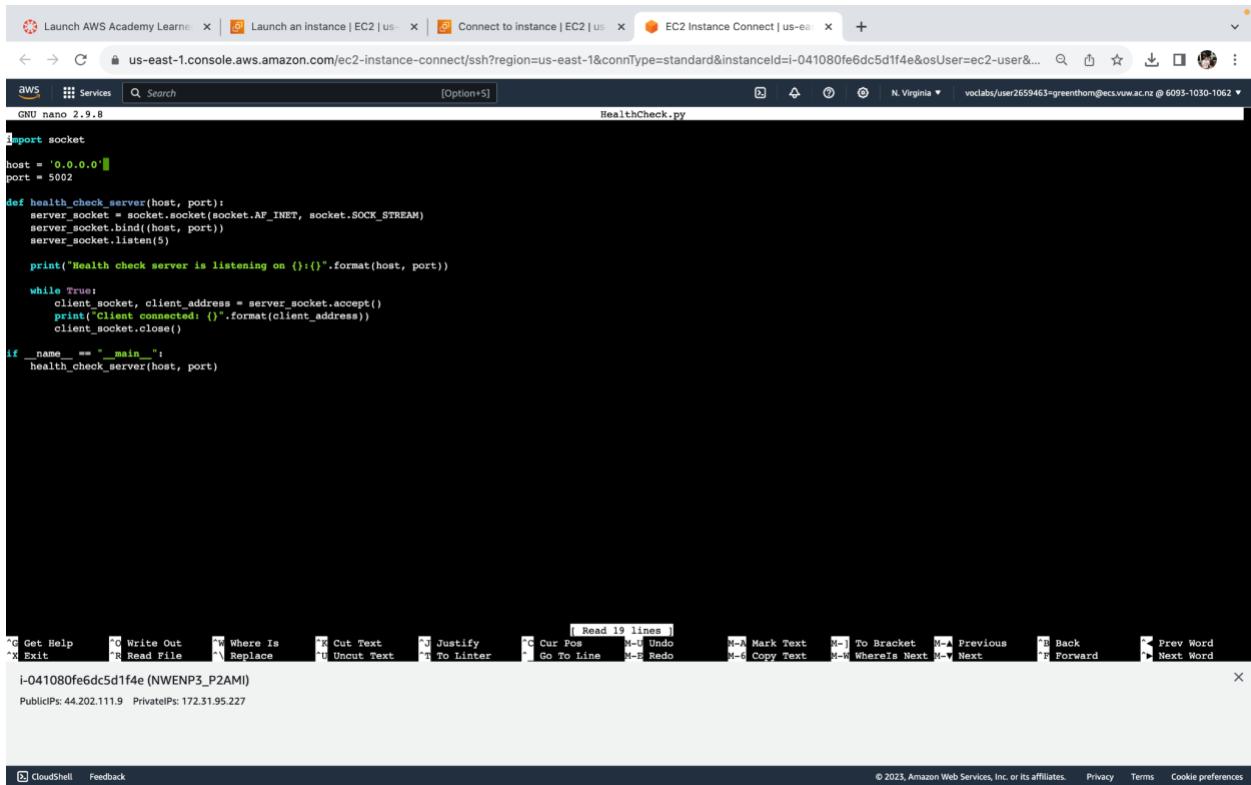


NWEN243 - Project 3
greenthom - 300536064

PART A

Made AMI from project 2 MusicGuruServer. And created HealthCheck.py in that ec2 instance.

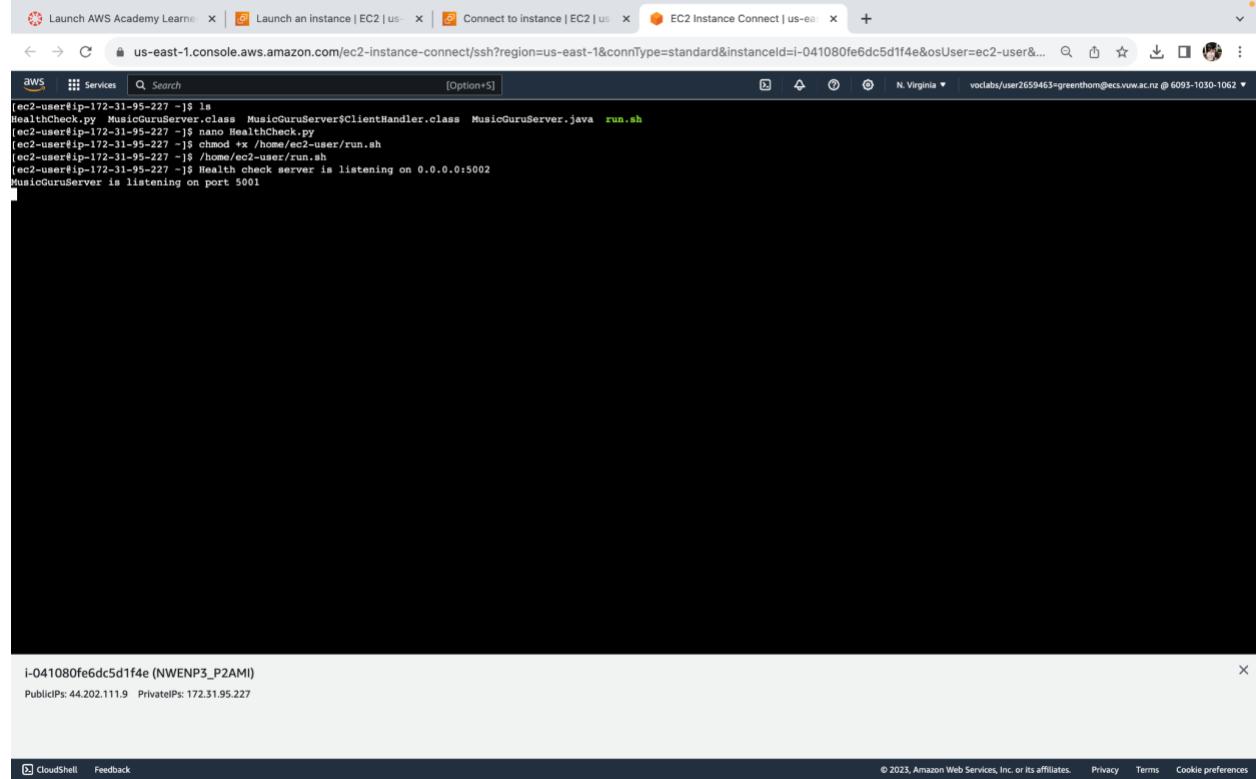


The screenshot shows a terminal window titled "HealthCheck.py" running on an AWS CloudShell. The script code is as follows:

```
GNU nano 2.9.8
import socket
host = '0.0.0.0'
port = 5002
def health_check_server(host, port):
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind((host, port))
    server_socket.listen(5)
    print("Health check server is listening on {}".format(host, port))
    while True:
        client_socket, client_address = server_socket.accept()
        print("Client connected: {}".format(client_address))
        client_socket.close()
if __name__ == "__main__":
    health_check_server(host, port)
```

The terminal shows the script is running and listening on port 5002. The bottom of the terminal displays a menu bar with various text editing commands like Get Help, Write Out, Where Is, Cut Text, Justify, Cur Pos, Undo, Redo, etc., and a status bar indicating the instance ID (i-041080fe6dc5d1f4e), Public IP (44.202.111.9), Private IP (172.31.95.227), and the AWS region (N. Virginia).

I added HealthCheck.py to my run.sh. MusicGuruServer is running at port 5001 (same as from project 2,)and HealthChecks is at port 5002. Rebooted the instance and connected back to it. File permissions are set to ensure that the script is executable using chmod +x /home/ec2-user/run.sh. Here, you can see in the instance that HealthCheck and MusicGuruServer are running properly from run.sh, using the command /home/ec2-user/run.sh



The screenshot shows a CloudShell terminal window with the following session details:

- Region: N. Virginia
- User: vocabs/user2659463=greenthom@ecs.vuw.ac.nz @ 6093-1030-1062

The terminal output shows the execution of the run.sh script:

```
[ec2-user@ip-172-31-95-227 ~]$ ls
healthCheck.py  MusicGuruServer.class  MusicGuruServer$ClientHandler.class  MusicGuruServer.java  run.sh
[ec2-user@ip-172-31-95-227 ~]$ nano HealthCheck.py
[ec2-user@ip-172-31-95-227 ~]$ chmod +x /home/ec2-user/run.sh
[ec2-user@ip-172-31-95-227 ~]$ ./home/ec2-user/run.sh
[ec2-user@ip-172-31-95-227 ~]$ Health check server is listening on 0.0.0.0:5002
MusicGuruServer is listening on port 5001
```

Below the terminal, the instance metadata is displayed:

i-041080fe6dc5d1f4e (NWENP3_P2AMI)
PublicIPs: 44.202.111.9 PrivateIPs: 172.31.95.227

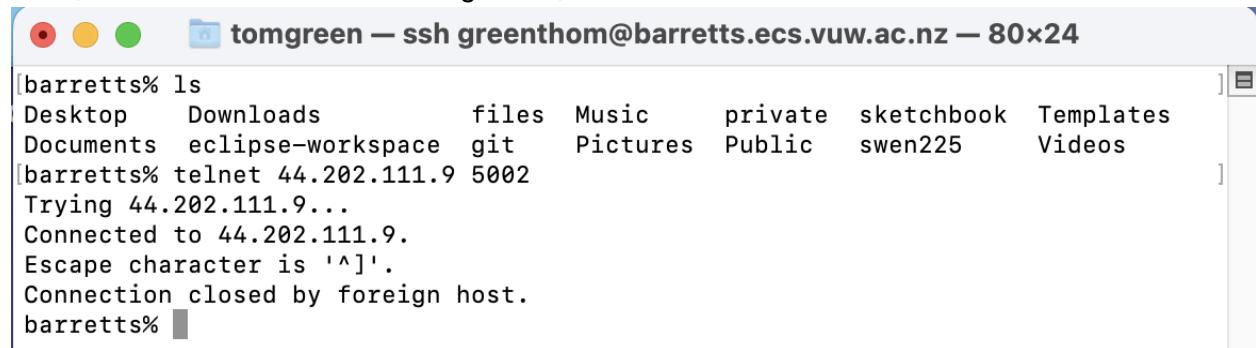
CloudShell Feedback © 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

This is the public IP address of my instance running my HealthCheck and MusicGuruServer code

Public IP address

 44.202.111.9

Here, we contact the instance using telnet, which contacts the HealthCheck server code.



The terminal window title is "tomgreen — ssh greenthom@barretts.ecs.vuw.ac.nz — 80x24". The session output shows:

```
[barretts% ls
Desktop      Downloads      files   Music      private  sketchbook  Templates
Documents    eclipse-workspace git    Pictures  Public    swen225     Videos
[barretts% telnet 44.202.111.9 5002
Trying 44.202.111.9...
Connected to 44.202.111.9.
Escape character is '^]'.
Connection closed by foreign host.
barretts%
```

Now that we have checked that it works, we then create an AMI from the instance with the HealthCheck server code

The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with various navigation options: New EC2 Experience, EC2 Dashboard, EC2 Global View, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images (AMIs, AMI Catalog), Elastic Block Store (Volumes, Snapshots, Lifecycle Manager), Network & Security (Security Groups, Elastic IPs, Placement Groups, Key Pairs, Network Interfaces), Load Balancing (Load Balancers), CloudShell, and Feedback.

The main content area displays the 'Instances (1/2) Info' table. It lists two instances:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic...
NWENP3_P2AMI	i-041080fe6dc5d1f4e	Running	t2.micro	2/2 checks passed	No alarms	us-east-1c	ec2-44-202-111-9.com...	44.202.111.9	-
NWEN_AMIP2	i-0c7c38c487dac9d9f	Terminated	t2.micro	-	No alarms	us-east-1c	-	-	-

Below the table, the details for the selected instance (i-041080fe6dc5d1f4e) are shown. The 'Details' tab is active, displaying the following information:

Details	Security	Networking	Storage	Status checks	Monitoring	Tags
Instance: i-041080fe6dc5d1f4e (NWENP3_P2AMI)						
Instance summary						
Instance ID	i-041080fe6dc5d1f4e (NWENP3_P2AMI)					
IPv6 address	-					
Hostname type	Private IP address (IPv4 only)					
IP name: ip-172-31-95-227.ec2.internal	ip-172-31-95-227.ec2.internal					
Answer private resource DNS name	Instance type					
IPv4 (A)	t2.micro					
Private IP4 address	Private IP4 addresses					
44.202.111.9 [open address]	172.31.95.227					
Instance state	Public IP4 DNS					
Running	ec2-44-202-111-9.compute-1.amazonaws.com [open address]					
Instance type	Elastic IP addresses					
t2.micro	-					

At the bottom right of the details page, there are links for 2023, Amazon Web Services, Inc. or its affiliates., Privacy, Terms, and Cookie preferences.

Here, we can see the AMI that we created for project three, which we will use throughout parts A and part b

The screenshot shows the AWS EC2 service interface. On the left, there's a navigation sidebar with various EC2-related options like Dashboard, Instances, and Network & Security. The main area displays a table titled "Amazon Machine Images (AMIs) (1/6)".

Name	AMI ID	AMI name	Source	Owner	Visibility	Status	Created
ami-0ccc100113f65de2c		MusicGuruAMI	609310301062/MusicGuruAMI	609310301062	Private	Available	2023-09-29T17:58:15Z
ami-0b6cc2857dcbf0504		NWEN243_PROJECT3	609310301062/NWEN243_PROJECT3	609310301062	Private	Available	2023-09-29T17:58:15Z
ami-0253e20e82d954abb		nwemp2	609310301062/nwemp2	609310301062	Private	Available	2023-09-29T17:58:15Z
ami-0606451954c60e81c		NWEN_FINAL_IMAGE	609310301062/NWEN_FINAL_IMAGE	609310301062	Private	Available	2023-09-29T17:58:15Z
ami-035a4febffa63d847		NWEN_TAKE3	609310301062/NWEN_TAKE3	609310301062	Private	Available	2023-09-29T17:58:15Z
ami-0be5978f535831d90		NWEN_IMAGE	609310301062/NWEN_IMAGE	609310301062	Private	Available	2023-09-29T17:58:15Z

Below the table, a modal window is open for the selected AMI (AMI ID: ami-0b6cc2857dcbf0504). The modal has tabs for Details, Permissions, Storage, and Tags. The Details tab is active, showing the following information:

AMI ID	ami-0b6cc2857dcbf0504	Image type	machine	Platform details	Linux/UNIX	Root device type	EBS
AMI name	NWEN243_PROJECT3	Owner account ID	609310301062	Architecture	x86_64	Usage operation	RunInstances
Root device name	/dev/xvda	Status	Available	Source	609310301062/NWEN243_PROJECT3	Virtualization type	hvm
Boot mode	-	State reason	-	Creation date	Fri Sep 29 2023 17:58:15 GMT+1300 (New Zealand Daylight Time)	Kernel ID	-
Description	-	Product codes	-	RAM disk ID	-	Deprecation time	-
Last launched time	-	Block devices	/dev/xvda=snap-007928fc7f525529e:8:true:gp2				

We created a new VPC from here, which I named 'nwen243_p3-vpc' at IPv4 CIDR '192.168.0.0/23.'

The screenshot shows the AWS VPC Details page for the VPC 'nwen243_p3-vpc'. The VPC ID is vpc-0ec3f87ac6fabb6fb, the State is Available, and the IPv4 CIDR is 192.168.0.0/23. The VPC has four subnets: nwen243_p3-subnet-public1-us-east-1a, nwen243_p3-subnet-private1-us-east-1a, nwen243_p3-subnet-public2-us-east-1b, and nwen243_p3-subnet-private2-us-east-1b. There are four route tables: nwen243_p3-rtb-private1-us-east-1a, nwen243_p3-rtb-private2-us-east-1b, nwen243_p3-rtb-public, and rtb-043a43ae4bd423b3c. Two network connections are shown: nwen243_p3-igw and nwen243_p3-vpc-e-s3. A feedback pop-up asks if the resource map was helpful today.

From here, we create two public subsets (public one and public 2) with a divided CIDR - Public One: 192.168.0.0/24 and Public Two: 192.168.1.0/24. Also, both have different zones at us-east-1a and us-east-1b, respectively.

The screenshot shows the AWS VPC Management Console interface. On the left, there's a navigation sidebar with various services like VPC dashboard, EC2 Global View, and several under Virtual private cloud. The main content area is titled 'Subnets' and shows a table of three subnets:

Name	Subnet ID	State	VPC	IPv4 CIDR	IPv6 CIDR	Ava
Public 2	subnet-060a0d3e65da8cbeb	Available	vpc-0ec3fb7ac6fabb6fb nwen...	192.168.1.0/24	-	251
-	subnet-0d2f319d9ab1dfe94	Available	vpc-05900b3f0d4317157	172.31.80.0/20	-	405
Public 1	subnet-0503d048ed2f18008	Available	vpc-0ec3fb7ac6fabb6fb nwen...	192.168.0.0/24	-	251

At the top of the main content area, there are two success messages: "You have successfully deleted subnet-0d129e661529d764b" and "You have successfully created 1 subnet: subnet-060a0d3e65da8cbeb". The bottom of the page has a section titled "Select a subnet" and some footer links.

I made an internet gateway and have named it 'nwen_p3_gateway.'

The screenshot shows the AWS VPC Internet Gateways page. The URL is <https://us-east-1.console.aws.amazon.com/vpc/home?region=us-east-1#InternetGateway:internetGatewayId=igw-04183d54ca1ce6a09>. The page title is "igw-04183d54ca1ce6a09 / nwen_p3_gateway".

Details (Info)

Internet gateway ID	igw-04183d54ca1ce6a09	State	Detached	VPC ID	-	Owner	609310301062
---------------------	-----------------------	-------	----------	--------	---	-------	--------------

Tags

Key	Value
Name	nwen_p3_gateway

Actions: Attach to a VPC

Navigation: VPC > Internet_gateways > igw-04183d54ca1ce6a09

Left sidebar:

- Virtual private cloud
 - Your VPCs [New](#)
 - Subnets
 - Route tables
 - Internet gateways**
 - Egress-only internet gateways
 - Carrier gateways
 - DHCP option sets
 - Elastic IPs
 - Managed prefix lists
 - Endpoints
 - Endpoint services
 - NAT gateways
 - Peering connections
- Security
 - Network ACLs
 - Security groups
- DNS firewall
 - Rule groups
 - Domain lists
- Network Firewall
 - Firewalls
 - Firewall policies
 - Network Firewall rule groups

Bottom right:

© 2023, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

Here, we have attached our gateway to the VPC we just created before. Here, we can see the state of the gateway has gone from detached to attached, showing it is successful

The screenshot shows the AWS VPC Internet Gateways page. At the top, there is a success message: "Internet gateway igw-04183d54ca1ce6a09 successfully attached to vpc-0ec3f87ac6fabb6fb". The main content area displays the details for the internet gateway "igw-04183d54ca1ce6a09 / nwen_p3_gateway". The "Details" tab is selected, showing the following information:

Internet gateway ID	State	VPC ID	Owner
igw-04183d54ca1ce6a09	Attached	vpc-0ec3f87ac6fabb6fb nwen243_p3-vpc	609310301062

Below the details, there is a "Tags" section with one tag: "Name: nwen_p3_gateway". On the right side of the page, there is a "Actions" dropdown menu.

The left sidebar contains a navigation menu for VPC services, including:

- VPC dashboard
- EC2 Global View
- Filter by VPC: Select a VPC
- Virtual private cloud
 - Your VPCs (New)
 - Subnets
 - Route tables
 - Internet gateways** (selected)
 - Egress-only internet gateways
 - Carrier gateways
 - DHCP option sets
 - Elastic IPs
 - Managed prefix lists
 - Endpoints
 - Endpoint services
 - NAT gateways
 - Peering connections
- Security
 - Network ACLs
 - Security groups
- DNS firewall
 - Rule groups
 - Domain lists
- Network Firewall
 - Firewalls
 - Firewall policies
 - Network Firewall rule groups

At the bottom of the page, there are links for CloudShell, Feedback, and copyright information: © 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences.

Here, we can see the default route

The screenshot shows the AWS VPC Route Tables page. On the left, there's a navigation sidebar with sections like EC2 Global View, Filter by VPC, Virtual private cloud, Route tables, Security, DNS firewall, and Network Firewall. The main content area has tabs for 'Route tables (1/1)' and 'Info'. Under 'Route tables', it shows a table with one row:

Name	Route table ID	Explicit subnet associations	Edge associations	Main	VPC	Owner ID
rtb-043a43ae4bd423b3c	rtb-043a43ae4bd423b3c	-	-	Yes	vpc-0ec3f87ac6fabb6fb nwren...	609310301062

Below this, under 'rtb-043a43ae4bd423b3c', there's a 'Routes' tab with a table showing one route:

Destination	Target	Status	Propagated
192.168.0.0/23	local	Active	No

From here, we edit this to include the internet gateway we added before and add the destination to 0.0.0.0/0

The screenshot shows the AWS VPC Route Table Editor. At the top, there are tabs for Launch AWS Academy Learner, Launch an instance | EC2 | us-east-1, VPC | us-east-1, and EC2 Instance Connect | us-east-1. Below the tabs, the URL is us-east-1.console.aws.amazon.com/vpc/home?region=us-east-1#EditRoutes:RouteTableId=rtb-043a43ae4bd423b3c. The main area is titled "Edit routes". It displays two routes in a table:

Destination	Target	Status	Propagated
192.168.0.0/23	local	Active	No
0.0.0.0/0	igw-04183d54ca1ce6a09	-	No

At the bottom right of the table are "Cancel", "Preview", and "Save changes" buttons. The "Save changes" button is highlighted in orange. The bottom of the screen includes standard AWS navigation links like CloudShell, Feedback, and copyright information.

The next thing we have to do is associate the new routing table with the public subnets. Here, we can see that we have added the Public 1 and 2 subsets we created before.

The screenshot shows the AWS VPC Route Tables page. On the left, there's a navigation sidebar with sections like VPC dashboard, EC2 Global View, Filter by VPC (with a dropdown for 'Select a VPC' showing 'vpc-0ec3f87ac6fabb6fb'), Virtual private cloud (with 'Your VPCs' and 'New' buttons), Route tables (selected), Internet gateways, Egress-only Internet gateways, Carrier gateways, DHCP option sets, Elastic IPs, Managed prefix lists, Endpoints, Endpoint services, NAT gateways, Peering connections, Security (Network ACLs, Security groups), DNS firewall (Rule groups, Domain lists), and Network Firewall. The main content area is titled 'Route tables (1/1)'. It shows a table with one row for 'rtb-043a43ae4bd423b3c'. The table columns include Name (rtb-043a43ae4bd423b3c), Route table ID (rtb-043a43ae4bd423b3c), Explicit subnet associations (2 subnets), Edge associations (-), Main (Yes), VPC (vpc-0ec3f87ac6fabb6fb), and Owner ID (609310301062). Below this, under 'rtb-043a43ae4bd423b3c', there are tabs for Details, Routes, Subnet associations (selected), Edge associations, Route propagation, and Tags. The 'Subnet associations' tab shows two entries: 'Public 2' (subnet-060a0d3e65da8cceb, 192.168.1.0/24) and 'Public 1' (subnet-0305d048ed2f18008, 192.168.0.0/24). There are 'Edit subnet associations' buttons for both. Below this, there's a section for 'Subnets without explicit associations' which states 'No subnets without explicit associations' and 'All your subnets are associated with a route table.'

From here, we then go and make a security group and have named it 'NWEN_P3SG'. We replaced the default VPC with the one we created before. We then add inbound rules: one for SSH and two for our Server and HealthCheck code at ports 5001 and 5002, respectively.

The screenshot shows the AWS Cloud Console interface for managing security groups. The main title bar indicates the user is on the 'us-east-1.console.aws.amazon.com/vpc/home?region=us-east-1#SecurityGroup:groupId=sg-00c2d5e6ca6bc506e' page. The left sidebar navigation includes 'Virtual private cloud' (selected), 'Your VPCs', 'Subnets', 'Route tables', 'Internet gateways', 'Egress-only internet gateways', 'Carrier gateways', 'DHCP option sets', 'Elastic IPs', 'Managed prefix lists', 'Endpoints', 'Endpoint services', 'NAT gateways', and 'Peering connections'. Under 'Security', there are sections for 'Network ACLs', 'Security groups' (selected), 'DNS firewall', 'Rule groups', and 'Domain lists'. At the bottom of the sidebar are links for 'CloudShell' and 'Feedback'.

The main content area displays a success message: 'Security group (sg-00c2d5e6ca6bc506e | NWEN_P3SG) was created successfully'. Below this, the security group details are shown:

- Security group name:** NWEN_P3SG
- Security group ID:** sg-00c2d5e6ca6bc506e
- Description:** created sg nwemp3 number1
- VPC ID:** vpc-05900b3f0d4317157
- Owner:** 609310301062
- Inbound rules count:** 3 Permission entries
- Outbound rules count:** 1 Permission entry

The 'Inbound rules' tab is selected, showing three rules:

Name	Security group rule...	IP version	Type	Protocol	Port range	Source
-	sgr-00c5e6d384868c2...	IPv4	SSH	TCP	22	0.0.0.0/0
-	sgr-07d4759b1a2615...	IPv4	Custom TCP	TCP	5002	0.0.0.0/0
-	sgr-048ff1c205ef72a2b	IPv4	Custom TCP	TCP	5001	0.0.0.0/0

At the bottom of the page, there are links for '© 2023, Amazon Web Services, Inc. or its affiliates.', 'Privacy', 'Terms', and 'Cookie preferences'.

Now, we go and create a launch template for the P3 AMI. We named it 'NWEN_P3_LT' and checked the box to provide EC2 autoscaling.

The screenshot shows the 'Create launch template' wizard in the AWS Management Console. The top navigation bar includes tabs for 'Launch AWS Academy Learner' and 'Create launch template | EC2'. The main page title is 'Create launch template' with the URL 'us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#CreateTemplate'. The left sidebar shows 'EC2 > Launch templates > Create launch template'. The main content area is titled 'Create launch template' and contains several configuration sections:

- Launch template name and description**:
 - 'Launch template name - required': NWEN_P3_LT
 - 'Template version description': A prod webserver for MyApp
- Auto Scaling guidance**:
 - Info: Select this if you intend to use this template with EC2 Auto Scaling
 - Provide guidance to help me set up a template that I can use with EC2 Auto Scaling
- Template tags**:
 - ▶
 - Source template
- Launch template contents**:
 - Specify the details of your launch template below. Leaving a field blank will result in the field not being included in the launch template.
 - Application and OS Images (Amazon Machine Image) - required**:
 - 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 bar: Q Search our full catalog including 1000s of application and OS images

A tooltip is displayed over the 'Provide guidance...' checkbox, detailing the Free tier benefits for the first year:

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, 30 GiB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GB of bandwidth to the internet.

At the bottom right are 'Cancel' and 'Create launch template' buttons.

From here, we go down to Application and OS Images, where we select the AMI, we created at the start called 'NWEN243_PROJECT3.'

The screenshot shows the AWS EC2 'Create launch template' wizard. In the top navigation bar, there are tabs for 'Launch AWS Academy Learner', 'Create launch template | EC2', and a '+' button. The main search bar contains 'NWEN243_PROJECT3'. The top right corner shows the region 'N. Virginia' and a user email 'vocabls/user2659463=greenthom@ecs.vuw.ac.nz @ 6093-1030-1062'.

Launch template contents

Specify the details of your launch template below. Leaving a field blank will result in the field not being included in the launch template.

Application and OS Images (Amazon Machine Image) - required [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

Recent AMIs 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)

NWEN243_PROJECT3
ami-0b6cc2857dcfb0f0504
2023-09-29T04:58:15.000Z Virtualization: hvm ENA enabled: true Root device type: ebs

Description

Architecture AMI ID
x86_64 ami-0b6cc2857dcfb0f0504

Instance type [Info](#) Advanced

Instance type t2.micro

Cancel **Create launch template**

Summary

Software Image (AMI)
NWEN243_PROJECT3
ami-0b6cc2857dcfb0f0504

Virtual server type (instance type)
t2.micro

Firewall (security group)
nwenv_inst_sg

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, 30 GiB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GiB of bandwidth to the internet.

© 2023, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

From here, we make sure our instance type is using t2.micro. We then create a new key pair for ssh login called 'nwenkeyp3'. As well as selecting our VPC, we made and added an SSH and two TCP rules, one for MusicGuruServer and one for HealthCheck. Since this was my second time doing all of this, I already had a security group called 'nwen_inst_sg'.

The screenshot shows the 'Create launch template | EC2' page in the AWS Management Console. The 'Instance type' section is set to 't2.micro' (Free tier eligible). The 'Key pair (login)' section contains 'nwenkeyp3'. In the 'Network settings' section, the 'Subnet info' dropdown is set to 'Don't include in launch template'. The 'Firewall (security groups)' section includes 'nwen_inst_sg'. A tooltip for the 'Free tier' button provides information about usage limits. The 'Summary' section lists the AMI, instance type, security group, and storage. At the bottom right is a large orange 'Create launch template' button.

Here, we can see the security group rules showing SSH and TCP.

The screenshot shows the AWS EC2 'Create launch template' interface. The top navigation bar includes tabs for 'Launch AWS Academy Learner' and 'Create launch template | EC2'. The main content area is titled 'Compare security group rules' with a sub-instruction: 'Amazon EC2 evaluates all the rules of the selected security groups to control inbound and outbound traffic. You can select more security groups to view their inbound rules to help you to decide how to secure your instance from incoming traffic.' Below this, a section titled 'Common security groups' shows a dropdown menu with the selected item 'nwen_inst_sg sg-0c784d03236c5c2a9 VPC vpc-0ec3f87ac6fbabb6f'. A note states: 'Security groups that you add or remove here will be added to or removed from all your network interfaces.' The 'Inbound rules (3)' table lists three rules:

Security group name	Security group ID	Type	Protocol	Port range	Source	Description
nwen_inst_sg	sg-0c784d03236c5c2a9	ssh	tcp	22	0.0.0.0/0	-
nwen_inst_sg	sg-0c784d03236c5c2a9	Custom TCP	tcp	5002	0.0.0.0/0	-
nwen_inst_sg	sg-0c784d03236c5c2a9	Custom TCP	tcp	5001	0.0.0.0/0	-

At the bottom right of the table, there are 'Cancel' and 'Select security groups' buttons. The footer includes links for 'cloudShell', 'Feedback', and copyright information: '© 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences'.

We then go to advanced network configuration and get the drop-down. We then add a network interface and enable auto-assign public IP and delete on termination.

The screenshot shows the AWS CloudFormation 'Create launch template' interface. In the top navigation bar, there are tabs for 'Launch AWS Academy Learner' and 'Create launch template | EC2'. The main content area is titled 'us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#CreateTemplate:'.

The interface is divided into several sections:

- Network interface 1**:
 - Device index**: Info (0) [New interface] [Remove]
 - Subnet**: Info (Don't include in launch template)
 - Primary IP**: Info (Not applicable for EC2 Auto Scaling)
 - IPv4 Prefixes**: Info (Don't include in launch template)
 - Delete on termination**: Info (Yes)
- Security groups**: Info (Select security groups) [Show all selected (1)]
- Auto-assign public IP**: Info (Enable)
- IPv6 IP**: Info (Don't include in launch template)
- IPv6 Prefixes**: Info (Don't include in launch template)
- Elastic Fabric Adapter**: Info (Enable)
- Network card index**: Info (Don't include in launch template)

Summary section:

- Software Image (AMI)**: NWEN243_PROJECT3 ami-0b6cc22857cd0f0504
- Virtual server type (instance type)**: t2.micro
- Firewall (security group)**: nwen_inst_sg
- Storage (volumes)**: 1 volume(s) - 8 GiB

A callout box highlights the **Free tier** information:

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, 30 GiB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GiB of bandwidth to the internet.

At the bottom right, there are 'Cancel' and 'Create launch template' buttons.

Here, we can see our Launch Template created successfully

The screenshot shows the AWS EC2 'Create launch template' page. At the top, there are two tabs: 'Launch AWS Academy Learner' and 'Create launch template | EC2'. The URL in the address bar is 'us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#CreateTemplate'. The main content area has a green success message box that says 'Successfully created NWFN_P3_LT(lt-0db06440e87049cea)'. Below this, there's a 'Actions log' section and a 'Next steps' section. The 'Next steps' section contains three items: 'Launch an instance', 'Create an Auto Scaling group from your template', and 'Create Spot Fleet'. Each item has a brief description and a link to further details. At the bottom right of the main content area is an orange button labeled 'View launch templates'.

We can now test if the launch template is working. We use the public one subnet.

The screenshot shows the 'Network settings' step of the 'Launch instance from template' wizard. In the 'Subnet info' section, it lists 'subnet-0303d048e01f118008' as 'Public 1'. Under 'Firewall (security groups)', the 'Select existing security group' radio button is selected, and 'nwen_inst_sg' is chosen. A tooltip for the free tier is visible. In the 'Storage (volumes)' section, 'EBS Volumes' are listed. At the bottom right, there is a 'Launch instance' button.

The public two subnet.

The screenshot shows the 'Network settings' step of the 'Launch instance from template' wizard for a two-subnet configuration. It lists two subnets: 'subnet-060a0d1e65da8cbeb' (Public 2) and 'subnet-060a0d1e65da8cbeb' (Public 1). Under 'Firewall (security groups)', the 'Select existing security group' radio button is selected, and 'nwen_inst_sg' is chosen. A tooltip for the free tier is visible. In the 'Storage (volumes)' section, 'EBS Volumes' are listed. At the bottom right, there is a 'Launch instance' button.

From here, we can see that Public 1 and Public 2 were created successfully at the same availability zone that our subsets are working at.

The screenshot shows the AWS EC2 Instances page with the following details:

Instances (8) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic
NWENP3_P2AMI	i-041080fe6dc5d1f4e	Terminated	t2.micro	-	No alarms	us-east-1c	-	-	-
Public2_Instance	i-078293fba3b8a8d57	Terminated	t2.micro	-	No alarms	us-east-1b	-	-	-
-	i-06783022df64bbb18	Terminated	t2.micro	-	No alarms	us-east-1b	-	-	-
Public1_Instance	i-00e51509a62c98dec	Terminated	t2.micro	-	No alarms	us-east-1a	-	-	-
-	i-0ef0abfb0f5b694c	Terminated	t2.micro	-	No alarms	us-east-1a	-	-	-
-	i-0436f5990d3403305	Terminated	t2.micro	-	No alarms	us-east-1a	-	-	-
Public1	i-0c6133ec3d5de64d	Running	t2.micro	Initializing	No alarms	us-east-1a	ec2-54-243-2-31.comp...	54.243.2.31	-
Public2	i-0bb9a72fa6b323ddd	Running	t2.micro	-	No alarms	us-east-1b	ec2-44-204-211-29.co...	44.204.211.29	-

Select an instance

CloudShell Feedback © 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Next, we make a load balancer and a target group. Here, we are making a target group. We set the target group name to 'NWENP3_TG' and selected the protocol as TCP with the respective MusicGuruServer port number. We leave the IP address type as IPv4 and select the VPC that we made.

Target group name

NWENP3_TG

A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

Protocol

Port

TCP



: 5001

1-65535

IP address type

Only targets with the indicated IP address type can be included in this target group.

IPv4

IPv6

Each target you register must have an assigned primary IPv6 address. This is configured on the instances default network interface (eth0). [Learn more](#)

VPC

Select the VPC with the instances that you want to include in the target group. Only VPCs that support the IP address type selected above are available in this list.

nwen243_p3-vpc

vpc-0ec3f87ac6fabb6fb

IPv4: 192.168.0.0/23

We next go and edit the health check settings. We make the HealthCheck protocol TCP. We also changed the advanced health check settings. We leave all of them to default, apart from the health check port being set to 5002; that is what we are running the TCP on.

The screenshot shows the AWS CloudFormation 'Create target group' configuration for a Load Balancer. The 'Health checks' section is open, showing the following settings:

- Health check protocol:** TCP
- Health check port:** Override (selected), port 5002
- Healthy threshold:** 5
- Unhealthy threshold:** 2
- Timeout:** 10 seconds
- Interval:** 30 seconds

At the bottom right of the page, there are links for CloudShell, Feedback, Privacy, Terms, and Cookie preferences.

Here is the target group we created, with the corresponding port, protocol, and VPC ID selected

The screenshot shows the AWS EC2 Target Groups page. The left sidebar includes sections for Instances, Images, Elastic Block Store, Network & Security, and Load Balancing. The main content area displays a table titled "Target groups (1) [Info]" with one item: "NWENP3-TG". The table columns are Name, ARN, Port, Protocol, Target type, Load balancer, and VPC ID. The "NWENP3-TG" row shows "arn:aws:elasticloadbalancing:us-east-1:6093-1062" in the ARN column, "5001" in the Port column, "TCP" in the Protocol column, "Instance" in the Target type column, "None associated" in the Load balancer column, and "vpc-0ec3f87ac6fabb6fb" in the VPC ID column. A message at the top states "Successfully created target group: NWENP3-TG". Below the table, a message says "0 target groups selected" and "Select a target group above."

Name	ARN	Port	Protocol	Target type	Load balancer	VPC ID
NWENP3-TG	arn:aws:elasticloadbalancing:us-east-1:6093-1062	5001	TCP	Instance	None associated	vpc-0ec3f87ac6fabb6fb

Next, we create a load balancer. We set the load balancer name as 'NWENP3_LB' and left the defaults as internet facing and IPv4.

The screenshot shows the 'Create network load balancer' wizard in the AWS Management Console. The current step is 'Basic configuration'. The 'Load balancer name' field is filled with 'NWENP3_LB'. The 'Scheme' section has 'Internet-facing' selected. Under 'IP address type', 'IPv4' is chosen. In the 'Network mapping' section, a VPC is selected with subnet 'vpc-0590003f0d4317157' and IP range 'IPv4: 172.31.0.0/16'. A mapping for 'us-east-1c (use1-az2)' is shown. The bottom of the screen includes standard AWS navigation links like CloudShell, Feedback, and copyright information.

Next, we sort the network mapping. We select the VPC we created. We sort the mappings by selecting the Public 1 or Public 2 that we also created earlier. We also leave the IPv4 addresses assigned by AWS.

The screenshot shows the 'Network mapping' step of the AWS Create Network Load Balancer Wizard. It displays two sections for mapping subnets to availability zones:

- us-east-1a (use1-az6):** Subnet `subnet-0303d048ed2f18008` is mapped to `Public 1`. The IPv4 address is listed as `Assigned by AWS`.
- us-east-1b (use1-az1):** Subnet `subnet-060a0d3e65da8cbeb` is mapped to `Public 2`. The IPv4 address is listed as `Assigned by AWS`.

Security groups: A security group is selected to control traffic to the load balancer.

Next, we add the security group we made (We did this before updates were made to the script, so did not add the default one as well). Next, we sort listeners and routing; for the protocol, leave it at TCP and port at 5001 (MusicGuruServer) and choose the target group we made (NWENP3-TG)

The screenshot shows the AWS CloudFormation Create Network Load Balancer wizard. The top navigation bar includes tabs for Launch AWS Academy Learner, Instances | EC2 | us-east-1, and Create network load balancer. The URL is us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#CreateNLBWizard.

Security groups (Info): A security group is a set of firewall rules that control the traffic to your load balancer. Select an existing security group, or you can create a new security group. A recommended security group, 'nwen_inst_sg', is selected. It is associated with VPC vpc-0ec3fb7ac6fabbd6fb.

Listeners and routing (Info): A listener is a process that checks for connection requests using the port and protocol you configure. The rules that you define for a listener determine how the load balancer routes requests to its registered targets. A listener named 'Listener TCP:5001' is configured with the following settings:

- Protocol: TCP
- Port: 5001
- Default action: Forward to target group NWENP3-TG (Target type: Instance, IPv4)

Below the listener configuration, there are sections for Listener tags - optional and Add listener.

At the bottom of the page, there are links for CloudShell, Feedback, and a footer with copyright information: © 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences.

Here, our created load balancer (NWENP3-LB) shows a provisioning state.

The screenshot shows the AWS Cloud Console interface. The top navigation bar includes tabs for Launch AWS Academy Learner, Instances | EC2 | us-east-1, and Load balancers | Load Balancer. The main search bar contains the query "us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LoadBalancers:search=NWENP3-LB". The left sidebar is collapsed, showing the New EC2 Experience button and a "Tell us what you think" link. The main content area is titled "EC2 > Load balancers". A sub-header "Load balancers (1/1)" indicates there is one item. Below it, a note states "Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic." A search bar and a "Clear filters" button are present. A table lists the single load balancer entry: Name (NWENP3-LB), DNS name (NWENP3-LB-4f3669b8d7be0c1b.elb.us-east-1.amazonaws.com), State (Provisioning), VPC ID (vpc-0ec3f87ac6fabb6fb), Availability Zones (2 Availability Zones), Type (network), and Date created (September 30, 2023, 00:45 (UTC+13:00)). An "Actions" dropdown menu and a "Create load balancer" button are at the top right of the table. A modal window titled "Load balancer: NWENP3-LB" is open, showing the "Details" tab. It displays the following information:

Load balancer type	Status	VPC	IP address type
Network	Provisioning	vpc-0ec3f87ac6fabb6fb	IPv4
Scheme	Hosted zone	Availability Zones	Date created
Internet-facing	Z26RNLL4JYFTOTI	subnet-060a03e65da8cbeb us-east-1b (use1-az1) subnet-0303d048ed2f18008 us-east-1a (use1-az6)	September 30, 2023, 00:45 (UTC+13:00)
Load balancer ARN	DNS name		
arn:aws:elasticloadbalancing:us-east-1:609310301062:loadbalance/net/NWENP3-LB/4f3669b8d7be0c1b	NWENP3-LB-4f3669b8d7be0c1b.elb.us-east-1.amazonaws.com (A Record)		

At the bottom of the modal, there are tabs for Details, Listeners, Network mapping, Security, Monitoring, Integrations, Attributes, and Tags. The footer of the page includes links for CloudShell, Feedback, and various AWS terms like © 2023, Amazon Web Services, Inc. or its affiliates., Privacy, Terms, and Cookie preferences.

Next, we create an auto-scaling group. We named the auto-scaling group 'NWENP3-AS' and selected the launch template we had just made. We changed the version to Latest(1). We then click next.

The screenshot shows the 'Create Auto Scaling group' wizard on the AWS Management Console. The current step is 'Step 1: Choose launch template'. The left sidebar lists steps from 1 to 7. The main area shows a form for creating an Auto Scaling group. In the 'Name' field, 'NWENP3-AS' is entered. Below it, a note states: 'Auto Scaling group name Enter a name to identify the group. Must be unique to this account in the current Region and no more than 255 characters.' In the 'Launch template' section, 'NWEN_P3_LT' is selected from a dropdown. A note below says: 'For accounts created after May 31, 2023, the EC2 console only supports creating Auto Scaling groups with launch templates. Creating Auto Scaling groups with launch configurations is not recommended but still available via the CLI and API until December 31, 2023.' The 'Version' dropdown shows 'Latest (1)'. The configuration table includes fields for Description (empty), Launch template (NWEN_P3_LT), Instance type (t2.micro), AMI ID (ami-0b6cc2857dcf0504), Security groups (empty), Request Spot Instances (No), Key pair name (empty), and Security group IDs (empty).

We then assign the VPC we created and the Public 1 and 2 subsets.

The screenshot shows the AWS Auto Scaling group creation wizard, Step 2: Choose instance launch options. The 'Network' tab is selected. In the 'VPC' section, the VPC 'vpc-0ec3fb7ac6fab6fb (nwen243_p3-vpc)' is chosen. Under 'Availability Zones and subnets', two subnets are selected: 'us-east-1a | subnet-0303d048ed2f18008 (Public 1)' and 'us-east-1b | subnet-060a0d5e65da8cbeb (Public 2)'. In the 'Instance type requirements' section, the launch template 'NIVEN_P3_LT' and instance type 't2.micro' are specified. Navigation buttons at the bottom include 'Cancel', 'Skip to review', 'Previous', and 'Next'.

Now we select to attach to an existing load balancer, where we then do and attack it to the target group we made earlier (NWENP3-TG)

The screenshot shows the AWS CloudFormation Create Stack Wizard Step 4: Configure advanced options - optional. The 'Attach to an existing load balancer' option is selected. Under 'Existing load balancer target groups', the 'NWENP3-TG | TCP' target group is selected.

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 [X](#)

NWENP3-TG | TCP [X](#)
Network Load Balancer: NWENP3-LB

VPC Lattice integration options [Info](#)
To improve networking capabilities and scalability, integrate your Auto Scaling group with VPC Lattice. VPC Lattice facilitates communications between AWS services and helps you connect and manage your applications across compute services in AWS.

Select VPC Lattice service to attach

CloudShell Feedback © 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

We then scroll down and do not turn on health checks, and we enable group metrics collection within Cloudwatch

The screenshot shows the 'Create Auto Scaling group | EC2' configuration page. In the 'VPC Lattice service' section, the 'No VPC Lattice service' option is selected. In the 'Health checks' section, the 'EC2 health checks' option is selected and labeled as 'Always enabled'. Under 'Additional health check types - optional', there is a note about 'Turn on Elastic Load Balancing health checks' being recommended. In the 'Monitoring' section, the 'Enable group metrics collection within CloudWatch' checkbox is checked. At the bottom, there are 'Cancel', 'Skip to review', 'Previous', and 'Next' buttons, with 'Next' highlighted.

Next, we configure the group size and scaling policies. We set our desired capacity at 2, our minimum capacity at two, and our maximum capacity at 3. This provides the number of instances that will be made. Next, we turn it on for our scaling policies by selecting the target tracking scaling policy. We leave it as default.

The screenshot shows the AWS Auto Scaling group creation wizard, Step 3: Configure group size and scaling policies - optional. The left sidebar lists steps: Step 1 (Choose launch template), Step 2 (Choose instance launch options), Step 3 (Configure group size and scaling policies - optional), Step 4 (Configure advanced options), Step 5 (Add notifications), Step 6 (Add tags), and Step 7 (Review). The main content area is titled "Configure group size and scaling policies - optional". It includes sections for "Group size - optional" and "Scaling policies - optional". In the "Group size - optional" section, "Desired capacity" is set to 2, "Minimum capacity" is set to 2, and "Maximum capacity" is set to 3. In the "Scaling policies - optional" section, "Target tracking scaling policy" is selected, and "Scaling policy name" is set to "Target Tracking Policy". The "Metric type" dropdown is set to "CPU Utilization".

We then skip through notifications and tags and look at the review.

The screenshot shows the AWS CloudFormation console during the 'Create Stack' process. The top navigation bar includes tabs for 'Launch AWS Academy Learner', 'Instances | EC2 | us-east-1', and 'Create Auto Scaling group | EC2'. The main content area is titled 'Review' with a 'Info' link. On the left, a sidebar lists steps from 1 to 7: 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 policies), Step 5 (optional: Add notifications), Step 6 (optional: Add tags), and Step 7 (Review). The 'Review' step is currently active, showing:

- Step 1: Choose launch template**: Shows 'Group details' with an 'Edit' button. The 'Auto Scaling group name' is set to 'NWENP3-AS'. Below it, the 'Launch template' section shows 'Launch template' as 'NWEN_P3_LT' (with a dropdown arrow) and 'Version' as 'Latest'. The description 'lt-0db06440e87049cea' is also visible.

Launch template	Version	Description
NWEN_P3_LT	Latest	lt-0db06440e87049cea
- Step 2: Choose instance launch options**: Shows 'Network' settings. It lists a VPC ('vpc-0ec3f87ac6fabbb6fb') and two subnets: 'us-east-1a' (subnet-0303d048ed2f18008, 192.168.0.0/24) and 'us-east-1b' (subnet-060a0d3e65da8cbeb, 192.168.1.0/24).

Availability Zone	Subnet	
us-east-1a	subnet-0303d048ed2f18008	192.168.0.0/24
us-east-1b	subnet-060a0d3e65da8cbeb	192.168.1.0/24
- Step 3: Choose instance type requirements**: This section is partially visible below the network settings.

At the bottom of the page, there are links for 'CloudShell', 'Feedback', and copyright information: '© 2023, Amazon Web Services, Inc. or its affiliates.' followed by 'Privacy', 'Terms', and 'Cookie preferences'.

Here, we can see the autoscaling group we just created

The screenshot shows the AWS EC2 Auto Scaling groups page. At the top, there are three tabs: "Launch AWS Academy Learner", "Instances | EC2 | us-east-1", and "Auto Scaling groups | EC2 | us-east-1". The "Auto Scaling groups" tab is selected. The main content area has a header "Auto Scaling groups (1) Info". Below it is a search bar "Search your Auto Scaling groups". A table lists one Auto Scaling group:

Name	Launch template/configuration	Instances	Status	Desired capacity	Min	Max	Availability Zones
NWENP3-AS	NWEN_P3_LT Version Latest	0	Updating capacity...	2	2	3	us-east-1a, us-east-1b

At the bottom left, it says "0 Auto Scaling groups selected". The footer includes links for "CloudShell", "Feedback", and copyright information: "© 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences".

Here, we can see the newly created instances that match my policies.

The screenshot shows the AWS EC2 Instances page with the following details:

Instances (5) Info

Find instance by attribute or tag (case-sensitive)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Ela
-	i-06086aa211e13f0d0	Running	t2.micro	Initializing	No alarms	+ us-east-1b	ec2-3-238-35-239.com...	5.258.35.239	-
Public2	i-0bb9a72fa6b323ddd	Running	t2.micro	2/2 checks passed	No alarms	+ us-east-1b	ec2-44-204-211-29.co...	44.204.211.29	-
-	i-0ef0abfb0f5b694c	Terminated	t2.micro	-	No alarms	+ us-east-1a	-	-	-
Public1	i-0c6153e3d5dbe64d	Running	t2.micro	2/2 checks passed	No alarms	+ us-east-1a	ec2-54-243-2-31.comp...	54.243.2.31	-
-	i-0b8c606e21a8a89df	Running	t2.micro	Initializing	No alarms	+ us-east-1a	ec2-23-20-152-79.com...	23.20.152.79	-

Select an instance

© 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Next, we go onto the load balancer we created (NWENP3-LB) and copy the DNS name. We use this later when testing that our code works.

The screenshot shows the AWS CloudWatch Metrics Insights interface. A search bar at the top contains the query: `CloudWatch Metrics Insights Metrics`. Below the search bar, there are two tabs: `Metrics` and `Logs`. The `Metrics` tab is selected. The results table has three columns: `Series`, `Dimensions`, and `Metrics`. One row is visible, showing the series `CloudWatch Metrics Insights Metrics`, dimensions `CloudWatch Metrics Insights Metrics`, and metrics `CloudWatch Metrics Insights Metrics`.

We then go to our target groups to find the health status of the instances created. As we can see, they are healthy.

The screenshot shows the AWS Lambda console with the URL us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#TargetGroups. The left sidebar includes sections for Events, Instances (with sub-options like Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations), Images (AMIs, AMI Catalog), Elastic Block Store (Volumes, Snapshots, Lifecycle Manager), Network & Security (Security Groups, Elastic IPs, Placement Groups, Key Pairs, Network Interfaces), Load Balancing (Load Balancers, Target Groups), and Auto Scaling (Auto Scaling Groups). The main content area displays the 'Target groups' section for the 'NWENP3-TG' target group. The table shows one target: 'arn:aws:elasticloadbalancing:us-east-1:1609310301062:targetgroup/NWENP3-TG/ae8dce56c92c248f'. The 'Details' tab is selected, showing the target type as 'Instance', protocol as 'TCP: 5001', VPC as 'vpc-0ec3f87ac6fabb6fb', and IP address type as 'IPv4'. The 'Targets' tab shows two total targets, both marked as 'Healthy' (green). The 'Monitoring' tab shows no data, and the 'Health checks' tab shows no data. The 'Attributes' and 'Tags' tabs are also present but empty.

Next, we use the DNS name info we got before to check our MusicGuruClient runs with these instances created, as we can see the IP address of the responder changes.

The screenshot shows a terminal window titled 'Desktop — zsh — 138x24'. The command entered is 'javac MusicGuruClient.java' followed by 'java MusicGuruClient NWENP3-LB-4f3669b8d7be0c1b.elb.us-east-1.amazonaws.com 5001 0'. The output of the command is displayed, showing song information for years 1992, 1997, and 1993, including artists like Eric Clapton, Foo Fighters, and Wu Tang Clan. The terminal window has a dark theme with red, yellow, and green circular icons in the top-left corner.

```
[tomgreen@Toms-Air-3 Desktop % javac MusicGuruClient.java
[tomgreen@Toms-Air-3 Desktop % java MusicGuruClient NWENP3-LB-4f3669b8d7be0c1b.elb.us-east-1.amazonaws.com 5001 0
Specified year out of range (1990 - 1999), using a random year instead: 1992
In 1992 number 3 song was 'Tears In Heaven' by Eric Clapton (192.168.1.225)
[tomgreen@Toms-Air-3 Desktop % java MusicGuruClient NWENP3-LB-4f3669b8d7be0c1b.elb.us-east-1.amazonaws.com 5001 0
Specified year out of range (1990 - 1999), using a random year instead: 1997
In 1997 number 4 song was 'Everlong' by Foo Fighters (192.168.1.225)
[tomgreen@Toms-Air-3 Desktop % java MusicGuruClient NWENP3-LB-4f3669b8d7be0c1b.elb.us-east-1.amazonaws.com 5001 0
Specified year out of range (1990 - 1999), using a random year instead: 1993
In 1993 number 5 song was 'C.R.E.A.M.' by Wu Tang Clan (192.168.0.62)
```

QUESTIONS

1. Is there any pattern? Experiment a little to find out.

Based on the output, we can see that the requests made to the MusicGuruService through the load balancer are distributed between two different IP addresses (182.168.1.225 and 192.168.0.62). This is due to two instances serving requests, and as we can see, these instances are randomly providing us an output when connecting to the server. The load balancer seems to evenly distribute requests across instances, as there isn't a consistent pattern of requests going to a specific instance. Instances with IP addresses 192.168.1.225 and 192.168.0.62 are both handling requests, indicating that the load balancer is appropriately directing traffic to these instances.

Here, we can see that we have terminated one instance operating our 'Public 2' subnet indicated as running in the availability zone us-east-1b. This provided me only the ability to run the 'Public 1' subnet, giving me the same IP each time due to only one instance running my MusicGuruServer.

The screenshot shows the AWS CloudShell interface with the following details:

- Top Bar:** Launch AWS Academy Learner, Instances | EC2 | us-east-1, and a search bar for us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#instances:.
- Sidebar (Services):**
 - Instances:** Submenu includes Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations.
 - Images:** Submenu includes AMIs, AMI Catalog.
 - Elastic Block Store:** Submenu includes Volumes, Snapshots, Lifecycle Manager.
 - Network & Security:** Submenu includes Security Groups, Elastic IPs, Placement Groups, Key Pairs, Network Interfaces.
 - Load Balancing:** Submenu includes Load Balancers, Target Groups.
 - Auto Scaling:** Submenu includes Auto Scaling Groups.
- Main Content Area:**
 - Instances (4) Info:** A table showing four EC2 instances. The columns are: Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, Public IPv4 DNS, and Public IPv4

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...
-	i-06086aa211e13f0d0	Terminated	t2.micro	-	No alarms	us-east-1b	-	-
Public2	i-0bb9a72fa6b323ddd	Running	t2.micro	2/2 checks passed	No alarms	us-east-1b	ec2-44-204-211-29.co...	44.204.211.29
Public1	i-0c6133ec3d5dbe64d	Running	t2.micro	2/2 checks passed	No alarms	us-east-1a	ec2-54-243-2-31.comp...	54.243.2.31
-	i-08c606e21a8a89df	Running	t2.micro	2/2 checks passed	No alarms	us-east-1a	ec2-23-20-152-79.com...	23.20.152.79
 - Select an instance:** A dropdown menu for selecting an instance.

However, when I terminate an instance, it replaces it with a new one corresponding to the terminated instance details (Public1 or Public2).

The screenshot shows the AWS EC2 Instances page. On the left, a sidebar lists various services: Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images, AMIs, AMI Catalog, Elastic Block Store, Volumes, Snapshots, Lifecycle Manager, Network & Security, Security Groups, Elastic IPs, Placement Groups, Key Pairs, Network Interfaces, Load Balancing, Load Balancers, Target Groups, Auto Scaling, and Auto Scaling Groups. The main content area has a header "Instances (5) Info" with filters for "Find instance by attribute or tag (case-sensitive)". It displays five instances:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Ela
-	i-06086aa211e13f0d0	Terminated	t2.micro	-	No alarms	+ us-east-1b	-	-	-
-	i-0392875b9fd122850	Running	t2.micro	-	No alarms	+ us-east-1b	ec2-3-231-162-165.co...	3.231.162.165	-
Public2	i-0bb9a72fa6b523ddd	Running	t2.micro	2/2 checks passed	No alarms	+ us-east-1b	ec2-44-204-211-29.co...	44.204.211.29	-
Public1	i-0c6133ec3d5dbe64d	Running	t2.micro	2/2 checks passed	No alarms	+ us-east-1a	ec2-54-243-2-31.comp...	54.243.2.31	-
-	i-0b8c606e21a8a89df	Running	t2.micro	2/2 checks passed	No alarms	+ us-east-1a	ec2-23-20-152-79.com...	23.20.152.79	-

A modal window titled "Select an instance" is open at the bottom of the page.

Here, we terminated all the instances providing me with no output on my MusicGuruServer when attempting to run at this time. We terminated all the instances to see if the load balancer would create the instances again.

The screenshot shows the AWS EC2 Instances page. The left sidebar includes sections for Launch AWS Academy Learner, Services (with AWS selected), Search, Notifications (0), and a list of services like Events, Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images, AMIs, AMI Catalog, Elastic Block Store, Volumes, Snapshots, Lifecycle Manager, Network & Security, Security Groups, Elastic IPs, Placement Groups, Key Pairs, Network Interfaces, Load Balancing, Load Balancers, Target Groups, Auto Scaling, and Auto Scaling Groups. The main content area displays a table titled "Instances (5) Info" with the following data:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Ela
-	i-06086aa211e13f0d0	Terminated	t2.micro	-	No alarms	us-east-1b	-	-	-
-	i-0392875b9fd122850	Terminated	t2.micro	-	No alarms	us-east-1b	-	-	-
Public2	i-0bb9a72fa9b323ddd	Running	t2.micro	2/2 checks passed	No alarms	us-east-1b	ec2-44-204-211-29.co...	44.204.211.29	-
Public1	i-0c6133ec3d5dbe64d	Running	t2.micro	2/2 checks passed	No alarms	us-east-1a	ec2-54-243-2-31.comp...	54.243.2.31	-
-	i-08c606e21a8a89df	Terminated	t2.micro	-	No alarms	us-east-1a	-	-	-

A modal window titled "Select an instance" is open at the bottom of the table. The footer of the page includes links for CloudShell, Feedback, and copyright information: © 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences.

As we can see here, our load balancer is working correctly, as it made two more instances due to the originals being terminated. The auto scaling group recognized that the number of instances had fallen below the desired capacity of 2. Therefore, to maintain the desired capacity it initiated 2 new instances to replace the terminated ones reflecting the Public one and two subsets.

The screenshot shows the AWS EC2 Instances page with the following details:

- Success Message:** Successfully terminated i-0392875b9fd122850...-0b8c606e21a8a89df
- Instances Table:**

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic
-	i-06086aa211e13f0d0	Terminated	t2.micro	-	No alarms	us-east-1b	-	-	-
-	i-0392875b9fd122850	Terminated	t2.micro	-	No alarms	us-east-1b	-	-	-
Public2	i-0tb9a72fa6b323ddd	Running	t2.micro	2/2 checks passed	No alarms	us-east-1b	ec2-44-204-211-29.co...	44.204.211.29	-
Public1	i-0c6133e3d5dbe64d	Running	t2.micro	2/2 checks passed	No alarms	us-east-1a	ec2-54-243-2-31.comp...	54.243.2.31	-
-	i-0b8c606e21a8a89df	Terminated	t2.micro	-	No alarms	us-east-1a	-	-	-
-	i-0b4bec88a1626d8ea	Running	t2.micro	Initializing	No alarms	us-east-1a	ec2-35-173-219-71.co...	35.173.219.71	-
-	i-04e48d8ea43678d29	Running	t2.micro	2/2 checks passed	No alarms	us-east-1b	ec2-3-237-89-168.com...	3.237.89.168	-
- Actions:** Launch instances, Connect, Instance state, Actions, Notifications.
- Filters:** Find instance by attribute or tag (case-sensitive).
- Navigation:** Instances (7) Info, Select an instance.
- Footer:** cloudShell, Feedback, © 2023, Amazon Web Services, Inc. or its affiliates., Privacy, Terms, Cookie preferences.

Question 2: What happens? It can take a long time - so be patient.

Here, we edit our auto-scaling groups to determine the desired capacity at 4. This will, therefore, increase the amount of our instances.

The screenshot shows the AWS Management Console with the URL us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#EditAutoScalingGroup:id=NWENP3-AS. The left sidebar is collapsed, and the main content area is titled "Edit NWENP3-AS".

Group size (Info)

Specify the size of the Auto Scaling group by changing the desired capacity. You can also specify minimum and maximum capacity limits. Your desired capacity must be within the limit range.

Desired capacity: 4

Minimum capacity: 3

Maximum capacity: 5

Launch template (Info)

For accounts created after May 31, 2023, the EC2 console only supports creating Auto Scaling groups with launch templates. Creating Auto Scaling groups with launch configurations is not recommended but still available via the CLI and API until December 31, 2023.

Launch template: NWEN_P3_LT

Create a launch template

Version: Latest (1)

Create a launch template version

Description: Launch template Instance type:

CloudShell Feedback © 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

As you can see, our desired capacity has increased to the amount we set. Now, we can observe our instances to see if these updates.

The screenshot shows the AWS Management Console with the URL us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#AutoScalingGroups:id=NWENP3-AS;view=details. The left sidebar is collapsed, and the main content area displays the Auto Scaling groups page. A green banner at the top says "Auto Scaling group updated successfully". The table lists one Auto Scaling group:

Name	Launch template/configuration	Instances	Status	Desired capacity	Min	Max	Availability Zones
NWENP3-AS	NWEN_P3_LT Version Latest	2	Updating capacity...	4	3	5	us-east-1a, us-east-1b

Below the table, a modal window titled "Auto Scaling group: NWENP3-AS" is open, showing the "Details" tab. It displays the following information:

Auto Scaling group name	Desired capacity	Status	Amazon Resource Name (ARN)
NWENP3-AS	4	Updating capacity	arn:aws:autoscaling:us-east-1:609310301062:autoScalingGroup:0348a24c-3b1f-439c-aa55-0e1c9

At the bottom of the page, there are links for "CloudShell", "Feedback", and copyright information: "© 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences".

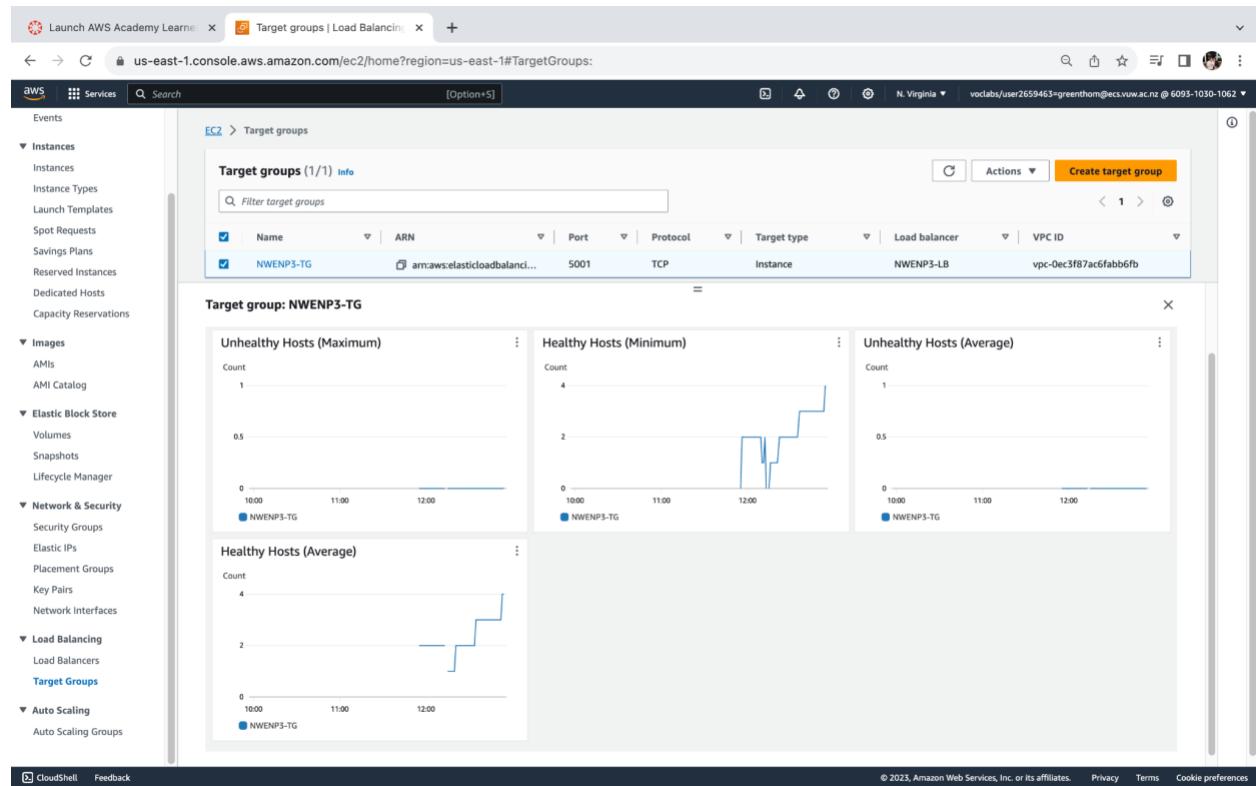
When auto scaling needs to launch a new instance, it has to create these instances based on the launch configuration I have defined. This process includes setting up the instance, installing the necessary software, and configuring it as specified. It can take some time, especially if my instances have complex setups or require specific configurations. Auto Scaling uses AMIs to create new instances. If my chosen AMI is large or not readily available, it can contribute to longer launch times. Auto scaling might introduce a delay between scaling activities.

The screenshot shows the AWS Management Console interface for the EC2 service. The left sidebar contains navigation links for Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images, AMIs, AMI Catalog, Elastic Block Store, Volumes, Snapshots, Lifecycle Manager, Network & Security, Security Groups, Elastic IPs, Placement Groups, Key Pairs, Network Interfaces, Load Balancing, Load Balancers, Target Groups, and Auto Scaling (selected). The main content area displays a table titled "Instances (8) Info" with the following columns: Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, Public IPv4 DNS, Public IPv4 IP, and Elastic IP. The table lists eight instances, each with a unique name and instance ID, their current state (Running or Terminated), instance type (t2.micro), and various status metrics. At the bottom of the table, there is a "Select an instance" dropdown menu. The top of the page shows the URL as "us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#Instances:" and the AWS logo.

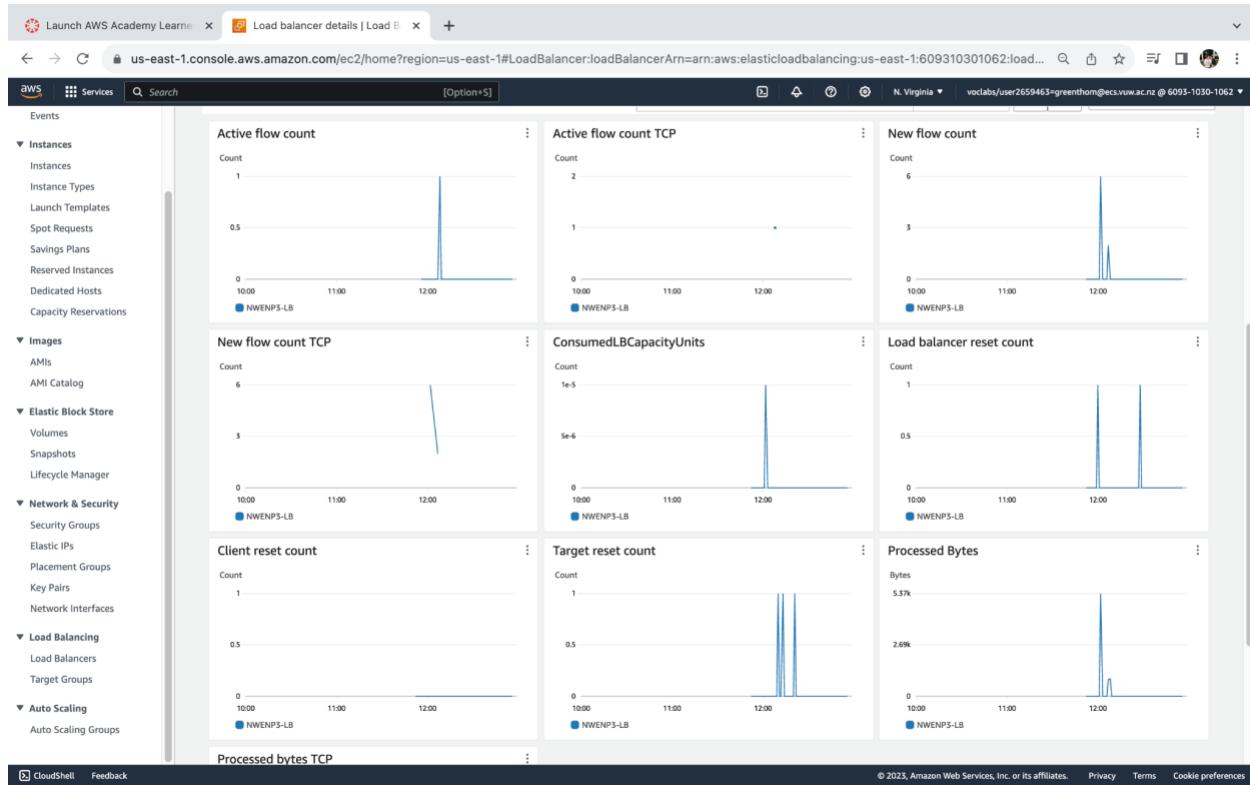
Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 IP	Elastic IP
-	i-06086aa211e13fd0	Terminated	t2.micro	-	No alarms	us-east-1b	-	-	-
-	i-0392875b9fd122850	Terminated	t2.micro	-	No alarms	us-east-1b	-	-	-
Public2	i-0bb9a72fa6b323ddd	Running	t2.micro	2/2 checks passed	No alarms	us-east-1b	ec2-44-204-211-29.co...	44.204.211.29	-
-	i-05e7a224fc073662	Running	t2.micro	2/2 checks passed	No alarms	us-east-1a	ec2-54-84-158-72.com...	54.84.158.72	-
Public1	i-0c6133e3d50be64d	Running	t2.micro	2/2 checks passed	No alarms	us-east-1a	ec2-54-243-2-31.comp...	54.243.2.31	-
-	i-0b8c606e21a8a89df	Terminated	t2.micro	-	No alarms	us-east-1a	-	-	-
-	i-0b4bec88a1626d8ea	Running	t2.micro	2/2 checks passed	No alarms	us-east-1a	ec2-35-173-219-71.co...	35.173.219.71	-
-	i-04e48d8ea43678d29	Running	t2.micro	2/2 checks passed	No alarms	us-east-1b	ec2-5-237-89-168.com...	3.237.89.168	-

Question 3: Why is it so slow?

The termination of instances serves as the trigger for the auto-scaling service to work in response to an unmet desired capacity. The service initiates an analysis of the configured policies to determine the appropriate course of action. However, this decision-making process causes a notable slowdown as it waits for the health-checker to ascertain the health status of instances where these health checks occur at regular intervals (30 seconds). The auto-scaler operates with a health check grace period of 300 seconds, this is implemented to cause scaling cooldowns to stop rapid and potentially unnecessary scaling. These cooldown periods create time delays between scaling actions helping the environment to stabilize due to the demand of new instances. Also have using t2.micro which has limited CPU power.



As we can see, this is the target group we set up for our instances. We can see that we have no unhealthy hosts based on our HealthCheck server. We can also know since we upped our autoscaling group's minimum value, we can see that we have four healthy running instances. Since we also upped our autoscaling group's average value, we can also see the counter of that increase showing we have healthy hosts. We can also see we have no unhealthy hosts (average). Load balancer metrics in CloudWatch show the request count and healthy host count, the load on the instances, and how it correlates with scaling actions.



Our CloudWatch metrics for the autoscaling group show the total number of instances. We can see how quickly the instances are added or terminated. The new flow count shows where I have been terminating and autoscaling, making more instances. We can see that the healthy hosts drop briefly before the AWS makes up a new instance. You can see little dips making and removing instances.

PART B

Here, we create an instance from our Project3AMI that was made at the start of project 3a)

The screenshot shows the AWS EC2 Instances page. On the left, a sidebar navigation menu includes: EC2 Dashboard, EC2 Global View, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images (AMIs, AMI Catalog), Elastic Block Store (Volumes, Snapshots, Lifecycle Manager), Network & Security (Security Groups, Elastic IPs, Placement Groups, Key Pairs, Network Interfaces), and Load Balancing (Load Balancers). The main content area displays a table of instances. A new instance, 'NWEN_TAKES' (ID: i-022d82db1395e2ac6), is being created, indicated by the 'Running' status and a progress bar. Other instances listed include Public1, NWEN_P3_PARTB, NWENPARTB_TAKE3, and several terminated instances. The top right of the page shows the region as N. Virginia and the user as vclabs/user2659463=greenthom@ecsvvuw.ac.nz @ 6093-1030-1062.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...
-	i-06619d14056433d0e	Stopped	t2.micro	-	No alarms	+ us-east-1b	-	-
-	i-0356de335131a377	Stopped	t2.micro	-	No alarms	+ us-east-1a	-	-
Public1	i-0c6133ec3d5dbe64d	Terminated	t2.micro	-	No alarms	+ us-east-1a	-	-
-	i-03f270d10522835e	Stopped	t2.micro	-	No alarms	+ us-east-1a	-	-
NWEN_P3_PARTB	i-0a4b2fe0a713e08d8	Terminated	t2.micro	-	No alarms	+ us-east-1c	-	-
NWENPARTB_TAKE3	i-037dca86ff2add5e1	Terminated	t2.micro	-	No alarms	+ us-east-1c	-	-
-	i-0a16142189974605c	Stopped	t2.micro	Initializing	No alarms	+ us-east-1b	-	-
NWEN_TAKES	i-022d82db1395e2ac6	Running	t2.micro	-	No alarms	+ us-east-1c	ec2-44-204-181-142.co...	44.204.181.142...

Instance: i-022d82db1395e2ac6 (NWEN_TAKES)

Details | Security | Networking | Storage | Status checks | Monitoring | Tags

Instance summary

Instance ID	i-022d82db1395e2ac6 (NWEN_TAKES)	Public IPv4 address	44.204.181.142 [open address]	Private IPv4 addresses	172.31.80.118
IPv6 address	-	Instance state	Running	Public IPv4 DNS	ec2-44-204-181-142.compute-1.amazonaws.com [open address]
Hostname type	IP name: ip-172-31-80-118.ec2.internal	Private IP DNS name (IPv4 only)	ip-172-31-80-118.ec2.internal	Elastic IP addresses	-
Answer private resource DNS name	IPV4 (A)	Instance type	t2.micro	AWS Compute Optimizer finding	Opt-in to AWS Compute Optimizer for recommendations. Learn more
Auto-assigned IP address	44.204.181.142 [Public IP]	VPC ID	vpc-05900b3f0d4317157	Auto Scaling Group name	-
IAM Role	-	Subnet ID	subnet-0d2f319d9ab1dfe94		
IMDSv2					

From here, we ssh into our instance

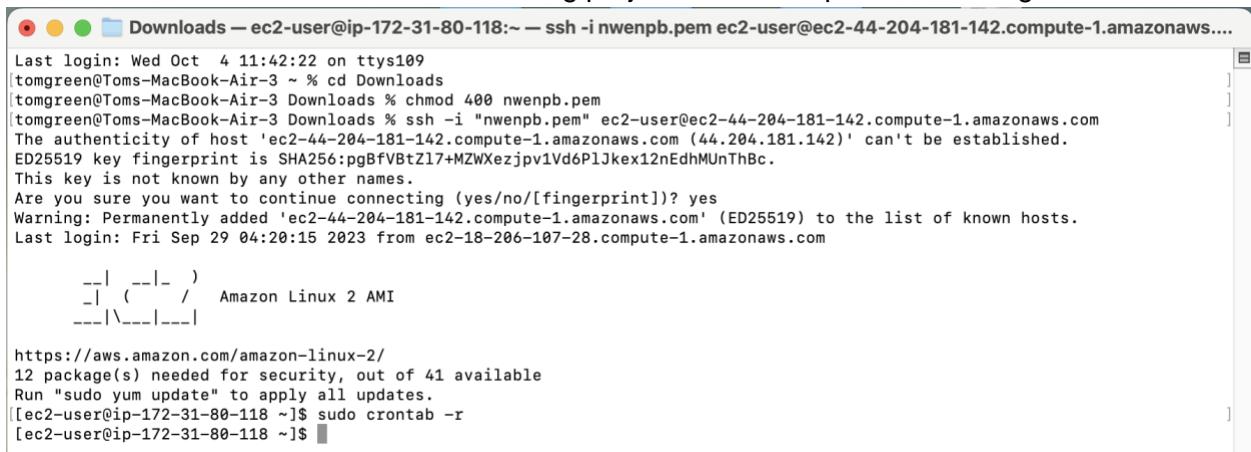


```
Last login: Wed Oct 4 11:42:22 on ttys109
[tomgreen@Toms-MacBook-Air-3 ~ % cd Downloads
[tomgreen@Toms-MacBook-Air-3 Downloads % chmod 400 nwenpb.pem
[tomgreen@Toms-MacBook-Air-3 Downloads % ssh -i "nwenpb.pem" ec2-user@ec2-44-204-181-142.compute-1.amazonaws.com
The authenticity of host 'ec2-44-204-181-142.compute-1.amazonaws.com (44.204.181.142)' can't be established.
ED25519 key fingerprint is SHA256:pgBfVbtZl7+MZWXezjpv1Vd6PlJkex12nEdhMuNThBc.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-44-204-181-142.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
Last login: Fri Sep 29 04:20:15 2023 from ec2-18-206-107-28.compute-1.amazonaws.com

--| --|- )
_| ( _ / Amazon Linux 2 AMI
---|\---|---|]

https://aws.amazon.com/amazon-linux-2/
12 package(s) needed for security, out of 41 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-80-118 ~]$
```

We must remove the crontab created during project two so the port is not being used.



```
Last login: Wed Oct 4 11:42:22 on ttys109
[tomgreen@Toms-MacBook-Air-3 ~ % cd Downloads
[tomgreen@Toms-MacBook-Air-3 Downloads % chmod 400 nwenpb.pem
[tomgreen@Toms-MacBook-Air-3 Downloads % ssh -i "nwenpb.pem" ec2-user@ec2-44-204-181-142.compute-1.amazonaws.com
The authenticity of host 'ec2-44-204-181-142.compute-1.amazonaws.com (44.204.181.142)' can't be established.
ED25519 key fingerprint is SHA256:pgBfVbtZl7+MZWXezjpv1Vd6PlJkex12nEdhMuNThBc.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-44-204-181-142.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
Last login: Fri Sep 29 04:20:15 2023 from ec2-18-206-107-28.compute-1.amazonaws.com

--| --|- )
_| ( _ / Amazon Linux 2 AMI
---|\---|---|]

https://aws.amazon.com/amazon-linux-2/
12 package(s) needed for security, out of 41 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-80-118 ~]$ sudo crontab -r
[ec2-user@ip-172-31-80-118 ~]$
```

Next, we run sudo yum update to run a general update on the instance

Last login: Wed Oct 4 11:42:22 on ttys109
[tomgreen@Toms-MacBook-Air-3 ~ % cd Downloads
[tomgreen@Toms-MacBook-Air-3 Downloads % chmod 400 nwenpb.pem
[tomgreen@Toms-MacBook-Air-3 Downloads % ssh -i "nwenpb.pem" ec2-user@ec2-44-204-181-142.compute-1.amazonaws.com
The authenticity of host 'ec2-44-204-181-142.compute-1.amazonaws.com (44.204.181.142)' can't be established.
ED25519 key fingerprint is SHA256:pgBFvBtZ17+MZXezjpv1Vd6PlJkex12nEdhMUUnThBc.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-44-204-181-142.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
Last login: Fri Sep 29 04:20:15 2023 from ec2-18-206-107-28.compute-1.amazonaws.com

```
--| --|- )  
_| ( _ / Amazon Linux 2 AMI  
---\_\_|-|_--|  
  
https://aws.amazon.com/amazon-linux-2/  
12 package(s) needed for security, out of 41 available  
Run "sudo yum update" to apply all updates.  
[ec2-user@ip-172-31-80-118 ~]$ sudo crontab -r  
[ec2-user@ip-172-31-80-118 ~]$ sudo yum update  
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd  
Resolving Dependencies  
--> Running transaction check  
---- Package bind-export-libs.x86_64 32:9.11.4-26.P2.amzn2.13.3 will be updated  
---- Package bind-export-libs.x86_64 32:9.11.4-26.P2.amzn2.13.4 will be an update  
---- Package bind-libs.x86_64 32:9.11.4-26.P2.amzn2.13.3 will be updated  
---- Package bind-libs.x86_64 32:9.11.4-26.P2.amzn2.13.4 will be an update  
---- Package bind-libs-lite.x86_64 32:9.11.4-26.P2.amzn2.13.3 will be updated  
---- Package bind-libs-lite.x86_64 32:9.11.4-26.P2.amzn2.13.4 will be an update  
---- Package bind-license.noarch 32:9.11.4-26.P2.amzn2.13.3 will be updated  
---- Package bind-license.noarch 32:9.11.4-26.P2.amzn2.13.4 will be an update  
---- Package bind-utils.x86_64 32:9.11.4-26.P2.amzn2.13.3 will be updated  
---- Package bind-utils.x86_64 32:9.11.4-26.P2.amzn2.13.4 will be an update  
---- Package curl.x86_64 0:8.2.1-1.amzn2.0.2 will be updated  
---- Package curl.x86_64 0:8.2.1-1.amzn2.0.3 will be an update  
---- Package dhclient.x86_64 12:4.2.5-79.amzn2.1.4 will be updated  
---- Package dhclient.x86_64 12:4.2.5-79.amzn2.1.5 will be an update  
---- Package dhcp-common.x86_64 12:4.2.5-79.amzn2.1.4 will be updated  
---- Package dhcp-common.x86_64 12:4.2.5-79.amzn2.1.5 will be an update  
---- Package dhcp-libs.x86_64 12:4.2.5-79.amzn2.1.4 will be an update  
---- Package elfutils-default-yama-scope.noarch 0:0.176-2.amzn2.0.1 will be updated  
---- Package elfutils-default-yama-scope.noarch 0:0.176-2.amzn2.0.2 will be an update  
---- Package elfutils-libelf.x86_64 0:0.176-2.amzn2.0.1 will be updated
```

Next, we run 'sudo amazon-linux-extras install docker'

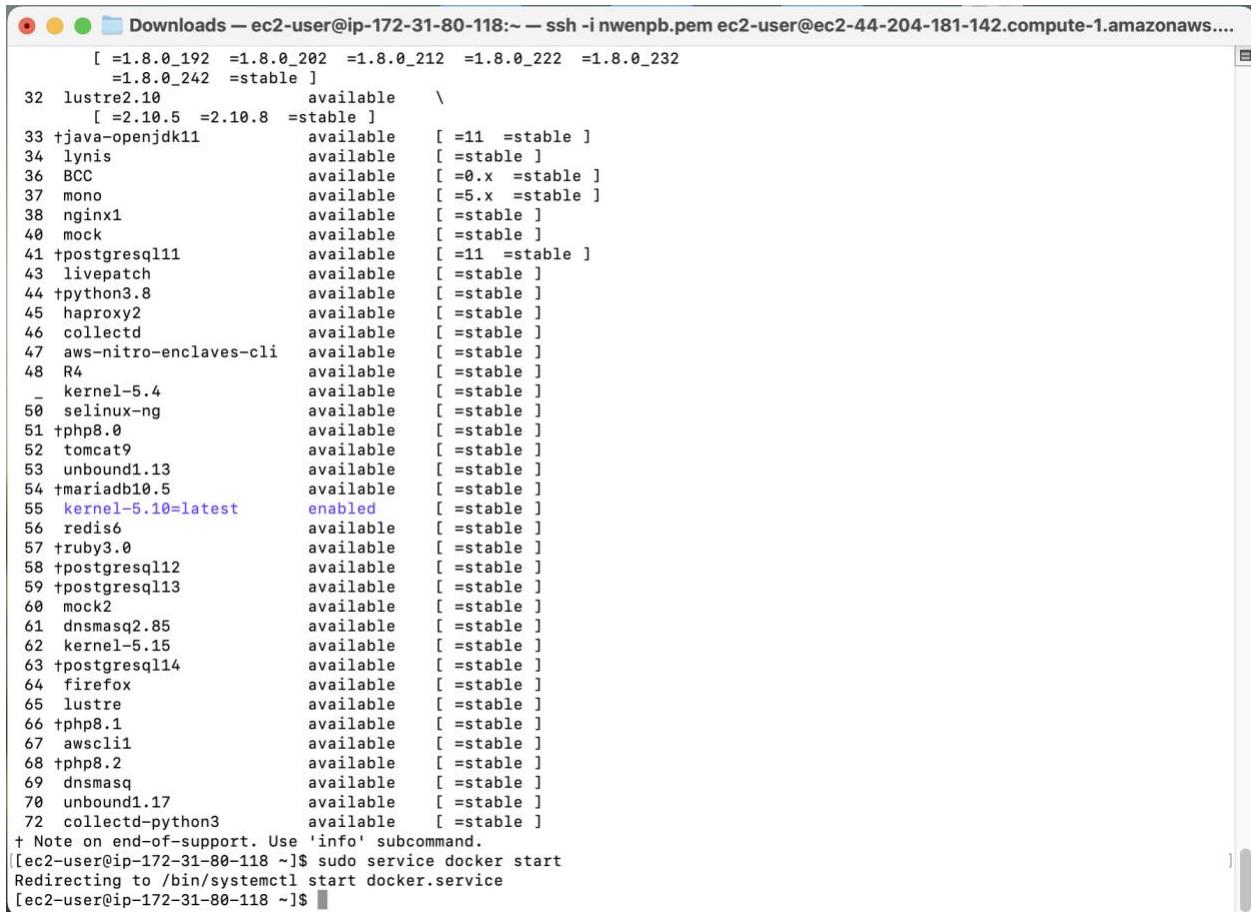
```
Downloads — ec2-user@ip-172-31-80-118:~ — ssh -i nwenpb.pem ec2-user@ec2-44-204-181-142.compute-1.amazonaws....
```

```
[ec2-user@ip-172-31-80-118 ~]$ sudo amazon-linux-extras install docker
Installing docker
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Cleaning repos: amzn2-core amzn2extra-docker amzn2extra-kernel-5.10
21 metadata files removed
6 sqlite files removed
0 metadata files removed
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core                                         | 3.6 kB  00:00:00
amzn2extra-docker                                | 3.0 kB  00:00:00
amzn2extra-kernel-5.10                           | 3.0 kB  00:00:00
(1/7): amzn2-core/2/x86_64/group_gz              | 2.7 kB  00:00:00
(2/7): amzn2-core/2/x86_64/updateinfo            | 707 kB   00:00:00
(3/7): amzn2extra-kernel-5.10/2/x86_64/primary   | 8.5 MB   00:00:00
(4/7): amzn2extra-docker/2/x86_64/updateinfo      | 13 kB   00:00:00
(5/7): amzn2extra-kernel-5.10/2/x86_64/updateinfo | 37 kB   00:00:00
(6/7): amzn2extra-docker/2/x86_64/primary_db       | 111 kB   00:00:00
(7/7): amzn2-core/2/x86_64/primary                | 30 MB   00:00:00
Resolving Dependencies
--> Running transaction check
--> Package docker.x86_64 0:20.10.23-1.amzn2.0.1 will be installed
--> Processing Dependency: runc >= 1.0.0 for package: docker-20.10.23-1.amzn2.0.1.x86_64
--> Processing Dependency: libcgroup >= 0.40.rc1-5.15 for package: docker-20.10.23-1.amzn2.0.1.x86_64
--> Processing Dependency: containerd >= 1.3.2 for package: docker-20.10.23-1.amzn2.0.1.x86_64
--> Processing Dependency: pigz for package: docker-20.10.23-1.amzn2.0.1.x86_64
--> Running transaction check
--> Package containerd.x86_64 0:1.6.19-1.amzn2.0.3 will be installed
--> Package libcgroup.x86_64 0:0.41-21.amzn2 will be installed
--> Package pigz.x86_64 0:2.3.4-1.amzn2.0.1 will be installed
--> Package runc.x86_64 0:1.1.7-3.amzn2 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package          Arch      Version           Repository      Size
=====
Installing:
 docker           x86_64    20.10.23-1.amzn2.0.1      amzn2extra-docker  41 M
Installing for dependencies:
 containerd        x86_64    1.6.19-1.amzn2.0.3      amzn2extra-docker  28 M
 libcgroup         x86_64    0.41-21.amzn2          amzn2-core          66 k
 pigz             x86_64    2.3.4-1.amzn2.0.1      amzn2-core          81 k
 runc             x86_64    1.1.7-3.amzn2        amzn2extra-docker  3.0 M
```

Next, we start the docker by running sudo service docker start

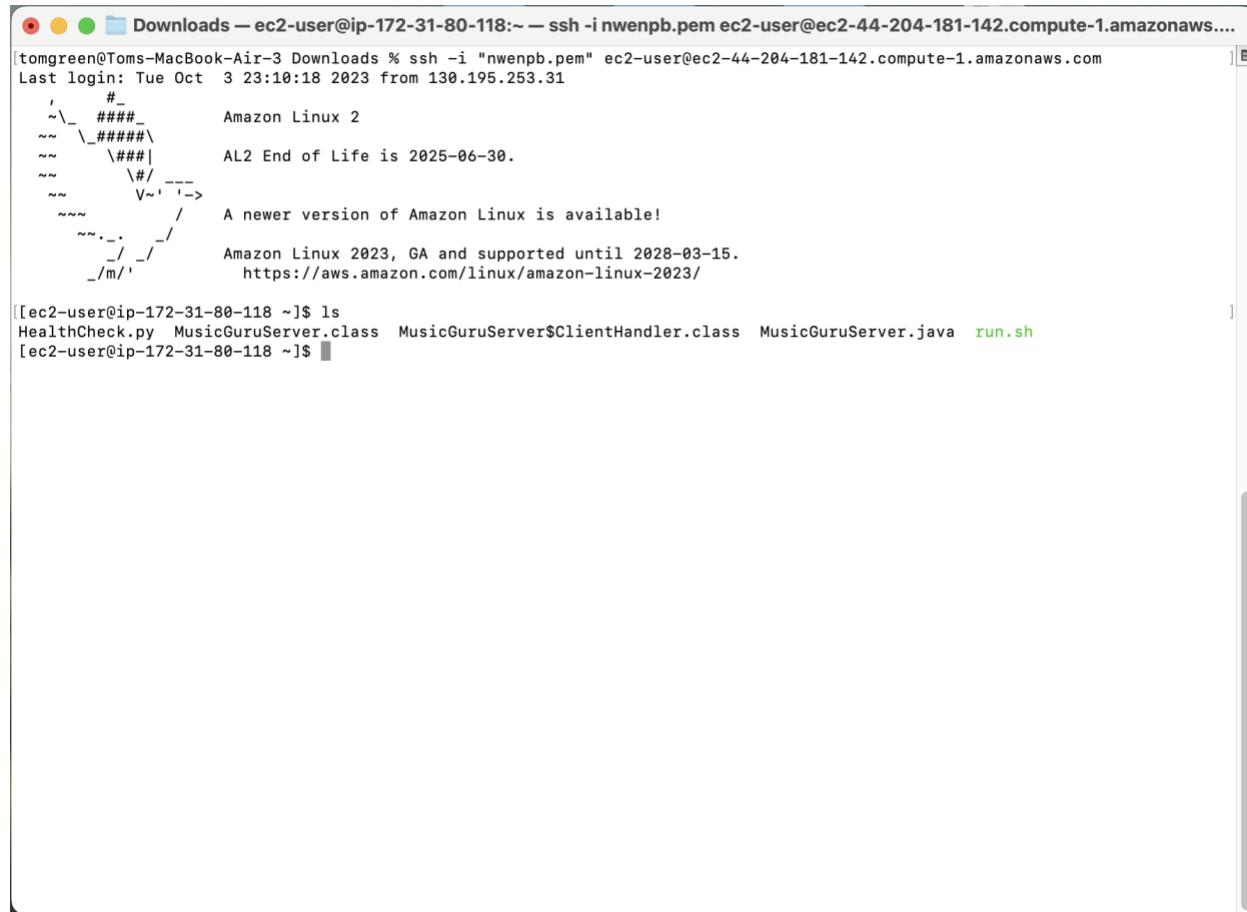


```
[=1.8.0_192  =1.8.0_202  =1.8.0_212  =1.8.0_222  =1.8.0_232
 =1.8.0_242  =stable ]
32 lustre2.10      available  \
[ =2.10.5  =2.10.8  =stable ]
33 tjava-openjdk11  available  [ =11  =stable ]
34 lynis           available  [ =stable ]
36 BCC            available  [ =0.x  =stable ]
37 mono           available  [ =5.x  =stable ]
38 nginx1         available  [ =stable ]
40 mock           available  [ =stable ]
41 tpostgresql11  available  [ =11  =stable ]
43 livepatch       available  [ =stable ]
44 tpython3.8      available  [ =stable ]
45 haproxy2        available  [ =stable ]
46 collectd        available  [ =stable ]
47 aws-nitro-enclaves-cli  available  [ =stable ]
48 R4              available  [ =stable ]
~ kernel-5.4      available  [ =stable ]
50 selinux-ng      available  [ =stable ]
51 tphp8.0         available  [ =stable ]
52 tomcat9        available  [ =stable ]
53 unbound1.13    available  [ =stable ]
54 tmariadb10.5   available  [ =stable ]
55 kernel-5.10=latest  enabled   [ =stable ]
56 redis6          available  [ =stable ]
57 truby3.0        available  [ =stable ]
58 tpostgresql12  available  [ =stable ]
59 tpostgresql13  available  [ =stable ]
60 mock2           available  [ =stable ]
61 dnsmasq2.85   available  [ =stable ]
62 kernel-5.15     available  [ =stable ]
63 tpostgresql14  available  [ =stable ]
64 firefox         available  [ =stable ]
65 lustre          available  [ =stable ]
66 tphp8.1         available  [ =stable ]
67 awscli1         available  [ =stable ]
68 tphp8.2         available  [ =stable ]
69 dnsmasq         available  [ =stable ]
70 unbound1.17    available  [ =stable ]
72 collectd-python3  available  [ =stable ]
+ Note on end-of-support. Use 'info' subcommand.
[ec2-user@ip-172-31-80-118 ~]$ sudo service docker start
Redirecting to /bin/systemctl start docker.service
[ec2-user@ip-172-31-80-118 ~]$
```

Next, we run ‘sudo usermod -a -G docker ec2-user’ to add the default user. From here, we then exit so it refreshes the permissions we just set

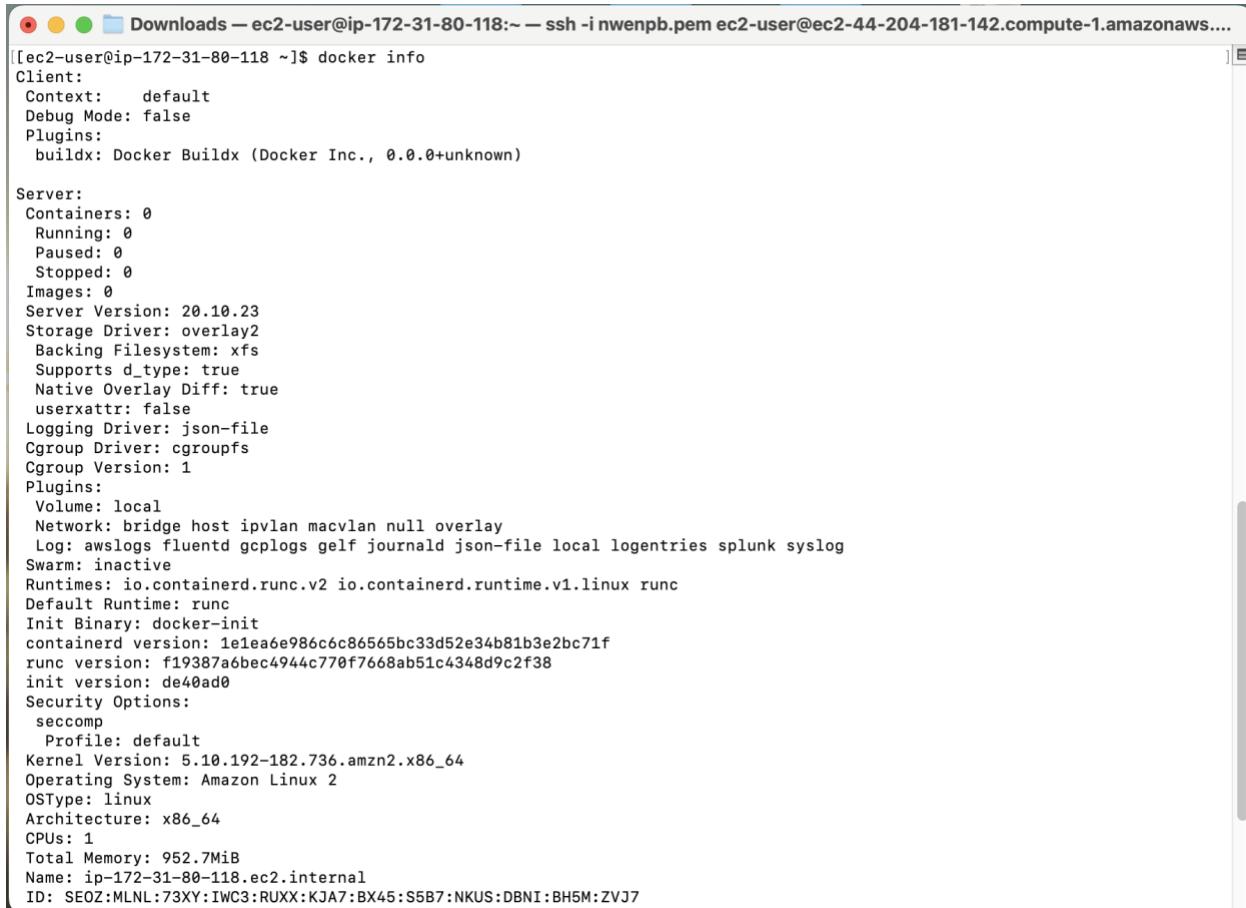
```
Downloads -- ec2-user@ip-172-31-80-118:~ - zsh - 124x44
33 tjava-openjdk11      available  [ =11  =stable ]
34 lynis                available  [ =stable ]
36 BCC                  available  [ =0.x  =stable ]
37 mono                 available  [ =5.x  =stable ]
38 nginx1x              available  [ =stable ]
40 mock                 available  [ =stable ]
41 tpostgresql11        available  [ =11  =stable ]
43 livepatch             available  [ =stable ]
44 tpython3.8             available  [ =stable ]
45 haproxy2              available  [ =stable ]
46 collectd              available  [ =stable ]
47 aws-nitro-enclaves-cl available  [ =stable ]
48 R4                   available  [ =stable ]
- kernel-5.4              available  [ =stable ]
50 selinux-ng             available  [ =stable ]
51 tphp8.0               available  [ =stable ]
52 tomcat9               available  [ =stable ]
53 unbound1.13            available  [ =stable ]
54 +mariadb10.5           available  [ =stable ]
55 kernel-5.10=latest     enabled    [ =stable ]
56 redis6                available  [ =stable ]
57 ruby3.0                available  [ =stable ]
58 tpostgresql12           available  [ =stable ]
59 tpostgresql13           available  [ =stable ]
60 mock2                 available  [ =stable ]
61 dnsmasq2.85             available  [ =stable ]
62 kernel-5.15              available  [ =stable ]
63 tpostgresql14           available  [ =stable ]
64 firefox                available  [ =stable ]
65 lustre                 available  [ =stable ]
66 tphp8.1                available  [ =stable ]
67 awscli1                available  [ =stable ]
68 tphp8.2                available  [ =stable ]
69 dnsmasq                available  [ =stable ]
70 unbound1.17             available  [ =stable ]
72 collectd-python3          available  [ =stable ]
+ Note on end-of-support. Use 'info' subcommand.
[ec2-user@ip-172-31-80-118 ~]$ sudo service docker start
Redirecting to /bin/systemctl start docker.service
[ec2-user@ip-172-31-80-118 ~]$ sudo usermod -a -G docker ec2-user
[ec2-user@ip-172-31-80-118 ~]$ exit
logout
Connection to ec2-44-204-181-142.compute-1.amazonaws.com closed.
tomgreen@Toms-MacBook-Air-3 Downloads %
```

We then ssh back into the instance



```
[tomgreen@Toms-MacBook-Air-3 Downloads -- ec2-user@ip-172-31-80-118:~ -- ssh -i "nwenpb.pem" ec2-user@ec2-44-204-181-142.compute-1.amazonaws.com]
Last login: Tue Oct  3 23:10:18 2023 from 130.195.253.31
'      #
~\_ #####_      Amazon Linux 2
~~ \_#####\_
~~  \###|      AL2 End of Life is 2025-06-30.
~~   \#/      --
~~   V~/  '-->
~~   /      A newer version of Amazon Linux is available!
~~ .-. /      /
~/m/  /      Amazon Linux 2023, GA and supported until 2028-03-15.
https://aws.amazon.com/linux/amazon-linux-2023/
[ec2-user@ip-172-31-80-118 ~]$ ls
HealthCheck.py  MusicGuruServer.class  MusicGuruServer$ClientHandler.class  MusicGuruServer.java  run.sh
[ec2-user@ip-172-31-80-118 ~]$
```

From here, we use the command docker info, where we can see a status dump where we can see that no containers are hosted



```
[ec2-user@ip-172-31-80-118 ~]$ docker info
Client:
  Context:    default
  Debug Mode: false
  Plugins:
    buildx: Docker Buildx (Docker Inc., 0.0.0+unknown)

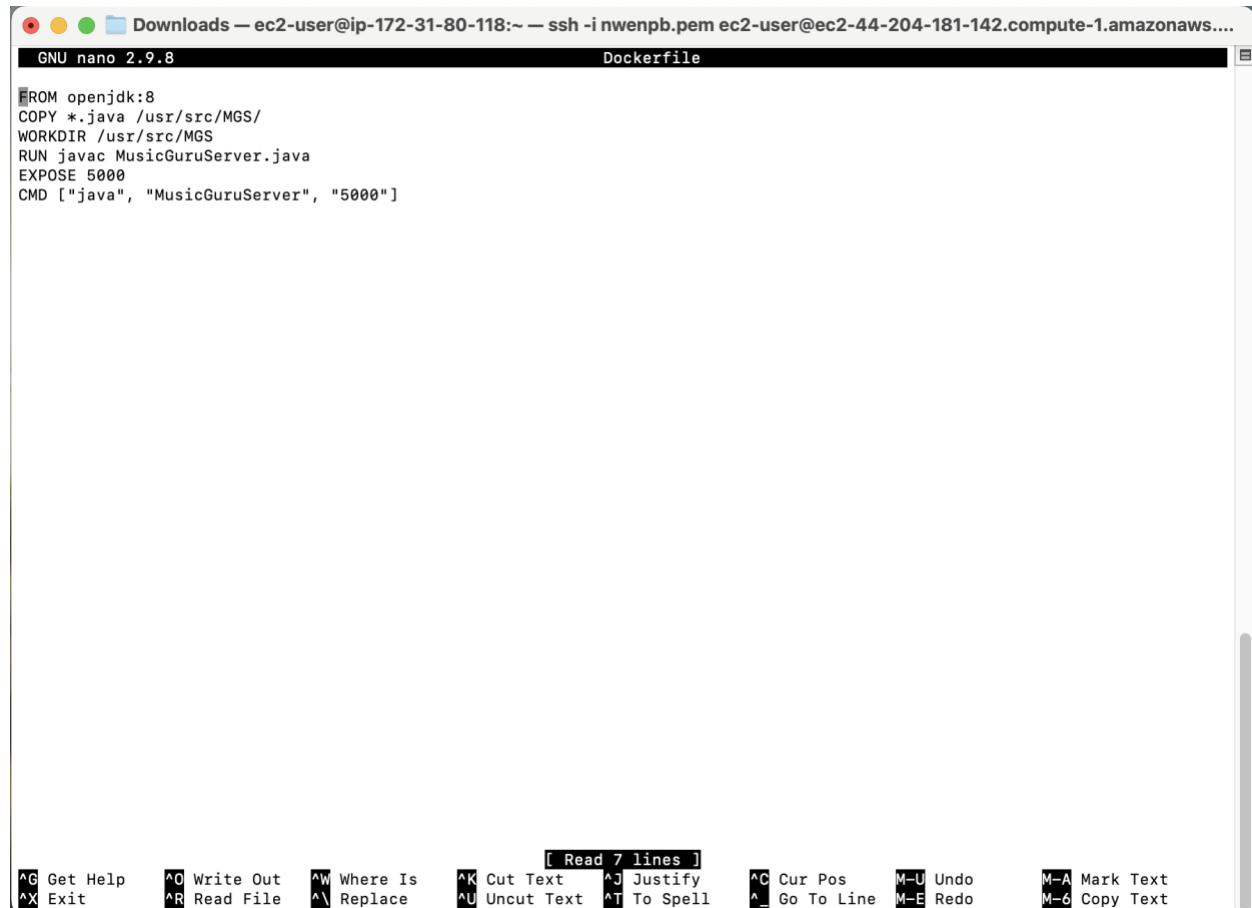
Server:
  Containers: 0
  Running: 0
  Paused: 0
  Stopped: 0
  Images: 0
  Server Version: 20.10.23
  Storage Driver: overlay2
    Backing Filesystem: xfs
    Supports d_type: true
    Native Overlay Diff: true
  userxattr: false
  Logging Driver: json-file
  Cgroup Driver: cgroupfs
  Cgroup Version: 1
  Plugins:
    Volume: local
    Network: bridge host ipvlan macvlan null overlay
    Log: awslogs fluentd gcplogs gelf journalctl json-file local logentries splunk syslog
  Swarm: inactive
  Runtimes: io.containerd.runc.v2 io.containerd.runtime.v1.linux runc
  Default Runtime: runc
  Init Binary: docker-init
  containerd version: 1e1ea6e986c6c86565bc33d52e34b81b3e2bc71f
  runc version: f19387a6bec4944c770f7668ab51c4348d9c2f38
  init version: de40ad0
  Security Options:
    seccomp
      Profile: default
  Kernel Version: 5.10.192-182.736.amzn2.x86_64
  Operating System: Amazon Linux 2
  OSType: linux
  Architecture: x86_64
  CPUs: 1
  Total Memory: 952.7MiB
  Name: ip-172-31-80-118.ec2.internal
  ID: SEOZ:MLNL:73XY:IWC3:RUXX:KJA7:BX45:S5B7:NKUS:DBNI:BH5M:ZVJ7
```

Now, what we are doing is creating a 'Dockerfile.'



```
[ec2-user@ip-172-31-80-118 ~]$ ls
HealthCheck.py  MusicGuruServer.class  MusicGuruServer$ClientHandler.class  MusicGuruServer.java  run.sh
[ec2-user@ip-172-31-80-118 ~]$ nano Dockerfile
```

From here, we use the Java Dockerfile. Since we did not use a text/JSON file to hold our songs, we do not need to add the second COPY line.



The screenshot shows a terminal window titled "Downloads — ec2-user@ip-172-31-80-118:~ — ssh -i nwenpb.pem ec2-user@ec2-44-204-181-142.compute-1.amazonaws...." with the file name "Dockerfile". The terminal is running the command-line editor "GNU nano 2.9.8". The Dockerfile contains the following code:

```
FROM openjdk:8
COPY *.java /usr/src/MGS/
WORKDIR /usr/src/MGS
RUN javac MusicGuruServer.java
EXPOSE 5000
CMD ["java", "MusicGuruServer", "5000"]
```

At the bottom of the screen, there is a status bar with various keyboard shortcuts for nano, including:

- [Read 7 lines]
- Get Help (^G)
- Write Out (^O)
- Where Is (^W)
- Cut Text (^K)
- Justify (^J)
- Cur Pos (^C)
- Undo (^U)
- Exit (^X)
- Read File (^R)
- Replace (^V)
- Uncut Text (^U)
- To Spell (^T)
- Go To Line (^L)
- Redo (^E)
- Mark Text (^A)
- Copy Text (^M-6)

From here, we build the docker using the command 'docker build -t mgs /home/ec2-user'. As we can see, the build was successful

```
Downloads — ec2-user@ip-172-31-80-118:~ — ssh -i nwenpb.pem ec2-user@ec2-44-204-181-142.compute-1.amazonaws....  
[ec2-user@ip-172-31-80-118 ~]$ ls  
HealthCheck.py MusicGuruServer.class MusicGuruServer$ClientHandler.class MusicGuruServer.java run.sh  
[ec2-user@ip-172-31-80-118 ~]$ nano Dockerfile  
[ec2-user@ip-172-31-80-118 ~]$ docker build -t mgs.  
invalid argument "mgs." for "-t, --tag" flag: invalid reference format  
See 'docker build --help'.  
[ec2-user@ip-172-31-80-118 ~]$ docker build -t mgs /home/ec2-user  
Sending build context to Docker daemon 10.41MB  
Step 1/6 : FROM openjdk:8  
8: Pulling from library/openjdk  
001c52e26ad5: Pull complete  
d9d4b9b6e964: Pull complete  
2068746827ec: Pull complete  
9daef329d350: Pull complete  
d85151f15b66: Pull complete  
52a8c426d30b: Pull complete  
8754a66e0050: Pull complete  
Digest: sha256:86e863cc57215cfb181bd319736d0baf625fe8f150577f9eb58bd937f5452cb8  
Status: Downloaded newer image for openjdk:8  
--> b273004037cc  
Step 2/6 : COPY *.java /usr/src/MGS/  
--> 0e0c481bcdb2  
Step 3/6 : WORKDIR /usr/src/MGS  
--> Running in f6f3fadf51dd  
Removing intermediate container f6f3fadf51dd  
--> 45f06fd93cda  
Step 4/6 : RUN javac MusicGuruServer.java  
--> Running in b3d7f39ffe90  
Removing intermediate container b3d7f39ffe90  
--> e50ec08fa3e7  
Step 5/6 : EXPOSE 5001  
--> Running in fe68bd1e3a5c  
Removing intermediate container fe68bd1e3a5c  
--> b8da1aaef2089  
Step 6/6 : CMD ["java", "MusicGuruServer", "5001"]  
--> Running in 68d319659907  
Removing intermediate container 68d319659907  
--> e7def129d086  
Successfully built e7def129d086  
Successfully tagged mgs:latest  
[ec2-user@ip-172-31-80-118 ~]$
```

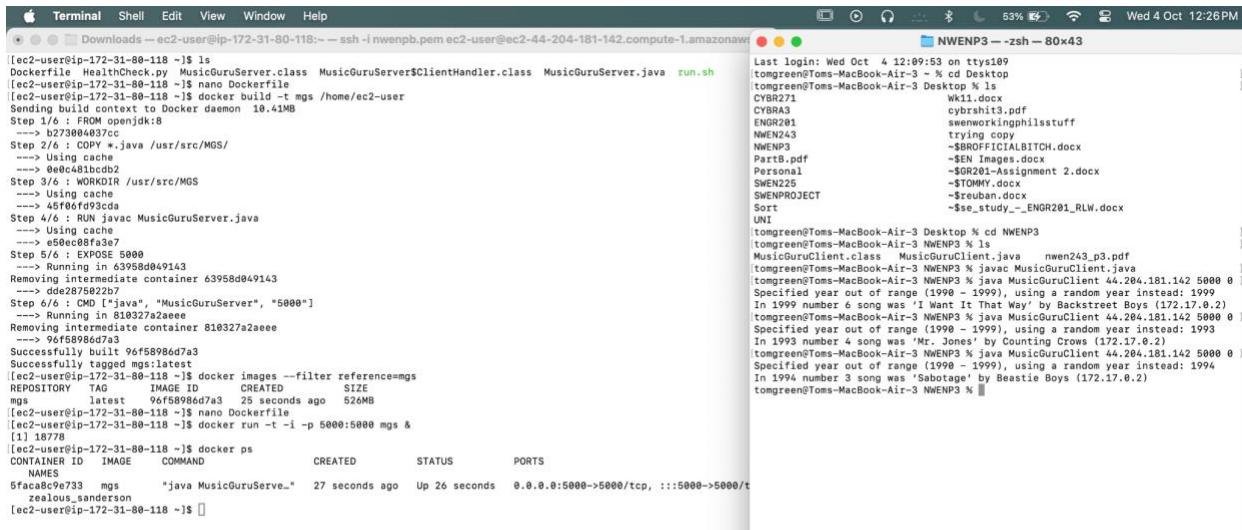
From here, we can check if the docker builds the image properly using the command 'docker images --filter reference=mgs.' As we can see, it made our mgs successfully. Next, we can finally run the built image. We exposed the dockerfile at port 5000. We need to map the exposed port on the container to a port on the host machine. We do that by running the command 'docker run -t -i -p 5000:5000 mgs &'. This gives us a response that says we have job one running in the background, and its PID is '18778'.

```
Downloads — ec2-user@ip-172-31-80-118:~ — ssh -i nwenpb.pem ec2-user@ec2-44-204-181-142.compute-1.amazonaws....  
[[ec2-user@ip-172-31-80-118 ~]$ ls  
Dockerfile  HealthCheck.py  MusicGuruServer.class  MusicGuruServer$ClientHandler.class  MusicGuruServer.java  run.sh  
[[ec2-user@ip-172-31-80-118 ~]$ nano Dockerfile  
[[ec2-user@ip-172-31-80-118 ~]$ docker build -t mgs /home/ec2-user  
Sending build context to Docker daemon 10.41MB  
Step 1/6 : FROM openjdk:8  
--> b273004037cc  
Step 2/6 : COPY *.java /usr/src/MGS/  
--> Using cache  
--> 0e0c481bcdb2  
Step 3/6 : WORKDIR /usr/src/MGS  
--> Using cache  
--> 45f06fd93cda  
Step 4/6 : RUN javac MusicGuruServer.java  
--> Using cache  
--> e50ec08fa3e7  
Step 5/6 : EXPOSE 5000  
--> Running in 63958d049143  
Removing intermediate container 63958d049143  
--> dde2875022b7  
Step 6/6 : CMD ["java", "MusicGuruServer", "5000"]  
--> Running in 810327a2aeee  
Removing intermediate container 810327a2aeee  
--> 96f58986d7a3  
Successfully built 96f58986d7a3  
Successfully tagged mgs:latest  
[[ec2-user@ip-172-31-80-118 ~]$ docker images --filter reference=mgs  
REPOSITORY TAG IMAGE ID CREATED SIZE  
mgs latest 96f58986d7a3 25 seconds ago 526MB  
[[ec2-user@ip-172-31-80-118 ~]$ nano Dockerfile  
[[ec2-user@ip-172-31-80-118 ~]$ docker run -t -i -p 5000:5000 mgs &  
[1] 18778  
[ec2-user@ip-172-31-80-118 ~]$ ]
```

From here, we run the command docker ps to see if everything is running. We can see the port mapping, the command image, and the container ID showing us that this was successful.

```
[ec2-user@ip-172-31-80-118 ~]$ ls
Dockerfile HealthCheck.py MusicGuruServer.class MusicGuruServer$ClientHandler.class MusicGuruServer.java run.sh
[ec2-user@ip-172-31-80-118 ~]$ nano Dockerfile
[ec2-user@ip-172-31-80-118 ~]$ docker build -t mgs /home/ec2-user
Sending build context to Docker daemon 10.41MB
Step 1/6 : FROM openjdk:8
--> b273004037cc
Step 2/6 : COPY *.java /usr/src/MGS/
--> Using cache
--> 0e0c481bcd2
Step 3/6 : WORKDIR /usr/src/MGS
--> Using cache
--> 45f06fd93cda
Step 4/6 : RUN javac MusicGuruServer.java
--> Using cache
--> e50ec08fa3e7
Step 5/6 : EXPOSE 5000
--> Running in 63958d049143
Removing intermediate container 63958d049143
--> dde2875022b7
Step 6/6 : CMD ["java", "MusicGuruServer", "5000"]
--> Running in 810327a2aeee
Removing intermediate container 810327a2aeee
--> 96f58986d7a3
Successfully built 96f58986d7a3
Successfully tagged mgs:latest
[ec2-user@ip-172-31-80-118 ~]$ docker images --filter reference=mgs
REPOSITORY TAG IMAGE ID CREATED SIZE
mgs latest 96f58986d7a3 25 seconds ago 526MB
[ec2-user@ip-172-31-80-118 ~]$ nano Dockerfile
[ec2-user@ip-172-31-80-118 ~]$ docker run -t -i -p 5000:5000 mgs &
[1] 18778
[ec2-user@ip-172-31-80-118 ~]$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
NAMES
5fac8c9e733 mgs "java MusicGuruServe..." 27 seconds ago Up 26 seconds 0.0.0.0:5000->5000/tcp, :::5000->5000/tcp
zealous_sanderson
[ec2-user@ip-172-31-80-118 ~]$
```

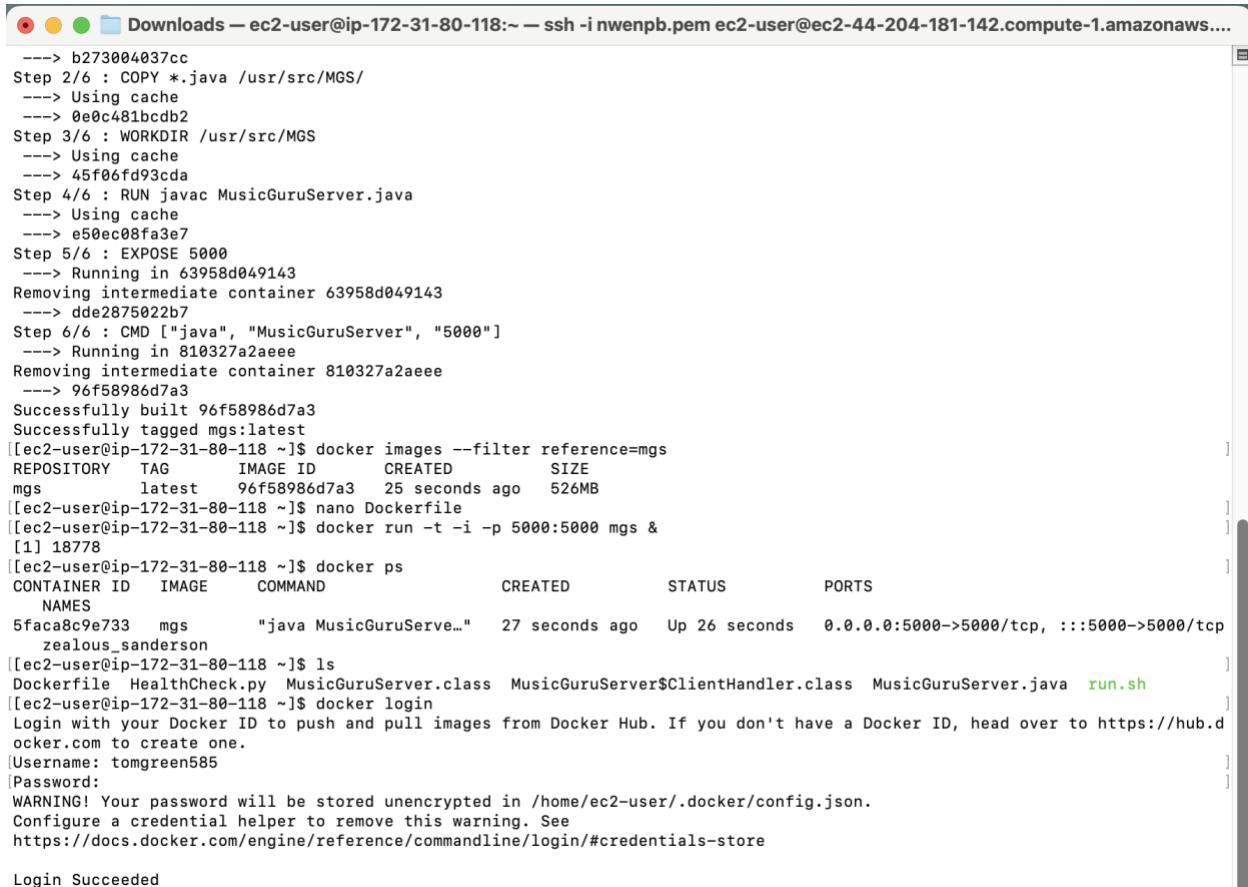
From here, we run the original client from my local machine using the public IP of the instance, the port number that the docker is running on, and the year. As we can see, this was successful.



```
Terminal Shell Edit View Window Help
Downloads -- ec2-user@ip-172-31-80-118:~ ssh -i nwenpb.pem ec2-user@ec2-44-204-181-142.compute-1.amazonaws.com
Last login: Wed Oct 4 12:09:53 on ttys109
tomgreen@Toms-MacBook-Air-3 % cd Desktop
tomgreen@Toms-MacBook-Air-3 Desktop % ls
CVB77
CYBR43
ENGR201
NWEN243
NWENP3
PartB.pdf
Personal
SWE225
SWEPROJECT
Sort
UNI
tomgreen@Toms-MacBook-Air-3 Desktop % cd NWENP3
tomgreen@Toms-MacBook-Air-3 NWENP3 % ls
MusicGuruClient.class MusicGuruClient.java nwe243.p3.pdf
tomgreen@Toms-MacBook-Air-3 NWENP3 % java MusicGuruClient
tomgreen@Toms-MacBook-Air-3 NWENP3 % java MusicGuruClient 44.204.181.142 5000 0
Specified year out of range (1998 - 1999), using a random year instead: 1999
In 1999 number 6 song was 'I Want It That Way' by Backstreet Boys (172.17.0.2)
tomgreen@Toms-MacBook-Air-3 NWENP3 % java MusicGuruClient 44.204.181.142 5000 0
Specified year out of range (1998 - 1999), using a random year instead: 1993
In 1993 number 4 song was 'Mr. Jones' by Counting Crows (172.17.0.2)
tomgreen@Toms-MacBook-Air-3 NWENP3 % java MusicGuruClient 44.204.181.142 5000 0
Specified year out of range (1998 - 1999), using a random year instead: 1994
In 1994 number 3 song was 'Sabotage' by Beastie Boys (172.17.0.2)
tomgreen@Toms-MacBook-Air-3 NWENP3 %

[ec2-user@ip-172-31-80-118 ~]$ ls
[ec2-user@ip-172-31-80-118 ~]$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
NAMES
5fac8c9e733 mgs "java MusicGuruServe..." 27 seconds ago Up 26 seconds 0.0.0.0:5000->5000/tcp, :::5000->5000/tcp
zealous_sanderson
[ec2-user@ip-172-31-80-118 ~]$
```

The next thing we do is create a hub.docker.com account. I was signed up with the username 'tomgreen585'. I made a repository under 'mgs' to push and pull my docker from the AWS instance. From here, I signed into my docker account on my instance by running the command 'docker login'. As we can see, it was successful.



A screenshot of a terminal window titled 'Downloads -- ec2-user@ip-172-31-80-118:~ -- ssh -i nwenpb.pem ec2-user@ec2-44-204-181-142.compute-1.amazonaws....'. The terminal shows the following sequence of commands:

```
--> b273004037cc
Step 2/6 : COPY *.java /usr/src/MGS/
--> Using cache
--> 0e0c481bcd2
Step 3/6 : WORKDIR /usr/src/MGS
--> Using cache
--> 45f06fd93cda
Step 4/6 : RUN javac MusicGuruServer.java
--> Using cache
--> e50ec0fa3e7
Step 5/6 : EXPOSE 5000
--> Running in 63958d049143
Removing intermediate container 63958d049143
--> dde2875022b7
Step 6/6 : CMD ["java", "MusicGuruServer", "5000"]
--> Running in 810327a2aeee
Removing intermediate container 810327a2aeee
--> 96f58986d7a3
Successfully built 96f58986d7a3
Successfully tagged mgs:latest
[[ec2-user@ip-172-31-80-118 ~]$ docker images --filter reference=mgs
REPOSITORY TAG IMAGE ID CREATED SIZE
mgs latest 96f58986d7a3 25 seconds ago 526MB
[[ec2-user@ip-172-31-80-118 ~]$ nano Dockerfile
[[ec2-user@ip-172-31-80-118 ~]$ docker run -t -i -p 5000:5000 mgs &
[1] 18778
[[ec2-user@ip-172-31-80-118 ~]$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
NAMES
5faca8c9e733 mgs "java MusicGuruServer..." 27 seconds ago Up 26 seconds 0.0.0.0:5000->5000/tcp, :::5000->5000/tcp
zealous_sanderson
[[ec2-user@ip-172-31-80-118 ~]$ ls
Dockerfile HealthCheck.py MusicGuruServer.class MusicGuruServer$ClientHandler.class MusicGuruServer.java run.sh
[[ec2-user@ip-172-31-80-118 ~]$ docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
[Username: tomgreen585
>Password:
WARNING! Your password will be stored unencrypted in /home/ec2-user/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store
Login Succeeded
```

They checked that the server was still running using the command 'docked ps.'

```
Downloads — ec2-user@ip-172-31-80-118:~ — ssh -i nwenpb.pem ec2-user@ec2-44-204-181-142.compute-1.amazonaws.com
```

```
--> Using cache
--> 45f06fd93cda
Step 4/6 : RUN javac MusicGuruServer.java
--> Using cache
--> e50ec08fa3e7
Step 5/6 : EXPOSE 5000
--> Running in 63958d049143
Removing intermediate container 63958d049143
--> dde2875022b7
Step 6/6 : CMD ["java", "MusicGuruServer", "5000"]
--> Running in 810327a2aeee
Removing intermediate container 810327a2aeee
--> 96f58986d7a3
Successfully built 96f58986d7a3
Successfully tagged mgs:latest
[[ec2-user@ip-172-31-80-118 ~]$ docker images --filter reference=mgs
REPOSITORY TAG IMAGE ID CREATED SIZE
mgs latest 96f58986d7a3 25 seconds ago 526MB
[[ec2-user@ip-172-31-80-118 ~]$ nano Dockerfile
[[ec2-user@ip-172-31-80-118 ~]$ docker run -t -i -p 5000:5000 mgs &
[1] 18778
[[ec2-user@ip-172-31-80-118 ~]$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
 NAMES
5facac8c9e733 mgs "java MusicGuruServer..." 27 seconds ago Up 26 seconds 0.0.0.0:5000->5000/tcp, :::5000->5000/tcp
zealous_sanderson
[[ec2-user@ip-172-31-80-118 ~]$ ls
Dockerfile HealthCheck.py MusicGuruServer.class MusicGuruServer$ClientHandler.class MusicGuruServer.java run.sh
[[ec2-user@ip-172-31-80-118 ~]$ docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
[Username: tomgreen585
>Password:
WARNING! Your password will be stored unencrypted in /home/ec2-user/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

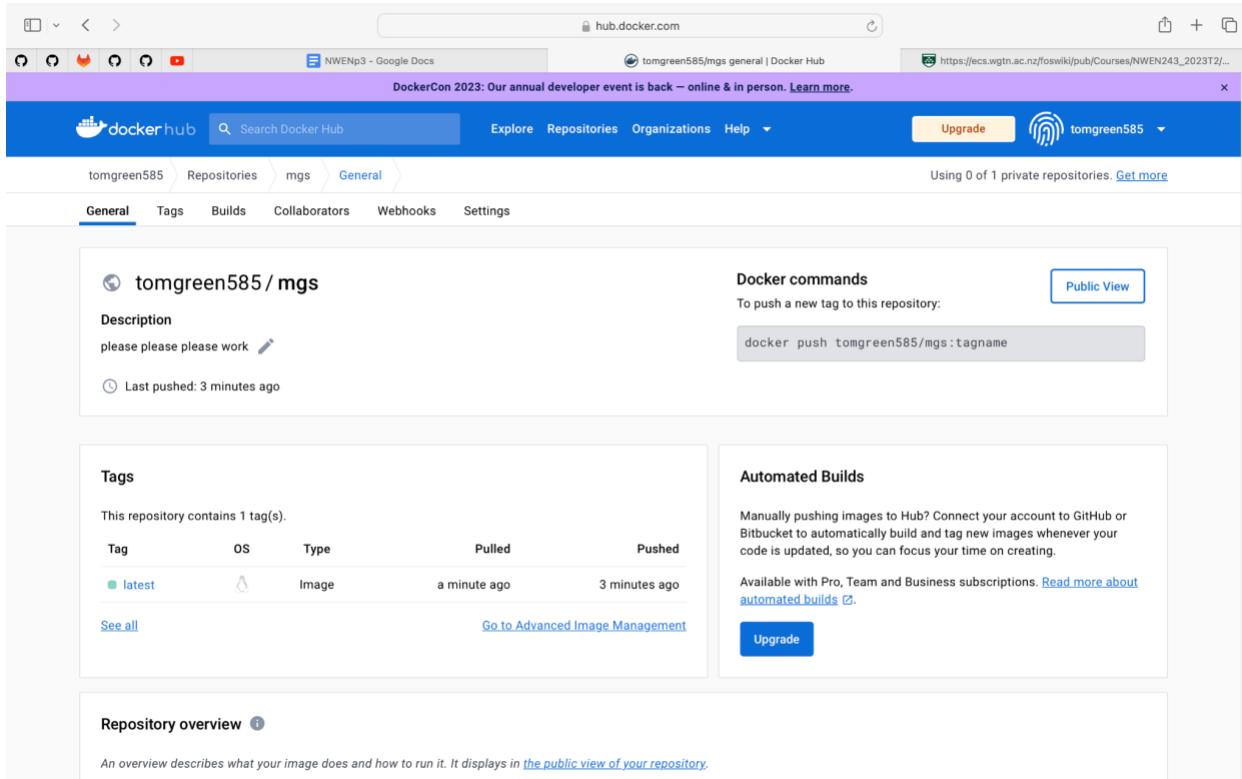
Login Succeeded
[[ec2-user@ip-172-31-80-118 ~]$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
 NAMES
5facac8c9e733 mgs "java MusicGuruServer..." About an hour ago Up About an hour 0.0.0.0:5000->5000/tcp, :::5000->5000/tcp
zealous_sanderson
[ec2-user@ip-172-31-80-118 ~]$
```

From here, I tagged my docker by using the command ' docker tag mgs tomgreen585/mgs:latest'. 'mgs' is my docker name, and I am tagging it with my username/reponame:latest, which gets the which as you can see above is the tag given to my docker image. From here, we push the idea to my repository in DockerHub using the command 'docker push tomgreen585/mgs:latest'. From here, we can see that it pushed my mgs docker successfully. 'docker' is the tag used to create a new tag for an existing image. 'mgs' is the name of the existing docker image. 'tomgreen585/mgs' is the repository I created and 'latest' is the tag of my mgs repository.



```
[ec2-user@ip-172-31-80-118 ~]$ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
mgs latest 96f58986d7a3 About an hour ago 526MB
<none> <none> e7def129d086 2 hours ago 526MB
openjdk 8 b273004037cc 14 months ago 526MB
[ec2-user@ip-172-31-80-118 ~]$ docker tag mgs USERNAME/REPOSITORY:latest
Error parsing reference: "USERNAME/REPOSITORY:latest" is not a valid repository/tag: invalid reference format: repository na
me must be lowercase
[ec2-user@ip-172-31-80-118 ~]$ docker tag mgs tomgreen585/mgs:latest
[ec2-user@ip-172-31-80-118 ~]$ docker push tomgreen585/mgs:latest
The push refers to repository [docker.io/tomgreen585/mgs]
a50e7d33139a: Pushed
3d75f720c2af: Pushed
6b5aaff44254: Mounted from library/openjdk
53a0b163e995: Mounted from library/openjdk
b626401ef603: Mounted from library/openjdk
9b55156abf26: Mounted from library/openjdk
293d5db30c9f: Mounted from library/openjdk
03127cdb479b: Mounted from library/openjdk
9c742cd6c7a5: Mounted from library/openjdk
latest: digest: sha256:ba738dc649d0a755f4156fac9f5e4a270d79cb689013e18c0405fee872ccf376 size: 2211
[ec2-user@ip-172-31-80-118 ~]$
```

Here, we can see that our docker was pushed successfully to my repository on DockerHub



The screenshot shows a browser window with the Docker Hub URL: hub.docker.com. The address bar also displays "NWErp3 - Google Docs". The Docker Hub header includes "DockerCon 2023: Our annual developer event is back – online & in person. Learn more." and a user profile for "tomgreen585". The main navigation menu has options: Explore, Repositories, Organizations, Help, Upgrade, and a user icon. Below the menu, it says "Using 0 of 1 private repositories. Get more". The current repository path is "tomgreen585/mgs General". The repository card for "tomgreen585/mgs" shows a description: "please please please work" with an edit icon. It indicates the last push was 3 minutes ago. A "Docker commands" section contains the command "docker push tomgreen585/mgs:tagname" and a "Public View" button. The "Tags" section lists one tag: "latest" (Image type, pulled a minute ago, pushed 3 minutes ago). The "Automated Builds" section explains how to automatically build images and mentions availability for Pro, Team, and Business subscriptions. The "Repository overview" section provides a brief description of what the image does and how to run it.

Docker commands

To push a new tag to this repository:

```
docker push tomgreen585/mgs:tagname
```

Tags

This repository contains 1 tag(s).

Tag	OS	Type	Pulled	Pushed
latest	Image		a minute ago	3 minutes ago

See all [Go to Advanced Image Management](#)

Automated Builds

Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.

Available with Pro, Team and Business subscriptions. [Read more about automated builds](#)

Repository overview

An overview describes what your image does and how to run it. It displays in [the public view of your repository](#).

Next, we made a new EC2 instance called 'NWEN_TESTFORTEST' (sorry about the other tests; I just wasn't setting it up correctly) to check that the pushed image in DockerHub was correct.

Launch AWS Academy Learner Instances | EC2 | us-east-1 AWS Load Balancing Assignments

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#Instances:

New EC2 Experience Tell us what you think

EC2 Dashboard EC2 Global View Events Instances Instances Instance Types Launch Templates Spot Requests Savings Plans Reserved Instances Dedicated Hosts Capacity Reservations Images AMIs AMI Catalog

Elastic Block Store Volumes Snapshots Lifecycle Manager Network & Security Security Groups Elastic IPs Placement Groups Key Pairs Network Interfaces Load Balancing Load Balancers CloudShell Feedback

Instances (1/18) Info Find instance by attribute or tag (case-sensitive)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...
-	i-0a1614218974605c	Stopped	t2.micro	Initializing	No alarms +	us-east-1b	-	-
NWEN_TAKE5	i-022d82b139562ac6	Running	t2.micro	2/2 checks passed	No alarms +	us-east-1c	ec2-44-204-181-142.co...	44.204.181.142
-	i-0b18c13784a62b91	Running	t2.micro	2/2 checks passed	No alarms +	us-east-1b	ec2-3-93-35-184.comp...	3.93.35.184
-	i-00bde1b79403c4064	Running	t2.micro	2/2 checks passed	No alarms +	us-east-1b	ec2-44-214-143-179.co...	44.214.143.179
-	i-0d43458e1ee96b612	Running	t2.micro	2/2 checks passed	No alarms +	us-east-1a	ec2-54-91-65-96.comp...	54.91.65.96
NWEN_P3_TestDocker	i-05821b31ddf466d59	Terminated	t2.micro	-	No alarms +	us-east-1c	-	-
NWEN_TEST	i-0dedca5c85a24b8a	Terminated	t2.micro	-	No alarms +	us-east-1c	-	-
NWEN_OFFICIALTEST	i-048319ebf178a8d76	Terminated	t2.micro	-	No alarms +	us-east-1c	-	-
MGS_TEST	i-0ed3aae281021382d	Terminated	t2.micro	-	No alarms +	us-east-1c	-	-
NWEN_TESTFORTEST	i-006800bf254f593f6	Running	t2.micro	-	No alarms +	us-east-1c	ec2-34-239-123-149.co...	34.239.123.149

Instance: i-006800bf254f593f6 (NWEN_TESTFORTEST)

Details Security Networking Storage Status checks Monitoring Tags

Instance summary Info

Instance ID	Public IPv4 address	Private IPv4 addresses
i-006800bf254f593f6 (NWEN_TESTFORTEST)	34.239.123.149 [open address]	172.31.85.17
IPv6 address	Instance state	Public IPv4 DNS
-	Running	ec2-34-239-123-149.compute-1.amazonaws.com [open address]
Hostname type	Private IP DNS name (IPv4 only)	Elastic IP addresses
IP name: ip-172-31-83-17.ec2.internal	ip-172-31-83-17.ec2.internal	-
Answer private resource DNS name	Instance type	AWS Compute Optimizer finding
IPv4 (A)	t2.micro	Opt-in to AWS Compute Optimizer for recommendations.
Auto-assigned IP address	VPC ID	Learn more
34.239.123.149 [Public IP]	vpc-05900b3fd4317157	

Use commands from the handout 2a.b.c.d

```
Terminal Shell Edit View Window Help
Downloads — ec2-user@ip-172-31-83-17:~ — ssh -i nwenpb.pem ec2-user@ec2-34-239-123-149.compute-1.amazonaws.com:22
(1/5): libcgroup-0.41-21.amzn2.x86_64.rpm | 66 kB 00:00:00
(2/5): pigz-2.3.4-1.amzn2.0.1.x86_64.rpm | 81 kB 00:00:00
(3/5): containerd-1.6.19-1.amzn2.0.3.x86_64.rpm | 28 MB 00:00:00
(4/5): docker-20.10.23-1.amzn2.0.1.x86_64.rpm | 41 MB 00:00:00
(5/5): runc-1.1.7-3.amzn2.x86_64.rpm | 3.0 MB 00:00:00
Total 76 MB/s | 73 MB 00:00:00
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : runc-1.1.7-3.amzn2.x86_64 1/5
  Installing : containerd-1.6.19-1.amzn2.0.3.x86_64 2/5
  Installing : libcgroup-0.41-21.amzn2.x86_64 3/5
  Installing : pigz-2.3.4-1.amzn2.0.1.x86_64 4/5
  Installing : docker-20.10.23-1.amzn2.0.1.x86_64 5/5
  Verifying  : containerd-1.6.19-1.amzn2.0.3.x86_64 1/5
  Verifying  : runc-1.1.7-3.amzn2.x86_64 2/5
  Verifying  : docker-20.10.23-1.amzn2.0.1.x86_64 3/5
  Verifying  : pigz-2.3.4-1.amzn2.0.1.x86_64 4/5
  Verifying  : libcgroup-0.41-21.amzn2.x86_64 5/5
Installed:
  docker.x86_64 0:20.10.23-1.amzn2.0.1

Dependency Installed:
  containerd.x86_64 0:1.6.19-1.amzn2.0.3    libcgroup.x86_64 0:0.41-21.amzn2    pigz.x86_64 0:2.3.4-1.amzn2.0.1
  runc.x86_64 0:1.1.7-3.amzn2

Complete!
[[ec2-user@ip-172-31-83-17 ~]$ sudo service docker start
Redirecting to /bin/systemctl start docker.service
[[ec2-user@ip-172-31-83-17 ~]$ sudo usermod -a -G docker ec2-user
[[ec2-user@ip-172-31-83-17 ~]$
```

We exit the ssh and return to the instance to refresh the permissions we set. We then run the command 'docker info' to see a status dump and to check that this was successful.



The screenshot shows a macOS Terminal window with the title bar "Terminal" and various menu options like Shell, Edit, View, Window, Help. The window content displays the output of the "docker info" command. The output provides detailed information about the Docker environment, including Cgroup Driver (cgroupfs), Cgroup Version (1), Plugins (awslogs, fluentd, gelf, journald, json-file, local, logentries, splunk, syslog), Swarm (inactive), Runtimes (io.containerd.runc.v2, io.containerd.runtime.v1.linux, runc), Default Runtime (runc), Init Binary (docker-init), containerd version (1e1ea6e986c6c86565bc33d52e34b81b3e2bc71f), runc version (f19387a6bec4944c770f7668ab51c4348d9c2f38), init version (de40ad0), Security Options (seccomp), Profile (default), Kernel Version (5.10.192-183.736.amzn2.x86_64), Operating System (Amazon Linux 2), OSType (linux), Architecture (x86_64), CPUs (1), Total Memory (952.7MiB), Name (ip-172-31-83-17.ec2.internal), ID (4ACE:F6TV:CQQ5:2LB4:3ERK:GKTF:DFA7:SRFA:SR2R:423U:XOZH:E4FU), Docker Root Dir (/var/lib/docker), Debug Mode (false), Registry (https://index.docker.io/v1/), Labels, Experimental (false), Insecure Registries (127.0.0.0/8), and Live Restore Enabled (false). The prompt at the bottom of the terminal is "[ec2-user@ip-172-31-83-17 ~]\$".

```
Cgroup Driver: cgroupfs
Cgroup Version: 1
Plugins:
  Volume: local
  Network: bridge host ipvlan macvlan null overlay
  Log: awslogs fluentd gelf journald json-file local logentries splunk syslog
Swarm: inactive
Runtimes: io.containerd.runc.v2 io.containerd.runtime.v1.linux runc
Default Runtime: runc
Init Binary: docker-init
containerd version: 1e1ea6e986c6c86565bc33d52e34b81b3e2bc71f
runc version: f19387a6bec4944c770f7668ab51c4348d9c2f38
init version: de40ad0
Security Options:
  seccomp
    Profile: default
Kernel Version: 5.10.192-183.736.amzn2.x86_64
Operating System: Amazon Linux 2
OSType: linux
Architecture: x86_64
CPUs: 1
Total Memory: 952.7MiB
Name: ip-172-31-83-17.ec2.internal
ID: 4ACE:F6TV:CQQ5:2LB4:3ERK:GKTF:DFA7:SRFA:SR2R:423U:XOZH:E4FU
Docker Root Dir: /var/lib/docker
Debug Mode: false
Registry: https://index.docker.io/v1/
Labels:
Experimental: false
Insecure Registries:
  127.0.0.0/8
Live Restore Enabled: false
[ec2-user@ip-172-31-83-17 ~]$
```

Next, we sign into our DockerHub account by running the command docker login' so we can pull our docker image from our repository



A screenshot of a macOS Terminal window titled "Terminal". The window shows the command "docker login" being run in a terminal session. The session starts with the prompt "[ec2-user@ip-172-31-83-17 ~]\$ docker login". The user is prompted to enter their Docker ID, with the message "Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one." The user enters their username "tomgreen585" and password. A warning message appears: "WARNING! Your password will be stored unencrypted in /home/ec2-user/.docker/config.json. Configure a credential helper to remove this warning. See https://docs.docker.com/engine/reference/commandline/login/#credentials-store". Finally, the message "Login Succeeded" is displayed.

```
[ec2-user@ip-172-31-83-17 ~]$ docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
Username: tomgreen585
Password:
WARNING! Your password will be stored unencrypted in /home/ec2-user/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
[ec2-user@ip-172-31-83-17 ~]$
```

Here, we run 'ls' to show that our repository has nothing in it

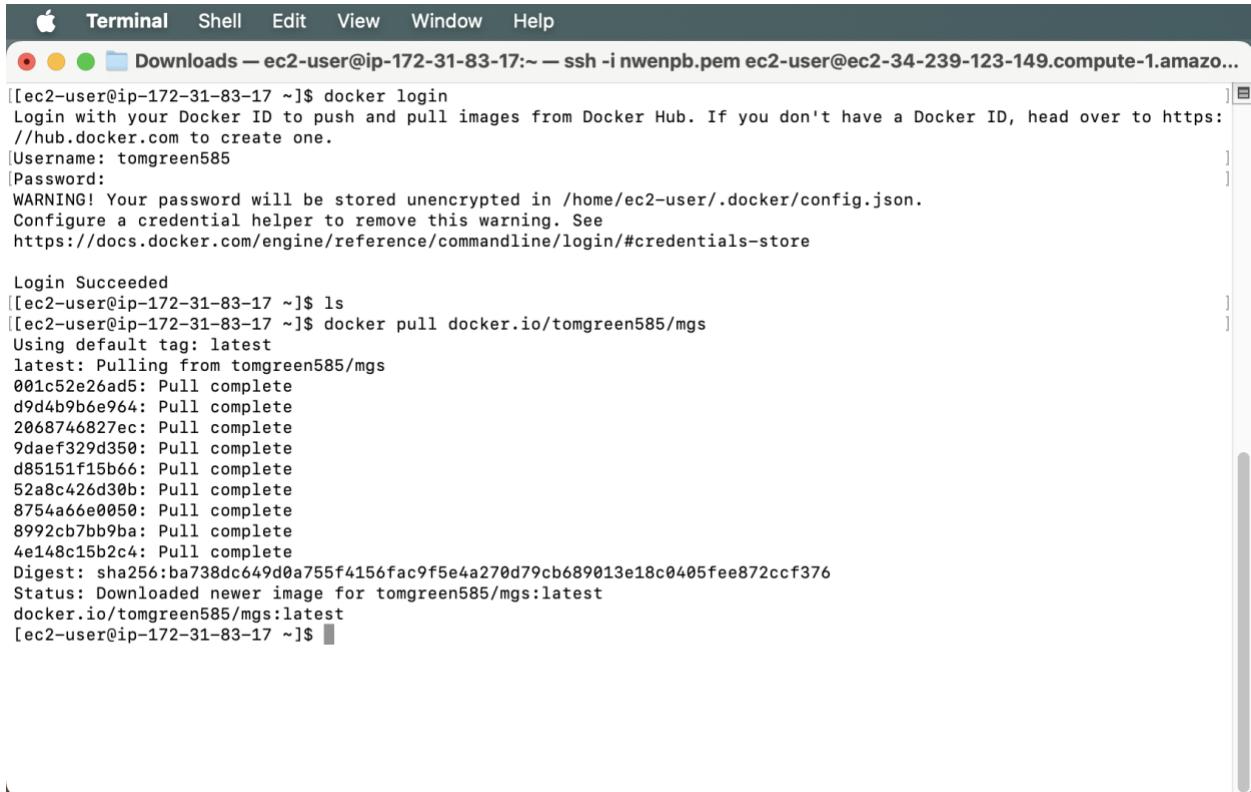


A screenshot of a Mac OS X Terminal window. The title bar shows "Terminal" and the current location as "Downloads — ec2-user@ip-172-31-83-17:~ — ssh -i nwenpb.pem ec2-user@ec2-34-239-123-149.compute-1.amazonaws.com". The main pane contains the following text:

```
[ec2-user@ip-172-31-83-17 ~]$ docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
Username: tomgreen585
Password:
WARNING! Your password will be stored unencrypted in /home/ec2-user/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
[ec2-user@ip-172-31-83-17 ~]$ ls
[ec2-user@ip-172-31-83-17 ~]$
```

Next, we run the command 'docker pull docker.io/tomgreen585/mgs', which pulls the repository docker I pushed earlier using my username and repository name. As we can see, it was successful, and downloaded a newer image, 'latest,' which was the tag we gave our mgs docker. 'docker pull' is used to fetch a docker image from a docker repository. 'docker.io/tomgreen585/mgs:latest' is the complete name and tag of the docker image file that we are pulling it specifies that we want to obtain the 'latest' docker from the repository tomgreen585/mgs. This allows it to be obtained.

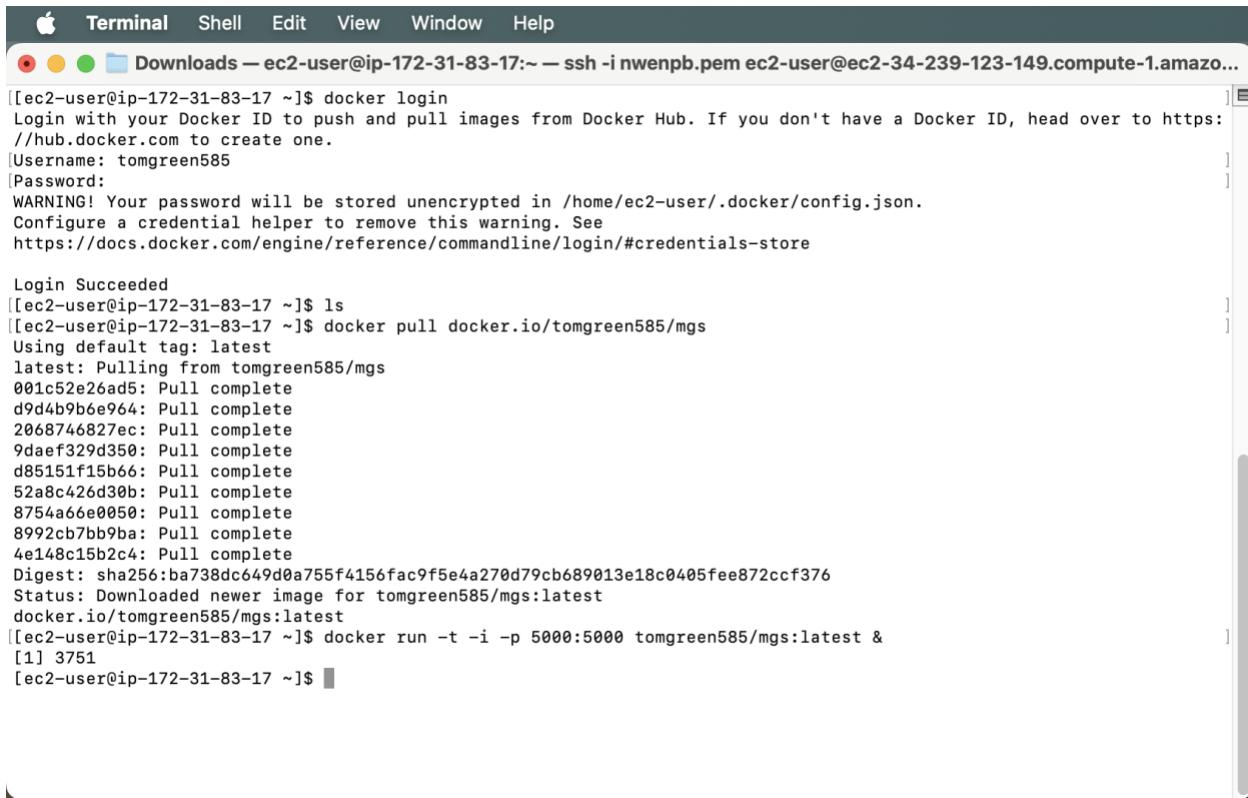


The screenshot shows a Mac OS X Terminal window with the title bar "Terminal" and various menu options like Shell, Edit, View, Window, Help. The window content displays a session on an EC2 instance:

```
[ec2-user@ip-172-31-83-17 ~]$ docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
Username: tomgreen585
Password:
WARNING! Your password will be stored unencrypted in /home/ec2-user/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
[[ec2-user@ip-172-31-83-17 ~]$ ls
[[ec2-user@ip-172-31-83-17 ~]$ docker pull docker.io/tomgreen585/mgs
Using default tag: latest
latest: Pulling from tomgreen585/mgs
001c52e26ad5: Pull complete
d9d4b9b6e964: Pull complete
2068746827ec: Pull complete
9daef329d350: Pull complete
d85151f15b66: Pull complete
52a8c426d30b: Pull complete
8785a66e0050: Pull complete
8992cb7bb9ba: Pull complete
4e148c15b2c4: Pull complete
Digest: sha256:ba738dc649d0a755f4156fac9f5e4a270d79cb689013e18c0405fee872ccf376
Status: Downloaded newer image for tomgreen585/mgs:latest
docker.io/tomgreen585/mgs:latest
[ec2-user@ip-172-31-83-17 ~]$ ]]
```

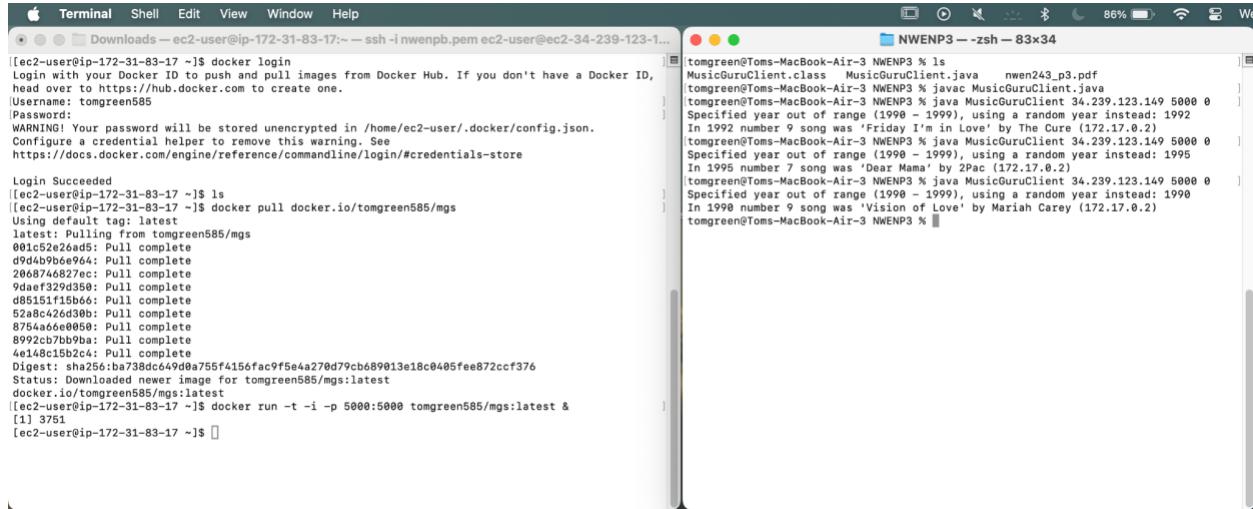
We then run the docker by setting it to a port map using the command 'docker run -t -i -p 5000:5000 tomgreen585/mgs:latest &'. I'm mapping port 5000 on my EC2 instance to port 5000 within the Docker container. This mapping is essential because it enables me to access services running inside the container via port 5000 on my EC2 instance. I specify the docker image I want to run 'tomgreen585/mgs:latest &'. The '&' allows the docker container to run in the background.



```
Terminal Shell Edit View Window Help
Downloads — ec2-user@ip-172-31-83-17:~ — ssh -i nwenzpb.pem ec2-user@ec2-34-239-123-149.compute-1.amazonaws.com
[ec2-user@ip-172-31-83-17 ~]$ docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
Username: tomgreen585
Password:
WARNING! Your password will be stored unencrypted in /home/ec2-user/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
[[ec2-user@ip-172-31-83-17 ~]$ ls
[[ec2-user@ip-172-31-83-17 ~]$ docker pull docker.io/tomgreen585/mgs
Using default tag: latest
latest: Pulling from tomgreen585/mgs
001c52e26ad5: Pull complete
d9d4b9b6e964: Pull complete
2068746827ec: Pull complete
9daef329d350: Pull complete
d85151f15b66: Pull complete
52a8c426d30b: Pull complete
8754a66e0050: Pull complete
8992cb7bb9ba: Pull complete
4e148c15b2c4: Pull complete
Digest: sha256:ba738dc649d0a755f4156fac9f5e4a270d79cb689013e18c0405fee872ccf376
Status: Downloaded newer image for tomgreen585/mgs:latest
docker.io/tomgreen585/mgs:latest
[[ec2-user@ip-172-31-83-17 ~]$ docker run -t -i -p 5000:5000 tomgreen585/mgs:latest &
[1] 3751
[ec2-user@ip-172-31-83-17 ~]$
```

Now we run our MusicGuruClient connecting to our new instance running the pulled docker using the instance IP, the port number we set, the docker image, and the year. As we can see, it was successful.



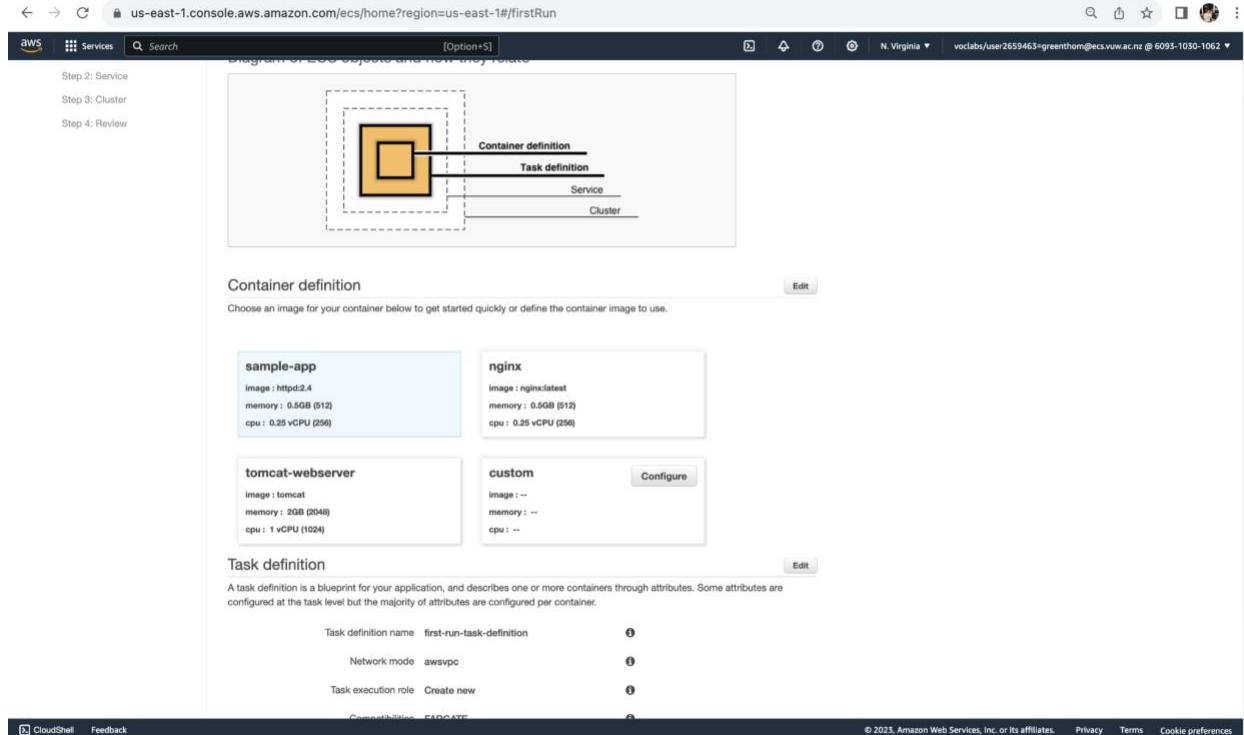
```

Terminal Shell Edit View Window Help
Downloads - ec2-user@ip-172-31-83-17:~ ssh -i nwenpb.pem ec2-user@ec2-34-239-123-1...
[ec2-user@ip-172-31-83-17 ~]$ docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
Username: tomgreen585
>Password:
WARNING! Your password will be stored unencrypted in /home/ec2-user/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
[ec2-user@ip-172-31-83-17 ~]$ ls
[ec2-user@ip-172-31-83-17 ~]$ docker pull docker.io/tomgreen585/mgs
Using default tag: latest
latest: Pulling from tomgreen585/mgs
001c52e24ad5: Pull complete
d9d4b9b64944: Pull complete
2068746827ec: Pull complete
9daeaf329d35b0: Pull complete
d85151f15b66: Pull complete
52a8c426d30b: Pull complete
8754a66e0050: Pull complete
8992cb7b9b9a: Pull complete
4e148c15b2c4: Pull complete
Digest: sha256:ba738dc6490a755f4156fac9f5e4a270d79cb689013e18c0405fee872ccf376
Status: Downloaded newer image for tomgreen585/mgs:latest
docker.io/tomgreen585/mgs:latest
[ec2-user@ip-172-31-83-17 ~]$ docker run -t -i -p 5000:5000 tomgreen585/mgs:latest &
[1] 3751
[ec2-user@ip-172-31-83-17 ~]$ 

```

Next, we make an Amazon Elastic Container Service (ECS) using Fargate



us-east-1.console.aws.amazon.com/ecs/home?region=us-east-1#/firstRun

aws Services Search [Option+I]

Step 2: Service
Step 3: Cluster
Step 4: Review

Container definition

Choose an image for your container below to get started quickly or define the container image to use.

sample-app	nginx
image : httpd:2.4	image : nginx:latest
memory : 0.5GB (512)	memory : 0.5GB (512)
cpu : 0.25 vCPU (256)	cpu : 0.25 vCPU (256)

tomcat-webserver	custom
image : tomcat	image : --
memory : 2GB (2048)	memory : --
cpu : 1 vCPU (1024)	cpu : --

Task definition

A task definition is a blueprint for your application, and describes one or more containers through attributes. Some attributes are configured at the task level but the majority of attributes are configured per container.

Task definition name: first-run-task-definition	Configure
Network mode: awsvpc	Configure
Task execution role: Create new	Configure
Compatibility: FARGATE	Configure

CloudShell Feedback © 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

We then edit the container, setting it a name 'NWEN_P2_Cluster', our docker.io repository 'docker.io/tomgreen585/mgs:latest'. Ensured the protocol is TCP and the port set to '5000'.

The screenshot shows the 'Edit container' dialog box from the AWS CloudWatch Metrics service. The 'Container definition' section is visible on the left, showing two sample configurations: 'sample-app' and 'tomcat-webserver'. The 'Container' tab is selected, displaying the configuration for 'NWEN_P2_Cluster'. The 'Container name' field is set to 'NWEN_P2_Cluster'. The 'Image' field is set to 'docker.io/tomgreen585/mgs:latest'. Under 'Memory Limits (MiB)', the 'Soft limit' is set to 128 and the 'Hard limit' is set to 128. In the 'Port mappings' section, a single mapping is defined: 'Container port' 5000 mapped to 'Protocol' TCP. A note below states: 'Host port mappings are not valid when the network mode for a task definition is host or awsvpc. To specify different host and container port mappings, choose the Bridge network mode.' At the bottom right, there are 'Cancel' and 'Update' buttons, with a note '* Required'.

Next, we edit the task definition. We keep the definition name. We changed the task execution role to LabRole.

We set the cluster name to NWENP2clu

We check the ports and roles are correct and then create it as we can see it completed successfully.

As we can see, it pulled the docker from the repository correctly and is running

Task	Task Definition	Last status	Desired status	Group	Launch type	Platform version
d4e78e614f594459b7915e24...	first-run-task-definition:3	RUNNING	RUNNING	service:NWEN_P2_Cluster-s...	FARGATE	1.4.0

Next, we need to find the IP address for the serverless service. As we can see here for this task, our public IP is 3.88.46.6

The screenshot shows the AWS ECS console interface. On the left, there's a sidebar with navigation links like 'Clusters', 'Task Definitions', 'Account Settings', etc. The main area displays a task named 'd4e78e614f594459b76f5e241b355126' in the 'NWENP2clu' cluster. The 'Details' tab is selected, showing information such as Cluster (NWENP2clu), Launch type (FARGATE), Platform version (1.4.0), Task definition (first-run-task-definition:3), Group (service:NWEN_P2_Cluster-service), Task role (None), Last status (RUNNING), Desired status (RUNNING), Created at (2023-10-04 15:23:26 +1300), and Started at (2023-10-04 15:23:58 +1300). Below this, the 'Network' section shows network mode (awsvpc), ENI ID (eni-0549c79d5acdb97d), Subnet ID (subnet-08cba729af0633afc), Private IP (10.0.0.123), Public IP (3.88.46.6), and Mac address (0e:7d:67:cd:e2:85). The 'Containers' section lists a single container named 'NWEN_P2...' with a status of RUNNING, using the docker.io/tomgreen585/mgs:latest image, and SHA256 digest sha256:ba738dc649d0a755f4156f... The table also includes columns for CPU Units, Hard/Soft memory limits, Essential, and Resource ID.

From here, we run the MusicGuruClient on our local machine, connecting it with the IP we found. As you can see, this was successful.

The terminal window title is 'NWENP3 — zsh — 97x24'. The command entered is 'java MusicGuruClient 3.88.46.6 5000 0'. The output shows the client connecting to the service and printing out song information for different years:

```
[tomgreen@Toms-MacBook-Air-3 NWENP3 % ls
MusicGuruClient.class  MusicGuruClient.java      nwen243_p3.pdf
[tomgreen@Toms-MacBook-Air-3 NWENP3 % javac MusicGuruClient.java
[tomgreen@Toms-MacBook-Air-3 NWENP3 % java MusicGuruClient 3.88.46.6 5000 0
Specified year out of range (1990 - 1999), using a random year instead: 1997
In 1997 number 7 song was 'Good Riddance (Time of Your Life)' by Green Day (10.0.0.123)
[tomgreen@Toms-MacBook-Air-3 NWENP3 % java MusicGuruClient 3.88.46.6 5000 0
Specified year out of range (1990 - 1999), using a random year instead: 1990
In 1990 number 4 song was 'Vogue' by Madonna (10.0.0.123)
[tomgreen@Toms-MacBook-Air-3 NWENP3 % java MusicGuruClient 3.88.46.6 5000 0
Specified year out of range (1990 - 1999), using a random year instead: 1998
In 1998 number 6 song was 'Hard Knock Life' by Jay-Z (10.0.0.123)
tomgreen@Toms-MacBook-Air-3 NWENP3 %
```

QUESTIONS for B

The classic Amazon ECS console is reaching the end of life and will no longer be available after December 4, 2023. We recommend that you switch immediately to the new Amazon ECS console for a better experience. To learn more about the new Amazon ECS console experience, visit the [documentation page](#), and you can also review and follow the new Amazon ECS console roadmap on [GitHub](#).

Service : NWENQUESTION

Cluster	NWENP2clu	Desired count	3
Status	ACTIVE	Pending count	0
Task definition	first-run-task-definition:4	Running count	3
Service type	REPLICAS		
Launch type	FARGATE		
Service role	AWSServiceRoleForECS		
Created By	arn:aws:iam::609310301062:role/voclabs		

Tasks

Task status	Running	Stopped
3162d1aee71f496783ada1e32c2cd3f4	RUNNING	RUNNING
8f150bc82e284ef2a23d2fb42bf679e3	RUNNING	RUNNING
c1b19ba4927c6748bb79abcc0f0ee15	RUNNING	RUNNING

Task : 3162d1aee71f496783ada1e32c2cd3f4

Details	Tags	Logs
Cluster	NWENP2clu	
Launch type	FARGATE	
Platform version	1.4.0	
Task definition	first-run-task-definition:4	
Group	service:NWENQUESTION	
Task role	None	
Last status	RUNNING	
Desired status	RUNNING	
Created at	2023-10-04 15:40:13 +1300	
Started at	2023-10-04 15:40:50 +1300	

Network

Network mode	awsvpc
ENI Id	eni-038e1e94929b4bcd8
Subnet Id	subnet-056cae7a688ec115
Private IP	10.0.1.216
Public IP	3.236.186.228
Mac address	02:35:43:15:67:d7

Containers

Name	Container Runtime I...	Status	Image	Image Digest	CPU Units	Hard/Soft mem... m	Essential	Resource ID
NWENQU...	3162d1aee71f496783...	RUNNING	docker.io/tomgreen585/mgs:latest	sha256:ba738dc649d0a755f4156f...	--	-/-	true	2c05029d-828e...

← → C us-east-1.console.aws.amazon.com/ecs/home?region=us-east-1#/clusters/NWENP2clu/tasks/8f150bc82e284ef2a23d2fb42bf679e3/details

AWS Services Search [Option+S] N. Virginia v vocabs/user2659463=greenhom@ecs.vvu.ac.nz @ 6093-1030-1062 ▾

Amazon ECS Clusters Task Definitions Account Settings Amazon EKS Clusters Amazon ECR Repositories AWS Marketplace Discover software Subscriptions

Clusters > NWENP2clu > Task: 8f150bc82e284ef2a23d2fb42bf679e3

Task : 8f150bc82e284ef2a23d2fb42bf679e3 Run more like this Stop

Details Tags Logs

Cluster NWENP2clu Launch type FARGATE Platform version 1.4.0 Task definition first-run-task-definition:4 Group service:NWENQUESTION Task role None Last status RUNNING Desired status RUNNING Created at 2023-10-04 15:40:13 +1300 Started at 2023-10-04 15:40:50 +1300

Network

Network mode awsvpc ENI Id eni-0e4e097212933c7b2 Subnet Id subnet-056caef7688ec115 Private IP 10.0.1.212 Public IP 3.239.164.197 Mac address 02:fb:76:9b:c4:8d

Containers

Last updated on October 4, 2023 3:42:14 PM (0m ago) CloudShell Feedback © 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Name	Container Runtime I...	Status	Image	Image Digest	CPU Units	Hard/Soft mem...	Essential	Resource ID
▶ NWENQU...	8f150bc82e284ef2a2...	RUNNING	docker.io/tomgreen585/mgs:latest	sha256:ba738dc649d0a755f4156f...	--	--/-	true	132f31c2-74eb-...

← → C us-east-1.console.aws.amazon.com/ecs/home?region=us-east-1#/clusters/NWENP2clu/tasks/c1b19ba4927c4c748bb79ab9cde8ee15/details

AWS Services Search [Option+S] N. Virginia v vocabs/user2659463=greenhom@ecs.vvu.ac.nz @ 6093-1030-1062 ▾

Tell us what you think New ECS Experience Amazon ECS Clusters Task Definitions Account Settings Amazon EKS Clusters Amazon ECR Repositories AWS Marketplace Discover software Subscriptions

Clusters > NWENP2clu > Task: c1b19ba4927c4c748bb79ab9cde8ee15

Task : c1b19ba4927c4c748bb79ab9cde8ee15 Run more like this Stop

Details Tags Logs

Cluster NWENP2clu Launch type FARGATE Platform version 1.4.0 Task definition first-run-task-definition:4 Group service:NWENQUESTION Task role None Last status RUNNING Desired status RUNNING Created at 2023-10-04 15:40:13 +1300 Started at 2023-10-04 15:40:37 +1300

Network

Network mode awsvpc ENI Id eni-0fd94b831db185db Subnet Id subnet-056caef7688ec115 Private IP 10.0.1.178 Public IP 3.210.198.2 Mac address 02:72:54:8f:0e:cf

Containers

Last updated on October 4, 2023 3:42:28 PM (0m ago) CloudShell Feedback © 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Name	Container Runtime I...	Status	Image	Image Digest	CPU Units	Hard/Soft mem...	Essential	Resource ID
▶ NWENQU...	c1b19ba4927c4c748...	RUNNING	docker.io/tomgreen585/mgs:latest	sha256:ba738dc649d0a755f4156f...	--	--/-	true	f75acc99-101c-...

Question 1:

IP public and private as well as MAC addresses are unique across. This is to distinguish between each of them.

Question 2:

Load balancer, better communication between client and server it would make it horizontally scalable and handle more or less users. Therefore, as users connect and disconnect, new replicas could be created. This would satisfy the demand and not waste resources. It will evenly distribute client requests across multiple replicas, preventing any single replica from getting overloaded. This helps with sharing the workload and ensuring redundancy. Health checks monitor the status of each replica. If a replica isn't healthy, its automatically taken out of the rotation, so clients are directed only to the healthy ones. Scaling policies automatically adjust the number of replicas based on how much work is coming in. This ensures that clients always have the right number of replicas, not matter how busy things get.