

1. In the AWS Academy Learned Lab click on Modules

ALLv2EN-US-LTI13-125325

[Home](#)

[Modules](#)

[Discussions](#)

[Grades](#)

[Lucid \(Whiteboard\)](#)

## AWS Academy Learner Lab [125325]



AWS Academy Learner Lab provides a long-running sandbox environment for ad hoc exploration of AWS services. Within this class, students will

[View Course](#)

[View Course](#)

[View Course](#)

**To Do**

Nothing for now

**Recent Feedback**

Nothing for now

2. Scroll down and click on Launch AWS Academy Learner Lab

[Discussions](#)

[Grades](#)

[Lucid \(Whiteboard\)](#)

[!\[\]\(f58128c41dc307543fa2591fa073e87a\_img.jpg\) Learn how to effectively use the AWS Academy Learner Lab](#)

[!\[\]\(f5126919fb264baa65afc980ba29ad65\_img.jpg\) Module Knowledge Check](#)  
100 pts Score at least 70.0

[!\[\]\(1a455106cb811baa352b4f5964fd6a2f\_img.jpg\) ▾ AWS Academy Learner Lab](#)

[!\[\]\(053ecc0f11ce3eaaf59579a64bc6e912\_img.jpg\) Launch AWS Academy Learner Lab](#)

[!\[\]\(fb8baa27ba84e6b5f4f4533f09b4c70a\_img.jpg\) ▾ AWS Academy Learner Lab Resources](#)

3. Click on Start Lab and wait until the circle next to AWS turns green

ALLv2EN-... > Modules > AWS Acad...

> Launch AWS Academy Learner Lab

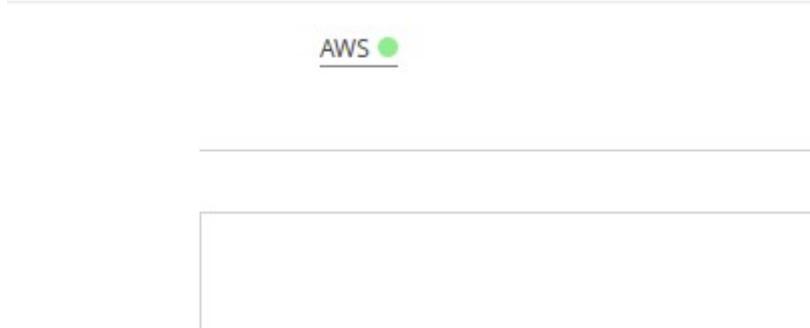
AWS ● 04:00 Start Lab End Lab AWS Details Readme  
Used \$0 of \$50

Home  
Modules EN-US  
Discussions  
Grades  
Lucid (Whiteboard)



[Environment Overview](#)  
[Environment Navigation](#)  
[Access the AWS Management Console](#)  
[Region restriction](#)  
[Service usage and restrictions](#)  
[Using the terminal](#)

4. Click on AWS after it turns green



5. It will open the AWS console in the browser. In the search bar search for ec2 and click the EC2 from result tab

sole

lecen

Services

Features

Resources New

Documentation

Knowledge articles

Marketplace

Blog posts

EC2 Virtual Servers in the Cloud

Top features

Dashboard Launch templates Instances Spot Instance requests Savings plans

EC2 Image Builder

In the EC2 page in the left hand side you will see a tab for Instances. Click on the Instances link below it

EC2

Dashboard

EC2 Global View

Events

Instances

Instances

Instance Types

Launch Templates

Spot Requests

Savings Plans

Reserved Instances

Dedicated Hosts

Capacity Reservations

You can change your default landing page for EC2.

Permanently dismiss Change landing page

Resources

EC2 Global View

Auto Scaling Groups

Dedicated Hosts

Instances

Load balancers

Security groups

Instances (running)	0	Auto Scaling Groups
Capacity Reservations	0	Dedicated Hosts
Elastic IPs	0	Instances
Key pairs	1	Load balancers
Placement groups	0	Security groups

6. In the Instances page. Currently you do not have any instances created. Click the Orange Launch Instances button

The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with 'EC2' selected. Under 'Instances', 'Launch instances' is highlighted. The main area has a search bar and filters for 'Name', 'Instance ID', 'Instance state', 'Instance type', 'Status check', and 'Alarm stat'. It displays a message: 'No instances' and 'You do not have any Instances in this region'. A large blue 'Launch instances' button is at the bottom.

7. Set the name as master to recognize it properly. Then select Ubuntu for Os Image:

The screenshot shows the AWS Launch Wizard. In the first step, 'Name and tags', the 'Name' field is set to 'master'. In the second step, 'Application and OS Images (Amazon Machine Image)', there's a search bar and a 'Quick Start' section with icons for Amazon Linux, macOS, Ubuntu, Windows, Red Hat, SUSE Linux, and Debian. Ubuntu is selected. Below this, the 'Amazon Machine Image (AMI)' details are shown: 'Ubuntu Server 24.04 LTS (HVM), SSD Volume Type', 'ami-020cba7c55df1f615 (64-bit (x86)) / ami-07041441b708acbd6 (64-bit (Arm))', 'Virtualization: hvm', 'ENAv enabled: true', 'Root device type: ebs', and a note 'Free tier eligible'.

For Instance type select t2.small.

The screenshot shows the 'Instance type' section of the AWS CloudFormation configuration. A dropdown menu is open for 't2.small'. The menu displays the following details:

- Family: t2
- 1 vCPU
- 2 GiB Memory
- Current generation: true
- On-Demand Windows base pricing: 0.032 USD per Hour
- On-Demand Linux base pricing: 0.023 USD per Hour
- On-Demand RHEL base pricing: 0.0376 USD per Hour
- On-Demand SUSE base pricing: 0.053 USD per Hour
- On-Demand Ubuntu Pro base pricing: 0.025 USD per Hour

Below the dropdown is a note: "Additional costs apply for AMIs with pre-installed software". To the right of the dropdown are two buttons: "All generations" (with a toggle switch) and "Compare instance types".

Notice that as per our need we can select instance of any size. Since this is for testing we are using small instance

Click on Create new key pair. You can also choose the default one but is good practice to create new key for different systems.

The screenshot shows the 'Key pair (login)' section. It includes a note: "You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance." Below this is a dropdown menu labeled "Key pair name - required" with the value "Select". To the right of the dropdown is a "Create new key pair" button with a circular arrow icon.

Give it a name. Select RSA for Key pair type and the format as .pem as we will be using openssh for access.

Then click on Create Key Pair

## Create key pair

X

### Key pair name

Key pairs allow you to connect to your instance securely.

pyspark

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

### Key pair type

RSA

RSA encrypted private and public key pair

ED25519

ED25519 encrypted private and public key pair

### Private key file format

.pem

For use with OpenSSH

.ppk

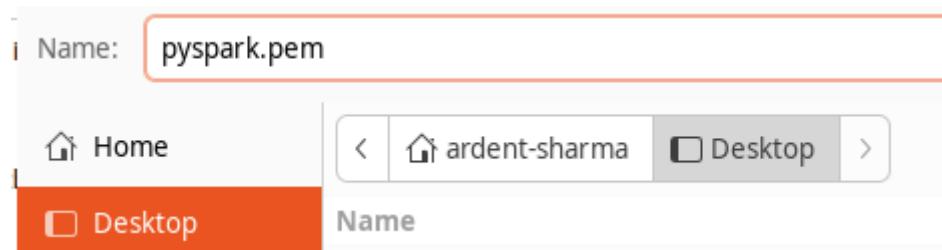
For use with PuTTY

**A** When prompted, store the private key in a secure and accessible location on your computer. You will need it later to connect to your instance. [Learn more](#) ↗

Cancel

Create key pair

It should automatically download the .pem key. Make sure you store it in safe place. I am saving it in Desktop. **It is very essential that you download and save this key as it is what we will be using in future to access our instance.**



Leave the network settings to default. This is not recommended for production environment. But since we are testing and also not an network/security expert lets leave it to default. Remember the name of the security group being created by default ‘launch-wizard-1’ as well be using this for slaves instances as well.

The screenshot shows the 'Network settings' section of the AWS Lambda configuration interface. At the top right is an 'Edit' button. Below it, under 'Network', is the identifier 'vpc-0e922083198bce206'. Under 'Subnet', it says 'No preference (Default subnet in any availability zone)'. Under 'Auto-assign public IP', it is set to 'Enable'. In the 'Firewall (security groups)' section, there is a note: 'A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.' Two options are shown: 'Create security group' (selected, indicated by a blue border) and 'Select existing security group'. Below this, it says 'We'll create a new security group called 'launch-wizard-1' with the following rules:' and lists three checkboxes: 'Allow SSH traffic from Anywhere (0.0.0.0/0)' (selected), 'Allow HTTPS traffic from the internet', and 'Allow HTTP traffic from the internet'. A warning message at the bottom states: '⚠ Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.' with an 'X' icon to close it.

Leave all the other settings to default as well.

Finally, in the summary section verify everything is as expected and click Launch Instance.

## ▼ Summary

Number of instances | [Info](#)

1



### Software Image (AMI)

Canonical, Ubuntu, 24.04, amd64...[read more](#)  
ami-020cba7c55df1f615

### Virtual server type (instance type)

t2.small

### Firewall (security group)

New security group

### Storage (volumes)

1 volume(s) - 8 GiB

[Cancel](#)

[Launch instance](#)

[Preview code](#)

You will get a success screen.

Success  
Successfully initiated launch of Instance i-0cec8d9181300bcf0

[▶ Launch log](#)

Click on the instance id starting from i-..... You will be redirected to the instance summary page.  
You can also access this page from previous route EC2 > Instances

Now let us connect to the newly created instance from our terminal using ssh and configure pyspark in that instance.

Instance summary for i-0cec8d9181300bcf0 (master) [Info](#)

Updated less than a minute ago

<b>Instance ID</b> <a href="#">i-0cec8d9181300bcf0</a>	<b>Public IPv4 address</b> <a href="#">54.81.189.82</a>   <a href="#">open address</a>	<b>Private IPv4 addresses</b> <a href="#">172.31.40.25</a>
<b>IPv6 address</b> -	<b>Instance state</b> <a href="#">Running</a>	<b>Public DNS</b> <a href="#">ec2-54-81-189-82.compute-1.amazonaws.com</a>   <a href="#">open address</a>
<b>Hostname type</b> IP name: ip-172-31-40-25.ec2.internal	<b>Private IP DNS name (IPv4 only)</b> <a href="#">ip-172-31-40-25.ec2.internal</a>	<b>Elastic IP addresses</b> -
<b>Answer private resource DNS name</b> IPv4 (A)	<b>Instance type</b> t2.small	<b>AWS Compute Optimizer finding</b> <a href="#">Opt-in to AWS Compute Optimizer for recommendations.</a>   <a href="#">Learn more</a>
<b>Auto-assigned IP address</b> <a href="#">54.81.189.82 [Public IP]</a>	<b>VPC ID</b> <a href="#">vpc-0e922083198bce206</a>	<b>Auto Scaling Group name</b> -
<b>IAM Role</b> -	<b>Subnet ID</b> <a href="#">subnet-0b1872c93129cb966</a>	<b>Managed</b> false
<b>IMDSv2</b> Required	<b>Instance ARN</b> <a href="#">arn:aws:ec2:us-east-1:193711398728:instance/i-0cec8d9181300bcf0</a>	
<b>Operator</b> -		

**Details**   [Status and alarms](#)   [Monitoring](#)   [Security](#)   [Networking](#)   [Storage](#)   [Tags](#)

**▼ Instance details** [Info](#)

<b>AMI ID</b> <a href="#">ami-020cba7c55df1f615</a>	<b>Monitoring</b> disabled	<b>Platform details</b> Linux/UNIX
<b>AMI name</b> <a href="#">ubuntu/images/hvm-ssd-gp3/ubuntu-noble-24.04-amd64-server-2025061</a>	<b>Allowed image</b> -	<b>Termination protection</b> Disabled
<b>Launch protection</b> None	<b>Launch time</b> -	<b>AMI location</b> -

8. Open terminal. Change directory to where you downloaded the .pem file and change its permission to read-only for owner and no access for other users. This is for security purpose.

```
ardent@ardent:~$ cd Desktop/
ardent@ardent:~/Desktop$ chmod 400 pyspark.pem
ardent@ardent:~/Desktop$
```

9. Connect to the instance using following command. Make sure you use your own pem file and public ipv4 address. Click yes when prompted to register the fingerprint.

ssh -i pyspark.pem [ubuntu@54.81.189.82](#)

Here,

ssh is the ssh-server tool we are using

-i is the flag for identify\_file or your security file. After this file you need to pass either the absolute or relative path to the location of .pem file created for the particular instance we are trying to connect to

ubuntu is the default username provided by aws

54.81.189.82 is the Public IPv4 address of the instance

```
ardent@ardent:~/Desktop$ ssh -i pyspark.pem ubuntu@54.81.189.82
The authenticity of host '54.81.189.82 (54.81.189.82)' can't be established.
ED25519 key fingerprint is SHA256:bwa9r6liAHDGN9bjfyIF5pVskhIFJrPU0hfccuHyLzhU.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '54.81.189.82' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.8.0-1029-aws x86_64)

 * Documentation: https://help.ubuntu.com
```

This should be your final output

```
individual files in /usr/share/doc/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-40-25:~$
```

This means you are accessing the terminal of the instance you just created in aws. All the commands you run from now on we will be executed on that instance. Let us setup this instance to run pyspark as a master/driver

```
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-40-25:~$ sudo apt update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [12
6 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [
```

Run the first two command we always run on a linux system after setup.

1. sudo apt update
2. sudo apt upgrade Press Y when prompted

```
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-40-25:~$ sudo apt update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [12
6 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [
```

```
Reading state information... Done
98 packages can be upgraded. Run 'apt list --upgradable' to see them.
ubuntu@ip-172-31-40-25:~$ sudo apt upgrade
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
The following NEW packages will be installed:
  linux-aws-6.14-headers-6.14.0-1010 linux-aws-6.14-tools-6.14.0-1010
```

Download spark 3.3.1 from apache archive directory using command line tool wget or curl. I am using wget.

```
ubuntu @ session #2: apt[1559], sshd[1059,1185]
ubuntu @ user manager service: systemd[1078]
ubuntu@ip-172-31-40-25:~$ wget https://archive.apache.org/dist/spark/spark-3.3.1
/spark-3.3.1-bin-hadoop3.tgz
--2025-08-02 05:39:35-- https://archive.apache.org/dist/spark/spark-3.3.1/spark
-3.3.1-bin-hadoop3.tgz
Resolving archive.apache.org (archive.apache.org)... 65.108.204.189, 2a01:4f9:1a
:a084::2
Connecting to archive.apache.org (archive.apache.org)|65.108.204.189|:443... con
nected.
HTTP request sent, awaiting response... 200 OK
Length: 299350810 (285M) [application/x-gzip]
Saving to: 'spark-3.3.1-bin-hadoop3.tgz'

n-hadoop3.tgz      5%[>          ]  16.94M   340KB/s  eta 13m 16s
```

wget <https://archive.apache.org/dist/spark/spark-3.3.1/spark-3.3.1-bin-hadoop3.tgz>

Wait for a while as it might take some time depending on the internet connection. After the download has completed extract the compressed files using following command

```
ubuntu@ip-172-31-40-25:~$ ls
spark-3.3.1-bin-hadoop3.tgz
ubuntu@ip-172-31-40-25:~$ tar -xvzf spark-3.3.1-bin-hadoop3.tgz
spark-3.3.1-bin-hadoop3/
spark-3.3.1-bin-hadoop3/LICENSE
  NOTICE
```

tar -xvzf spark-3.3.1-bin-hadoop3.tgz

Let us install java as spark requires java. We will be using the below command to install. Notice how the steps are similar to what we did to setup our own system and to setup a new server instance.

```
full log in /opt/spark/logs/spark-ubuntu-org.apache.spark.deploy.master.Master-1  
-ip-172-31-40-25.out  
ubuntu@ip-172-31-40-25:~$ sudo apt install openjdk-11-jdk  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done
```

```
sudo apt install openjdk-11-jdk
```

Now, Let us move the extracted directory to /opt/spark . This is the standard directory for spark. Use the below command. Make sure you use sudo before mv since only root user is allowed access to root directories

```
ubuntu@ip-172-31-40-25:~$ mv spark-3.3.1-bin-hadoop3 /opt/spark  
mv: cannot move 'spark-3.3.1-bin-hadoop3' to '/opt/spark': Permission denied  
ubuntu@ip-172-31-40-25:~$ sudo mv spark-3.3.1-bin-hadoop3 /opt/spark  
ubuntu@ip-172-31-40-25:~$
```

```
sudo mv spark-3.3.1-bin-hadoop3 /opt/spark
```

Now let us update the bashrc file to provide information regarding spark directory. Open the .bashrc file using either nano or vim. I will be using vim. Whether you use nano or vim. Make sure you know how to make changes and how to save and exit properly

```
buntu@ip-172-31-40-25:~$ sudo mv spark-3.3.1-bin-hadoop3 /opt/spark  
buntu@ip-172-31-40-25:~$ vim .bashrc  
buntu@ip-172-31-40-25:~$
```

```
fi  
fi  
  
export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64  
export SPARK_HOME=/opt/spark  
export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin  
export PYSPARK_PYTHON=/usr/bin/python3  
  
".bashrc" 123L, 3944B written
```

118,0-1 Bot

```
export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64  
export SPARK_HOME=/opt/spark  
export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin  
export PYSPARK_PYTHON=/usr/bin/python3
```

Since the terminal does not know that the .bashrc has been modified we need to apply/source the changes ourselves using below command. This should not be new to you anymore. If it is spend more time exploring the terminal in your linux system.

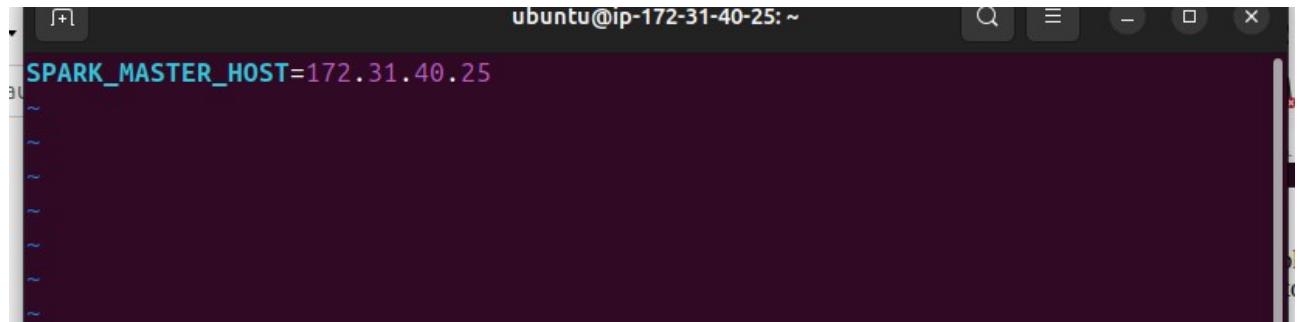
```
No containers need to be restarted.  
User sessions running outdated binaries:  
ubuntu @ session #2: sshd[1059,1185]  
ubuntu @ user manager service: systemd[1078]  
ubuntu@ip-172-31-40-25:~$ vim .bashrc  
ubuntu@ip-172-31-40-25:~$ source .bashrc  
ubuntu@ip-172-31-40-25:~$
```

source .bashrc

Let us modify the spark-env file to provide ip of the master. Use the below command to create and open the file in vim. As before you can use any other terminal based text editors. Make sure you use sudo to open the file. Also if you cd into the conf directory you will notice that there is a template file for spark-env which you can look at your own leisure to get idea on what kind of configuration can be done.

```
ubuntu@ip-172-31-40-25:~$ sudo vim /opt/spark/conf/spark-env.sh
```

Add the private ip of the master instance. And assign it to SPARK\_MASTER\_HOST. Make sure you use the same name and there is no spaces before and after the assignment operator. The syntax is same for all bash files.



A screenshot of a terminal window titled "ubuntu@ip-172-31-40-25: ~". The window contains a single line of text: "SPARK\_MASTER\_HOST=172.31.40.25". The background of the terminal is dark, and the text is white.

Before we run our spark master server. We need to make one small change we overlook initially. In order to view whether the master is running or not from other system. We need to allow http/https (technically only http should suffice but lets give all access available) access to that system. If you remember initially we only provided ssh accessing

Go to EC2> Security Groups and open the launch-wizard-1 security group by clicking the link under security group id

The screenshot shows the AWS EC2 console under the 'Security Groups' section. A single security group named 'launch-wizard-1' is listed. The table includes columns for Name, Security group ID, VPC ID, Description, and Owner. The 'Description' column shows the creation details: 'launch-wizard-1 created 2025-08-02T0... 193711398728'.

Click on Edit inbound rules

The screenshot shows the 'Inbound rules' tab selected. One rule is listed: 'sgr-092a21814a9dc04f6' (IPv4, SSH, TCP port 22, source 0.0.0.0/0). There are tabs for Outbound rules, Sharing - new, VPC associations - new, and Tags.

Remove the existing rule and add new rule For All Traffics and range 0.0.0.0/0. This essentially means any type access from anywhere. This setup is security hazard waiting to happen and is not recommended at all. We are only doing it for ease.

The screenshot shows the 'Edit inbound rules' page. A new rule is being added with the following configuration: Security group rule ID 'sgr-02f793bdd82140be4', Type 'All traffic', Protocol 'TCP', Port range 'All', Source '0.0.0.0/0', and Description 'optional'. A warning message at the bottom states: '⚠ Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.' Buttons for Add rule, Cancel, Preview changes, and Save rules are visible.

It is finally time to start the master-server. Use the below command:

```
ubuntu@ip-172-31-40-25:~$ start-master.sh
starting org.apache.spark.deploy.master.Master, logging to /opt/spark/logs/spark-ubuntu-org.apache.spark.deploy.master.Master-1-ip-172-31-40-25.out
ubuntu@ip-172-31-40-25:~$
```

start-master.sh

If you check this script is available in /opt/spark/sbin which we have setup in our .bashrc file's PATH variable hence we can run it from anywhere.

Now if you go to the browser and use your public\_ip:8080 you should be able to see the below view. Also notice that the actual spark master is running at private\_ip:7077 . Port 8080 provides the webui

Spark Master at spark://1

Not Secure http://54.81.189.82:8080

Apache Spark 3.3.1

Spark Master at spark://172.31.40.25:7077

URL: spark://172.31.40.25:7077

Alive Workers: 0

Cores in use: 0 Total, 0 Used

Memory in use: 0.0 B Total, 0.0 B Used

Resources in use:

Applications: 0 Running, 0 Completed

Drivers: 0 Running, 0 Completed

Status: ALIVE

Workers (0)

Worker Id	Address	State	Cores	Memory	Resources
-----------	---------	-------	-------	--------	-----------

Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
----------------	------	-------	---------------------	------------------------	----------------	------	-------	----------

Completed Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
----------------	------	-------	---------------------	------------------------	----------------	------	-------	----------

Now, it is time to spin up two new instances that will be the slave/executor. The initial setup is identical to that of master. Everything is exactly the same. So you should be able to do it. Just make sure you provide relevant name to them such as slave1 and slave2. Use the same key we generate earlier and the same security group we created and modify earlier for ease of use.

Since the setup is same I will just provide images for reference. If you need detail go above.

**Name and tags** [Info](#)

**Name**

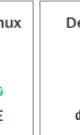
[Add additional tags](#)

**▼ Application and OS Images (Amazon Machine Image)** [Info](#)

An AMI contains the operating system, application server, and applications for your instance. If you don't see a suitable AMI below, use the search field or choose [Browse more AMIs](#).

Search our full catalog including 1000s of application and OS Images

[Recents](#) [Quick Start](#)

[Browse more AMIs](#)   
Including AMIs from AWS, Marketplace and the Community

**Amazon Machine Image (AMI)**

Ubuntu Server 24.04 LTS (HVM), SSD Volume Type ami-020cba7c55df1f615 (64-bit (x86)) / ami-07041441b708acbd6 (64-bit (Arm)) Virtualization: hvm ENA enabled: true Root device type: ebs	<a href="#">Free tier eligible</a> ▾
--	--------------------------------------

**Description**  
Ubuntu Server 24.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).  
Canonical, Ubuntu, 24.04, amd64 noble image

For instance type select t2.medium. Since these our executor we are choosing slightly better instance. There are several instance to choose from in aws. Some are memory optimized, some are compute optimized, some general purpose and so on. Note that we are not considering any of that currently. Since pyspark is memory heavy. A memory optimized instance for executor/slave would be a better choice. Driver/master , however, could benefit for better compute and so on.

**▼ Instance type** [Info](#) | [Get advice](#)

**Instance type**

**t2.medium**

Family: t2 2 vCPU 4 GiB Memory Current generation: true  
On-Demand Ubuntu Pro base pricing: 0.0499 USD per Hour On-Demand Linux base pricing: 0.0464 USD per Hour  
On-Demand RHEL base pricing: 0.0752 USD per Hour On-Demand Windows base pricing: 0.0644 USD per Hour  
On-Demand SUSE base pricing: 0.1464 USD per Hour

[All generations](#) [Compare instance types](#)

**Additional costs apply for AMIs with pre-installed software**

Notice we selected previous created key and existing network security group

## ▼ Key pair (login) [Info](#)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - **required**

pyspark

 [Create new key pair](#)

## ▼ Network settings [Info](#)

 [Edit](#)

Network 

vpc-0e922083198bce206

Subnet 

No preference (Default subnet in any availability zone)

Auto-assign public IP 

Enable

Firewall (security groups) 

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

[Create security group](#)

[Select existing security group](#)

Common security groups 

Select security groups

 [Compare security group rules](#)

launch-wizard-1 sg-08f3a5c71bd5dae6a 

VPC: vpc-0e922083198bce206

Security groups that you add or remove here will be added to or removed from all your network interfaces.

For ease of use in the summary section select number of instance to 2 and click launch instnace

## ▼ Summary

Number of instances | [Info](#)

2



When launching more than 1 instance, [consider EC2 Auto Scaling](#)

### Software Image (AMI)

Canonical, Ubuntu, 24.04, amd64...[read more](#)

ami-020cba7c55df1f615

### Virtual server type (instance type)

t2.medium

### Firewall (security group)

New security group

### Storage (volumes)

1 volume(s) - 8 GiB

[Cancel](#)

**Launch instance**

[Preview code](#)

You will be redirected to success page. Go to EC2>Instances page. You will see the two newly created instances. Change the name of one of them from slave1 to slave2 for easy recognition

<input type="checkbox"/> slave1	i-0754091c2e03d62a2	<span>Running</span>	t2.medium	<span>Initializing</span>	<a href="#">View alarms +</a>	us-east-1d	ec2-54-174-126-58.co...	54.174.126.38	-
<input checked="" type="checkbox"/> slave2	x ✓ i-05afc6796104147ac	<span>Running</span>	t2.medium	<span>Initializing</span>	<a href="#">View alarms +</a>	us-east-1d	ec2-44-202-83-45.com...	44.202.83.45	-

Open two new terminal in your system. Go to directory with .pem file, connect to two new instance using ssh and configure them. I won't be going in detail as initial configuration part is same. The configuration is exactly the same except for the executing part. Master will be started using command start-master.sh and Slave will be started using start-worker.sh master\_ip

After following exactly the same process except for the security part as we are using the same group which has already been configured for ease. We will run the following command:

```
ubuntu@ip-172-31-84-118:~$ start-worker.sh spark://172.31.40.25:7077
starting org.apache.spark.deploy.worker.Worker, logging to /opt/spark/logs/spark-ubuntu-org.apache.spark.deploy.worker.W
orker-1-ip-172-31-84-118.out
ubuntu@ip-172-31-84-118:~$
```

start-worker.sh spark://<master node IP address>:7077

Now, if you go to the web ui and refresh you should be able to see worker added in Workers Section

The screenshot shows the Apache Spark 3.3.1 Web UI. At the top, it says "Spark Master at spark://172.31.40.25:7077". Below that, it lists system metrics: URL: spark://172.31.40.25:7077, Alive Workers: 1, Cores in use: 2 Total, 0 Used, Memory in use: 2.8 GiB Total, 0.0 B Used, Resources in use: Applications: 0 Running, 0 Completed, Drivers: 0 Running, 0 Completed, Status: ALIVE. Under the "Workers" section, there is one entry: worker-20250802072335-172.31.84.118-43049, which is ALIVE with 2 cores and 2.8 GiB memory. There are also sections for "Running Applications" (0) and "Completed Applications" (0), both currently empty.

Configure the next worker/slave/executor as well and then refresh the web ui and it should be reflected as well.

The screenshot shows the Apache Spark 3.3.1 Web UI with two workers listed under the "Workers" section. The first worker is the same as before: worker-20250802072335-172.31.84.118-43049, ALIVE with 2 cores and 2.8 GiB memory. The second worker is worker-20250802073428-172.31.84.134-43003, also ALIVE with 2 cores and 2.8 GiB memory. The other sections ("Running Applications" and "Completed Applications") remain empty.

**We finally have our cluster setup. Our master is running and connected to two slave nodes. Unless you are doing below steps parallel to ec2 instances cluster setup. It is wise to first stop the instances so that the credit does not accumulate**

Instances (3/3) Info

Stop instance

Start instance

Reboot instance

Hibernate instance

Terminate (delete) instance

3 instances selected

Let us setup our data storage in AWS. We will be using S3 for this task. We will only be running the transform and load section of the code in our cluster. So let us create a bucket in s3 and store the output of our extract stage their. I encourage students to later implement the extract section as well by themselves

#### 10. Go to search bar and search S3. Click the result

aws

Search: s3

S3 Scalable Storage in the Cloud

In the Amazon S3 home page click Create bucket.

Storage

**Amazon S3**

Store and retrieve any amount of data from anywhere

Amazon S3 is an object storage service that offers industry-leading scalability, data availability, security, and performance.

Create a bucket

Every object in S3 is stored in a bucket. To upload files and folders to S3, you'll need to create a bucket where the objects will be stored.

Create bucket

Give it a name. Make sure the name is unique. It needs to be unique across all users in a particular region. Use your name or something unique. Remember the name you give it will be important to access the files later

Amazon S3 > Buckets > Create bucket

## Create bucket Info

Buckets are containers for data stored in S3.

### General configuration

AWS Region  
US East (N. Virginia) us-east-1

Bucket type Info

General purpose  
Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.

Directory  
Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.

Bucket name Info  
spotifydatasummerclass

Bucket names must be 3 to 63 characters and unique within the global namespace. Bucket names must also begin and end with a letter or number. Valid characters are a-z, 0-9, periods (.), and hyphens (-). [Learn More](#)

Copy settings from existing bucket - optional  
Only the bucket settings in the following configuration are copied.  
[Choose bucket](#)

Format: s3://bucket/prefix

Scroll to the bottom. Keep everything else as default. Click Create Bucket

► Advanced settings

ⓘ After creating the bucket, you can upload files and folders to the bucket, and configure additional bucket settings.

[Cancel](#) [Create bucket](#)

Click on the bucket name to open the bucket

Amazon S3 > Buckets

Successfully created bucket "spotifydatasummerclass"  
To upload files and folders, or to configure additional bucket settings, choose [View details](#).

[View details](#) [X](#)

General purpose buckets All AWS Regions [Directory buckets](#)

General purpose buckets (1) Info

[Copy ARN](#) [Empty](#) [Delete](#) [Create bucket](#)

Buckets are containers for data stored in S3.

[spotifydatasummerclass](#) US East (N. Virginia) us-east-1 August 2, 2025, 16:41:58 (UTC+05:45)

Find buckets by name

Name AWS Region Creation date

► Account snapshot Info  
Updated daily [View dashboard](#)  
Storage Lens provides visibility into storage usage and activity trends.

► External access summary - new  
Updated daily [Info](#)  
External access findings help you identify bucket permissions that allow public access or access from other AWS accounts.

Click on Create Folder to create new folder inside the bucket

The screenshot shows the 'Objects' tab selected in the navigation bar. Below it, there's a section titled 'Objects (0)' with various actions like Copy S3 URI, Copy URL, Download, Open, Delete, Actions, Create folder, and Upload. A note says 'Objects are fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)'.

Give name to bucket as extract. This is where we will upload the output of our extract

The screenshot shows the 'Create folder' dialog. In the 'Folder' section, the 'Folder name' field contains 'extract'. A note below it says 'Folder names can't contain "/"'. A blue-bordered box contains a warning: 'Your bucket policy might block folder creation. If your bucket policy prevents uploading objects without specific tags, metadata, or access control list (ACL) grantees, you will not be able to create a folder using this configuration. Instead, you can use the [upload configuration](#) to upload an empty folder and specify the appropriate settings.'

Leave everything else to default and click on Create Folder

The screenshot shows the 'Create folder' dialog with encryption settings. A note says 'The following encryption settings apply only to the folder object and not to sub-folder objects.' Under 'Server-side encryption', the 'Don't specify an encryption key' option is selected. A note says 'The bucket settings for default encryption are used to encrypt the folder object when storing it in Amazon S3.' The 'Specify an encryption key' option is also present. A note says 'The specified encryption key is used to encrypt the folder object before storing it in Amazon S3.' A yellow-bordered box contains a warning: 'If your bucket policy requires objects to be encrypted with a specific encryption key, you must specify the same encryption key when you create a folder. Otherwise, folder creation will fail.' At the bottom are 'Cancel' and 'Create folder' buttons.

Open the newly created Folder by clicking on it

The screenshot shows the AWS S3 Objects page. At the top, there is a toolbar with buttons for Copy S3 URI, Copy URL, Download, Open, Delete, Actions, Create folder, and Upload. Below the toolbar, a message states: "Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 Inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)". A search bar labeled "Find objects by prefix" contains the text "extract/". Below the search bar is a table with columns: Name, Type, Last modified, Size, and Storage class. There is one item in the table: "extract/" (Folder). The table has sorting arrows for Name, Last modified, and Storage class.

Click on the Upload button to upload our files

The screenshot shows the AWS S3 Objects page for the folder "extract/". The interface is identical to the previous screenshot, with the same toolbar, message, search bar, and table. The table below the search bar displays the message "No objects" and "You don't have any objects in this folder." At the bottom of the table area, there is a prominent orange "Upload" button.

Click on Add Files to upload files

**Upload** Info

Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDKs or Amazon S3 REST API. [Learn more](#)

Drag and drop files and folders you want to upload here, or choose **Add files** or **Add folder**.

**Files and folders (0)**

All files and folders in this table will be uploaded.

Find by name

Name	Folder	Type	Size
No files or folders You have not chosen any files or folders to upload.			

**Destination**

Browse to the extract directory where you have extracted your files. Select all 3 files and Upload them

**Upload** Info

Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDKs or Amazon S3 REST API. [Learn more](#)

Drag and drop files and folders you want to upload here, or choose **Add files** or **Add folder**.

**Files and folders (0)**

All files and folders in this table will be uploaded.

Find by name

Name	Folder	Type	Modified
artists.csv		Text	28 Jun
fixed_da.json		Program	28 Jun
tracks.csv		Text	28 Jun

Open files read-only

**File Upload**

Cancel

Recent Home Document Downloads Music Pictures Videos

ardent Data extract

Name Size Type Modified

artists.csv 64.9 MB Text 28 Jun

fixed\_da.json 259.4 MB Program 28 Jun

tracks.csv 111.4 MB Text 28 Jun

All Files

Scroll down and click the upload button. Note the Destination url this is important as it will be our input directory for our transform code

The screenshot shows the AWS Lambda function configuration page. At the top, there is a search bar labeled "Find by name" and filter options for "Name", "Folder", "Type", and "Size". Below this, a section titled "Destination" has an "Info" link. The "Destination" field contains the value "s3://spotifydatasummerclass/extract/" with a copy icon next to it. A "Destination details" section describes bucket settings for new objects. Below this are sections for "Permissions" (granting public access) and "Properties" (specifying storage class, encryption, and tags). At the bottom right are "Cancel" and "Upload" buttons.

It might take some time depending on the internet. Wait for a while

The screenshot shows the AWS Lambda function configuration page with an "Uploading" progress bar at the top. It indicates 2% completion, with 3 files totaling 406.5 MB (97.83% complete). The estimated time remaining is 10 minutes, and the transfer rate is 706.3 KB/s. Below the progress bar, the destination URL is "s3://spotifydatasummerclass/extract/". At the bottom, there are tabs for "Files and folders" (selected) and "Configuration". The "Files and folders" tab shows a summary of 3 total files (415.5 MB) and a search bar. The file list table includes columns for Name, Folder, Type, Size, Status, and Error.

**Upload succeeded**  
For more information, see the [Files and folders](#) table.

**Summary**

Destination	Succeeded	Failed
<a href="#">s3://spotifydatasummerclass/extract/</a>	3 files, 415.5 MB (100.00%)	0 files, 0 B (0%)

**Files and folders** [Configuration](#)

**Files and folders (3 total, 415.5 MB)**

Name	Folder	Type	Size	Status	Error
<a href="#">tracks.csv</a>	-	text/csv	106.2 MB	<span style="color: green;">Succeeded</span>	-
<a href="#">fixed_da.json</a>	-	application/json	247.4 MB	<span style="color: green;">Succeeded</span>	-
<a href="#">artists.csv</a>	-	text/csv	61.9 MB	<span style="color: green;">Succeeded</span>	-

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

When it has completed click on Destination url and you will be taken to the relevant folder of bucket.

[Amazon S3](#) > [Buckets](#) > [spotifydatasummerclass](#) > [extract/](#)

**extract/**

[Copy S3 URI](#) [Copy URL](#) [Download](#) [Open](#) [Delete](#) [Actions](#) [Create folder](#) [Upload](#)

**Objects (3)**

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 Inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Name	Type	Last modified	Size	Storage class
<a href="#">artists.csv</a>	csv	August 2, 2025, 16:45:56 (UTC+05:45)	61.9 MB	Standard
<a href="#">fixed_da.json</a>	json	August 2, 2025, 16:45:56 (UTC+05:45)	247.4 MB	Standard
<a href="#">tracks.csv</a>	csv	August 2, 2025, 16:45:56 (UTC+05:45)	106.2 MB	Standard

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Click Copy s3 Uri available at the top: your uri should be something like this:  
 s3://spotifydatasummerclass/extract/  
 s3://The name of your bucket/folder

For pyspark however, we will need to pass an extra a after s3 i.e.,  
 s3a://The name of your bucket/folder

This behavior can be changed from config but we will let it be for now. Remember this as we will need this later

While it is uploading let us setup Access keys using AWS IAM which is user and identity management module of AWS. Generally we would have to do this ourselves (sysadmin/devops) but in our learner lab the access keys have already been provided by default.

Go to the initial lab page

The screenshot shows the AWS Academy Learner Lab interface. On the left is a sidebar with icons for Account, Dashboard, Courses, Calendar, Inbox, History, and Help. The main area has a breadcrumb navigation: ALLv2EN-US-LTI13-125325 > Modules > AWS Academy Learner Lab > Launch AWS Academy Learner Lab. The top right shows session details: 03:24, Start Lab, End Lab, AWS Details, Readme, Reset, and a budget of Used \$0 of \$50. Below this is a terminal window showing the command 'eee\_W\_4713859@runweb184014:~\$'. To the right is a 'Learner Lab' sidebar with links to Environment Overview, Environment Navigation, Access the AWS Management Console, Region restriction, Service usage and other restrictions, and Using the terminal in the browser. At the bottom are navigation buttons for Previous and Next.

Click the aws detail button

The screenshot shows the AWS Academy Learner Lab interface with the AWS Details button highlighted. The modal window displays session information: Cloud Access, AWS CLI (with a Show button), Cloud Labs, Remaining session time: 03:22:05(203 minutes), Session started at: 2025-08-02T03:51:40-0700, and Session to end at: [redacted].

Click the Show button right next to AWS CLI. Copy the aws\_access\_key\_id, aws\_secret\_access\_key, aws\_session\_token and save it somewhere. Or you can visit this page when you need it later for setup

The screenshot shows the AWS Cloud Access page with the title 'eee\_W\_47138590'. It displays the AWS CLI configuration section with the following content:

```
[default]
aws_access_key_id=ASIA26QZL5ENFFH3A7R
aws_secret_access_key=oz27QpYN8cj0k84L4JGXaXbTAaVmjkwoPuP17A+K
aws_session_token=IQoJb3JpZ2luX2VjEN//////////wEaXvzLXdIc3QtMjJIMEYCIQDnG8WIvN53s
IcrUkHyoXZklzWtXV40uCkDjLGh/ux9TQiAJgtfCxHSdX/0lU0RcpLQFhywa1vMuylJo7CRjNRquoYKr8C
CBQQABoMMTkzNzExMzk4NzI4Igw12U6n7kxUx80A95gqmAJdhvho8X32J9LSKcoz19efHm9b6EH4BcRIhq
zPDetyI/AJisT2WdjXCUGGBsR3k2JQw5auN40XjN8whI1ZMEld58Rip9oE8YKYWhx1H6LuMg6i8TsKolGx1
KLUhniTnDcy7ggMDPp0Q1lVvV/Chd1Guv2av001DTmrkA14DfGvDm0m0rUUMTdnYp+GahcDgAAnDnuKv
```

Let us make changes to our code to make it more dynamic: In the create\_spark\_session of both transform and load and a new parameter for spark\_config

```
7     sys.path.append(os.path.abspath(os.path.join(os.path.dirname(__file__), ..)))
8     from utility.utility import setup_logging, format_time
9
10    def create_spark_session(logger, spark_config):
11        """Initialize Spark session."""
12        logger.info("Starting spark session")
13        return (SparkSession.builder
14            .master(f"spark://{{spark_config['master_ip']}":7077})
15            .appName("SpotifyDataTransform")
16            .config("spark.driver.memory", spark_config["driver_memory"])
17            .config("spark.executor.memory", spark_config["executor_memory"])
18            .config("spark.executor.cores", spark_config["executor_cores"])
19            .config("spark.executor.instances", spark_config["executor_instances"])
20            .getOrCreate())
21
22
```

We will be getting them as argument at the start

```
if __name__ == "__main__":
    logger = setup_logging("transform.log")

    if len(sys.argv) != 8:
        logger.critical("Usage: python3 transform/execute.py <input_dir> <output_dir> master_ip d_mem e_mem e_core e_inst")
        sys.exit(1)

    input_dir = sys.argv[1]
    output_dir = sys.argv[2]
    spark_config = {}
    spark_config["master_ip"] = sys.argv[3]
    spark_config["driver_memory"] = sys.argv[4]
    spark_config["executor_memory"] = sys.argv[5]
    spark_config["executor_cores"] = sys.argv[6]
    spark_config["executor_instances"] = sys.argv[7]

    start = time.time()
    spark = create_spark_session(logger, spark_config)
    artists_df, recommendations_df, tracks_df = load_and_clean(logger, spark, input_dir, output_dir)
```

Do the same for load module. Additionally, for load module make sure postgres username, postgres password and postgres host all are also being passed as arguments

```
if __name__ == "__main__":
    logger = setup_logging("load.log")

    if len(sys.argv) != 9:
        logger.error("Usage: python load/execute.py <input_dir> <pg_un> <pg_pw> pg_host d_mem e_mem e_core e_inst")
        sys.exit(1)

    input_dir = sys.argv[1]
    pg_un = sys.argv[2]
    pg_pw = sys.argv[3]
    pg_host = sys.argv[4]

    spark_config = {}
    spark_config["master_ip"] = sys.argv[5]
    spark_config["driver_memory"] = sys.argv[6]
    spark_config["executor_memory"] = sys.argv[7]
    spark_config["executor_cores"] = sys.argv[8]
    spark_config["executor_instances"] = sys.argv[9]

    logger.info("Load stage started")
    start = time.time()

    spark = create_spark_session(logger, spark_config)
    create_postgres_tables(logger, pg_un, pg_pw, pg_host)
```

Push the changes to git. Note here I did not need to do git add as I have not added anything new but it is matter of habit and it does no harm

```
(venv) ardent@ardent:~/Workspace/etl$ git add .
(venv) ardent@ardent:~/Workspace/etl$ git commit -m "chore: logger message changes"
[master dcac62e] chore: logger message changes
 2 files changed, 2 insertions(+), 2 deletions(-)
(venv) ardent@ardent:~/Workspace/etl$ git push origin master
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 568 bytes | 568.00 KiB/s, done.
Total 6 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To github.com:neotheweobserver/spark-etl.git
  2ab260d..dcac62e  master -> master
(venv) ardent@ardent:~/Workspace/etl$
```

**Now it is time to start the instances again. If you never stopped the instances then you do not need start again. The public ip of the instance might change when you start it again. So make sure you take note of it. The process of starting is same as stopping. Go to EC2> Instance select all instance and click on Start Instance. Let us open three different terminal as before and ssh into the 3 instances.**

The screenshot shows the AWS CloudWatch Instances console. On the left, there's a sidebar with navigation links like 'Instances', 'Launch instances', 'Stop instances', 'Create instance type', 'Create instance template', 'Create instance profile', and 'Create instance role'. The main area is titled 'Instances (3/3)' and contains a table with three rows. The columns are 'Alarm status', 'Availability Zone', 'Public IPv4 DNS', 'Public IPv4 IP', 'Elastic IP', and 'IPv6 IPs'. The first two rows have a blue background, and the third row has a red background. Each row has a 'View alarms' button and a link to its public DNS. At the bottom of the table, it says '3 instances selected'.

Use the public ip's to ssh using the previous pem key. At this point you should be able to do this on your own.

```
ardent@ardent:~/Desktop$ ssh -i pyspark.pem ubuntu@54.236.77.201
The authenticity of host '54.236.77.201 (54.236.77.201)' can't be established.
ED25519 key fingerprint is SHA256:bwa9r6liAHGN9bjfyIF5pVskhIFJrPU0hfcuHyLzhU.
This host key is known by the following other names/addresses:
  ~/.ssh/known_hosts:4: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
```

```
Connection to 54.174.126.38 closed by remote host.
Connection to 54.174.126.38 closed.
ardent@ardent:~/Desktop$ ssh -i pyspark.pem ubuntu@54.236.141.139
The authenticity of host '54.236.141.139 (54.236.141.139)' can't be established.
ED25519 key fingerprint is SHA256:NPyFKjMk/at2BV7wYJnpHdDictDdMj2QR+O9gFoTvp4.
This host key is known by the following other names/addresses:
  ~/.ssh/known_hosts:7: [hashed name]
```

```
Connection to 44.202.83.45 closed by remote host.
Connection to 44.202.83.45 closed.
ardent@ardent:~/Desktop$ ssh -i pyspark.pem ubuntu@44.206.230.43
The authenticity of host '44.206.230.43 (44.206.230.43)' can't be established.
ED25519 key fingerprint is SHA256:LduE6 rtn48PSmLjtdtT2BGmUBDGabemleWx fbUK94ME.
This host key is known by the following other names/addresses:
  ~/.ssh/known_hosts:10: [hashed name]
```

Now that we have connected to the master as well as 2 worker node. Let us perform below task in all nodes.

In the .bashrc file of all 3 nodes. Add/export the aws key id, secret key, and access\_token we copied earlier. This is required for our instance to access s3 buckets and other aws services.

```
if
fi

export AWS_ACCESS_KEY_ID=ASIAS2GQZL5ENFFH3A7R
export AWS_SECRET_ACCESS_KEY=o2z7QpYN8cj0k84L4JGXaXbTAaVmjkwoPuPi7A+K
export AWS_SESSION_TOKEN=IQoJb3JpZ2luX2VjEnv//////////wEaCXVzLXdIc3QtMiJIMEYCIQDnG8WIvN53sIcrUkHyoXZklzWtXV40uCkDjLGH
9TQIhAJgtfCxHSdx/0lU0RpclQFhywa1vMuylJo7CRjNRquoYKr8CCBQQABoMMTkzNzExMzk4NzI4Igw12U6n7kxUx80A9SgqnAJdhvho8X32J90LSKcc
efHm9b6EH4BcR1hzqPDetyI/AjisT2WdjXCUGGBsr3k2JQw5auN40Xjn8whI1ZMeld58RIP9oE8YKvWnx1H6LuMg618TsKolGx1KLHhniIaDCXzqgMPD
UcVVChda1GaHY3cX90JPInreKL1PFGUpcOm8ocHHMDpXrtGahsBgAeDvwKxsp/m3Xn5WFv46E1qChl/RS84mJAwP/kmKJrdBhBfZHkyGpvFwjJa6lBx
nJyeeRUUje09UUFcLC9i5oclQi1d/AcnT2j8Z7CfCjdEAn0WXebngr01Ql1jLiY2MHjh80sQeI/L3x5s+0RvLQcbpMgPA5IKa3GEE+d5mAqsyTi7BCFB
927fEBjqcaQiqVpf3M46n4w31rGcXXwQXgx5Ik1ExEr5ntmMDmnSeERXQLayJ9TrUV+mYcbcjRhkpHr45TwDgztI4gy23y/FzF+vogVp8pnab35Ppd
/C02+5Ah+0lYwEeh+DTADLi3AQeYS05ySa0FEMIp/VKF3iiwOpwMiHeyLl1EiMEHuNTm1rKLVkJ8pJ61GyX0169qS0g7FuCMsA==

export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
export SPARK_HOME=/opt/spark
export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin
export PYSPARK_PYTHON=/usr/bin/python3
```

Everytime the .bashrc file is changed it needs to be sourced to update

```
ubuntu@ip-172-31-40-25:~$ source .bashrc
```

### Do this for all 3 instances.

Now, we need to download the required jars for our pyspark to be able to connect to aws. This will be done for all 3 nodes as well

Change directory to jars of /opt/spark → remember this is where we kept our spark files

```
ubuntu@ip-172-31-40-25:~$ vim .bashrc
ubuntu@ip-172-31-40-25:~$ cd /opt/spark/jars
ubuntu@ip-172-31-40-25:/opt/spark/jars$
```

Download two jars file

```
ubuntu@ip-172-31-40-25:~$ cd /opt/spark/jars
ubuntu@ip-172-31-40-25:/opt/spark/jars$ wget https://repo1.maven.org/maven2/org/apache/hadoop/hadoop-aws/3.3.1/hadoop-aws-3.3.1.jar
--2025-08-02 11:48:42-- https://repo1.maven.org/maven2/org/apache/hadoop/hadoop-aws/3.3.1/hadoop-aws-3.3.1.jar
Resolving repo1.maven.org (repo1.maven.org)... 146.75.32.209, 2a04:4e42:78::209
Connecting to repo1.maven.org (repo1.maven.org)|146.75.32.209|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 870644 (850K) [application/java-archive]
Saving to: 'hadoop-aws-3.3.1.jar'

hadoop-aws-3.3.1.jar      100%[=====] 850.24K  ---KB/s   in 0.009s
2025-08-02 11:48:42 (90.1 MB/s) - 'hadoop-aws-3.3.1.jar' saved [870644/870644]

ubuntu@ip-172-31-40-25:/opt/spark/jars$ wget https://repo1.maven.org/maven2/com/amazonaws/aws-java-sdk-bundle/1.12.99/aws-java-sdk-bundle-1.12.99.jar
--2025-08-02 11:49:23-- https://repo1.maven.org/maven2/com/amazonaws/aws-java-sdk-bundle/1.12.99/aws-java-sdk-bundle-1.12.99.jar
Resolving repo1.maven.org (repo1.maven.org)... 146.75.36.209, 2a04:4e42:79::209
Connecting to repo1.maven.org (repo1.maven.org)|146.75.36.209|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 246882976 (235M) [application/java-archive]
Saving to: 'aws-java-sdk-bundle-1.12.99.jar'

aws-java-sdk-bundle-1.12.99.jar 100%[=====] 235.45M  122MB/s   in 1.9s
2025-08-02 11:49:25 (122 MB/s) - 'aws-java-sdk-bundle-1.12.99.jar' saved [246882976/246882976]

ubuntu@ip-172-31-40-25:/opt/spark/jars$
```

[wget https://repo1.maven.org/maven2/org/apache/hadoop/hadoop-aws/3.3.1/hadoop-aws-3.3.1.jar](https://repo1.maven.org/maven2/org/apache/hadoop/hadoop-aws/3.3.1/hadoop-aws-3.3.1.jar)  
[wget https://repo1.maven.org/maven2/com/amazonaws/aws-java-sdk-bundle/1.12.99/aws-java-sdk-bundle-1.12.99.jar](https://repo1.maven.org/maven2/com/amazonaws/aws-java-sdk-bundle/1.12.99/aws-java-sdk-bundle-1.12.99.jar)

### Do the above jar download task for all 3 instances.

Now let us start the spark in master and worker and worker nodes. For master use start-master.sh. For worker use start-worker master\_ip:7077

```
ubuntu@ip-172-31-40-25:~$ start-master.sh
starting org.apache.spark.deploy.master.Master, logging to /opt/spark/logs/spark-ubuntu-org.apache.spark.deploy.master.Master-1-ip-172-31-40-25.out
ubuntu@ip-172-31-40-25:~$
```

```
ubuntu@ip-172-31-84-118:~$ start-worker.sh spark://54.236.77.201:7077
starting org.apache.spark.deploy.worker.Worker, logging to /opt/spark/logs/spark-ubuntu-org.apache.spark.deploy.worker.Worker-1-ip-172-31-84-118.out
ubuntu@ip-172-31-84-118:~$
```

```
ubuntu@ip-172-31-84-134:~$ start-worker.sh spark://54.236.77.201:7077
starting org.apache.spark.deploy.worker.Worker, logging to /opt/spark/logs/spark-ubuntu-org.apache.spark.deploy.worker.Worker-1-ip-172-31-84-134.out
ubuntu@ip-172-31-84-134:~$
```

Verify the web ui using ip address of master and 8080 port i.e., master\_ip:8080

Worker Id	Address	State	Cores	Memory	Resources
worker-20250802121737-172.31.84.118-43811	172.31.84.118:43811	ALIVE	2 (0 Used)	2.8 GiB (0.0 B Used)	
worker-20250802121808-172.31.84.134-42085	172.31.84.134:42085	ALIVE	2 (0 Used)	2.8 GiB (0.0 B Used)	

You should get above ui with Workers listing 2 workers

## Do the below tasks only in master node

In the master node. Let us install pyspark globally so that our python script can recognize the code we have written.

Initially install python3-pip using command

```
Command pip not found, but can be installed with:
sudo apt install python3-pip
ubuntu@ip-172-31-40-25:~$ sudo apt install python3-pip
Reading package lists... Done
Building dependency tree... Done
```

```
sudo apt install python3-pip
```

Then install pyspark globally using below command

```
No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-40-25:~$ pip install --break-system-packages pyspark==3.3.1
Defaulting to user installation because normal site-packages is not writeable
Collecting pyspark==3.3.1
```

```
pip install --break-system-packages pyspark==3.3.1
```

Now let us clone our etl from github in our master node. This is where we will be executing all our codes. Make sure you change directory to your home if you are not already there using  
cd ~  
before running the command to clone

```
ubuntu@ip-172-31-40-25:/opt/spark/jars$ cd ~
ubuntu@ip-172-31-40-25:~$ git clone https://github.com/neotheobserver/spark-etl.git etl
Cloning into 'etl'...
remote: Enumerating objects: 18, done.
remote: Counting objects: 100% (18/18), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 18 (delta 2), reused 18 (delta 2), pack-reused 0 (from 0)
Receiving objects: 100% (18/18), 4.86 KiB | 1.21 MiB/s, done.
Resolving deltas: 100% (2/2), done.
ubuntu@ip-172-31-40-25:~$
```

For those who have been following properly use your own repo and work on it. For those who have not been able to follow can use my repo from below but make sure you do your own work as well. Notice etl after “git clone github\_url “ It is the name of the directory where the repository will be cloned.

```
git clone https://github.com/neotheobserver/spark-etl.git etl
```

Change directory to etl (where we cloned our code)

```
Resolving deltas: 100% (4/4), done.
ubuntu@ip-172-31-40-25:~$ cd etl
ubuntu@ip-172-31-40-25:~/etl$ python3 transform/execute.py s3://spotifydatasummerclass/extract/ s3://spotify
```

Try to execute the transform code

```
ubuntu@ip-172-31-40-25:~/etl$ python3 transform/execute.py
2025-08-02 12:09:23,875 [MainThread ] [CRITI] Usage: python3 transform/execute.py <input_dir> <output_dir>
_e_mem _e_core _e_inst
```

We get an error as we should because we did not pass the required input variables

```
_mem _e_mem _e_core _e_inst
ubuntu@ip-172-31-40-25:~/etl$ python3 transform/execute.py s3://spotifydatasummerclass/extract/ s3://spotifydatasummerclass/transform/
2025-08-02 12:10:46,278 [MainThread ] [INFO ] Starting spark session
```

Now let us pass the required parameters, input\_directory, output\_directory, master\_ip, driver\_mem  
2g means 2 gb, executor mem, executor core, no of executor

```
ubuntu@ip-172-31-40-25:~$ cd etl
ubuntu@ip-172-31-40-25:~/etl$ python3 transform/execute.py s3://spotifydatasummerclass/extract/ s3://spotifydatasummerclass/transform/ 54.236.77.201 2g 4g 2 2
2025-08-02 12:21:50,918 [MainThread ] [INFO ] Starting spark session
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
25/08/02 12:21:55 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
2025-08-02 12:21:58,793 [MainThread ] [INFO ] Loading data in dataframes
25/08/02 12:22:01 WARN FileStreamSink: Assume no metadata directory. Error while looking for metadata directory in the path: s3://spotifydatasummerclass/extract/artists.csv.
org.apache.hadoop.fs.UnsupportedFileSystemException: No FileSystem for scheme "s3"
    at org.apache.hadoop.fs.FileSystem.getFileSystemClass(FileSystem.java:3443)
    at org.apache.hadoop.fs.FileSystem.createFileSystem(FileSystem.java:3466)
    at org.apache.hadoop.fs.FileSystem.access$300(FileSystem.java:174)
    at org.apache.hadoop.fs.FileSystem$Cache.getInternal(FileSystem.java:3574)
    at org.apache.hadoop.fs.FileSystem$Cache.get(FileSystem.java:3521)
```

This error was expected. It says it cannot find s3. This is because spark needs s3a uri unless you change the config. Let us run the code again after modify the input parameters value. Make sure you use your own bucket names, master ip and not copy mine as it will be different for everyone.

For executor memory pass a little less then what you have total as there is some overhead that needs to be accounted for

```
ubuntu@ip-172-31-40-25:~/etl$ python3 transform/execute.py s3a://spotifydatasummerclass/extract/ s3a://spotifydatasummerclass/transform/ 54.236.77.201 2g 2g 2 2
2025-08-02 12:44:01,340 [MainThread ] [INFO ] Starting spark session
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
25/08/02 12:44:06 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
2025-08-02 12:44:09,245 [MainThread ] [INFO ] Loading data in dataframes
25/08/02 12:44:11 WARN MetricsConfig: Cannot locate configuration: tried hadoop-metrics2-s3a-file-system.properties,hadoop-metrics2.properties
25/08/02 12:45:01 WARN package: Truncated the string representation of a plan since it was too large. This behavior can be adjusted by setting 'spark.sql.debug.maxToStringFields'.
2025-08-02 12:45:14,809 [MainThread ] [INFO ] Stage 1: Cleaned data saved
2025-08-02 12:45:14,809 [MainThread ] [INFO ] Creation of master table started
2025-08-02 12:45:40,590 [MainThread ] [INFO ] Stage 2: Master table saved
2025-08-02 12:45:40,590 [MainThread ] [INFO ] Creation of individual tables started
2025-08-02 12:46:19,767 [MainThread ] [INFO ] Stage 3: Query-optimized tables saved
2025-08-02 12:46:19,768 [MainThread ] [INFO ] Transformation pipeline completed
2025-08-02 12:46:19,768 [MainThread ] [INFO ] Total time taken 0 hours, 2 minutes, 18 seconds
2025-08-02 12:46:19,768 [MainThread ] [INFO ] Closing down clientserver connection
ubuntu@ip-172-31-40-25:~/etl$
```

python3 transform/execute.py s3a://spotifydatasummerclass/extract/  
s3a://spotifydatasummerclass/transform/ 54.236.77.201 2g 2g 2 2

Notice that the memory supplied for executor is only 2g even though our total memory is 4gb this is because there are several overheads for setup, for storage and so on. So only a portion of the memory can be used. If you created instance using different instance having different memory and core modify the input parameter accordingly

You should be able to access the spark-ui as you did for local using master\_ip:4040

The screenshot shows the Apache Spark UI Executors page. At the top, there are tabs for Jobs, Stages, Storage, Environment, Executors (selected), and SQL / DataFrame. The URL is http://54.236.77.201:4040/executors/. On the right, it says SpotifyDataTransform application.

## Executors

Show Additional Metrics

### Summary

	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write	Excluded
Active(3)	0	299.3 KiB / 3 GiB	0.0 B	4	4	0	8	12	1.8 min (2 s)	123.8 MiB	75 MiB	75 MiB	0
Dead(0)	0	0.0 B / 0.0 B	0.0 B	0	0	0	0	0	0.0 ms (0.0 ms)	0.0 B	0.0 B	0.0 B	0
Total(3)	0	299.3 KiB / 3 GiB	0.0 B	4	4	0	8	12	1.8 min (2 s)	123.8 MiB	75 MiB	75 MiB	0

### Executors

Show 20 entries Search:

Executor ID	Address	Status	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)
-------------	---------	--------	------------	----------------	-----------	-------	--------------	--------------	----------------	-------------	---------------------

After the job has completed let us go to our s3 bucket and verify whether new files have been created in respective folder or not

The screenshot shows the AWS S3 console. The path is Amazon S3 > Buckets > spotifydatasummerclass > transform/ > stage2/ > master\_table/. There is a 'Copy S3 URI' button.

### master\_table/

[Objects](#) [Properties](#)

**Objects (6)**

[Copy S3 URI](#) [Copy URL](#) [Download](#) [Open](#) [Delete](#) [Actions](#) [Create folder](#) [Upload](#)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 Inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Name	Type	Last modified	Size	Storage class
<a href="#">_SUCCESS</a>	-	August 2, 2025, 18:30:41 (UTC+05:45)	0 B	Standard
<a href="#">part-00000-4bad2949-4d95-4689-8671-9d35b887373e-c000.snappy.parquet</a>	parquet	August 2, 2025, 18:30:40 (UTC+05:45)	18.9 MB	Standard
<a href="#">part-00001-4bad2949-4d95-4689-8671-9d35b887373e-c001.snappy.parquet</a>	parquet	August 2, 2025, 18:30:39 (UTC+05:45)	16.4 MB	Standard

Congratulation! You have ran your first ETL job using AWS based cloud infrastructure