

Northeastern University

CS 6620 Cloud Computing

Project #1 [200 points]

EC2 Instances Creation and Operations

Guidelines

Project #1 should be electronically submitted to Canvas by midnight on the due date. A submission link is provided on Canvas.

Assignment Overview

For this project, you will get familiar with the creation of multiple EC2 instances on AWS and conducting different operations on them, via programming as well as using the AWS console. Part I requires programming. Part II is basic preparation via the console, for the next project assignment (i. e., Project 2).

Please post any questions to Piazza and attend TA office hours to address any questions. Get started early.

PART I: Basic EC2 Operations via Programming

- 1) Using an appropriate method you need to set up an EC2 instance
- 2) Modify the EC2 instance initiated from #1 above by assigning a tag to the instance
- 3) Further modify Q1 to reboot the EC2 instance that you had set up at the beginning
- 4) When the entire process of set up has been completed then stop the EC2 instance.
- 5) Start 5 to 6 EC2 instances and assign tags randomly ensuring there are 2 different tags in total
- 6) Find all the instance id that are actively running and report the same
- 7) Filter the EC2 instance based on tags
- 8) Describe the properties of all the EC2 instances
- 9) Update the tag to make sure that there is only one tag that is assigned to all the ec2 instance
- 10) List all the instance id and their tags
- 11) Stop all the instances using the instance id's

PART II: Study the concepts for the next assignment (needs to be done in console)

- 1) Set up the EC2 instance, using t2.micro, add tag to it and it has to be accessed via any ip address
- 2) Explain in brief the process by setting up a routing table
- 3) Set up an Application Load Balancer in the console
- 4) Create a VPC
- 5) Write a 'hello world program' in Python using AWS

Explain using screenshots, code and text as needed to fully answer these questions.

Other notes:

Please see TA announcement regarding project submitting guidelines. A PDF report showing the results via screenshots and brief explanations are required. In addition, submit the code file as appropriate.

Evaluation

Your work will be evaluated on how well the code, results including screenshots and explanations (text) are in conformance to the requirements above.

Executive Summary

Part of your completed assignment submission should be an executive summary containing an “Assignment overview” (1 paragraph, up to about 250 words) explaining what you understand to be the purpose and scope of the assignment and a “technical impression” (1–2 paragraphs, about 200–500 words) describing your experiences while carrying out the assignment. The assignment overview shows how well you understand the assignment; the technical impression section helps to determine what parts of the assignment need clarification, improvement, etc., for the future.

Evaluation

The grade for your executive summary is based on the effort you put into the assignment overview and technical impression. In general, if you put some effort into your writing, you will receive full credit for your executive summary (provided that it is properly formatted and submitted as a plain text file).

Project Deliverables

The following items should be archived together, e.g., placed in a .zip file or tarball file (*.tgz or *.tar.gz), and electronically submitted via the link is provided on the course Moodle page.

1. All source code files plus any additional support code you developed to answer Part 1.
2. Your executive summary and the PDF report answering all questions.

References

<https://aws.amazon.com/ec2/getting-started/>
<https://www.youtube.com/watch?v=bm-6D7iAVS8>

ANS (Qichen An)

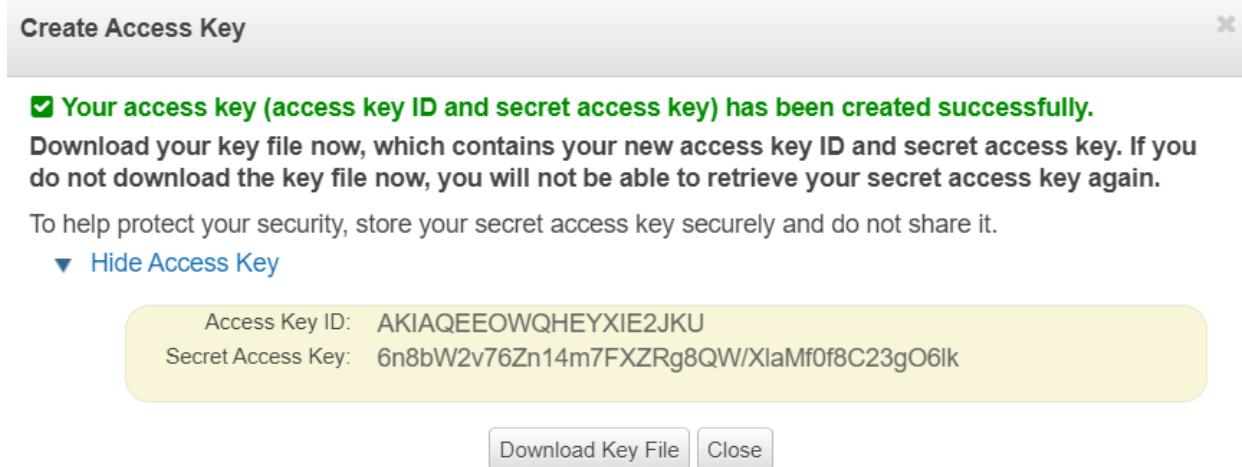
Part 1

1. Using an appropriate method you need to set up an EC2 instance

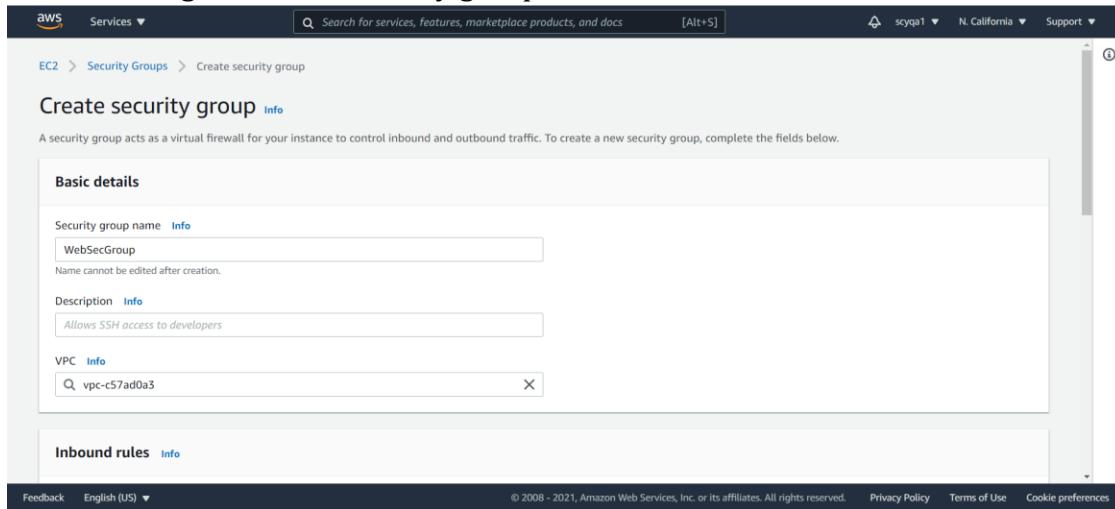
Configure the environment (boto3)

```
(base) D:\Year21-22\CS 6650 Fundamentals of Cloud Computing\CW\project-individual>pip install boto3
Collecting boto3
  Downloading boto3-1.18.50-py3-none-any.whl (131 kB)
    |██████████| 131 kB 1.3 MB/s
Collecting botocore<1.22.0,>=1.21.50
  Downloading botocore-1.21.50-py3-none-any.whl (8.0 MB)
    |██████████| 8.0 MB 1.6 MB/s
Collecting jmespath<1.0.0,>=0.7.1
```

Create Access Key



Before coding, I create a security group in case it is needed for instance creation



Inbound rules

Type	Protocol	Port range	Source	Description - optional
SSH	TCP	22	Anywhere	
HTTP	TCP	80	Anywhere	

Add rule

Outbound rules

Type	Protocol	Port range	Destination	Description - optional
All traffic	All	All	Custom	0.0.0.0/0

Feedback English (US)

Services

Security Groups (1/5)

Name	Security group ID	Security group name	VPC ID	Description	Owner
sg-02df927e032efbeb1	launch-wizard-2	vpc-c57ad0a3	launched-wizard-2 created...	008889205193	
sg-04afe16dafa442508	default	vpc-0ef7df9a77d00bbb2	default VPC security gr...	008889205193	
sg-0aeffd14f430d4848	WebSecGroup	vpc-c57ad0a3	For Project 1	008889205193	
sg-0fc13e8181c00c328	launch-wizard-1	vpc-c57ad0a3	launched-wizard-1 created...	008889205193	
sg-a62825ec	default	vpc-c57ad0a3	default VPC security gr...	008889205193	

Run Reachability Analyzer

EC2 > Security Groups > sg-0aeffd14f430d4848 - WebSecGroup

sg-0aeffd14f430d4848 - WebSecGroup

Details

Security group name	Security group ID	Description	VPC ID
WebSecGroup	sg-0aeffd14f430d4848	For Project 1	vpc-c57ad0a3
Owner	008889205193	Inbound rules count	2 Permission entries
		Outbound rules count	1 Permission entry

Inbound rules (2)

Name	Security group rule...	IP version	Type	Protocol	Port range	Source	Description
-	sgr-0720495de3a05...	IPv6	SSH	TCP	22	::/0	-
-	sgr-0a7def96808c8beba	IPv6	HTTP	TCP	80	::/0	-

Similarly, I create key pair

The screenshot shows the 'Create key pair' wizard. In the 'Name' field, 'Project1KeyPair' is entered. Under 'Key pair type', 'RSA' is selected. Under 'Private key file format', '.ppk' is selected. There are no tags added. The 'Create key pair' button is at the bottom.

The screenshot shows the 'Key pairs' list. One key pair named 'Project1KeyPair' is listed, showing its type as 'rsa' and ID as 'key-09936bb279ba591b2'. The 'Create key pair' button is visible at the top right of the list.

Final Result

```

Run:  ec2-create-instance
D:\IDE\PyCharm\anaconda\ins\python.exe "D:/Year21-22/CS 6650 Fundamentals of Cloud Computing/CW/project-individual1/ec2-create-instance.py"
Connecting to EC2
Launching instance with AMI-ID ami-011996ff98de391d1, with keypair Project1KeyPair,    instance type t2.micro, security group      sg-0aeffd14f438d4848
{'Groups': [], 'Instances': {'amiLaunchIndex': 0, 'ImageId': 'ami-011996ff98de391d1', 'InstanceId': 'i-06ae088a767a1285f', 'InstanceType': 't2.micro', 'KeyName': 'Project1KeyPair'}}
Waiting for instance to be up and running
Status: pending
Status: pending
Status: running

Instance is now running. Instance details are:
Instance Type: t2.micro
Instance State: running
Instance Launch Time: 2021-09-28 21:48:14+00:00
Instance Public DNS: ec2-52-53-239-17.us-west-1.compute.amazonaws.com
Instance Private DNS: ip-172-31-1-49.us-west-1.compute.internal
Instance IP: 52.53.239.17
Instance Private IP: 172.31.1.49

Process finished with exit code 0

```

Log on the aws



2. Modify the EC2 instance initiated from #1 above by assigning a tag to the instance

The screenshot shows the PyCharm IDE interface. The project structure on the left includes files like ans.docx, Cloud Comp Project 1 2021.docx, ec2-addtag.py, ec2-create-instance.py, and ~\$ans.docx. The code editor window contains a Python script named ec2-addtag.py:

```
import boto3

AWS_KEY="AKIAQEEOWQHEYXIE2JKU"
AWS_SECRET="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"
REGION="us-west-1"

print("Connecting to EC2")

ec2 = boto3.client('ec2', aws_access_key_id=AWS_KEY,
                    aws_secret_access_key=AWS_SECRET,
                    region_name=REGION)

ec2.create_tags(Resources=["i-06ae088a767a1285f"], Tags=[{"Key": "name", "Value": "Project1"}]
```

The run tab at the bottom shows the command being executed: D:\IDE\PyCharm\anaconda\ins\python.exe "D:/Year21-22/CS 6650 Fundamentals of Cloud Computing/CW/project-individual1/ec2-addtag.py". The output shows "Connecting to EC2" and "Process finished with exit code 0".

Log on the aws

The screenshot shows the AWS Management Console with the EC2 service selected. The left navigation menu includes EC2 Global View, Events, Tags, Limits, Instances (with 'Instances' highlighted), Images, Elastic Block Store, Network & Security, and Load Balancing. The main content area displays the 'Instance summary for i-06ae088a767a1285f' page. The instance details include:

- Public IPv4 address: 52.53.239.17 | open address
- Private IPv4 address: 172.31.1.49
- Public IPv4 DNS: ec2-52-53-239-17.us-west-1.compute.amazonaws.com | open address
- Instance state: Running
- Instance type: t2.micro
- Subnet ID: subnet-9d6c0d72
- AWS Compute Optimizer finding: Opt-in to AWS Compute Optimizer for recommendations.
- IAM Role: -

The 'Tags' tab is selected in the bottom navigation bar. A table shows one tag: Key: name, Value: Project1. There is a 'Manage tags' button and a small navigation bar for more tags.

3. Further modify Q1 to reboot the EC2 instance that you had set up at the beginning

The screenshot shows the PyCharm IDE interface with the following details:

- File Bar:** File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help, project-individual1 - reboot.py - PyCharm
- Project Tree:** project-individual1 (D:\Year21-22\CS 6650 Fundamentals o...), containing ans.docx, Cloud Comp Project 1 2021.docx, ec2-addtag.py, ec2-create-instance.py, reboot.py, ~Sans.docx, External Libraries, and Scratches and Consoles.
- Code Editor:** The file "reboot.py" is open, showing Python code using the boto3 library to connect to EC2 and reboot an instance. The code includes constants for AWS keys and secret, and a function to reboot an instance by ID.
- Run Tab:** A run configuration named "reboot" is selected, pointing to the command: D:\IDE\PyCharm\anaconda\ins\python.exe "D:/Year21-22/CS 6650 Fundamentals of Cloud Computing/CW/project-individual1/reboot.py". The output shows the process connecting to EC2 and exiting with code 0.
- Bottom Status Bar:** Shows the instance ID i-06ae088a767a1285f, status Running, t2.micro, 2/2 checks passed, No alarms, us-west-1a, ec2-52-53-239-17.us-w..., 52.53.239.17, and disabled WebSecGroup.

4. When the entire process of set up has been completed then stop the EC2 instance.

The screenshot shows the PyCharm IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, and Help. The title bar says "project-individual1 - stop_instance.py - PyCharm". The left sidebar shows a Project tree with "project-individual1" containing files like ans.docx, Cloud Comp Project 1 2021.docx, ec2-addtag.py, ec2-create-instance.py, reboot.py, stop_instance.py, and ~\$ans.docx. Below the project tree are External Libraries and Scratches and Consoles. The main code editor window displays the following Python script:

```
import boto3

AWS_KEY="AKIAQEEOWQHEYXIE2JKU"
AWS_SECRET="6n8bWv76Zn14m7FXZRg8QW/XLaMf0f8C23g06lk"
REGION="us-west-1"

print("Connecting to EC2")

ec2 = boto3.client('ec2', aws_access_key_id=AWS_KEY,
                    aws_secret_access_key=AWS_SECRET,
                    region_name=REGION)

my_ec2_instance_id = ['i-06ae088a767a1285f']# add ec2 instance id but also best stored in file

ec2.stop_instances(InstanceIds=my_ec2_instance_id)
```

The Run tab at the bottom shows the command "D:\IDE\PyCharm\anaconda\ins\python.exe "D:/Year21-22/CS 6650 Fundamentals of Cloud Computing/CW/project-individual1/stop_instance.py"" and the output "Connecting to EC2" followed by "Process finished with exit code 0". The status bar at the bottom indicates "Smart commands execution: Highlighted commands can be interpreted and executed by the IDE in a smart way. // Press Ctrl+Enter to try this, or E... (today 14 13:45 CRLF UTF-8 4 spaces Python 3.8)".

5. Start 5 to 6 EC2 instances and assign tags randomly ensuring there are 2 different tags in total

For this part, I launch 5 EC2 instances, and the first three are with Tag 'Project1_1' and the last two are with Tag 'Project1_2'

The screenshot shows the PyCharm IDE interface with the following details:

- Project Structure:** The project is named "project-individual1". It contains several files: ans.docx, Cloud Comp Project 1 2021.docx, ec2-addtag.py, ec2-create-instance.py, instances_and_tags.py, reboot.py, stop_instance.py, and ~\$ans.docx.
- Code Editor:** The file "instances_and_tags.py" is open. The code uses the Boto3 library to create EC2 instances and add tags. It defines variables like `INSTANCE_TYPE` (t2.micro), `SECURITYGROUP_ID` (sg-0aeffd14f430d4848), and `AMIB_ID` (ami-011996ff98de391d1). It creates three instances with tag 'Project1_1' and two instances with tag 'Project1_2'. The code also handles printing responses and creating tags.
- Run Tab:** The "instances_and_tags" run configuration is selected. The output window shows the process of connecting to EC2 and launching instances. The log output includes instance IDs and their corresponding tags.
- Terminal Tab:** The terminal shows the command "D:\IDE\PyCharm\anaconda\bin\python.exe D:/Year21-22/CS 6650 Fundamentals of Cloud Computing/CW/project-individual1/instances_and_tags.py" being run, followed by the execution results.

Log on the aws

□	-	i-0125aef4d823e31fa	Running	○○	t2.micro	○ Initializing	No alarms	+	us-west-1b	ec2-54-177-149-139.us...	54.177.149.139	-	-	disabled	WebSecGroup
□	-	i-026756ab0d965dad12	Running	○○	t2.micro	○ Initializing	No alarms	+	us-west-1b	ec2-18-144-65-173.us...	18.144.65.173	-	-	disabled	WebSecGroup
□	-	i-095706ab6888eba4a	Running	○○	t2.micro	○ 2/2 checks passed	No alarms	+	us-west-1b	ec2-13-56-59-132.us-w...	13.56.59.132	-	-	disabled	WebSecGroup
□	-	i-018b24b263d573101	Running	○○	t2.micro	○ Initializing	No alarms	+	us-west-1b	ec2-54-219-28-90.us-w...	54.219.28.90	-	-	disabled	WebSecGroup
□	-	i-09c979be50570a04f	Running	○○	t2.micro	○ 2/2 checks passed	No alarms	+	us-west-1b	ec2-54-67-121-160.us...	54.67.121.160	-	-	disabled	WebSecGroup

Project1_1

Tags		Manage tags
<input type="text"/>		< 1 > ⌂
Key	Value	
name	Project1_1	

Tags		Manage tags
<input type="text"/>		< 1 > ⌂
Key	Value	
name	Project1_1	

Tags		Manage tags
<input type="text"/>		< 1 > ⌂
Key	Value	
name	Project1_1	

Project1_2

Tags		Manage tags
<input type="text"/>		< 1 > ⌂
Key	Value	
name	Project1_2	

Tags		Manage tags
<input type="text"/>		< 1 > ⌂
Key	Value	
name	Project1_2	

6. Find all the instance id that are actively running and report the same

```

File Edit View Navigate Code Refactor Run Tools VCS Window Help project-individual1 - find_run.py - PyCharm
project-individual1 find_run.py
Project Project-Individual1 D:\Year21-22\CS 6650 Fundamentals o
  ans.docx
  Cloud Comp Project 1 2021.docx
  ec2-addtag.py
  ec2-create-instance.py
  find_run.py
  instances_and_tags.py
  reboot.py
  stop_instance.py
  ~Sans.docx
  ~WRL2106.tmp
External Libraries
Scratches and Consoles

AWS_KEY = "AKIAQEEOWQHEYXIE2JKU"
AWS_SECRET = "6n8bIW2v76zh14m7FXZRg8QW/XlaMf0f8C23g06lk"
REGION = "us-west-1"
AMI_ID = "ami-011996ff98de391d1"
EC2_KEY_HANDLE = "Project1KeyPair"
INSTANCE_TYPE = "t2.micro"
SECURITY_GROUP_ID = "sg-0aeffd14f430d4848"
print("Connecting to EC2")

ec2 = boto3.resource('ec2', aws_access_key_id=AWS_KEY,
                     aws_secret_access_key=AWS_SECRET,
                     region_name=REGION)

for i in ec2.instances.all():
    if i.state['Name'] == 'running':
        print(i.id)

```

Run: find_run

- D:\IDE\PyCharm\anaconda\ins\python.exe "D:/Year21-22/CS 6650 Fundamentals of Cloud Computing/CW/project-individual1/find_run.py"

Smart commands execution: Highlighted commands can be interpreted and executed by the IDE in a smart way. // Press Ctrl+Enter to try this, or Enter to run the command in the console as usual. // You can turn this behavior off in the settings.

```

D:\IDE\PyCharm\anaconda\ins\python.exe "D:/Year21-22/CS 6650 Fundamentals of Cloud Computing/CW/project-individual1/find_run.py"
Connecting to EC2
i-0125aef4d823e31fa
i-026756a0d965dad12
i-095706ab6888eba4a
i-018b24b263d573101
i-09c979be50570a04f

Process finished with exit code 0

```

Log on the aws

The screenshot shows the AWS Management Console with the EC2 Instances page open. The left sidebar includes links for EC2 Dashboard, EC2 Global View, Events, Tags, Limits, Instances (with sub-links for Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations), Images (AMIs), and Elastic Block Store (Volumes, Snapshots). The main content area displays a table of instances:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
-	i-0516629c50a77cd5d	Terminated	t2.micro	-	No alarms	us-west-1b	-
-	i-0125aef4d823e31fa	Running	t2.micro	2/2 checks passed	No alarms	us-west-1b	ec2-54-177-149-
-	i-026756a0d965dad12	Running	t2.micro	2/2 checks passed	No alarms	us-west-1b	ec2-18-144-65-1
-	i-095706ab6888eba4a	Running	t2.micro	2/2 checks passed	No alarms	us-west-1b	ec2-13-56-59-13
-	i-018b24b263d573101	Running	t2.micro	2/2 checks passed	No alarms	us-west-1b	ec2-54-219-28-9
-	i-09c979be50570a04f	Running	t2.micro	2/2 checks passed	No alarms	us-west-1b	ec2-54-67-121-1
-	i-06ae088a767a1285f	Stopped	t2.micro	-	No alarms	us-west-1a	-

A modal dialog box at the bottom says "Select an instance above".

7. Filter the EC2 instance based on tags

The screenshot shows the PyCharm IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help, and project-individual1 - fileterByTag.py - PyCharm. The Project tool window on the left shows a directory structure for 'project-individual1' containing files like ans.docx, Cloud Comp Project 1 2021.docx, ec2-addtag.py, ec2-create-instance.py, fileterByTag.py, find_run.py, instances_and_tags.py, reboot.py, stop_instance.py, and -sans.docx. The file 'fileterByTag.py' is open in the main editor area, displaying Python code for filtering EC2 instances by tag. The code uses the Boto3 library to connect to EC2 and filter instances based on a specific tag value. The Run tool window at the bottom shows the output of running the script, which lists instances filtered by two tags: 'Project1_1' and 'Project1_2'. The terminal output shows the command run and the resulting list of instance IDs.

```
project-individual1 D:\Year21-22\CS 6650 Fundamentals o
File Edit View Navigate Code Refactor Run Tools VCS Window Help project-individual1 - fileterByTag.py - PyCharm
project-individual1 D:\Year21-22\CS 6650 Fundamentals o
ans.docx
Cloud Comp Project 1 2021.docx
ec2-addtag.py
ec2-create-instance.py
fileterByTag.py
find_run.py
instances_and_tags.py
reboot.py
stop_instance.py
-sans.docx
-WRL2106.tmp
External Libraries
Scratches and Consoles

Run: fileterByTag ×
Connecting to EC2
Instances with Tag "Name=Project1_1":
- Instance ID: i-0125aef4d823e31fa
- Instance ID: i-018b24b263d573101
- Instance ID: i-09c979be50570a04f
Instances with Tag "Name=Project1_2":
- Instance ID: i-026756a0d965dad12
- Instance ID: i-095706ab6888eba4a
Smart commands execution: Highlighted commands can be interpreted and executed by the IDE in a smart way. // Press Ctrl+Enter to try this, or Enter to run the command in the console as usual. // You can turn this behavior off in the settings. (today) 47:27 CRLF UTF-8 4 spaces Python 3.8 Event Log

Run: fileterByTag ×
D:\IDE\PyCharm\anaconda\ins\python.exe "D:/Year21-22/CS 6650 Fundamentals of Cloud Computing/CW/project-individual1/fileterByTag.py"
Connecting to EC2
Instances with Tag "Name=Project1_1":
- Instance ID: i-0125aef4d823e31fa
- Instance ID: i-018b24b263d573101
- Instance ID: i-09c979be50570a04f
Instances with Tag "Name=Project1_2":
- Instance ID: i-026756a0d965dad12
- Instance ID: i-095706ab6888eba4a
Process finished with exit code 0
```

8. Describe the properties of all the EC2 instances

The screenshot shows the PyCharm IDE interface with the file 'describe_property.py' open. The code uses the Boto3 library to describe EC2 instances. It prints the attributes of each instance, including metadata options, instance type, and state.

```
print(f'Instance {INSTANCE_ID} attributes:')

for reservation in response['Reservations']:
    print(json.dumps(
        reservation,
        indent=4,
        default=json_datetime_serializer
    ))
    ec2_resource = boto3.resource('ec2', aws_access_key_id=AWS_KEY,
                                  aws_secret_access_key=AWS_SECRET,
                                  region_name=REGION)
    for i in ec2_resource.instances.all():
        describe(i.id)
```

The output would be in a json format

The screenshot shows the PyCharm IDE interface with the run output for 'describe_property.py'. The output displays the attributes of a single EC2 instance, including its ID, launch index, monitoring status, and network details.

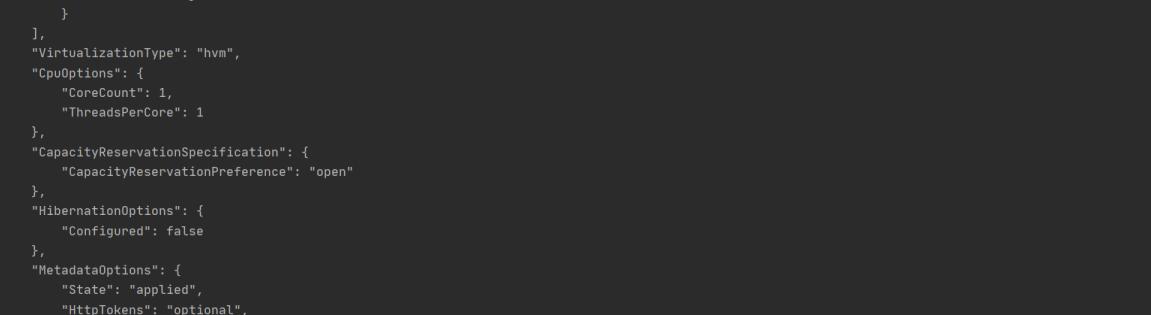
```
D:\IDE\PyCharm\anaconda\ins\python.exe "D:/Year21-22/CS 6650 Fundamentals of Cloud Computing/CW/project-individual1/describe_property.py"
Connecting to EC2
Instance i-0125aef4d823e31fa attributes:
{
    "Groups": [],
    "Instances": [
        {
            "AmiLaunchIndex": 0,
            "ImageId": "ami-011996ff98de391d1",
            "InstanceId": "i-0125aef4d823e31fa",
            "InstanceType": "t2.micro",
            "KeyName": "Project1KeyPair",
            "LaunchTime": "2021-09-28T23:24:21+00:00",
            "Monitoring": {
                "State": "disabled"
            },
            "Placement": {
                "AvailabilityZone": "us-west-1b",
                "GroupName": "",
                "Tenancy": "default"
            },
            "PrivateDnsName": "ip-172-31-31-35.us-west-1.compute.internal",
            "PrivateIpAddress": "172.31.31.35",
            "ProductCodes": [],
            "PublicDnsName": "ec2-54-177-149-139.us-west-1.compute.amazonaws.com",
            "PublicIpAddress": "54.177.149.139",
            "State": {
                "Code": 16,
                "Name": "running"
            },
            "StateTransitionReason": "",
            "SubnetId": "subnet-6ec1a408"
        }
    ]
}
```

```

File Edit View Navigate Code Refactor Run Tools VCS Window Help project-individual1 - describe_property.py - PyCharm
project-individual1 describe_property.py
Project Run: describe_property ×
Run: describe_property ×
    "StateTransitionReason": "",
    "SubnetId": "subnet-6ec1a408",
    "VpcId": "vpc-c57ad0a3",
    "Architecture": "x86_64",
    "BlockDeviceMappings": [
        {
            "DeviceName": "/dev/xvda",
            "Ebs": {
                "AttachTime": "2021-09-28T23:24:22+00:00",
                "DeleteOnTermination": true,
                "Status": "attached",
                "VolumeId": "vol-007660f4fb0344d59"
            }
        },
        ],
    "ClientToken": "3b6b96c4-902e-4841-9298-92650e70176f",
    "EbsOptimized": false,
    "EnaSupport": true,
    "Hypervisor": "xen",
    "NetworkInterfaces": [
        {
            "Association": {
                "IpOwnerId": "amazon",
                "PublicDnsName": "ec2-54-177-149-139.us-west-1.compute.amazonaws.com",
                "PublicIp": "54.177.149.139"
            },
            "Attachment": {
                "AttachTime": "2021-09-28T23:24:21+00:00",
                "AttachmentId": "eni-attach-017b93252e2cfdc46",
                "DeleteOnTermination": true,
                "DeviceIndex": 0,
                "Status": "attached".
            }
        },
        ],
    "Description": "",
    "Groups": [
        {
            "GroupName": "WebSecGroup",
            "GroupId": "sg-0aeffd14f430d4848"
        }
    ],
    "Ipv6Addresses": [],
    "MacAddress": "02:7f:93:99:0c:37",
    "NetworkInterfaceId": "eni-01da7b73c384116f6",
    "OwnerId": "008889205193",
    "PrivateDnsName": "ip-172-31-35.us-west-1.compute.internal",
    "PrivateIpAddress": "172.31.31.35",
    "PrivateIpAddresses": [
        {
            "Association": {
                "IpOwnerId": "amazon",
                "PublicDnsName": "ec2-54-177-149-139.us-west-1.compute.amazonaws.com",
                "PublicIp": "54.177.149.139"
            },
            "Primary": true,
            "PrivateDnsName": "ip-172-31-31-35.us-west-1.compute.internal",
            "PrivateIpAddress": "172.31.31.35"
        }
    ],
    "SourceDestCheck": true,
    "Status": "in-use",
    "SubnetId": "subnet-6ec1a408",
    "VpcId": "vpc-c57ad0a3",
    "InterfaceType": "interface"
}

```

Smart commands execution: Highlighted commands can be interpreted and executed by the IDE in a smart way. // Press Ctrl+Enter to try this, or Enter to run the command in the console as usual. // You can turn this behavior off in the settings.



The screenshot shows the PyCharm IDE interface with the 'File', 'Edit', 'View', 'Navigate', 'Code', 'Refactor', 'Run', 'Tools', 'VCS', 'Window', and 'Help' menus at the top. The title bar displays 'project-individual1 - describe_property.py - PyCharm'. The main editor window contains Python code for describing an AWS instance's properties. The code includes imports for 'boto3', 'logging', and 'argparse', and defines functions for 'get_instance_by_id', 'get_instance_by_name', and 'main'. The 'main' function takes command-line arguments for instance ID or name, region, and output format (json or yaml). It uses the AWS SDK to get the instance details and prints them to the console.

```
File Edit View Navigate Code Refactor Run Tools VCS Window Help project-individual1 - describe_property.py - PyCharm
project-individual1 describe_property.py
Project ▾ Run: describe_property ×
File Edit View Navigate Code Refactor Run Tools VCS Window Help project-individual1 - describe_property.py - PyCharm
Project Structure
Run: describe_property ×


```

 "Value": "Project_1_1"
 },
],
 "VirtualizationType": "hvm",
 "CpuOptions": {
 "CoreCount": 1,
 "ThreadsPerCore": 1
 },
 "CapacityReservationSpecification": {
 "CapacityReservationPreference": "open"
 },
 "HibernationOptions": {
 "Configured": false
 },
 "MetadataOptions": {
 "State": "applied",
 "HttpTokens": "optional",
 "HttpPutResponseHopLimit": 1,
 "HttpEndpoint": "enabled",
 "HttpProtocolIpv6": "disabled"
 },
 "EnclaveOptions": {
 "Enabled": false
 }
}
],
"OwnerId": "008889205193",
"ReservationId": "r-0eff7b56d67066adc"
}
Instance i-026756a0d965dad12 attributes:
{
 "Groups": []
}

```


Event Log
Smart commands execution: Highlighted commands can be interpreted and executed by the IDE in a smart way. // Press Ctrl+Enter to try this, or Enter to run the command in the console as usual. // You can turn this behavior off in Settings | Tools | Smart Keys.

```

As you can see from the progress bar on the right side of the terminal, there are too many properties, so that I don't include all the screenshots.

9. Update the tag to make sure that there is only one tag that is assigned to all the ec2 instance

The screenshot shows the PyCharm IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help, and project-individual1 - update_tag.py - PyCharm. The left sidebar shows a Project structure with files like b-instance.py, ec2-addtag.py, reboot.py, stop_instance.py, instances_and_tags.py, find_run.py, filterByTag.py, describe_property.py, and update_tag.py. The main code editor window displays Python code for updating EC2 instance tags:

```
response = ec2.describe_instances()
for each_res in response['Reservations']:
    for each_inst in each_res['Instances']:
        # I set this matchhost as a flag as I iterate through each host.
        matchhost = True
        # On each host, I iterate through all the tags on the host.
        for each_tag in each_inst['Tags']:
            if each_tag['Key'] == 'name':
                if each_tag['Value'] != 'Project1':
                    matchhost = False
                    break
        if matchhost:
            response = ec2.create_tags(
                Resources=[each_inst['InstanceId']],
                Tags=[{
                    'Key': 'name',
                    'Value': 'Project1'
                }]
)
```

The Run tab at the bottom shows the command: D:/IDE/PyCharm/anaconda/bin/python.exe "D:/Year21-22/CS 6650 Fundamentals of Cloud Computing/CW/project-individual1/update_tag.py". The status message indicates "Process finished with exit code 0".

By the function of 'create_tags', all tags could be updated to new tag 'Project1'

The screenshot shows the AWS EC2 Instances page. The left sidebar includes New EC2 Experience, EC2 Dashboard, EC2 Global View, Events, Tags, Limits, Instances (selected), Instances (New), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances (New), Dedicated Hosts, Capacity Reservations, Images (AMIs), and Elastic Block Store (Volumes, Snapshots). The main content area shows a table titled "Instances (1/6) Info" with columns: Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IPv4 DNS. There are six instances listed, all in a "Running" state. An "Actions" dropdown and a "Launch instances" button are visible. Below the table, a modal window for "Instance: i-0125aef4d823e31fa" is open, showing the "Tags" tab with a single tag: name: Project1.

Three screenshots of the AWS EC2 Instances page showing the selection of specific instances for modification.

Screenshot 1: The first instance, i-026756a0d965dad12, is selected. Its details are shown in the modal window, including its status as Running, type t2.micro, and a single tag named "name" with the value "Project1".

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
-	i-0125aef4db23e31fa	Running	t2.micro	2/2 checks passed	No alarms	us-west-1b	ec2-54-177-149-
<input checked="" type="checkbox"/>	i-026756a0d965dad12	Running	t2.micro	2/2 checks passed	No alarms	us-west-1b	ec2-18-144-65-1
-	i-095706ab6888eba4a	Running	t2.micro	2/2 checks passed	No alarms	us-west-1b	ec2-13-56-59-13
-	i-018b24b263d573101	Running	t2.micro	2/2 checks passed	No alarms	us-west-1b	ec2-54-219-28-9
-	i-09c979be50570a04f	Running	t2.micro	2/2 checks passed	No alarms	us-west-1b	ec2-54-67-121-1
-	i-06ae088a767a1285f	Stopped	t2.micro	-	No alarms	us-west-1a	-

Screenshot 2: The second instance, i-095706ab6888eba4a, is selected. Its details are shown in the modal window, including its status as Running, type t2.micro, and a single tag named "name" with the value "Project1".

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
-	i-0125aef4db23e31fa	Running	t2.micro	2/2 checks passed	No alarms	us-west-1b	ec2-54-177-149-
<input checked="" type="checkbox"/>	i-026756a0d965dad12	Running	t2.micro	2/2 checks passed	No alarms	us-west-1b	ec2-18-144-65-1
<input checked="" type="checkbox"/>	i-095706ab6888eba4a	Running	t2.micro	2/2 checks passed	No alarms	us-west-1b	ec2-13-56-59-13
-	i-018b24b263d573101	Running	t2.micro	2/2 checks passed	No alarms	us-west-1b	ec2-54-219-28-9
-	i-09c979be50570a04f	Running	t2.micro	2/2 checks passed	No alarms	us-west-1b	ec2-54-67-121-1
-	i-06ae088a767a1285f	Stopped	t2.micro	-	No alarms	us-west-1a	-

Screenshot 3: The third instance, i-06ae088a767a1285f, is selected. Its details are shown in the modal window, including its status as Stopped, type t2.micro, and no tags.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
-	i-0125aef4db23e31fa	Running	t2.micro	2/2 checks passed	No alarms	us-west-1b	ec2-54-177-149-
-	i-026756a0d965dad12	Running	t2.micro	2/2 checks passed	No alarms	us-west-1b	ec2-18-144-65-1
-	i-095706ab6888eba4a	Running	t2.micro	2/2 checks passed	No alarms	us-west-1b	ec2-13-56-59-13
-	i-018b24b263d573101	Running	t2.micro	2/2 checks passed	No alarms	us-west-1b	ec2-54-219-28-9
-	i-09c979be50570a04f	Running	t2.micro	2/2 checks passed	No alarms	us-west-1b	ec2-54-67-121-1
<input checked="" type="checkbox"/>	i-06ae088a767a1285f	Stopped	t2.micro	-	No alarms	us-west-1a	-

10. List all the instance id and their tags

The screenshot shows the PyCharm IDE interface with a project named "project-individual1". The code editor displays a Python script named "get_id_tag.py". The script uses the boto3 library to connect to the EC2 service and print the instance ID and tags for each instance. The run output shows the execution of the script and its results.

```
File Edit View Navigate Code Refactor Run Tools VCS Window Help project-individual1 - get_id_tag.py - PyCharm
project-individual1 ) get_id_tag.py
Project Project D:\Year21-22\CS 6650 Fundamentals o
  ans.docx
  Cloud Comp Project 1 2021.docx
  describe_property.py
  ec2-addtag.py
  ec2-create-instance.py
  filterByTag.py
  find_run.py
  get_id_tag.py
  instances_and_tags.py
  reboot.py
  stop_instance.py
  update_tag.py
  ~$ans.docx
  ~WRL2106.tmp
External Libraries
Scratches and Consoles

import boto3

AWS_KEY = "AKIAQEEOWQHEYXIE2JKU"
AWS_SECRET = "6nbblW2v76zh14m7FXZRg8QW/XLaMf0f8C23g06lk"
REGION = "us-west-1"

print("Connecting to EC2")
ec2 = boto3.resource('ec2', aws_access_key_id=AWS_KEY,
                     aws_secret_access_key=AWS_SECRET,
                     region_name=REGION)

instances = ec2.instances.all()

for instance in instances:
    print(f' - Instance ID: {instance.id}, Instance Tag: {instance.tags}')


for instance in instances

Run: get_id_tag x
D:\IDE\PyCharm\anaconda\ins\python.exe "D:/Year21-22/CS 6650 Fundamentals of Cloud Computing/CW/project-individual1/get_id_tag.py"
Connecting to EC2
- Instance ID: i-0125aef4d823e31fa; Instance Tag: [{"Key": "name", "Value": "Project1"}]
- Instance ID: i-026756ab0d965dad12; Instance Tag: [{"Key": "name", "Value": "Project1"}]
- Instance ID: i-095706ab6b888eba4a; Instance Tag: [{"Key": "name", "Value": "Project1"}]
- Instance ID: i-018824b263d573101; Instance Tag: [{"Key": "name", "Value": "Project1"}]
- Instance ID: i-09c979be50570a04f; Instance Tag: [{"Key": "name", "Value": "Project1"}]
- Instance ID: i-06ae088a767a1285f; Instance Tag: [{"Key": "name", "Value": "Project1"}]

Smart commands execution: Highlighted commands can be interpreted and executed by the IDE in a smart way. // Press Ctrl+Enter to try this, or Enter to run the command in the console as usual. // You can turn this behavior off in Settings | Editor | Smart Keys | Smart Command Execution. (today) 19:44 CRLF UTF-8 4 spaces Python 3.8 Event Log
```

11. Stop all the instances using the instance id's

The screenshot shows the PyCharm IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help, and project-individual1 - stopAll.py - PyCharm. The left sidebar shows a project structure with files like ans.docx, Cloud Comp Project 1 2021.docx, and several Python scripts including ec2-addtag.py, ec2-reboot.py, etc. The main code editor window contains the following Python script:

```
AWS_SECRET = "6n8bll2v76Zn14m7FXZRg8QW/XlaMf0f8C23g06lk"
REGION = "us-west-1"

print("Connecting to EC2")
ec2 = boto3.resource('ec2', aws_access_key_id=AWS_KEY,
                     aws_secret_access_key=AWS_SECRET,
                     region_name=REGION)

instances = ec2.instances.all()

for instance in instances:
    ec2.instances.filter(instanceIds=[instance.id]).stop() # for stopping an ec2 instance
```

The run tab at the bottom shows the command: D:\IDE\PyCharm\anaconda\ins\python.exe "D:/Year21-22/CS 6650 Fundamentals of Cloud Computing/CW/project-individual1/stopAll.py". The output says "Connecting to EC2" and "Process finished with exit code 0".

Log on the aws

The screenshot shows the AWS Management Console with the EC2 service selected. The left navigation pane includes links for New EC2 Experience, EC2 Dashboard, EC2 Global View, Events, Tags, Limits, Instances (with sub-links for Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations), Images (AMIs), and Elastic Block Store (Volumes, Snapshots). The main content area displays the 'Instances (6)' page with a table of running instances. The table columns are: Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IPv4 DNS. The instances listed are:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
-	i-0125aeaf4d823e31fa	Stopping	t2.micro	-	No alarms	us-west-1b	ec2-54-177-149-
-	i-026756a0d965dad12	Stopping	t2.micro	-	No alarms	us-west-1b	ec2-18-144-65-1
-	i-095706ab6888eba4a	Stopping	t2.micro	-	No alarms	us-west-1b	ec2-13-56-59-13
-	i-018b24b263d573101	Stopping	t2.micro	-	No alarms	us-west-1b	ec2-54-219-28-9
-	i-09c979be50570a04f	Stopping	t2.micro	-	No alarms	us-west-1b	ec2-54-67-121-1
-	i-06ae088a767a1285f	Stopped	t2.micro	-	No alarms	us-west-1a	-

A modal dialog box titled "Select an instance above" is open at the bottom of the screen.

Part 2

1) Set up the EC2 instance, using t2.micro, add tag to it and it has to be accessed via any IP address

Step 1: choose an Amazon Machine Image (AMI)

I select the Amazon Linux 2 AMI.

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.

Search for an AMI by entering a search term e.g. "Windows"

Quick Start

My AMIs

AWS Marketplace

Community AMIs

Free tier only ⓘ

Amazon Linux Free tier eligible

Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-011996ff98de391d1 (64-bit x86) / ami-0d455ad8ecc5556a (64-bit Arm)

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. This AMI is the successor of the Amazon Linux AMI that is approaching end of life on December 31, 2020 and has been removed from this wizard.

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Red Hat Free tier eligible

Red Hat Enterprise Linux 8 (HVM), SSD Volume Type - ami-054965c6cd7c6e462 (64-bit x86) / ami-05f88a4bc91f4ea7 (64-bit Arm)

Red Hat Enterprise Linux version 8 (HVM), EBS General Purpose (SSD) Volume Type

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

SUSE Free tier eligible

SUSE Linux Enterprise Server 15 SP2 (HVM), SSD Volume Type - ami-05c558c169cf8d99 (64-bit x86) / ami-05e206de142efa13a (64-bit Arm)

SUSE Linux Enterprise Server 15 Service Pack 2 (HVM), EBS General Purpose (SSD) Volume Type. Amazon EC2 AMI Tools preinstalled, Apache 2.2, MySQL 5.5, PHP 5.3, Python 3.6, Ruby 2.5, Node.js 10.16, Java 8, .NET Core 2.1, and .NET Framework 4.7.2.

Select

64-bit (x86)
 64-bit (Arm)

Select

64-bit (x86)
 64-bit (Arm)

Select

64-bit (x86)

Feedback English (US) © 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use Cookie preferences

Step 2: choose an instance type

I select the type 't2.micro'.

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 families Current generation Show/Hide Columns

Family	Type	vCPUs ⓘ	Memory (GiB) ⓘ	Instance Storage (GB) ⓘ	EBS-Optimized Available ⓘ	Network Performance ⓘ	IPv6 Support ⓘ	
t2	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes	
<input checked="" type="checkbox"/>	t2	t2.micro <small>Free tier eligible</small>	1	1	EBS only	-	Low to Moderate	Yes
	t2.small	1	2	EBS only	-	Low to Moderate	Yes	
	t2.medium	2	4	EBS only	-	Low to Moderate	Yes	
	t2.large	2	8	EBS only	-	Low to Moderate	Yes	
	t2.xlarge	4	16	EBS only	-	Moderate	Yes	
	t2.2xlarge	8	32	EBS only	-	Moderate	Yes	

Cancel Previous Review and Launch Next: Configure Instance Details

Feedback English (US) © 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use Cookie preferences

Step 3: configure instance details.

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

Purchasing option: Request Spot Instances

Network: vpc-c57ad0a3 (default)

Subnet: No preference (default subnet in any Availability Zone)

Auto-assign Public IP: Use subnet setting (Enable)

Placement group: Add instance to placement group

Capacity Reservation: Open

Domain join directory: No directory

IAM role: None

Shutdown behavior: Stop

Stop - Hibernate behavior: Enable hibernation as an additional stop behavior

Step 4: add storage.

Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)	Delete on Termination	Encryption
Root	/dev/xvda	snap-0a28026ce5236baa	8	General Purpose SSD (gp2)	100 / 3000	N/A	<input checked="" type="checkbox"/>	Not Encrypted

Add New Volume

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. [Learn more](#) about free usage tier eligibility and usage restrictions.

Step 5: add tags

Key	(128 characters maximum)	Value	(256 characters maximum)	Instances	Volumes	Network Interfaces
MyFirst				<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Add another tag (Up to 50 tags maximum)

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.

Step 6: configure security group

The screenshot shows the AWS EC2 Step 6: Configure Security Group wizard. At the top, there are tabs: 1. Choose AMI, 2. Choose Instance Type, 3. Configure Instance, 4. Add Storage, 5. Add Tags, 6. Configure Security Group (which is selected), and 7. Review.

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

Security group name: launch-wizard-3
Description: launch-wizard-3 created 2021-09-18T23:21:43.126-07:00

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	Custom 0.0.0.0/0	e.g. SSH for Admin Desktop

Add Rule

Warning:
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.

Inbound rules

Type	Protocol	Port range	Source	Description - optional
SSH	TCP	22	Anywhere	
HTTP	TCP	80	Anywhere	

Add rule

Outbound rules

Type	Protocol	Port range	Destination	Description - optional
All traffic	All	All	Custom 0.0.0.0/0	

Add rule

According to the figure about security group configuration above, the inbound rules of source is anywhere, and the outbound rules is all traffic. So it can be accessed via any IP address.

Final result

Instances (1/3) [Info](#)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
MyfirstEC2	i-04174bd90577cccd4	Running	t2.micro	2/2 checks passed	No alarms	us-west-1b	ec2-3-101-56-221.us-west-1.compute.amazonaws.com
-	i-0ec19f73a4c80ab17	Running	t2.micro	2/2 checks passed	No alarms	us-west-1a	ec2-52-53-169-2
-	i-01e3b6ad82b196afb	Running	t2.micro	2/2 checks passed	No alarms	us-west-1a	ec2-54-241-59-1

Instance: i-04174bd90577cccd4

[Details](#) [Security](#) [Networking](#) [Storage](#) [Status checks](#) [Monitoring](#) [Tags](#)

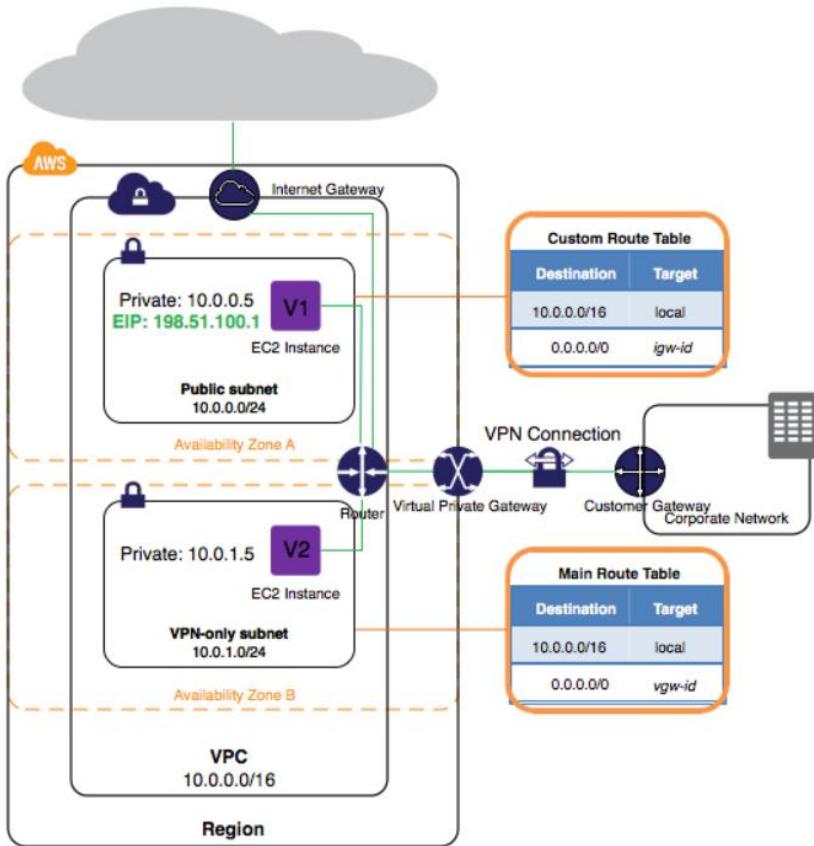
Instance summary [Info](#)

Instance ID i-04174bd90577cccd4	Public IPv4 address 3.101.56.221 open address	Private IPv4 addresses 172.31.31.61
IPv6 address -	Instance state Running	Public IPv4 DNS ec2-3-101-56-221.us-west-1.compute.amazonaws.com open address

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy Policy](#) [Terms of Use](#) [Cookie preferences](#)

2) Explain in brief the process by setting up a routing table

The following figure shows the routing of a VPC with both an Internet gateway and a virtual private gateway, as well as a public subnet and a VPN-only subnet.



Using Amazon Web Services, I can create route table in VPC services.

The screenshot shows the 'Create route table' interface in the AWS Management Console. The 'Route table settings' section includes:

- Name - optional:** Project1_routeTable
- VPC:** vpc-c57ad0a3

The 'Tags' section allows adding a tag named 'Name' with value 'Project1_routeTable'.

At the bottom, there are links for 'Feedback', 'English (US)', '© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.', 'Privacy Policy', 'Terms of Use', and 'Cookie preferences'.

Create route table

In the process, I need to choose a Virtual Private Cloud line to be used for the route table.

Result:

The screenshot shows the AWS VPC console. In the search bar, 'vpc' is typed. A green banner at the top right indicates that a route table named 'rtb-02fedf29b348fbfe2 | Project1_routeTable' was created successfully. Below the banner, the title 'rtb-02fedf29b348fbfe2 / Project1_routeTable' is displayed. A message says 'You can now check network connectivity with Reachability Analyzer' with a 'Run Reachability Analyzer' button. The 'Details' tab is selected, showing route table ID 'rtb-02fedf29b348fbfe2', Main status 'No', VPC 'vpc-c57ad0a3', Owner ID '008889205193', and explicit subnet associations and edge associations both listed as '-'. Below this, there are tabs for 'Routes', 'Subnet associations', 'Edge associations', 'Route propagation', and 'Tags'. The 'Routes' tab is active, showing one route entry: Destination '172.31.0.0/16' and Target 'rtb-02fedf29b348fbfe2'. An 'Edit routes' button is available. At the bottom, there are links for Feedback, English (US), and various AWS footer links.

3) Set up an Application Load Balancer in the console

Step 1: create target group

Configure settings.

The screenshot shows the 'Create target group' wizard, Step 1: Specify group details. The title is 'Specify group details'. It says 'Your load balancer routes requests to the targets in a target group and performs health checks on the targets.' A section titled 'Basic configuration' states 'Settings in this section cannot be changed after the target group is created.' Below it, 'Choose a target type' has three options: 'Instances' (selected), 'IP addresses', and 'Lambda function'. 'Instances' includes a note: 'Supports load balancing to instances within a specific VPC.' 'IP addresses' includes notes: 'Supports load balancing to VPC and on-premises resources.', 'Facilitates routing to multiple IP addresses and network interfaces on the same instance.', and 'Offers flexibility with microservice based architectures, simplifying inter-application communication.' 'Lambda function' includes notes: 'Facilitates routing to a single Lambda function.' and 'Accessible to Application Load Balancers only.' At the bottom, there is a 'Target group name' field and standard AWS footer links.

Register targets.

The screenshot shows the 'Register targets' step in the AWS EC2 console. On the left, a sidebar lists 'Step 1 Specify group details' and 'Step 2 Register targets'. The main area is titled 'Available instances (3/3)' and displays three instances:

Instance ID	Name	State	Security groups	Zone	Subnet ID
i-04174bd90577cccd4	MyfirstEC2	running	launch-wizard-2	us-west-1b	subnet-6ec1a408
i-0ec19f73a4c80ab17		running	default	us-west-1a	subnet-9d6cdcc7
i-01e3b6ad82b196afb		running	launch-wizard-1	us-west-1a	subnet-9d6cdcc7

A message indicates '3 selected'. Below the table, a port value '80' is entered in a field labeled 'Ports for the selected instances'. A button 'Include as pending below' is visible at the bottom.

Result

The screenshot shows the 'Target groups' page in the AWS EC2 console. The sidebar includes 'New EC2 Experience' feedback, 'EC2 Dashboard', 'Events', 'Tags', 'Limits', 'Instances' (with sub-options like 'Instances', 'Instance Types', 'Launch Templates', 'Spot Requests', 'Savings Plans', 'Reserved Instances', 'Dedicated Hosts', 'Capacity Reservations'), 'Images', 'AMIs', 'Elastic Block Store' (with sub-options like 'Volumes', 'Snapshots'), and 'Feedback'. The main area shows one target group:

Name	ARN	Port	Protocol	Target type	Load balancer
MyfirstEC2	arn:aws:elasticloadbalancing...	80	HTTP	Instance	MyFirstALB

A message at the bottom says 'Select a target group above.'

Step 2: create load balancer

Create load balancer type. I select the application load balancer.

Select load balancer type

Elastic Load Balancing supports four types of load balancers: Application Load Balancers, Network Load Balancers, Gateway Load Balancers, and Classic Load Balancers. Choose the load balancer type that meets your needs. [Learn more about which load balancer is right for you](#)

Application Load Balancer	Network Load Balancer	Gateway Load Balancer
Create	Create	Create
Choose an Application Load Balancer when you need a flexible feature set for your web applications with HTTP and HTTPS traffic. Operating at the request level, Application Load Balancers provide advanced routing and visibility features targeted at application architectures, including microservices and containers. Learn more >	Choose a Network Load Balancer when you need ultra-high performance, TLS offloading at scale, centralized certificate deployment, support for UDP, and static IP addresses for your application. Operating at the connection level, Network Load Balancers are capable of handling millions of requests per second securely while maintaining ultra-low latencies. Learn more >	Choose a Gateway Load Balancer when you need to deploy and manage a fleet of third-party virtual appliances that support GENEVE. These appliances enable you to improve security, compliance, and policy controls. Learn more >

[Cancel](#)

Feedback English (US) ▾

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy Policy](#) [Terms of Use](#) [Cookie preferences](#)

Configure.

EC2 > Load balancers > Create Application Load Balancer

Create Application Load Balancer [Info](#)

The Application Load Balancer distributes incoming HTTP and HTTPS traffic across multiple targets such as Amazon EC2 instances, microservices, and containers, based on request attributes. When the load balancer receives a connection request, it evaluates the listener rules in priority order to determine which rule to apply, and if applicable, it selects a target from the target group for the rule action.

▶ How Application Load Balancers work

Basic configuration

Load balancer name
Name must be unique within your AWS account and cannot be changed after the load balancer is created.
A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

Scheme [Info](#)
Scheme cannot be changed after the load balancer is created.

Internet-facing
An internet-facing load balancer routes requests from clients over the internet to targets. Requires a public subnet. [Learn more](#)

Internal
An internal load balancer routes requests from clients to targets using private IP addresses.

IP address type [Info](#)
Select the type of IP addresses that your subnets use.

IPv4

Feedback English (US) ▾

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy Policy](#) [Terms of Use](#) [Cookie preferences](#)

Final result

The screenshot shows the AWS EC2 Load Balancer console. On the left, there's a sidebar with navigation links for EC2 Dashboard, Events, Tags, Limits, Instances (with sub-links for Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations), Images (AMIs), and Elastic Block Store (Volumes, Snapshots, Lifecycle Manager). The main content area has a search bar at the top. Below it is a table with columns: Name, DNS name, State, VPC ID, Availability Zones, Type, and Created. A single row is visible: MyFirstALB, MyFirstALB-1322200524.us..., Provisioning, vpc-c57ad0a3, us-west-1b, us-west-1a, application, September 2021. Below the table is a detailed view of the MyFirstALB load balancer, showing tabs for Description, Listeners, Monitoring, Integrated services, and Tags. The Description tab is selected, displaying the Name (MyFirstALB) and ARN (arn:aws:elasticloadbalancing:us-west-1:008889205193:loadbalancer/app/MyFirstALB/2c6191b303779579). The bottom of the page includes standard AWS footer links: Feedback, English (US), Privacy Policy, Terms of Use, and Cookie preferences.

4) Create a VPC

The screenshot shows the AWS VPC Dashboard. On the left, there's a sidebar with sections for Virtual Private Cloud (Your VPCs, Subnets, Route Tables, Internet Gateways, Egress Only Internet Gateways, DHCP Options Sets, Elastic IPs, Managed Prefix Lists, Endpoints, Endpoint Services, NAT Gateways, Peering Connections) and Security. The main area displays 'Resources by Region' with counts for VPCs (2), Subnets (3), Route Tables (3), Internet Gateways (2), Egress-only Internet Gateways (0), DHCP options sets (1), and Elastic IPs (0) across the N. California region. To the right, the 'Service Health' section shows 'Amazon EC2 - US West (N. California)' with a status of 'Service is operating normally'. There are also 'Settings' and 'Additional Information' sections.

Launch EC2 instances

Step 1: choose AMI

Similarly, I choose Amazon Linux 2 AMI.

The screenshot shows the 'Choose an Amazon Machine Image (AMI)' wizard, step 1. It has tabs for 1. Choose AMI, 2. Choose Instance Type, 3. Configure Instance, 4. Add Storage, 5. Add Tags, 6. Configure Security Group, and 7. Review. The 'Choose AMI' tab is selected. A search bar at the top allows searching for AMIs by name. The results list includes:

- Amazon Linux** (Free tier eligible): Description: 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, Gilbc 2.26, Binutils 2.29.1, and the latest software packages through extras. This AMI is the successor of the Amazon Linux AMI that is approaching end of life on December 31, 2020 and has been removed from this wizard. Root device type: ebs, Virtualization type: hvm, ENA Enabled: Yes. Buttons: Select (radio buttons for 64-bit (x86) and 64-bit (Arm)).
- Red Hat Enterprise Linux 8 (HVM)**, SSD Volume Type: ami-054965c6cd7c6e462 (64-bit x86) / ami-05f88a4bc91f4ea7 (64-bit Arm): Description: Red Hat Enterprise Linux version 8 (HVM), EBS General Purpose (SSD) Volume Type. Root device type: ebs, Virtualization type: hvm, ENA Enabled: Yes. Buttons: Select (radio buttons for 64-bit (x86) and 64-bit (Arm)).
- SUSE Linux Enterprise Server 15 SP2 (HVM)**, SSD Volume Type: ami-05c558c169cfe8d99 (64-bit x86) / ami-05e206de142efa13a (64-bit Arm): Description: SUSE Linux Enterprise Server 15 Service Pack 2 (HVM), EBS General Purpose (SSD) Volume Type. Amazon EC2 AMI Tools preinstalled; Apache 2.2, MySQL 5.5, PHP 5.3, and Ruby 1.8.7 available. Root device type: ebs, Virtualization type: hvm, ENA Enabled: Yes. Buttons: Select (radio buttons for 64-bit (x86) and 64-bit (Arm)).

Step 2: choose an instance type

Similarly, I select t2.micro.

The screenshot shows the AWS EC2 instance creation wizard at Step 2: Choose an Instance Type. The 't2.micro' instance is selected, highlighted with a green border. The table lists various t2 and t3 instance types with their specifications: Family, Type, vCPUs, Memory (GiB), Instance Storage (GB), EBS-Optimized Available, Network Performance, and IPv6 Support. The 't2.micro' row is marked as 'Free tier eligible'. The 'Review and Launch' button is visible at the bottom right.

Step 3: configure

The screenshot shows the AWS EC2 instance creation wizard at Step 3: Configure Instance Details. The 't2.micro' instance is selected. The configuration options include Number of instances (1), Purchasing option (Request Spot instances), Network (vpc-c57ad0a3), Subnet (No preference), Auto-assign Public IP (Use subnet setting), Placement group (Add instance to placement group), Capacity Reservation (Open), Domain join directory (No directory), IAM role (None), Shutdown behavior (Stop), Stop - Hibernate behavior (Enable hibernation as an additional stop behavior), Enable termination protection (Protect against accidental termination), and Monitoring (Enable CloudWatch detailed monitoring). The 'Review and Launch' button is visible at the bottom right.

Step 4: add storage

The screenshot shows the 'Add Storage' step of the AWS EC2 instance creation wizard. The top navigation bar includes the AWS logo, services dropdown, search bar, and account information (scyqa1, N. California, Support). Below the navigation is a progress bar with steps 1-7: Choose AMI, Choose Instance Type, Configure Instance, Add Storage (highlighted in orange), Add Tags, Configure Security Group, and Review. The main content area is titled 'Step 4: Add Storage' with a sub-instruction: '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.' A table displays storage configuration for the 'Root' volume:

Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)	Delete on Termination	Encryption
Root	/dev/xvda	snap-0a28026ce52366baa	8	General Purpose SSD (gp2)	100 / 3000	N/A	<input checked="" type="checkbox"/>	Not Encrypted

A note below the table states: 'Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. Learn more about free usage tier eligibility and usage restrictions.'

At the bottom right are 'Cancel', 'Previous', 'Review and Launch' (highlighted in blue), and 'Next: Add Tags' buttons.

Step 5: add tags

The screenshot shows the 'Add Tags' step of the AWS EC2 instance creation wizard. The top navigation bar and progress bar are identical to the previous step. The main content area is titled 'Step 5: Add Tags' with a sub-instruction: '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.' A table allows adding tags:

Key	(128 characters maximum)	Value	(256 characters maximum)	Instances	Volumes	Network Interfaces
This resource currently has no tags						

Below the table, instructions say: 'Choose the Add tag button or click to add a Name tag. Make sure your IAM policy includes permissions to create tags.'

At the bottom right are 'Cancel', 'Previous', 'Review and Launch' (highlighted in blue), and 'Next: Configure Security Group' buttons.

Step 6: configure security group

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 name: launch-wizard-3

Description: launch-wizard-3 created 2021-09-18T23:51:46.016-07:00

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	Custom 0.0.0.0/0	e.g. SSH for Admin Desktop

Add Rule

Warning
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.

Cancel **Previous** **Review and Launch**

Final result

Your VPCs (1/2) Info

Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR	IPv6 ports
vpc-c57ad0a3	vpc-c57ad0a3	Available	172.31.0.0/16	-	-
vpc-0ef7df9a77d00bbb2	vpc-0ef7df9a77d00bbb2	Available	10.0.0.0/16	-	-

vpc-c57ad0a3

Details **CIDRs** **Flow logs** **Tags**

Details			
VPC ID vpc-c57ad0a3	State Available	DNS hostnames Enabled	DNS resolution Enabled

5) Write a 'hello world program' in Python using AWS

Create Lambda Function in Python

The screenshot shows the 'Create function' wizard in the AWS Lambda console. The 'Author from scratch' option is selected, with the sub-instruction 'Start with a simple Hello World example.' visible. Below this, the 'Basic information' section is shown, where the function name is set to 'HelloWorld'. The runtime is chosen as 'Python 3.8'. The permissions section notes that Lambda will create an execution role with CloudWatch Logs permissions.

Choose Author from scratch and configure environment of the python 3.8

The screenshot shows the 'Code source' editor for the 'HelloWorld' function. The code editor displays the following Python code:

```
1 import json
2
3 def lambda_handler(event, context):
4     # TODO implement
5     return {
6         'statusCode': 200,
7         'body': json.dumps('Hello from Lambda!')
8     }
```

Configurations

I can change the configurations for your Lambda function. For Example, set the request time out and memory.

The screenshot shows the AWS Lambda configuration page for a function named "HelloWorld". The "Configuration" tab is selected. On the left, a sidebar lists various configuration options: Triggers, Permissions, Destinations, Environment variables, Tags, VPC, Monitoring and operations tools, Concurrency, Asynchronous invocation, Code signing, Database proxies, File systems, and State machines. The main panel displays the "General configuration" section with fields for Description (empty), Memory (MB) set to 128, and Timeout set to 0 min 3 sec. A callout box for "AWS Compute Optimizer" suggests opting in for memory recommendations. At the bottom right of the main panel is an "Edit" button.

Configure test event

The screenshot shows the "Configure test event" dialog box overlaid on the AWS Lambda code editor. The dialog has a title bar "Configure test event" with a close button. It contains instructions: "A function can have up to 10 test events. The events are persisted so you can switch to another computer or web browser and test your function with the same events." Below this are two radio buttons: "Create new test event" (selected) and "Edit saved test events". The "Event template" dropdown is set to "hello-world". The "Event name" input field contains "MyEventName". Below it is a code editor showing a JSON object:

```
1  [{}  
2   "key1": "value1",  
3   "key2": "value2",  
4   "key3": "value3"  
5 ]
```

Final Response:

The screenshot shows the AWS Lambda Test console interface. At the top, a green banner displays the message: "The test event HelloWorld was successfully saved." Below this, the main area has tabs for "Code source" and "Info". The "Test" tab is selected, showing the "Execution result" for a test event named "HelloWorld". The response body is displayed as:

```
{ "statusCode": 200, "body": "Hello from Lambda!" }
```

Below the response, the "Function Logs" section shows the following log entries:

```
START RequestId: eaa02cbc-3d6c-4bb7-b798-f2ef0e05d234 Version: $LATEST
END RequestId: eaa02cbc-3d6c-4bb7-b798-f2ef0e05d234
REPORT RequestId: eaa02cbc-3d6c-4bb7-b798-f2ef0e05d234 Duration: 1.07 ms Billed Duration: 2 ms Memory Size: 128 MB Max Memory Used: 39 MB Init Duration: 116.80 ms
Request ID
eaa02cbc-3d6c-4bb7-b798-f2ef0e05d234|
```

At the bottom of the page, there are links for "Feedback", "English (US)", "Privacy Policy", "Terms of Use", and "Cookie preferences".

Summary:

As an overview of this project, I have to say I put a lot of work into it, especially in the condition we have an assignment and a project in just one week. Through this project, for the part1, I learnt a lot about python coding for amazon web service using the library ‘boto3’. And for the part 2, I reviewed the work of first week, and learnt some techniques about Lambda on AWS. This helps with understanding and long-term memory. More specific details about such as AWS and EC2 VPC will be covered in the next section.

The Amazon Web Services Global Cloud Infrastructure offers cloud computing services globally through its secure, flexible, effective platform. One of the remarkable advantages of the AWS is to lower the cost needed to establish the up-front capital infrastructure, which allows users to spin up numbers of servers instantly. There are also some other benefits by AWS. Firstly, AWS provides a vast global cloud infrastructure for users to make innovations, do experiments and iterate computations quickly. It is easy to build new applications and adapt them to the required scales of the workload. Secondly, AWS provides different use of services such as development platform or programming model for different users, which enables users to choose corresponding modes and experience the flexibility according to their work demand. Moreover, AWS is also a durable and reliable technology with official certifications and audits. In order to guarantee the integrity and safety of user's data, the AWS system is built with multiple layers of operational and physical security.

Through a lot of programming, I understand what the AWS can serve better. For example, the instances, the tags, the operations of reboot, creation, stop, update and so on. Through the comprehensive application of them, I believe, in the future, I can cooperate with team members of the class could computing, my friends or colleagues to complete larger projects in a more efficient way. I'm grateful for the experience