.**

select sum(credit\_score) as sum, years\_current\_job, home\_ownership from loan\_data group by years\_current\_job,home\_ownership;

spark.sql("use bank\_loan");

df=spark.sql("select sum(credit\_score) as sum, years\_current\_job, home\_ownership from loan\_data group by years\_current\_job,home\_ownership");

df.show();

import pandas

pdf = df.toPandas()

final\_pdf = pdf.dropna(axis=0,subset=['sum'])

final\_pdf