

Project 7: Bash scripting

- **Project 7: Bash scripting**
 - GitHub repo
 - Goal of the project
 - Diagram showing my automation workflow
- **Detailed steps**
 - **1. Initial VM creation**
 - **2a. Developing and implementing a Bash script to provision the database VM**
 - What I learnt
 - **2b. Developing and implementing a Bash script to provision the app VM**
 - What I learnt
 - **3. Creating and testing images to run the app and database**
 - What I learnt
 - **(Optional) 4. Following the same steps for Azure using Terraform**
 - What I learnt
 - **Blockers**
 - **Benefits I personally saw from the project**

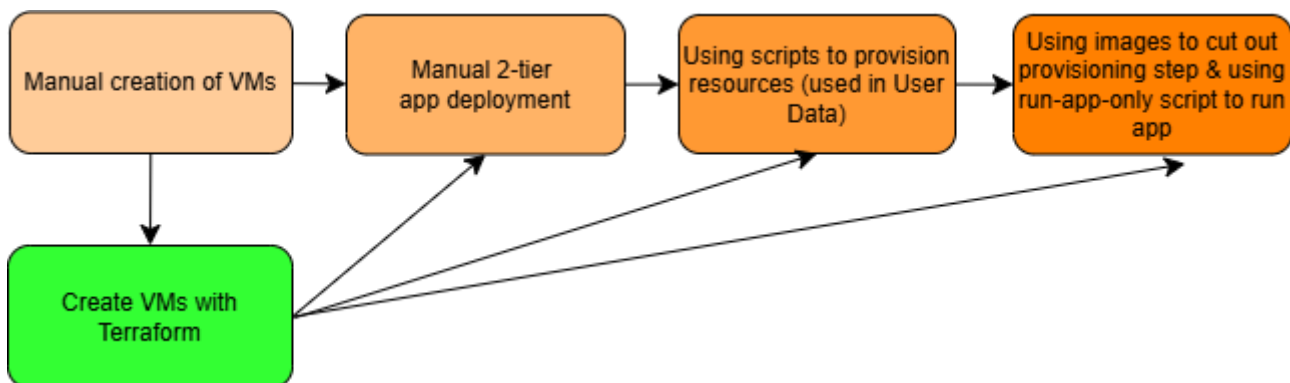
GitHub repo

[Available here](#)

Goal of the project

- To automate the deployment of a 2-tier app using **Bash scripts** and images that provision two VMs (in this case, EC2 instances) with all the required dependencies for the app and its database, without needing user input or manual configuration at any stage
 - these scripts need to be idempotent, i.e. repeatable without error on the same machine (this will be tested by running them as **User Data** when creating the EC2s, and then again successfully as a script file on the EC2s)

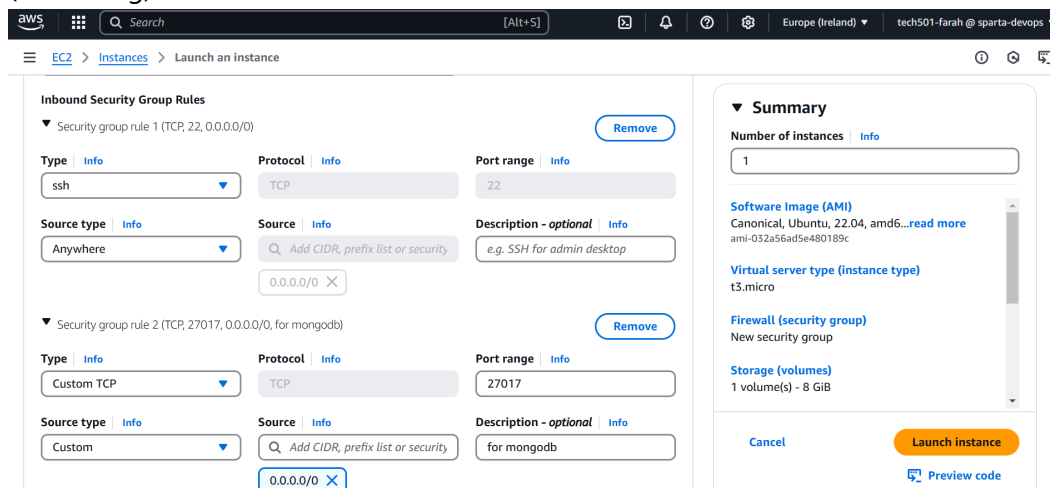
Diagram showing my automation workflow



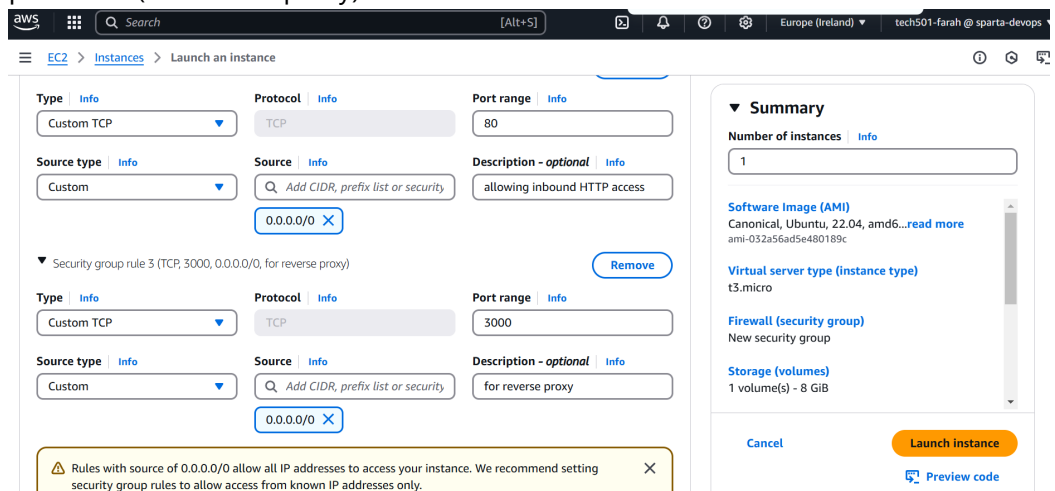
Detailed steps

1. Initial VM creation

- I used the following settings to create the EC2s:
 - For DB VM:
 - AMI:** Ubuntu 22.04 LTS (HVM), SSD volume type
 - Instance type:** t3.micro
 - Key pair:** my AWS key
 - Network:** default VPC & subnet
 - Security rules:**
 - allowed SSH (this is done **by default**)
 - added a new rule** to allow traffic on port 27017 (for MongoDB) from all sources (for testing)



- For app VM:
 - AMI:** Ubuntu 22.04 LTS (HVM), SSD volume type
 - Instance type:** t3.micro
 - Key pair:** my AWS key
 - Network:** default VPC & subnet
 - 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)



2a. 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

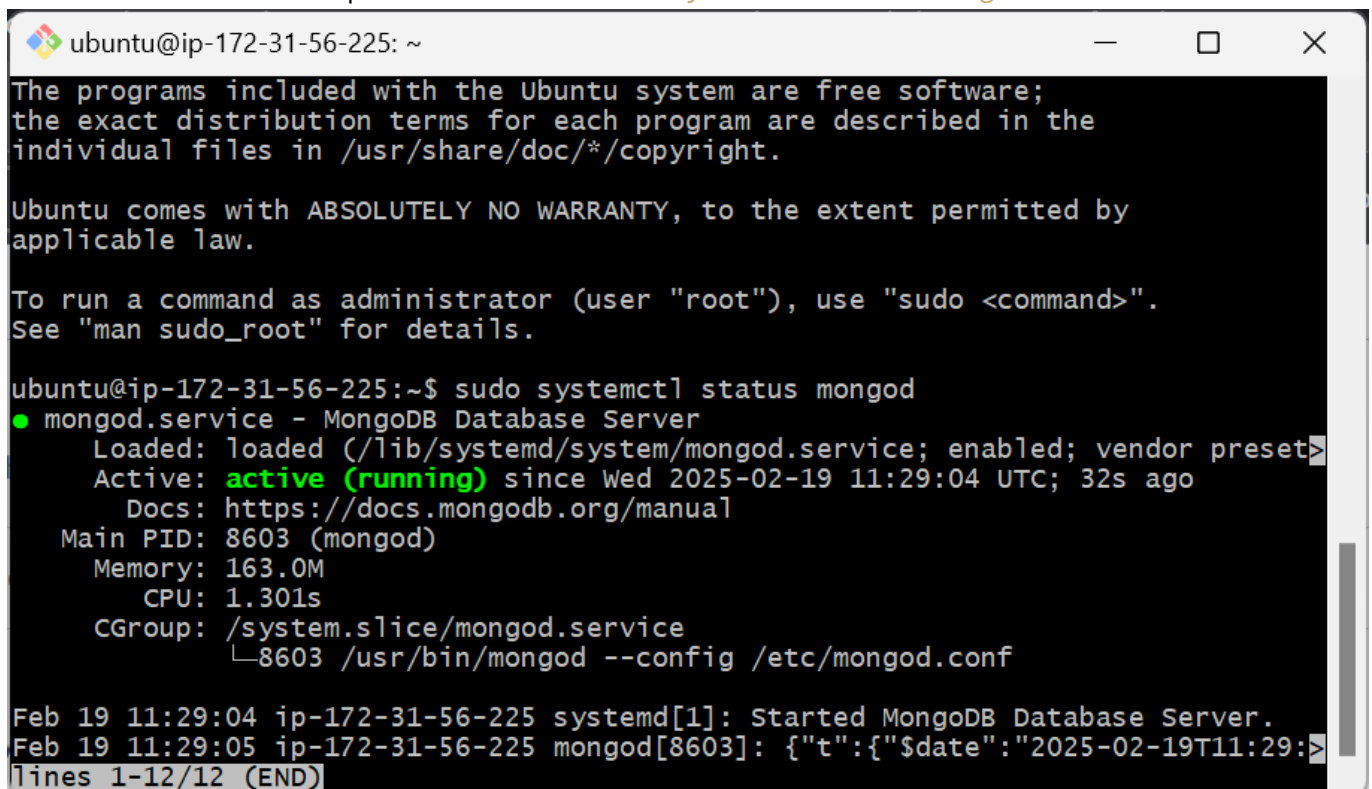
- To avoid the need for any manual configuration, I used the Bash `sed` command to change MongoDB's BindIP setting to 0.0.0.0: `sudo sed -i 's/bindIp: 127.0.0.1/bindIp: 0.0.0.0/' /etc/mongod.conf`
- I later added some commands to the top of this script file that created a log file that would store a log of when each command was being run like:

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

# Redirect output & errors 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 improve security)

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



```
ubuntu@ip-172-31-56-225: ~
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

```

ubuntu@ip-172-31-51-68: ~
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

2b. 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 it does not have a static IP)

- To avoid the need for any manual configuration, I used a Bash `sed` command 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`

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 instance)
[PM2] Done.
```

id	name	mode	u	status	cpu	memory
0	app	fork	0	online	0%	39.3mb

```
ubuntu@ip-172-31-49-161:~$ sh scriptfarahapp.sh
git:1 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
git:2 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease
git:3 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
git:4 https://deb.nodesource.com/node_20.x nodistro InRelease
git:5 http://security.ubuntu.com/ubuntu jammy-security InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
Reading package lists...
Building dependency tree...
Reading state information...
```

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



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 is also used as a standard delimiter
- 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

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

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

- **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.

Instances (1/4) Info

Last updated 4 minutes ago

Launch instances

Launch instance from template

Migrate a server

Connect

Stop instance

Start instance

Reboot instance

Hibernate instance

Terminate (delete) instance

Instance settings

Networking

Security

Image and templates

Monitor and troubleshoot

Find Instance by attribute or tag (case-sensitive)

farah

Clear filters

	Name	Instance ID	Instance type	Status check	Alarm
<input checked="" type="checkbox"/>	tech501-farah...	i-007744230f815d3c5	t3.micro	✔ 3/3 checks passed	View
<input type="checkbox"/>	tech501-farah...	i-03cbfee23d395f4a	t3.micro	✔ 3/3 checks passed	View
<input type="checkbox"/>	tech501-farah...	i-0e9f3db08f5046b	t3.micro	✔ 3/3 checks passed	View

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

Details

Status and alarms

Monitoring

Create image

Create template from instance

Launch more like this

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

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

EC2 > Instances

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

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

Launch instances

Launch instance from template

Migrate a server

Connect

Stop instance

Start instance

Reboot instance

Hibernate instance

Terminate (delete) instance

Instance settings

Networking

Security

Image and templates

Monitor and troubleshoot

Instance state

Actions

Launch instances

Running

Find Instance by attribute or tag (case)

farah

Clear filters

Name

Instance ID

tech501-farah...

i-007744230f815d3c5

tech501-farah...

i-03cbfee23d395f4ab

tech501-farah...

i-0e9f3db08

Instance type

Status check

Alarm status

t3.micro

3/3 checks passed

View alarms +

t3.micro

3/3 checks passed

View alarms +

t3.micro

3/3 checks passed

View alarms +

i-03cbfee23d395f4ab (tech501-farah-udemy-app-from-script-ec2)

Details

Status and alarms

Create image

Create template from instance

Launch more like this

Tags

Amazon Machine Images (AMIs) (1/2)

Info

Recycle Bin

EC2 Image Builder

Actions

Launch instance from AMI

Owned by me

Find AMI by attribute or tag

farah

Clear filters

Source

Owner

Visibility

Status

Creation time

1cf5d948a

135928476890/tech501-farah-udemy-app-from-script

135928476890

Private

Available

2025/06/30 10:00

10e65ff13

135928476890/tech501-farah-udemy-app-from-script

135928476890

Private

Available

2025/06/30 10:00

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 used a *run-app-only.sh* script when creating the app EC2, which overwrites the old connection string and starts the app on this machine when it runs for the first time

3. I tested that the `/posts` page worked and it did

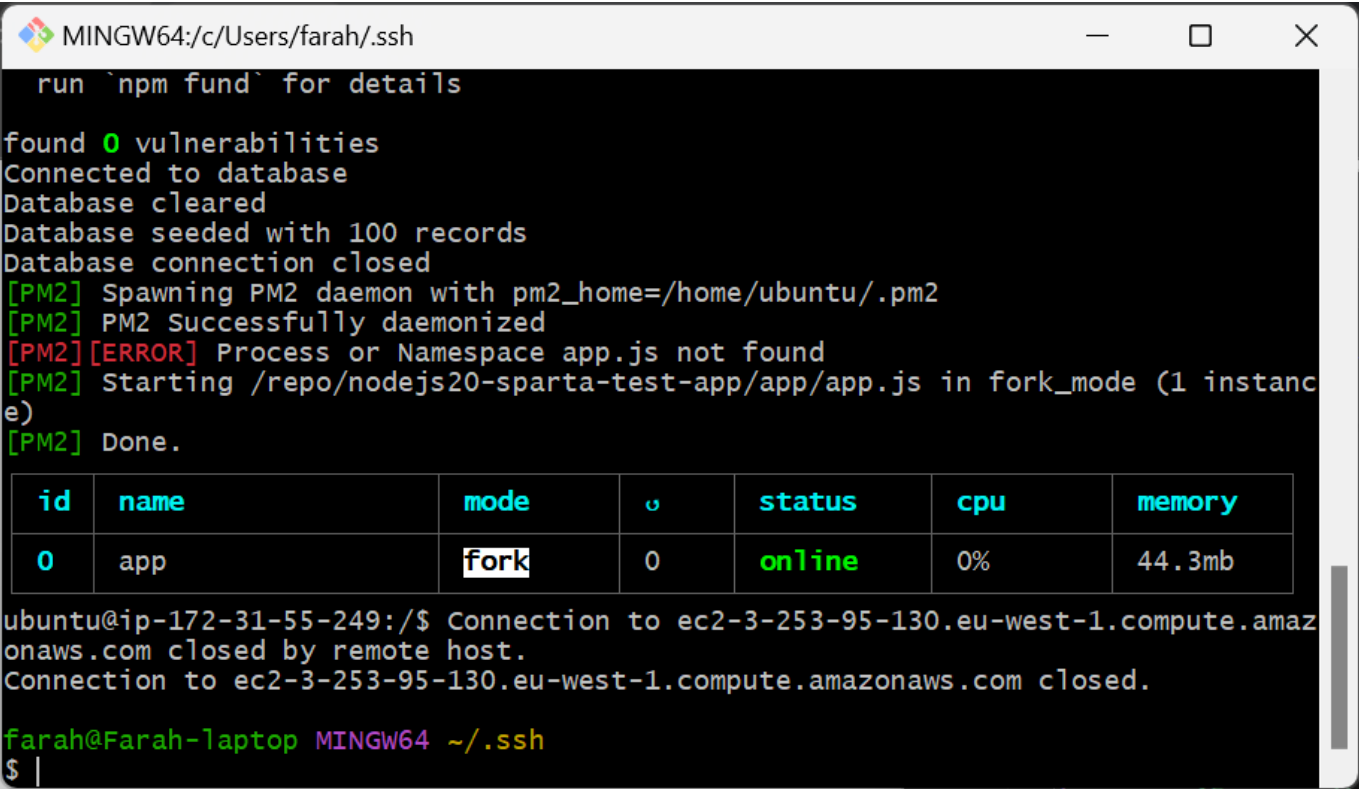


Recent Posts

navigating Licensed Gorgeous

Dignissimos est exercitationem tempora qui id. Cum sequi quas dolor culpa distinctio nihil ipsa quis dolores. Alias sint quo sed doloremque consequatur. Nihil voluptatem ea minus. Libero ex quia sequi eius. Nobis eos ea. Cum maxime ut. Alias atque saepe earum optio atque sint odit debitis incidunt. Eos est vero exercitationem aut dolorem rerum nostrum quos eos. Neque molestias est numquam ut consectetur commodi sed. Sequi at qui molestiae nemo accusantium qui modi. Pariatur autem ut.

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)



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.

- The public IP changes on reboot on AWS (unless we have an elastic IP); this is different from Azure
- 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) 4. Following the same steps for Azure using Terraform

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

- The `.tf` files used for this task can be found [here](#) (with sensitive information hidden in a `git-ignored` variables file)

2. VN created using this method:

tech501-farah-terraform-2-subnet-vnet

Virtual network

Search

Move

Delete

Refresh

Give feedback

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Settings

Monitoring

Automation

Help

Tags

Add tags

Topology

Properties

Capabilities (5)

Recommendations

Tutorials

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:

Microsoft Azure

Search resources, services, and docs (G+/I)

Copilot

fcheded@spartaglobal.c...

SPARTA GLOBAL

Home

Virtual machines

Sparta Global

Create

Switch to classic

Reservations

Manage view

Refresh

Export to CSV

Open query

Assign tags

Start

Restart

Stop

Filter for any field...

Subscription equals all

Type equals all

Resource group equals all

Location equals all

Add filter

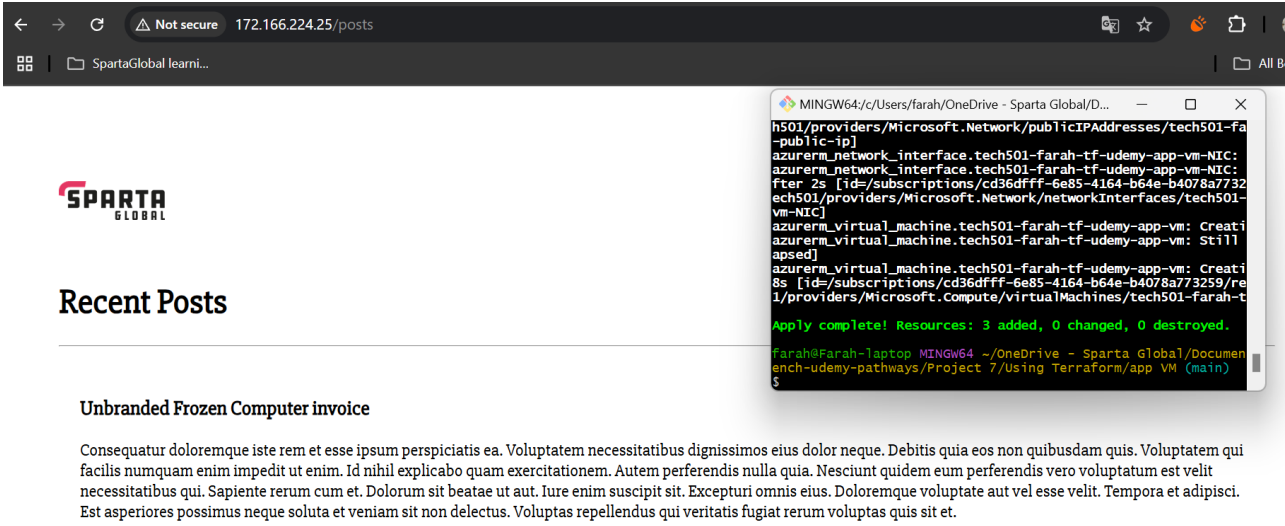
Showing 1 to 2 of 2 records.

No grouping

List view

<input type="checkbox"/>	Name ↑↓	Subscription ↑↓	Resource group ↑↓	Location ↑↓	Status ↑↓	Operating system ↑↓	Size ↑↓	Public IP address ↑↓
<input type="checkbox"/>	tech501-farah-tf-udemy-app...	Azure Training	tech501	UK South	Running	Linux	Standard_B1s	172.166.224.25
<input type="checkbox"/>	tech501-farah-tf-udemy-db...	Azure Training	tech501	UK South	Running	Linux	Standard_B1s	-

- Successful posts page using this method:



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

Create

Manage view

Refresh

Export to CSV

Open query

Assign tags

farah-tf

Subscription equals all

Resource group equals all

Location equals all

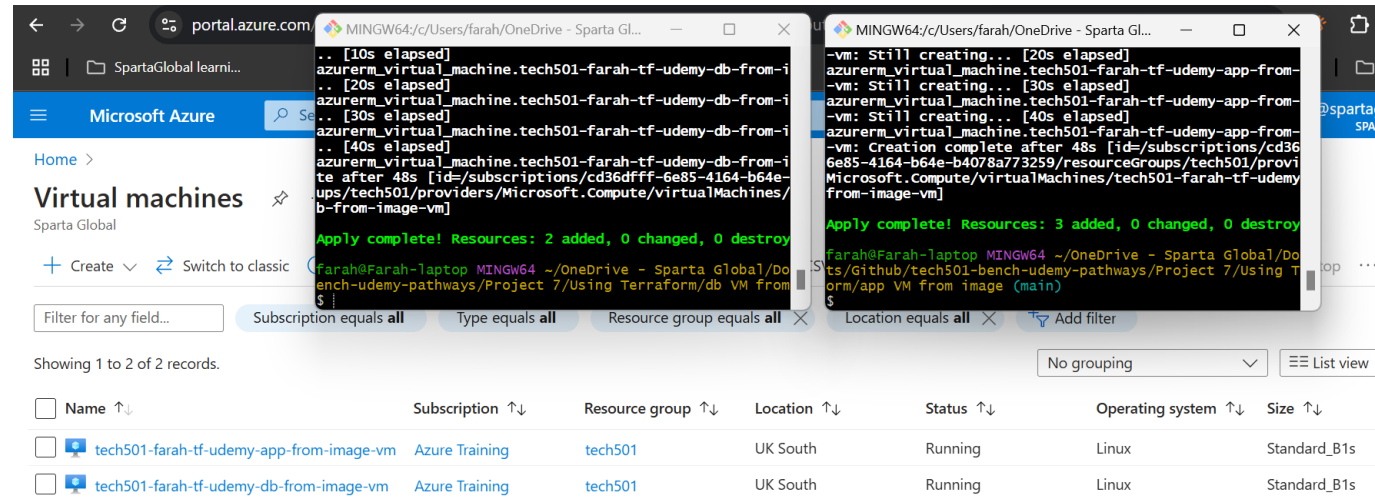
Add filter

Showing 1 to 2 of 2 records.

No grouping

<input type="checkbox"/>	Name	Source...	OS type	Resource group	Location
<input type="checkbox"/>	tech501-farah-tf-udemy-app-vm-image-20250220154933	tech501-far...	Linux	tech501	UK South
<input type="checkbox"/>	tech501-farah-tf-udemy-db-vm-image-20250220155130	tech501-far...	Linux	tech501	UK South

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



6. Finally, I tested these latest VMs by accessing the *posts* page:



Recent Posts

cross-platform

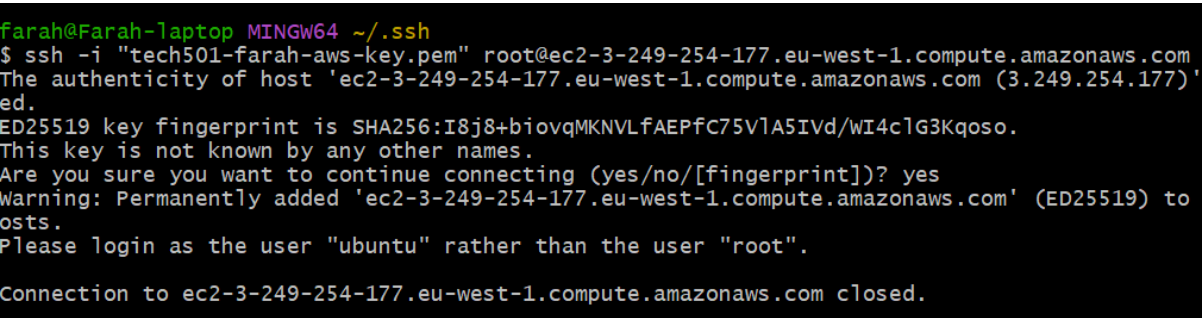
Voluptatem sunt quis consequatur amet aut expedita quibusdam dolore. Laudantium similique repudiandae et qui reprehenderit quibusdam. Beatae dolorum velit aut ea dolorem et at. Possimus in nam quod ut. Ipsum quia aut. Possimus ipsam est rerum saepe voluptatum ducimus ipsa officia. Necessitatibus tenetur 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



- 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 running of the app got
- Because of this, I preferred using Terraform over the cloud providers' GUIs, as I could easily & quickly build and tear down any resources with 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 the IP address used was always the same, even on different iterations of my database VM