IPYTHON (JUPYTER) NOTEBOOK WITH SPARK ON AWS EC2 INSTANCE

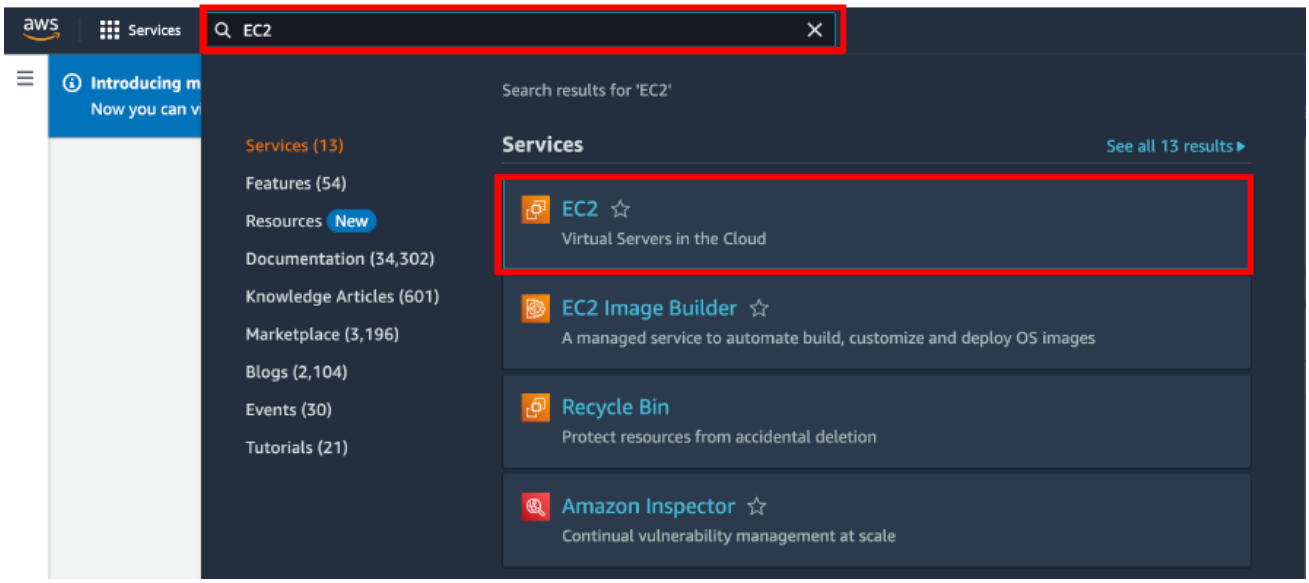
1. CREATE AWS EDUCATE ACCOUNT

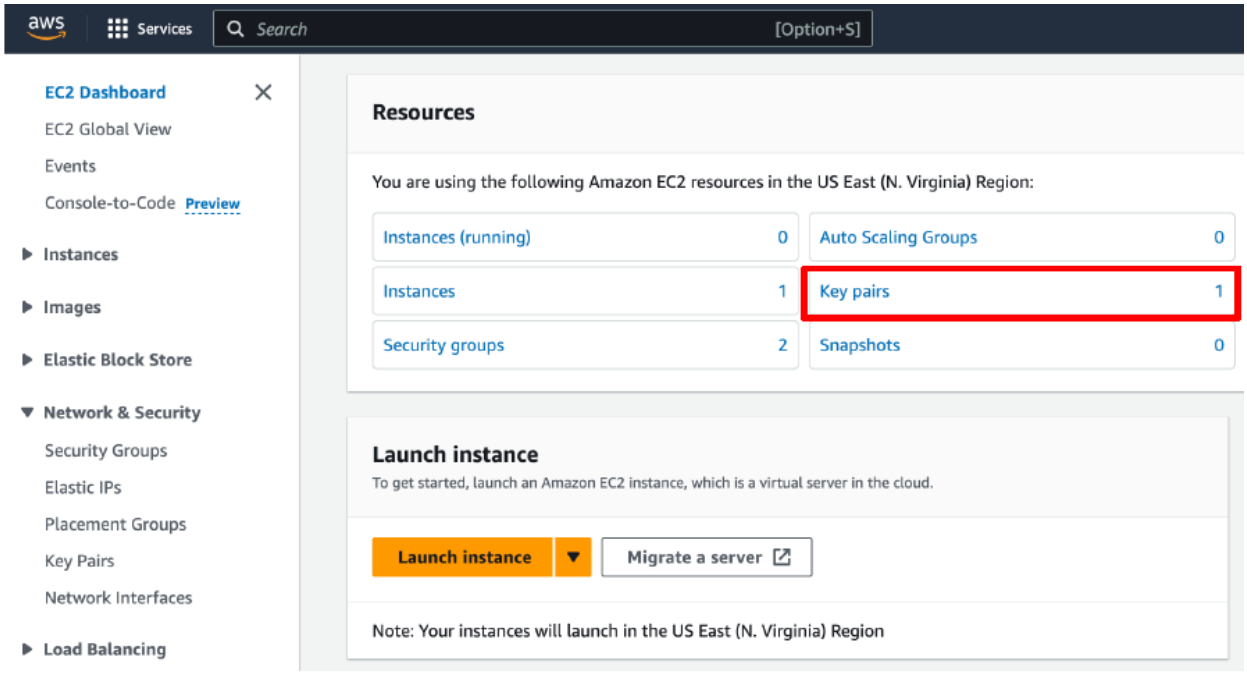
You can create a normal AWS account (Credit card required) with your **UF email.**

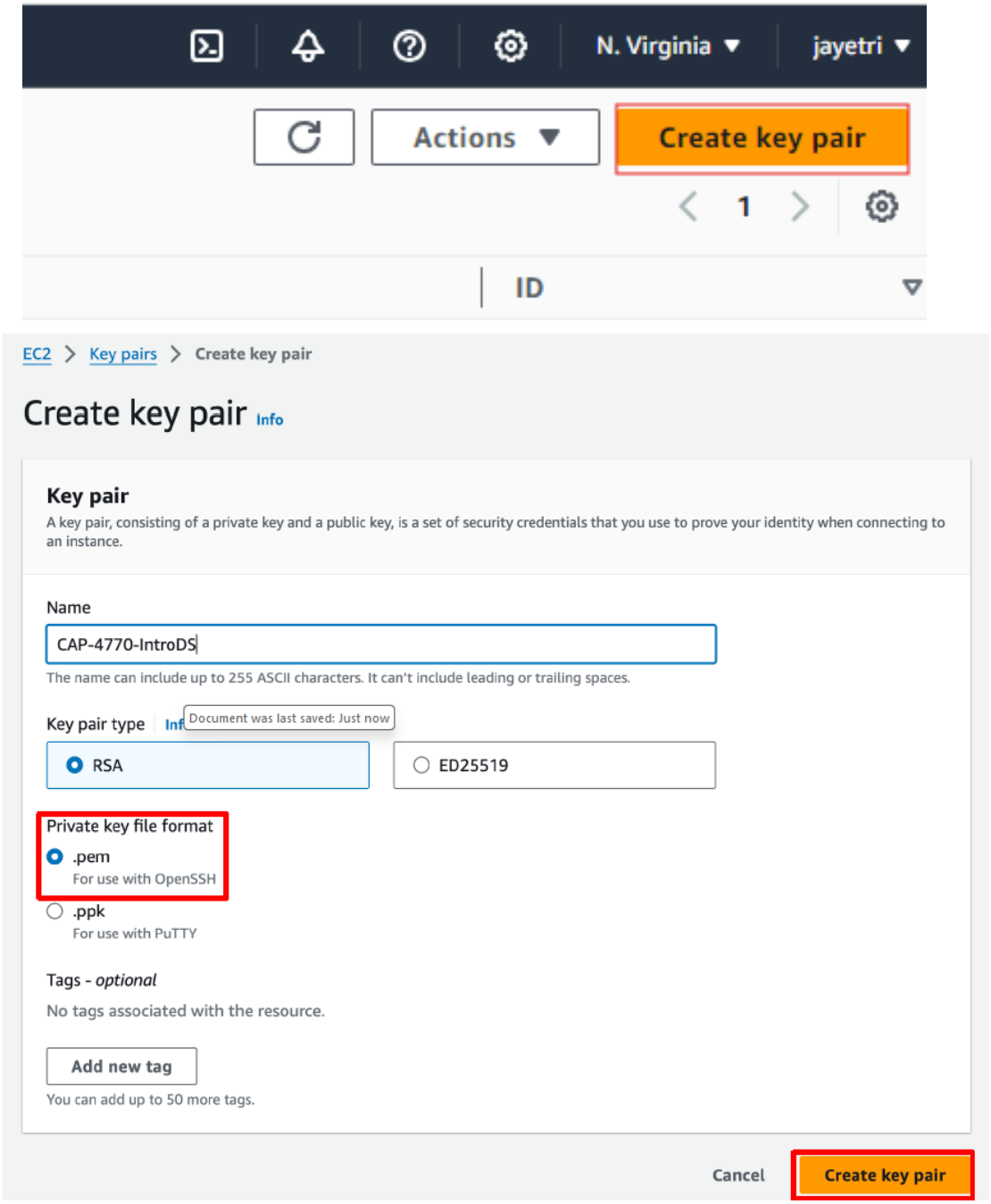
2. LAUNCH AND CONNECT TO AN EC2 INSTANCE

1. Sign into your AWS account console: https://aws.amazon.com/

   Make sure you are in Region US East (N. Virginia) in the top right corner.
2. Get key pair:
   On your console, click 'Services' -> EC2 -> 0 Key Pairs -> Create Key Pair



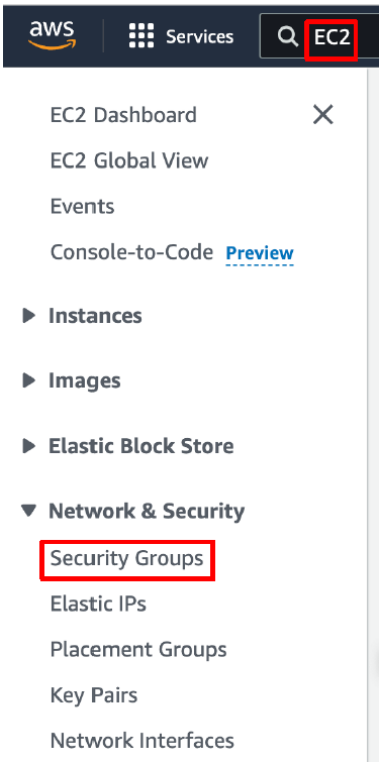




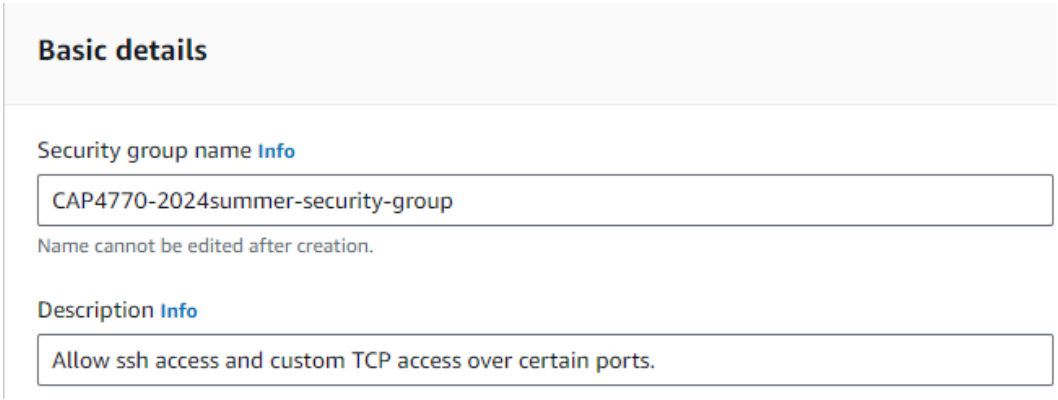
Download the 'pem' file in a safe space for later use.

Put the 'pem' file in your local lab folder, and open terminal in the local folder,
$ cd < path to the local lab folder >
type:
$ chmod 400 ~/your_key.pem
for ssh client to read

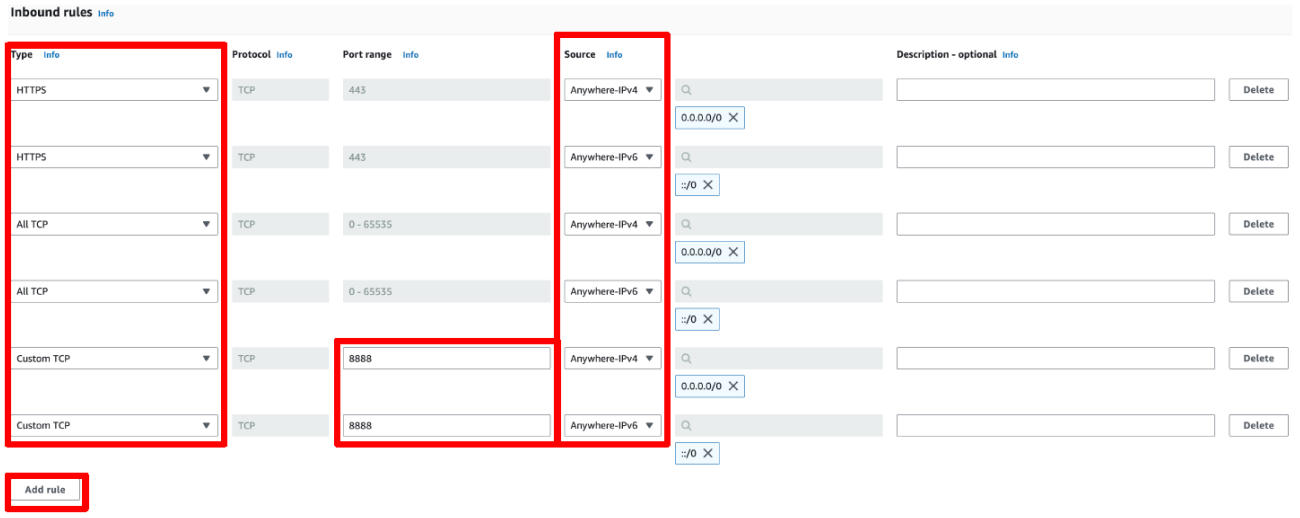
3. Create security group by clicking ‘Services’ -> EC2 -> Security Groups -> Create Security Group
   Make sure you are in Region US East (N. Virginia) in the top right corner.

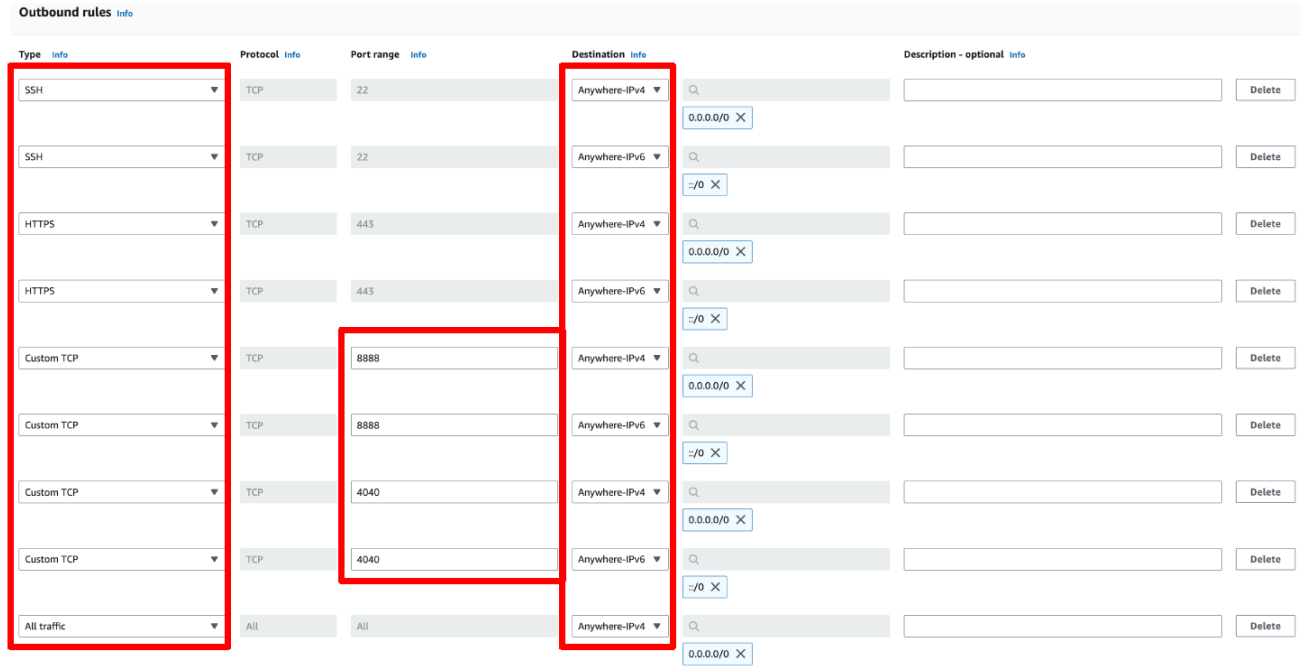


Give the name and description of the security group.

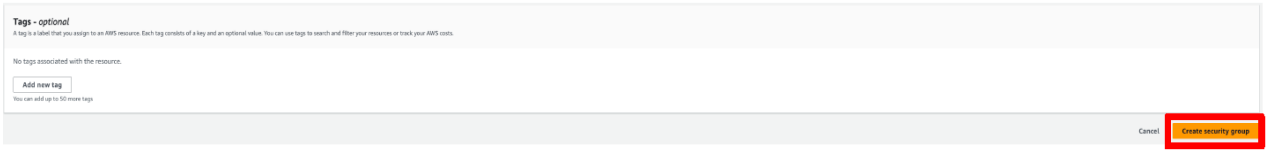


Then add the following inbound and outbound rules.

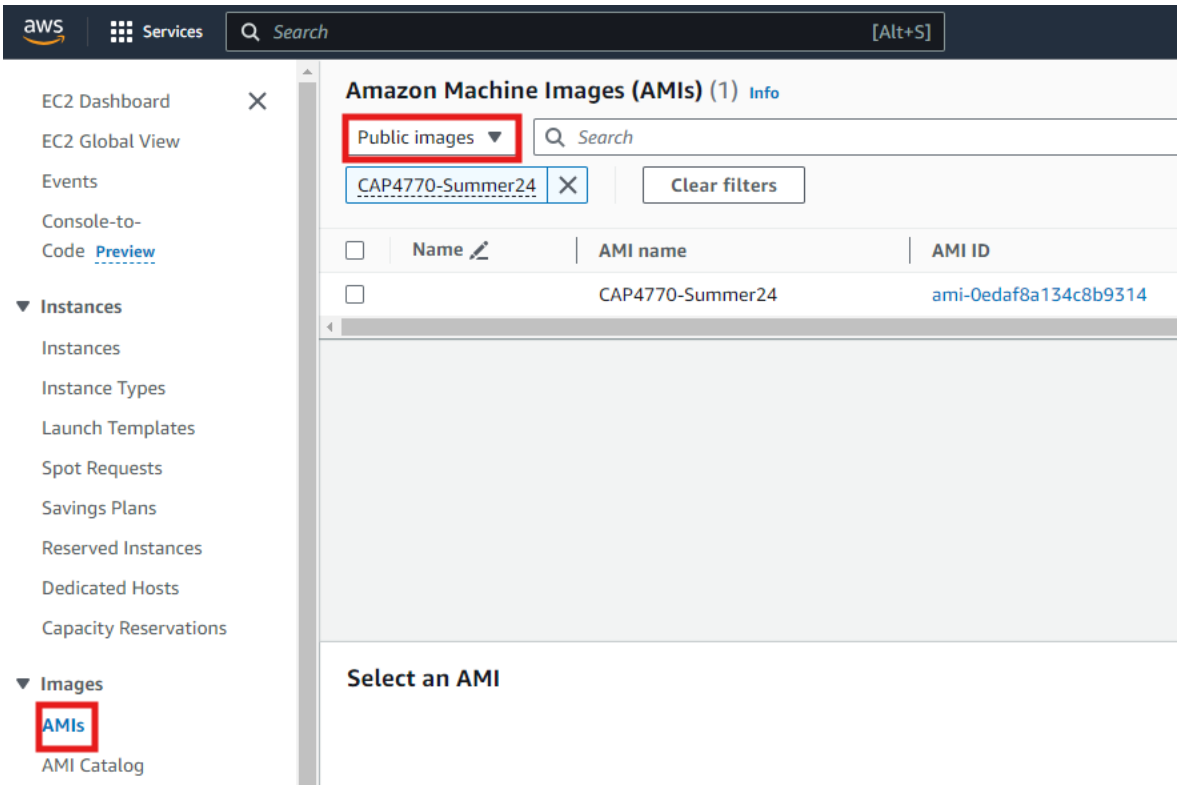




Then, create the security group.



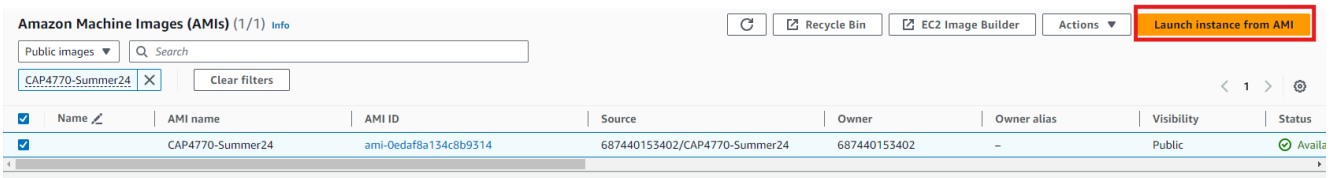
4. Start EC2 instance from preconfigured AMI click ‘Services’ -> EC2 -> AMIs



Make sure under the launch button, it says public images, and you are in Region US East (N. Virginia) in the top right corner.

In the search bar, search for 'CAP4770-Summer24'

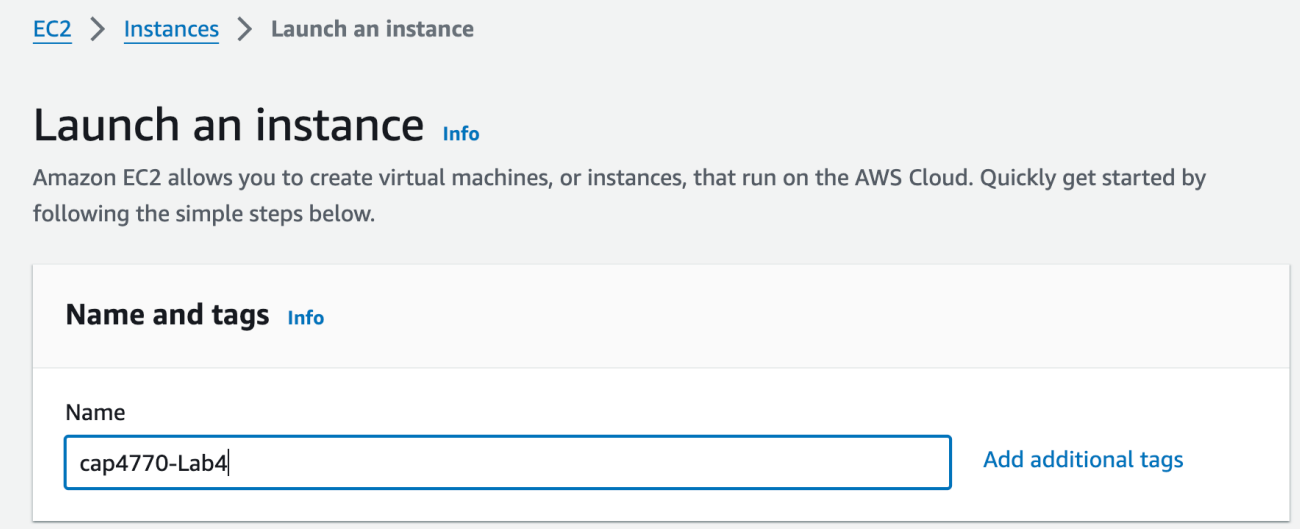
An AMI image with the name and ID 'ami-0edaf8a134c8b9314' should show up.



Select the image, and click 'Launch instance from AMI' button as shown above, and follow the steps below:

Step 1: Choose AMI: Already chosen.

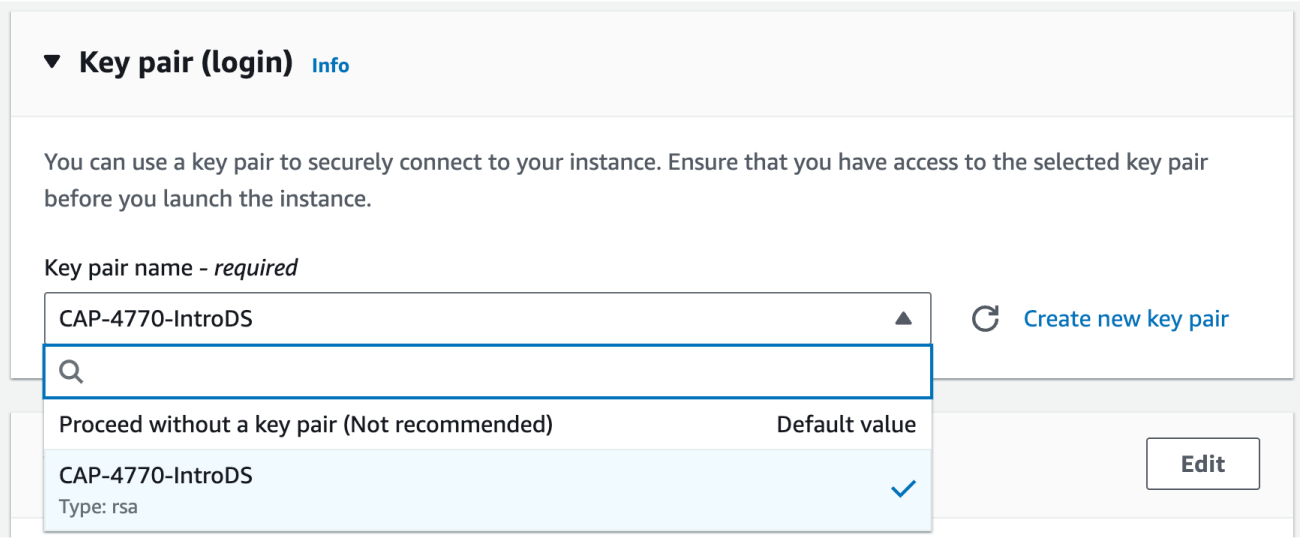
Step 2: Give your instance a name.



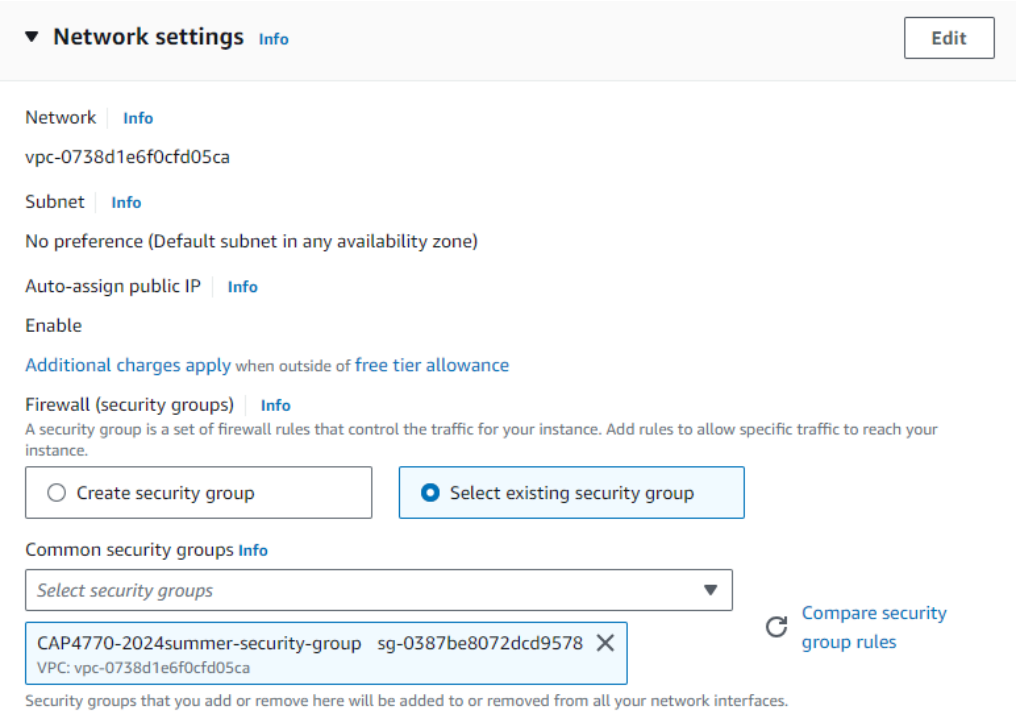
Step 3: Choose Instance Type: t2.micro (free tier), **you can update this to large instance later if you want to process larger dataset.**
Note: Sometimes the micro instance is not sufficient to run pyspark. Incase, you experience errors and timeouts while executing the flatmap function or while saving the file to S3, try with t2.medium. Make sure to terminate the instance after capturing the required screenshots to avoid being charged when the instance is not used.

For this lab, t2.medium is recommended over t2.micro to avoid timeouts.

Step 4: Select the key pair created from the dropdown menu.



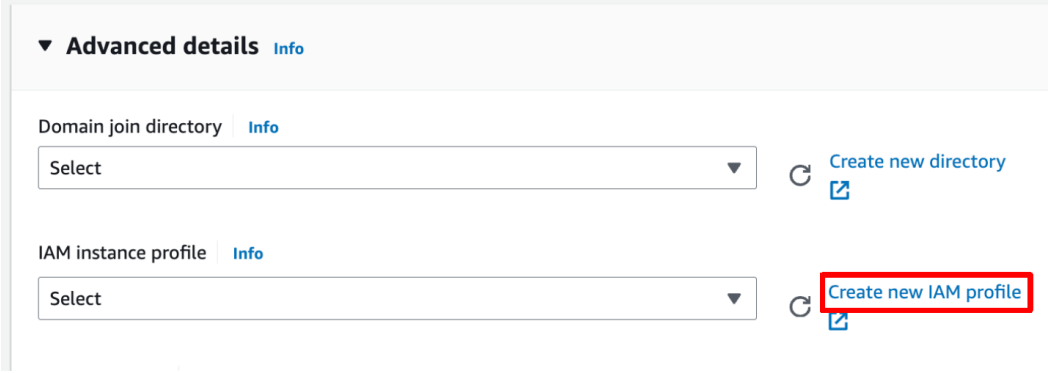
Step 5: Network settings: Select existing group and choose the security group created earlier.



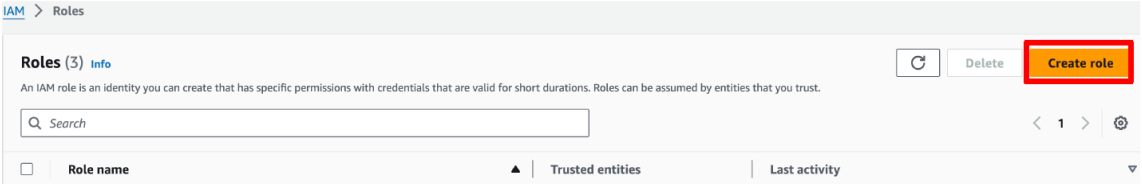
Step 6: Config Storage: default, skip

Step 7: Configure Instance (In Advanced details):
In AWS setup, create an IAM role manually, following these steps: (If you leave the "IAM role" as "None", then you will see an error "Unable to load AWS credentials from any provider in the chain" when trying to access S3 in ipython notebook)

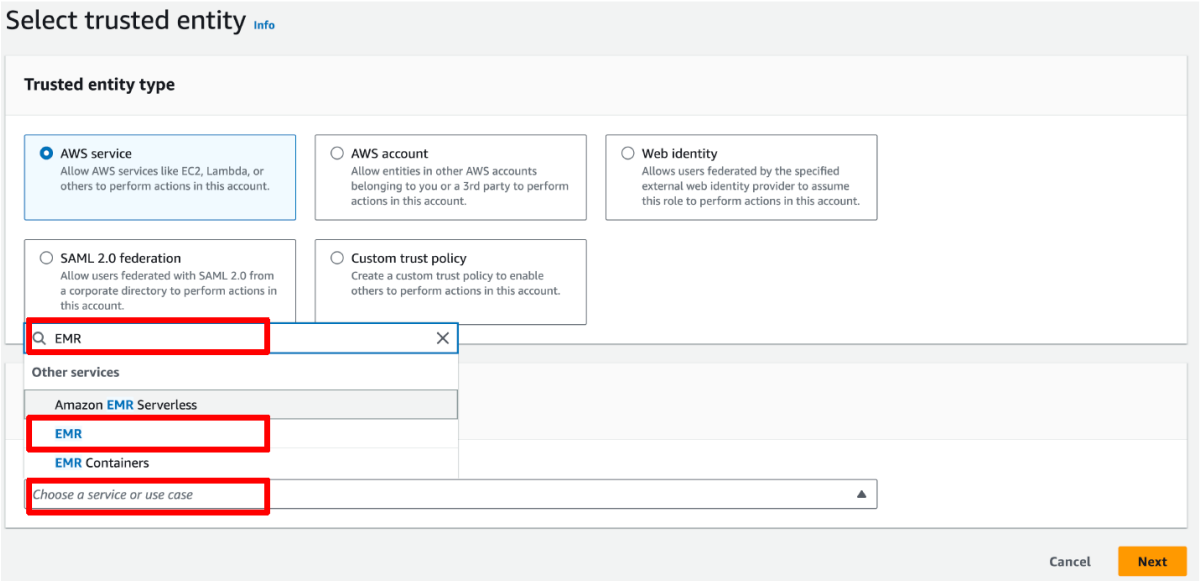
1) Click "Create new IAM role."



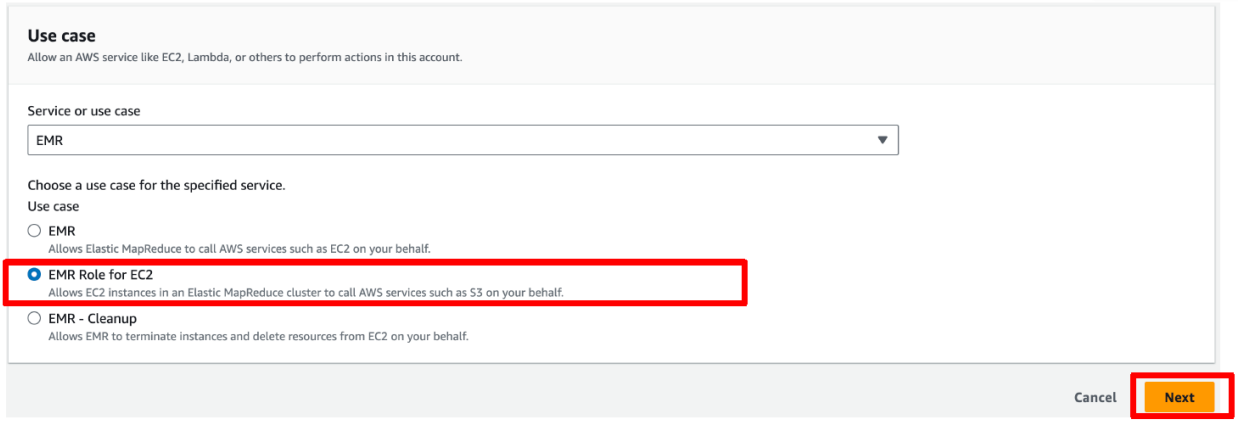
2) Click "Create role"



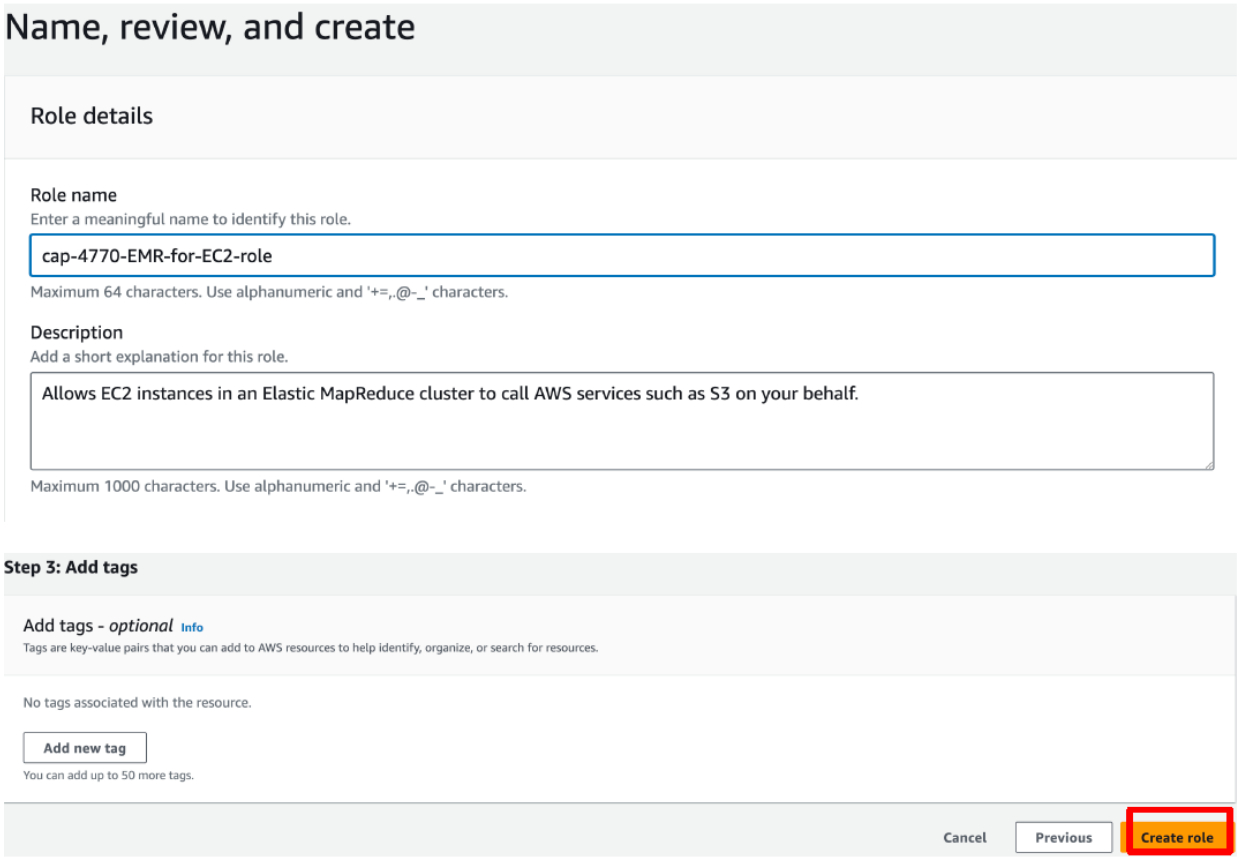
3) Search and Click "EMR" in “Use case” section:



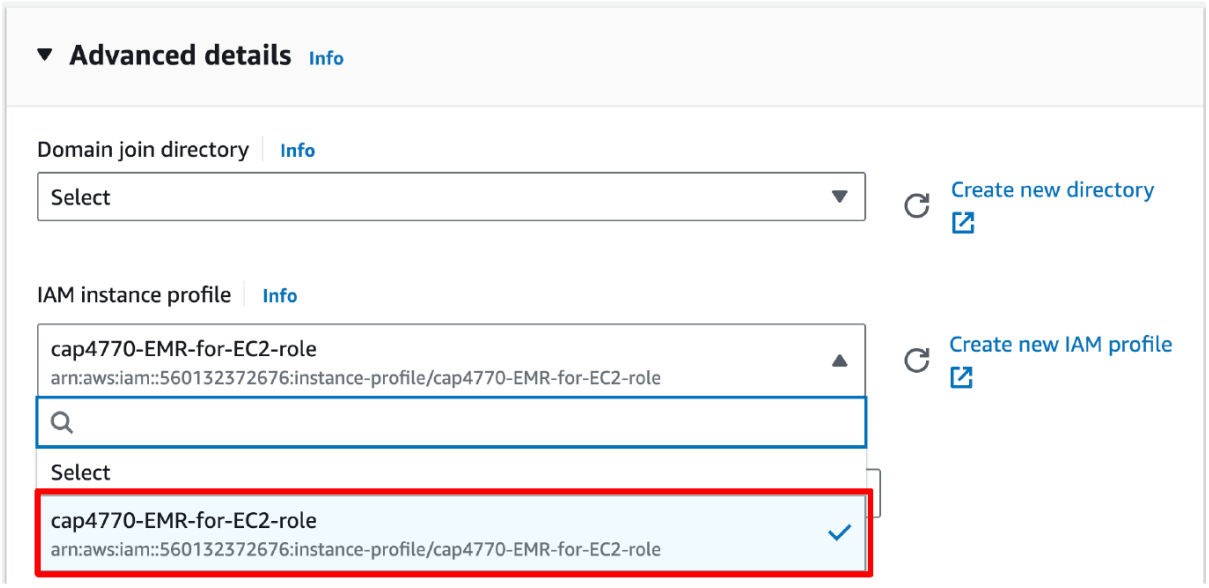
4) Click "EMR Role for EC2" then click "Next: ..." buttons (in lower-right corner) until you see "Create role" button (also in lower-right corner).



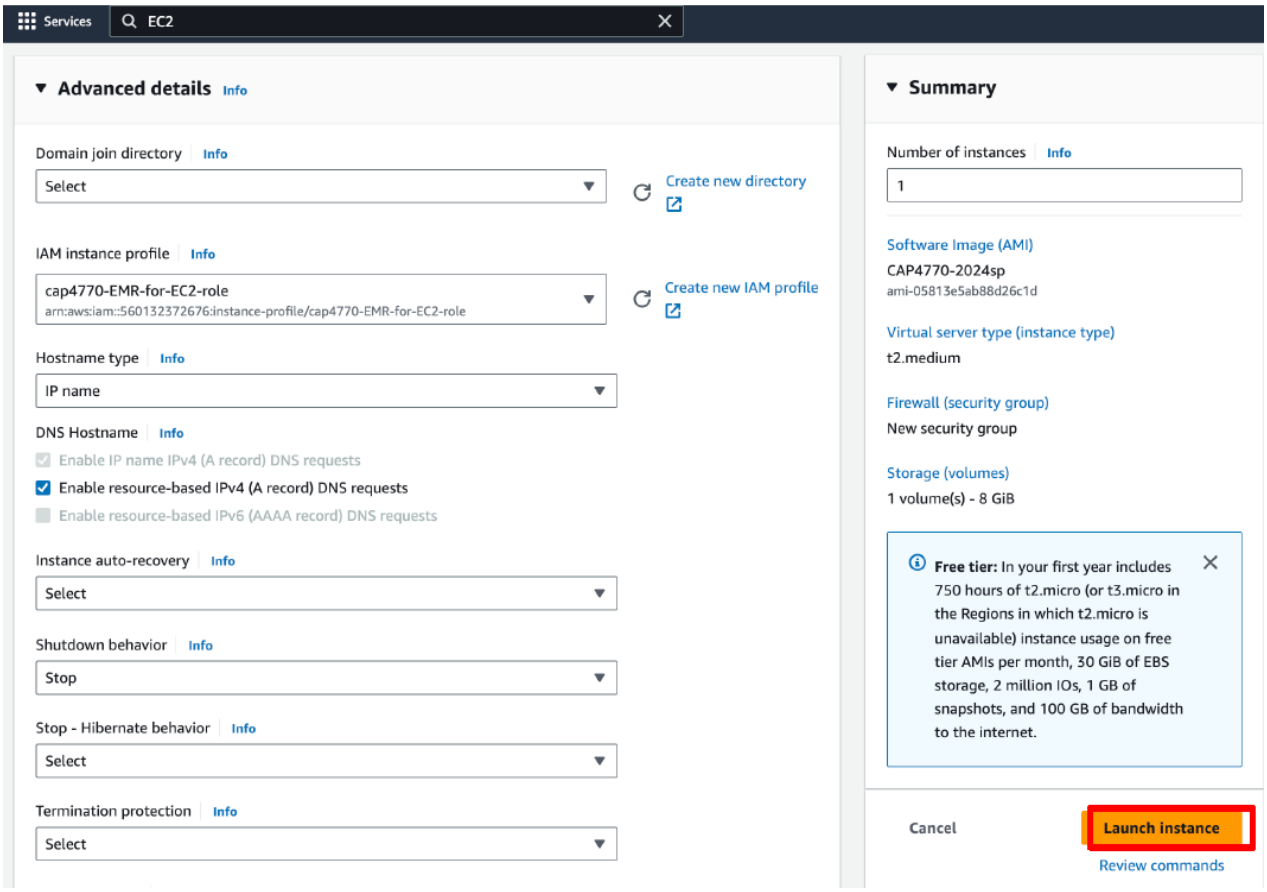
5) Give it a name and click "Create role".



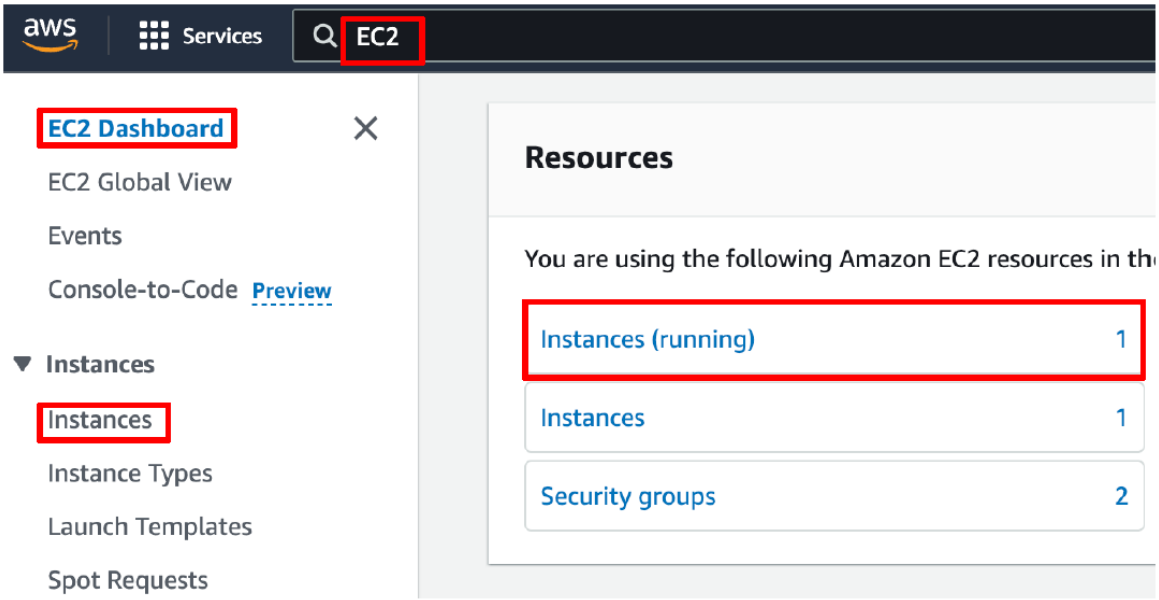
Then you will see this role in the "IAM role" section when setting EC2 instance. Just select it like mentioned above.



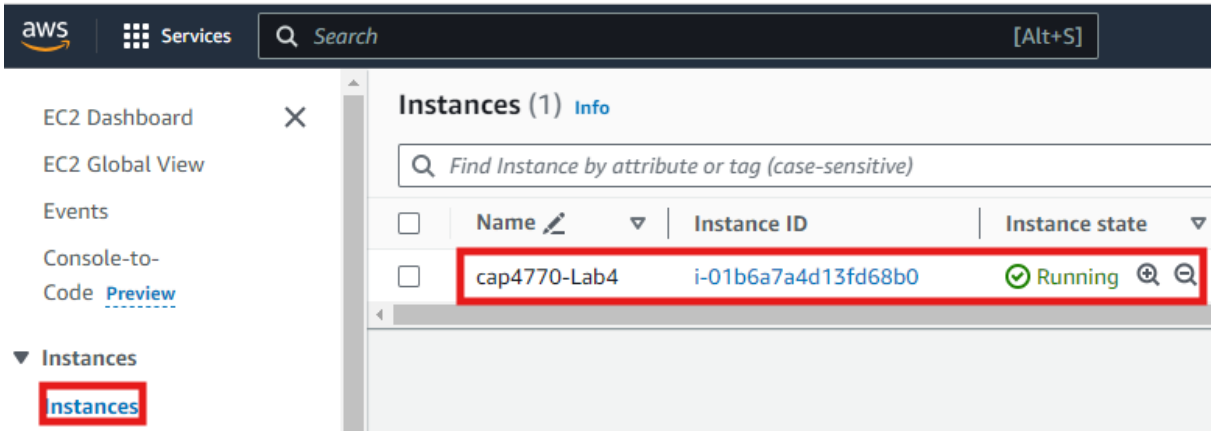
Step8: Review and Click “Launch instance”



Now click on Services -> EC2 -> Instances -> Instances (or you can again click Services -> EC2 -> Instances (running)). You will be able to view the instance created.

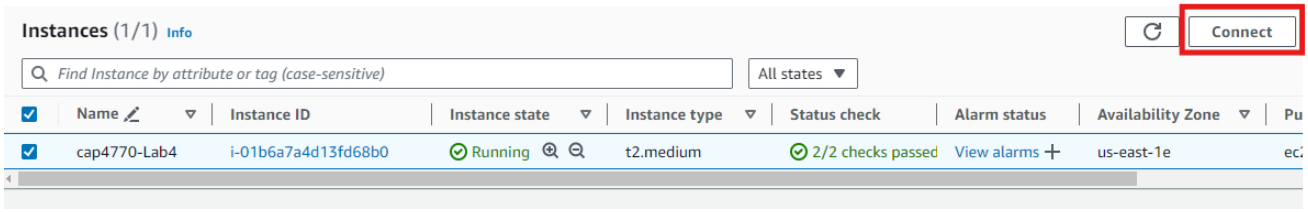


And you'll see the instance we just created.

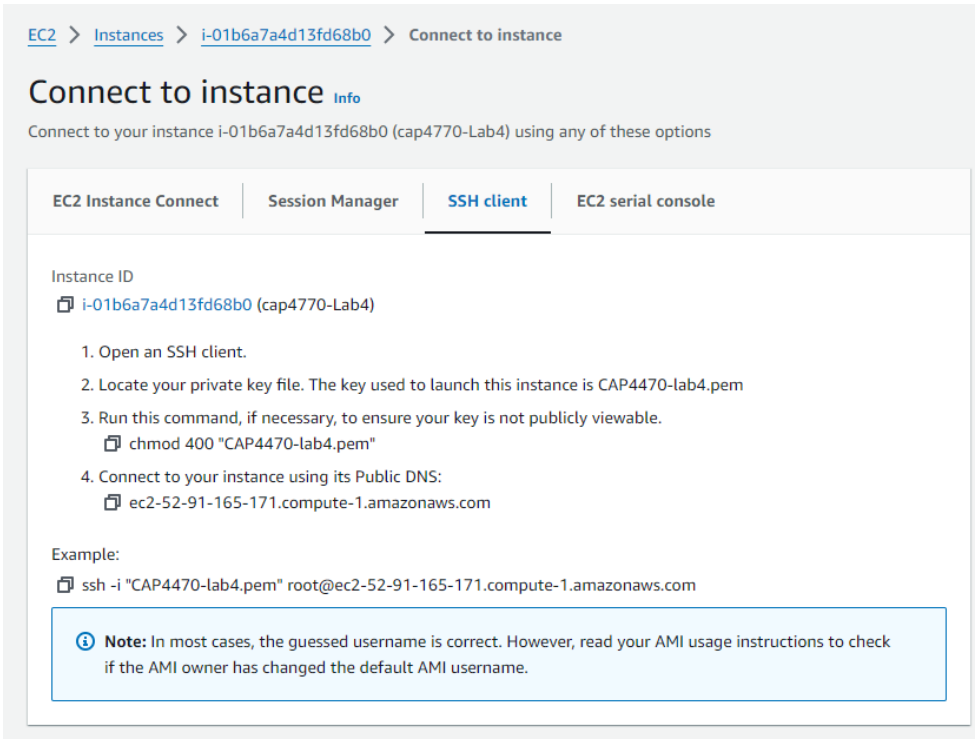


5. Connect to the instance.

Select the running instance and click 'connect'.



Go to the SSH client tab, you'll see a screen like this:



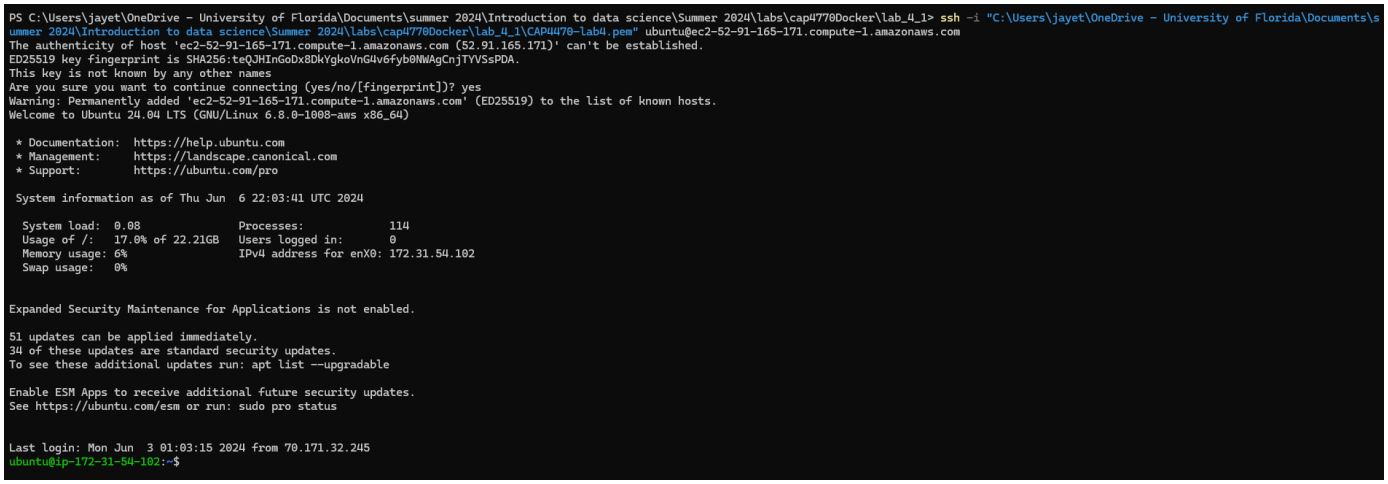
Switch to your local machine, in the terminal, copy and paste the example command (note: update with your actual DNS, and make sure to log in as user 'ubuntu'):

```
$ ssh -i "<path to the .pem file that you downloaded before>" ubuntu@ec2-3-85-136-34.compute-1.amazonaws.com

if denied because of bad key permission try:
$ chmod 400 "<path to the .pem file that you downloaded before>"

Or just use sudo command:

$ sudo ssh -i ""<path to the .pem file that you downloaded before>" ubuntu@ec2-3-85-136-34.compute-1.amazonaws.com
```



And you have logged into the instance in the cloud via ssh. (Note: your actual .pem file and public DNS will be different from the example)

Important Notes:

1. Don't forget to stop/terminate your EC2 Instance after using it, otherwise it will cost your extra **credits/money**!

   Also Please note that different regions are independent, so make sure you stop/terminate all running instances in different regions.

2. Your data on the EC2 instance will be lost if you terminate it. You can save your output results on S3, for Jupyter Notebooks you create, you can download them from the browser via File -> Download as Jupyter notebook.

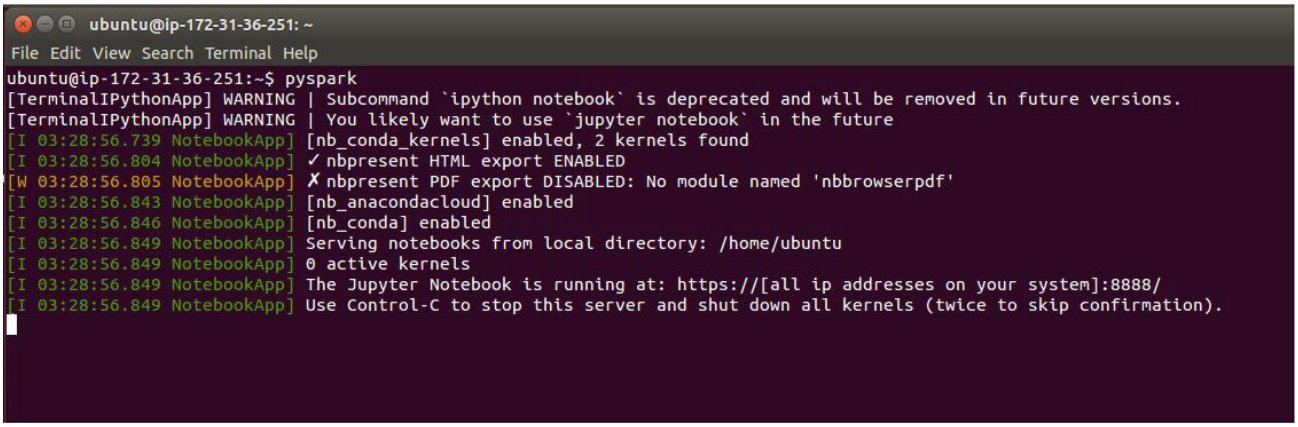
3. RUN SPARK WITH IPYTHON NOTEBOOK IN REMOTE INSTANCE

In the terminal where you ssh into the remote instance from last step above,

type

```
$ pyspark
```

you'll see following:

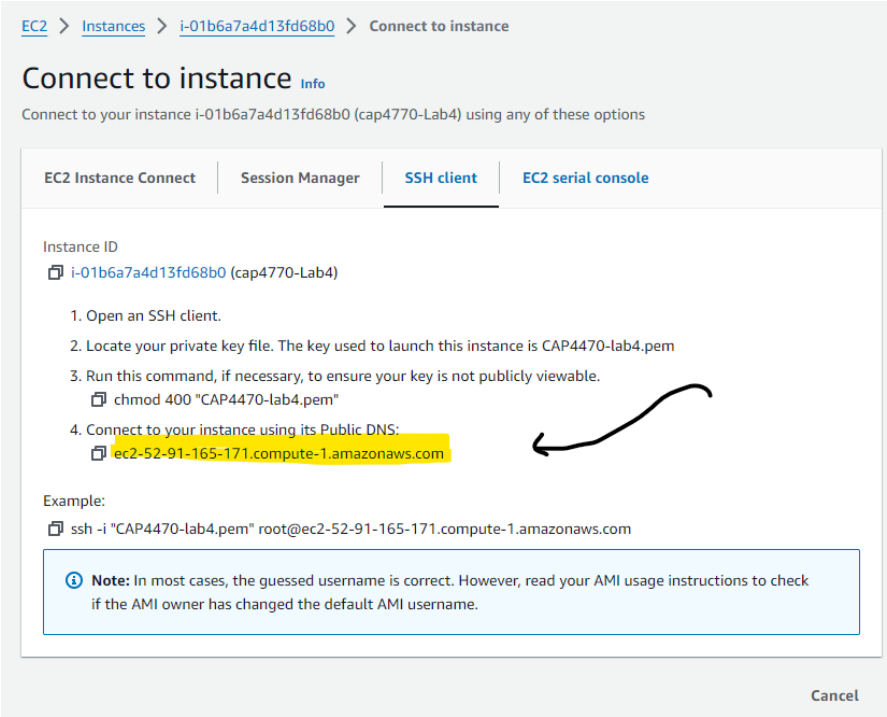


Now Open your browser on your laptop (can be the browser in your Host OS), and input the address in the URL (**Note it may start with "https"**):

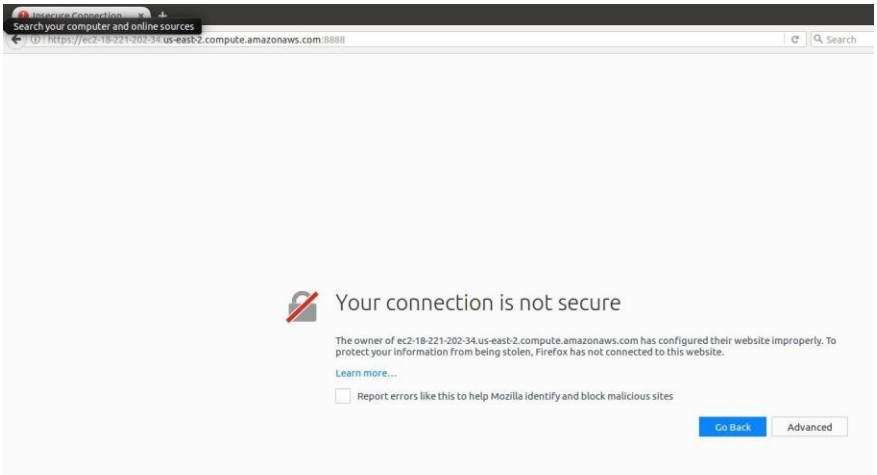
http://ec2-52-91-165-171.compute-1.amazonaws.com:8888/

Note your DNS should be different in your case, change accordingly.

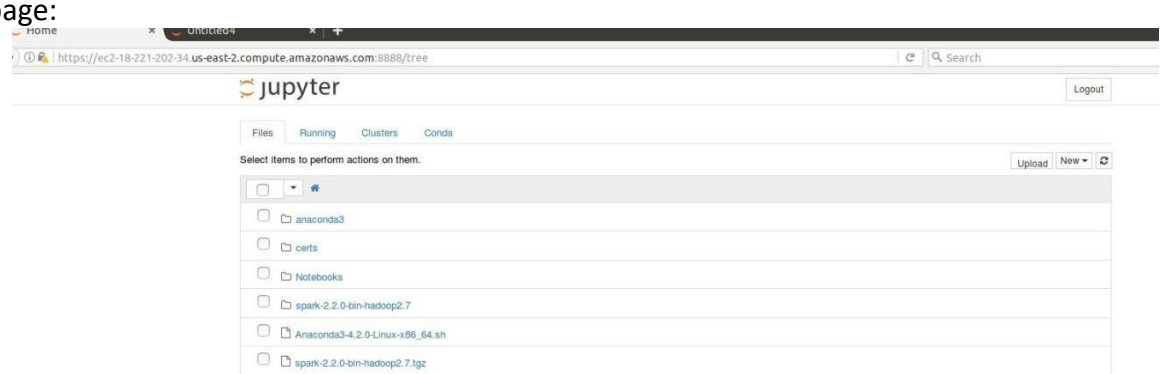
The public IPV4 DNS address can be found in the 'connect to instance screen'
(under SSH client):



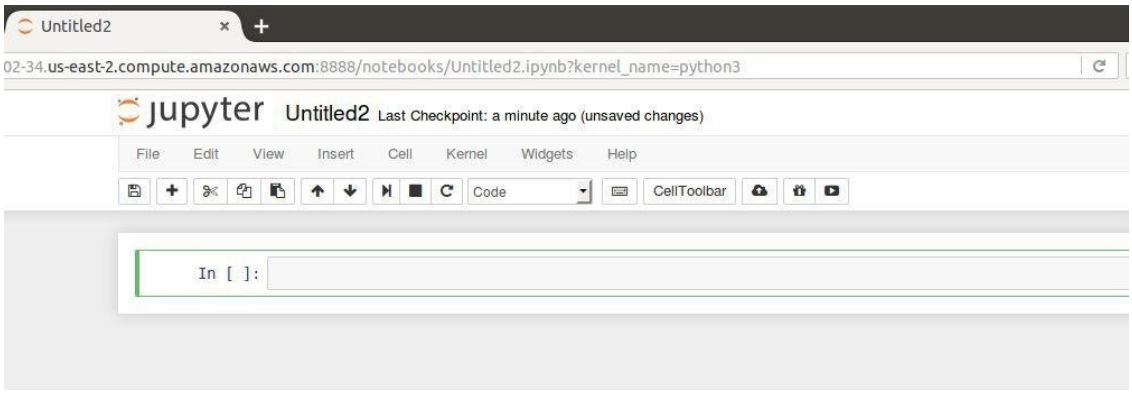
If you see a warning, click advanced and add Exception to continue.



And input your password (password is 123456 by default. You'll see a familiar Jupyter Notebook page:

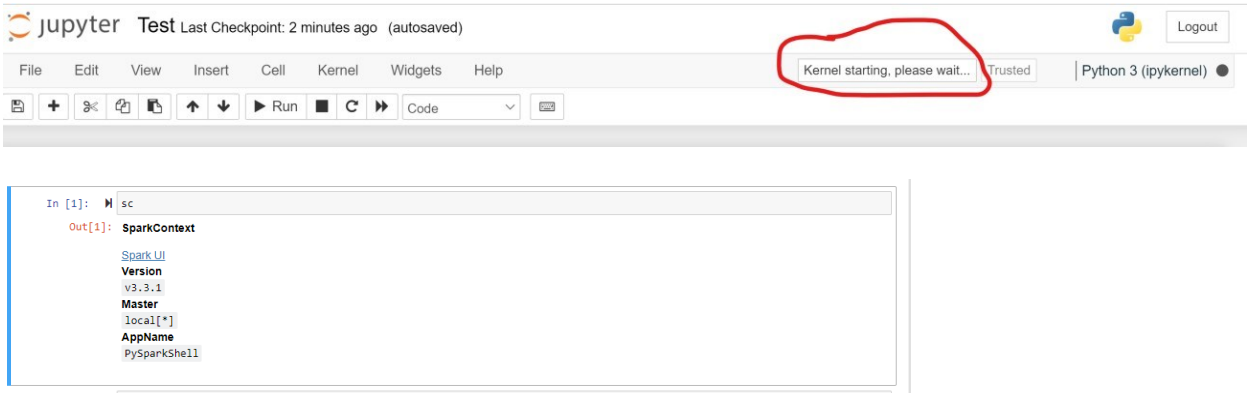


Create a test Ipython notebook by click New -> Python [default], you'll be prompted to a new page:



Type sc in the cell and you'll see (in a short while, ignore the warnings)

(Note: Always wait for the kernel to be ready, then run your code blocks)



Now you have a remote server running Spark with Ipython Notebook and you can start coding using PySparkShell interactively!

Optionally, you can monitor the status of the remote Spark via link below (again change to your DNS here too. **Note it must start with "http", not https**):

http://ec2-52-91-165-171.compute-1.amazonaws.com:4040/

4. CONFIGURE ACCESS WITH S3

The remote instance we create does not hold data indefinitely -- any data on the instance will be lost if we terminate the EC2 instance.

AWS offers S3 for permanent data storage. Here is how to use it:

Terminate the pyspark process above (Ctrl + c twice), and re-run with additional dependency to read/write to S3 as below:

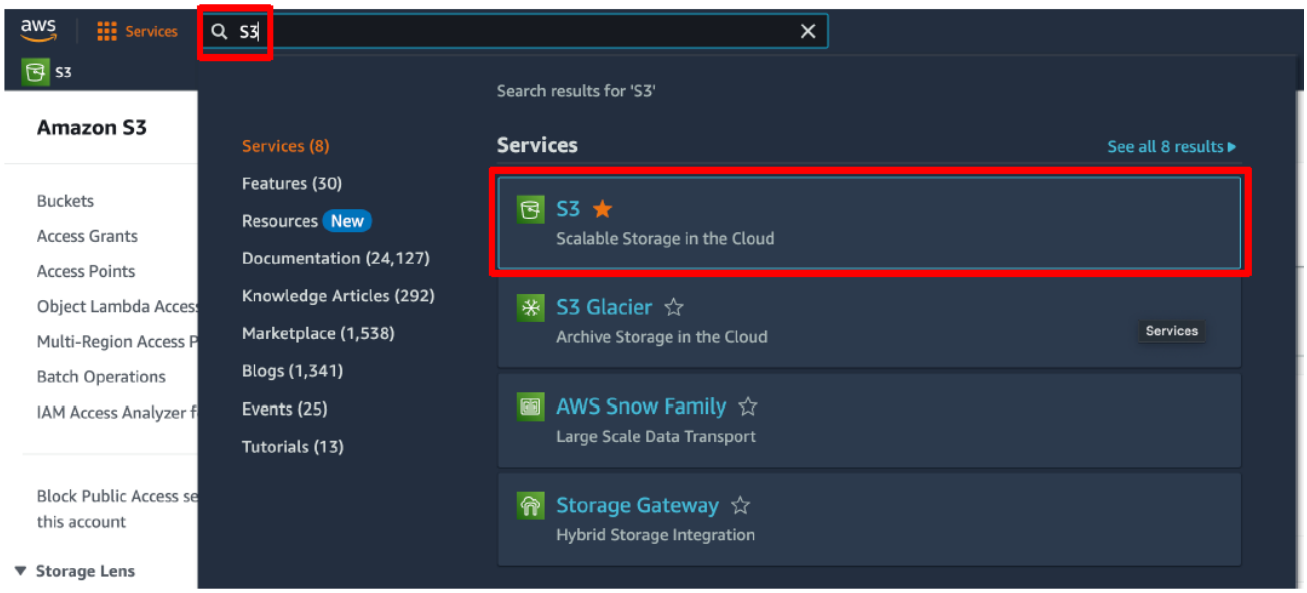
```
$ pyspark --packages  com.amazonaws:aws-java-sdk:1.12.403,org.apache.hadoop:hadoop-aws:3.3.1
```

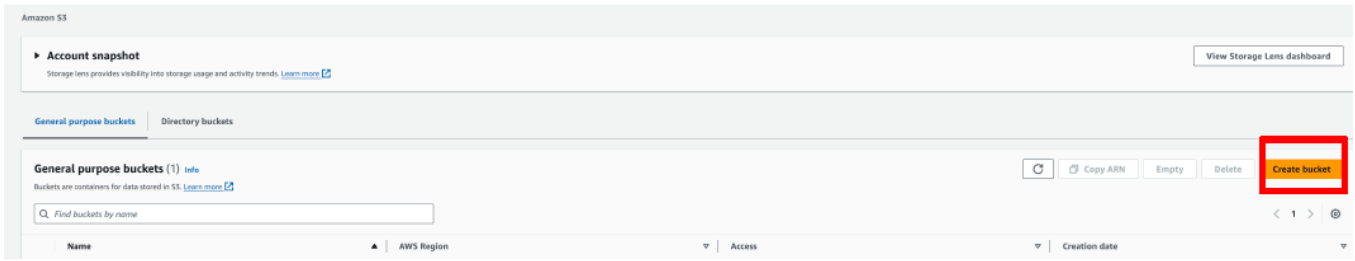
Reload your previous webpage to reconnect to this new Spark PySparkShell instance.

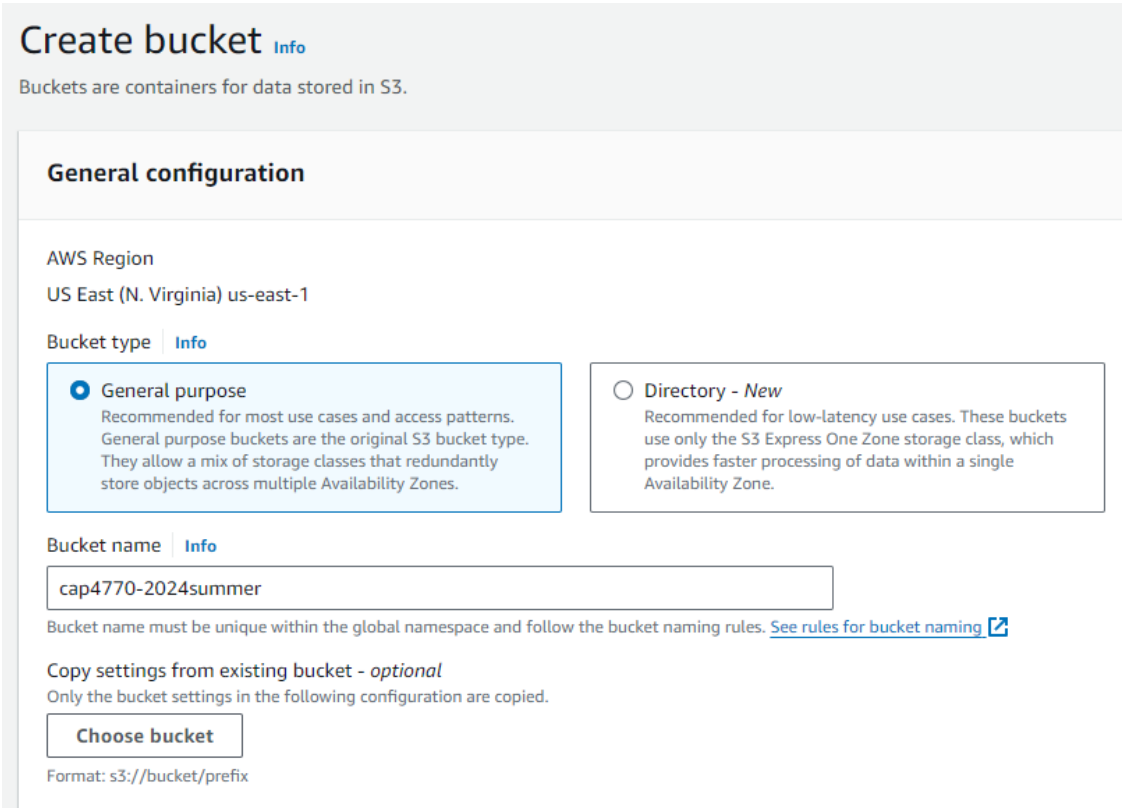
(Note: Always wait for the kernel to be ready, then run your code blocks)

You also need to create bucket(s) on S3 to store your data:
Go to Services -> Search S3 -> click on S3.

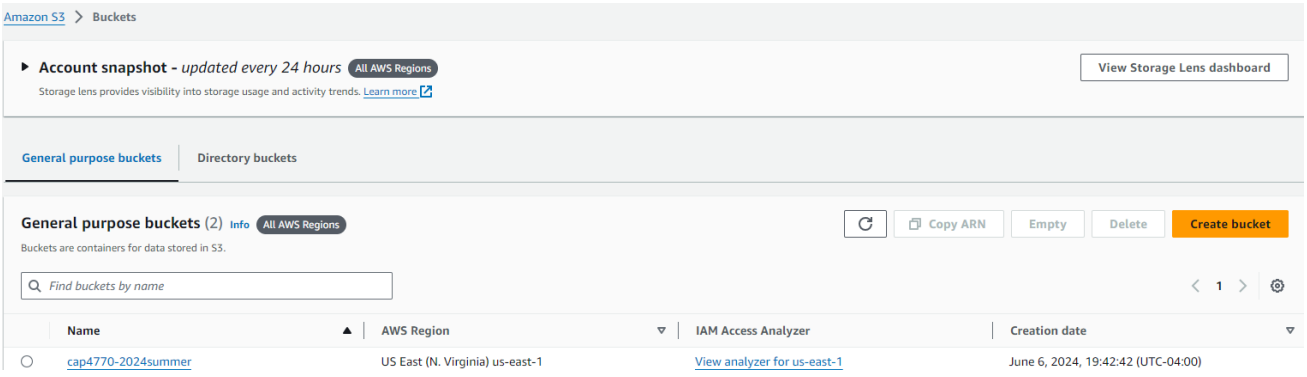
1. Upload file to S3
2. Go to Services -> Search S3 -> click on S3 -> Create Bucket





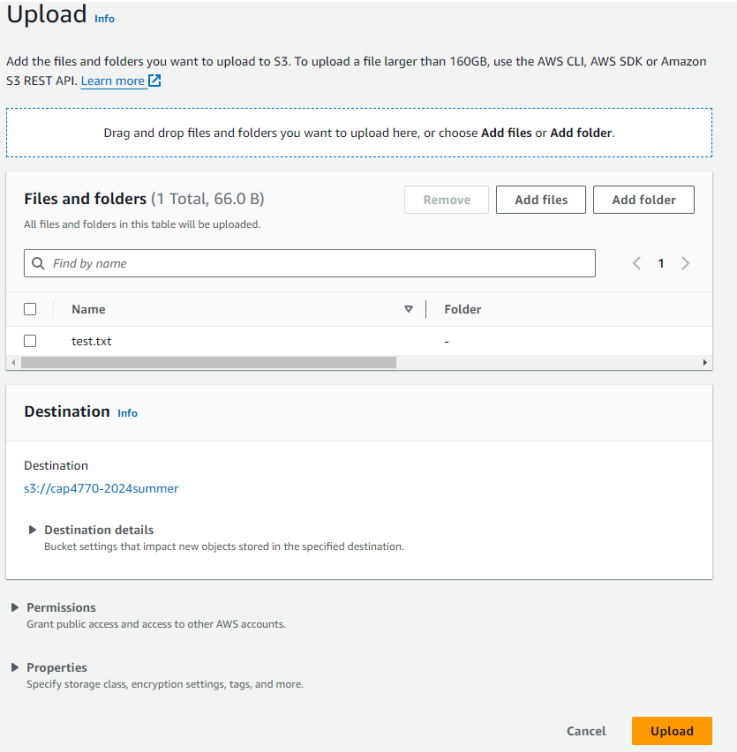


Create a bucket name, e.g., 'cap4770-2024summer'. Click next and use default settings. After creation, a new bucket will show up:



Click on the bucket and you can upload files (by clicking on add files)

Create a local test file, say 'test.txt' containing some random sentences, and upload it to the bucket we just created like below.



Now in the jupyter notebook file, type something like this to read it (note: change to your bucket name and file name):

In [3]: sc.textFile('s3a://<name-of-your-bucket>/test.txt).collect()

```
[5]: sc
```

```
[5]: SparkContext

Spark UI

Version    v3.5.1
Master     local[*]
AppName    PySparkShell
```

```
[6]: sc.textFile('s3a://cap4770-2024summer/test.txt').collect()
```

```
[6]: ['Data science is an interdisciplinary field that uses scientific methods, algorithms, and systems to extract insights from data. It combines domains like
data science, statistics, and information science to analyze and interpret complex data.']
```

Now, try executing the following code on Jupyter Notebook:

```
[9]: rdd = sc.textFile('s3a://cap4770-2024summer/test.txt')
     rdd1 = rdd.flatMap(lambda line: line.split(' '))

[10]: rdd2 = rdd1.map(lambda word: (word, 1))

[11]: rdd2

[11]: PythonRDD[10] at RDD at PythonRDD.scala:53

[12]: rdd3 = rdd2.reduceByKey(lambda x, y: x + y)

[13]: rdd3.collect()

[13]: [('science', 2),
       ('an', 1),
       ('field', 1),
       ('uses', 1),
       ('and', 3),
       ('to', 2),
       ('insights', 1),
       ('from', 1),
       ('data.', 2),
       ('It', 1),
       ('like', 1),
       ('science,', 1),
       ('statistics,', 1),
       ('information', 1),
       ('analyze', 1),
       ('Data', 1),
       ('is', 1),
       ('interdisciplinary', 1),
       ('that', 1),
       ('scientific', 1),
       ('methods,', 1),
       ('algorithms,', 1),
       ('systems', 1),
       ('extract', 1),
       ('combines', 1),
       ('domains', 1),
       ('data', 1),
       ('interpret', 1),
       ('complex', 1)]
```

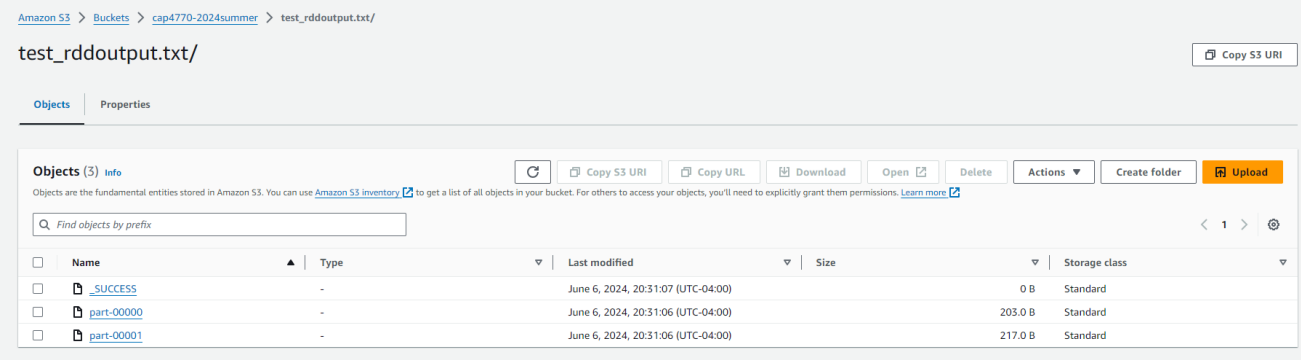
Or more concisely:

```
[16]: rdd = sc.textFile('s3a://cap4770-2024summer/test.txt')
      rdd.flatMap(lambda line: line.split(' ')) \
          .map(lambda word: (word, 1)) \
          .reduceByKey(lambda x, y: x + y) \
          .collect()

[16]: [('science', 2),
       ('an', 1),
       ('field', 1),
       ('uses', 1),
       ('and', 3),
       ('to', 2),
       ('insights', 1),
       ('from', 1),
       ('data.', 2),
       ('It', 1),
       ('like', 1),
       ('science,', 1),
       ('statistics,', 1),
       ('information', 1),
       ('analyze', 1),
       ('Data', 1),
       ('is', 1),
       ('interdisciplinary', 1),
       ('that', 1),
       ('scientific', 1),
       ('methods,', 1),
       ('algorithms,', 1),
       ('systems', 1),
       ('extract', 1),
       ('combines', 1),
       ('domains', 1),
       ('data', 1),
       ('interpret', 1),
       ('complex', 1)]
```

Now that we have get the word counts, we can save them back to S3 for permanent storage, by replacing '.collect()' with '.saveAsTextFile('s3a://<name-of-your-bucket>/test_rddoutput.txt')' above.

Now we can see a folder on S3 containing the results



Please go over the programming guide https://spark.apache.org/docs/latest/rdd-programming-guide.html for more operations and detailed explanations.
You can skip the "initializing spark shell" and "linking with spark" parts in the guide.

Experiment with your test.txt and see how it works.