# ASSIGNMENT 2

## Auto-Scaling Web Application

Development Operations
26670 (2023 - 2024)

Student: Michael McKibbin ID: 20092733

## Contents

Planning: Application selection and architecture planning, including architecture diagram. AMI creation ........ 4

Installation of nvm, node, and app.js ........ 5

VPC and load balancer implementation. ........ 8

Testing: Test traffic and load balancing demonstration. ........ 26

Monitoring: Own script. ........ 29

APPENDIX A: Automated instance setup. ........ 30

APPENDIX B: monitor.sh - Instance info. ........ 39

APPENDIX C: memv2.sh - Cloudwatch. ........ 40

## Table of Figures

Figure 1 Installing nvm via SSH ........ 5

Figure 2 Installing Node.js and running app.js ........ 5

Figure 3 Initial version of app.js - Hello World. ........ 6

Figure 4 Server running. ........ 6

Figure 5 Updated app.js - Server instance IP address. ........ 7

Figure 6 Killing the process and restarting app.js to start new version. ........ 7

Figure 7 Preview of VPC configuration ........ 8

Figure 8 VPC workflow complete. ........ 9

Figure 9 VPC configured and active. ........ 10

Figure 10 VPC Route Tables. ........ 10

Figure 11 Security Groups ........ 11

Figure 12 VPC Resource Map with a route highlighted. ........ 11

Figure 13 Editing Subnet Settings. ........ 12

Figure 14 Set IP address to Auto Assignment. ........ 12

Figure 15 Launch Template Success. ........ 13

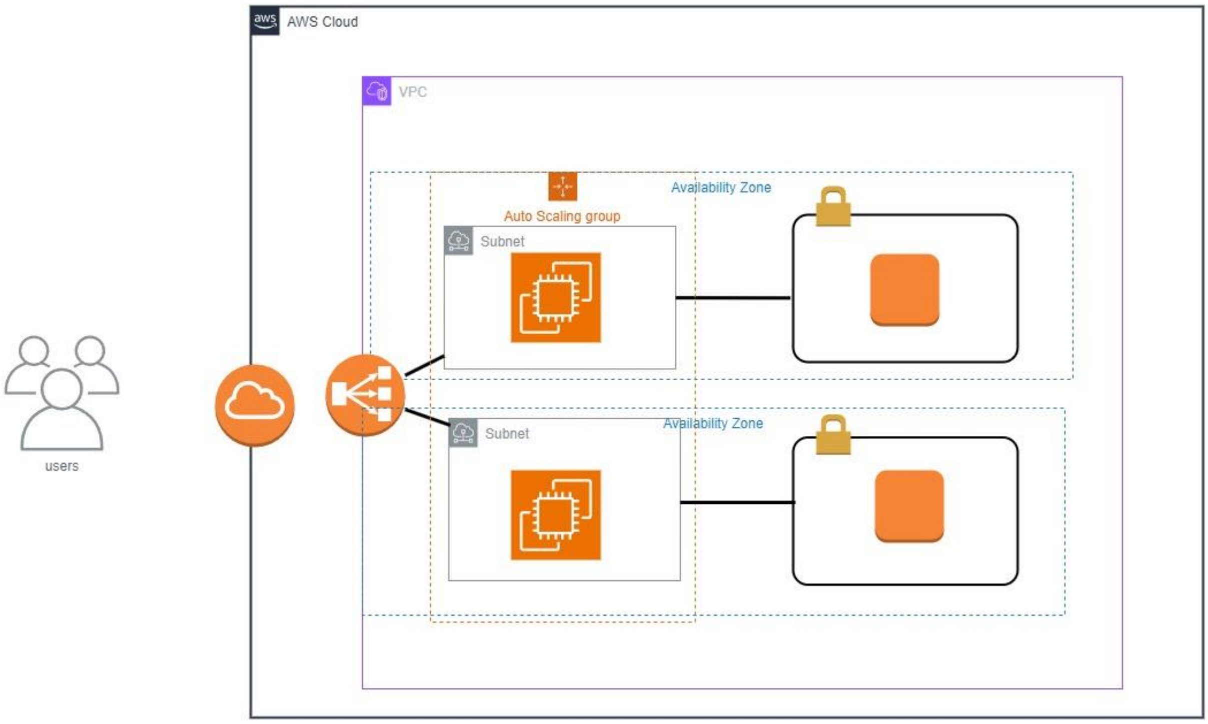
Figure 16 Launching an Instance - part 1 – choosing an AMI. ........ 14

Figure 17 Launching an Instance - part 2 – Key Pair, Network Settings (Subnets), and Security Group. ........ 15

Figure 18  Launching an Instance - part 3 - adding a script.................................16
Figure 19 Launch Instance with key pair and Network settings..........................17
Figure 20 Before Target Group setup................................................................18
Figure 21 Load balancing Target Group. ...........................................................18
Figure 22 Load Balancer created successfully. ...................................................19
Figure 23 Load Balancer details in browser.......................................................19
Figure 24 Creating an Auto Scaling group. ........................................................20
Figure 25 Auto Scaling - Choosing a Launch Template and Network..................21
Figure 26 Auto Scaling -Maximum instances, SNS Notifications, and Tags.........22
Figure 27 Auto Scaling - First Auto Scaled Instance. .........................................23
Figure 28 Auto Scaling - First Instance Running. ...............................................23
Figure 29 Auto Scaling - Targets registered.......................................................24
Figure 30 Auto Scaling - Cloudwatch Metrics....................................................25
Figure 31 Auto Scaling - Cloudwatch Alarms in progress...................................25
Figure 32 Auto Scaling - Using an infinite loop to create traffic..........................26
Figure 33 Curl command generates 100 requests. ...........................................27
Figure 34 Adding Listeners on port 80 and port 3000. ......................................27
Figure 35 Auto Scaled instances running. .........................................................28
Figure 36 Auto Scaled instances - Running and Terminated..............................28

# Planning: Application selection and architecture planning, including architecture diagram. AMI creation

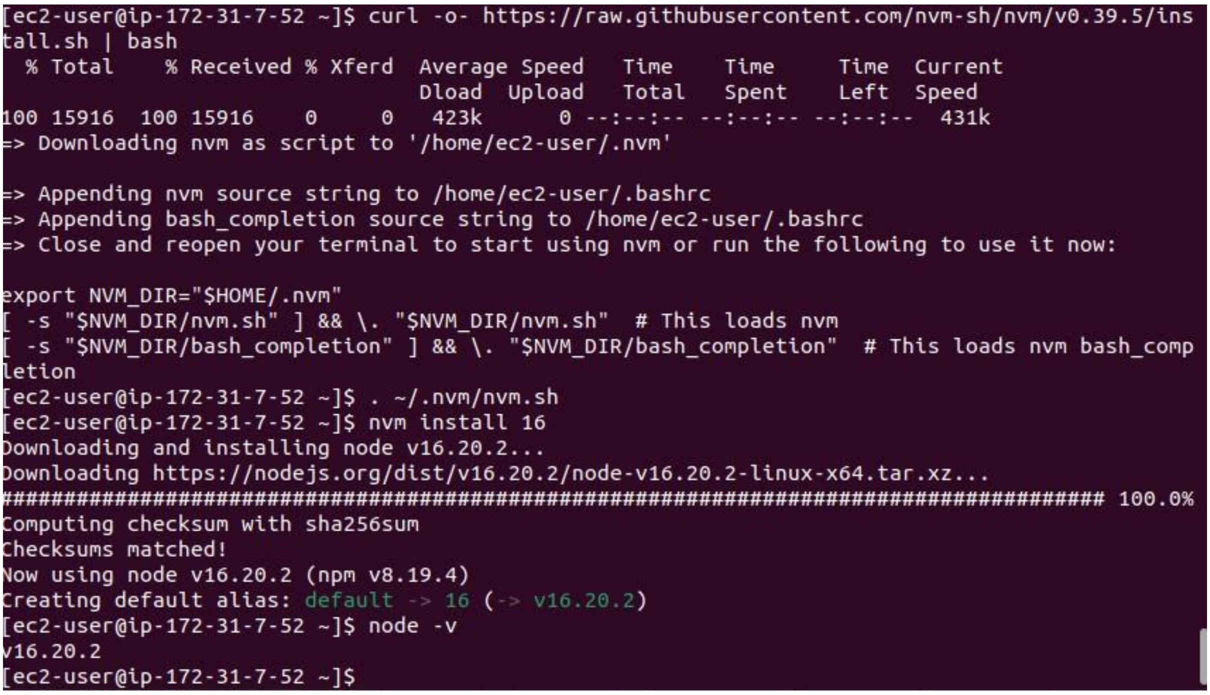
Basic architecture:
Cloud – VPC – Availability Zones – Autoscaling group – Subnets.



I used a custom script to launch an EC2 instance, set the key pair and security group, install Node.js, upload app.js, run the app, open a web browser page to show instance information and another page to show the app in action. First ‘Hello World’ and later showing the Server instance IP address.

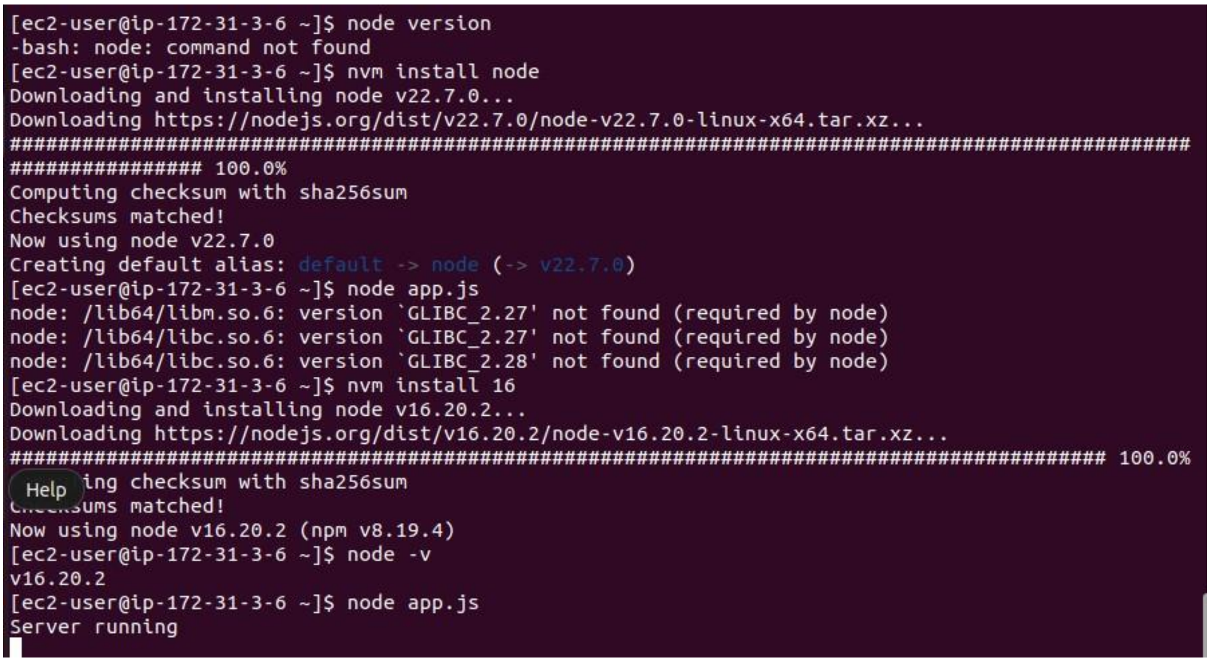
From this instance I created a custom AMI and updated the script to use this AMI for future instances which created instances with the index.html page, the monitoring script, and the application running on port 3000.

# Installation of nvm, node, and app.js



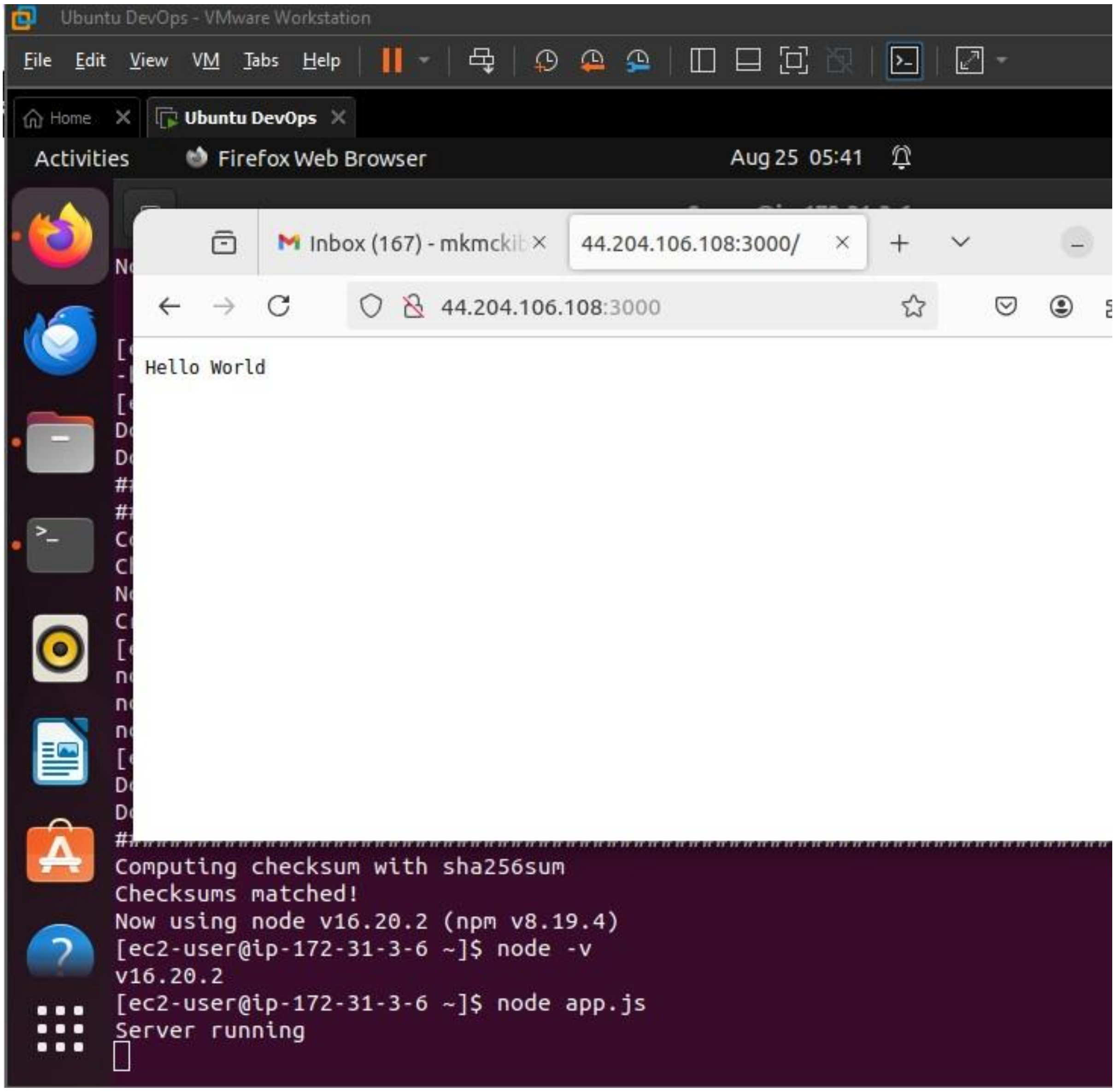
```
[ec2-user@ip-172-31-7-52 ~]$ curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.5/ins
tall.sh | bash
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 15916  100 15916    0     0   423k      0 --:--:-- --:--:-- --:--:--  431k
=> Downloading nvm as script to '/home/ec2-user/.nvm'

=> Appending nvm source string to /home/ec2-user/.bashrc
=> Appending bash_completion source string to /home/ec2-user/.bashrc
=> Close and reopen your terminal to start using nvm or run the following to use it now:

export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"  # This loads nvm
[ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion"  # This loads nvm bash_comp
letion
[ec2-user@ip-172-31-7-52 ~]$ . ~/.nvm/nvm.sh
[ec2-user@ip-172-31-7-52 ~]$ nvm install 16
Downloading and installing node v16.20.2...
Downloading https://nodejs.org/dist/v16.20.2/node-v16.20.2-linux-x64.tar.xz...
####################################################################################### 100.0%
Computing checksum with sha256sum
Checksums matched!
Now using node v16.20.2 (npm v8.19.4)
Creating default alias: default -> 16 (-> v16.20.2)
[ec2-user@ip-172-31-7-52 ~]$ node -v
v16.20.2
[ec2-user@ip-172-31-7-52 ~]$
```

*Figure 1 Installing nvm via SSH*



```
[ec2-user@ip-172-31-3-6 ~]$ node version
-bash: node: command not found
[ec2-user@ip-172-31-3-6 ~]$ nvm install node
Downloading and installing node v22.7.0...
Downloading https://nodejs.org/dist/v22.7.0/node-v22.7.0-linux-x64.tar.xz...
##################################################################################################
################ 100.0%
Computing checksum with sha256sum
Checksums matched!
Now using node v22.7.0
Creating default alias: default -> node (-> v22.7.0)
[ec2-user@ip-172-31-3-6 ~]$ node app.js
node: /lib64/libm.so.6: version `GLIBC_2.27' not found (required by node)
node: /lib64/libc.so.6: version `GLIBC_2.27' not found (required by node)
node: /lib64/libc.so.6: version `GLIBC_2.28' not found (required by node)
[ec2-user@ip-172-31-3-6 ~]$ nvm install 16
Downloading and installing node v16.20.2...
Downloading https://nodejs.org/dist/v16.20.2/node-v16.20.2-linux-x64.tar.xz...
########################################################################################### 100.0%
Computing checksum with sha256sum
Checksums matched!
Now using node v16.20.2 (npm v8.19.4)
[ec2-user@ip-172-31-3-6 ~]$ node -v
v16.20.2
[ec2-user@ip-172-31-3-6 ~]$ node app.js
Server running
```

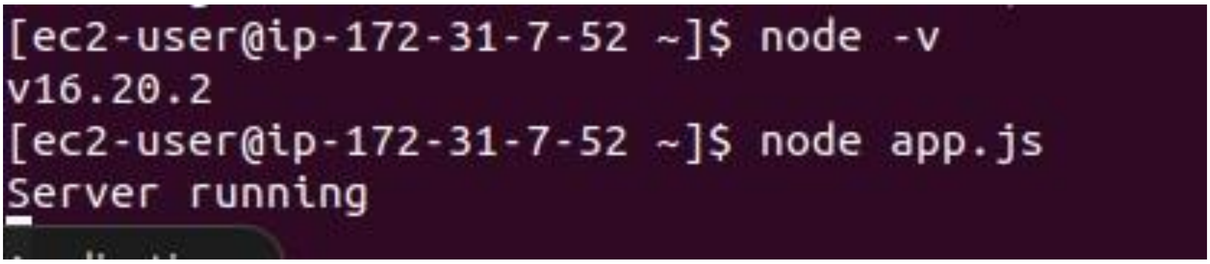
*Figure 2 Installing Node.js and running app.js*



```
Hello World
```

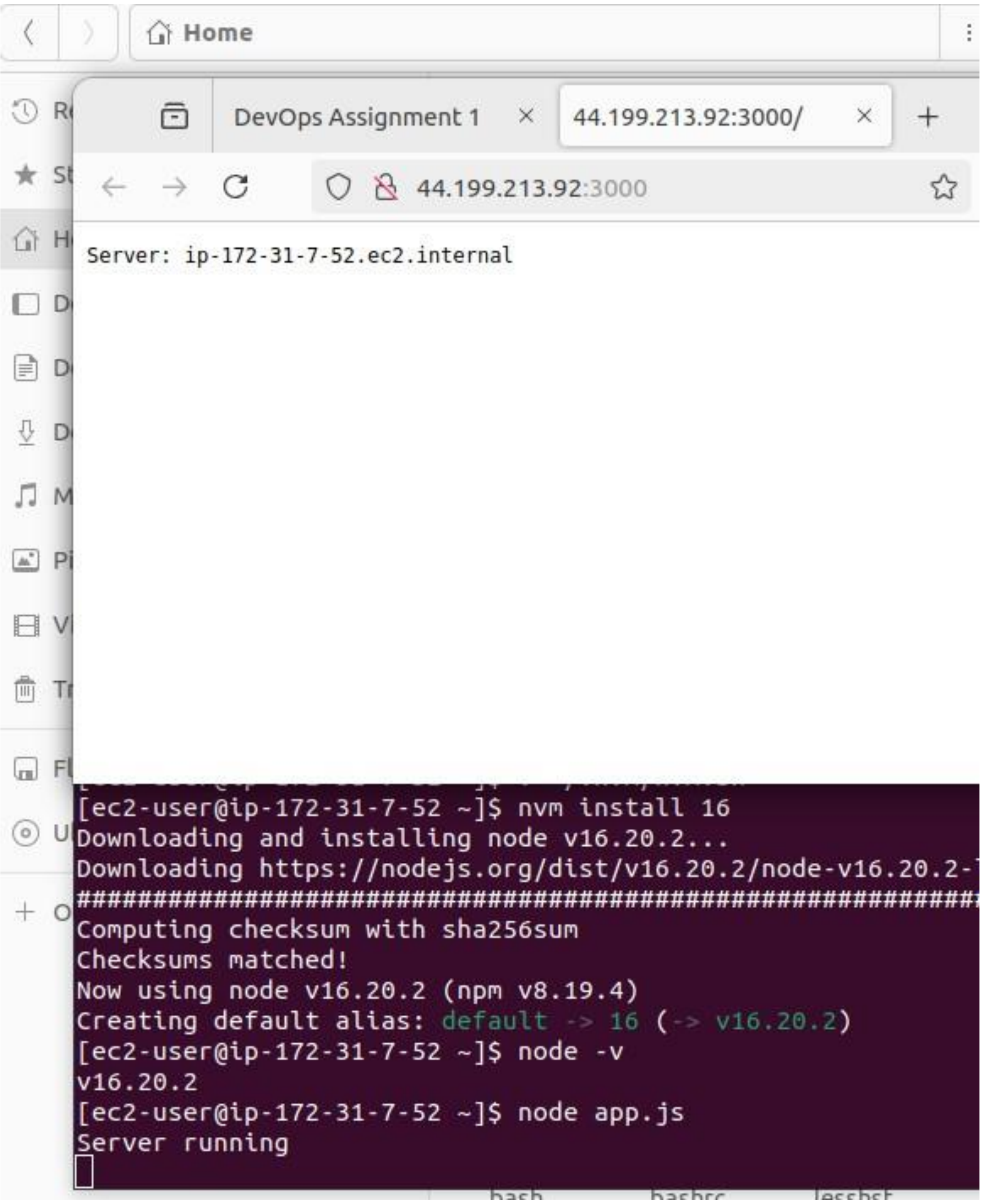
```
Computing checksum with sha256sum
Checksums matched!
Now using node v16.20.2 (npm v8.19.4)
[ec2-user@ip-172-31-3-6 ~]$ node -v
v16.20.2
[ec2-user@ip-172-31-3-6 ~]$ node app.js
Server running
```

*Figure 3 Initial version of app.js - Hello World.*



```
[ec2-user@ip-172-31-7-52 ~]$ node -v
v16.20.2
[ec2-user@ip-172-31-7-52 ~]$ node app.js
Server running
```

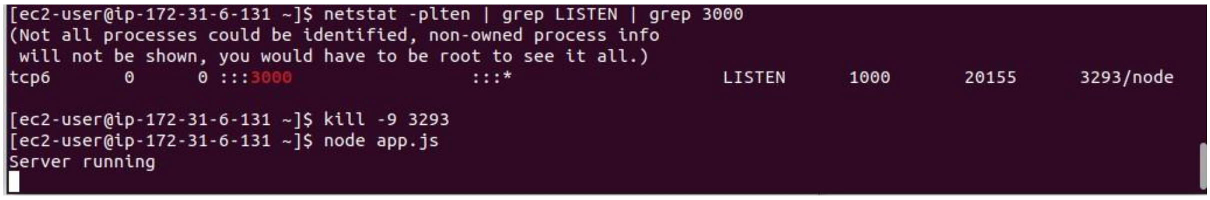
*Figure 4 Server running.*



```
Server: ip-172-31-7-52.ec2.internal
```

```
[ec2-user@ip-172-31-7-52 ~]$ nvm install 16
Downloading and installing node v16.20.2...
Downloading https://nodejs.org/dist/v16.20.2/node-v16.20.2-
############################################################
Computing checksum with sha256sum
Checksums matched!
Now using node v16.20.2 (npm v8.19.4)
Creating default alias: default -> 16 (-> v16.20.2)
[ec2-user@ip-172-31-7-52 ~]$ node -v
v16.20.2
[ec2-user@ip-172-31-7-52 ~]$ node app.js
Server running
```

*Figure 5 Updated app.js - Server instance IP address.*



```
[ec2-user@ip-172-31-6-131 ~]$ netstat -plten | grep LISTEN | grep 3000
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp6       0      0 :::3000                 :::*                    LISTEN      1000       20155      3293/node

[ec2-user@ip-172-31-6-131 ~]$ kill -9 3293
[ec2-user@ip-172-31-6-131 ~]$ node app.js
Server running
```

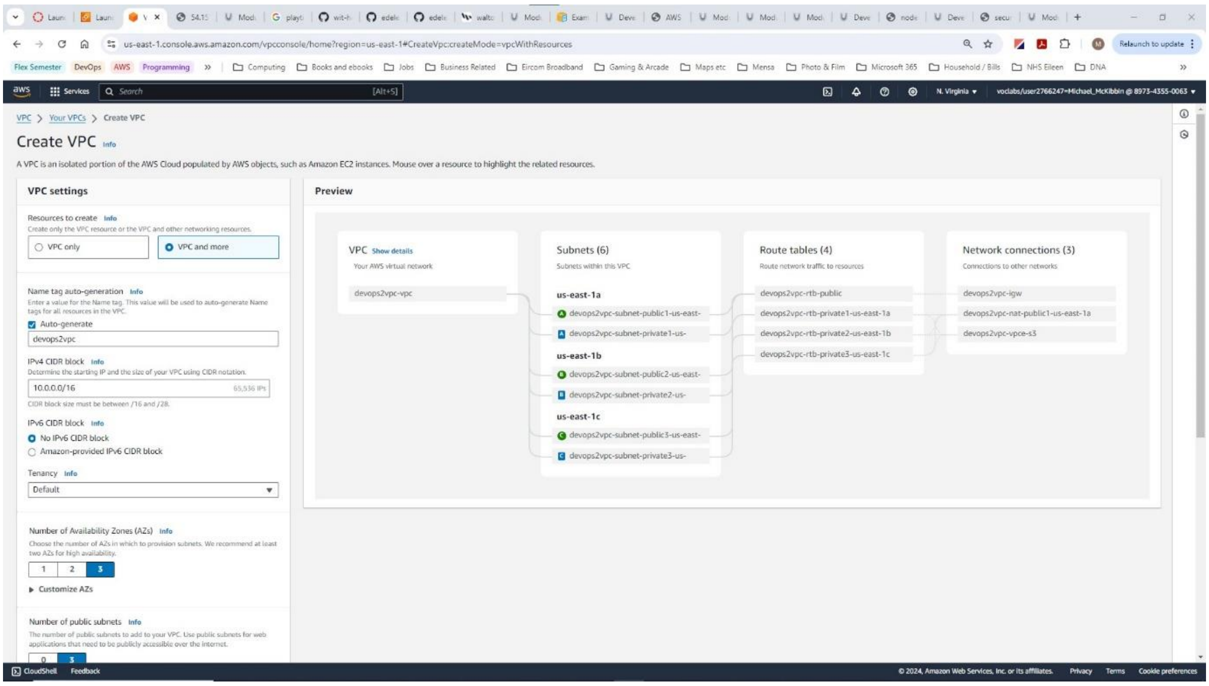
*Figure 6 Killing the process and restarting app.js to start new version.*

## VPC and load balancer implementation.

I created a VPC with three availability zones, us-east-1a, 1b, and 1c ,with public and private subnets in each.

An Elastic Load Balancer was configured to use my custom AMI and linked to an Autoscaling Group

The Load Balancer used Simple Scaling triggered by Cloudwatch alarms to increase or decrease the number of instances based on CPU usage. When usage is over 50% another instance is created, with a limit of three. When the work creating loop is terminated the low usage alarm kills unneeded instances when CPU usage drops below 30%.



us-east-1.console.aws.amazon.com/vpcconsole/home?region=us-east-1#CreateVpc:createMode=vpcWithResources

VPC > Your VPCs > Create VPC

Create VPC Info

A VPC is an isolated portion of the AWS Cloud populated by AWS objects, such as Amazon EC2 instances. Mouse over a resource to highlight the related resources.

VPC settings

Resources to create Info
Create only the VPC resource or the VPC and other networking resources.
○ VPC only ● VPC and more

Name tag auto-generation Info
Enter a value for the Name tag. This value will be used to auto-generate Name tags for all resources in the VPC.
☑ Auto-generate
devops2vpc

IPv4 CIDR block Info
Determine the starting IP and the size of your VPC using CIDR notation.
10.0.0.0/16 65,536 IPs
CIDR block size must be between /16 and /28.

IPv6 CIDR block Info
● No IPv6 CIDR block
○ Amazon-provided IPv6 CIDR block

Tenancy Info
Default

Number of Availability Zones (AZs) Info
Choose the number of AZs in which to provision subnets. We recommend at least two AZs for high availability.
1 2 [✓ 3]
▶ Customize AZs

Number of public subnets Info
The number of public subnets to add to your VPC. Use public subnets for web applications that need to be publicly accessible over the internet.
0 [✓ 3]

Preview

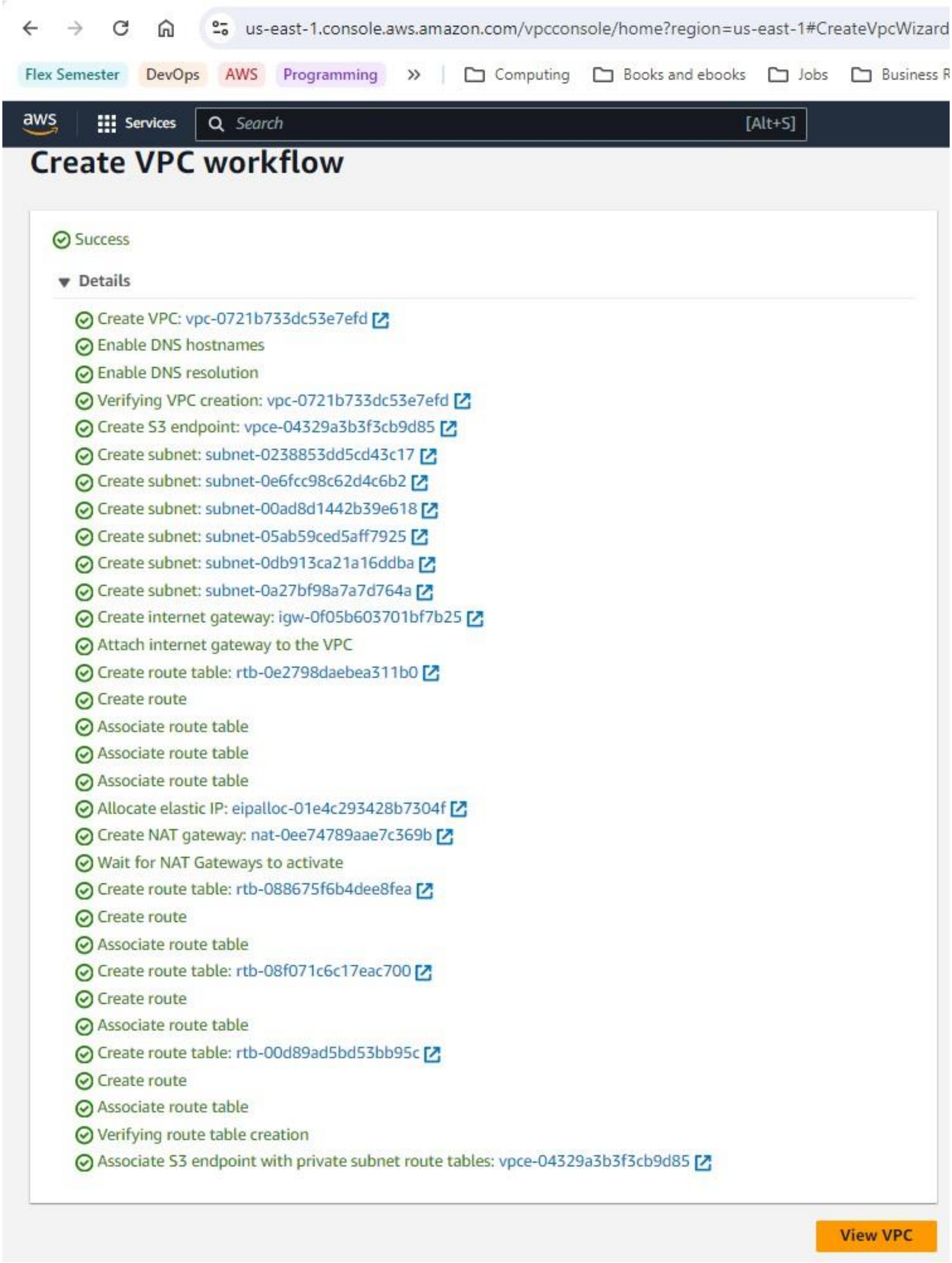
VPC Show details
Your AWS virtual network
devops2vpc-vpc

Subnets (6)
Subnets within this VPC
us-east-1a
devops2vpc-subnet-public1-us-east-
devops2vpc-subnet-private1-us-
us-east-1b
devops2vpc-subnet-public2-us-east-
devops2vpc-subnet-private2-us-
us-east-1c
devops2vpc-subnet-public3-us-east-
devops2vpc-subnet-private3-us-

Route tables (4)
Route network traffic to resources
devops2vpc-rtb-public
devops2vpc-rtb-private1-us-east-1a
devops2vpc-rtb-private2-us-east-1b
devops2vpc-rtb-private3-us-east-1c

Network connections (3)
Connections to other networks
devops2vpc-igw
devops2vpc-nat-public1-us-east-1a
devops2vpc-vpce-s3

*Figure 7 Preview of VPC configuration*



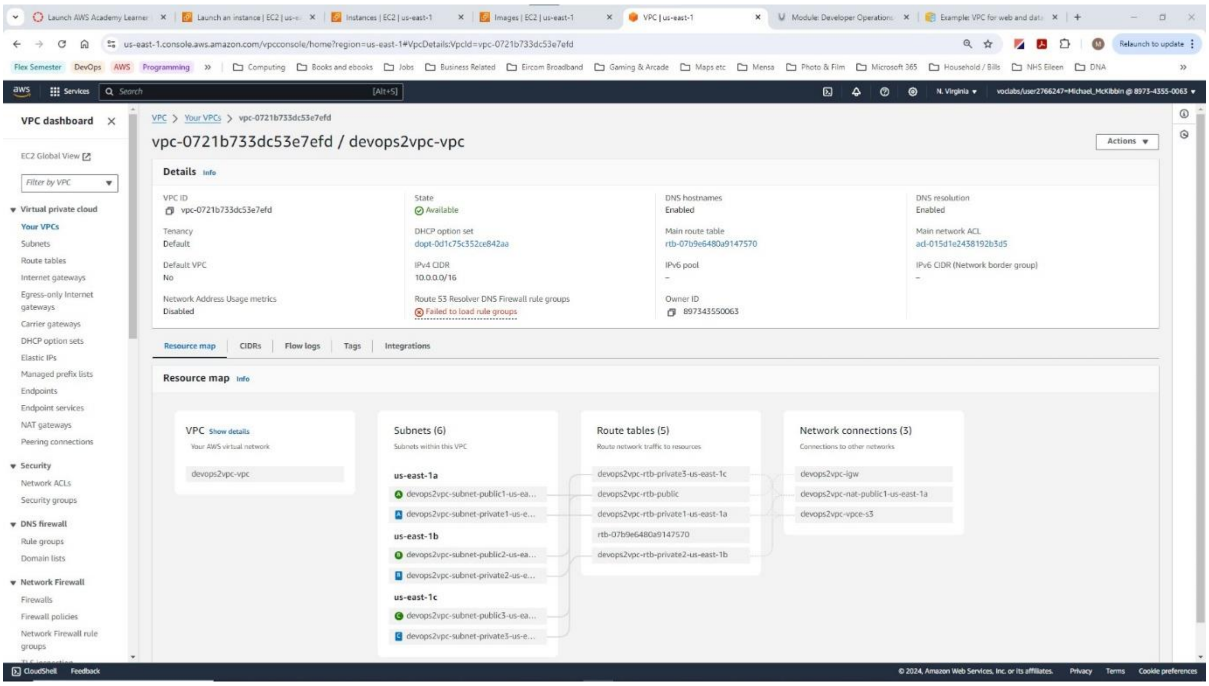
Create VPC workflow

✓ Success

▼ Details

- ✓ Create VPC: vpc-0721b733dc53e7efd
- ✓ Enable DNS hostnames
- ✓ Enable DNS resolution
- ✓ Verifying VPC creation: vpc-0721b733dc53e7efd
- ✓ Create S3 endpoint: vpce-04329a3b3f3cb9d85
- ✓ Create subnet: subnet-0238853dd5cd43c17
- ✓ Create subnet: subnet-0e6fcc98c62d4c6b2
- ✓ Create subnet: subnet-00ad8d1442b39e618
- ✓ Create subnet: subnet-05ab59ced5aff7925
- ✓ Create subnet: subnet-0db913ca21a16ddba
- ✓ Create subnet: subnet-0a27bf98a7a7d764a
- ✓ Create internet gateway: igw-0f05b603701bf7b25
- ✓ Attach internet gateway to the VPC
- ✓ Create route table: rtb-0e2798daebea311b0
- ✓ Create route
- ✓ Associate route table
- ✓ Associate route table
- ✓ Associate route table
- ✓ Allocate elastic IP: eipalloc-01e4c293428b7304f
- ✓ Create NAT gateway: nat-0ee74789aae7c369b
- ✓ Wait for NAT Gateways to activate
- ✓ Create route table: rtb-088675f6b4dee8fea
- ✓ Create route
- ✓ Associate route table
- ✓ Create route table: rtb-08f071c6c17eac700
- ✓ Create route
- ✓ Associate route table
- ✓ Create route table: rtb-00d89ad5bd53bb95c
- ✓ Create route
- ✓ Associate route table
- ✓ Verifying route table creation
- ✓ Associate S3 endpoint with private subnet route tables: vpce-04329a3b3f3cb9d85

*Figure 8 VPC workflow complete.*



us-east-1.console.aws.amazon.com/vpcconsole/home?region=us-east-1#VpcDetails:VpcId=vpc-0721b733dc53e7efd

VPC > Your VPCs > vpc-0721b733dc53e7efd

vpc-0721b733dc53e7efd / devops2vpc-vpc

Details

| | | | |
|---|---|---|---|
| VPC ID<br>vpc-0721b733dc53e7efd | State<br>Available | DNS hostnames<br>Enabled | DNS resolution<br>Enabled |
| Tenancy<br>Default | DHCP option set<br>dopt-0d1c75c352ce842aa | Main route table<br>rtb-07b9e6480a9147570 | Main network ACL<br>acl-015d1e2438192b3d5 |
| Default VPC<br>No | IPv4 CIDR<br>10.0.0.0/16 | IPv6 pool<br>– | IPv6 CIDR (Network border group)<br>– |
| Network Address Usage metrics<br>Disabled | Route 53 Resolver DNS Firewall rule groups<br>Failed to load rule groups | Owner ID<br>897343550063 | |

Resource map | CIDRs | Flow logs | Tags | Integrations

Resource map

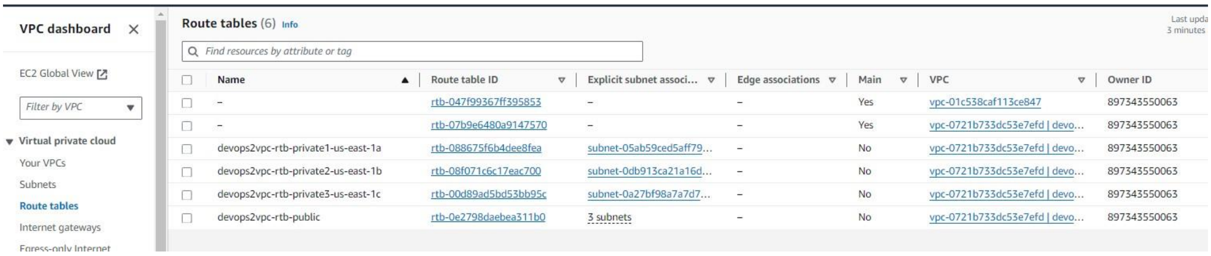
VPC — Your AWS virtual network
- devops2vpc-vpc

Subnets (6) — Subnets within this VPC
- us-east-1a
  - devops2vpc-subnet-public1-us-ea...
  - devops2vpc-subnet-private1-us-e...
- us-east-1b
  - devops2vpc-subnet-public2-us-ea...
  - devops2vpc-subnet-private2-us-e...
- us-east-1c
  - devops2vpc-subnet-public3-us-ea...
  - devops2vpc-subnet-private3-us-e...

Route tables (5) — Route network traffic to resources
- devops2vpc-rtb-private3-us-east-1c
- devops2vpc-rtb-public
- devops2vpc-rtb-private1-us-east-1a
- rtb-07b9e6480a9147570
- devops2vpc-rtb-private2-us-east-1b

Network connections (3) — Connections to other networks
- devops2vpc-igw
- devops2vpc-nat-public1-us-east-1a
- devops2vpc-vpce-s3

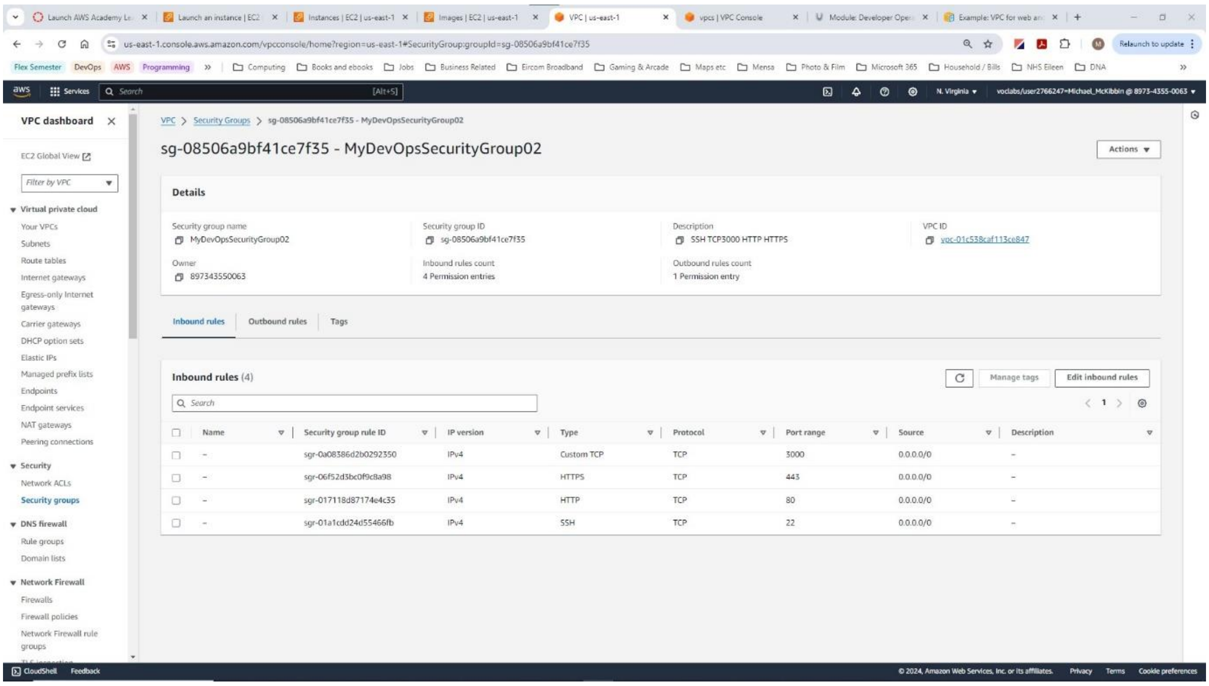
*Figure 9 VPC configured and active.*



Route tables (6)

| Name | Route table ID | Explicit subnet associ... | Edge associations | Main | VPC | Owner ID |
|---|---|---|---|---|---|---|
| – | rtb-047f99367ff395853 | – | – | Yes | vpc-01c538caf113ce847 | 897343550063 |
| – | rtb-07b9e6480a9147570 | – | – | Yes | vpc-0721b733dc53e7efd \| devo... | 897343550063 |
| devops2vpc-rtb-private1-us-east-1a | rtb-088675f6b4dee8fea | subnet-05ab59ced5aff79... | – | No | vpc-0721b733dc53e7efd \| devo... | 897343550063 |
| devops2vpc-rtb-private2-us-east-1b | rtb-08f071c6c17eac700 | subnet-0db913ca21a16d... | – | No | vpc-0721b733dc53e7efd \| devo... | 897343550063 |
| devops2vpc-rtb-private3-us-east-1c | rtb-00d89ad5bd53bb95c | subnet-0a27bf98a7a7d7... | – | No | vpc-0721b733dc53e7efd \| devo... | 897343550063 |
| devops2vpc-rtb-public | rtb-0e2798daebea311b0 | 3 subnets | – | No | vpc-0721b733dc53e7efd \| devo... | 897343550063 |

*Figure 10 VPC Route Tables.*



us-east-1.console.aws.amazon.com/vpcconsole/home?region=us-east-1#SecurityGroup:groupId=sg-08506a9bf41ce7f35

VPC > Security Groups > sg-08506a9bf41ce7f35 - MyDevOpsSecurityGroup02

sg-08506a9bf41ce7f35 - MyDevOpsSecurityGroup02

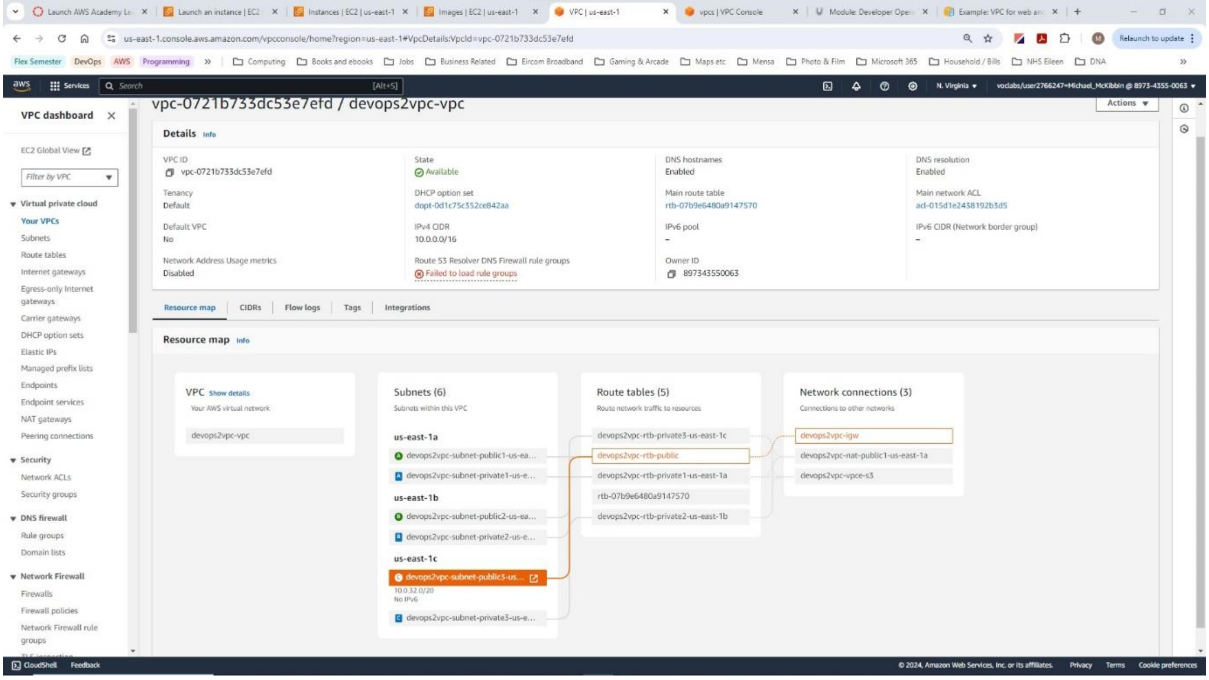
Details

| | | | |
|---|---|---|---|
| Security group name<br>MyDevOpsSecurityGroup02 | Security group ID<br>sg-08506a9bf41ce7f35 | Description<br>SSH TCP3000 HTTP HTTPS | VPC ID<br>vpc-01c538caf113ce847 |
| Owner<br>897343550063 | Inbound rules count<br>4 Permission entries | Outbound rules count<br>1 Permission entry | |

Inbound rules (4)

| Name | Security group rule ID | IP version | Type | Protocol | Port range | Source | Description |
|---|---|---|---|---|---|---|---|
| – | sgr-0a08386d2b0292350 | IPv4 | Custom TCP | TCP | 3000 | 0.0.0.0/0 | – |
| – | sgr-06f52d3bc0f9c8a98 | IPv4 | HTTPS | TCP | 443 | 0.0.0.0/0 | – |
| – | sgr-017118d87174e4c35 | IPv4 | HTTP | TCP | 80 | 0.0.0.0/0 | – |
| – | sgr-01a1cdd24d55466fb | IPv4 | SSH | TCP | 22 | 0.0.0.0/0 | – |

*Figure 11 Security Groups*


VPC dashboard
Actions ▼
Resource map
CIDRs
Flow logs
Tags
Integrations
VPC Show details
Your AWS virtual network
devops2vpc-vpc
Subnets (6)
Subnets within this VPC
us-east-1a
devops2vpc-subnet-public1-us-ea...
devops2vpc-subnet-private1-us-e...
us-east-1b
devops2vpc-subnet-public2-us-ea...
devops2vpc-subnet-private2-us-e...
us-east-1c
devops2vpc-subnet-public3-us...
10.0.32.0/20
No IPv6
devops2vpc-subnet-private3-us-e...
Route tables (5)
Route network traffic to resources
devops2vpc-rtb-private3-us-east-1c
devops2vpc-rtb-public
devops2vpc-rtb-private1-us-east-1a
rtb-07b9e6480a9147570
devops2vpc-rtb-private2-us-east-1b
Network connections (3)
Connections to other networks
devops2vpc-igw
devops2vpc-nat-public1-us-east-1a
devops2vpc-vpce-s3

us-east-1.console.aws.amazon.com/vpcconsole/home?region=us-east-1#VpcDetails:VpcId=vpc-0721b733dc53e7efd

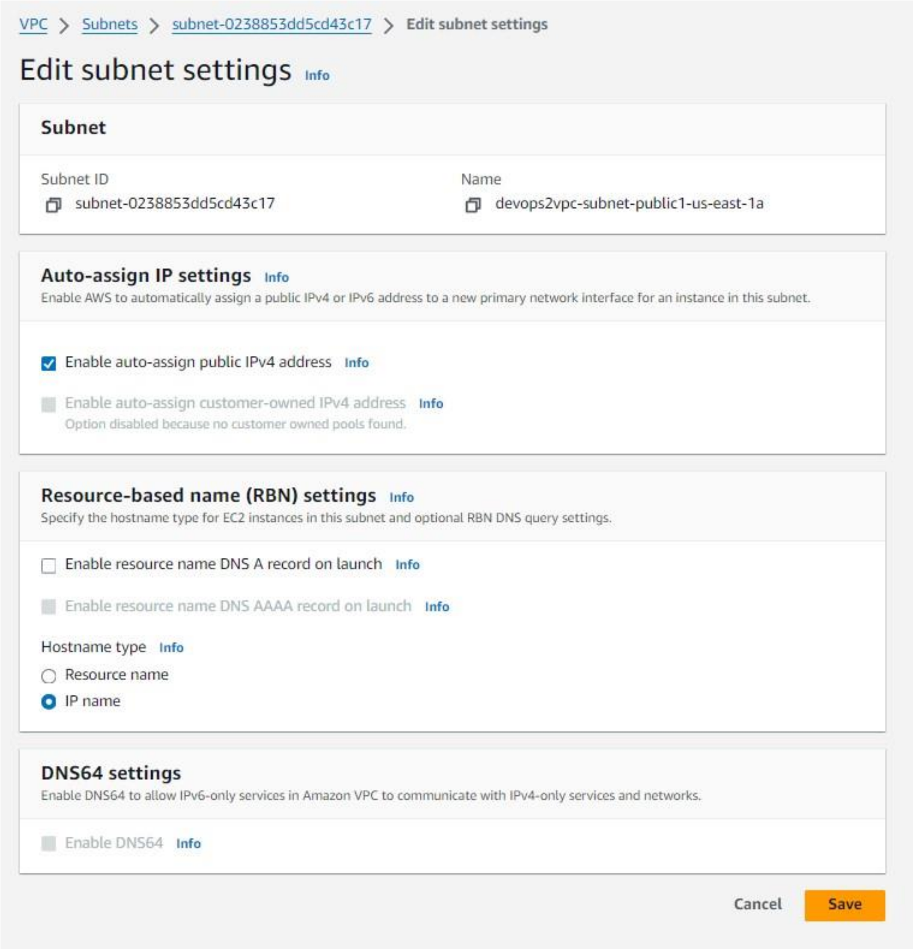
vpc-0721b733dc53e7efd / devops2vpc-vpc

Details

| | | | |
|---|---|---|---|
| VPC ID<br>vpc-0721b733dc53e7efd | State<br>Available | DNS hostnames<br>Enabled | DNS resolution<br>Enabled |
| Tenancy<br>Default | DHCP option set<br>dopt-0d1c75c352ce842aa | Main route table<br>rtb-07b9e6480a9147570 | Main network ACL<br>acl-015d1e2438192b3d5 |
| Default VPC<br>No | IPv4 CIDR<br>10.0.0.0/16 | IPv6 pool<br>– | IPv6 CIDR (Network border group)<br>– |
| Network Address Usage metrics<br>Disabled | Route 53 Resolver DNS Firewall rule groups<br>Failed to load rule groups | Owner ID<br>897343550063 | |

Resource map

*Figure 12 VPC Resource Map with a route highlighted.*



VPC > Subnets > subnet-0238853dd5cd43c17 > Edit subnet settings

Edit subnet settings Info

**Subnet**

Subnet ID
subnet-0238853dd5cd43c17

Name
devops2vpc-subnet-public1-us-east-1a

**Auto-assign IP settings** Info
Enable AWS to automatically assign a public IPv4 or IPv6 address to a new primary network interface for an instance in this subnet.

☑ Enable auto-assign public IPv4 address Info

□ Enable auto-assign customer-owned IPv4 address Info
Option disabled because no customer owned pools found.

**Resource-based name (RBN) settings** Info
Specify the hostname type for EC2 instances in this subnet and optional RBN DNS query settings.

□ Enable resource name DNS A record on launch Info

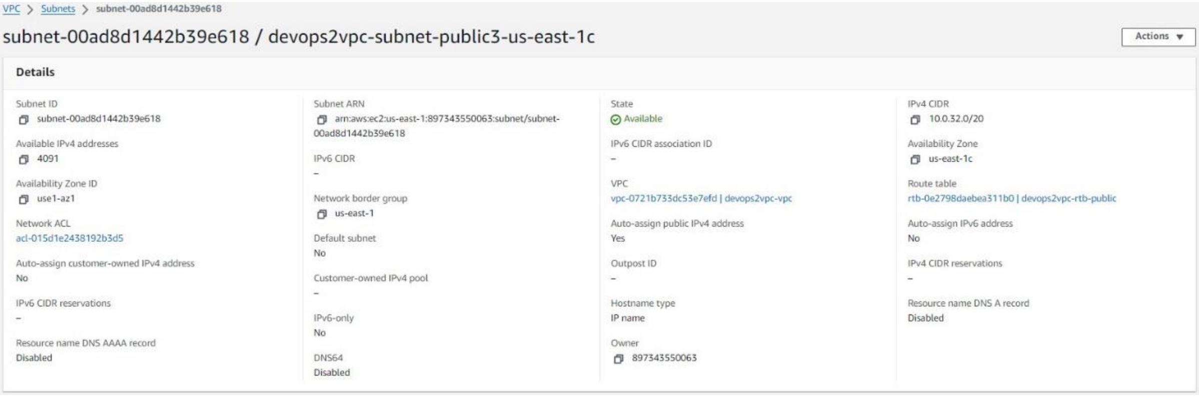
□ Enable resource name DNS AAAA record on launch Info

Hostname type Info
○ Resource name
● IP name

**DNS64 settings**
Enable DNS64 to allow IPv6-only services in Amazon VPC to communicate with IPv4-only services and networks.

□ Enable DNS64 Info

*Figure 13 Editing Subnet Settings.*



VPC > Subnets > subnet-00ad8d1442b39e618

subnet-00ad8d1442b39e618 / devops2vpc-subnet-public3-us-east-1c

**Details**

Subnet ID
subnet-00ad8d1442b39e618

Available IPv4 addresses
4091

Availability Zone ID
use1-az1

Network ACL
acl-015d1e2438192b3d5

Auto-assign customer-owned IPv4 address
No

IPv6 CIDR reservations
–

Resource name DNS AAAA record
Disabled

Subnet ARN
arn:aws:ec2:us-east-1:897343550063:subnet/subnet-00ad8d1442b39e618

IPv6 CIDR
–

Network border group
us-east-1

Default subnet
No

Customer-owned IPv4 pool
–

IPv6-only
No

DNS64
Disabled

State
Available

IPv6 CIDR association ID
–

VPC
vpc-0721b733dc53e7efd | devops2vpc-vpc

Auto-assign public IPv4 address
Yes

Outpost ID
–

Hostname type
IP name

Owner
897343550063

IPv4 CIDR
10.0.32.0/20

Availability Zone
us-east-1c

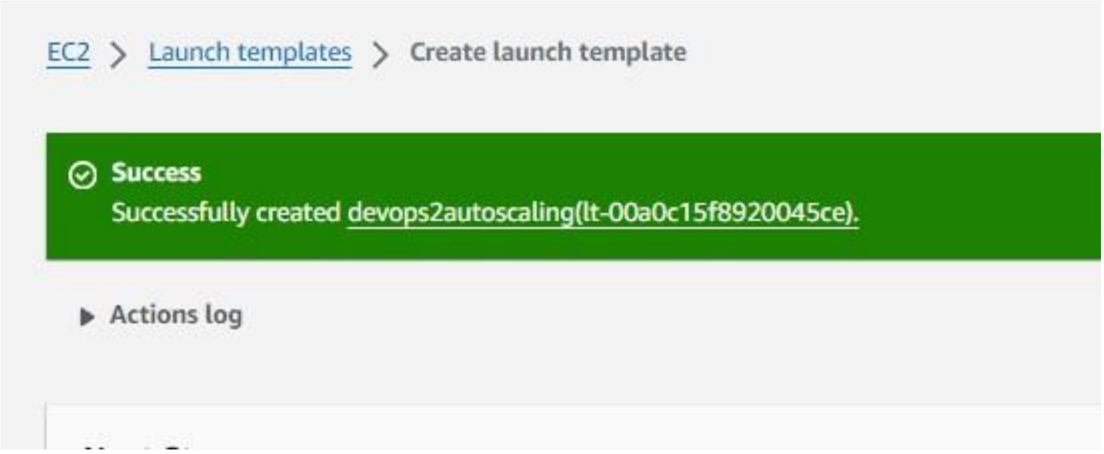
Route table
rtb-0e2798daebea311b0 | devops2vpc-rtb-public

Auto-assign IPv6 address
No

IPv4 CIDR reservations
–

Resource name DNS A record
Disabled

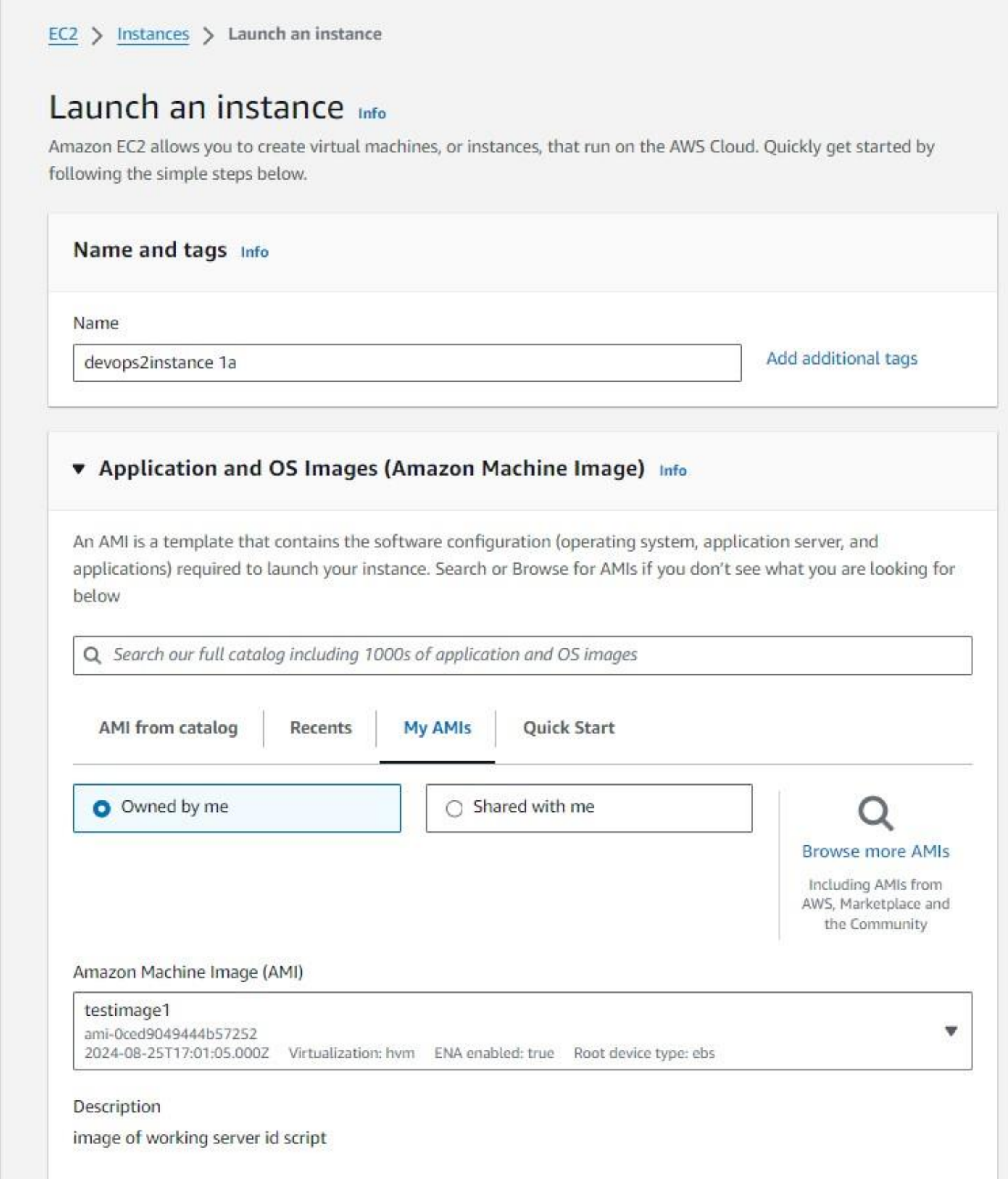
*Figure 14 Set IP address to Auto Assignment*



EC2 > Launch templates > Create launch template

**Success**
Successfully created devops2autoscaling(lt-00a0c15f8920045ce).

*Figure 15 Launch Template Success.*



EC2 > Instances > Launch an instance

Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags Info

Name

devops2instance 1a

Add additional tags

▼ Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

*Search our full catalog including 1000s of application and OS images*

AMI from catalog | Recents | My AMIs | Quick Start

● Owned by me ○ Shared with me

Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

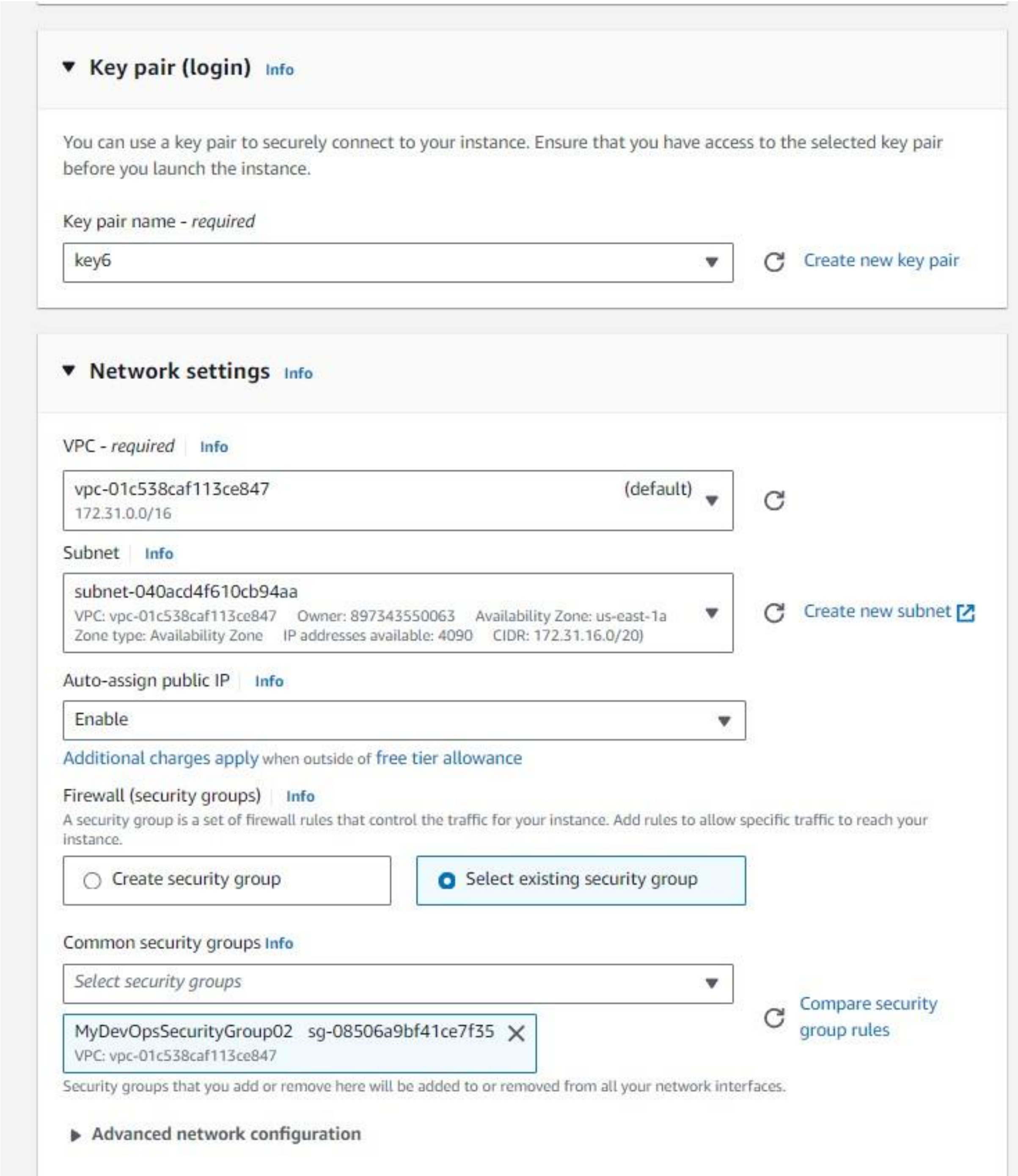
Amazon Machine Image (AMI)

testimage1
ami-0ced9049444b57252
2024-08-25T17:01:05.000Z Virtualization: hvm ENA enabled: true Root device type: ebs

Description

image of working server id script

*Figure 16 Launching an Instance - part 1 – choosing an AMI.*



▼ **Key pair (login)** Info

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

Key pair name - *required*

key6 ▼

Create new key pair

▼ **Network settings** Info

VPC - *required* Info

vpc-01c538caf113ce847 (default)
172.31.0.0/16

Subnet Info

subnet-040acd4f610cb94aa
VPC: vpc-01c538caf113ce847 Owner: 897343550063 Availability Zone: us-east-1a
Zone type: Availability Zone IP addresses available: 4090 CIDR: 172.31.16.0/20)

Create new subnet

Auto-assign public IP Info

Enable

Additional charges apply when outside of free tier allowance

Firewall (security groups) Info

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

○ Create security group ● Select existing security group

Common security groups Info

*Select security groups*

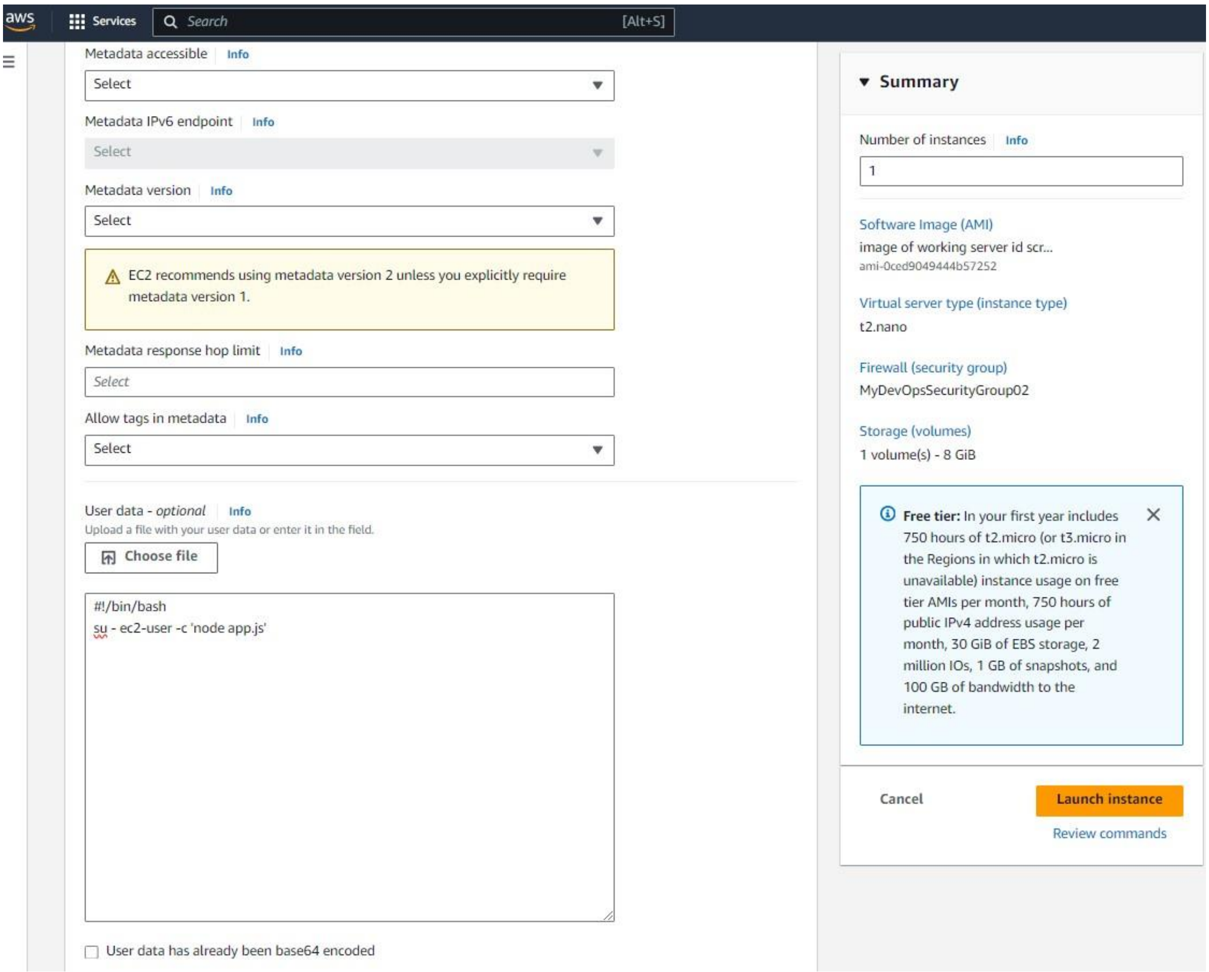
MyDevOpsSecurityGroup02 sg-08506a9bf41ce7f35
VPC: vpc-01c538caf113ce847

Compare security group rules

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

▶ **Advanced network configuration**

*Figure 17 Launching an Instance - part 2 – Key Pair, Network Settings (Subnets), and Security Group.*



Metadata accessible | Info

Select

Metadata IPv6 endpoint | Info

Select

Metadata version | Info

Select

⚠ EC2 recommends using metadata version 2 unless you explicitly require metadata version 1.

Metadata response hop limit | Info

Select

Allow tags in metadata | Info

Select

User data - *optional* | Info

Upload a file with your user data or enter it in the field.

Choose file

```
#!/bin/bash
su - ec2-user -c 'node app.js'
```

□ User data has already been base64 encoded

▼ **Summary**

Number of instances | Info

1

Software Image (AMI)

image of working server id scr...

ami-0ced9049444b57252

Virtual server type (instance type)

t2.nano

Firewall (security group)

MyDevOpsSecurityGroup02

Storage (volumes)

1 volume(s) - 8 GiB

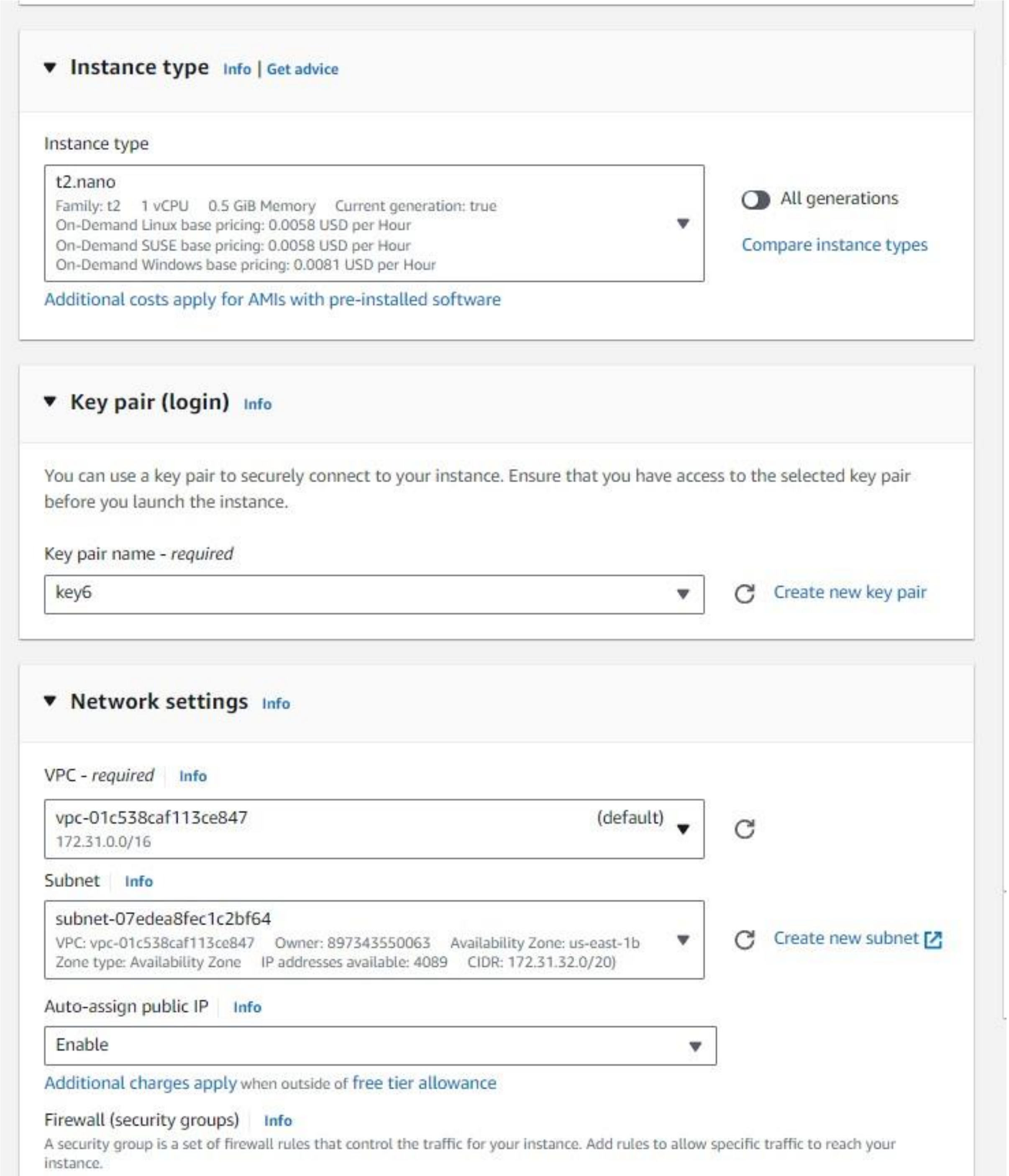
**Free tier:** In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4 address usage per month, 30 GiB of EBS storage, 2 million IOs, 1 GB of snapshots, and 100 GB of bandwidth to the internet.

Cancel

Launch instance

Review commands

*Figure 18  Launching an Instance - part 3 - adding a script.*



▼ Instance type Info | Get advice

Instance type

t2.nano
Family: t2 1 vCPU 0.5 GiB Memory Current generation: true
On-Demand Linux base pricing: 0.0058 USD per Hour
On-Demand SUSE base pricing: 0.0058 USD per Hour
On-Demand Windows base pricing: 0.0081 USD per Hour

All generations

Compare instance types

Additional costs apply for AMIs with pre-installed software

▼ Key pair (login) Info

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

Key pair name - *required*

key6

Create new key pair

▼ Network settings Info

VPC - *required* Info

vpc-01c538caf113ce847 (default)
172.31.0.0/16

Subnet Info

subnet-07edea8fec1c2bf64
VPC: vpc-01c538caf113ce847 Owner: 897343550063 Availability Zone: us-east-1b
Zone type: Availability Zone IP addresses available: 4089 CIDR: 172.31.32.0/20)

Create new subnet

Auto-assign public IP Info

Enable

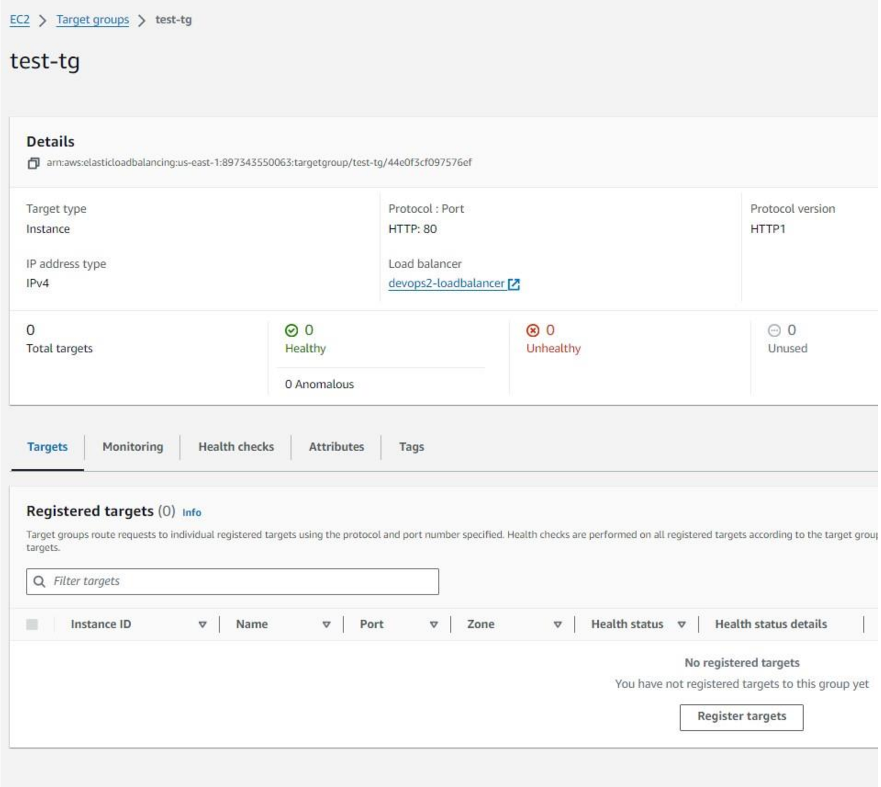
Additional charges apply when outside of free tier allowance

Firewall (security groups) Info

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

*Figure 19 Launch Instance with key pair and Network settings.*

## Auto-scaling implementation and demonstration of scaling activity based on CloudWatch alarm.



test-tg

**Details**

arn:aws:elasticloadbalancing:us-east-1:897343550063:targetgroup/test-tg/44e0f3cf097576ef

| Target type | Protocol : Port | Protocol version |
|---|---|---|
| Instance | HTTP: 80 | HTTP1 |
| IP address type | Load balancer | |
| IPv4 | devops2-loadbalancer | |

| 0 Total targets | 0 Healthy<br>0 Anomalous | 0 Unhealthy | 0 Unused |
|---|---|---|---|

**Registered targets (0)** Info

Target groups route requests to individual registered targets using the protocol and port number specified. Health checks are performed on all registered targets according to the target grou targets.

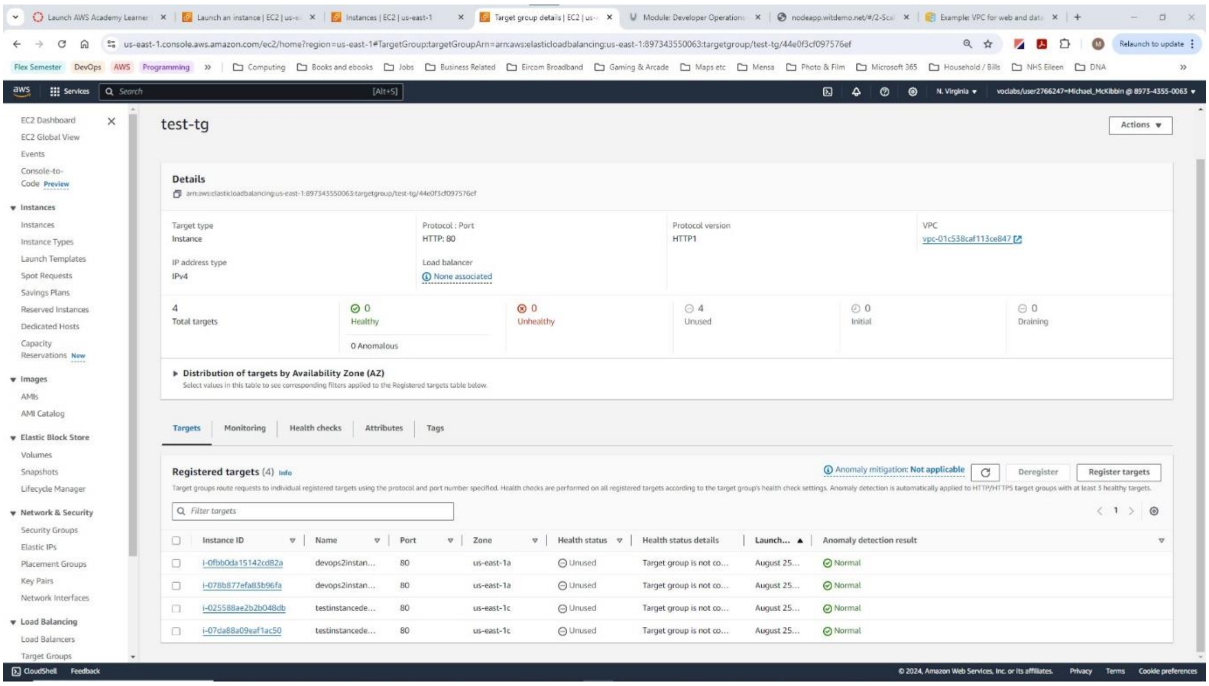
Filter targets

| Instance ID | Name | Port | Zone | Health status | Health status details |
|---|---|---|---|---|---|

No registered targets

You have not registered targets to this group yet

*Figure 20 Before Target Group setup.*



test-tg

**Details**

arn:aws:elasticloadbalancing:us-east-1:897343550063:targetgroup/test-tg/44e0f3cf097576ef

| Target type | Protocol : Port | Protocol version | VPC |
|---|---|---|---|
| Instance | HTTP: 80 | HTTP1 | vpc-01c538caf113ce847 |
| IP address type | Load balancer | | |
| IPv4 | None associated | | |

| 4 Total targets | 0 Healthy<br>0 Anomalous | 0 Unhealthy | 4 Unused | 0 Initial | 0 Draining |
|---|---|---|---|---|---|

**Distribution of targets by Availability Zone (AZ)**

Select values in this table to see corresponding filters applied to the Registered targets table below.

**Registered targets (4)** Info

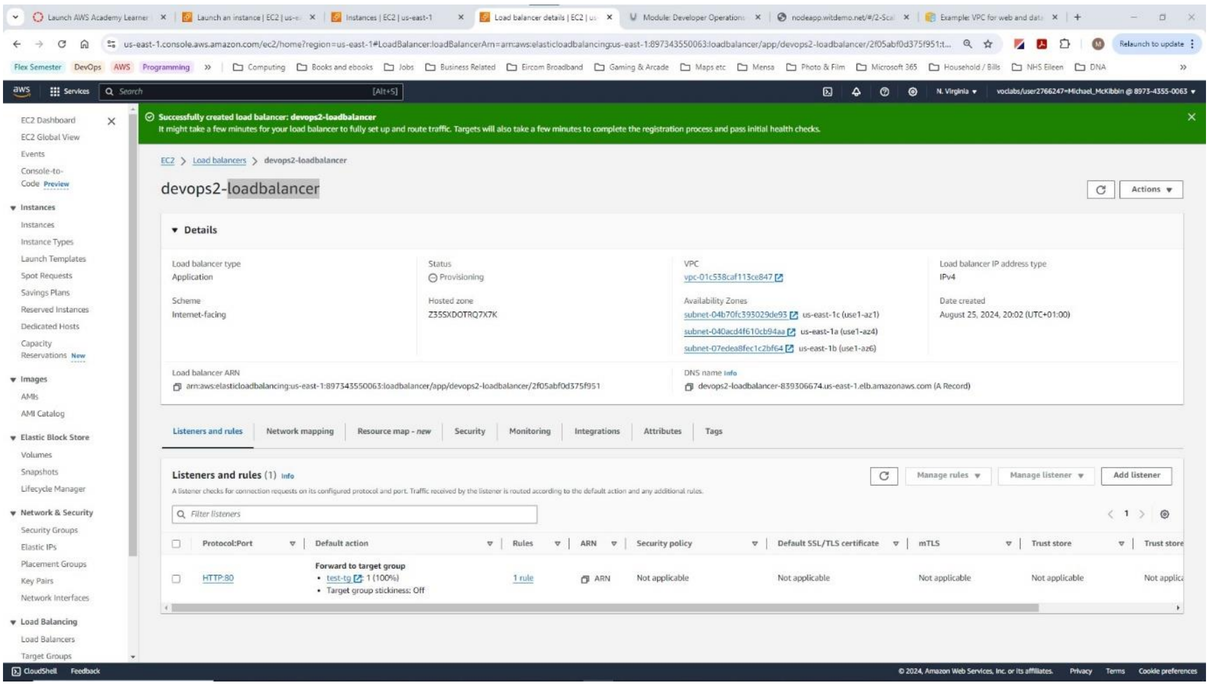
Anomaly mitigation: Not applicable

Target groups route requests to individual registered targets using the protocol and port number specified. Health checks are performed on all registered targets according to the target group's health check settings. Anomaly detection is automatically applied to HTTP/HTTPS target groups with at least 3 healthy targets.

Filter targets

| Instance ID | Name | Port | Zone | Health status | Health status details | Launch... | Anomaly detection result |
|---|---|---|---|---|---|---|---|
| i-0fbb0da15142cd82a | devops2instan... | 80 | us-east-1a | Unused | Target group is not co... | August 25... | Normal |
| i-078b877efa83b96fa | devops2instan... | 80 | us-east-1a | Unused | Target group is not co... | August 25... | Normal |
| i-025588ae2b2b048db | testinstancede... | 80 | us-east-1c | Unused | Target group is not co... | August 25... | Normal |
| i-07da88a09eaf1ac50 | testinstancede... | 80 | us-east-1c | Unused | Target group is not co... | August 25... | Normal |

*Figure 21 Load balancing Target Group.*



**Successfully created load balancer: devops2-loadbalancer**
It might take a few minutes for your load balancer to fully set up and route traffic. Targets will also take a few minutes to complete the registration process and pass initial health checks.

devops2-loadbalancer

**Details**

Load balancer type: Application

Status: Provisioning

VPC: vpc-01c538caf113ce847

Load balancer IP address type: IPv4

Scheme: Internet-facing

Hosted zone: Z35SXDOTRQ7X7K

Availability Zones:
subnet-04b70fc393029de93 us-east-1c (use1-az1)
subnet-040acd4f610cb94aa us-east-1a (use1-az4)
subnet-07edea8fec1c2bf64 us-east-1b (use1-az6)

Date created: August 25, 2024, 20:02 (UTC+01:00)

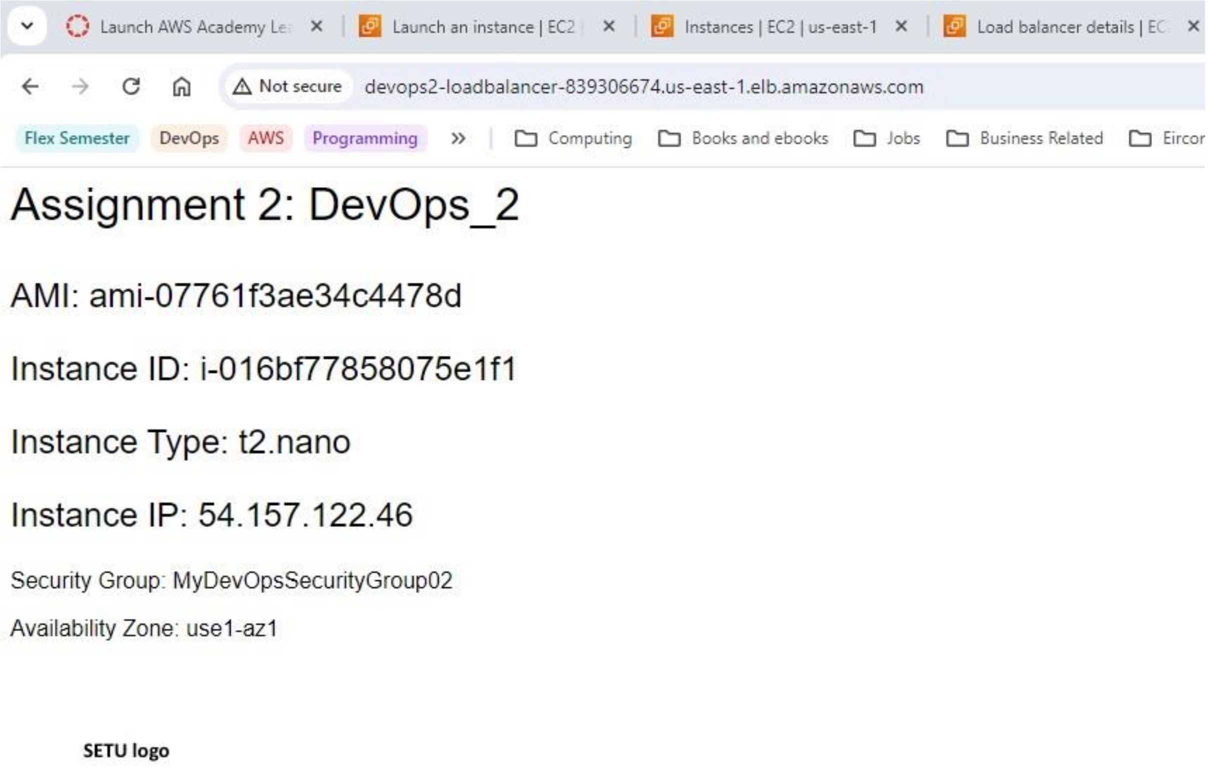
Load balancer ARN: arn:aws:elasticloadbalancing:us-east-1:897343550063:loadbalancer/app/devops2-loadbalancer/2f05abf0d375f951

DNS name Info: devops2-loadbalancer-839306674.us-east-1.elb.amazonaws.com (A Record)

**Listeners and rules** (1) Info
A listener checks for connection requests on its configured protocol and port. Traffic received by the listener is routed according to the default action and any additional rules.

| Protocol:Port | Default action | Rules | ARN | Security policy | Default SSL/TLS certificate | mTLS | Trust store | Trust store |
|---|---|---|---|---|---|---|---|---|
| HTTP:80 | **Forward to target group**<br>• test-tg: 1 (100%)<br>• Target group stickiness: Off | 1 rule | ARN | Not applicable | Not applicable | Not applicable | Not applicable | Not applica |

*Figure 22 Load Balancer created successfully.*



# Assignment 2: DevOps_2

AMI: ami-07761f3ae34c4478d

Instance ID: i-016bf77858075e1f1

Instance Type: t2.nano

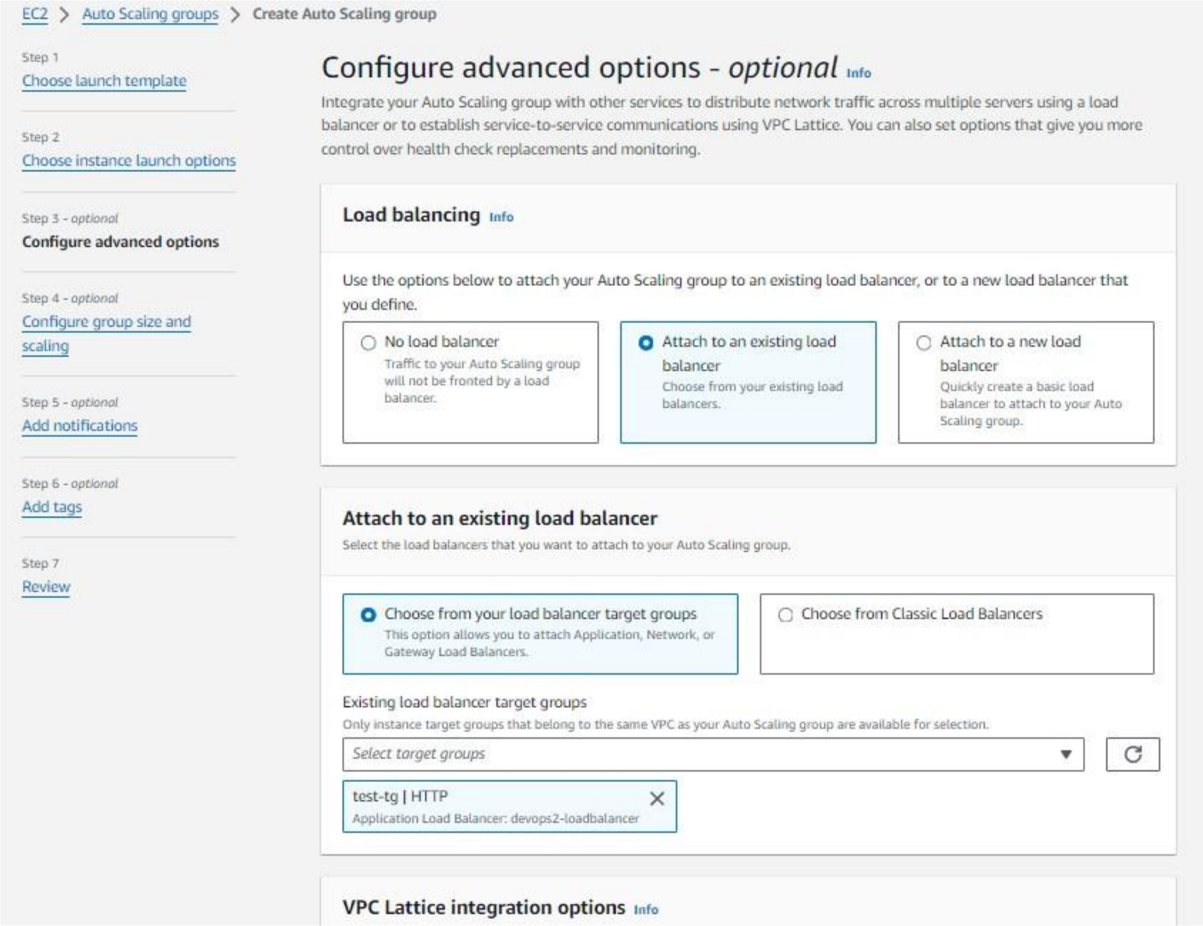
Instance IP: 54.157.122.46

Security Group: MyDevOpsSecurityGroup02

Availability Zone: use1-az1

**SETU logo**

*Figure 23 Load Balancer details in browser.*



EC2 > Auto Scaling groups > Create Auto Scaling group

Step 1
Choose launch template

Step 2
Choose instance launch options

Step 3 - *optional*
**Configure advanced options**

Step 4 - *optional*
Configure group size and scaling

Step 5 - *optional*
Add notifications

Step 6 - *optional*
Add tags

Step 7
Review

## Configure advanced options - *optional* Info

Integrate your Auto Scaling group with other services to distribute network traffic across multiple servers using a load balancer or to establish service-to-service communications using VPC Lattice. You can also set options that give you more control over health check replacements and monitoring.

### Load balancing Info

Use the options below to attach your Auto Scaling group to an existing load balancer, or to a new load balancer that you define.

○ No load balancer
Traffic to your Auto Scaling group will not be fronted by a load balancer.

● Attach to an existing load balancer
Choose from your existing load balancers.

○ Attach to a new load balancer
Quickly create a basic load balancer to attach to your Auto Scaling group.

### Attach to an existing load balancer

Select the load balancers that you want to attach to your Auto Scaling group.

● Choose from your load balancer target groups
This option allows you to attach Application, Network, or Gateway Load Balancers.

○ Choose from Classic Load Balancers

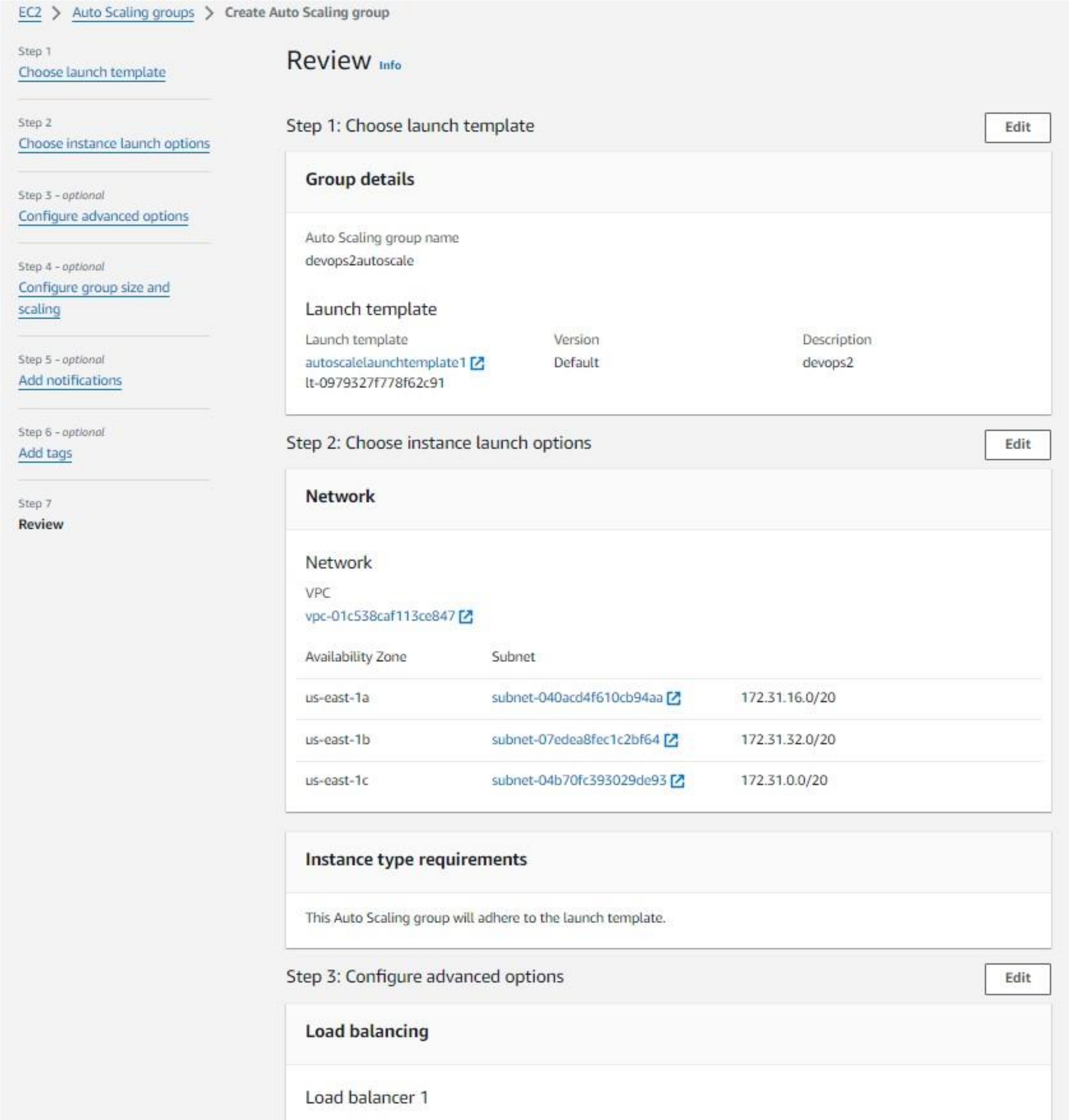
Existing load balancer target groups
Only instance target groups that belong to the same VPC as your Auto Scaling group are available for selection.

*Select target groups*

test-tg | HTTP
Application Load Balancer: devops2-loadbalancer

### VPC Lattice integration options Info

*Figure 24 Creating an Auto Scaling group.*



## Review

Step 1: Choose launch template

**Group details**

Auto Scaling group name
devops2autoscale

Launch template

| Launch template | Version | Description |
|---|---|---|
| autoscalelaunchtemplate1<br>lt-0979327f778f62c91 | Default | devops2 |

Step 2: Choose instance launch options

**Network**

Network

VPC
vpc-01c538caf113ce847

| Availability Zone | Subnet | |
|---|---|---|
| us-east-1a | subnet-040acd4f610cb94aa | 172.31.16.0/20 |
| us-east-1b | subnet-07edea8fec1c2bf64 | 172.31.32.0/20 |
| us-east-1c | subnet-04b70fc393029de93 | 172.31.0.0/20 |

**Instance type requirements**

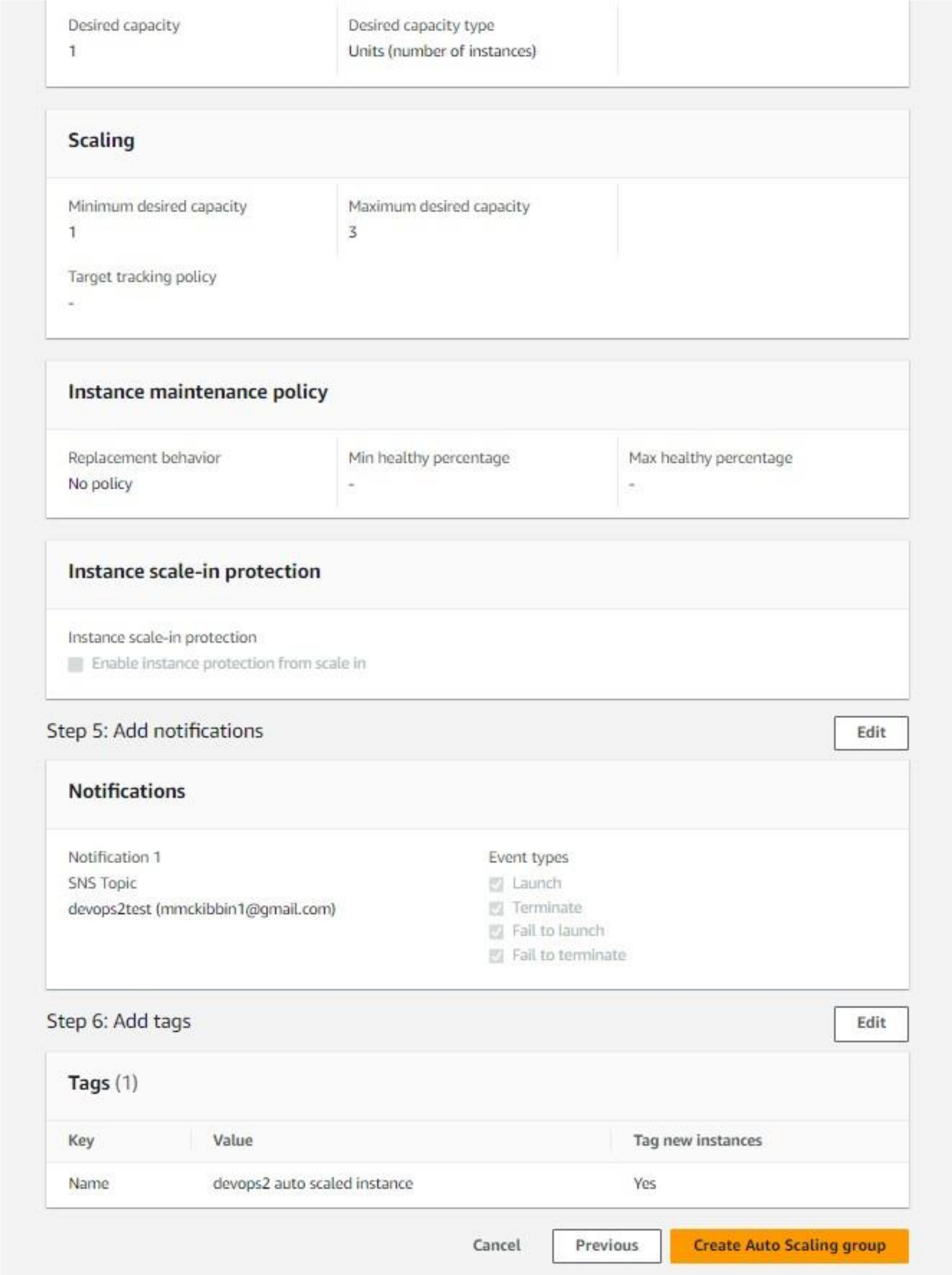
This Auto Scaling group will adhere to the launch template.

Step 3: Configure advanced options

**Load balancing**

Load balancer 1

*Figure 25 Auto Scaling - Choosing a Launch Template and Network.*



| | |
|---|---|
| Desired capacity<br>1 | Desired capacity type<br>Units (number of instances) |

**Scaling**

| | |
|---|---|
| Minimum desired capacity<br>1 | Maximum desired capacity<br>3 |
| Target tracking policy<br>- | |

**Instance maintenance policy**

| | | |
|---|---|---|
| Replacement behavior<br>No policy | Min healthy percentage<br>- | Max healthy percentage<br>- |

**Instance scale-in protection**

Instance scale-in protection

□ Enable instance protection from scale in

Step 5: Add notifications

**Notifications**

Notification 1
SNS Topic
devops2test (mmckibbin1@gmail.com)

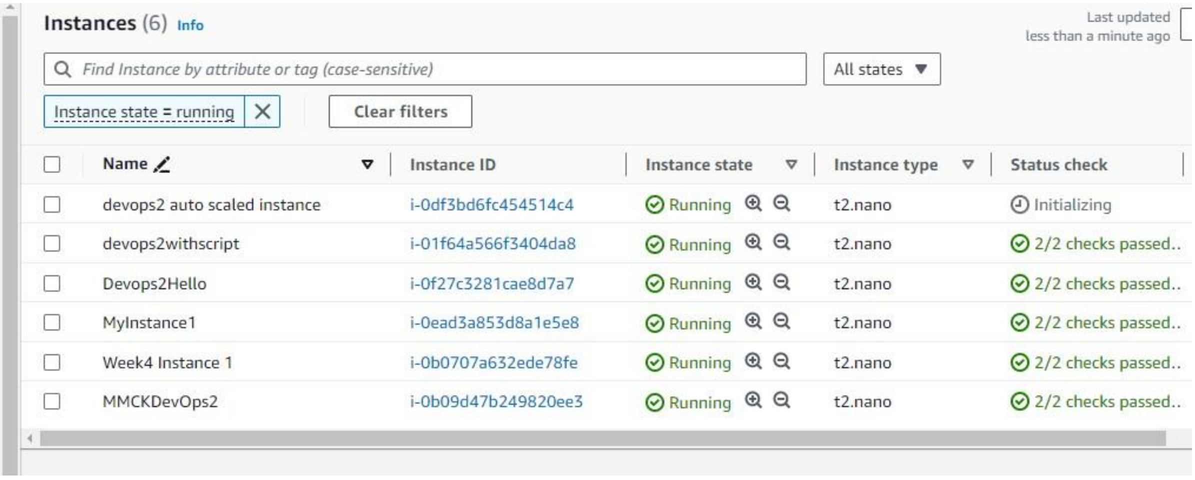
Event types
☑ Launch
☑ Terminate
☑ Fail to launch
☑ Fail to terminate

Step 6: Add tags

**Tags** (1)

| Key | Value | Tag new instances |
|---|---|---|
| Name | devops2 auto scaled instance | Yes |

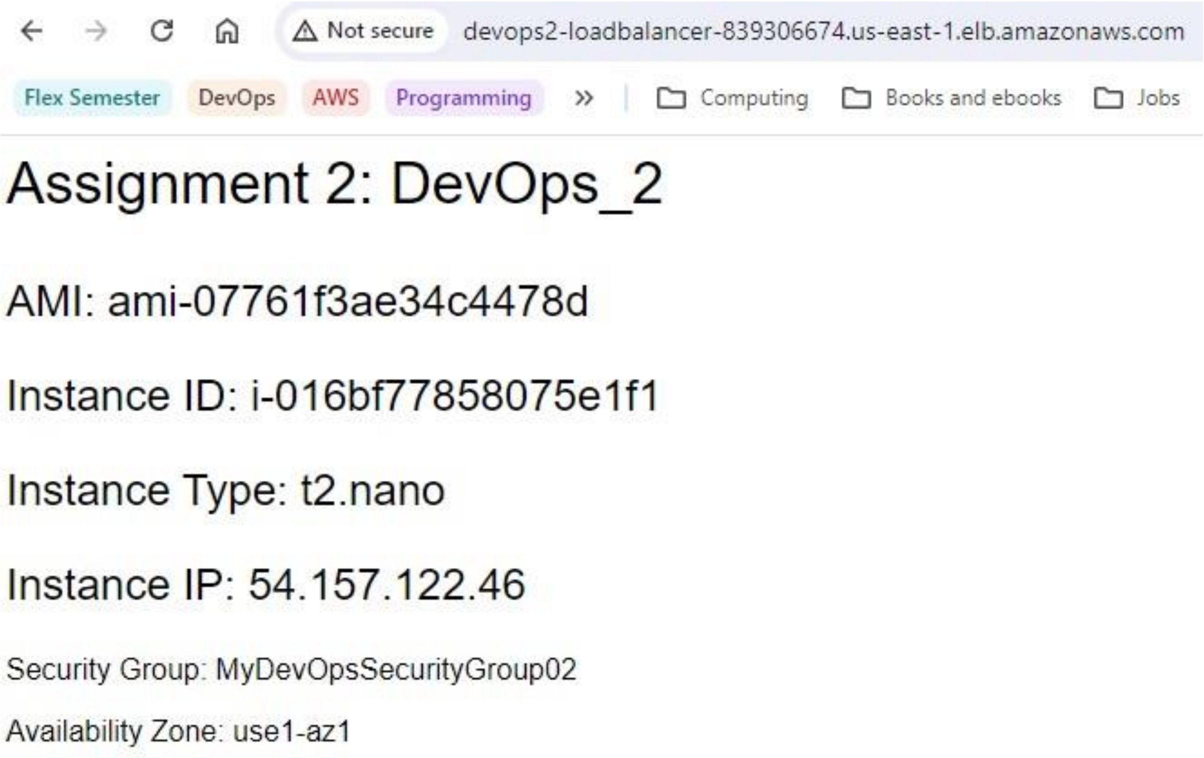
*Figure 26 Auto Scaling -Maximum instances, SNS Notifications, and Tags.*



Instances (6) Info
Last updated less than a minute ago
Find Instance by attribute or tag (case-sensitive)
All states ▼
Instance state = running ✕
Clear filters

| □ | Name | Instance ID | Instance state | Instance type | Status check |
|---|---|---|---|---|---|
| □ | devops2 auto scaled instance | i-0df3bd6fc454514c4 | ✓ Running | t2.nano | Initializing |
| □ | devops2withscript | i-01f64a566f3404da8 | ✓ Running | t2.nano | ✓ 2/2 checks passed.. |
| □ | Devops2Hello | i-0f27c3281cae8d7a7 | ✓ Running | t2.nano | ✓ 2/2 checks passed.. |
| □ | MyInstance1 | i-0ead3a853d8a1e5e8 | ✓ Running | t2.nano | ✓ 2/2 checks passed.. |
| □ | Week4 Instance 1 | i-0b0707a632ede78fe | ✓ Running | t2.nano | ✓ 2/2 checks passed.. |
| □ | MMCKDevOps2 | i-0b09d47b249820ee3 | ✓ Running | t2.nano | ✓ 2/2 checks passed.. |

*Figure 27 Auto Scaling - First Auto Scaled Instance.*



Assignment 2: DevOps_2

AMI: ami-07761f3ae34c4478d

Instance ID: i-016bf77858075e1f1

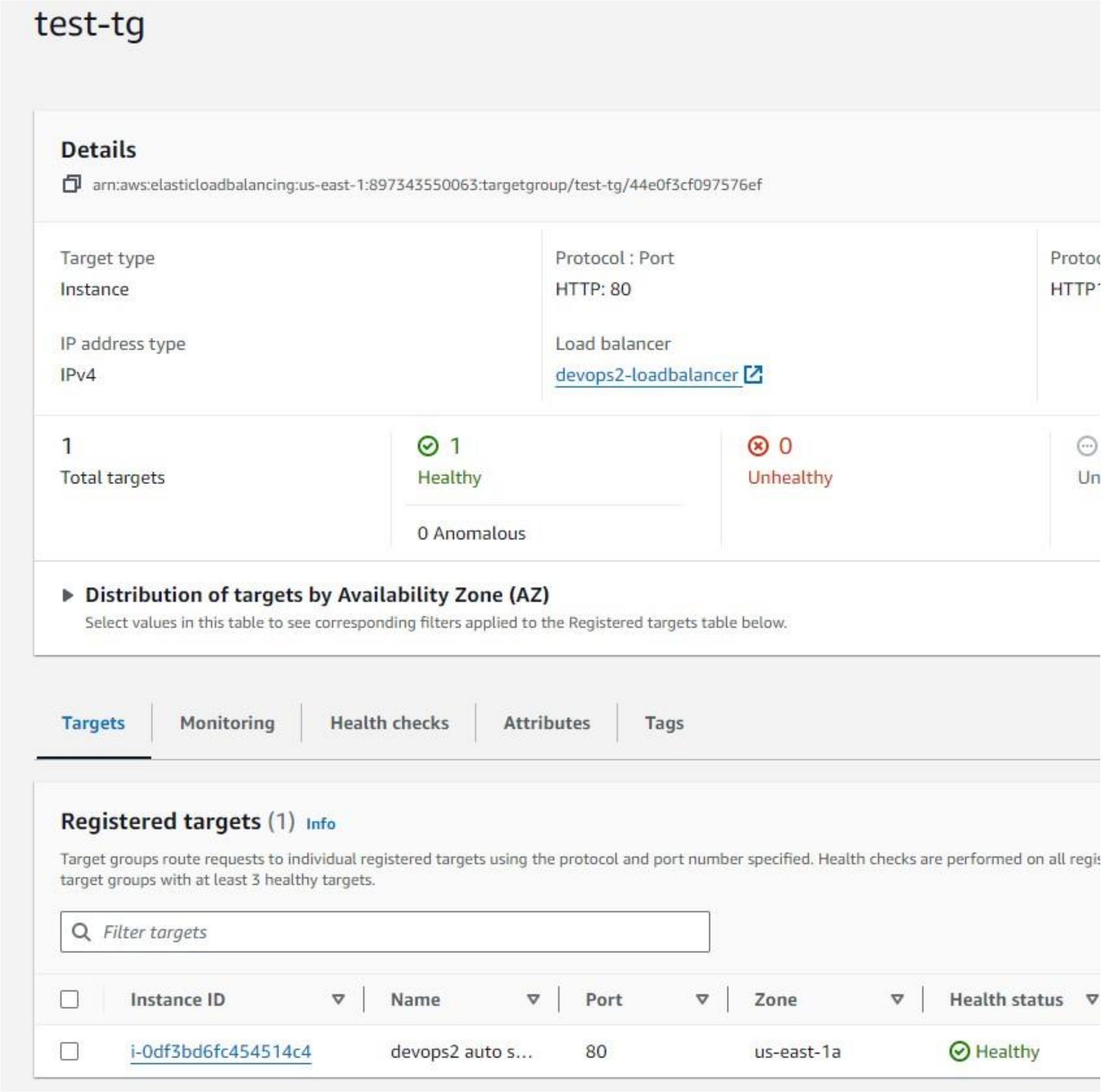
Instance Type: t2.nano

Instance IP: 54.157.122.46

Security Group: MyDevOpsSecurityGroup02

Availability Zone: use1-az1

*Figure 28 Auto Scaling - First Instance Running.*



test-tg

Details

arn:aws:elasticloadbalancing:us-east-1:897343550063:targetgroup/test-tg/44e0f3cf097576ef

Target type
Instance

IP address type
IPv4

Protocol : Port
HTTP: 80

Load balancer
devops2-loadbalancer

Protoc
HTTP

1
Total targets

✓ 1
Healthy

0 Anomalous

⊗ 0
Unhealthy

Un

▶ Distribution of targets by Availability Zone (AZ)
Select values in this table to see corresponding filters applied to the Registered targets table below.

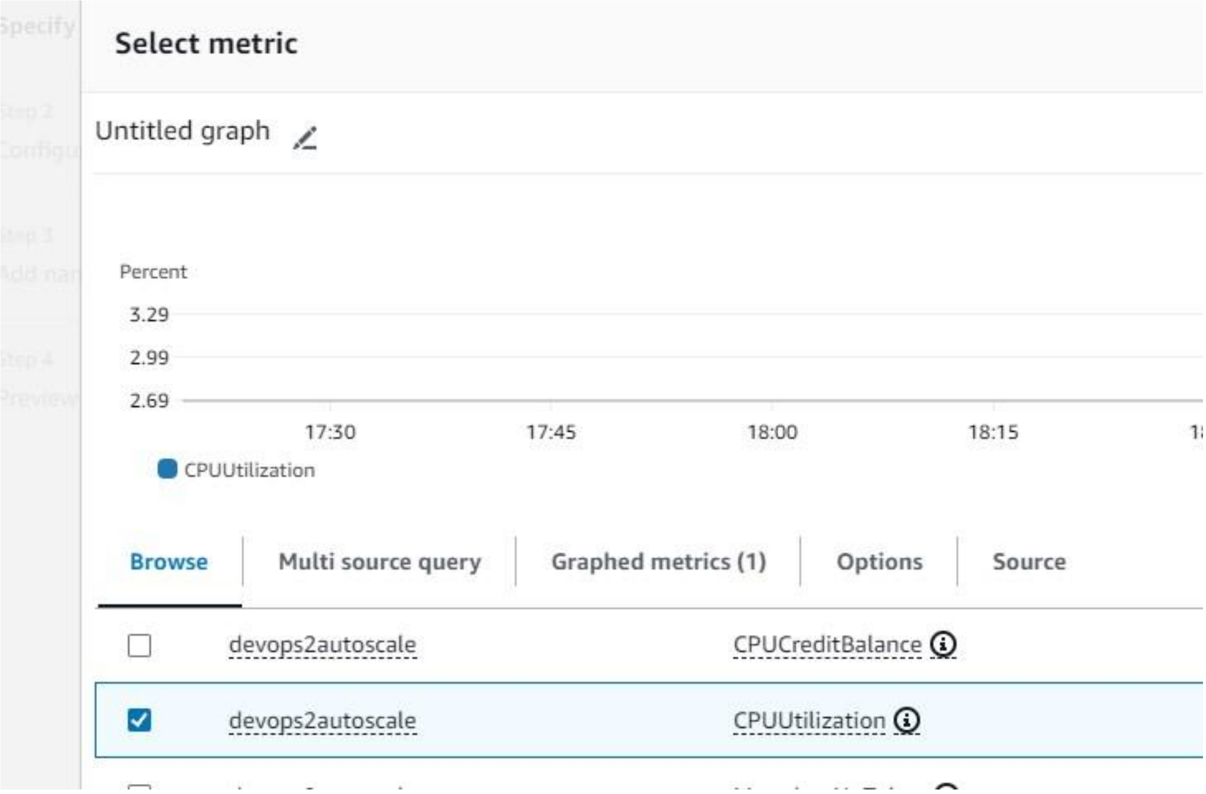
Registered targets (1) Info

Target groups route requests to individual registered targets using the protocol and port number specified. Health checks are performed on all regis
target groups with at least 3 healthy targets.

Filter targets

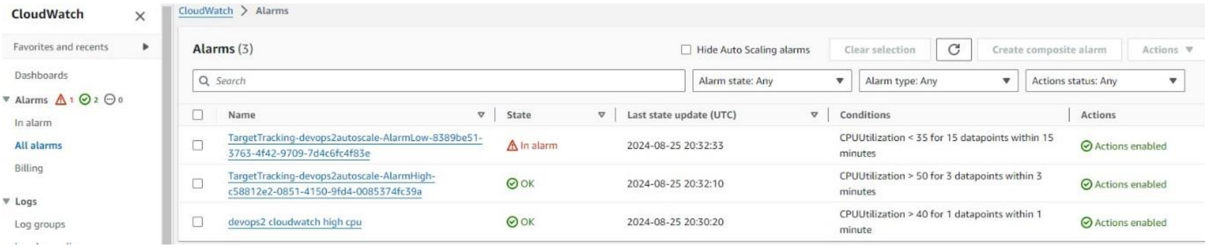
| □ | Instance ID | Name | Port | Zone | Health status |
|---|---|---|---|---|---|
| □ | i-0df3bd6fc454514c4 | devops2 auto s... | 80 | us-east-1a | ✓ Healthy |

*Figure 29 Auto Scaling - Targets registered.*



| | | |
|---|---|---|
| □ | devops2autoscale | CPUCreditBalance |
| ☑ | devops2autoscale | CPUUtilization |

*Figure 30 Auto Scaling - Cloudwatch Metrics.*



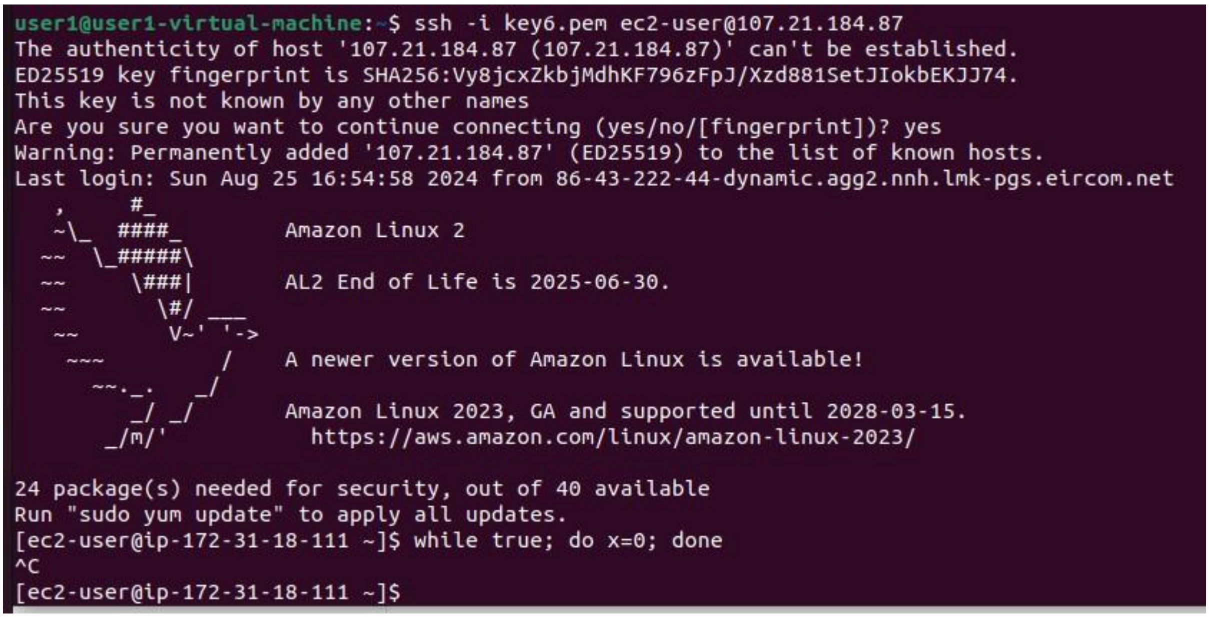
Alarms (3)

| □ | Name | State | Last state update (UTC) | Conditions | Actions |
|---|---|---|---|---|---|
| □ | TargetTracking-devops2autoscale-AlarmLow-8389be51-3763-4f42-9709-7d4c6fc4f83e | ⚠ In alarm | 2024-08-25 20:32:33 | CPUUtilization < 35 for 15 datapoints within 15 minutes | ✓ Actions enabled |
| □ | TargetTracking-devops2autoscale-AlarmHigh-c58812e2-0851-4150-9fd4-0085374fc39a | ✓ OK | 2024-08-25 20:32:10 | CPUUtilization > 50 for 3 datapoints within 3 minutes | ✓ Actions enabled |
| □ | devops2 cloudwatch high cpu | ✓ OK | 2024-08-25 20:30:20 | CPUUtilization > 40 for 1 datapoints within 1 minute | ✓ Actions enabled |

*Figure 31 Auto Scaling - Cloudwatch Alarms in progress.*

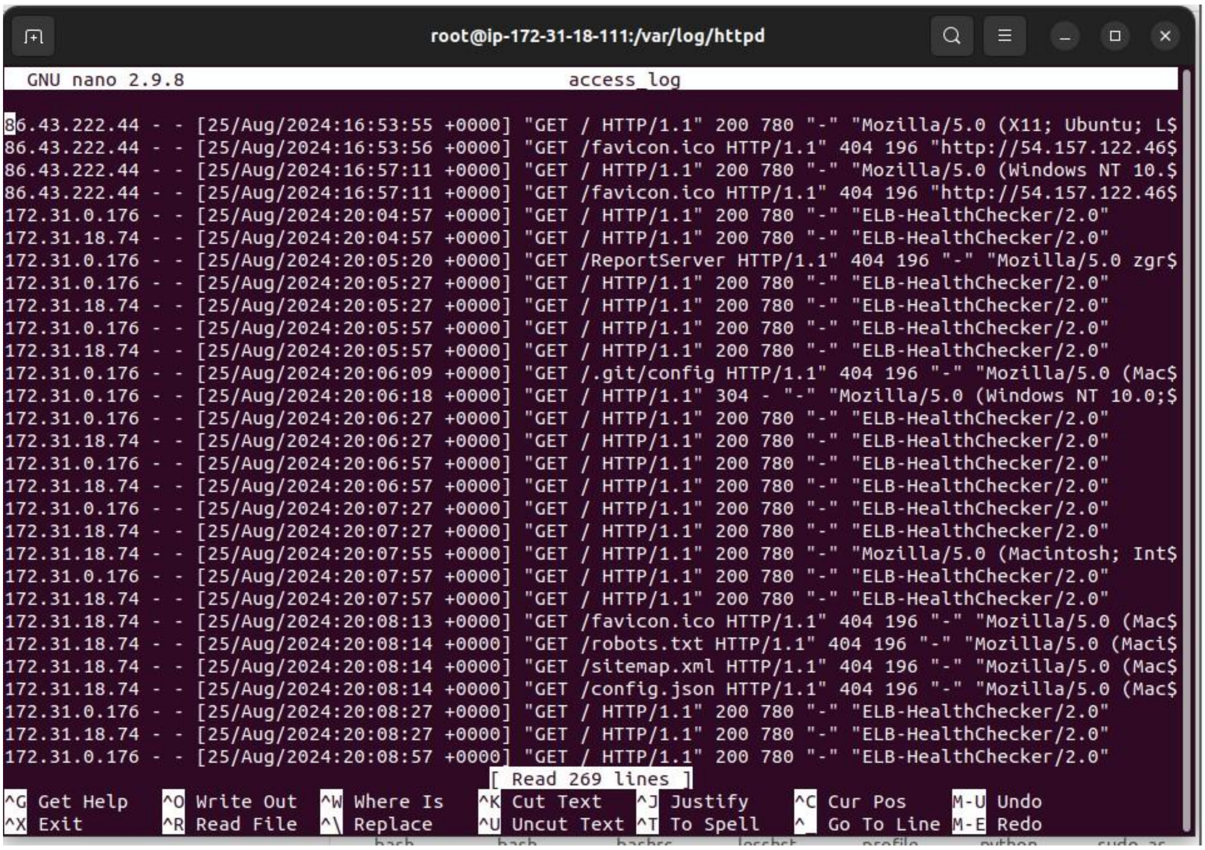
# Testing: Test traffic and load balancing demonstration.

Generation of test traffic to the load balancer.
Using an infinite loop over SSH.



```
user1@user1-virtual-machine:~$ ssh -i key6.pem ec2-user@107.21.184.87
The authenticity of host '107.21.184.87 (107.21.184.87)' can't be established.
ED25519 key fingerprint is SHA256:Vy8jcxZkbjMdhKF796zFpJ/Xzd881SetJIokbEKJJ74.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '107.21.184.87' (ED25519) to the list of known hosts.
Last login: Sun Aug 25 16:54:58 2024 from 86-43-222-44-dynamic.agg2.nnh.lmk-pgs.eircom.net
   ,     #_
   ~\_  ####_        Amazon Linux 2
  ~~  \_#####\
  ~~     \###|       AL2 End of Life is 2025-06-30.
  ~~       \#/ ___
   ~~       V~' '->
    ~~~         /    A newer version of Amazon Linux is available!
      ~~._.   _/
         _/ _/       Amazon Linux 2023, GA and supported until 2028-03-15.
       _/m/'           https://aws.amazon.com/linux/amazon-linux-2023/

24 package(s) needed for security, out of 40 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-18-111 ~]$ while true; do x=0; done
^C
[ec2-user@ip-172-31-18-111 ~]$
```

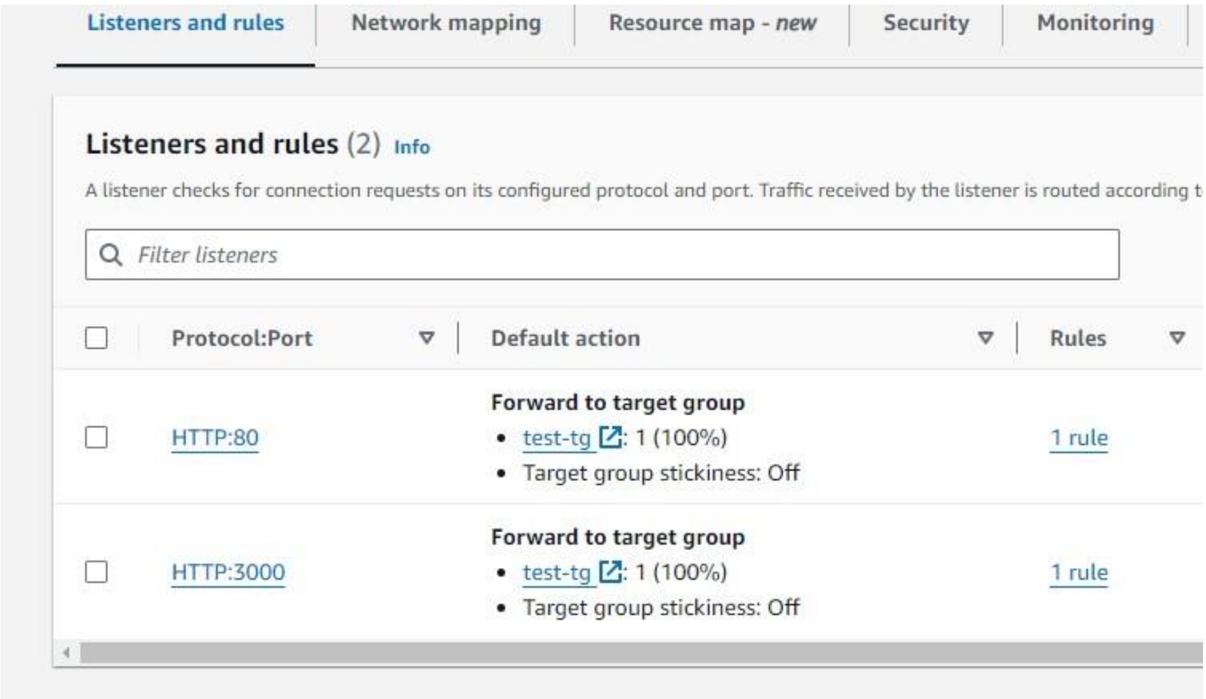
*Figure 32 Auto Scaling - Using an infinite loop to create traffic.*



root@ip-172-31-18-111:/var/log/httpd

```
  GNU nano 2.9.8                                    access_log

86.43.222.44 - - [25/Aug/2024:16:53:55 +0000] "GET / HTTP/1.1" 200 780 "-" "Mozilla/5.0 (X11; Ubuntu; L$
86.43.222.44 - - [25/Aug/2024:16:53:56 +0000] "GET /favicon.ico HTTP/1.1" 404 196 "http://54.157.122.46$
86.43.222.44 - - [25/Aug/2024:16:57:11 +0000] "GET / HTTP/1.1" 200 780 "-" "Mozilla/5.0 (Windows NT 10.$
86.43.222.44 - - [25/Aug/2024:16:57:11 +0000] "GET /favicon.ico HTTP/1.1" 404 196 "http://54.157.122.46$
172.31.0.176 - - [25/Aug/2024:20:04:57 +0000] "GET / HTTP/1.1" 200 780 "-" "ELB-HealthChecker/2.0"
172.31.18.74 - - [25/Aug/2024:20:04:57 +0000] "GET / HTTP/1.1" 200 780 "-" "ELB-HealthChecker/2.0"
172.31.0.176 - - [25/Aug/2024:20:05:20 +0000] "GET /ReportServer HTTP/1.1" 404 196 "-" "Mozilla/5.0 zgr$
172.31.0.176 - - [25/Aug/2024:20:05:27 +0000] "GET / HTTP/1.1" 200 780 "-" "ELB-HealthChecker/2.0"
172.31.18.74 - - [25/Aug/2024:20:05:27 +0000] "GET / HTTP/1.1" 200 780 "-" "ELB-HealthChecker/2.0"
172.31.0.176 - - [25/Aug/2024:20:05:57 +0000] "GET / HTTP/1.1" 200 780 "-" "ELB-HealthChecker/2.0"
172.31.18.74 - - [25/Aug/2024:20:05:57 +0000] "GET / HTTP/1.1" 200 780 "-" "ELB-HealthChecker/2.0"
172.31.0.176 - - [25/Aug/2024:20:06:09 +0000] "GET /.git/config HTTP/1.1" 404 196 "-" "Mozilla/5.0 (Mac$
172.31.0.176 - - [25/Aug/2024:20:06:18 +0000] "GET / HTTP/1.1" 304 - "-" "Mozilla/5.0 (Windows NT 10.0;$
172.31.0.176 - - [25/Aug/2024:20:06:27 +0000] "GET / HTTP/1.1" 200 780 "-" "ELB-HealthChecker/2.0"
172.31.18.74 - - [25/Aug/2024:20:06:27 +0000] "GET / HTTP/1.1" 200 780 "-" "ELB-HealthChecker/2.0"
172.31.0.176 - - [25/Aug/2024:20:06:57 +0000] "GET / HTTP/1.1" 200 780 "-" "ELB-HealthChecker/2.0"
172.31.18.74 - - [25/Aug/2024:20:06:57 +0000] "GET / HTTP/1.1" 200 780 "-" "ELB-HealthChecker/2.0"
172.31.0.176 - - [25/Aug/2024:20:07:27 +0000] "GET / HTTP/1.1" 200 780 "-" "ELB-HealthChecker/2.0"
172.31.18.74 - - [25/Aug/2024:20:07:27 +0000] "GET / HTTP/1.1" 200 780 "-" "ELB-HealthChecker/2.0"
172.31.18.74 - - [25/Aug/2024:20:07:55 +0000] "GET / HTTP/1.1" 200 780 "-" "Mozilla/5.0 (Macintosh; Int$
172.31.0.176 - - [25/Aug/2024:20:07:57 +0000] "GET / HTTP/1.1" 200 780 "-" "ELB-HealthChecker/2.0"
172.31.18.74 - - [25/Aug/2024:20:07:57 +0000] "GET / HTTP/1.1" 200 780 "-" "ELB-HealthChecker/2.0"
172.31.18.74 - - [25/Aug/2024:20:08:13 +0000] "GET /favicon.ico HTTP/1.1" 404 196 "-" "Mozilla/5.0 (Mac$
172.31.18.74 - - [25/Aug/2024:20:08:14 +0000] "GET /robots.txt HTTP/1.1" 404 196 "-" "Mozilla/5.0 (Maci$
172.31.18.74 - - [25/Aug/2024:20:08:14 +0000] "GET /sitemap.xml HTTP/1.1" 404 196 "-" "Mozilla/5.0 (Mac$
172.31.18.74 - - [25/Aug/2024:20:08:14 +0000] "GET /config.json HTTP/1.1" 404 196 "-" "Mozilla/5.0 (Mac$
172.31.0.176 - - [25/Aug/2024:20:08:27 +0000] "GET / HTTP/1.1" 200 780 "-" "ELB-HealthChecker/2.0"
172.31.18.74 - - [25/Aug/2024:20:08:27 +0000] "GET / HTTP/1.1" 200 780 "-" "ELB-HealthChecker/2.0"
172.31.0.176 - - [25/Aug/2024:20:08:57 +0000] "GET / HTTP/1.1" 200 780 "-" "ELB-HealthChecker/2.0"
                                       [ Read 269 lines ]
^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos     M-U Undo
^X Exit        ^R Read File   ^\ Replace     ^U Uncut Text  ^T To Spell    ^_ Go To Line  M-E Redo
```

*Figure 33 Curl command generates 100 requests.*

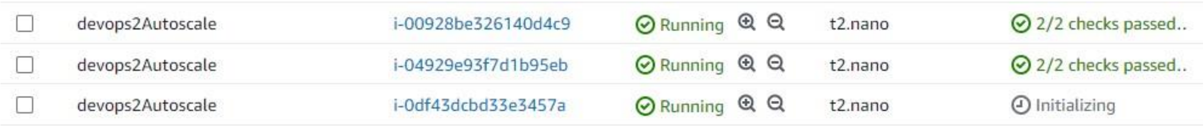


**Listeners and rules** (2) Info

A listener checks for connection requests on its configured protocol and port. Traffic received by the listener is routed according t

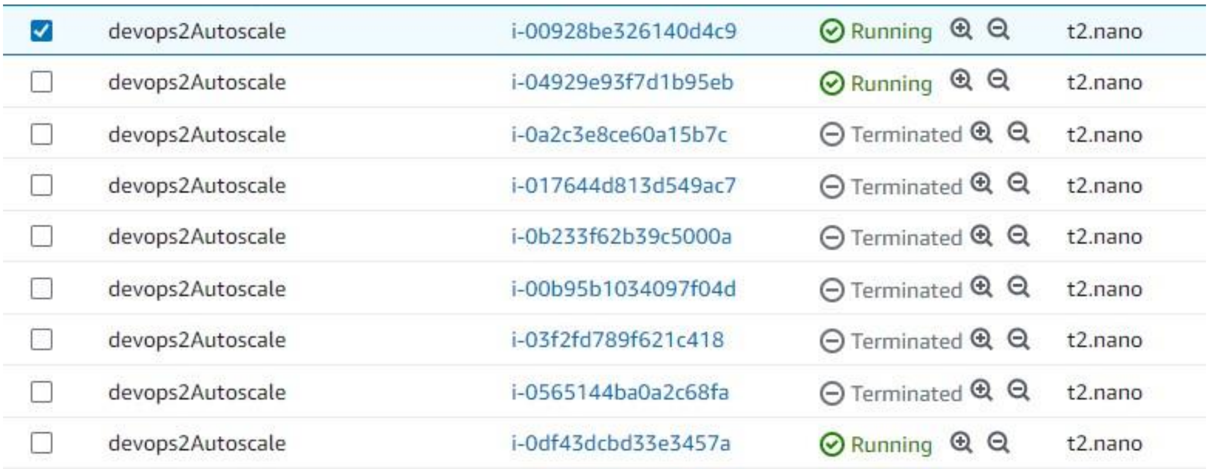
| □ | Protocol:Port | Default action | Rules |
|---|---|---|---|
| □ | HTTP:80 | **Forward to target group**<br>• test-tg: 1 (100%)<br>• Target group stickiness: Off | 1 rule |
| □ | HTTP:3000 | **Forward to target group**<br>• test-tg: 1 (100%)<br>• Target group stickiness: Off | 1 rule |

*Figure 34 Adding Listeners on port 80 and port 3000.*



| | | | | | |
|---|---|---|---|---|---|
| □ | devops2Autoscale | i-00928be326140d4c9 | ✓ Running | t2.nano | ✓ 2/2 checks passed.. |
| □ | devops2Autoscale | i-04929e93f7d1b95eb | ✓ Running | t2.nano | ✓ 2/2 checks passed.. |
| □ | devops2Autoscale | i-0df43dcbd33e3457a | ✓ Running | t2.nano | Initializing |

*Figure 35Auto Scaled instances running.*



| | | | | |
|---|---|---|---|---|
| ☑ | devops2Autoscale | i-00928be326140d4c9 | ✓ Running | t2.nano |
| □ | devops2Autoscale | i-04929e93f7d1b95eb | ✓ Running | t2.nano |
| □ | devops2Autoscale | i-0a2c3e8ce60a15b7c | ⊖ Terminated | t2.nano |
| □ | devops2Autoscale | i-017644d813d549ac7 | ⊖ Terminated | t2.nano |
| □ | devops2Autoscale | i-0b233f62b39c5000a | ⊖ Terminated | t2.nano |
| □ | devops2Autoscale | i-00b95b1034097f04d | ⊖ Terminated | t2.nano |
| □ | devops2Autoscale | i-03f2fd789f621c418 | ⊖ Terminated | t2.nano |
| □ | devops2Autoscale | i-0565144ba0a2c68fa | ⊖ Terminated | t2.nano |
| □ | devops2Autoscale | i-0df43dcbd33e3457a | ✓ Running | t2.nano |

*Figure 36 Auto Scaled instances - Running and Terminated.*

# Monitoring: Own script.

I used my own python program, devops_2.py, adapted from assignment 1, to launch an instance and set up Cloudwatch alarms. I also used it to load and run further scripts, monitor.sh, and memv2.sh, to monitor and report on the custom metrics on the servers and push these to CloudWatch.

# APPENDIX A: Automated instance setup.

Including launching and configuring instance based on AMI.

Setting Key Pair and Security group.

Create webpage with content and display in a browser window.

Get instance details.

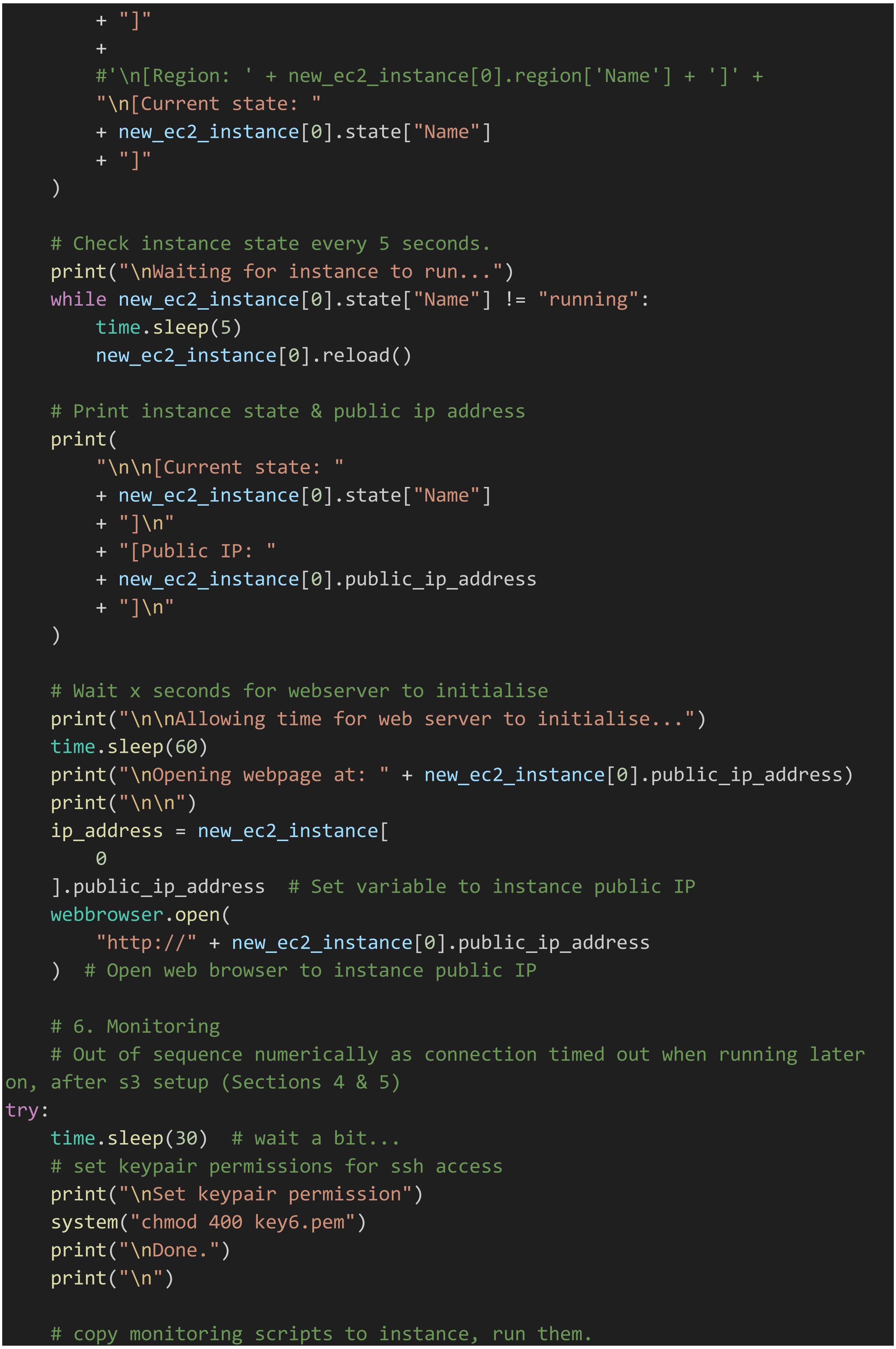
Install and run monitoring scripts – see Appendices B, and C.

Install Node.js, copy app.js, and run app in a browser window.

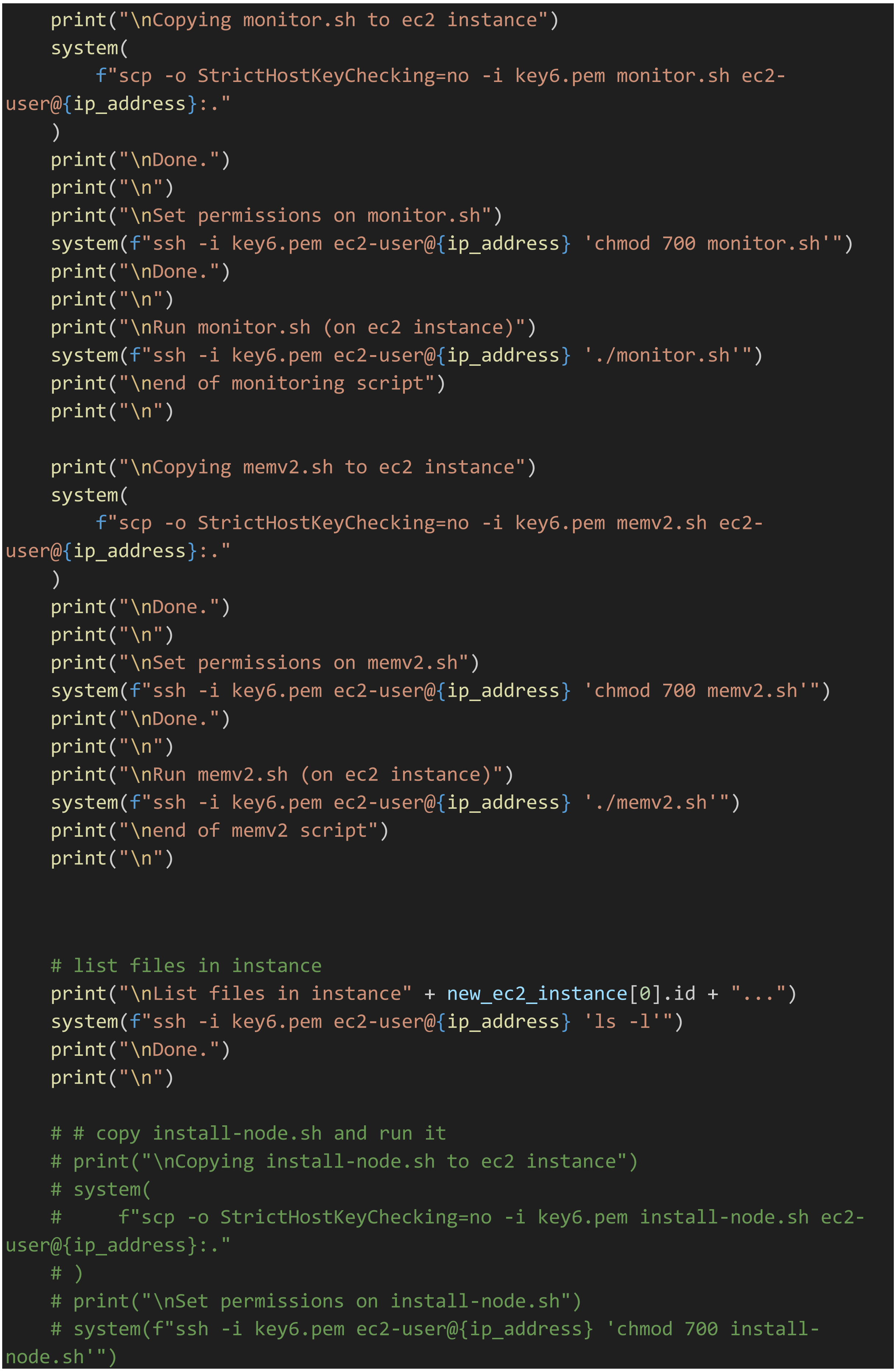
Set up Cloudwatch Alarms

```
import boto3
import time
import webbrowser
import random
import string
import urllib
import os
from cgitb import html
from ipaddress import ip_address
from os import system
from urllib import response
from botocore.exceptions import ClientError
import json

# Launch New EC2 Instance
# Configure instance settings
# Set up EC2 website
# Download and install Apache web server
# Create index.html and add content
# Get meta data
# Get image
# copy index.html to local drive

ec2 = boto3.resource("ec2")
try:
    print("\nCreating new EC2 instance\n")

    new_ec2_instance = ec2.create_instances(
        # Amazon Linux2 AMI (check current available AMIs before assignment
submission)
        ImageId="ami-0ced9049444b57252",
        # How many instances to launch. Min and Max
        MinCount=1,
        MaxCount=3,
        # Instance type (t2.nano)
        InstanceType="t2.nano",
        # Tag the instance
        TagSpecifications=[
            {
                "ResourceType": "instance",
                "Tags": [
                    {"Key": "Name", "Value": "devops2Autoscale"},
                    {"Key": "Owner", "Value": "mmckibbin"},
                ],
            },
        ],
        # MyDevOpsSecurityGroup01 Security group ID
        SecurityGroupIds=["sg-08506a9bf41ce7f35"],
        KeyName="key6",
        UserData="""#!/bin/bash
            yum install httpd -y
            systemctl enable httpd
            systemctl start httpd

            echo "Content-type: text/html"
            echo '<html>' > index.html
            echo '<head>' >> index.html
            echo '<title>DevOps Assignment 1</title>' >> index.html
            echo '</head>' >> index.html
            echo '<body>' >> index.html

            echo '<p style="font-family:Helvetica, sans-serif; font-
size:200%;">Assignment 2: DevOps_2</p>' >> index.html

            echo '<p style="font-family:Helvetica, sans-serif; font-
size:150%;">AMI: ' >> index.html
            echo $(curl http://169.254.169.254/latest/meta-data/ami-id) >>
index.html
            echo '</p>' >> index.html

            echo '<p style="font-family:Helvetica, sans-serif; font-
size:150%;">Instance ID: ' >> index.html
            echo $(curl http://169.254.169.254/latest/meta-data/instance-id)
>> index.html
            echo '</p>' >> index.html

            echo '<p style="font-family:Helvetica, sans-serif; font-
size:150%;">Instance Type: ' >> index.html
```

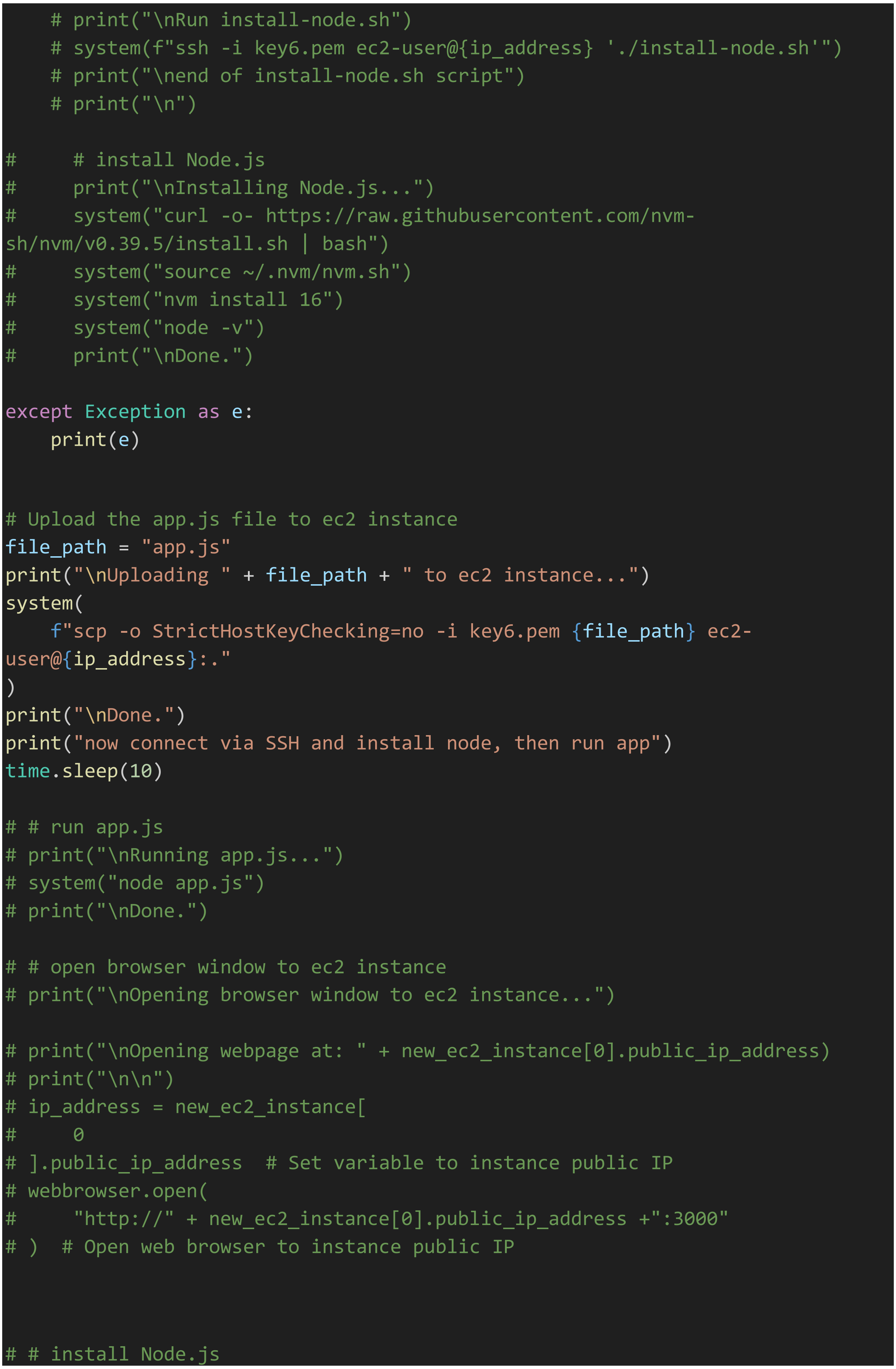
```
            echo $(curl http://169.254.169.254/latest/meta-data/instance-type) >> index.html
            echo '</p>' >> index.html

            echo '<p style="font-family:Helvetica, sans-serif; font-size:150%;">Instance IP: ' >> index.html
            echo $(curl http://169.254.169.254/latest/meta-data/public-ipv4) >> index.html
            echo '</p>' >> index.html

            echo '<p style="font-family:Helvetica, sans-serif;">Security Group: ' >> index.html
            echo $(curl http://169.254.169.254/latest/meta-data/security-groups) >> index.html
            echo '</p>' >> index.html

            echo '<p style="font-family:Helvetica, sans-serif;">Availability Zone: ' >> index.html
            echo $(curl http://169.254.169.254/latest/meta-data/placement/availability-zone-id ) >> index.html
            echo '</p>' >> index.html

            echo '<img src="http://devops.witdemo.net/logo.jpg"> logo.jpg ' >> index.html
            echo '</body>' >> index.html
            echo '</html>' >> index.html

            cp index.html /var/www/html/index.html
        """,
    )

except Exception as e:
    print("Error! \nThe EC2 creation process has encountered an error.\n")
    print(e)
    errorfile = open("error.log", "w")
    errorfile.write(str(e))
    errorfile.close()
    print("See error.log for details.")

else:
    # Print instance ID, type, & state
    print(
        "\nNew EC2 instance created successfully!"
        + "\n[ID: "
        + new_ec2_instance[0].id
        + "]"
        + "\n[Type: "
        + new_ec2_instance[0].instance_type
```



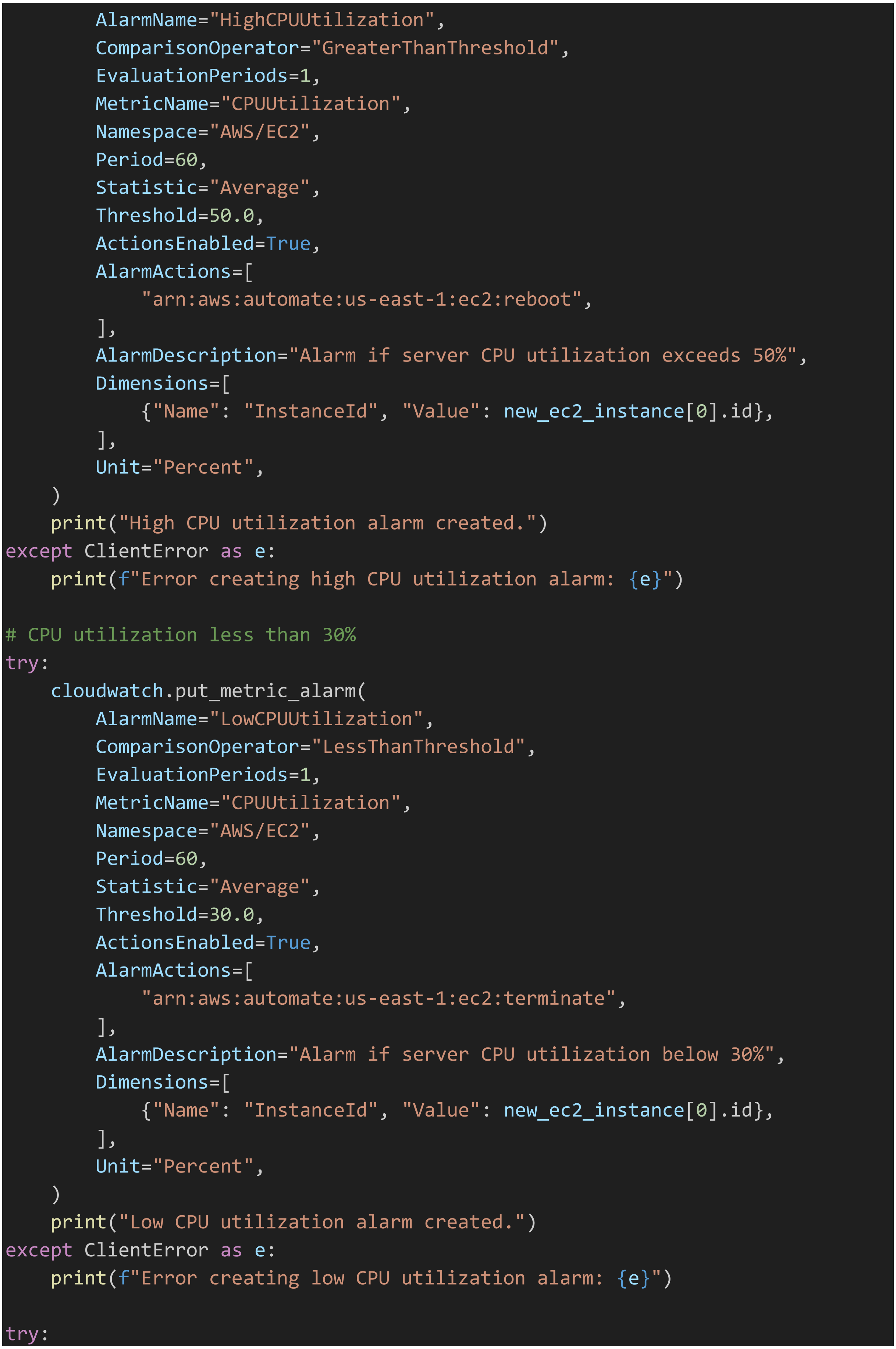
```
        + "]"
        +
        #'\n[Region: ' + new_ec2_instance[0].region['Name'] + ']' +
        "\n[Current state: "
        + new_ec2_instance[0].state["Name"]
        + "]"
    )

    # Check instance state every 5 seconds.
    print("\nWaiting for instance to run...")
    while new_ec2_instance[0].state["Name"] != "running":
        time.sleep(5)
        new_ec2_instance[0].reload()

    # Print instance state & public ip address
    print(
        "\n\n[Current state: "
        + new_ec2_instance[0].state["Name"]
        + "]\n"
        + "[Public IP: "
        + new_ec2_instance[0].public_ip_address
        + "]\n"
    )

    # Wait x seconds for webserver to initialise
    print("\n\nAllowing time for web server to initialise...")
    time.sleep(60)
    print("\nOpening webpage at: " + new_ec2_instance[0].public_ip_address)
    print("\n\n")
    ip_address = new_ec2_instance[
        0
    ].public_ip_address  # Set variable to instance public IP
    webbrowser.open(
        "http://" + new_ec2_instance[0].public_ip_address
    )  # Open web browser to instance public IP

    # 6. Monitoring
    # Out of sequence numerically as connection timed out when running later
on, after s3 setup (Sections 4 & 5)
try:
    time.sleep(30)  # wait a bit...
    # set keypair permissions for ssh access
    print("\nSet keypair permission")
    system("chmod 400 key6.pem")
    print("\nDone.")
    print("\n")

    # copy monitoring scripts to instance, run them.
```



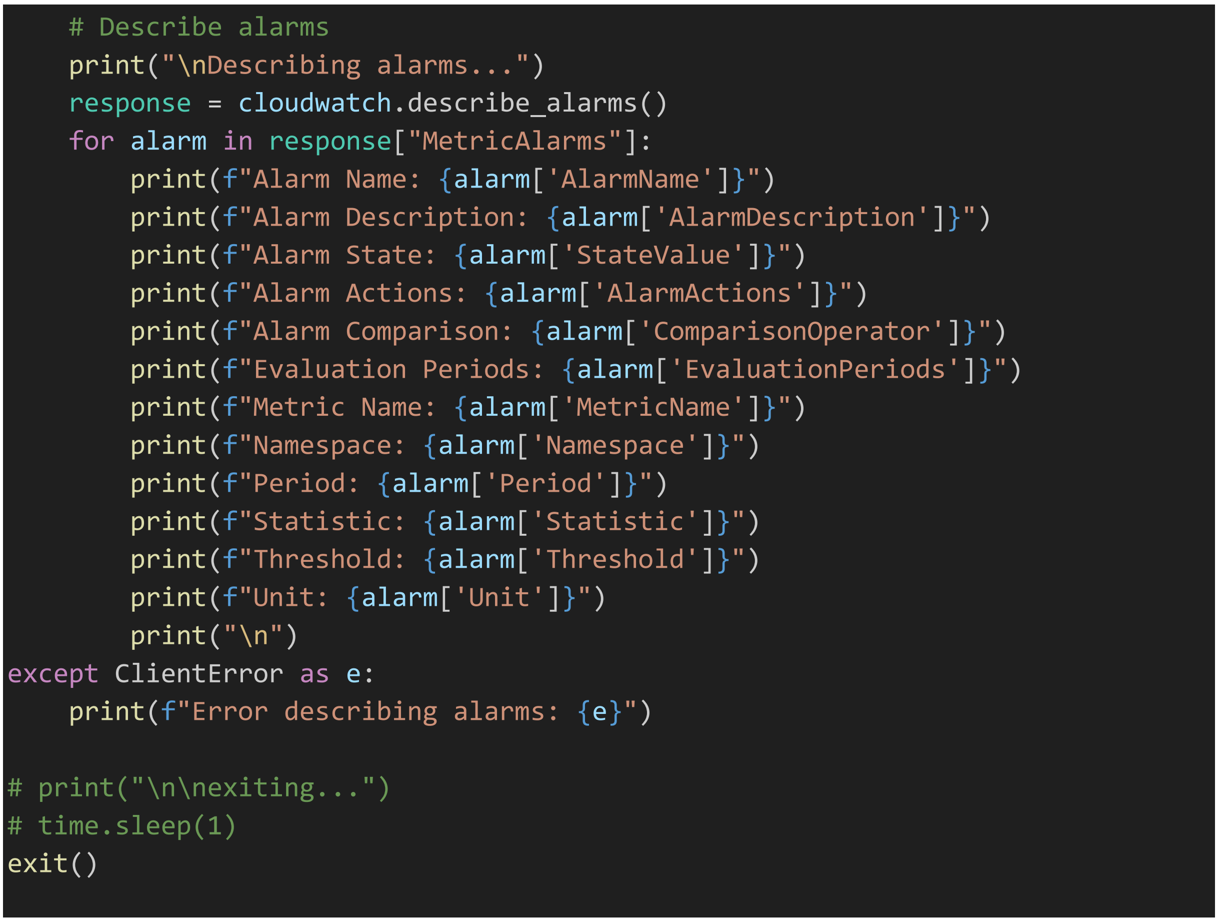
```
    print("\nCopying monitor.sh to ec2 instance")
    system(
        f"scp -o StrictHostKeyChecking=no -i key6.pem monitor.sh ec2-user@{ip_address}:."
    )
    print("\nDone.")
    print("\n")
    print("\nSet permissions on monitor.sh")
    system(f"ssh -i key6.pem ec2-user@{ip_address} 'chmod 700 monitor.sh'")
    print("\nDone.")
    print("\n")
    print("\nRun monitor.sh (on ec2 instance)")
    system(f"ssh -i key6.pem ec2-user@{ip_address} './monitor.sh'")
    print("\nend of monitoring script")
    print("\n")

    print("\nCopying memv2.sh to ec2 instance")
    system(
        f"scp -o StrictHostKeyChecking=no -i key6.pem memv2.sh ec2-user@{ip_address}:."
    )
    print("\nDone.")
    print("\n")
    print("\nSet permissions on memv2.sh")
    system(f"ssh -i key6.pem ec2-user@{ip_address} 'chmod 700 memv2.sh'")
    print("\nDone.")
    print("\n")
    print("\nRun memv2.sh (on ec2 instance)")
    system(f"ssh -i key6.pem ec2-user@{ip_address} './memv2.sh'")
    print("\nend of memv2 script")
    print("\n")



    # list files in instance
    print("\nList files in instance" + new_ec2_instance[0].id + "...")
    system(f"ssh -i key6.pem ec2-user@{ip_address} 'ls -l'")
    print("\nDone.")
    print("\n")

    # # copy install-node.sh and run it
    # print("\nCopying install-node.sh to ec2 instance")
    # system(
    #     f"scp -o StrictHostKeyChecking=no -i key6.pem install-node.sh ec2-user@{ip_address}:."
    # )
    # print("\nSet permissions on install-node.sh")
    # system(f"ssh -i key6.pem ec2-user@{ip_address} 'chmod 700 install-node.sh'")
```



```
    # print("\nRun install-node.sh")
    # system(f"ssh -i key6.pem ec2-user@{ip_address} './install-node.sh'")
    # print("\nend of install-node.sh script")
    # print("\n")

#     # install Node.js
#     print("\nInstalling Node.js...")
#     system("curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.5/install.sh | bash")
#     system("source ~/.nvm/nvm.sh")
#     system("nvm install 16")
#     system("node -v")
#     print("\nDone.")

except Exception as e:
    print(e)


# Upload the app.js file to ec2 instance
file_path = "app.js"
print("\nUploading " + file_path + " to ec2 instance...")
system(
    f"scp -o StrictHostKeyChecking=no -i key6.pem {file_path} ec2-user@{ip_address}:."
)
print("\nDone.")
print("now connect via SSH and install node, then run app")
time.sleep(10)

# # run app.js
# print("\nRunning app.js...")
# system("node app.js")
# print("\nDone.")

# # open browser window to ec2 instance
# print("\nOpening browser window to ec2 instance...")

# print("\nOpening webpage at: " + new_ec2_instance[0].public_ip_address)
# print("\n\n")
# ip_address = new_ec2_instance[
#     0
# ].public_ip_address  # Set variable to instance public IP
# webbrowser.open(
#     "http://" + new_ec2_instance[0].public_ip_address +":3000"
# )  # Open web browser to instance public IP



# # install Node.js
# print("\nInstalling Node.js...")
# system("curl -o- https://raw.githubusercontent.com/nvm-
sh/nvm/v0.39.5/install.sh | bash -")
# system("source ~/.nvm/nvm.sh")
# system("nvm install 16")
# #export nvm dir to bashrc
# system("export NVM_DIR=$HOME/.nvm")
# system("[ -s \"$NVM_DIR/nvm.sh\" ] && \. \"$NVM_DIR/nvm.sh\"")
# system("[ -s \"$NVM_DIR/bash_completion\" ] && \.
\"$NVM_DIR/bash_completion\"")
# system("nvm use 16")
# print("\ninstalling npm")
# system("npm install -g npm@latest")
# # get node version number
# print("\nNode version:")
# system("node -v")
# print("\nDone.")

# # Upload the app.js file
# file_path = "app.js"
# if os.path.exists(file_path):
#     try:
#         s3_client.upload_file(
#             file_path, bucket_name, "app.js", ExtraArgs={"ContentType":
"text/javascript"}
#         )
#         print(f"File {file_path} uploaded as app.js.")
#     except ClientError as e:
#         print(f"Error uploading {file_path}: {e}")
#         exit(1)
# else:
#     print(f"File {file_path} does not exist.")
#     exit(1)

# # run app.js
# print("\nRunning app.js...")
# system("node app.js")




#CloudWatch alarms setup

print("Setting up CloudWatch alarms...")
cloudwatch = boto3.client("cloudwatch")

# CPU utilization greater than 50%
try:
    cloudwatch.put_metric_alarm(
```



```
        AlarmName="HighCPUUtilization",
        ComparisonOperator="GreaterThanThreshold",
        EvaluationPeriods=1,
        MetricName="CPUUtilization",
        Namespace="AWS/EC2",
        Period=60,
        Statistic="Average",
        Threshold=50.0,
        ActionsEnabled=True,
        AlarmActions=[
            "arn:aws:automate:us-east-1:ec2:reboot",
        ],
        AlarmDescription="Alarm if server CPU utilization exceeds 50%",
        Dimensions=[
            {"Name": "InstanceId", "Value": new_ec2_instance[0].id},
        ],
        Unit="Percent",
    )
    print("High CPU utilization alarm created.")
except ClientError as e:
    print(f"Error creating high CPU utilization alarm: {e}")

# CPU utilization less than 30%
try:
    cloudwatch.put_metric_alarm(
        AlarmName="LowCPUUtilization",
        ComparisonOperator="LessThanThreshold",
        EvaluationPeriods=1,
        MetricName="CPUUtilization",
        Namespace="AWS/EC2",
        Period=60,
        Statistic="Average",
        Threshold=30.0,
        ActionsEnabled=True,
        AlarmActions=[
            "arn:aws:automate:us-east-1:ec2:terminate",
        ],
        AlarmDescription="Alarm if server CPU utilization below 30%",
        Dimensions=[
            {"Name": "InstanceId", "Value": new_ec2_instance[0].id},
        ],
        Unit="Percent",
    )
    print("Low CPU utilization alarm created.")
except ClientError as e:
    print(f"Error creating low CPU utilization alarm: {e}")

try:
```



```
    # Describe alarms
    print("\nDescribing alarms...")
    response = cloudwatch.describe_alarms()
    for alarm in response["MetricAlarms"]:
        print(f"Alarm Name: {alarm['AlarmName']}")
        print(f"Alarm Description: {alarm['AlarmDescription']}")
        print(f"Alarm State: {alarm['StateValue']}")
        print(f"Alarm Actions: {alarm['AlarmActions']}")
        print(f"Alarm Comparison: {alarm['ComparisonOperator']}")
        print(f"Evaluation Periods: {alarm['EvaluationPeriods']}")
        print(f"Metric Name: {alarm['MetricName']}")
        print(f"Namespace: {alarm['Namespace']}")
        print(f"Period: {alarm['Period']}")
        print(f"Statistic: {alarm['Statistic']}")
        print(f"Threshold: {alarm['Threshold']}")
        print(f"Unit: {alarm['Unit']}")
        print("\n")
except ClientError as e:
    print(f"Error describing alarms: {e}")

# print("\n\nexiting...")
# time.sleep(1)
exit()
```

# APPENDIX B: monitor.sh - Instance info.

```bash
#!/usr/bin/bash
# Michael MCKibbin 20092733
# Adapted from course provided script.
#
INSTANCE_ID=$(curl -s http://169.254.169.254/latest/meta-data/instance-id)
MEMORYUSAGE=$(free -m | awk 'NR==2{printf "%.2f%%", $3*100/$2 }')
PROCESSES=$(expr $(ps -A | grep -c .) - 1)
HTTPD_STATUS=$(systemctl status $1 | awk 'NR == 3')
HTTPD_UPTIME=$(uptime | awk '{print $3,$4}' | cut -d, -f1)
HTTPD_PROCESSES=$(ps -A | grep -c httpd)
APACHE_PROCESSES=$(ps -A | grep -c apache)
APACHE_PROCESSES_NoGREP=$(ps -ef | grep -v grep | grep -c apache)
HTTP_PORT=$(netstat -tuln | grep -c ":80")
HTTPS_PORT=$(netstat -tuln | grep -c ":443")
AVAILABILITY_ZONE=$(curl -s http://169.254.169.254/latest/meta-data/placement/availability-zone)
AMI_ID=$(curl -s http://169.254.169.254/latest/meta-data/ami-id)
SECURITY_GROUP=$(curl -s http://169.254.169.254/latest/meta-data/security-groups)
IPV4=$(curl -s http://169.254.169.254/latest/meta-data/public-ipv4)
SSH_PORT=$(netstat -tuln | grep -c ":22")

echo "Instance ID: $INSTANCE_ID"
echo "Memory utilisation: $MEMORYUSAGE"
if [ $HTTPD_PROCESSES -ge 1 ]
then
    echo "Web server is running"
else
    echo "Web server is NOT running"
fi

echo "No of HTTPD processes: $PROCESSES"
echo "HTTPD server status: $HTTPD_STATUS"
echo "HTTPD server uptime: $HTTPD_UPTIME"
echo "No of Apache processes: $APACHE_PROCESSES"
echo "No of Apache processes (No grep): $APACHE_PROCESSES_NoGREP"
echo "No of SSH connections: $SSH_PORT"
echo "No of HTTP connections: $HTTP_PORT"
echo "No of HTTPS connections: $HTTPS_PORT"
echo "Availability zone: $AVAILABILITY_ZONE"
echo "AMI ID: $AMI_ID"
echo "Security group: $SECURITY_GROUP"
echo "Public IPv4: $IPV4"
```

# APPENDIX C: memv2.sh - Cloudwatch.

```
#!/bin/bash
TOKEN=`curl -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-
metadata-token-ttl-seconds: 21600"`
INSTANCE_ID=$(curl -H "X-aws-ec2-metadata-token: $TOKEN"
http://169.254.169.254/latest/meta-data/instance-id)

#INSTANCE_ID=$(curl -s http://169.254.169.254/latest/meta-data/instance-id)
USEDMEMORY=$(free -m | awk 'NR==2{printf "%.2f\t", $3*100/$2 }')
TCP_CONN=$(netstat -an | wc -l)
TCP_CONN_PORT_80=$(netstat -an | grep 80 | wc -l)
IO_WAIT=$(iostat | awk 'NR==4 {print $5}')

aws cloudwatch put-metric-data --metric-name memory-usage --dimensions
Instance=$INSTANCE_ID --namespace "Custom" --value $USEDMEMORY
aws cloudwatch put-metric-data --metric-name Tcp_connections --dimensions
Instance=$INSTANCE_ID --namespace "Custom" --value $TCP_CONN
aws cloudwatch put-metric-data --metric-name TCP_connection_on_port_80 --
dimensions Instance=$INSTANCE_ID --namespace "Custom" --value
$TCP_CONN_PORT_80
aws cloudwatch put-metric-data --metric-name IO_WAIT --dimensions
Instance=$INSTANCE_ID --namespace "Custom" --value $IO_WAIT
```