Udemy Project 7

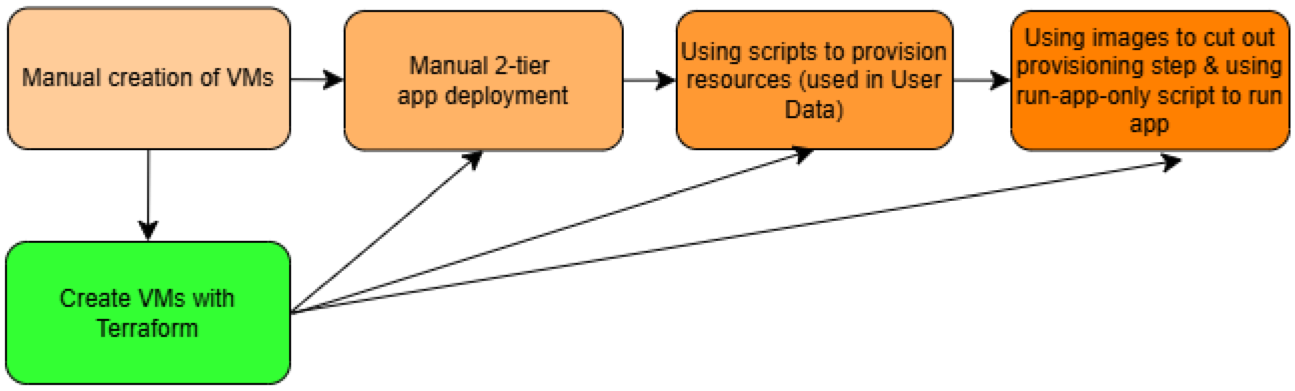
GitHub repo

Available here

Introduction to the problem

- We have a test app that uses Node JS v20
- This has a front page and a */posts* page that we need to get up and running
- We will be using a 2-tier app such that the app is on one VM/EC2 instance, and MongoDB (our database server) is on the other
- The **goal of this project** is to automate the deployment of this 2-tier app using a **script file** for each VM/EC2 that provisions it with all the required dependencies without needing user input at any stage or manual configuration
  - this needs to be idempotent i.e. work via the **User Data** option when creating the VMs/EC2s, and then it needs to run successfully as a script file again on these VMs/EC2s
- To further automate this setup, I also then want to **create images** of the two VMs/EC2s that have been provisioned by the scripts
  - After this stage, I will use one smaller ***run-app-only.sh* script** on the app VM/EC2 created from this image to get the app running -- again, this needs to be idempotent, so it should work when run as **User Data** when creating an app VM/EC2 from the previous image, as well as successfully run again multiple times on this VM/EC2

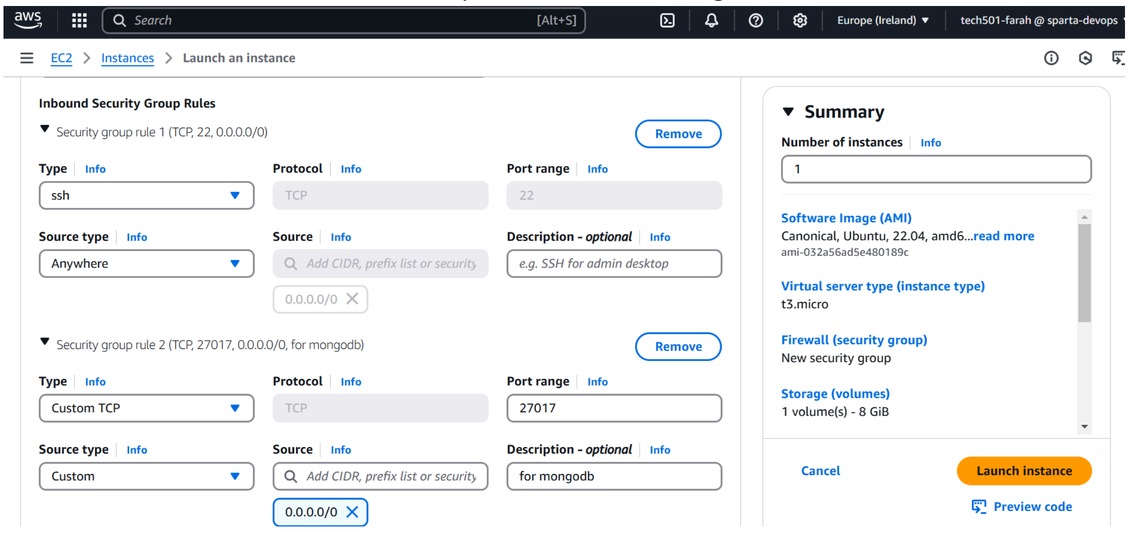
Diagram showing my automation workflow


Manual creation of VMs
Manual 2-tier app deployment
Using scripts to provision resources (used in User Data)
Using images to cut out provisioning step & using run-app-only script to run app
Create VMs with Terraform

Detailed steps

Initial VM creation

- I used the following settings on AWS:
  - **For DB VM**:
    - **VPC**: default, and default subnet
    - **AMI**: Ubuntu 22.04 LTS (HVM), SSD volume type
  - **Instance type**: t3.micro
  - **Key pair**: my AWS key
  - **Network**: default VPC
  - **Security rules**:
    - allowed SSH (this is done **by default**)
    - **added a new rule** to allow traffic on port 27017 (for mongoDB) from all sources



Inbound Security Group Rules

Security group rule 1 (TCP, 22, 0.0.0.0/0)

Type	Protocol	Port range
ssh	TCP	22

Source type	Source	Description - *optional*
Anywhere	Add CIDR, prefix list or security; 0.0.0.0/0	e.g. SSH for admin desktop

Security group rule 2 (TCP, 27017, 0.0.0.0/0, for mongodb)

Type	Protocol	Port range
Custom TCP	TCP	27017

Source type	Source	Description - *optional*
Custom	Add CIDR, prefix list or security; 0.0.0.0/0	for mongodb

Summary

Number of instances: 1

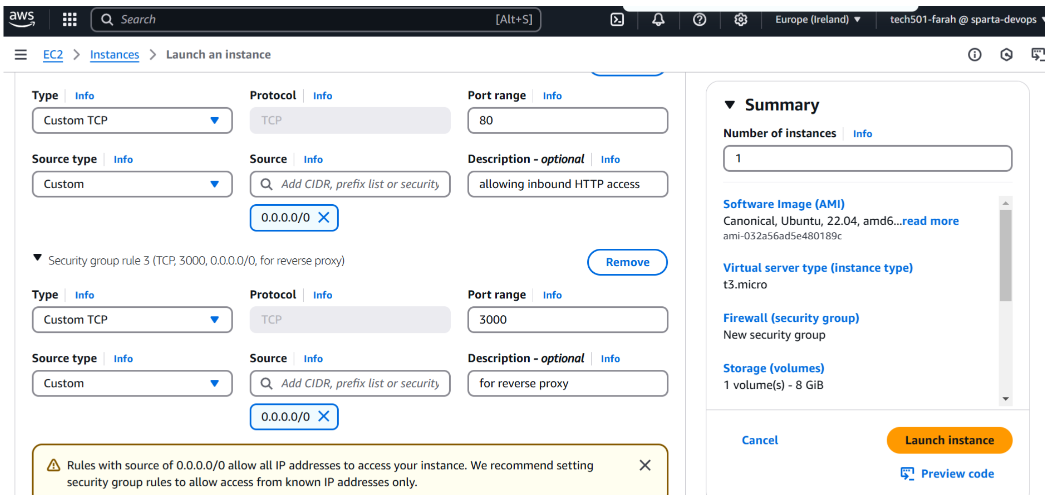
Software Image (AMI)
Canonical, Ubuntu, 22.04, amd6...read more
ami-032a56ad5e480189c

Virtual server type (instance type)
t3.micro

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

  - **For app VM**:
    - **VPC**: default, and default subnet
    - **AMI**: Ubuntu 22.04 LTS (HVM), SSD volume type
    - **Instance type**: t3.micro
    - **Key pair**: my AWS key
    - **Network**: default VPC
    - **Security rules**:
    - allowed SSH (this is done **by default**)
      - **added new rules** to allow inbound HTTP access (i.e. port 80) and inbound access on port 3000 (for reverse proxy)


aws
Search [Alt+S]
Europe (Ireland)
tech501-farah @ sparta-devops
EC2 > Instances > Launch an instance

Type: Custom TCP | Protocol: TCP | Port range: 80
Source type: Custom | Source: Add CIDR, prefix list or security | Description - optional: allowing inbound HTTP access
0.0.0.0/0

Security group rule 3 (TCP, 3000, 0.0.0.0/0, for reverse proxy) — Remove

Type: Custom TCP | Protocol: TCP | Port range: 3000
Source type: Custom | Source: Add CIDR, prefix list or security | Description - optional: for reverse proxy
0.0.0.0/0

Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Summary
Number of instances: 1
Software Image (AMI): Canonical, Ubuntu, 22.04, amd6...read more
ami-032a56ad5e480189c
Virtual server type (instance type): t3.micro
Firewall (security group): New security group
Storage (volumes): 1 volume(s) - 8 GiB

Cancel | Launch instance | Preview code

1a. Developing and implementing a Bash script to provision the database VM

1. I created a script file to provision the database EC2 via the **User Data field** on AWS's GUI that did not need user input

- To avoid the need for any manual configuration, I used the Bash `sed` command to change MongoDB's BindIP setting to 0.0.0.0:

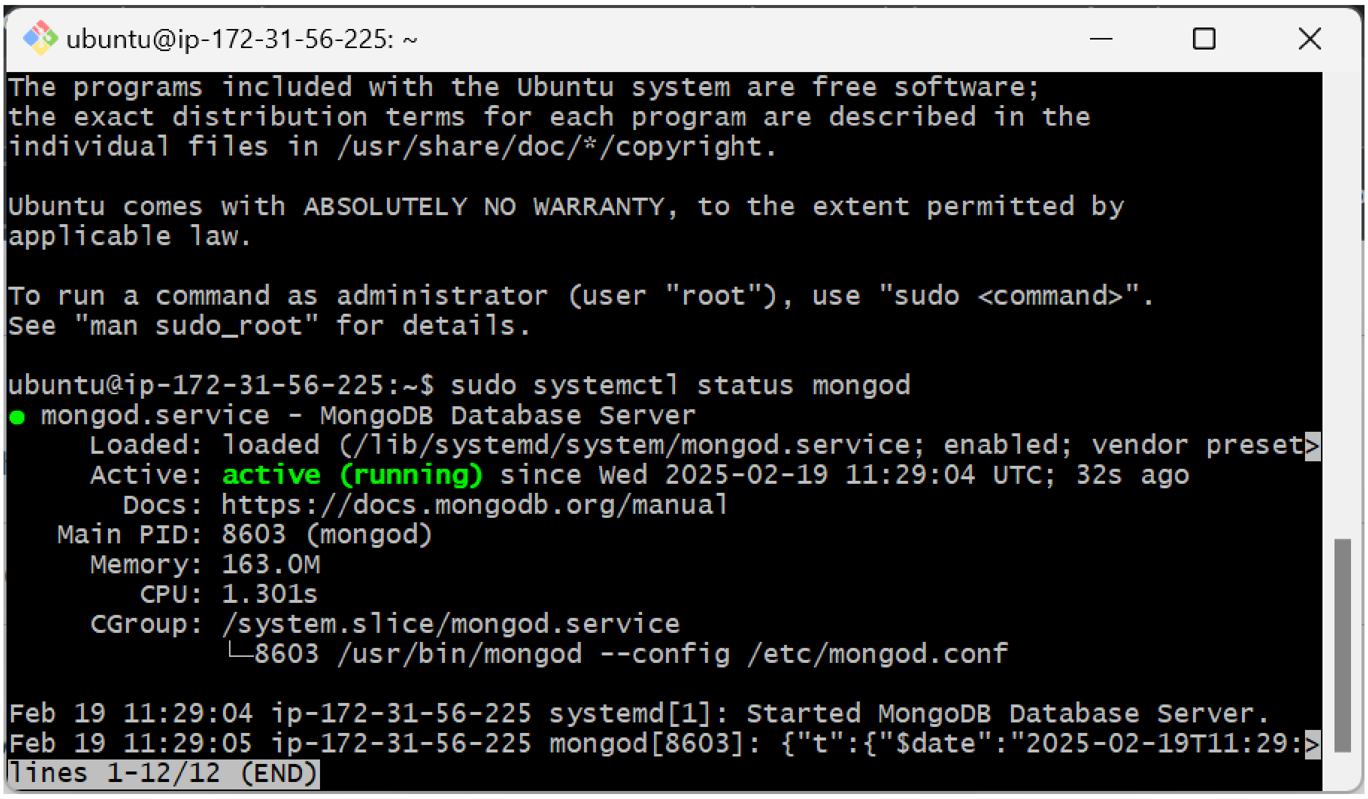
```
sudo sed -i 's/bindIp: 127.0.0.1/bindIp: 0.0.0.0/' /etc/mongod.conf
```

- I later added some commands to the top of this script file that created a log file that would store a log of when each command was being run:

```
#!/bin/bash

# Define log file
LOG_FILE="/farah_custom_data.log"

# Redirect stdout and stderr to the log file
exec > >(sudo tee -a "$LOG_FILE") 2>&1
```

2. After launching the instance, I waited 5 minutes before logging into it (note that, at this stage, I had provided a public IP to the database EC2/VM just so I could log in to troubleshoot, but I later removed it to increase security)

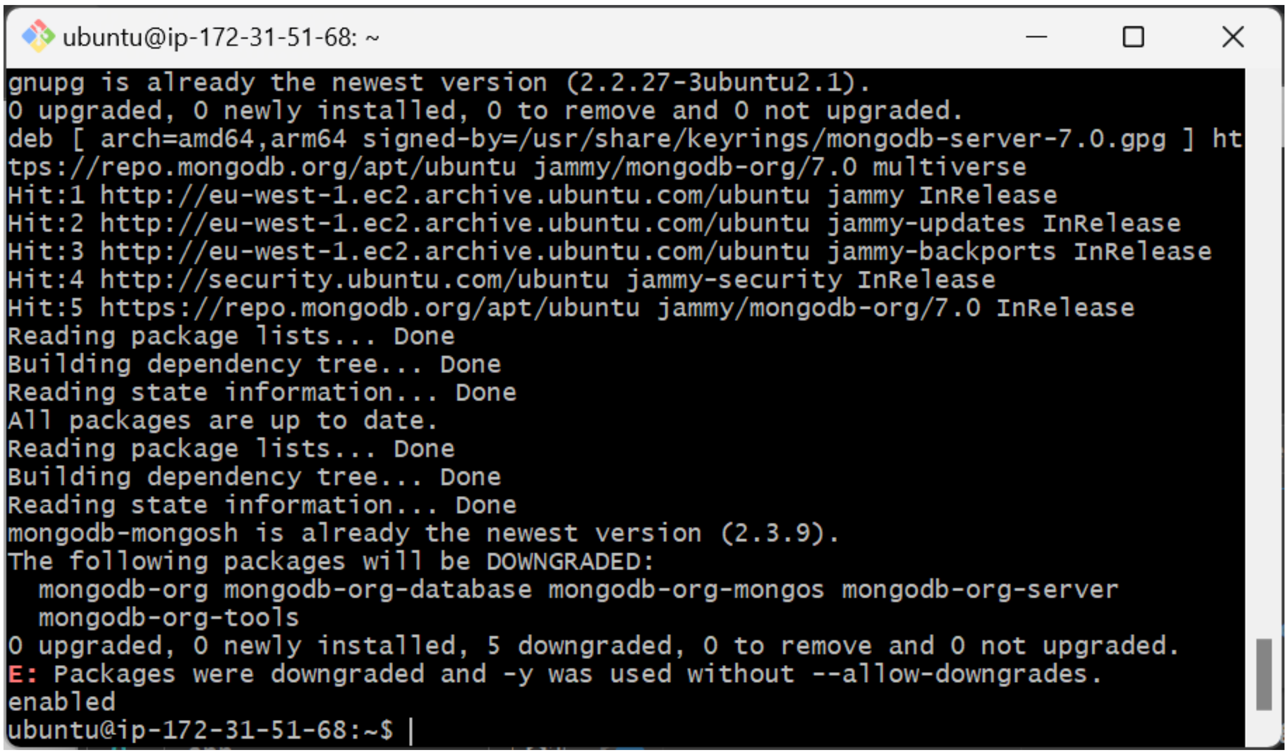
3. I then tested that the script had worked with `sudo systemctl status mongod`:



```
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

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

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

ubuntu@ip-172-31-56-225:~$ sudo systemctl status mongod
● mongod.service - MongoDB Database Server
     Loaded: loaded (/lib/systemd/system/mongod.service; enabled; vendor preset>
     Active: active (running) since Wed 2025-02-19 11:29:04 UTC; 32s ago
       Docs: https://docs.mongodb.org/manual
   Main PID: 8603 (mongod)
     Memory: 163.0M
        CPU: 1.301s
     CGroup: /system.slice/mongod.service
             └─8603 /usr/bin/mongod --config /etc/mongod.conf

Feb 19 11:29:04 ip-172-31-56-225 systemd[1]: Started MongoDB Database Server.
Feb 19 11:29:05 ip-172-31-56-225 mongod[8603]: {"t":{"$date":"2025-02-19T11:29:>
lines 1-12/12 (END)
```

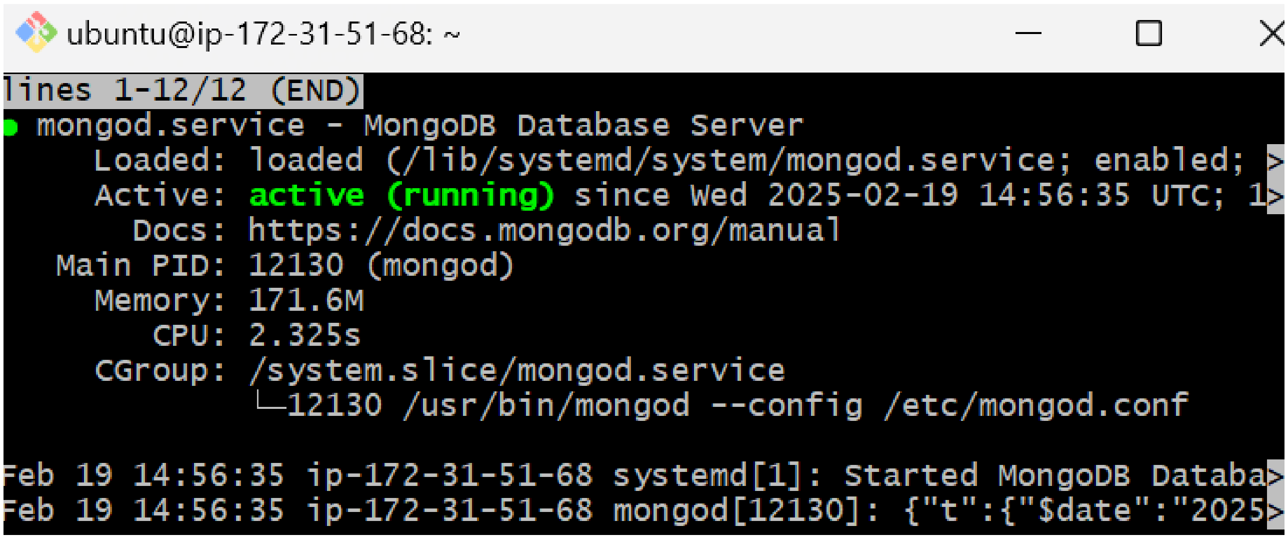
4. I then tested the idempotency of my script by running it again

- as the "downgraded" message shows, the script ran the `upgrade` command (which upgraded MongoDB) and then downgraded it to the version specified in my command



```
gnupg is already the newest version (2.2.27-3ubuntu2.1).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
deb [ arch=amd64,arm64 signed-by=/usr/share/keyrings/mongodb-server-7.0.gpg ] ht
tps://repo.mongodb.org/apt/ubuntu jammy/mongodb-org/7.0 multiverse
Hit:1 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:5 https://repo.mongodb.org/apt/ubuntu jammy/mongodb-org/7.0 InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
mongodb-mongosh is already the newest version (2.3.9).
The following packages will be DOWNGRADED:
  mongodb-org mongodb-org-database mongodb-org-mongos mongodb-org-server
  mongodb-org-tools
0 upgraded, 0 newly installed, 5 downgraded, 0 to remove and 0 not upgraded.
E: Packages were downgraded and -y was used without --allow-downgrades.
enabled
ubuntu@ip-172-31-51-68:~$ |
```

- and this shows that MongoDB was restarted afterwards



```
ubuntu@ip-172-31-51-68: ~
lines 1-12/12 (END)
● mongod.service - MongoDB Database Server
     Loaded: loaded (/lib/systemd/system/mongod.service; enabled; >
     Active: active (running) since Wed 2025-02-19 14:56:35 UTC; 1>
       Docs: https://docs.mongodb.org/manual
   Main PID: 12130 (mongod)
     Memory: 171.6M
        CPU: 2.325s
     CGroup: /system.slice/mongod.service
             └─12130 /usr/bin/mongod --config /etc/mongod.conf

Feb 19 14:56:35 ip-172-31-51-68 systemd[1]: Started MongoDB Databa>
Feb 19 14:56:35 ip-172-31-51-68 mongod[12130]: {"t":{"$date":"2025>
```

What I learnt

- the `sed` command syntax for replacing a string in a given file is: `sudo sed -i 's/<string to remove>/<string to add in its place>' <file path to do this in>`
  - the `-i` flag tells the `sed` command to write the results of the command to a file, not just to output it to the terminal

---

1b. Developing and implementing a Bash script to provision the app VM

1. I created a script file that did not need user input and contains the private IP of the above created DB EC2 in the connection string (noting that this would need to change for future iterations of the DB EC2, as the private IP will be different)

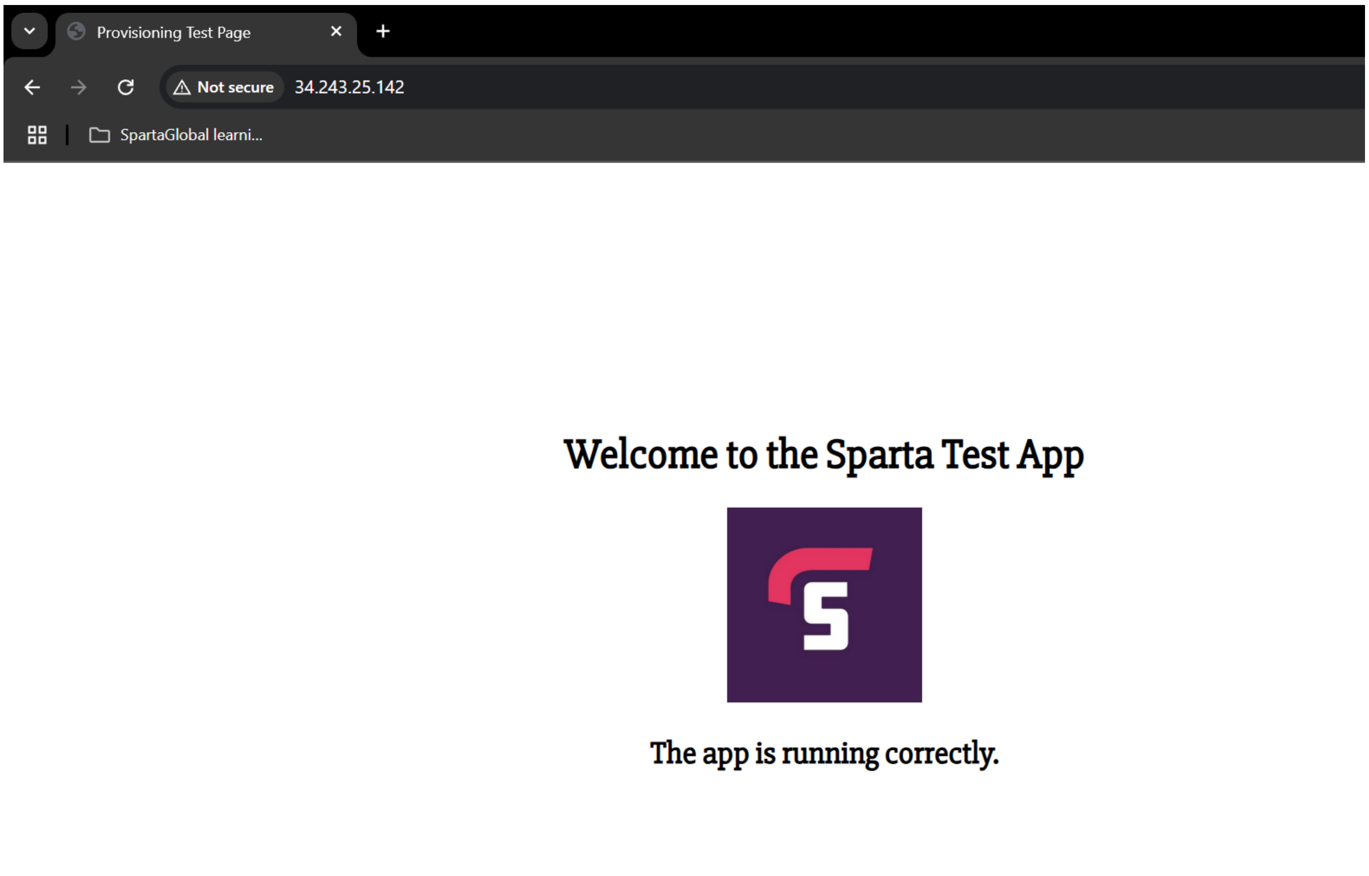
- To avoid the need for any manual configuration, I used the Bash `sed` command again to configure a reverse proxy via Nginx:

`sudo sed -i 's|try_files $uri $uri/ =404;|proxy_pass http://localhost:3000;|' /etc/nginx/sites-available/default`

- I later added some commands to the top of this script file that created a log file that would store a log of when each command was being run:

```
#!/bin/bash

# Define log file
LOG_FILE="/farah_custom_data.log"

# Redirect stdout and stderr to the log file
exec > >(sudo tee -a "$LOG_FILE") 2>&1
```

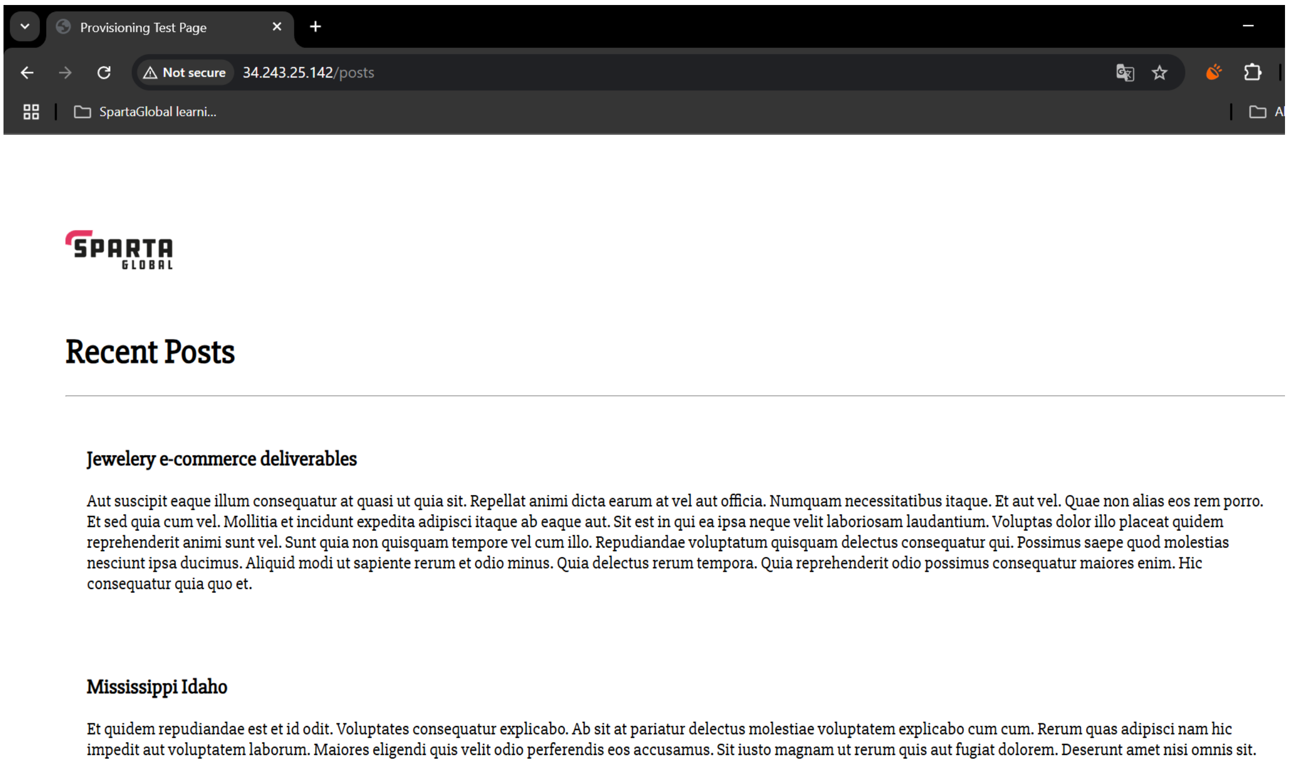
2. After launching the instance, I tested that the script had worked by navigating to the public IP of this app EC2:



Welcome to the Sparta Test App

The app is running correctly.

- and its */posts* page:



Recent Posts

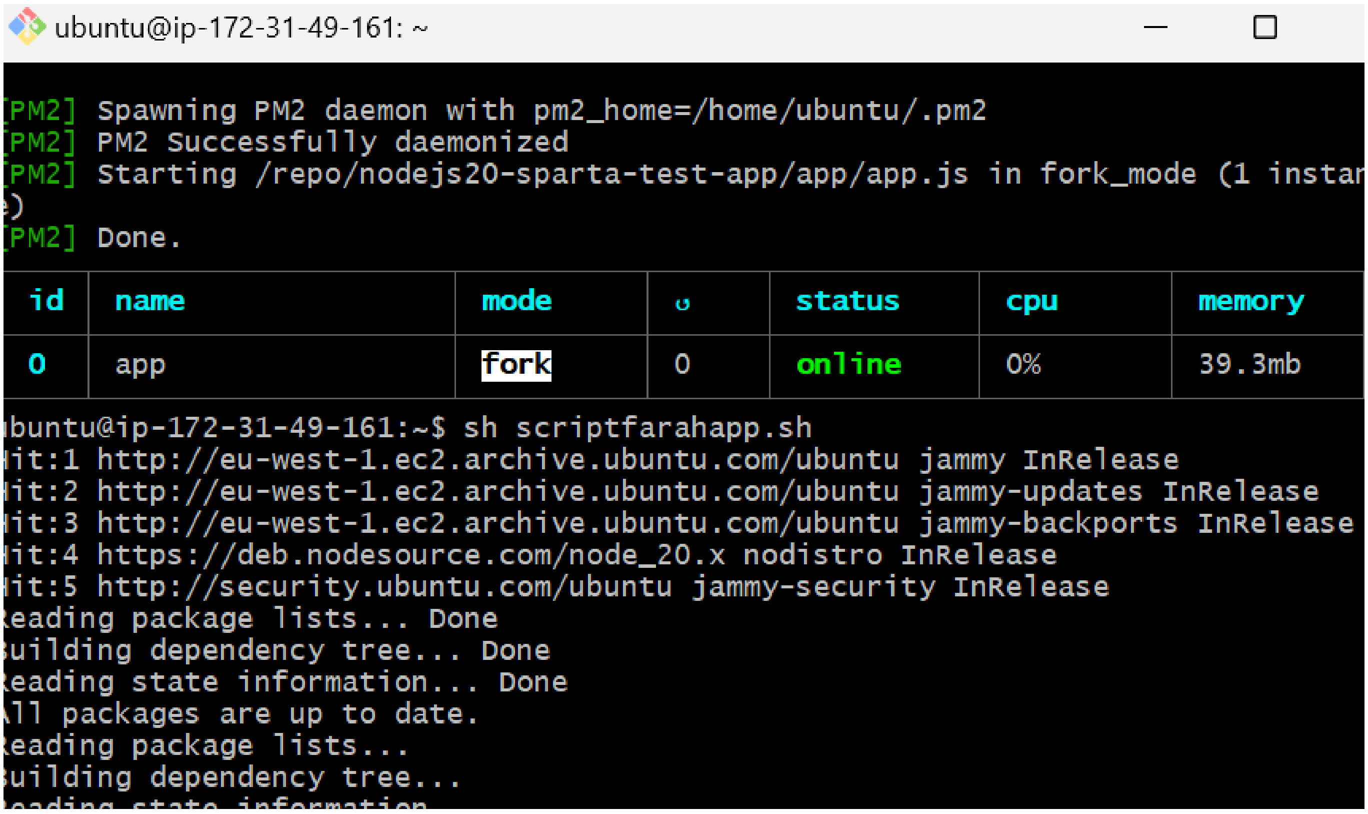
Jewelery e-commerce deliverables

Aut suscipit eaque illum consequatur at quasi ut quia sit. Repellat animi dicta earum at vel aut officia. Numquam necessitatibus itaque. Et aut vel. Quae non alias eos rem porro. Et sed quia cum vel. Mollitia et incidunt expedita adipisci itaque ab eaque aut. Sit est in qui ea ipsa neque velit laboriosam laudantium. Voluptas dolor illo placeat quidem reprehenderit animi sunt vel. Sunt quia non quisquam tempore vel cum illo. Repudiandae voluptatum quisquam delectus consequatur qui. Possimus saepe quod molestias nesciunt ipsa ducimus. Aliquid modi ut sapiente rerum et odio minus. Quia delectus rerum tempora. Quia reprehenderit odio possimus consequatur maiores enim. Hic consequatur quia quo et.

Mississippi Idaho

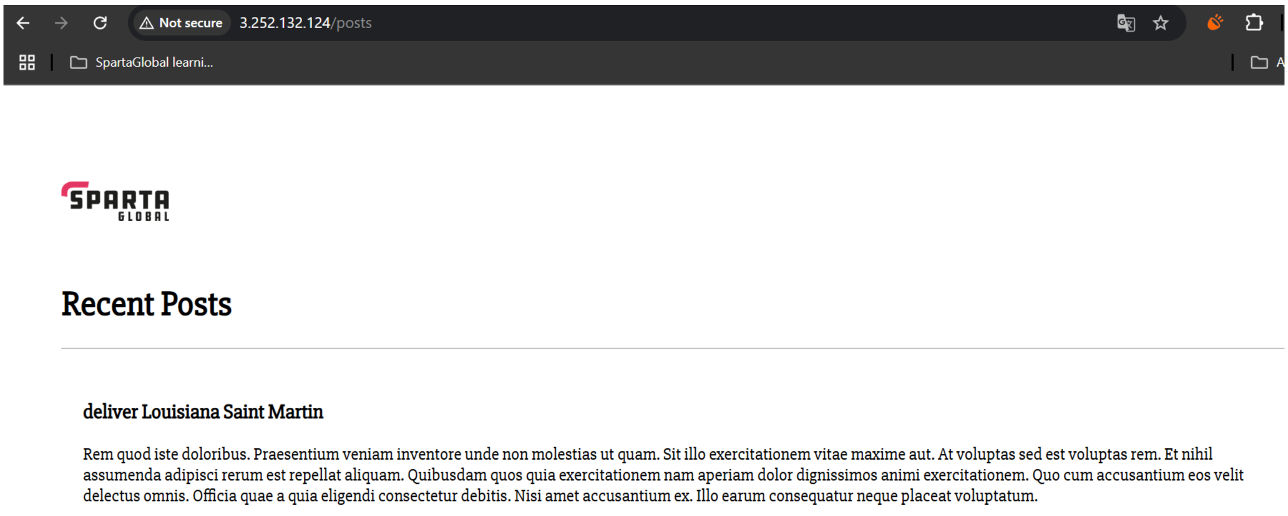
Et quidem repudiandae est et id odit. Voluptates consequatur explicabo. Ab sit at pariatur delectus molestiae voluptatem explicabo cum cum. Rerum quas adipisci nam hic impedit aut voluptatem laborum. Maiores eligendi quis velit odio perferendis eos accusamus. Sit iusto magnam ut rerum quis aut fugiat dolorem. Deserunt amet nisi omnis sit.

3. I then tested the idempotency of my script by running it twice when the app was already running



```
ubuntu@ip-172-31-49-161: ~
[PM2] Spawning PM2 daemon with pm2_home=/home/ubuntu/.pm2
[PM2] PM2 Successfully daemonized
[PM2] Starting /repo/nodejs20-sparta-test-app/app/app.js in fork_mode (1 instar
)
[PM2] Done.
│ id │ name  │ mode │ ↺ │ status │ cpu │ memory │
│ 0  │ app   │ fork │ 0 │ online │ 0%  │ 39.3mb │
ıbuntu@ip-172-31-49-161:~$ sh scriptfarahapp.sh
lit:1 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
lit:2 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease
lit:3 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
lit:4 https://deb.nodesource.com/node_20.x nodistro InRelease
lit:5 http://security.ubuntu.com/ubuntu jammy-security InRelease
eading package lists... Done
uilding dependency tree... Done
eading state information... Done
ll packages are up to date.
eading package lists...
uilding dependency tree...
```

- and it started the app again (note the different records, indicating that the `node seeds/seed.js` command had run successfully)



Not secure 3.252.132.124/posts

SpartaGlobal learni...

Recent Posts

deliver Louisiana Saint Martin

Rem quod iste doloribus. Praesentium veniam inventore unde non molestias ut quam. Sit illo exercitationem vitae maxime aut. At voluptas sed est voluptas rem. Et nihil assumenda adipisci rerum est repellat aliquam. Quibusdam quos quia exercitationem nam aperiam dolor dignissimos animi exercitationem. Quo cum accusantium eos velit delectus omnis. Officia quae a quia eligendi consectetur debitis. Nisi amet accusantium ex. Illo earum consequatur neque placeat voluptatum.

What I learnt

- in my `sed` command to configure the reverse proxy, I needed to switch to using | as the command delimiter because the / in the string confused it as it is also used as the standard delimiter: `sudo sed -i 's|try_files $uri $uri/ =404;|proxy_pass http://localhost:3000;|' /etc/nginx/sites-available/default`
- I needed to change ownership of the newly git-cloned repo folder because otherwise it wouldn't let me run `npm install` without `sudo`
  - However I retained the `sudo` in my commands just to be safe

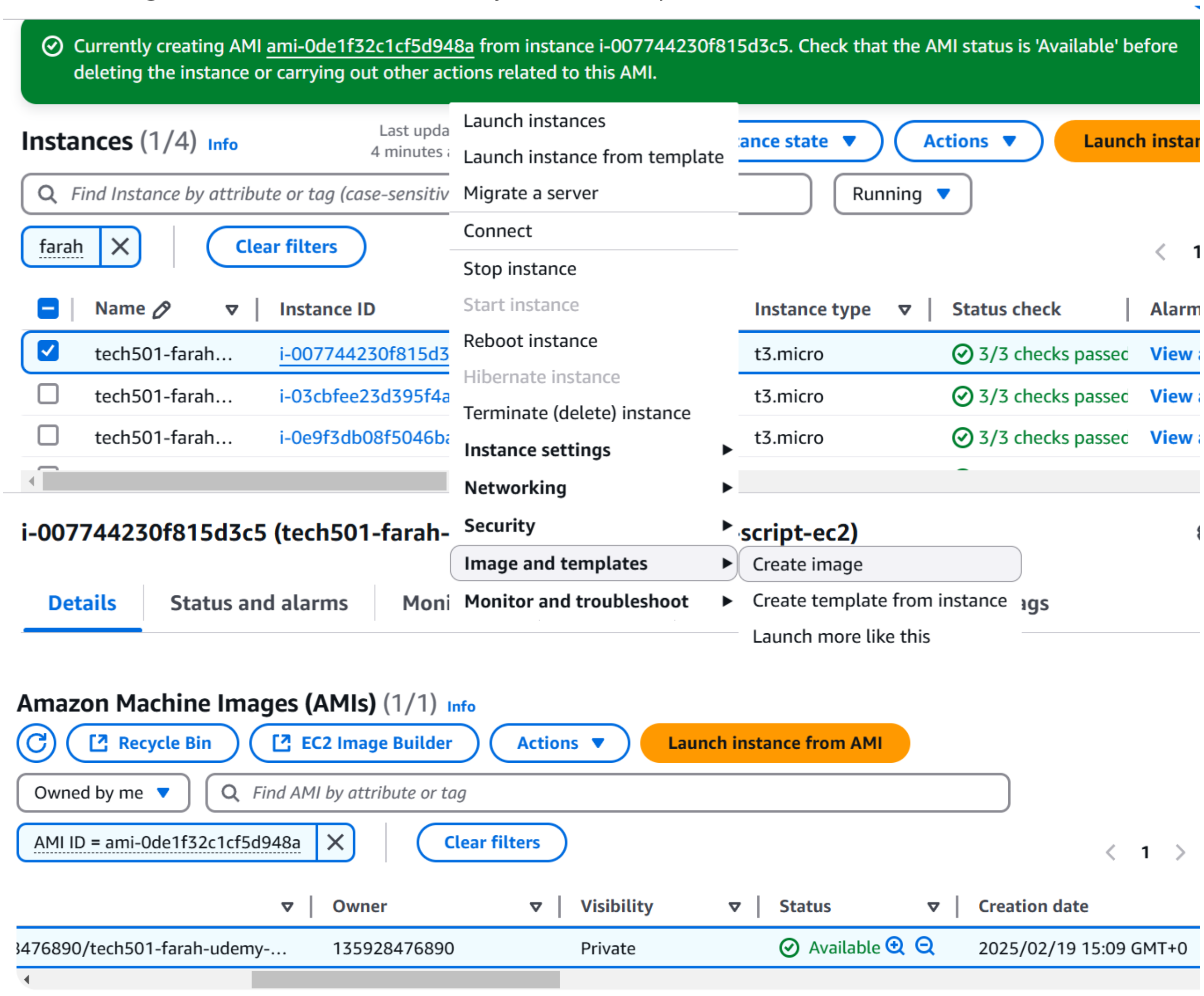
- I needed to add an `npm audit fix` command to my script to remove some errors that I couldn't get past otherwise
- I needed to add a `node seeds/seed.js` command to my script in case my */posts* page hadn't been seeded properly

---

2. Creating and testing images to run the app and database

1. I created images from both of the above provisioned EC2s:

1. **DB EC2 image name**: *tech501-farah-udemy-db-from-script*



Currently creating AMI ami-0de1f32c1cf5d948a from instance i-007744230f815d3c5. Check that the AMI status is 'Available' before deleting the instance or carrying out other actions related to this AMI.

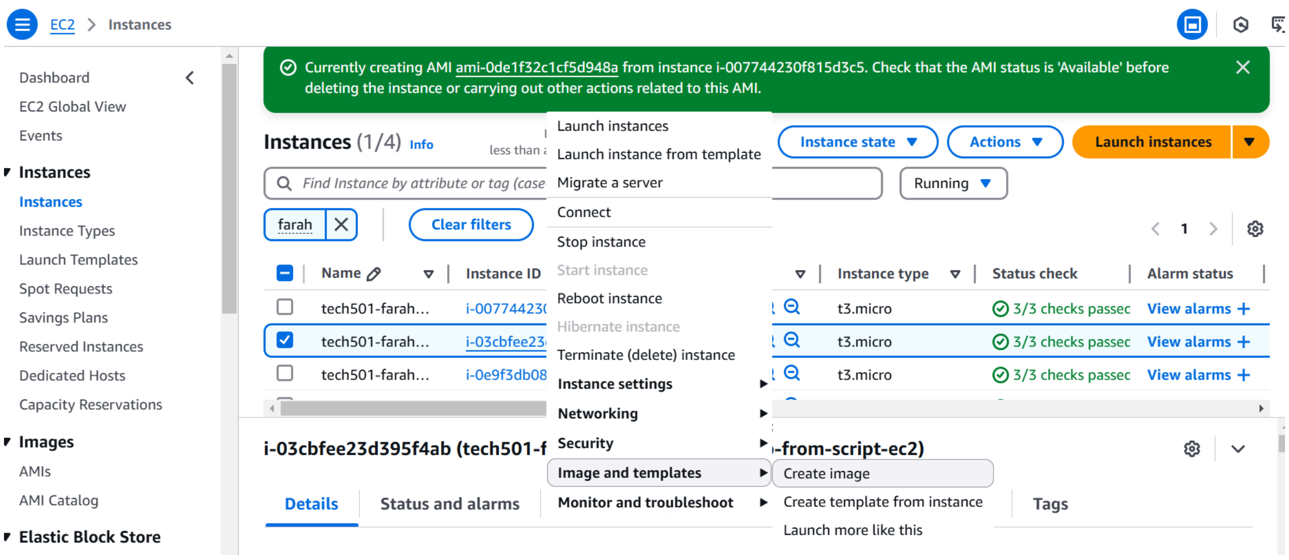
Name	Instance ID	Instance type	Status check
tech501-farah...	i-007744230f815d3	t3.micro	3/3 checks passec
tech501-farah...	i-03cbfee23d395f4a	t3.micro	3/3 checks passec
tech501-farah...	i-0e9f3db08f5046ba	t3.micro	3/3 checks passec

i-007744230f815d3c5 (tech501-farah- ... script-ec2)

Amazon Machine Images (AMIs) (1/1) Info
Recycle Bin
EC2 Image Builder
Actions
Launch instance from AMI
Owned by me
Find AMI by attribute or tag
AMI ID = ami-0de1f32c1cf5d948a
Clear filters

	Owner	Visibility	Status	Creation date
3476890/tech501-farah-udemy-...	135928476890	Private	Available	2025/02/19 15:09 GMT+0

2. **App EC2 image name**: *tech501-farah-udemy-app-from-script*



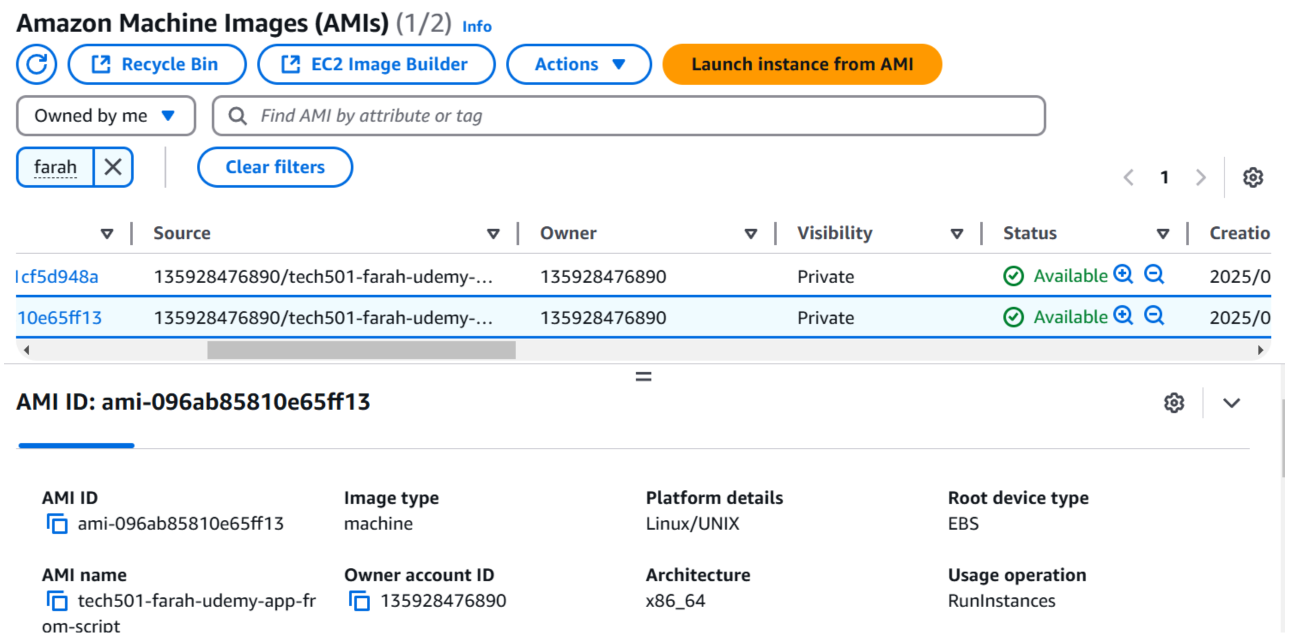
EC2 > Instances

Currently creating AMI ami-0de1f32c1cf5d948a from instance i-007744230f815d3c5. Check that the AMI status is 'Available' before deleting the instance or carrying out other actions related to this AMI.

Instances (1/4) Info

	Name	Instance ID	Instance type	Status check	Alarm status
☐	tech501-farah…	i-00774423(	t3.micro	3/3 checks passec	View alarms +
☑	tech501-farah…	i-03cbfee23	t3.micro	3/3 checks passec	View alarms +
☐	tech501-farah…	i-0e9f3db08	t3.micro	3/3 checks passec	View alarms +

i-03cbfee23d395f4ab (tech501-f …-from-script-ec2)



Amazon Machine Images (AMIs) (1/2) Info

Source	Owner	Visibility	Status	Creatio
135928476890/tech501-farah-udemy-…	135928476890	Private	Available	2025/0
135928476890/tech501-farah-udemy-…	135928476890	Private	Available	2025/0

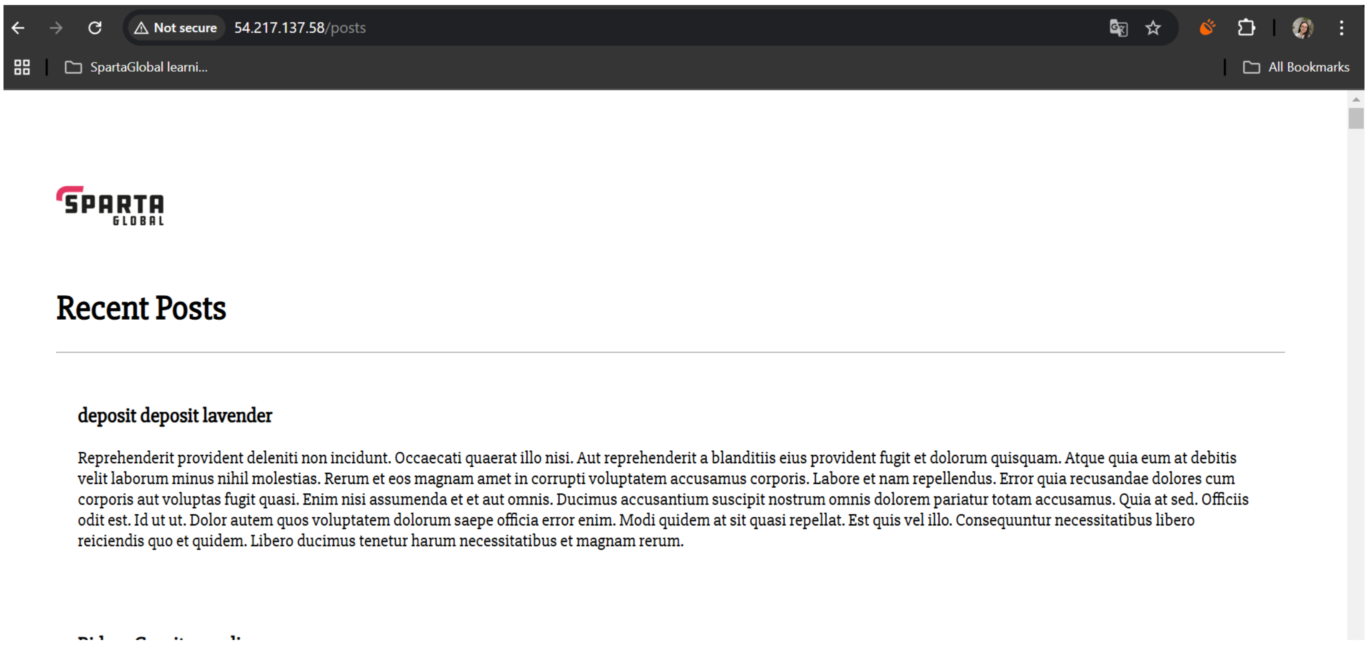
AMI ID: ami-096ab85810e65ff13

AMI ID	Image type	Platform details	Root device type
ami-096ab85810e65ff13	machine	Linux/UNIX	EBS
AMI name	Owner account ID	Architecture	Usage operation
tech501-farah-udemy-app-from-script	135928476890	x86_64	RunInstances

2. I then used the **Launch instance from AMI** feature to create EC2s from both of these images, using the same settings I initially created (apart from the AMI, as this is now different)

- I didn't use scripts in the **User data field**, except a *run-app-only.sh* script found here used when creating the app EC2 (this overwrites the old connection string and starts the app on this machine when it runs for the first time)

3. I then followed this process again, and on this second run, tested that the */posts* page worked and it did

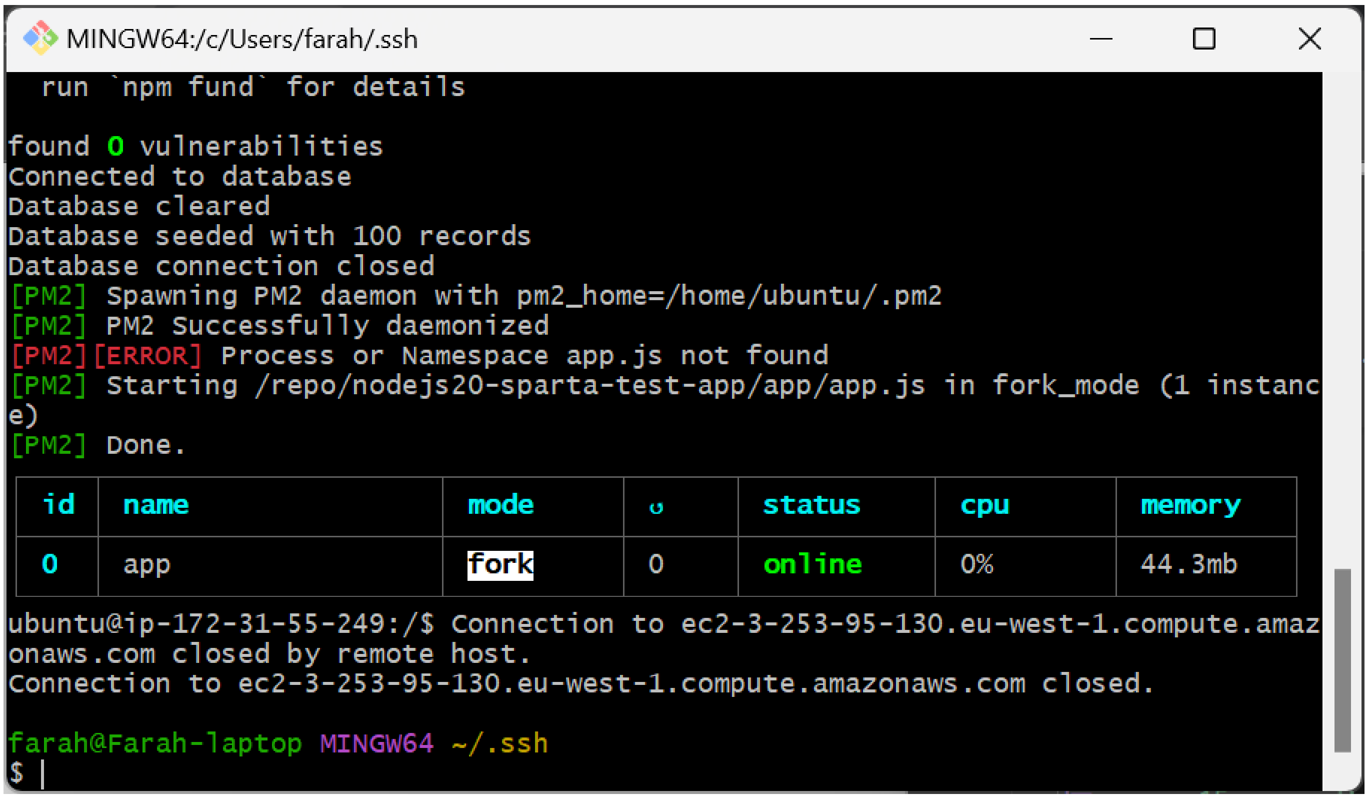


Recent Posts

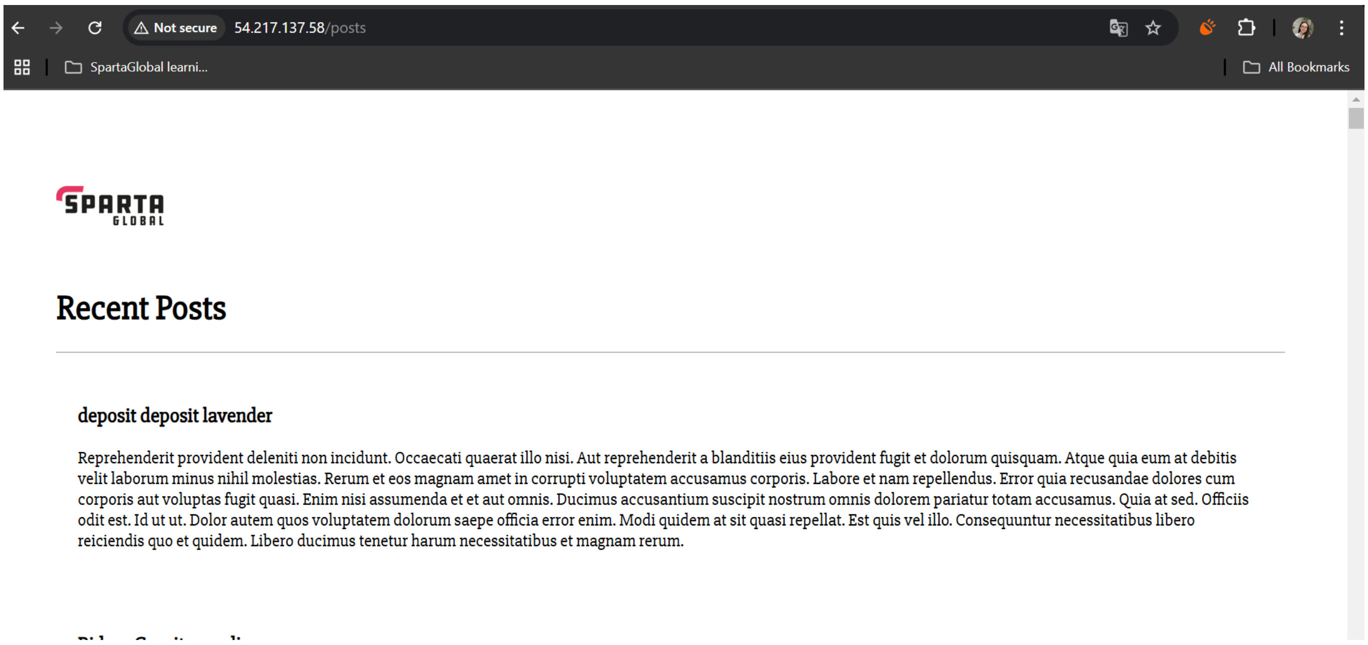
deposit deposit lavender

Reprehenderit provident deleniti non incidunt. Occaecati quaerat illo nisi. Aut reprehenderit a blanditiis eius provident fugit et dolorum quisquam. Atque quia eum at debitis velit laborum minus nihil molestias. Rerum et eos magnam amet in corrupti voluptatem accusamus corporis. Labore et nam repellendus. Error quia recusandae dolores cum corporis aut voluptas fugit quasi. Enim nisi assumenda et et aut omnis. Ducimus accusantium suscipit nostrum omnis dolorem pariatur totam accusamus. Quia at sed. Officiis odit est. Id ut ut. Dolor autem quos voluptatem dolorum saepe officia error enim. Modi quidem at sit quasi repellat. Est quis vel illo. Consequuntur necessitatibus libero reiciendis quo et quidem. Libero ducimus tenetur harum necessitatibus et magnam rerum.

4. I then logged out of this second app EC2, logged back in, created a local version of the *run-app-only.sh* script, ran it, and it worked perfectly (note the new records to indicate the reseeding of the database)



```
  run `npm fund` for details

found 0 vulnerabilities
Connected to database
Database cleared
Database seeded with 100 records
Database connection closed
[PM2] Spawning PM2 daemon with pm2_home=/home/ubuntu/.pm2
[PM2] PM2 Successfully daemonized
[PM2][ERROR] Process or Namespace app.js not found
[PM2] Starting /repo/nodejs20-sparta-test-app/app/app.js in fork_mode (1 instanc
e)
[PM2] Done.
┌────┬──────────────────────┬──────────┬──────┬───────────┬──────────┬──────────┐
│ id │ name                 │ mode     │ ↺    │ status    │ cpu      │ memory   │
├────┼──────────────────────┼──────────┼──────┼───────────┼──────────┼──────────┤
│ 0  │ app                  │ fork     │ 0    │ online    │ 0%       │ 44.3mb   │
└────┴──────────────────────┴──────────┴──────┴───────────┴──────────┴──────────┘
ubuntu@ip-172-31-55-249:/$ Connection to ec2-3-253-95-130.eu-west-1.compute.amaz
onaws.com closed by remote host.
Connection to ec2-3-253-95-130.eu-west-1.compute.amazonaws.com closed.

farah@Farah-laptop MINGW64 ~/.ssh
$ |
```



Recent Posts

deposit deposit lavender

Reprehenderit provident deleniti non incidunt. Occaecati quaerat illo nisi. Aut reprehenderit a blanditiis eius provident fugit et dolorum quisquam. Atque quia eum at debitis velit laborum minus nihil molestias. Rerum et eos magnam amet in corrupti voluptatem accusamus corporis. Labore et nam repellendus. Error quia recusandae dolores cum corporis aut voluptas fugit quasi. Enim nisi assumenda et et aut omnis. Ducimus accusantium suscipit nostrum omnis dolorem pariatur totam accusamus. Quia at sed. Officiis odit est. Id ut ut. Dolor autem quos voluptatem dolorum saepe officia error enim. Modi quidem at sit quasi repellat. Est quis vel illo. Consequuntur necessitatibus libero reiciendis quo et quidem. Libero ducimus tenetur harum necessitatibus et magnam rerum.

What I learnt

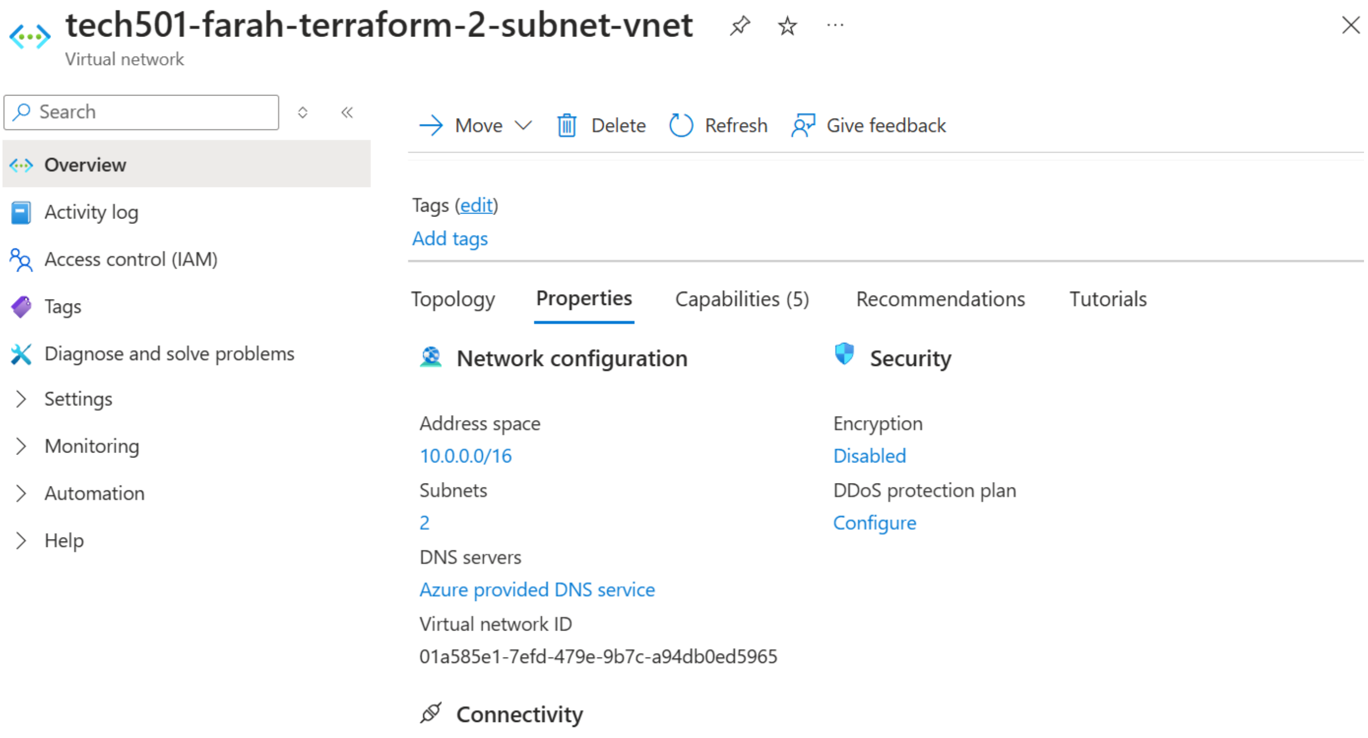
- That if I added a `node seeds/seed.js` command into my *run-app-only.sh* script (as well as the prov-app.sh script), I could be sure that the script was running successfully because the records on the */posts* page would be different — this helped me ensure the scripts really were running successfully, and not just being stored in my browser's cache
- Only the public IP changes on reboot on AWS (unless we have an elastic IP)
  - i.e. The private IP of an EC2 instance doesn't change after it has been restarted so my *run-app-only.sh* script can be used repeatedly without modifying it so long as I am working with the same DB EC2 (i.e. I haven't created another one)
- I modified my *run-app-only.sh* script so that, before the `pm2 start app.js` line, it ran `pm2 delete app.js` to remove the app from pm2's process list, which seemed to be causing errors when rerunning the script — this is to ensure the script is idempotent

(Optional) 3. Following the same steps for Azure using Terraform

1. After I successfully completed the above parts using AWS and its GUI, I wanted to redo the task on Azure using Terraform to set up the 2-subnet VN and the two VMs

- The *.tf* files used for this part can be found here (with sensitive information hidden in a git-ignored variable file)

2. VN created using this method:


tech501-farah-terraform-2-subnet-vnet
Virtual network

Tags (edit)
Add tags

Network configuration

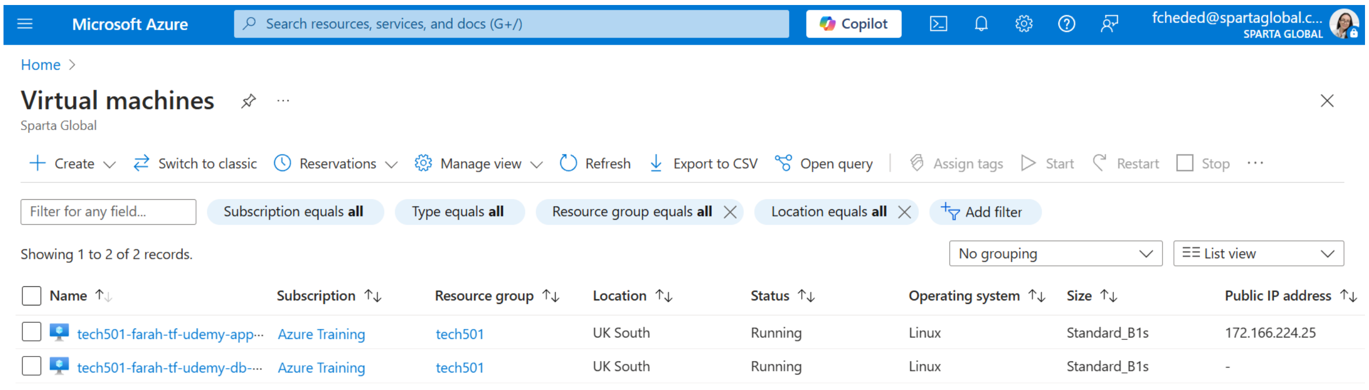
Address space
10.0.0.0/16
Subnets
2
DNS servers
Azure provided DNS service
Virtual network ID
01a585e1-7efd-479e-9b7c-a94db0ed5965

Security

Encryption
Disabled
DDoS protection plan
Configure

Connectivity

3. VMs created using this method:



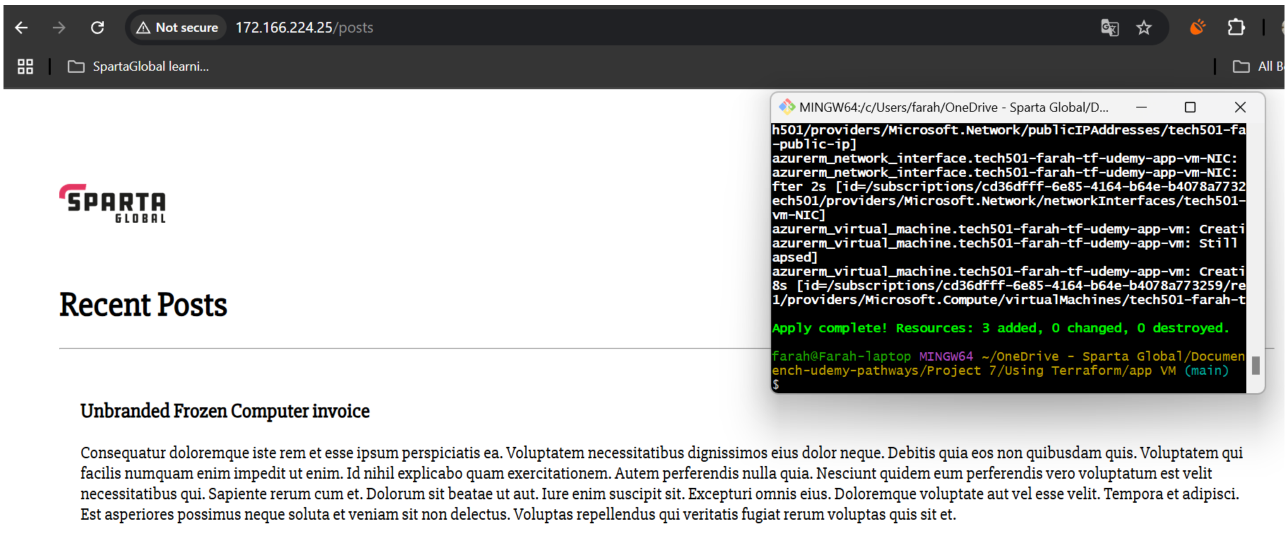
Virtual machines
Sparta Global

Subscription equals **all** | Type equals **all** | Resource group equals **all** | Location equals **all**

Showing 1 to 2 of 2 records.

Name	Subscription	Resource group	Location	Status	Operating system	Size	Public IP address
tech501-farah-tf-udemy-app-...	Azure Training	tech501	UK South	Running	Linux	Standard_B1s	172.166.224.25
tech501-farah-tf-udemy-db-...	Azure Training	tech501	UK South	Running	Linux	Standard_B1s	-

- Successful posts page using this method:



Not secure 172.166.224.25/posts

SpartaGlobal learni...

Recent Posts

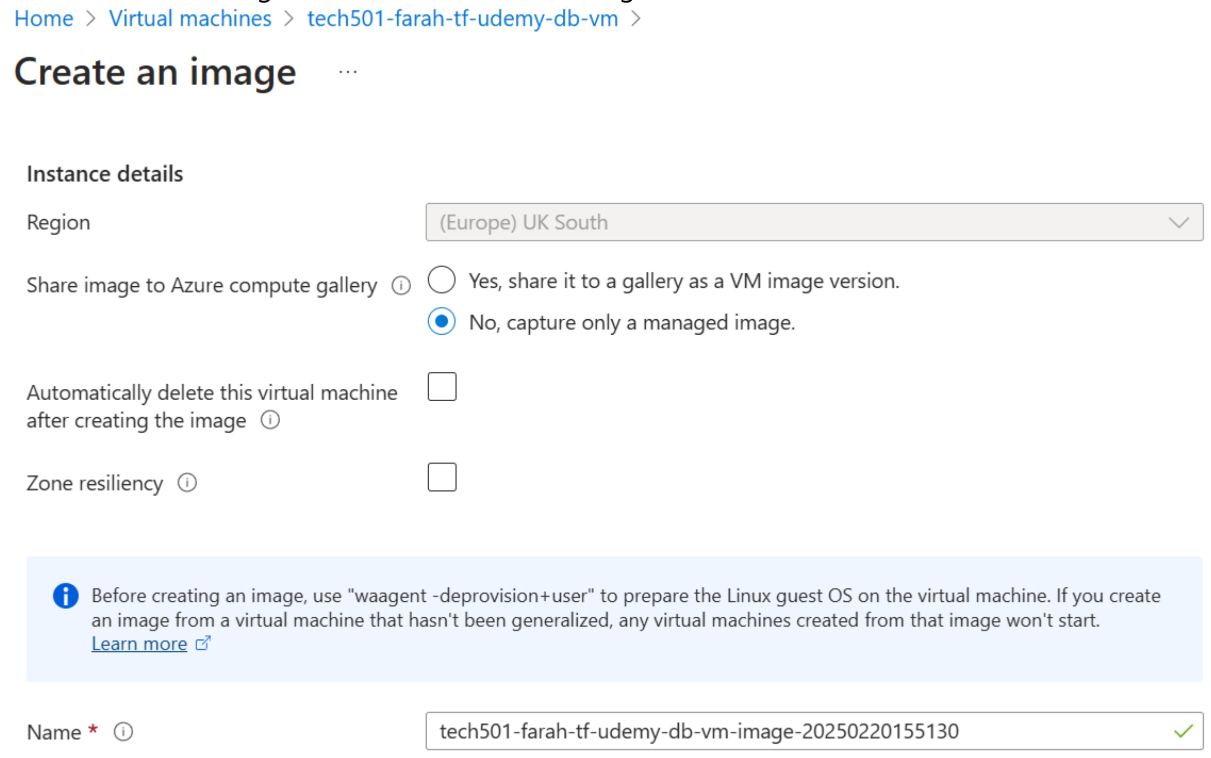
Unbranded Frozen Computer invoice

Consequatur doloremque iste rem et esse ipsum perspiciatis ea. Voluptatem necessitatibus dignissimos eius dolor neque. Debitis quia eos non quibusdam quis. Voluptatem qui facilis numquam enim impedit ut enim. Id nihil explicabo quam exercitationem. Autem perferendis nulla quia. Nesciunt quidem eum perferendis vero voluptatum est velit necessitatibus qui. Sapiente rerum cum et. Dolorum sit beatae ut aut. Iure enim suscipit sit. Excepturi omnis eius. Doloremque voluptate aut vel esse velit. Tempora et adipisci. Est asperiores possimus neque soluta et veniam sit non delectus. Voluptas repellendus qui veritatis fugiat rerum voluptas quis sit et.

MINGW64:/c/Users/farah/OneDrive - Sparta Global/D...

```
h501/providers/Microsoft.Network/publicIPAddresses/tech501-fa
-public-ip]
azurerm_network_interface.tech501-farah-tf-udemy-app-vm-NIC:
azurerm_network_interface.tech501-farah-tf-udemy-app-vm-NIC:
fter 2s [id=/subscriptions/cd36dfff-6e85-4164-b64e-b4078a7732
ech501/providers/Microsoft.Network/networkInterfaces/tech501-
vm-NIC]
azurerm_virtual_machine.tech501-farah-tf-udemy-app-vm: Creati
azurerm_virtual_machine.tech501-farah-tf-udemy-app-vm: Still
apsed]
azurerm_virtual_machine.tech501-farah-tf-udemy-app-vm: Creati
8s [id=/subscriptions/cd36dfff-6e85-4164-b64e-b4078a773259/re
1/providers/Microsoft.Compute/virtualMachines/tech501-farah-t

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.

farah@Farah-laptop MINGW64 ~/OneDrive - Sparta Global/Documen
ench-udemy-pathways/Project 7/Using Terraform/app VM (main)
$
```

4. I then created images of both the above VMs e.g.:



Home > Virtual machines > tech501-farah-tf-udemy-db-vm >

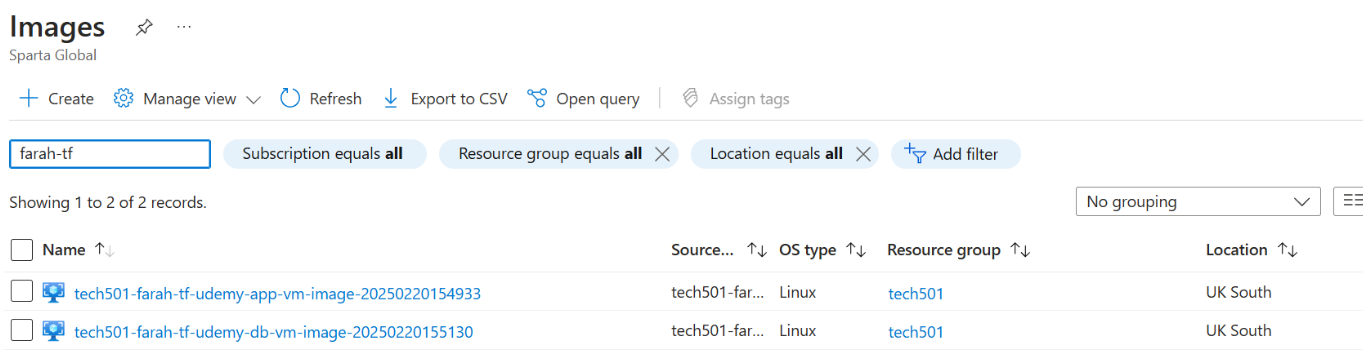
Create an image ...

Instance details

Region	(Europe) UK South
Share image to Azure compute gallery	Yes, share it to a gallery as a VM image version. [✓ No, capture only a managed image.]
Automatically delete this virtual machine after creating the image	☐
Zone resiliency	☐

Before creating an image, use "waagent -deprovision+user" to prepare the Linux guest OS on the virtual machine. If you create an image from a virtual machine that hasn't been generalized, any virtual machines created from that image won't start. Learn more

Name *	tech501-farah-tf-udemy-db-vm-image-20250220155130



Images

Sparta Global

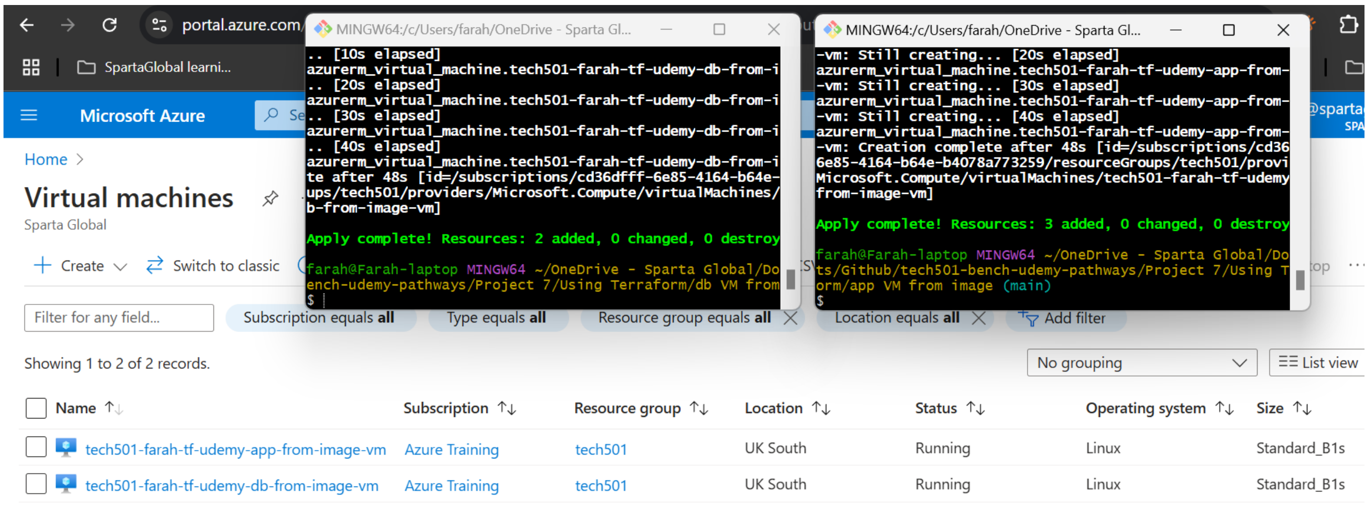
farah-tf | Subscription equals **all** | Resource group equals **all** | Location equals **all** | Add filter

Showing 1 to 2 of 2 records.

No grouping

Name	Source...	OS type	Resource group	Location
tech501-farah-tf-udemy-app-vm-image-20250220154933	tech501-far...	Linux	tech501	UK South
tech501-farah-tf-udemy-db-vm-image-20250220155130	tech501-far...	Linux	tech501	UK South

5. Using Terraform, I then created VMs from both of these images (with an added *run-app-only.sh* script provided as **custom data** for the app VM):



```
.. [10s elapsed]
azurerm_virtual_machine.tech501-farah-tf-udemy-db-from-i
.. [20s elapsed]
azurerm_virtual_machine.tech501-farah-tf-udemy-db-from-i
.. [30s elapsed]
azurerm_virtual_machine.tech501-farah-tf-udemy-db-from-i
.. [40s elapsed]
azurerm_virtual_machine.tech501-farah-tf-udemy-db-from-i
te after 48s [id=/subscriptions/cd36dfff-6e85-4164-b64e-
ups/tech501/providers/Microsoft.Compute/virtualMachines/
b-from-image-vm]

Apply complete! Resources: 2 added, 0 changed, 0 destroy

farah@Farah-laptop MINGW64 ~/OneDrive - Sparta Global/Do
ench-udemy-pathways/Project 7/Using Terraform/db VM from
$
```

```
-vm: Still creating... [20s elapsed]
azurerm_virtual_machine.tech501-farah-tf-udemy-app-from-
-vm: Still creating... [30s elapsed]
azurerm_virtual_machine.tech501-farah-tf-udemy-app-from-
-vm: Still creating... [40s elapsed]
azurerm_virtual_machine.tech501-farah-tf-udemy-app-from-
-vm: Creation complete after 48s [id=/subscriptions/cd36
6e85-4164-b64e-b4078a773259/resourceGroups/tech501/provi
Microsoft.Compute/virtualMachines/tech501-farah-tf-udemy
from-image-vm]

Apply complete! Resources: 3 added, 0 changed, 0 destroy

farah@Farah-laptop MINGW64 ~/OneDrive - Sparta Global/Do
ts/Github/tech501-bench-udemy-pathways/Project 7/Using T
orm/app VM from image (main)
$
```

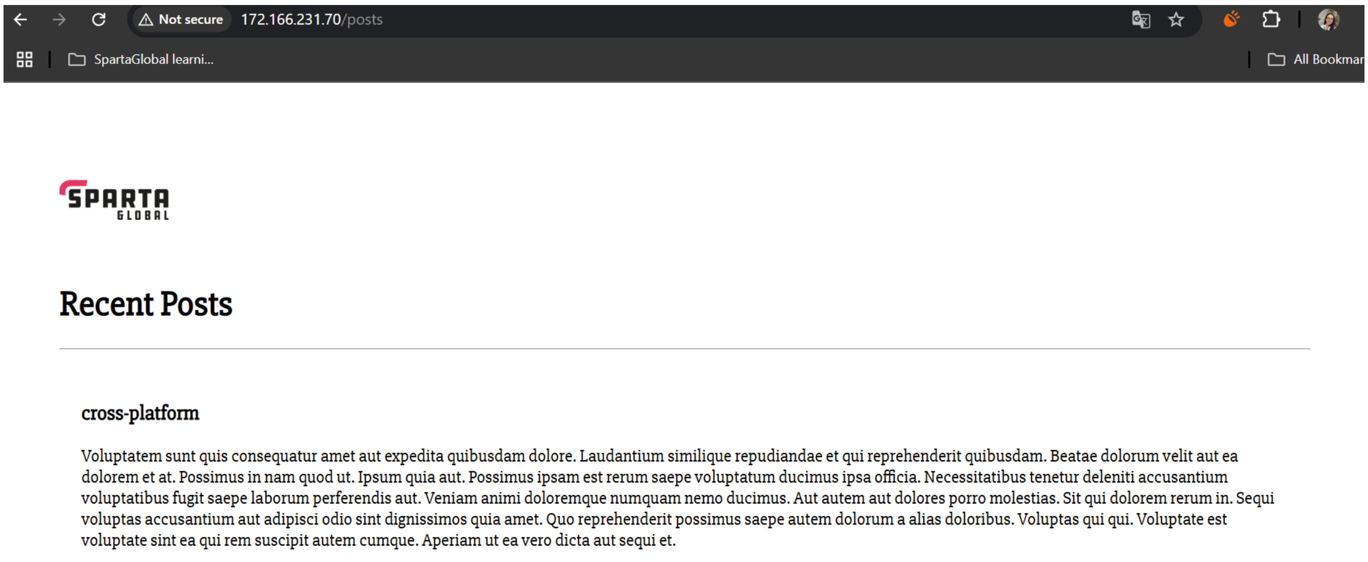
Virtual machines
Sparta Global

Filter: Subscription equals all · Type equals all · Resource group equals all · Location equals all

Showing 1 to 2 of 2 records.

Name	Subscription	Resource group	Location	Status	Operating system	Size
tech501-farah-tf-udemy-app-from-image-vm	Azure Training	tech501	UK South	Running	Linux	Standard_B1s
tech501-farah-tf-udemy-db-from-image-vm	Azure Training	tech501	UK South	Running	Linux	Standard_B1s

6. Finally, I tested these VMs from images by accessing the posts page:



Not secure 172.166.231.70/posts

SPARTA GLOBAL

Recent Posts

cross-platform

Voluptatem sunt quis consequatur amet aut expedita quibusdam dolore. Laudantium similique repudiandae et qui reprehenderit quibusdam. Beatae dolorum velit aut ea dolorem et at. Possimus in nam quod ut. Ipsum quia aut. Possimus ipsam est rerum saepe voluptatum ducimus ipsa officia. Necessitatibus tenetur deleniti accusantium voluptatibus fugit saepe laborum perferendis aut. Veniam animi doloremque numquam nemo ducimus. Aut autem aut dolores porro molestias. Sit qui dolorem rerum in. Sequi voluptas accusantium aut adipisci odio sint dignissimos quia amet. Quo reprehenderit possimus saepe autem dolorum a alias doloribus. Voluptas qui qui. Voluptate est voluptate sint ea qui rem suscipit autem cumque. Aperiam ut ea vero dicta aut sequi et.

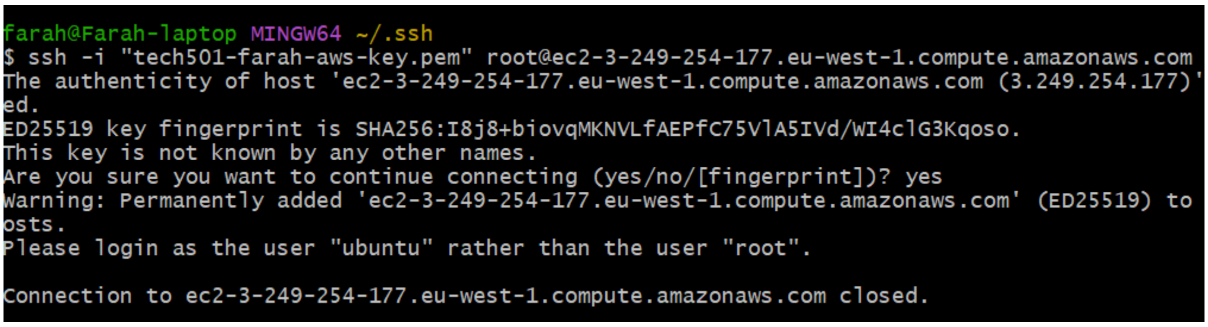
What I learnt

- That in Terraform, Azure uses names, not IDs, for its images

---

Blockers

- When logging into any of the EC2s created from my images, I needed to specify that I wanted to login as the `ubuntu` user like this: `ssh -i "tech501-farah-aws-key.pem" ubuntu@ec2`...
  - Otherwise I got this error



```
farah@Farah-laptop MINGW64 ~/.ssh
$ ssh -i "tech501-farah-aws-key.pem" root@ec2-3-249-254-177.eu-west-1.compute.amazonaws.com
The authenticity of host 'ec2-3-249-254-177.eu-west-1.compute.amazonaws.com (3.249.254.177)'
ed.
ED25519 key fingerprint is SHA256:I8j8+biovqMKNVLfAEPfC75VlA5IVd/WI4clG3Kqoso.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-3-249-254-177.eu-west-1.compute.amazonaws.com' (ED25519) to
osts.
Please login as the user "ubuntu" rather than the user "root".

Connection to ec2-3-249-254-177.eu-west-1.compute.amazonaws.com closed.
```

- I did run into a couple of errors getting the */posts* page to load, which was because I hadn't modified the private IP in the connection string when running the `export` command, so this was easily fixed
- In the early stages of the project, there were a couple of occasions when I was unsure if my User Data scripts were running successfully, so I modified my script files so that before a command was executed, it provided output to a log file — this way I could tell if the script was running successfully or not

Benefits I personally saw from the project

- The more I automated, the simpler the provisioning and starting of the app got
- Because of this, I preferred using Terraform over the cloud providers' GUIs, as I could easily build and tear down any resources with a few uses of my keyboard rather than following a ClickOps approach
- I preferred using Azure over AWS in general because, during testing (when I assigned a public IP to the MongoDB VM to enable manual logging in to check), I no longer had to modify the connection string used in the `export` command on the app VM as this was always the same, even on different iterations of my database VM
    - Because of this, in future, if I was creating something with **an IP address I needed to keep the same** (e.g. a VM for a Jenkins server) and I didn't have access to an Elastic IP on AWS, I would prefer to use Azure — because we discovered that AWS's changing IP address caused Jenkins to work very slowly after a reboot of the EC2 it was running on