

Machine Learning Engineering Bootcamp

Created by  WeCloudData

1. Getting Started with AWS (EC2, S3)

1.1 - Launch an EC2 Instance

1.1.1 - Sign in to AWS

Click the link below to sign in to AWS console

- <https://weclouddata.signin.aws.amazon.com/console>

An AWS account has been prepared for this lab, here's the log in credential:

- IAM user name:
- Password:



Sign in as IAM user

Account ID (12 digits) or account alias

weclouddata

IAM user name

smu-academy

Password

.....

Sign in

[Sign in using root user email](#)

[Forgot password?](#)

After successful signin, you will see the AWS Management Console

1.1.2 - Create a Security Group

A security group allows us to control the IP addresses that can access services on specific ports.

In the EC2 console, click **Network & Security -> Security Group**

Name	Security group ID	Security group name	VPC ID	Description
-	sg-0a2a2d6e	default	vpc-6833f40c	default VPC security gr...

Click **Create Security Group** and enter a group name.

- Click **Add Rule** in **Inbound Rules** section to add an SSH rule
 - **Port Range** set to 22
 - **Source** set to your IP address or **anywhere** (less secure)

The screenshot shows the 'Create security group' wizard. In the 'Basic details' section, the security group name is 'demo' and the description is 'demo security group'. It is associated with VPC 'vpc-6833f40c'. In the 'Inbound rules' section, a rule is being configured with Type 'Custom TCP', Protocol 'TCP', Port range '22', and Source 'Anywhere'. The 'Add rule' button is visible.

Click to create the security group and confirm in the next step

The screenshot shows the 'sg-0cdde928534c8df0b - demo' security group details. It has a security group name of 'demo', ID 'sg-0cdde928534c8df0b', description 'demo security group', and VPC ID 'vpc-6833f40c'. It has 2 inbound rules and 1 outbound rule. The 'Inbound rules' tab is selected, showing two entries: one for SSH (TCP port 22) from 0.0.0.0/0 and another for SSH (TCP port 22) from ::/0.

1.1.3 - Create a Key Pair

An EC2 key pair is a pair of security private/public keys. Amazon keep the public key while the user download the private key. When a user use SSH to connect to the remote EC2 instance (server), the private key will be uploaded to AWS and get compared to the public key that locates on the EC2 instance. User access will be authenticated if the two keys match.

In the EC2 console, click **Network & Security -> Key Pairs**

The screenshot shows the AWS EC2 service dashboard under the 'Key pairs' section. The left sidebar lists various EC2 features like Instances, Images, and Network & Security. The main area is titled 'Key pairs' with a search bar and a table header for 'Name', 'Fingerprint', and 'ID'. A message at the bottom says 'No key pairs to display'.

Click **Create Key Pair** button and enter the keypair name, then create the key pair

- After the create is created successfully, a `.pem` file will be download to your local drive automatically.
- Please put this keypair in a `~/.ssh/` folder. In linux/mac os, try `~/ssh/`

The dialog box is titled 'Create key pair'. It contains a 'Key pair' section with a description of what it is. Below it, there's a 'Name' input field containing 'demo', a 'File format' section with 'pem' selected, and a 'Tags (Optional)' section with an 'Add tag' button. At the bottom are 'Cancel' and 'Create key pair' buttons.

Copy the downloaded `demo.pem` to `~/ssh/` and change the permission of this private key file to 400.

```
chmod 400 ~/.ssh/demo.pem
```

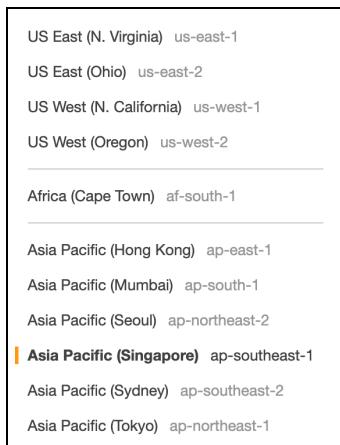
```
(base) ➔ Big Data Workshop cp ~/Downloads/demo.pem ~/.ssh/
(base) ➔ Big Data Workshop chmod 400 ~/.ssh/demo.pem
```

1.1.4 - Launch an EC2 Instance

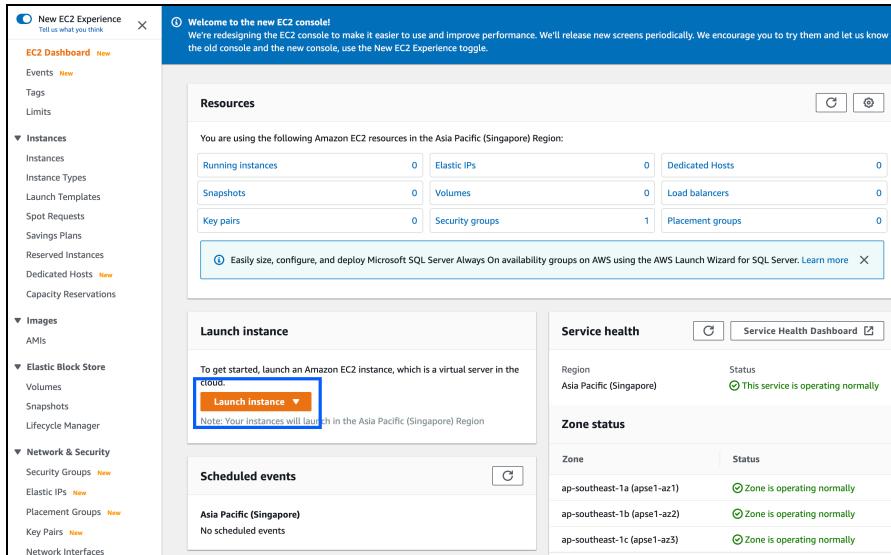
It takes 7 steps to launch an EC2 instance in the AWS console.

1. Before your launch an instance, make sure you're launching it to the right region. In the EC2 console, click the top right corner on **Region** tab, and select the region you want to use.

For this workshop, we will stick to the **Asia-Pacific (Singapore)** region



2. In the EC2 console, click **Launch Instance** and choose AMI



You will see the **Step 1** page

Step 1: Choose an Amazon Machine Image (AMI)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Quick Start

- My AMIs
- AWS Marketplace
- Community AMIs
- Free tier only

AMI Name	Description	Root device type	Virtualization type	ENI Enabled	Select
Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-0cd31be676780afa7 (64-bit x86)	Amazon Linux 2 comes with five years support. It provides Linux kernel 4.14 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through Extras.	sts	hvm	Yes	<input checked="" type="radio"/> 64-bit (x86) <input type="radio"/> 64-bit (Arm)
Amazon Linux 2018.03.0 (HVM), SSD Volume Type - ami-02d6387a2c625404f	The Amazon Linux AMI is an EBS-backed, AWS-supported image. The default image includes AWS command line tools, Python, Ruby, Perl, and Java. The repositories include Docker, PHP, MySQL, PostgreSQL, and other packages.	sts	hvm	Yes	<input type="radio"/> Select <input checked="" type="radio"/> 64-bit (x86) <input type="radio"/> 64-bit (Arm)
Red Hat Enterprise Linux 8 (HVM), SSD Volume Type - ami-02b6d9703a89265e9 (64-bit x86)	Red Hat Enterprise Linux version 8 (HVM), EBS General Purpose (SSD) Volume Type	sts	hvm	Yes	<input type="radio"/> Select <input checked="" type="radio"/> 64-bit (x86) <input type="radio"/> 64-bit (Arm)
SUSE Linux Enterprise Server 15 SP2 (HVM), SSD Volume Type - ami-0d42c4e60c5c4135 (64-bit x86)	SUSE Linux Enterprise Server 15 Service Pack 2 (HVM), EBS General Purpose (SSD) Volume Type. Public Cloud, Advanced Systems Management, Web and Scripting, and Legacy modules enabled.	sts	hvm	Yes	<input type="radio"/> Select <input checked="" type="radio"/> 64-bit (x86) <input type="radio"/> 64-bit (Arm)
Ubuntu Server 18.04 LTS (HVM), SSD Volume Type - ami-0007cf57783ff7e10 (64-bit x86)	Ubuntu Server 18.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).	sts	hvm	Yes	<input type="radio"/> Select <input checked="" type="radio"/> 64-bit (x86) <input type="radio"/> 64-bit (Arm)

Choose the Amazon Linux 2 AMI from the top of the page

Step 1: Choose an Amazon Machine Image (AMI)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Quick Start

- My AMIs
- AWS Marketplace
- Community AMIs
- Free tier only

AMI Name	Description	Root device type	Virtualization type	ENI Enabled	Select
Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-0cd31be676780afa7 (64-bit x86)	Amazon Linux 2 comes with five years support. It provides Linux kernel 4.14 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through Extras.	sts	hvm	Yes	<input checked="" type="radio"/> 64-bit (x86) <input type="radio"/> 64-bit (Arm)
Amazon Linux 2018.03.0 (HVM), SSD Volume Type - ami-02d6387a2c625404f	The Amazon Linux AMI is an EBS-backed, AWS-supported image. The default image includes AWS command line tools, Python, Ruby, Perl, and Java. The repositories include Docker, PHP, MySQL, PostgreSQL, and other packages.	sts	hvm	Yes	<input type="radio"/> Select <input checked="" type="radio"/> 64-bit (x86) <input type="radio"/> 64-bit (Arm)
Red Hat Enterprise Linux 8 (HVM), SSD Volume Type - ami-02b6d9703a89265e9 (64-bit x86)	Red Hat Enterprise Linux version 8 (HVM), EBS General Purpose (SSD) Volume Type	sts	hvm	Yes	<input type="radio"/> Select <input checked="" type="radio"/> 64-bit (x86) <input type="radio"/> 64-bit (Arm)
SUSE Linux Enterprise Server 15 SP2 (HVM), SSD Volume Type - ami-0d42c4e60c5c4135 (64-bit x86)	SUSE Linux Enterprise Server 15 Service Pack 2 (HVM), EBS General Purpose (SSD) Volume Type. Public Cloud, Advanced Systems Management, Web and Scripting, and Legacy modules enabled.	sts	hvm	Yes	<input type="radio"/> Select <input checked="" type="radio"/> 64-bit (x86) <input type="radio"/> 64-bit (Arm)
Ubuntu Server 18.04 LTS (HVM), SSD Volume Type - ami-0007cf57783ff7e10 (64-bit x86)	Ubuntu Server 18.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).	sts	hvm	Yes	<input type="radio"/> Select <input checked="" type="radio"/> 64-bit (x86) <input type="radio"/> 64-bit (Arm)

3. Choose Instance Type

Choose `t2.micro` as the instance type

Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. [Learn more](#) about instance types and how they can meet your computing needs.

Filter by:	All instance types	Current generation	Show/Hide Columns					
Currently selected:	<code>i2.micro</code> (Variable ECUs, 1 vCPU, 2.5 GHz, Intel Xeon Family, 1 GB memory, EBS only)							
	Family	Type	vGPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
	General purpose	<code>t2.nano</code>	1	0.5	EBS only	-	Low to Moderate	Yes
	General purpose	<code>t2.micro</code>	1	1	EBS only	-	Low to Moderate	Yes
	General purpose	<code>t2.small</code>	1	2	EBS only	-	Low to Moderate	Yes
	General purpose	<code>t2.medium</code>	2	4	EBS only	-	Low to Moderate	Yes

4. Configure Instance Details

In this lab, choose the default setting

Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Number of instances	1	Launch into Auto Scaling Group
Purchasing option	<input type="checkbox"/> Request Spot instances	
Network	<code>vpc-6833140c (default)</code>	<input type="checkbox"/> Create new VPC
Subnet	No preference (default subnet in any Availability Zone)	<input type="checkbox"/> Create new subnet
Auto-assign Public IP	<input type="checkbox"/> Use subnet setting (Enable)	
Placement group	<input type="checkbox"/> Add instance to placement group	
Capacity Reservation	<input type="checkbox"/> Open	
IAM role	None	<input type="checkbox"/> Create new IAM role
Shutdown behavior	<code>Stop</code>	
Stop - Hibernate behavior	<input type="checkbox"/> Enable hibernation as an additional stop behavior	
Enable termination protection	<input type="checkbox"/> Protect against accidental termination	
Monitoring	<input type="checkbox"/> Enable CloudWatch detailed monitoring	
Tenancy	<code>Shared - Run a shared hardware instance</code>	
T2/T3 Unlimited	<input type="checkbox"/> Enable	

5. Add Storage

On the next page, add storage (e.g., choose the default 8GB and click to the next step)

Step 4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. [Learn more](#) about storage options in Amazon EC2.

Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MiB/s)	Delete on Termination	Encryption
Root	/dev/xvda	snap-07f777fc29b1061b	8	General Purpose SSD (gp2)	100 / 3000	N/A	<input checked="" type="checkbox"/>	Not Encrypted

[Add New Volume](#)

6. Add Tag

You can add a tag for this lab or leave it empty

Step 5: Add Tags

A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver. A copy of a tag can be applied to volumes, instances or both. Tags will be applied to all instances and volumes. [Learn more](#) about tagging your Amazon EC2 resources.

Key	Value	Instances	Volumes
workshop	demo	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

[Add another tag](#) (Up to 50 tags maximum)

7. Choose the Security Group

Choose the security group **demo** that you created earlier from the existing security group list

Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. Learn more about Amazon EC2 security groups.

Assign a security group: Create a new security group Select an existing security group

Security Group ID	Name	Description	Actions
sg-02a2a2dfe	default	default VPC security group	Copy to new Copy to new
sg-0cdde928534c8df0b	demo	demo security group	Copy to new
sg-04331223e4cedde94	smu-academy	security group for smu academy big data workshop	Copy to new

Inbound rules for sg-0cdde928534c8df0b (Selected security groups: sg-0cdde928534c8df0b)

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	0.0.0.0/0	
SSH	TCP	22	::/0	

8. Review all the configurations

Review the configurations and if everything looks ok, click **Launch**

Step 7: Review Instance Launch
Please review your instance launch details. You can go back to edit changes for each section. Click Launch to assign a key pair to your instance and complete the launch process.

AMI Details

Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-0cd31bd676780fa7a
Free tier
Amazon Linux 2 comes with five years support. It provides Linux kernel 4.14 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extras.
Root Device Type: ebs
Virtualization type: hvm

Instance Type

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
t2.micro	Variable	1	1	EBS only	-	Low to Moderate

Security Groups

Security Group ID	Name	Description
sg-0cd31bd676780fa7a	demo	demo security group

All selected security groups inbound rules

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	0.0.0.0/0	
SSH	TCP	22	::/0	

Instance Details

[Edit instance details](#)

[Cancel](#) [Previous](#) [Launch](#)

9. Choose a Key Pair

The keypair selected at this step will be associated with this EC2 instance and cannot be changed later. Click **Create Instance** to launch the instance

Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about [removing existing key pairs from a public AMI](#).

I acknowledge that I have access to the selected private key file (demo.pem), and that without this file, I won't be able to log into my instance.

[Cancel](#) [Launch Instances](#)

10. Wait for the instance to be launched

Click **View Instances** to go back to the EC2 instance console

Launch Status

Your instances are now launching
The following instance launches have been initiated: i-0499f2776b1ef5589 [View launch log](#)

Get notified of estimated charges
Create billing alerts to get an email notification when estimated charges on your AWS bill exceed an amount you define (for example, if you exceed the free usage tier).

How to connect to your instances
Your instances are launching, and it may take a few minutes until they are in the running state, when they will be ready for you to use. Usage hours on your new instances will start immediately and continue to accrue until you stop or terminate your instances.
[Click View Instances](#) to monitor your instances' status. Once your instances are in the running state, you can connect to them from the Instances screen. [Find out](#) how to connect to your instances.

Here are some helpful resources to get you started

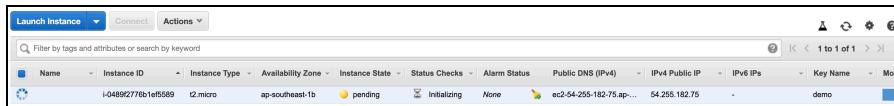
- How to connect to your Linux instance
- Learn about AWS Free Usage Tier
- Amazon EC2 User Guide
- Amazon EC2 Discussion Forum

While your instances are launching you can also

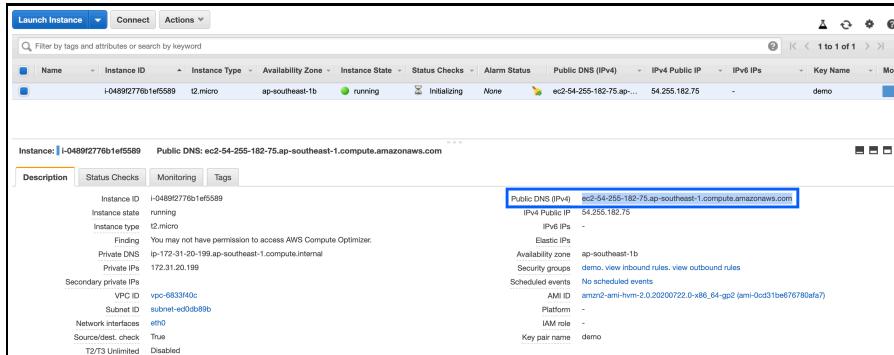
- Create status check alarms to be notified when these instances fail status checks. (Additional charges may apply)
- Create and attach additional EBS volumes (Additional charges may apply)
- Manage security groups

[View Instances](#)

Pending Status shows yellow



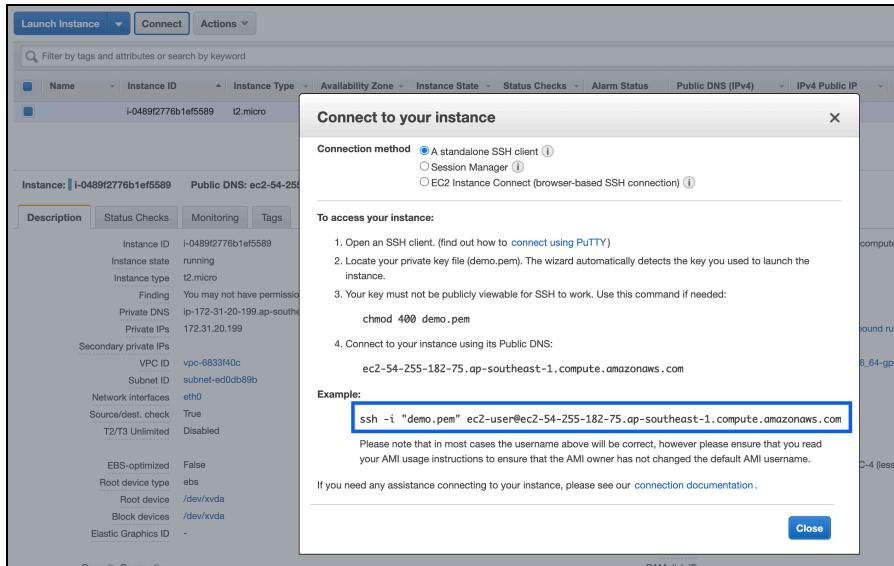
Instance Status will change to **green** and the instance is good to go. Click on the instance name and find the **public dns name** or **public ip**



11. On the EC2 instance page, click the **Connect** button to see the connect options.

In this lab, we will connect via SSH client.

- Windows users can use **gitbash**
- Mac users can use **ssh** in a terminal.



In a terminal, type the following command to connect to the instance via **ssh**

- Note:
 - `demo.pem` can be updated to your own pem file name accordingly
 - public dns name varies. You should have your own ip address

instead

- ec2-user is the default user name for an Amazon Linux AMI

```
ssh -i ~/.ssh/demo.pem ec2-user@ec2-54-255-182-75.ap-southeast-1.compute.amazonaws.com
```

```
(base) → Big Data Workshop ssh -i ~/.ssh/demo.pem ec2-user@ec2-54-255-182-75.ap-southeast-1.compute.amazonaws.com
The authenticity of host "ec2-54-255-182-75.ap-southeast-1.compute.amazonaws.com (54.255.182.75)" can't be established.
ECDSA Key fingerprint is SHA256:19aC9yHx3BwBfqU+6xEUKzMFxEoFPKvYXFrIgogEhNk.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'ec2-54-255-182-75.ap-southeast-1.compute.amazonaws.com,54.255.182.75' (ECDSA) to the list of known hosts.

--| --|-
-| ( -- / Amazon Linux 2 AMI
---\---\---\

https://aws.amazon.com/amazon-linux-2/
4 package(s) needed for security, out of 8 available
Run "sudo yum update" to apply all updates.
```

You will see the welcome message after successful SSH connection.

1.1.5 - Install Python Packages

Reference

- <https://aws.amazon.com/premiumsupport/knowledge-center/ec2-linux-python3-boto3/>

Check if python3 has been installed

```
[ec2-user ~]$ yum list installed | grep -i python3
```

If not, install Python3 on EC2 Linux

```
[ec2-user ~]$ sudo yum install python3 -y
```

```
[ec2-user@ip-172-31-20-199 ~]$ sudo yum install python3 -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-metad
amzn2-core
Resolving Dependencies
--> Running transaction check
--> Package python3.x86_64 0:3.7.8-1.amzn2.0.1 will be installed
--> Processing Dependency: python3-langs(x86-64) = 3.7.8-1.amzn2.0.1 for package: python3-3.7.8-1.amzn2.0.1.x86_64
--> Processing Dependency: python3-setuptools for package: python3-3.7.8-1.amzn2.0.1.x86_64
--> Processing Dependency: python3-pip for package: python3-3.7.8-1.amzn2.0.1.x86_64
--> Processing Dependency: libpython3.7m.so.1.0()(64bit) for package: python3-3.7.8-1.amzn2.0.1.x86_64
--> Running transaction check
--> Package python3-langs.x86_64 0:3.7.8-1.amzn2.0.1 will be installed
--> Package python3-pip.noarch 0:9.0.3-1.amzn2.0.2 will be installed
--> Package python3-setuptools.noarch 0:38.4.0-3.amzn2.0.6 will be installed
--> Finished Dependency Resolution

Dependencies Resolved
=====
Package           Arch      Version          Repository        Size
=====
Installing:
python3           x86_64   3.7.8-1.amzn2.0.1    amzn2-core       72 k
Installing for dependencies:
python3-langs     x86_64   3.7.8-1.amzn2.0.1    amzn2-core       9.2 M
python3-pip        noarch   9.0.3-1.amzn2.0.2    amzn2-core       1.9 M
python3-setuptools noarch   38.4.0-3.amzn2.0.6    amzn2-core       617 k

Transaction Summary
=====
Install 1 Package (+3 Dependent packages)
```

Install Packages

```
[ec2-user ~]$ sudo pip3 install tweepy
[ec2-user ~]$ sudo pip3 install boto3
```

```
[ec2-user@ip-172-31-20-199 ~]$ sudo pip3 install tweepy
WARNING: Running pip install with root privileges is generally not a good idea. Try `pip3 install --user` instead.
Collecting tweepy
  Downloading https://files.pythonhosted.org/packages/bb/7c/99d51f80f3b7b107ebae2634108717362c059a41384e1810d13e2429a81/tweepy-3.9.0-py2.py3-none-any.whl
Collecting requests-oauthlib>=0.7.0 (from tweepy)
  Downloading https://files.pythonhosted.org/packages/a3/12/b92740d845ab62ea4edf04d2f4164d82532b5a0b0b3836d4d4e71c6f3d379/requests_oauthlib-1.3.0-py2.py3-none-any.whl
Requirement already satisfied: six>=1.19.0 in /usr/local/lib/python3.7/site-packages (from tweepy)
Collecting requests[socks]>=2.11.1 (from tweepy)
  Downloading https://files.pythonhosted.org/packages/45/1e/0c169c6a5381e241ba7404532c16a21d8ab72c9bed8bcd4c423954103/requests-2.24.0-py2.py3-none-any.whl (61kB)
  100% [██████████] 71kB 7.4MB/s
Collecting oauthlib>=3.0.0 (from requests-oauthlib>=0.7.0->tweepy)
  Downloading https://files.pythonhosted.org/packages/05/57/ce2e7a8fa7c0afb54a0581b14a65b56e62b5759dbc98e80627142b8a3704/oauthlib-3.1.0-py2.py3-none-any.whl (147kB)
  100% [██████████] 153kB 4.8MB/s
Collecting idna<3,>=2.5 (from requests[socks]>=2.11.1->tweepy)
  Downloading https://files.pythonhosted.org/packages/a2/38/928dce2273eaa564f6f50de919327bf3a0ff91b5babaa8dfa946ff3a8a8/idna-2.18-py2.py3-none-any.whl (58kB)
  100% [██████████] 61kB 8.4MB/s
Collecting certifi>=2017.4.17 (from requests[socks]>=2.11.1->tweepy)
  Downloading https://files.pythonhosted.org/packages/5e/c4/6c4fe722df5343c33226f0b4e0bb042e4dc13483228b4718baf286f86d87/certifi-2020.6.20-py2.py3-none-any.whl (156kB)
  100% [██████████] 163kB 6.1MB/s
Requirement already satisfied: urllib3!=1.25.0,>=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/site-packages (from tweepy)
Collecting chardet<4,>=3.0.2 (from requests[socks]>=2.11.1->tweepy)
  Downloading https://files.pythonhosted.org/packages/bc/a9/01ffebfb562e4274b6487b4b1ddec7ca55ec751b22e4c51f14098443b8/chardet-3.0.4-py2.py3-none-any.whl (133kB)
  100% [██████████] 143kB 7.5MB/s
Collecting PySocks!=1.5.7,>=1.5.6; extra == "socks" (from requests[socks]>=2.11.1->tweepy)
  Downloading https://files.pythonhosted.org/packages/8d/59/b4572118e098ac8e46e3991dd0f2d85403ce8baad9ec79373ed6badaf9/PySocks-1.7.1-py3-none-any.whl
Installing collected packages: idna, certifi, chardet, PySocks, requests, oauthlib, requests-oauthlib, tweepy
Successfully installed PySocks-1.7.1 certifi-2020.6.20 chardet-3.0.4 idna-2.18 oauthlib-3.1.0 requests-2.24.0 requests-oauthlib-1.3.0 tweepy-3.9.0
[ec2-user@ip-172-31-20-199 ~]$
```

1.2 - Working with S3 via AWS CLI

AWS supports several APIs to work with different services. If we want to work with the object store S3, we will either use the `AWS CLI` (command line) API or the `boto3` python API.

1.2.1 - Set up AWS Credential

Install AWSCLI on EC2 instance (or your laptop locally)

Make sure you're SSH connected to EC2 instance

```
sudo pip3 install awscli
```

Configure AWS Credential on EC2 (or your laptop locally)

To have programmatical access to AWS services, you will need to have credentials, a set of keys/secret. You can configure the credentials from the command line. Please choose the AWS region that your instructor asks you to use. It's good to use the same region as the instructor because she/he may share some data with you during the course.

```
aws configure
```

Credentials for this lab (update accordingly):

- **AWS Access Key ID:**
- **AWS Secret Access Key:**
- **Default region name:** us-east-1
- **Default output format:** json

You're all set.

1.2.2 - Working with S3 via AWS CLI

The following s3 exercises can be done from the EC2 instance or your local laptop as long as the credentials are set up properly.

The demo below uses EC2 instance. Make sure you are connected to the instance via ssh.

Note that AWS S3 bucket name needs to be unique. So for this exercise, please switch the weclouddata-demo with your own bucket name. You won't be able to access the bucket created by the instructor unless it's made accessible to you. The weclouddata s3 bucket has some sample dataset and is made public to all.

Make a bucket

```
aws s3 mb s3://weclouddata-demo
```

List buckets

```
aws s3 ls s3://weclouddata/
aws s3 ls s3://weclouddata-demo/
```

Copy file

Upload a file from s3 to ec2 or local

```
touch test.csv
aws s3 cp test.csv s3://weclouddata-demo/
aws s3 ls s3://weclouddata-demo/
```

Download a file from local/ec2 to s3

```
aws s3 cp s3://weclouddata-demo/test.csv ./test1.csv
ll
```

Sync folder

```
mkdir test
cd test && touch test2.csv
cd ..
aws sync test s3://weclouddata-demo/test
aws s3 ls s3://weclouddata-demo/test
```

Delete object

```
aws s3 rm s3://weclouddata-demo/test
```

1.2.3 - AWS CLI Exercise

1. Explore the `weclouddata` s3 bucket
2. Download the `tweets_covid19.txt` from the `datasets/social/twitter/` folder
3. Upload a file `test_<yourname>.csv` to the `s3://<your-aws-s3-bucket>/datasets` folder
4. Delete the `test_<yourname>.csv` file.

► *Solution*

1.3 - Working with S3 via Boto3 SDK

Reference

- [Boto3 S3 API Doc](#)

SSH to EC2 instance and run Python3 from command line

```
python3
```

In python prompt

```
import boto3
# import os
```

1.3.1 - Get a `boto3` session

- Option 1: run `aws configure`
- Option 2: set *ACCESSKEY* and *SECRETKEY* in the OS environment variable
- Option 3: hard code (not suggested)

Option 1: create boto3 custom session

```
import boto3
session = boto3.Session()
```

Option 2: environment variable

```
import os
import boto3

ACCESS_KEY = os.getenv('ACCESS_KEY')
SECRET_KEY = os.getenv('SECRET_KEY')

session = boto3.Session(
    aws_access_key_id=ACCESS_KEY,
    aws_secret_access_key=SECRET_KEY
)
```

1.3.2 - Create s3 resource and client

```
s3 = session.resource('s3')
s3_client = session.client('s3')
```

1.3.3 - Working with s3 client

- client.list_buckets()
- client.list_objects()
- client.create_bucket()
- client.delete_bucket()
- client.put_object()
- client.copy()
- client.copy_object()
- client.delete_object()
- client.delete_objects()
- client.download_file()
- client.download_fileobj()
- upload_file()
- upload_fileobj()

create a new bucket ()

```
response = s3_client.create_bucket(
    Bucket='weclouddata-demo-bucket',
    CreateBucketConfiguration={'LocationConstraint': 'ap-southeast-1'}
)

print(response)
```

list all buckets

```
for key in s3_client.list_buckets()['Buckets']:
    print(key['Name'])
```

list all objects (including folders) in a bucket/folder

```
objects = s3_client.list_objects(  
    Bucket='weclouddata',  
    Prefix='datasets/social'  
)  
  
for key in objects['Contents']:  
    print(key['Key'])
```

Create a folder object in a bucket

```
response = s3_client.put_object(  
    Bucket='weclouddata-demo-bucket',  
    Key=('tmp/demo.csv'))  
  
print(response)
```

List the folder

```
objects = s3_client.list_objects(  
    Bucket='weclouddata-demo-bucket',  
    Prefix='tmp'  
)  
  
for key in objects['Contents']:  
    print(key['Key'])
```

Copy an object from one S3 location to another.

```
copy_source = {
    'Bucket': 'weclouddata-demo-bucket',
    'Key': 'tmp/demo.csv'
}
s3_client.copy(copy_source, 'weclouddata-demo-bucket', 'tmp1/demo1.csv')

objects = s3_client.list_objects(
    Bucket='weclouddata-demo-bucket',
    Prefix='tmp1'
)

for key in objects['Contents']:
    print(key['Key'])
```

Upload a file from local to s3

```
s3_client.upload_file('test.csv', 'weclouddata-demo-bucket', 'tmp/demo.csv')
```

Download from s3

```
s3_client.download_file('weclouddata-demo-bucket', 'tmp1/demo1.csv', 'tmp1/demo1.csv')
```

1.3.4 - Working with s3 session

- Bucket.copy()
- Bucket.create()
- Bucket.delete()
- Bucket.delete_objects()
- Bucket.download_file()
- Bucket.load()
- Bucket.put_object()
- Bucket.upload_file()
- Bucket.upload_fileobj()

list all objects in a bucket using resource

```
bucket = s3.Bucket('weclouddata-demo-bucket')
for obj in bucket.objects.all():
    print(obj.key)
```

upload a local file to s3 bucket

```
s3.Bucket('weclouddata-demo-bucket').Object('tmp/demo2.csv').upload_file(
    'tmp/demo2.csv')
for obj in bucket.objects.all():
    print(obj.key)
```

delete the bucket (need to delete all objects first)

```
bucket = s3.Bucket('weclouddata-demo-bucket')
bucket.delete()
```

delete all objects in a bucket

```
bucket = s3.Bucket('weclouddata-demo-bucket')
bucket.objects.all().delete()
bucket.delete()
```

1.4 - Working with EC2 via Boto3 (Optional)

Launch an ec2 instance

Required arguments:

- AMI ID
- Security Group Id
- VPC Subnet Id

```
import boto3
session = boto3.Session()
client = session.client('ec2', region_name='ap-southeast-1')

response = client.run_instances(
    BlockDeviceMappings=[
        {
            'DeviceName': '/dev/xvda',
            'Ebs': {
                'DeleteOnTermination': True,
                'VolumeSize': 8,
                'VolumeType': 'gp2'
            },
        },
    ],
    ImageId='ami-0cd31be676780afa7',
    InstanceType='t2.micro',
    MaxCount=1,
    MinCount=1,
    Monitoring={
        'Enabled': False
    },
    SecurityGroupIds=[
        'sg-04331223e4cedde94',
    ],
    SubnetId='subnet-ed0db89b'
)
```