

nyt/global//global/global
0lee2003open 0lee2003open 0anderson2015docker 00anderson2015docker 0fielding1997apache 00fielding1997apache
0widenius2002mysql 00widenius2002mysql 0lerdorf2002programming 00lerdorf2002programming
0laravel2022hashing 00laravel2022hashing 0laravel2022blade
00laravel2022blade 0amazon2022what 00amazon2022what
0aws2022ec2 00aws2022ec2 0amazon2022amazon 0ama-
zon2022aws 00amazon2022aws 0ama-
zon2022aws 00amazon2022aws 0dos2018automated
00dos2018automated 53127B6229A7272BCCB1EF6878DAFF2D

urlAvailable at #1

doiAvailable at <https://doi.org/#1>

englishbibliography = References,

isbnAvailable at <https://isbnsearch.org/isbn/#1>

main.bib

Cloud Computing with AWS

By Adam Cordner, Christopher Perry & Evie Snuffle (Group 4)

A report submitted as part of a required module
for the degree of BSc. (Hons.) in Computer Science
at the School of Computing and Digital Technology,
Birmingham City University, United Kingdom

May 2022,
CMP6210 Cloud Computing 2021–2022
Module Coordinator: Dr Khaled Mahbub

Student IDs: 18109958, 18103708, 18128599

Contents

1 Glossary	vi
2 Introduction	1
3 Web App	2
3.1 Software Stack	2
3.2 Database Design	2
3.3 Interface Design	5
4 Virtual Private Cloud (VPC)	8
4.1 Elastic IP Address	8
4.2 VPC and Subnets	8
5 Elastic Cloud Compute (EC2)	13
5.1 AWS Setup	13
5.2 EC2 Login	17
5.3 Package Setup	18
5.4 Web App Setup	19
5.5 systemd Services	21
6 Simple Storage Service (S3)	22
7 CloudFront	23
8 CloudWatch	31
9 CloudTrail	39
10 Relational Database Service (RDS)	40
11 Availability Zones	41
12 Elastic Load Balancing (ELB)	42
13 Security Practices	43
14 Cost Breakdown	44
14.1 Estimated Costs	44
14.2 Scaling Up to 10,000 Users	44
14.3 Scaling Up to One Million Users	44

14.4 Scaling Up to Ten Million Users	44
15 Testing	45
15.1 Testing EC2	45
15.2 Testing S3	47
15.3 Testing CloudFront	48
15.4 Testing RDS	51
15.5 Testing CloudWatch	51
15.6 Testing CloudTrail	51
15.7 Testing ELB	52
16 Future Enhancements	53
17 Conclusion	54
A Additional Screenshots	55
Bibliography	55

List of Figures

3.1	Database tables overview.	2
3.2	table.	3
3.3	table.	3
3.4	table.	4
3.5	<i>Digital-Ink</i> home page log in and sign up forms.	5
3.6	<i>Digital-Ink</i> story creation form.	6
3.7	<i>Digital-Ink</i> account page.	7
3.8	<i>Digital-Ink</i> stories page and story view.	7
4.1	Elastic IP addresses page.	8
4.2	Configuring the Elastic IP address.	9
4.3	Created Elastic IP address.	9
4.4	VPC Management Console.	9
4.5	Selecting a VPC configuration.	10
4.6	Configuring VPC public and private subnets.	11
4.7	Generating the VPC.	11
4.8	VPC successfully created.	11
4.9	Your VPCs page.	12
5.1	Selection of EC2 OS Image.	13
5.2	Selection of EC2 Instance.	14
5.3	Selection of EC2 Keypair.	14
5.4	Selection of EC2 Networking options.	15
5.5	Generated EC2 Keypair in the format.	15
5.6	Selection of EC2 Storage Configuration.	16
5.7	SSH command to log into EC2 instance.	17
5.8	Logging into EC2 instance.	17
5.9	Installing Git.	18
5.10	Installing Docker.	18
5.11	Cloning the web app from Github.	19
5.12	Containers required for the web app being pulled from Docker Hub.	19
5.13	Creation of tables through command.	20

5.14 Digital Ink shown when accessed through the IPv4 address.	20
5.15 Creation of Service.	21
7.1 Applying CloudFront bucket policy.	24
7.2 Applying CloudFront origins to S3 bucket.	24
7.3 Applying CloudFront Distribution Permissions.	25
7.4 Applying CloudFront Cache Keys and Origin Requests.	25
7.5 Function Association.	26
7.6 Applying CloudFront Settings.	27
7.7 Applying CloudFront Settings.	27
7.8 Created CloudFront Distribution.	28
7.9 Image location before CloudFront.	28
7.10 Image location after CloudFront.	29
7.11 Image location after CloudFront.	30
8.1 Selection of CloudWatch metric for EC2 instance.	31
8.2 Configuration of NetworkPacketsOut Metric.	32
8.3 Configuration of NetworkPacketsOut Metric.	32
8.4 Configuration of SNS Topic for email alerts on alarm activation.	33
8.5 CloudWatch SNS Topic Email.	33
8.6 Successful subscription to SNS topic.	33
8.7 Configuration for EC2 instance to reboot on alarm activation.	34
8.8 CloudWatch alarm description.	34
8.9 CloudWatch network alarm in dashboard.	35
8.10 Selection of CloudWatch metric.	35
8.11 Configuration of CloudWatch alarm.	36
8.12 Configuration of CloudWatch alarm.	36
8.13 Setting CloudWatch charges SNS topic	37
8.14 Description of CloudWatch alarm.	37
8.15 CloudWatch charges alarm live in CloudWatch dashboard.	38
15.1 Digital Ink image whilst connected to UK IP.	49
15.2 UK IP Location.	49
15.3 Digital Ink image whilst connected to US IP.	50
15.4 US IP Location.	50

Chapter 1

Glossary

- **ACL** — Access Control List
- **ACM** — AWS Certificate Manager
- **AMI** — Amazon Machine Images
- **App** — Application
- **AWS** — Amazon Web Services
- **AZ** — Availability Zone
- **CA** — Certificate Authority
- **CDS** — Content Delivery Network
- **CIDR** — Classless Inter-Domain Routing
- **CPU** — Central Processing Unit
- **DB** — Database
- **DIG** — Domain Information Groper
- **EC2** — Elastic Cloud Compute
- **ELB** — Elastic Load Balancing
- **.env** — Environment File Extension
- **GB** — Gigabyte
- **HTTP** — HyperText Transfer Protocol
- **HTTPS** — HyperText Transfer Protocol Secure
- **IAM** — Identity and Access Management
- **IEEE** — Institute of Electrical and Electronics Engineers
- **IP** — Internet Protocol
- **IPv4** — Internet Protocol Version 4

- **IPv6** — Internet Protocol Version 6
- **LAMP** — Linux, Apache, MySQL, PHP
- **ML** — Machine Learning
- **NAT** — Network Address Translation
- **nslookup** — Name Server Lookup
- **OS** — Operating System
- **.pem** — Privacy Enhanced Mail File Extension
- **PHP** — PHP: Hypertext Preprocessor
- **RAM** — Random Access Memory
- **S3** — Simple Storage Service
- **RAM** — Random Access Memory
- **RDBMS** — Relational Database Management System
- **RDS** — Relational Database Service
- **SaaS** — Software as a Service
- **SNS** — Simple Notification Service
- **SQL** — Structured Query Language
- **SSH** — Secure Shell
- **VPC** — Virtual Private Cloud
- **VPN** — Virtual Private Network
- **WAF** — Web Application Firewall

Chapter 2

Introduction

This report details the process of designing, developing, and deploying a cloud application onto Amazon Web Services (AWS). The application is called *Digital Ink* and allows users to create, edit, and delete their own short stories. Users can then view their own short stories and other users' short stories. It was first developed locally using a LAMP stack. This consisted of Linux - hosted through Docker - for the operating system, an Apache HTTP Server, MySQL for the relational database management, and PHP as the programming language.

After the application was built locally, it was gradually integrated onto AWS. This involved implementing several AWS cloud features to enhance the application, ensure application security, and increase availability. This was accomplished by using Simple Storage Service (S3), Elastic Compute Cloud (EC2), ELB (Elastic Load Balancing), and more. The process of implementing these cloud features will be discussed throughout the report.

After the application was integrated onto AWS, an evaluation of the process was conducted. This includes a discussion of the security practices used, estimated costs for different user scales, and thorough testing. Lastly, several enhancements which could be made to the application in the future will be discussed.

Chapter 3

Web App

This chapter of the report will detail the local design and development of the *Digital Ink* web application. We will first discuss the software stack used to develop the app, then the design of the database used, and, lastly, the design of the user interface.

3.1 Software Stack

Digital Ink was first developed locally using a LAMP stack. LAMP refers to a generic software stack, where each letter in the acronym stands for one the following open source building blocks: Linux, Apache HTTP Server, MySQL, and PHP lee2003open. The web app is hosted within a Docker container anderson2015docker which runs a minified version of the Linux operating system. Apache is an open-source web server software which is used to host the app on the web fielding1997apache. MySQL is an open-source relational database management system widenius2002mysql which is used to store all the data used within the app, including user details and story details. PHP is a programming language aimed towards web development, chosen due to its stability and reliability ler-dorf2002programming. Additionally, all developers involved have prior experience with PHP.

3.2 Database Design

As mentioned before, the web app uses the MySQL relational database management system to store its data. MySQL is a relational database management system (RDBMS) which stores data in the form of tables, where Structures Query Language (SQL) is used to access the database. As shown in Figure 3.1, the database which this web app uses consists of three tables:

,

, and

.

The

table (see Figure 3.2) contains records which correspond to the migrations within the

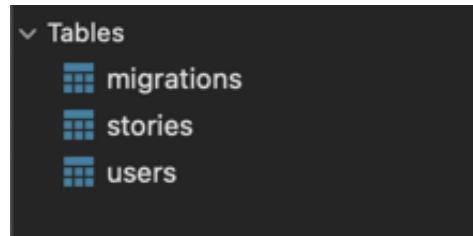


Figure 3.1: Database tables overview.

Laravel web app. These migrations contain the scripts required to automatically generate the

and

tables in SQL. It contains the following three columns:

- : the unique ID for each migration.
- : points to the scripts used to create tables.
- : how many times the script has been ran.

id	migration	batch
1	2014_10_12_000000_create_users_table	1
4	2020_03_13_105916_create_stories_table	1

Figure 3.2:
table.

The

table (see Figure 3.3) contains all the information about user accounts, and it contains the following seven columns:

- : the unique ID for each user account.
- : the name associated with user account.
- : the unique email used to log in.
- : the password used to log in, encrypted with 184 bit hashing by Bcrypt laravel2022hashing.
- : keeps the user logged in if they select "Remember me".
- : records what date and time the user account was first created at.
- : records what date and time the user account was last updated at.

MySQL 8.0.28 : TL5v1.2 : digital-ink : group4 : users

stories							users		
id	name	email	password	remember_token	created_at	updated_at			
1	Adam	adam.cordner@mail.bcu.ac.uk	\$2y\$10\$ISS.3eqO3o0dRXaK8sMy.eBqUapgD...	mDUIGRilVhyEi6...	2022-05-10 15:50:27	2022-05-10 15:50:27	2022-05-10 15:50:27	2022-05-10 15:50:27	
2	evie	^~^@snugg.ie	\$2y\$10\$vVNRMN/lRSIYTrCUjJEPu44jkVI0r5...	Mu1WdyOz1e8...	2022-05-10 15:51:29	2022-05-10 15:51:29	2022-05-10 15:51:29	2022-05-10 15:51:29	

Figure 3.3:
table.

The

table (see Figure 3.4) contains all the information about user-created stories, and it contains the following 11 columns:

- : the unique ID for each story.
- : the unique ID associated with the user who created the story.
- : the title associated with the story.
- : the genre associated with the story, which can be one of eight different genres.
- : a brief description of the story.
- : the full content of the story.
- : a thumbnail image for the story.
- : an optional PDF upload of the story.
- : 1 if the story has been made public, or 0 if it is a draft.
- : records what date and time the story was first created at.
- : records what date and time the story was last updated at.

[width=]resources/database/stories-records-1
[width=]resources/database/stories-records-2

Figure 3.4:
table.

3.3 Interface Design

The design of the web app was created using Blade, a powerful templating engine laravel2022blade. When the user initially accesses the web app, they are able to log in or sign up. This can be seen in Figure 3.5. When a user has created an account, a record is written to the table in the database.

```
[width=]resources/webapp/digital-ink-home  
[width=]resources/webapp/digital-ink-sign-up
```

Figure 3.5: *Digital-Ink* home page log in and sign up forms.

Once a user is signed in, they can create a story. Creating a story requires the user to enter a title, a genre, the story itself, a blurb, and, optionally, a thumbnail image. This can be seen in Figure 3.6. Once a story has been created, it is written to the table.

```
[width=]resources/webapp/digital-ink-create-story-1  
[width=]resources/webapp/digital-ink-create-story-2  
[width=]resources/webapp/digital-ink-create-story-3
```

Figure 3.6: *Digital-Ink* story creation form.

After this, the user can see all of their uploaded stories on their account page. This can be seen in Figure 3.7. From here, a story can be edited or deleted, which either updates a record in the table or removes a record from it.

Hi Adam!

The screenshot shows a user interface for managing published stories. At the top, a green banner displays the message "Yay! Your story has been published!". Below this, the text "Here are your published stories:" is centered. A table follows, with a single row visible. The table has three columns: "TITLE", "GENRE", and "ACTIONS". The "TITLE" column contains "Group 4 Loves AWS", the "GENRE" column contains "Romance", and the "ACTIONS" column contains two buttons: "Edit" (highlighted with a yellow circle) and "Delete" (highlighted with a red circle).

TITLE	GENRE	ACTIONS
Group 4 Loves AWS	Romance	Edit Delete

Figure 3.7: *Digital-Ink* account page.

Lastly, on the Stories page, a user can view and search through all uploaded stories across all users. Each story's title, genre, and blurb is shown in a list view. A user can click into one of these stories to see the thumbnail image and read the full story. These pages can be seen in Figure 3.8.

[width=]resources/webapp/digital-ink-stories [width=]resources/webapp/digital-ink-story

Figure 3.8: *Digital-Ink* stories page and story view.

Chapter 4

Virtual Private Cloud (VPC)

Amazon VPC allows for AWS resources to be launched in a virtual network that has been custom defined and configured. This virtual network is similar to a traditional network which operates within your own physical data center, with the added benefits of the scalable AWS infrastructure. A VPC can have multiple assigned subnets, which are a range of IP addresses accessible in the VPC. This chapter of the report will detail the creation of an Elastic IP address, a VPC, and the necessary subnets.

4.1 Elastic IP Address

Before creating the VPC, an Elastic IP address must be created, which can then be allocated to the VPC. To do this, the "Allocate Elastic IP address" button on the Elastic IP addresses page must be clicked.

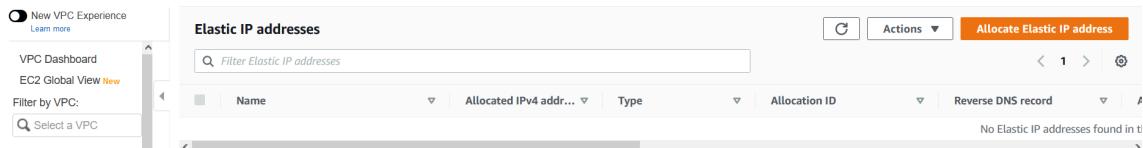


Figure 4.1: Elastic IP addresses page.

To configure the Elastic IP address, a Network Border Group must be selected, ensuring that this is the same as the Availability Zone selected when configuring the VPC. Once this has been set, and the remaining options are set to their default values, clicking the Allocate button will create the Elastic IP address.

After this, the Elastic IP address created can be seen from the Elastic IP addresses page.

4.2 VPC and Subnets

The first step of migrating the web app to the cloud was creating a VPC to deploy and manage AWS resources for the app. To do this, the VPC wizard must be used. This is accessed by navigating to the VPC Management Console and then click the Launch VPC Wizard button, which can be seen in Figure 4.4.

The wizard presents us with step one of creating a VPC: selecting a VPC configuration. There are several options for this, where each configuration has the following features:

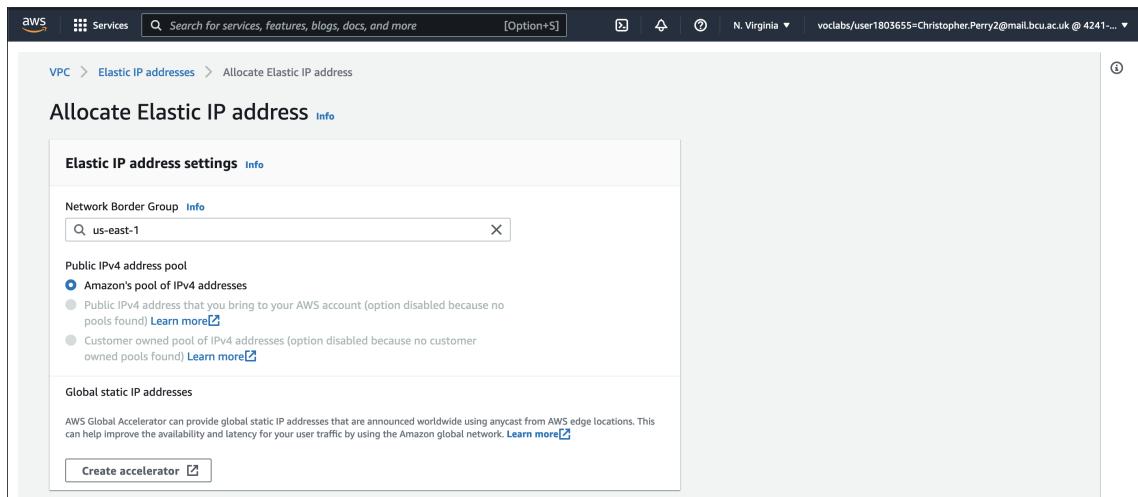


Figure 4.2: Configuring the Elastic IP address.

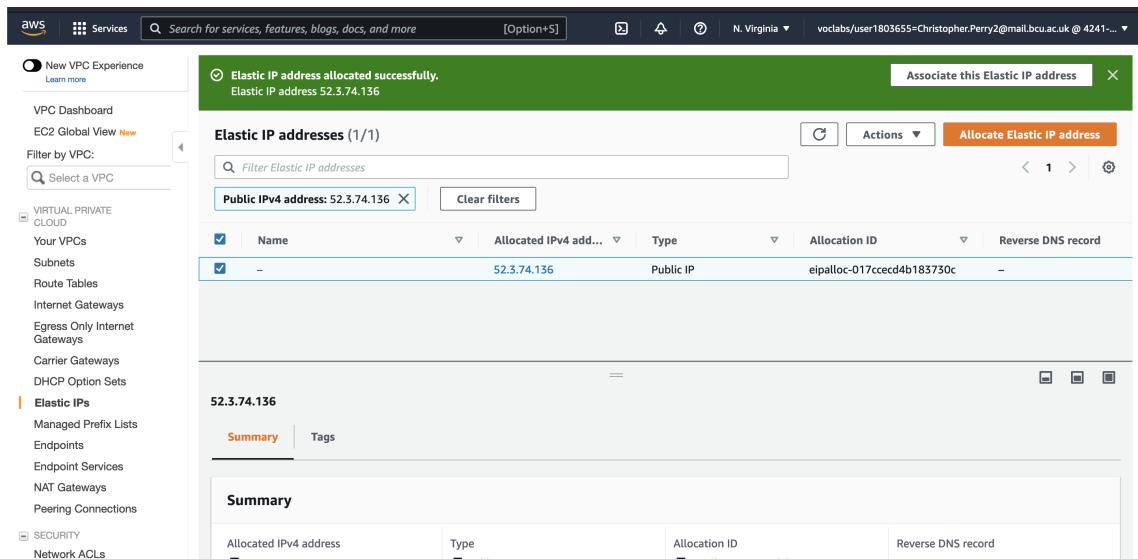


Figure 4.3: Created Elastic IP address.

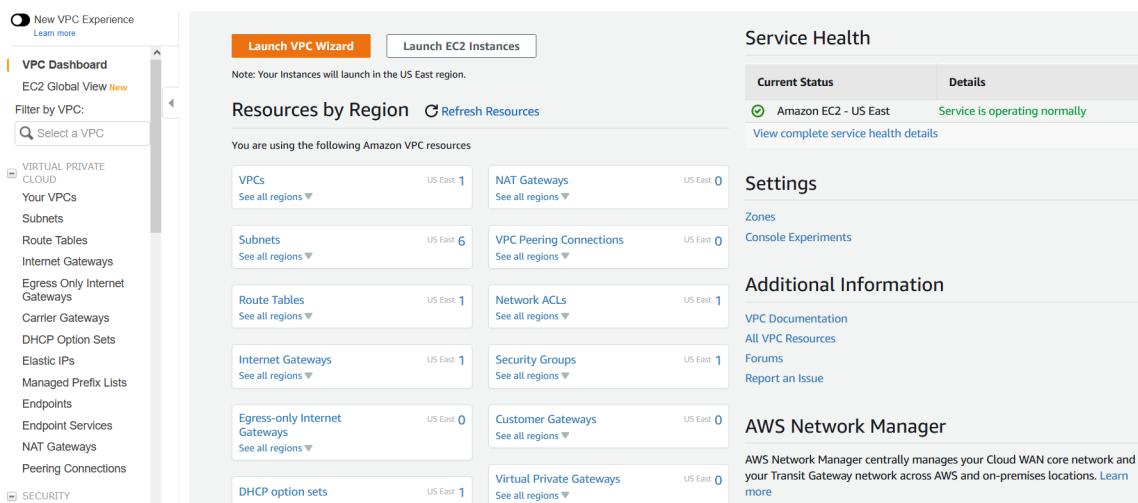


Figure 4.4: VPC Management Console.

- VPC with a Single Public Subnet: Creates a /16 network with a /24 subnet where the subnet instances use Elastic IPs or Public IPs for internet access.
- VPC with Public and Private Subnets: Creates a /16 network and two /24 subnets where the public subnet instances use Elastic IPs for internet access and the private subnet instances use NAT for internet access.
- VPC with Public and Private Subnets and Hardware VPN Access: Creates a /16 network with two /24 subnets where one subnet is connected to the internet and another is connected to your personal network via a VPN.
- VPC with a Private Subnet Only and Hardware VPN Access: Creates a /16 network and a /24 subnet where the subnet is connected to your personal network via a VPN.

For the deployment of this web app, hardware VPN access is not necessary, so the VPC configuration with public and private subnets was selected, as seen in Figure 4.5.

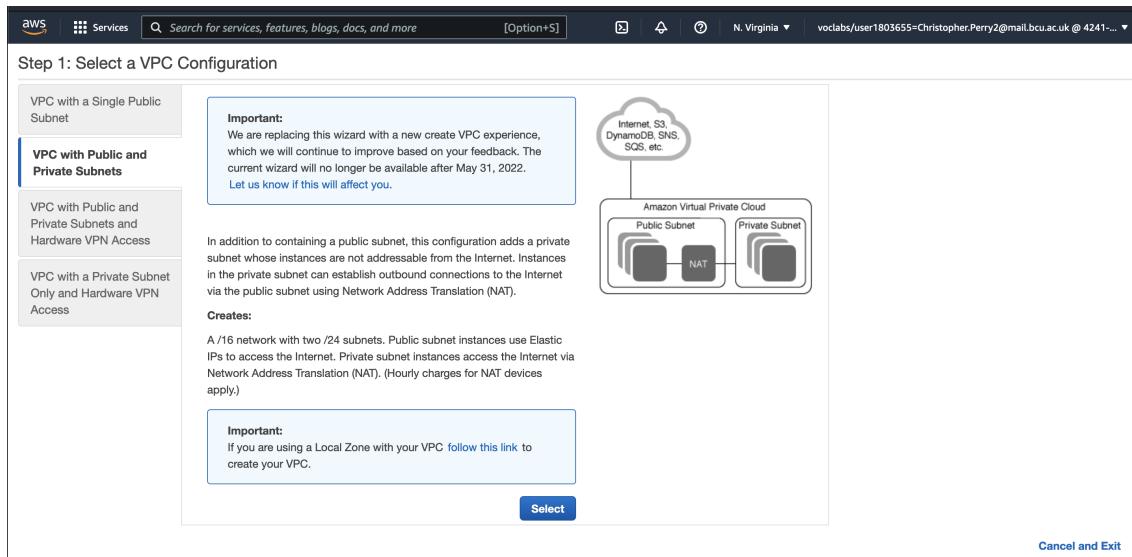


Figure 4.5: Selecting a VPC configuration.

Following this, the specific details, such as the IP address block, of the selected configuration must be entered. An IPv4 CIDR block of

was chosen as it provides a large amount of available IP addresses. Additionally, no IPv6 CIDR block was configured and the VPC was named

Next, the public and private subnets were configured. The public subnet was assigned an IPv4 CIDR of

, making 251 distinct IP addresses available for the public subnet. The availability zone of

was chosen and the subnet was named

. The private subnet was assigned an IPv4 CIDR of

, making 251 distinct IP addresses available for the private subnet as well. The availability zone of

was chosen and the subnet was named

Lastly, the previously created Elastic IP Allocation ID was selected, and the remaining settings were left at their default values. Clicking the Create VPC button now will generate the VPC with the specified configurations.

The screenshot shows the 'Step 2: VPC with Public and Private Subnets' configuration page. It includes fields for IPv4 and IPv6 CIDR blocks, VPC name ('Group4_VPC'), and two subnets: 'Public subnet 1' and 'Private subnet 1'. Each subnet has its own CIDR range (10.0.0.0/24 and 10.0.1.0/24 respectively), availability zone (us-east-1a), and an 'Add Endpoint' button for service endpoints. There are also sections for NAT gateway details, including an 'Elastic IP Allocation ID' (eipalloc-017cccd4b183730c) and options for enabling DNS hostnames and hardware tenancy.

Figure 4.6: Configuring VPC public and private subnets.

After this, AWS takes a few minutes to generate the VPC.

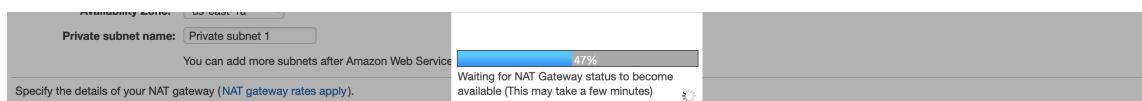


Figure 4.7: Generating the VPC.

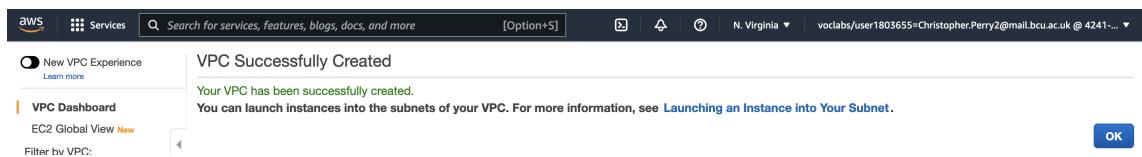


Figure 4.8: VPC successfully created.

Navigating from here to the "Your VPCs" page shows two available VPCs: the VPC created by the AWS sandbox environment, which will not be used, and the VPC which has just been created, named zsh|Group4_VPC|.

The screenshot shows the AWS VPC Dashboard. On the left, there's a sidebar with options like New VPC Experience, VPC Dashboard, EC2 Global View, Filter by VPC, and a list of services including 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, and Network ACLs. The main area displays 'Your VPCs (2) Info' with a table:

Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR
-	vpc-006b19b59ed484041	Available	172.31.0.0/16	-
Group4_VPC	vpc-0b0472507c8bf18c9	Available	10.0.0.0/16	-

A modal window for 'vpc-07657585bc0e3b3b5' is open, showing the 'Details' tab with the following information:

VPC ID vpc-07657585bc0e3b3b5	State Available	DNS hostnames Disabled	DNS resolution Enabled
---------------------------------	--------------------	---------------------------	---------------------------

Figure 4.9: Your VPCs page.

Chapter 5

Elastic Cloud Compute (EC2)

5.1 AWS Setup

After the VPC and subnets were configured, the initial deployment of the web app began with setting up EC2. This AWS service allows for scalable computing capacity through the use of a virtual computing environment hosted in the cloud aws2022ec2. The web app will be stored on an EC2 instance of Amazon Linux, known as Amazon Machine Image (AMI), which will then be launched through a docker container stored on the app.

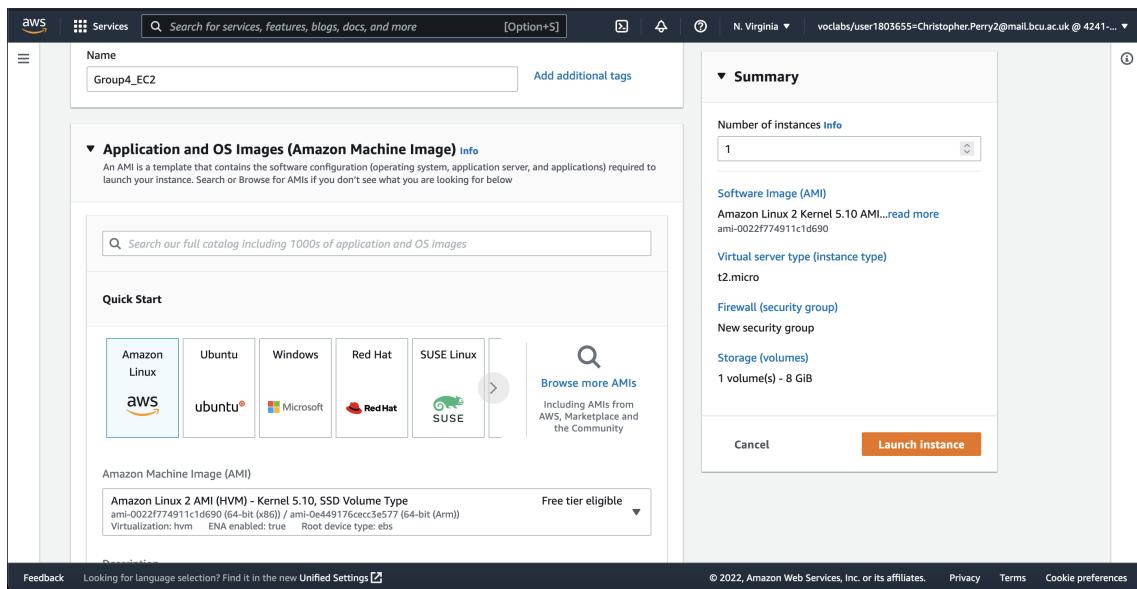


Figure 5.1: Selection of EC2 OS Image.

Figure 5.1 details the selection of the Operating System (OS) that will be used for the EC2 instance. The *Amazon Linux 2 AMI* was selected, as it is already configured with Linux and does not need any more setup.

Now that an AMI has been chosen, the specific instance type that will be used within this AMI can be selected. It was decided that the instance type of *t2.micro* would be used, as it contains only 1GB of Random Access Memory (RAM), as seen in Figure 5.2.

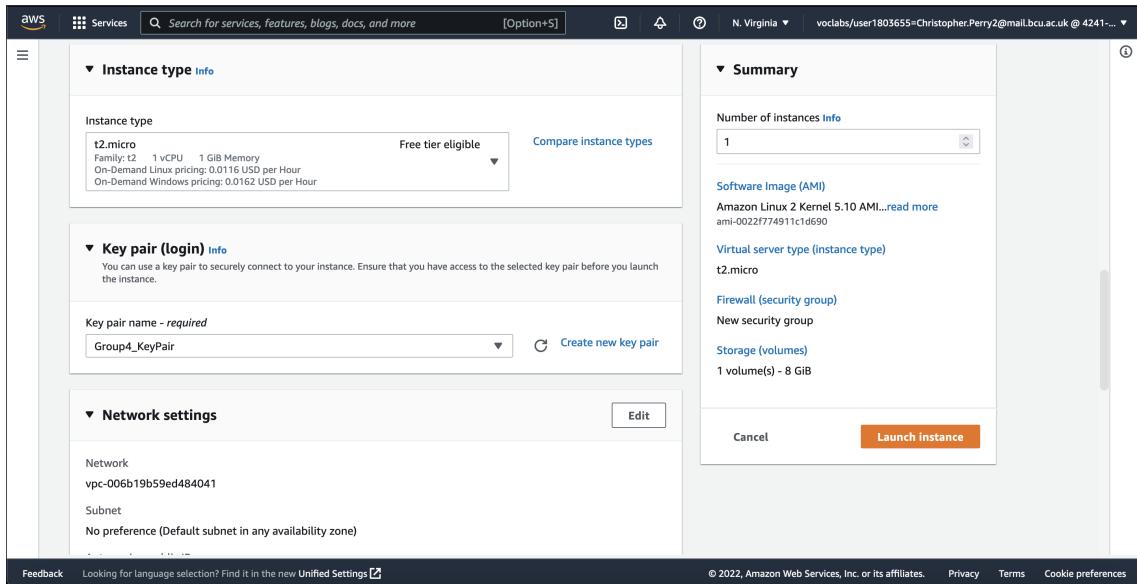


Figure 5.2: Selection of EC2 Instance.

A key pair will allow for the ability to sign in to the EC2 instance with a unique set of login credentials, heightening the security of the project.

The next stage of the setup process was to set up networking for the EC2 instance, in order for the web app to work with Docker to download relevant containers from Docker-Hub, which will allow a Laravel instance to be initialised, as discussed in Section 3. This process can be seen in Figure 5.3.

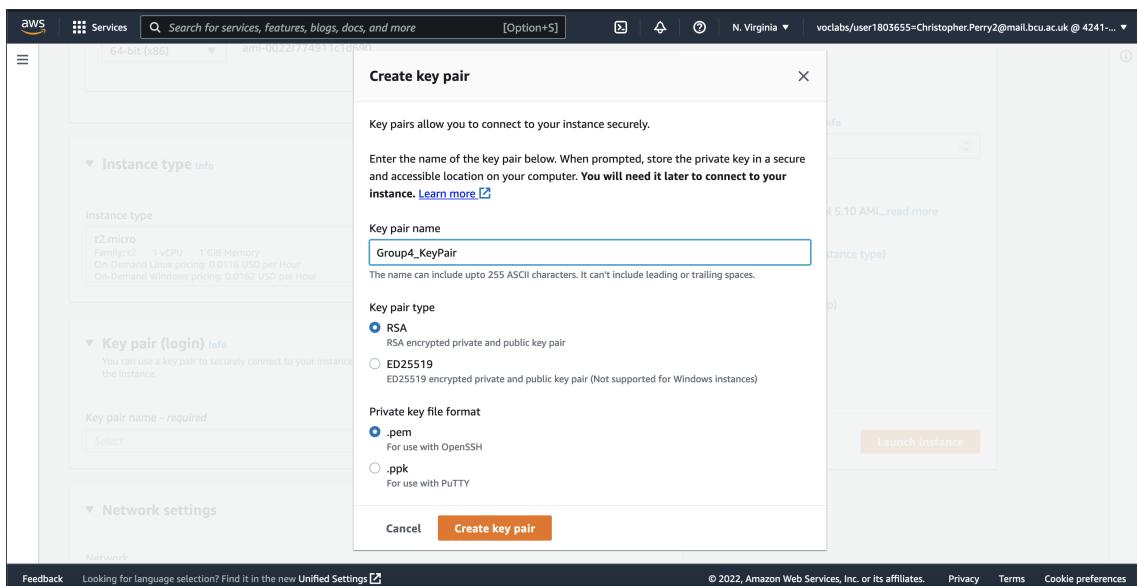


Figure 5.3: Selection of EC2 Keypair.

The instance is assigned the VPC created in Section ??, where it is assigned a subnet in the same availability zone of

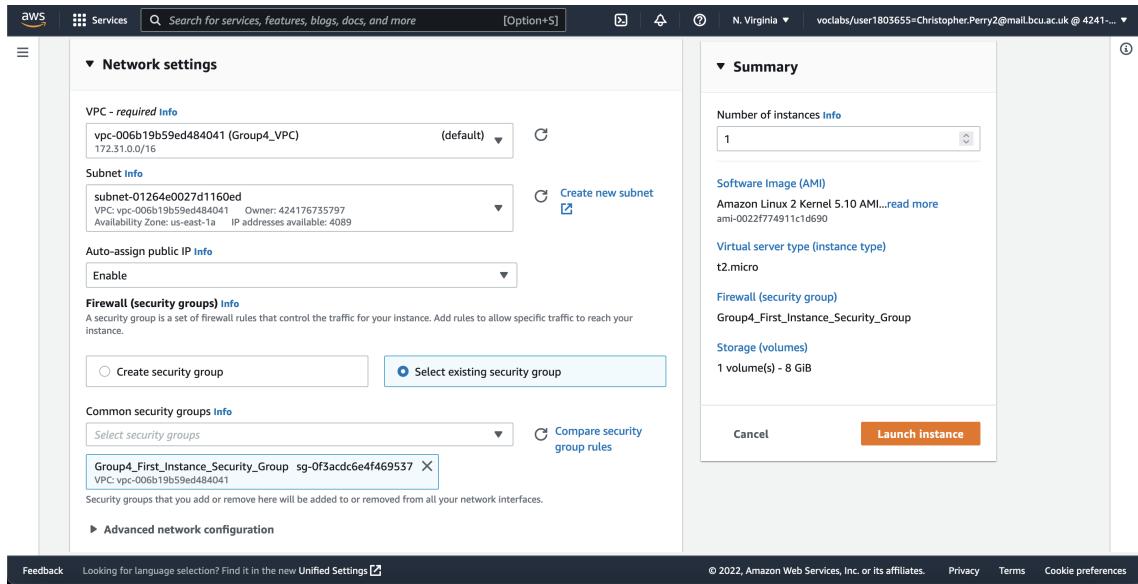


Figure 5.4: Selection of EC2 Networking options.

```
chris@Christophers-MacBook-Pro:~/Desktop/Keypair
└─ ssh -i Group4_KeyPair.pem ec2-user@52.45.13.111
```

The terminal window shows the command: ssh -i Group4_KeyPair.pem ec2-user@52.45.13.111. The prompt indicates the user is at 17:47:52.

Figure 5.5: Generated EC2 Keypair in the format.

This setup can be seen in Figure 5.4. An EC2 keypair is then generated in the format.

This is enough to comfortably run the web app without any issues. Storage for the AMI was subsequently chosen. It was decided that 8GB of storage would be used, as this is enough to run the web app and still provide leftover storage for any system-critical tasks.

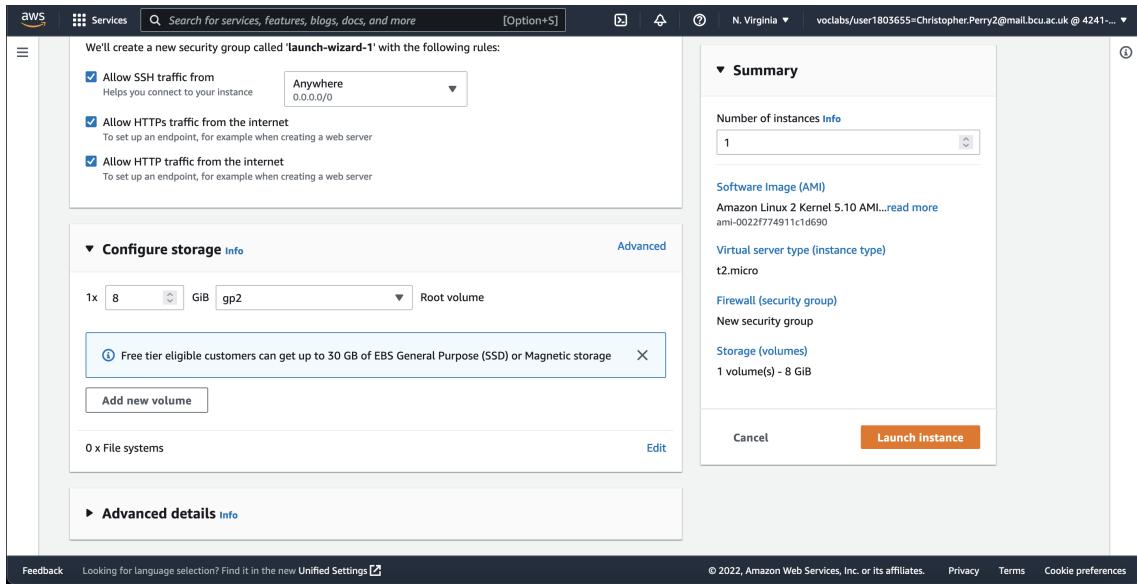


Figure 5.6: Selection of EC2 Storage Configuration.

The selection of these options can be found in Figure 5.6. In addition to this, the chosen options are eligible for "Free Tier", which means that it will use a limited amount of the \$100 budget allocated for the project.

5.2 EC2 Login

The EC2 instance

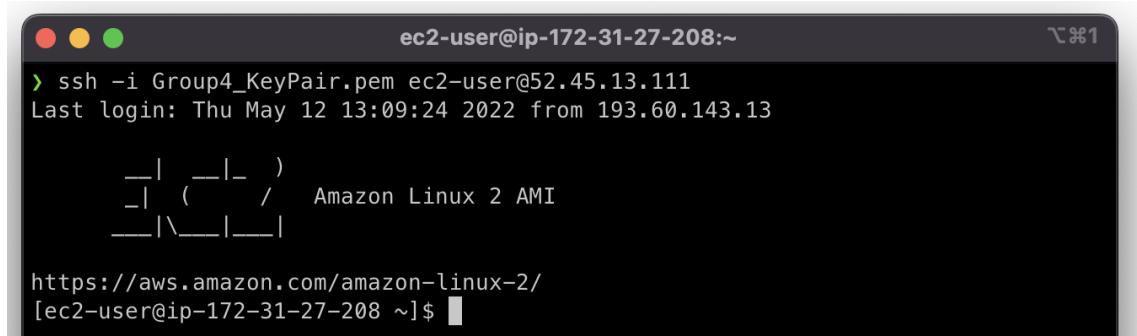
is now live, and the webapp can be loaded onto it. The instance is first logged in to through the use of the

command, followed by the

argument to specify an identify file, which was generated earlier, and then the public ipv4 address of the instance.

Figure 5.7: SSH command to log into EC2 instance.

This command can seen being executed in Figure 5.7.



```
ec2-user@ip-172-31-27-208:~$ ssh -i Group4_KeyPair.pem ec2-user@52.45.13.111
Last login: Thu May 12 13:09:24 2022 from 193.60.143.13
[ec2-user@ip-172-31-27-208 ~]$
```

The screenshot shows a terminal window with a dark background and light-colored text. At the top, it displays the session information: 'ec2-user@ip-172-31-27-208:~'. Below this, a command is entered: 'ssh -i Group4_KeyPair.pem ec2-user@52.45.13.111'. The response shows the last login details: 'Last login: Thu May 12 13:09:24 2022 from 193.60.143.13'. Following this, there is a stylized line-art logo consisting of vertical and horizontal lines forming a tree-like or bracketed shape. Below the logo, the URL 'https://aws.amazon.com/amazon-linux-2/' is displayed. At the bottom of the terminal window, the prompt '[ec2-user@ip-172-31-27-208 ~]\$' is visible.

Figure 5.8: Logging into EC2 instance.

The logged in EC2 instance can be seen in Figure 5.8.

5.3 Package Setup

The web app is stored on GitHub, and the AMI does not come with GitHub by default. Git is subsequently installed via

```

git-core           x86_64      2.32.0-1.amzn2.0.1          amzn2-core          4.8 M
git-core-doc     noarch      2.32.0-1.amzn2.0.1          amzn2-core          2.7 M
perl-Error        noarch      1:0.17020-2.amzn2          amzn2-core          32 k
perl-Git          noarch      2.32.0-1.amzn2.0.1          amzn2-core          43 k
perl-TermReadKey x86_64      2.30-20.amzn2.0.2          amzn2-core          31 k

Transaction Summary
=====
Install 1 Package (+6 Dependent packages)

Total download size: 7.8 M
Installed size: 38 M
Is this ok [y/d/N]: y
Downloading packages:
(1/7): emacs-filesystem-27.2-4.amzn2.0.1.noarch.rpm | 67 KB 00:00:00
(2/7): git-2.32.0-1.amzn2.0.1.x86_64.rpm | 126 KB 00:00:00
(3/7): git-core-doc-2.32.0-1.amzn2.0.1.noarch.rpm | 2.7 MB 00:00:00
(4/7): perl-Error-0.17020-2.amzn2.noarch.rpm | 32 KB 00:00:00
(5/7): perl-Git-2.32.0-1.amzn2.0.1.noarch.rpm | 43 KB 00:00:00
(6/7): git-core-2.32.0-1.amzn2.0.1.x86_64.rpm | 4.8 MB 00:00:00
(7/7): perl-TermReadKey-2.30-20.amzn2.0.2.x86_64.rpm | 31 KB 00:00:00

Total                                         29 MB/s | 7.8 MB 00:00:00

Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
Installing : git-core-2.32.0-1.amzn2.0.1.x86_64
Installing : git-core-doc-2.32.0-1.amzn2.0.1.noarch
Installing : perl-Error-0.17020-2.amzn2.noarch
Installing : 1:emacs-filesystem-27.2-4.amzn2.0.1.noarch
Installing : perl-TermReadKey-2.30-20.amzn2.0.2.x86_64
Installing : perl-Git-2.32.0-1.amzn2.0.1.noarch
Installing : git-2.32.0-1.amzn2.0.1.x86_64
Verifying  : perl-TermReadKey-2.30-20.amzn2.0.2.x86_64
Verifying  : git-core-doc-2.32.0-1.amzn2.0.1.noarch
Verifying  : perl-Git-2.32.0-1.amzn2.0.1.noarch
Verifying  : 1:emacs-filesystem-27.2-4.amzn2.0.1.noarch
Verifying  : git-2.32.0-1.amzn2.0.1.x86_64
Verifying  : git-core-2.32.0-1.amzn2.0.1.x86_64
Verifying  : perl-Error-0.17020-2.amzn2.noarch

Installed:
  git.x86_64 0:2.32.0-1.amzn2.0.1

Dependency Installed:
  emacs-filesystem.noarch 0:27.2-4.amzn2.0.1  git-core.x86_64 0:2.32.0-1.amzn2.0.1  git-core-doc.noarch 0:2.32.0-1.amzn2.0.1  perl-Error.noarch 1:0.17020-2.amzn2  perl-Git.noarch 0:2.32.0-1.amzn2.0.1
  perl-TermReadKey.x86_64 0:2.30-20.amzn2.0.2

Complete!
[ec2-user@ip-172-31-27-208 ~] |

```

Figure 5.9: Installing Git.

The web app also requires docker, and

is executed to install Docker as a result.

```

[ec2-user@ip-172-31-27-208 ~]$ sudo yum update
Loaded plugins: extras_suggestions, longpacks, priorities, update-motd
amzn2-core
No packages marked for update
[ec2-user@ip-172-31-27-208 ~]$ sudo yum install docker docker-compose
Loaded plugins: extras_suggestions, longpacks, priorities, update-motd
No package docker-compose available.
Resolving Dependencies
--> Running transaction check
--> Package docker.x86_64 0:20.10.13-2.amzn2 will be installed
--> Processing Dependency: libselinux >= 2.8.1-5.15 for package: docker-20.10.13-2.amzn2.x86_64
--> Processing Dependency: containerd >= 1.3.2 for package: docker-20.10.13-2.amzn2.x86_64
--> Processing Dependency: pigz for package: docker-20.10.13-2.amzn2.x86_64
--> Running transaction check
--> Package containerd.x86_64 0:1.4.13-2.amzn2.0.1 will be installed
--> Package libgroup.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 0:1.0.3-2.amzn2 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package      Arch      Version       Repository      Size
=====
Installing: docker      x86_64    20.10.13-2.amzn2      amzn2extra-docker   40 M
Installing for dependency:
  containerd   x86_64    1.4.13-2.amzn2.0.1      amzn2extra-docker   23 M
  libgroup    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    1.0.3-2.amzn2          amzn2extra-docker   3.0 M

Transaction Summary
=====
Install 1 Package (+4 Dependent packages)

Total download size: 67 M
Installed size: 280 M
Is this ok [y/d/N]: y

```

Figure 5.10: Installing Docker.

5.4 Web App Setup

The web app is firstly cloned from its repository via the command, and a new folder is made to store the contents.

```
[ec2-user@ip-172-31-27-208 ~]$ git clone https://github.com/ChrisP99/digital-ink.git  
-bash: git: command not found  
[ec2-user@ip-172-31-27-208 ~]$ git clone https://github.com/ChrisP99/digital-ink.git  
Cloning into 'digital-ink'...  
remote: Enumerating objects: 959, done.  
remote: Counting objects: 100% (959/959), done.  
remote: Compressing objects: 100% (324/324), done.  
remote: Total 959 (delta 593), reused 959 (delta 593), pack-reused 0  
Receiving objects: 100% (959/959), 3.32 MiB | 19.01 MiB/s, done.  
Resolving deltas: 100% (593/593), done.  
[ec2-user@ip-172-31-27-208 ~]$
```

Figure 5.11: Cloning the web app from Github.

The command is used to move into the folder, and the web app is subsequently launched through the to launch the web app as a detached Docker container. Relevant containers that are required to be downloaded from the are then pulled.

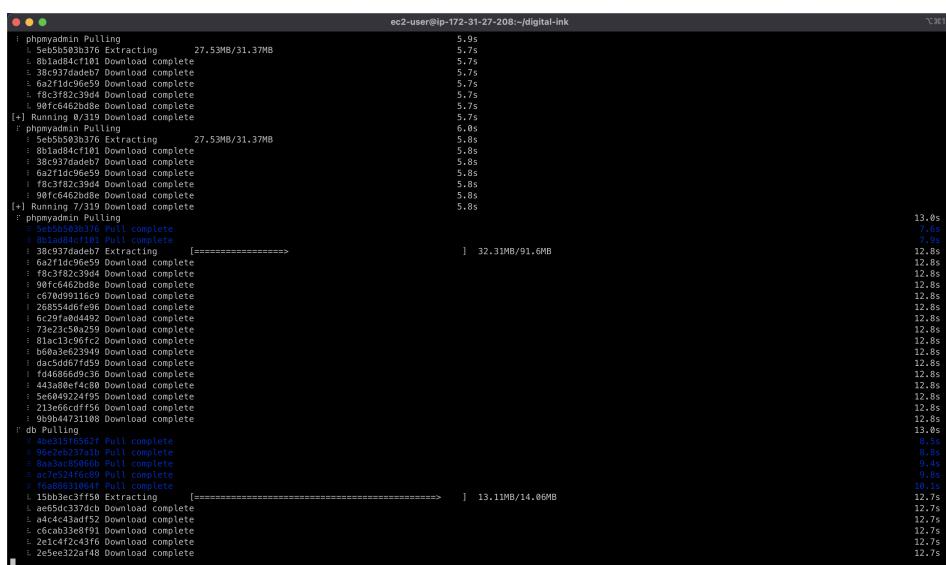


Figure 5.12: Containers required for the web app being pulled from Docker Hub.

The result of this command launches 3 containers:

1. : An instance of the web app which uses a custom Laravel container.
2. : An instance of a local database made in MySQL.
3. : A way to locally manage the database through a UI.

At the minute, the web app is live through the container, and is using a local version of MySQL as a database, stored within the Docker container. The database has no tables, but can be populated through the use of Laravel. The container is firstly accessed through , and the database is populated with tables with . This then generates tables to store users and their stories.

```
[ec2-user@ip-172-31-27-208 digital-ink]$ sudo /usr/local/bin/docker-compose exec app bash
root@dd0e13abddf12:/srv/app# php artisan migrate
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated: 2014_10_12_000000_create_users_table (0.08 seconds)
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated: 2014_10_12_100000_create_password_resets_table (0.07 seconds)
Migrating: 2019_08_19_000000_create_failed_jobs_table
Migrated: 2019_08_19_000000_create_failed_jobs_table (0.04 seconds)
Migrating: 2020_03_13_105916_create_stories_table
Migrated: 2020_03_13_105916_create_stories_table (0.16 seconds)
root@dd0e13abddf12:/srv/app#
```

Figure 5.13: Creation of tables through command.

The subsequent output of this command can be found in Figure 5.13. When the website is accessed at the public IPv4 address at , Digital Ink will now be shown.

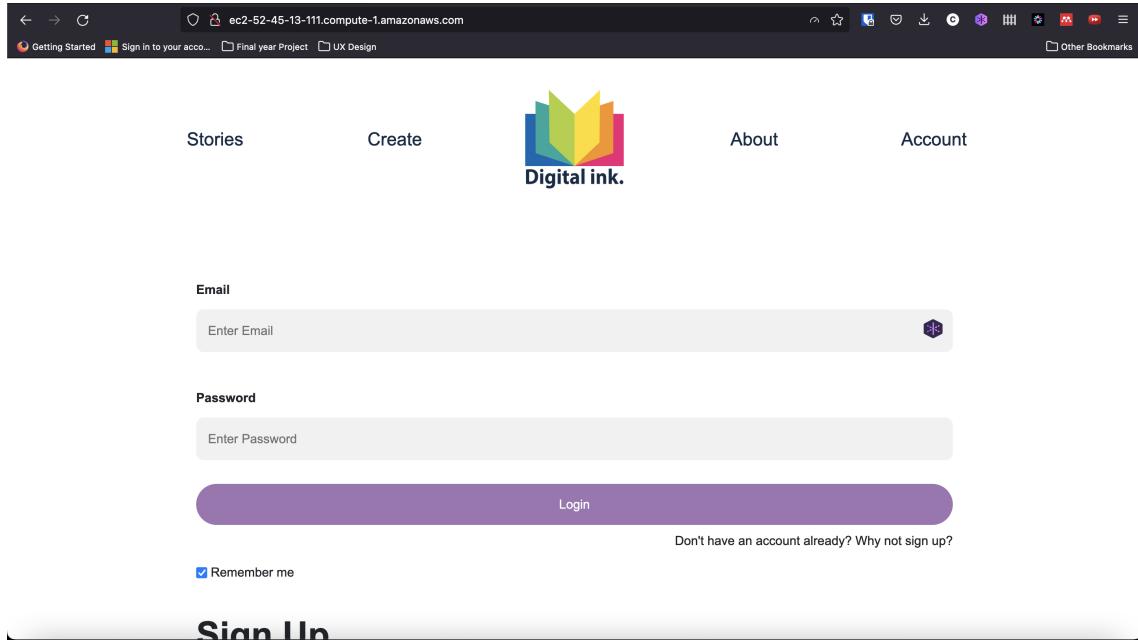
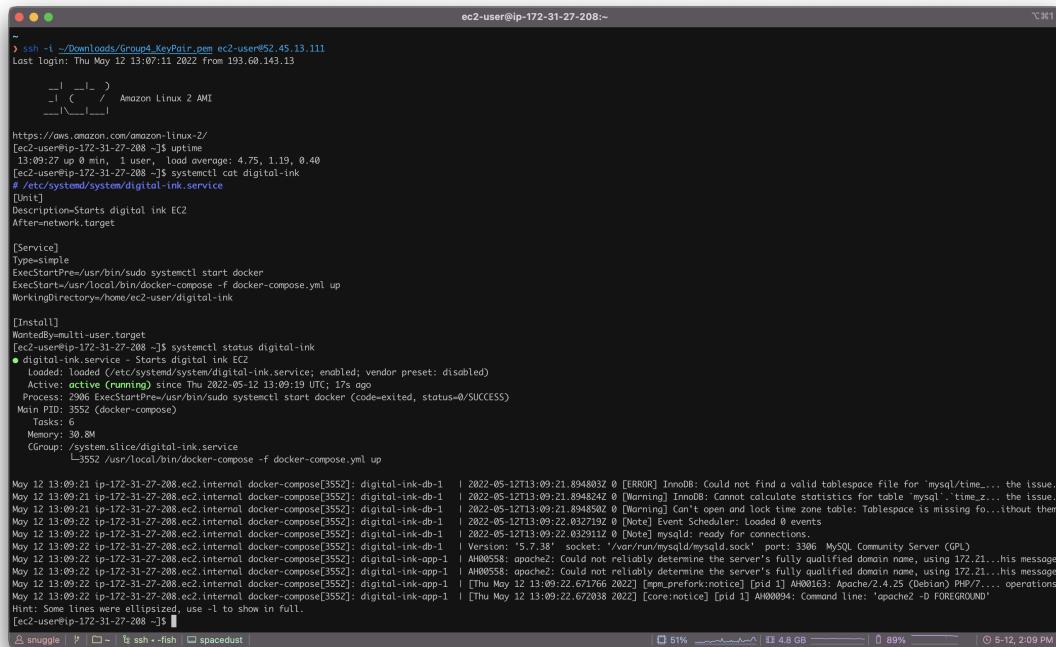


Figure 5.14: Digital Ink shown when accessed through the IPv4 address.

5.5 systemd Services

is a service manager, which is a program that launches and monitors different services across the system. The digital-ink application is automatically started upon boot by the systemd process, which ensures that the application is started correctly and without errors. If any errors occur, systemd will log them in a tool known as

and then follow a configuration file for instructions on how to handle those errors. This can range from simply restarting a program that has errored, to trying once again, or recording the failure and performing no action.



The screenshot shows a terminal window titled 'ec2-user@ip-172-31-27-208:~'. The user is creating a new service file named 'digital-ink.service' in the directory '/etc/systemd/system'. The service file contains the following configuration:

```
[Unit]
Description=Starts digital ink EC2
After=network.target

[Service]
Type=simple
ExecStartPre=/usr/bin/sudo systemctl start docker
ExecStart=/usr/local/bin/docker-compose -f docker-compose.yml up
WorkingDirectory=/home/ec2-user/digital-ink

[Install]
WantedBy=multi-user.target
```

After saving the file, the user runs 'systemctl status digital-ink' to verify its status. The output shows the service is active (running) and ready to accept connections.

```
[ec2-user@ip-172-31-27-208 ~]$ systemctl status digital-ink
● digital-ink.service - Starts digital ink EC2
   Loaded: loaded (/etc/systemd/system/digital-ink.service; enabled; vendor preset: disabled)
     Active: active (running) since Thu 2022-05-12 13:09:19 UTC; 17s ago
       Main PID: 3552 (docker-compose)
          Tasks: 6
         Memory: 30.8M
            CPU: 0.000 CPU(s) used (avg: 0.000 ms)
           CGroup: /system.slice/digital-ink.service
                   └─ 3552 /usr/local/bin/docker-compose -f docker-compose.yml up

May 12 13:09:21 ip-172-31-27-208.ec2.internal docker-compose[3552]: digital-ink-db-1 | 2022-05-12T13:09:21.894803Z 0 [ERR][OK] InnoDB: Could not find a valid tablespace file for 'mysql/time.... the issue.
May 12 13:09:21 ip-172-31-27-208.ec2.internal docker-compose[3552]: digital-ink-db-1 | 2022-05-12T13:09:21.894842Z 0 [Warning] InnoDB: Comm calculate statistics for table 'mysql'.time_z... the issue.
May 12 13:09:21 ip-172-31-27-208.ec2.internal docker-compose[3552]: digital-ink-db-1 | 2022-05-12T13:09:21.894850Z 0 [Warning] Can't open and lock zone table: Tablespace is missing fo...ithout them
May 12 13:09:21 ip-172-31-27-208.ec2.internal docker-compose[3552]: digital-ink-db-1 | 2022-05-12T13:09:22.032019Z [Note] Event Scheduler: Loaded 0 events
May 12 13:09:22 ip-172-31-27-208.ec2.internal docker-compose[3552]: digital-ink-db-1 | 2022-05-12T13:09:22.032019Z [Note] Event Scheduler: Enabled
May 12 13:09:22 ip-172-31-27-208.ec2.internal docker-compose[3552]: digital-ink-db-1 | Version: '5.7.38' socket: '/var/run/mysqld/mysqld.sock' port: 3306 MySQL Community Server (GPL)
May 12 13:09:22 ip-172-31-27-208.ec2.internal docker-compose[3552]: digital-ink-app-1 | AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.21....his message
May 12 13:09:22 ip-172-31-27-208.ec2.internal docker-compose[3552]: digital-ink-app-1 | AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.21....his message
May 12 13:09:22 ip-172-31-27-208.ec2.internal docker-compose[3552]: digital-ink-app-1 | [Thu May 12 13:09:22.671766 2022] [mpm_prefork:notice] [pid 1] AH00163: Apache/2.4.25 (Debian) PHP/7.0.37 operations
May 12 13:09:22 ip-172-31-27-208.ec2.internal docker-compose[3552]: digital-ink-app-1 | [Thu May 12 13:09:22.672038 2022] [core:notice] [pid 1] AH00094: Command line: 'apache2 -D FOREGROUND'
Hint: Some lines were ellipsized, use -l to show in full.
[ec2-user@ip-172-31-27-208 ~]$
```

The terminal window also shows the user is using 'snugle' as their shell, and the system status bar indicates battery at 51%, 4.8 GB RAM, 89% disk usage, and the date/time as 5-12, 2:09 PM.

Figure 5.15: Creation of Service.

Chapter 6

Simple Storage Service (S3)

AWS S3 is a form of cloud-based object storage. This is a style of network filesystem that treats each file as a separate object with a unique ID, which allows each object to be served individually over a network with a single URL, which can also be enhanced with a content delivery network.

Each S3 instance is separated into logical containers known as buckets. Each S3 bucket can have its own credentials, its own endpoint and other permissions and configuration. Object storage is particularly useful when creating an application that scales as you are billed per unit of storage that you use, and in theory are able to use infinite storage as your application demands. The only limitation with object storage is how much you are able to pay for. Each object is automatically replicated across many different nodes, providing data redundancy against multiple different availability zones.

S3 is perhaps the most popular AWS service and used by many different SaaS applications across the internet, for instance Instagram, Facebook, Discord and Twitter are all known for using S3 or S3-style storage. The advent of object storage has created an almost de-facto standard, which has lead for the creation of many 'S3-compatible' or 'S3-like' competitor solutions, such as those run by Google Cloud and Microsoft Azure.

Chapter 7

CloudFront

CloudFront will allow for the distribution of static and dynamic web images more efficiently. An international network of data centres, referred to as edge locations, are used to deliver content for CloudFront. An edge location which offers the lowest latency or delay in serving these files will be used. This will involve the creation of a CloudFront Distribution.

Firstly, an origin is chosen which is the S3 bucket created in Section 6. It is then given a name of

. The "Yes use OAI" option is selected, in order to restrict bucket access to only CloudFront. The "Bucket Policy" option is set to "Yes" to automatically update permissions on the bucket to allow read access for the OAI. These options can be seen configured in Figures 7.1 and 7.2.

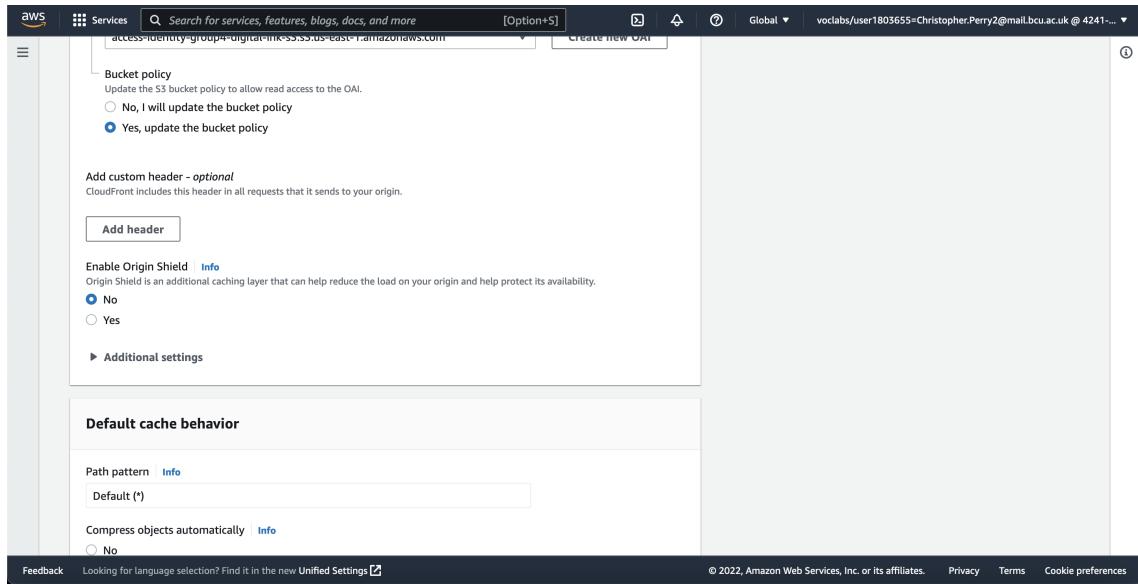


Figure 7.1: Applying CloudFront bucket policy.

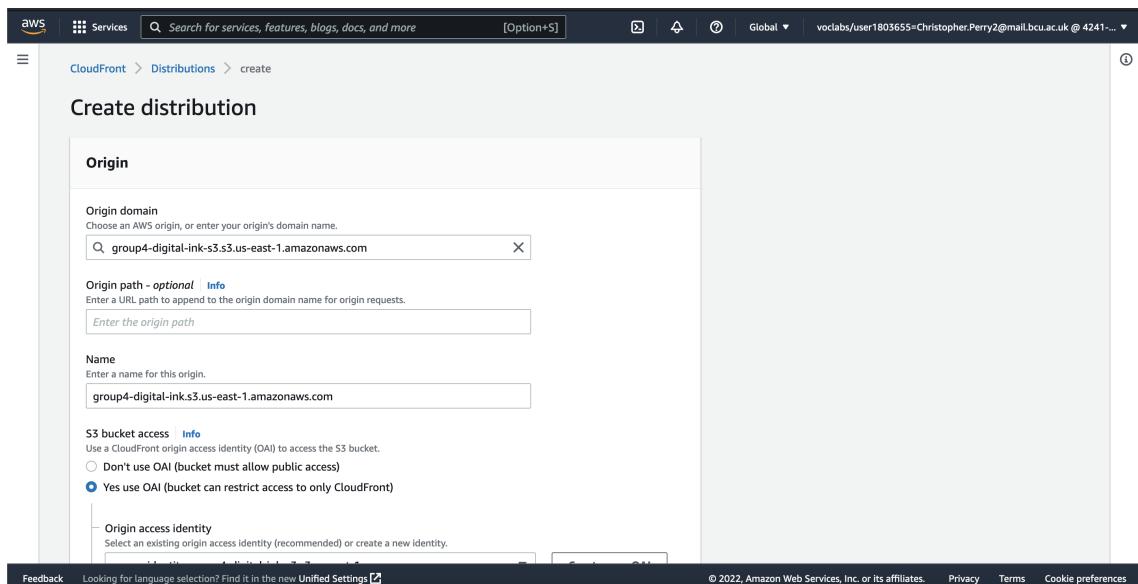


Figure 7.2: Applying CloudFront origins to S3 bucket.

Permissions are then applied to the CloudFront distribution itself. Objects are set to be compressed automatically to save space, and for viewing images, permissions are set to be in both HTTP and HTTPS, and to only allow

and

, so images can only be viewed.

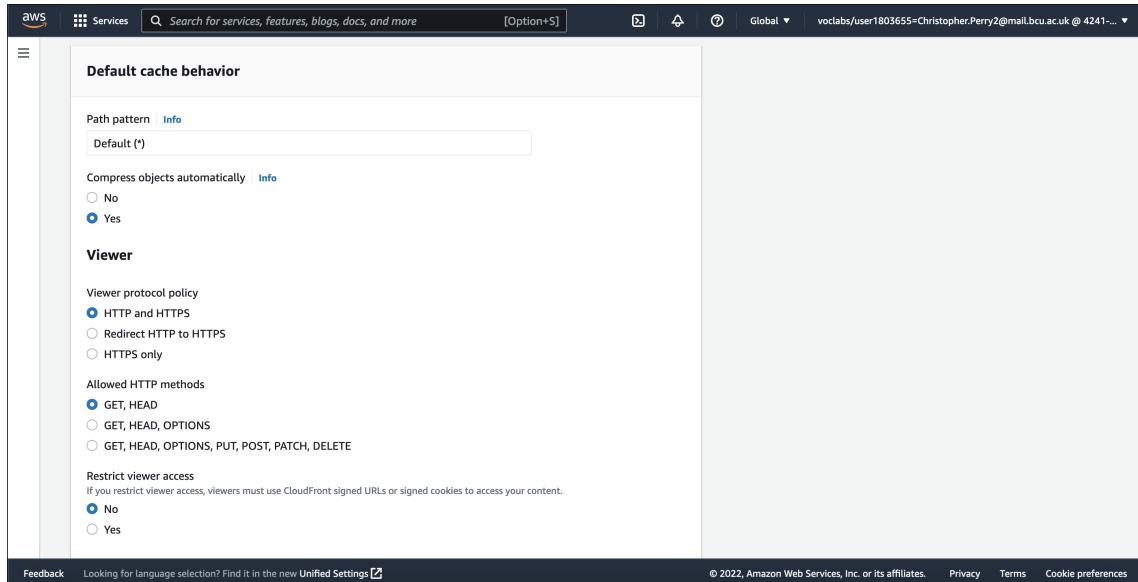


Figure 7.3: Applying CloudFront Distribution Permissions.

A cache policy and origin request policy is subsequently applied to the aforementioned permissions. The "CachingOptimized" policy is applied for compression. Origin Request policy is set to "CORS-S3Origin" in order for CORS to be automatically set up for the S3 bucket, Any GET requests are then compatible in the response headers through "SimpleCORS".

No function associations are set due to the lack of CloudFront functions and Lambdas.

The settings for the CloudFront Distribution can now be set. In order to ensure high availability, the option "Use all edge locations" is selected, so that images can be distributed to the user from the closest edge location in relation to their IP address. No ACL can be added to the WAF due to permissions issues, and a SSL Certificate cannot be set as the web app is not being served from the internet. For logging purposes "Standard Logging" is set to "On", and logs are saved to the same S3 bucket. Cookie logging is enabled, and IPv6 is set to "On", to allow for more IP addresses to access the CloudFront Distributed content. These settings can be found in Figures ?? and 7.7.

The screenshot shows the 'Cache key and origin requests' section of the AWS CloudFront configuration interface. It includes fields for selecting a cache policy (set to 'CachingOptimized'), an origin request policy ('CORS-S3Origin'), and response headers ('SimpleCORS'). A link to 'Additional settings' is also present.

Figure 7.4: Applying CloudFront Cache Keys and Origin Requests.

The screenshot shows the 'Function associations - optional' section of the AWS CloudFront configuration interface. It lists four categories: 'Viewer request', 'Viewer response', 'Origin request', and 'Origin response', each with a dropdown menu set to 'No association'. Below this is a 'Settings' section containing options for 'Price class' (set to 'Use all edge locations (best performance)'), 'AWS WAF web ACL' (dropdown menu set to 'Choose web ACL'), and a note about language selection.

Figure 7.5: Function Association.

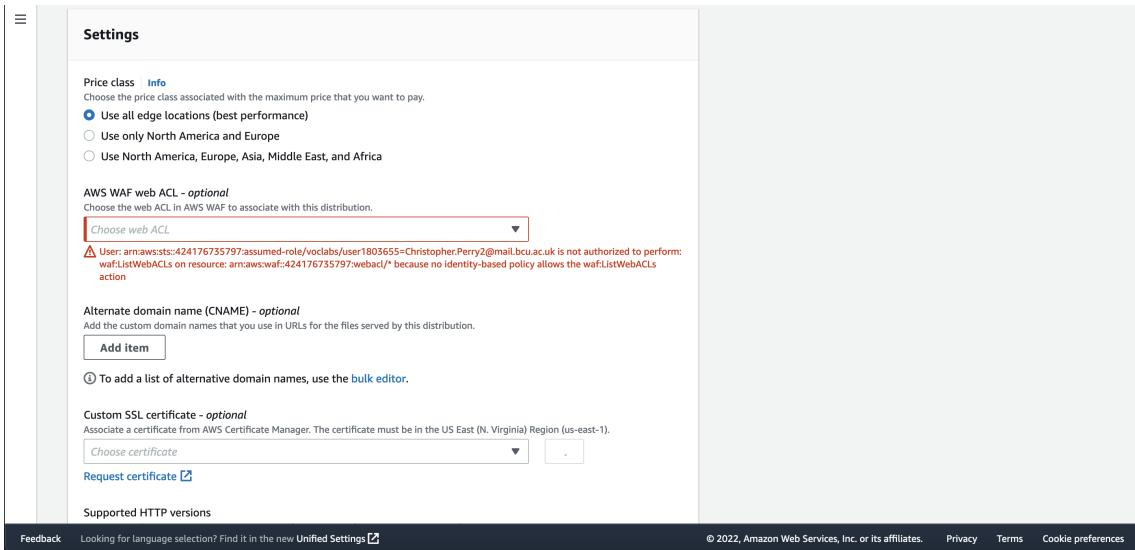


Figure 7.6: Applying CloudFront Settings.

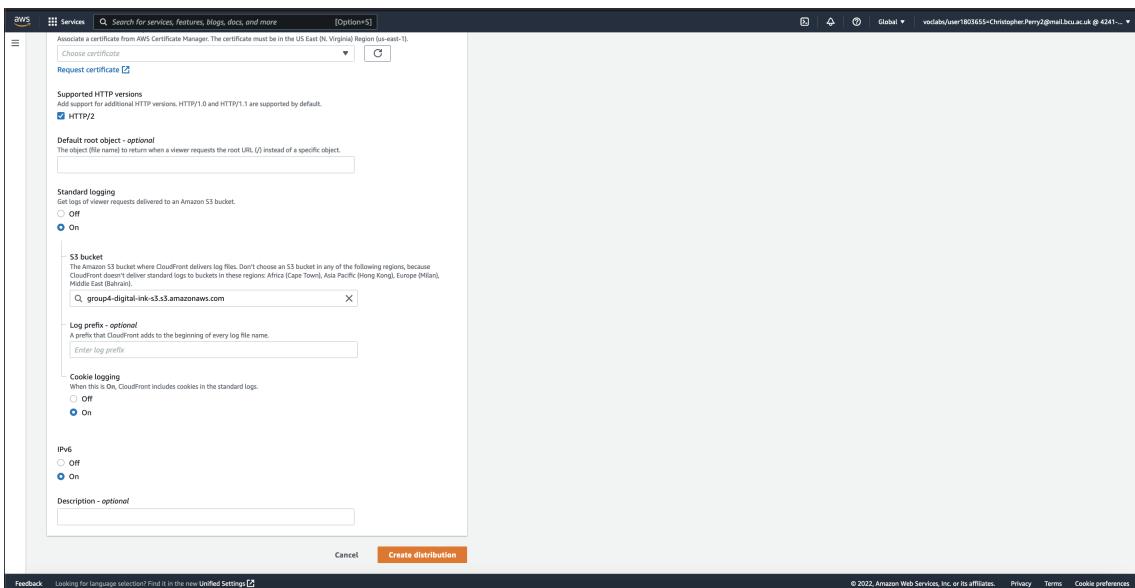


Figure 7.7: Applying CloudFront Settings.

A CloudFront Distribution has now been created, and can be found in Figure 7.8.

The screenshot shows the AWS CloudFront console. On the left, there's a sidebar with various navigation options like Distributions, Telemetry, Reports & analytics, Security, and Key management. The main area is titled 'Distributions (1) Info' and shows a table with one row. The table columns are ID, Description, Domain name, Alternate domain..., and Origins. The single entry is E21BWNC9VESEB5, with a description of -, a domain name of d1bdkf7iuqj4qy.cloudfront.net, and an origin group named 'group4-di'. There are buttons for Enable, Disable, Delete, and Create distribution at the top right of the table.

Figure 7.8: Created CloudFront Distribution.

Finally, the images contained within the webserver were changed from their local location to their relevant CloudFront locations. This change can be seen in Figures 7.9 and 7.10.

The screenshot shows a code editor with a file named 'base.blade.php' open. The code is a Blade template for a header. At line 51, there is a line of HTML that includes an image tag with a source attribute set to a local file path: '``'. This line is highlighted with a dark purple background. The rest of the code is standard HTML for a header, including links to stories, create, homepage, and about pages.

Figure 7.9: Image location before CloudFront.

The screenshot shows a code editor interface with a dark theme. The top navigation bar includes tabs for 'Project', 'resources', 'views', and 'base.blade.php'. The main area displays a portion of the 'base.blade.php' file. The code is as follows:

```
36 <header>
37   <div class="container">
38     <div class="row">
39       </-- stories page link -->
40       <div class="col header-option">
41         <a href="/stories" class="header-option">Stories</a>
42       </div>
43
44       <div class="col header-option">
45         <a href="/stories/create" class="header-option">Create</a>
46       </div>
47
48       </-- homepage link / logo -->
49       <div class="col header-option">
50         <a href="/">
51           
52         </a>
53       </div>
54
55       <div class="col header-option">
56         <a href="/about" class="header-option">About</a>
      </div>
    </div>.container .row .col.header-option a
```

The 'src' attribute of the image tag at line 51 contains a CloudFront URL: `https://d1bdkf7iuqj4qy.cloudfront.net/digital-ink-logo.png`. The code editor's status bar at the bottom right shows the file was last modified at 21:59:59.

Figure 7.10: Image location after CloudFront.

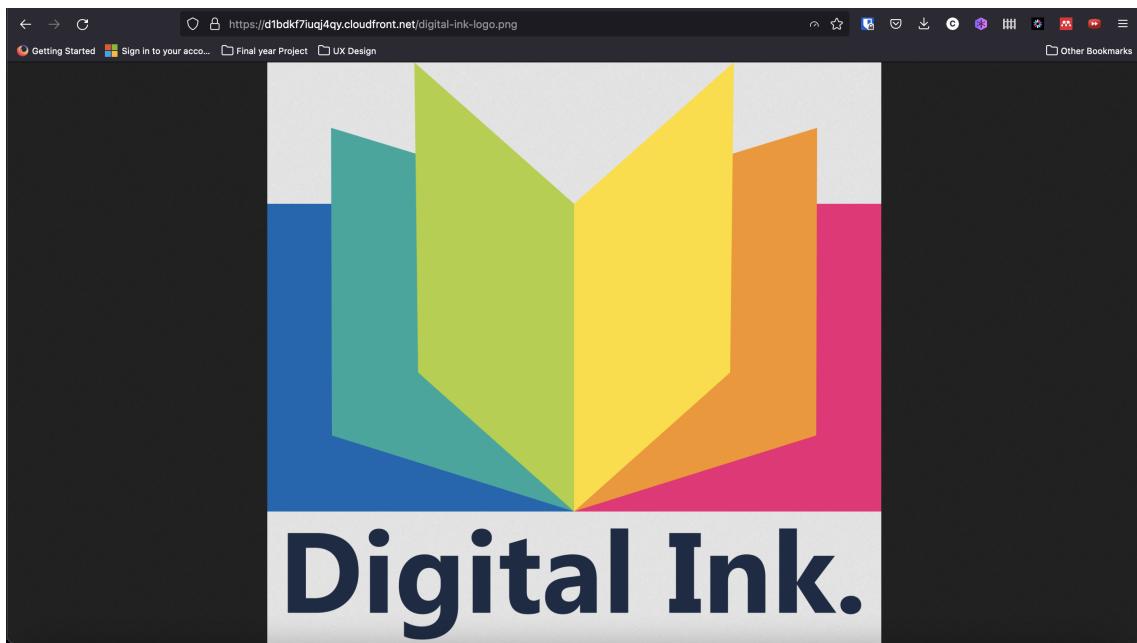


Figure 7.11: Image location after CloudFront.

As seen in Figure 7.11, the Digital Ink logo is now hosted on CloudFront.

Chapter 8

CloudWatch

Amazon CloudWatch is a monitoring and observation tool which can provide developers with insights to monitor applications, react to performance changes, and optimise resource consumption. CloudWatch collects monitoring and operational data about the application as logs and metrics to provide a complete overview of operation health, resource consumption, and the services currently running on AWS. This is useful for any cloud-based application as it allows developers to set alarms, visualise metric logs, troubleshoot errors and, most importantly, identify and correct anomalous behaviour amazon2022amazon. An example of CloudWatch usage would be an alarm set to alert developers when a specified resource consumption level is over a certain threshold.

For the purposes of the project, multiple CloudWatch alarms will be set up which will allow monitoring for budgeting and performance across the various AWS services used.

The first alarm which will be set up is a network alarm. Through selecting the

metric on the

instance, the packets which are being outputted can be monitored.

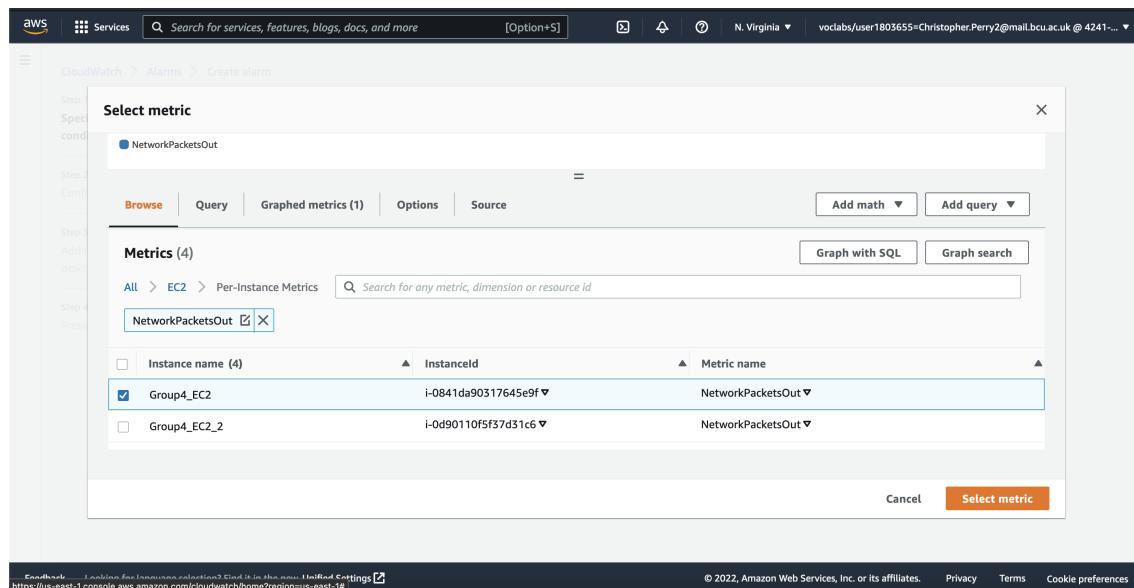


Figure 8.1: Selection of CloudWatch metric for EC2 instance.

The metric will be configured to alarm in the event that there is less than 5 packets of data sent per day from the instance. As the instance currently outputs nearly, 30,000 packets per day, this will be useful to check if the web app has failed.

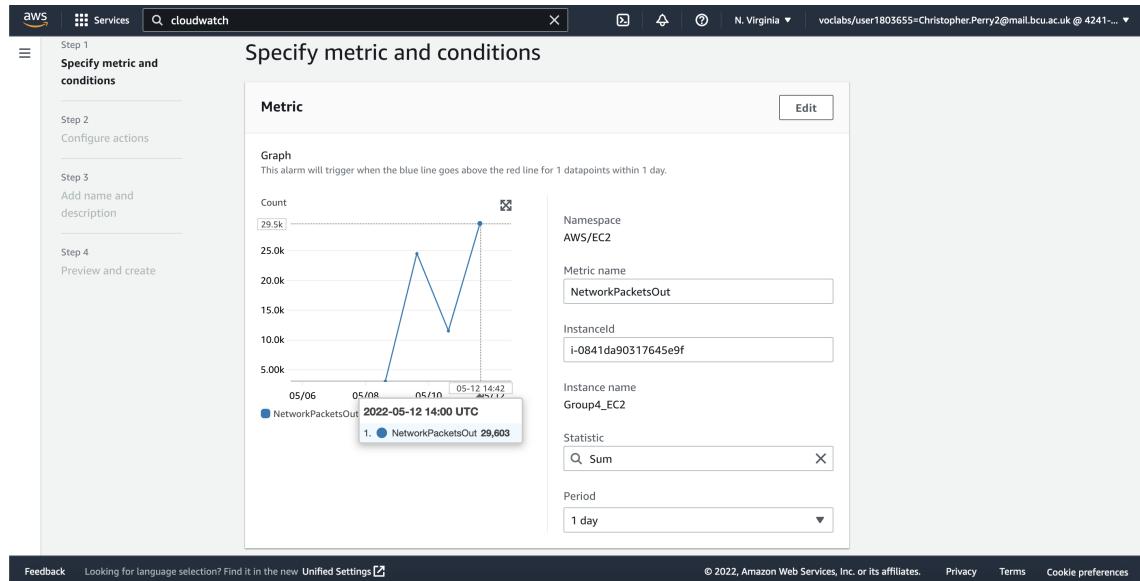


Figure 8.2: Configuration of NetworkPacketsOut Metric.

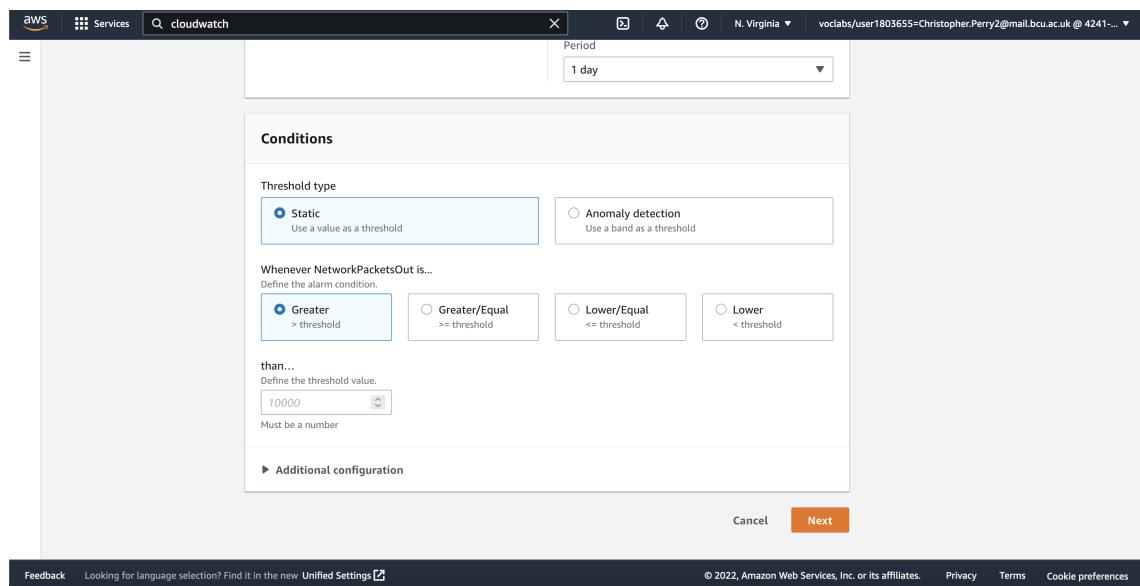


Figure 8.3: Configuration of NetworkPacketsOut Metric.

Figures 8.2 and 8.3 detail the alarm being set so that when the sum of packets sent is less than 5 per day, the alarm will activate.

In addition to the initial configuration of these CloudWatch alarms, SNS topics will also be set up so that every member of the group will be emailed in the event that the alarms activate. Figure 8.4 details this process.

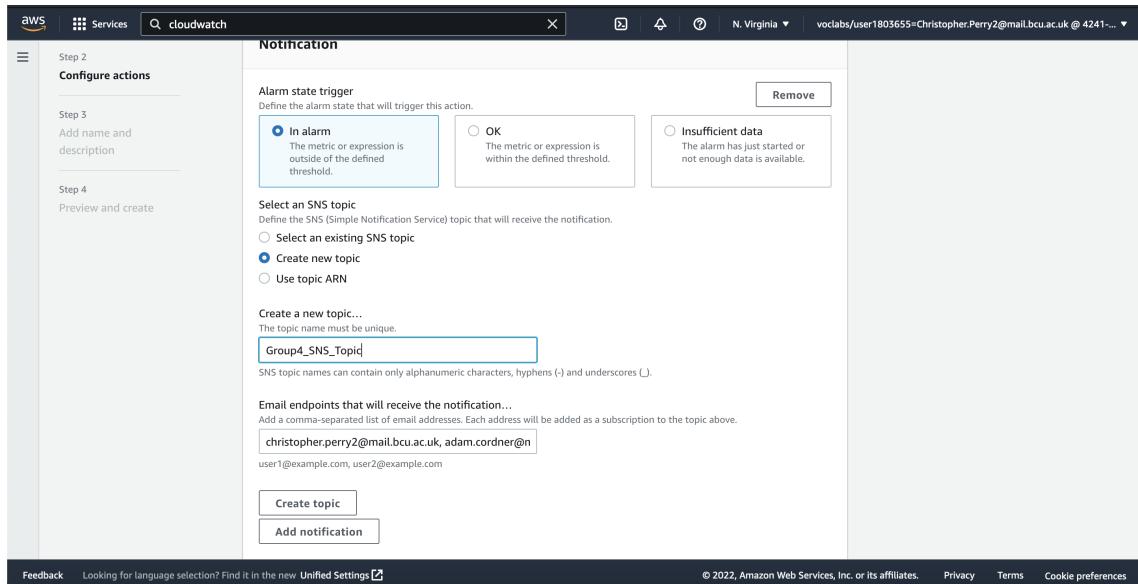


Figure 8.4: Configuration of SNS Topic for email alerts on alarm activation.

An email was then sent to all group members upon completion of this form, and the SNS topic was subsequently subscribed to, as shown in Figures 8.5 and 8.6.

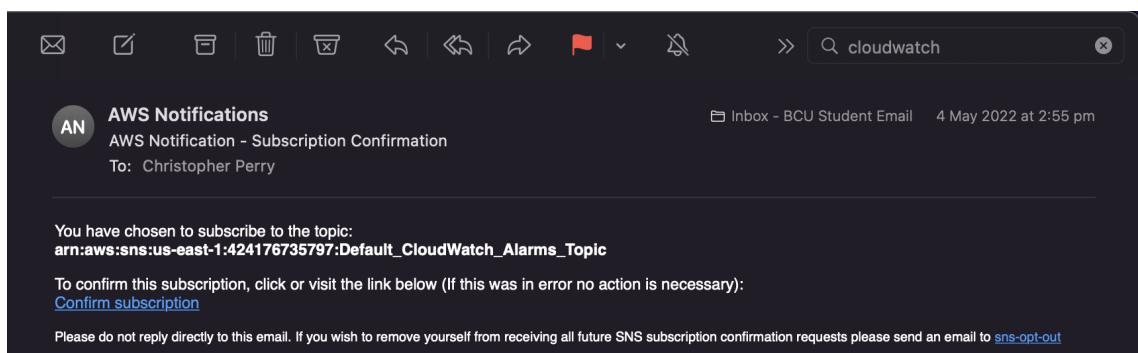


Figure 8.5: CloudWatch SNS Topic Email.

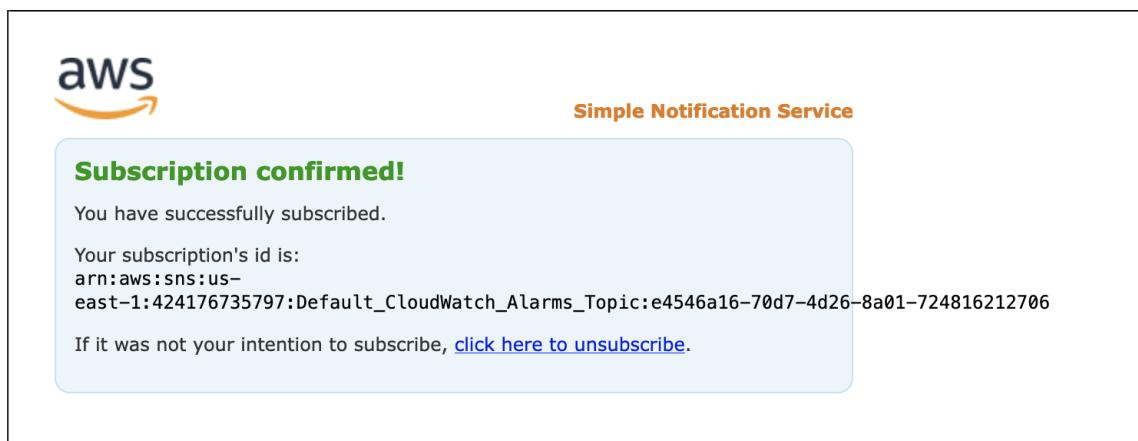


Figure 8.6: Successful subscription to SNS topic.

The EC2 instance has also been configured to restart when this alarm activates. This is done by automatically re-running the scripts used to start the web app on the instance starting up again.

This process can be seen in Figure 8.7.

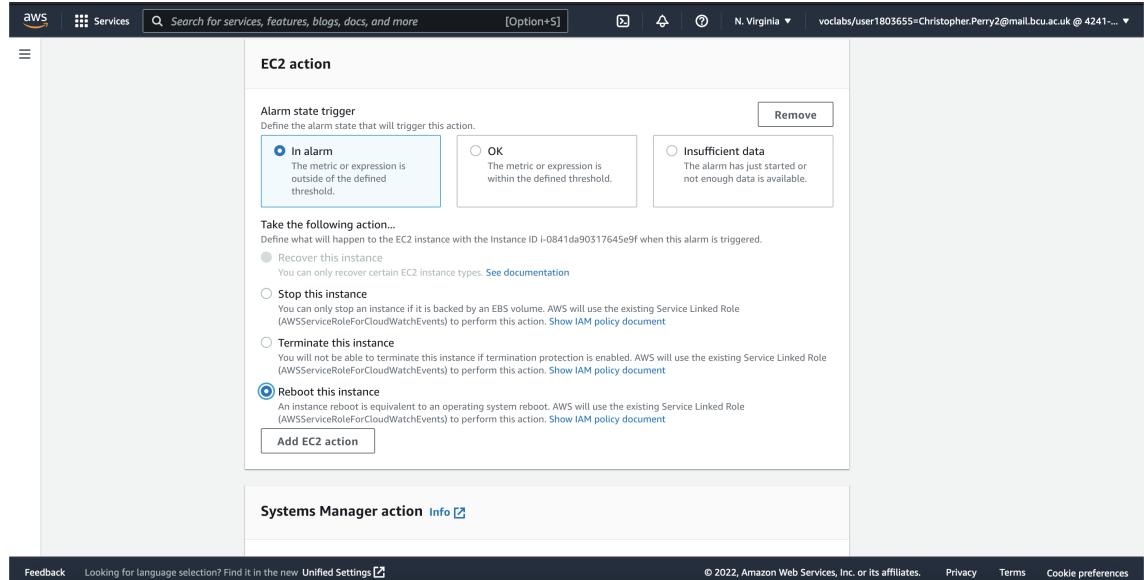


Figure 8.7: Configuration for EC2 instance to reboot on alarm activation.

A brief description of the CloudWatch alarm is then added. This can be seen in Figure 8.8.

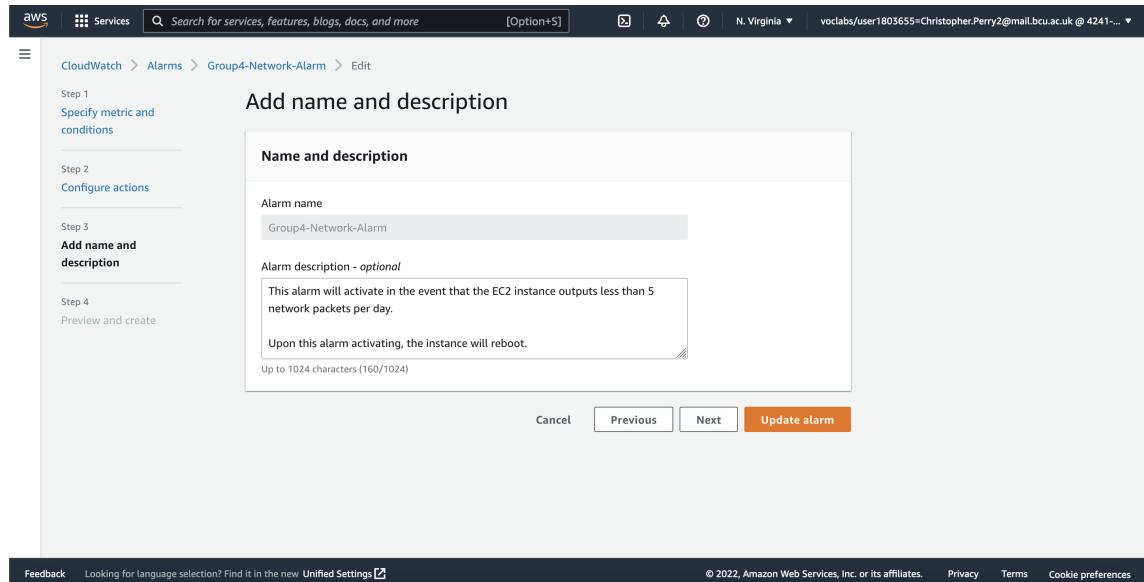


Figure 8.8: CloudWatch alarm description.

The alarm has now been set up, and can be seen in the CloudWatch Management Dashboard in Figure 8.9.

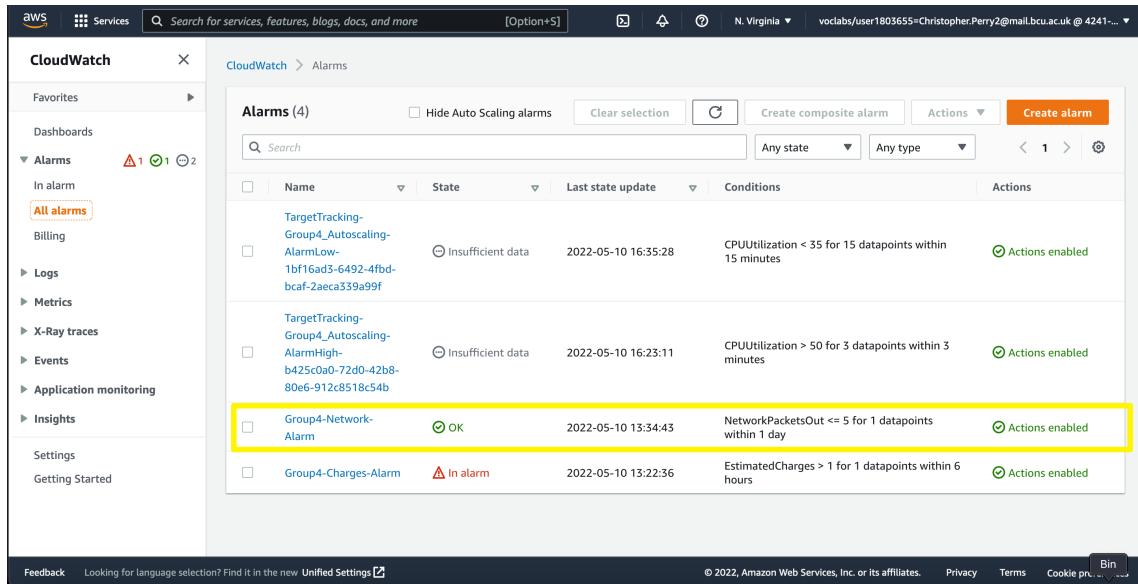


Figure 8.9: CloudWatch network alarm in dashboard.

In addition to the created network alarm, a charges alarm will also be set up. A similar process is followed whereby the metric of

is applied to all

instances, as detailed in Figure 8.10.

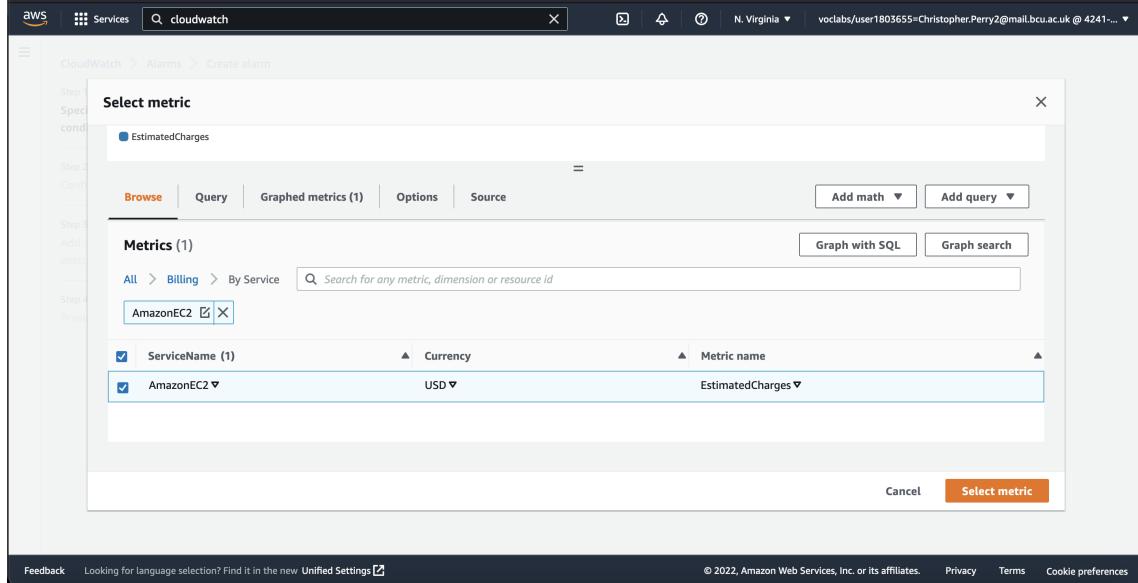


Figure 8.10: Selection of CloudWatch metric.

This metric will be configured to alarm in the event that the EC2 instance uses more than \$15 every 6 hours. This process can be seen in Figures 8.11 and 8.12.

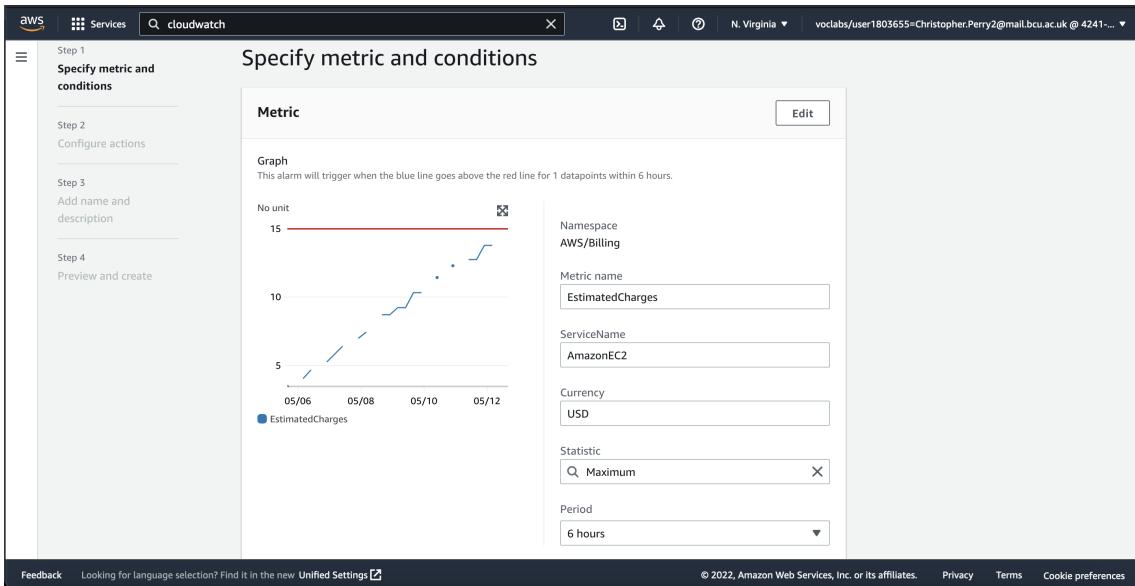


Figure 8.11: Configuration of CloudWatch alarm.

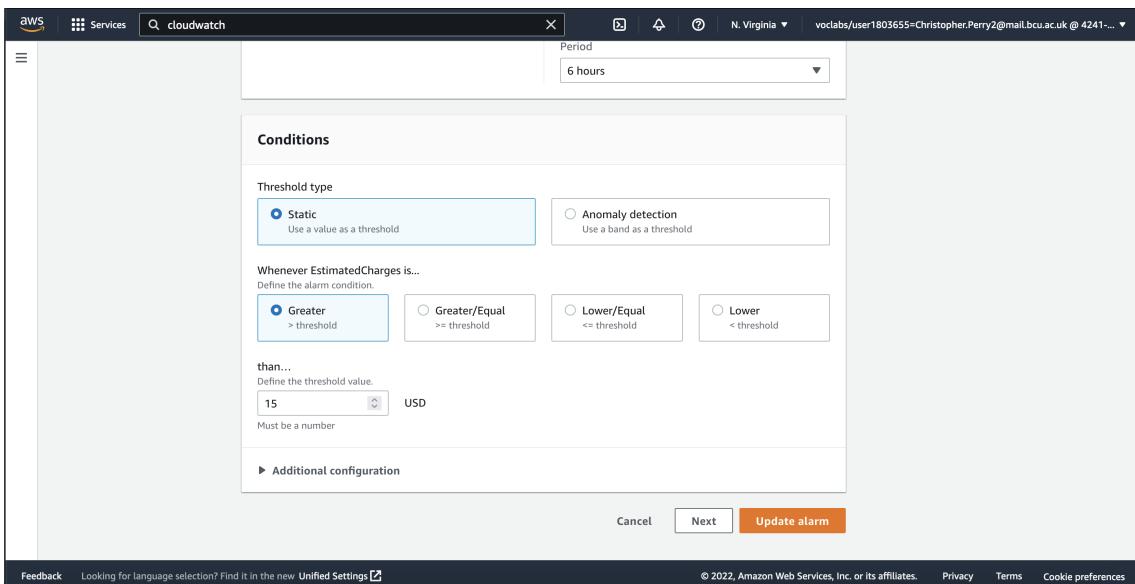


Figure 8.12: Configuration of CloudWatch alarm.

In addition to this, alerts will be sent to the same SNS topic configured earlier. This can be seen in Figure 8.13.

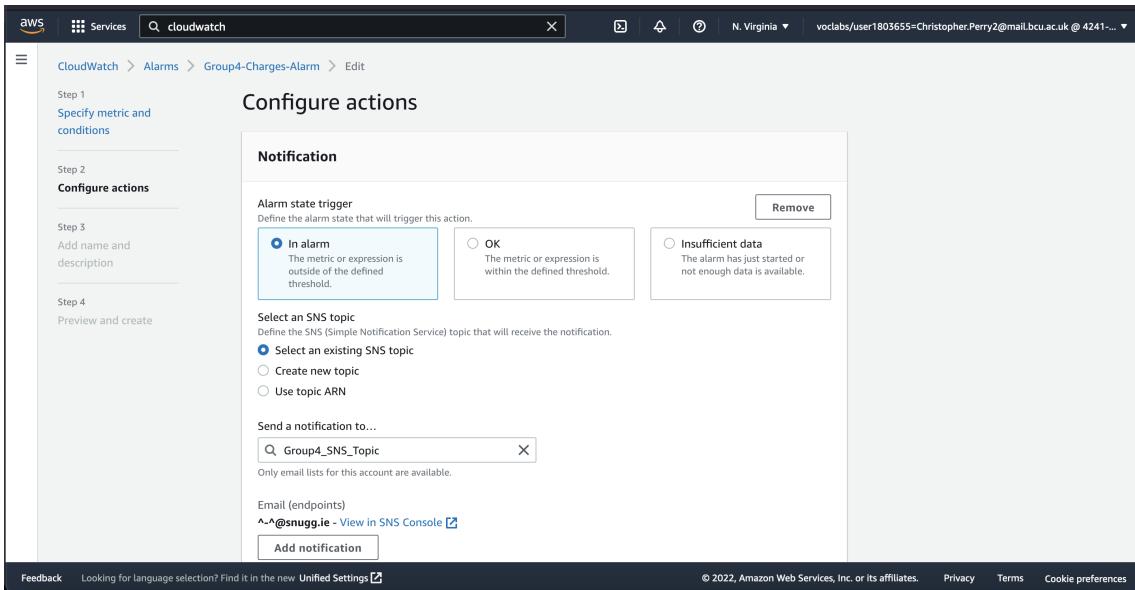


Figure 8.13: Setting CloudWatch charges SNS topic

A brief description was added to the alarm, as seen in Figure 8.14.

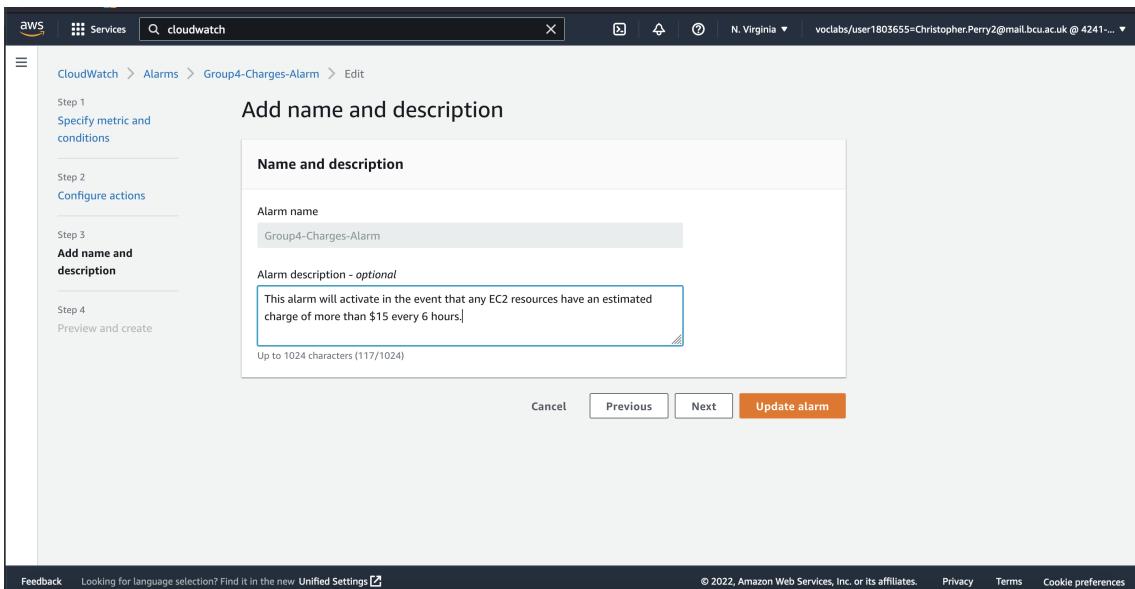
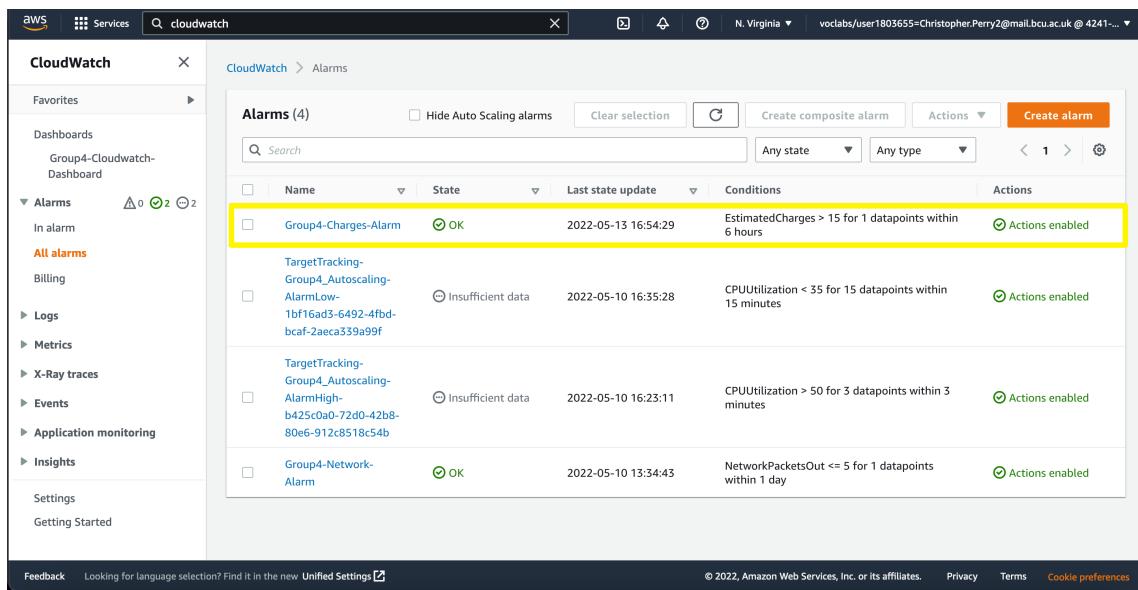


Figure 8.14: Description of CloudWatch alarm.

The alarm is now live and can be seen in Figure 8.15.



The screenshot shows the AWS CloudWatch Alarms dashboard. On the left, there's a sidebar with navigation links like Favorites, Dashboards, Alarms (selected), All alarms, Billing, Logs, Metrics, X-Ray traces, Events, Application monitoring, Insights, Settings, and Getting Started. The main area displays a table titled "Alarms (4)". The columns are Name, State, Last state update, Conditions, and Actions. The first row, "Group4-Charges-Alarm", is highlighted with a yellow border. Its details are as follows:

Name	State	Last state update	Conditions	Actions
Group4-Charges-Alarm	OK	2022-05-13 16:54:29	EstimatedCharges > 15 for 1 datapoints within 6 hours	Actions enabled
TargetTracking-Group4_Autoscaling-AlarmLow-	Insufficient data	2022-05-10 16:35:28	CPUUtilization < 35 for 15 datapoints within 15 minutes	Actions enabled
TargetTracking-Group4_Autoscaling-AlarmHigh-	Insufficient data	2022-05-10 16:23:11	CPUUtilization > 50 for 3 datapoints within 3 minutes	Actions enabled
Group4-Network-Alarm	OK	2022-05-10 13:34:43	NetworkPacketsOut <= 5 for 1 datapoints within 1 day	Actions enabled

Figure 8.15: CloudWatch charges alarm live in CloudWatch dashboard.

Chapter 9

CloudTrail

Chapter 10

Relational Database Service (RDS)

Amazon RDS service allows a user to create a fully-featured and highly-available SQL database that is automatically replicated to another availability zone. (TODO: Oops, no multi-az) This means that if the primary database becomes unavailable, there is automatic failover providing redundancy for all the data stored within.

To create an Amazon RDS instance, a suitable name/identifier for the database is required before created as well as a selection for the resource limits for the virtual server. The database requires a username and passphrase, although for additional security there is the option to automatically generate a passphrase.

Afterwards, the type of SQL database required (such as MySQL, PostgreSQL, MariaDB or others) will be selected and then the database should begin provisioning.

Chapter 11

Availability Zones

Chapter 12

Elastic Load Balancing (ELB)

Elastic Load Balancing is used to automatically scale your EC2 instances to meet any changes in demand from your users. ELB can be used to both scale up or down the instance's resources to ensure that your application is kept performant and available regardless of how many users visit. It is also possible to scale down the resources again, after the spike in usage has passed, to save money when it comes to billing and ensure you aren't paying for any additional resources that are not being used.

Chapter 13

Security Practices

Chapter 14

Cost Breakdown

If this web app was to be deployed to the public, or be expanded to a larger market, it would be important to gather estimated costs for hosting the app in the cloud with all the AWS services being used. The AWS Pricing Calculator can be used to calculate current costs and predict future costs. To calculate these costs, it is required to specify every implemented feature and several projected inputs and outputs, such as amount of data transferred on the app per month amazon2022aws. The monthly cost and a yearly cost of deploying the app in its current state was calculated first. Following this, the calculator was used to predict monthly and yearly costs for scaling the deployment up to larger user-bases. These figures were calculated for scaling the app up to 10,000 users, one million users, and ten million users, which would require upgrading some selected AWS features.

14.1 Estimated Costs

14.2 Scaling Up to 10,000 Users

14.3 Scaling Up to One Million Users

14.4 Scaling Up to Ten Million Users

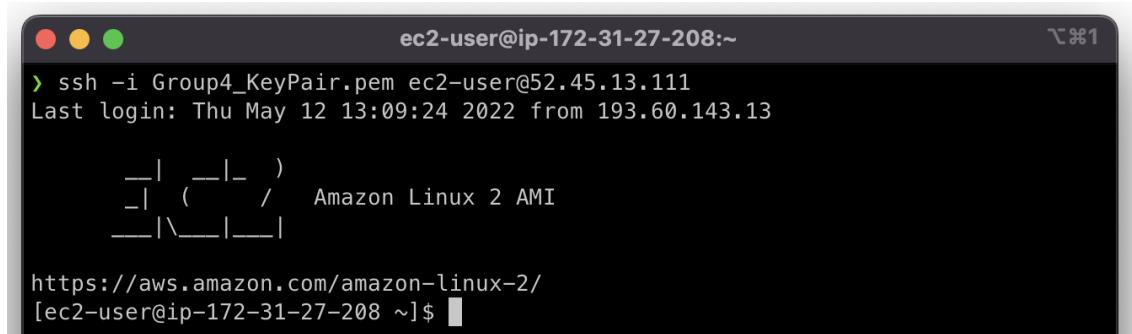
Chapter 15

Testing

This chapter of the report will detail the testing conducted on the configured AWS services. This was done to determine the accuracy and efficiency of the configurations made during the deployment process. The testing was conducted by using Gherkin, a language used to define behaviour and test cases dos2018automated. It is non-technical and is intended to be easily human-readable. Gherkin uses set keywords for structure and meaning: Given, When, and Then. An example of this structure can be seen in Figure ??.

EC2, S3, CloudFront, RDS, CloudWatch, and CloudTrail were all tested using this approach. Screenshots are included to illustrate the results of these tests.

15.1 Testing EC2



The screenshot shows a terminal window with a black background and white text. At the top, it displays the session information: 'ec2-user@ip-172-31-27-208:~'. Below this, a green arrow indicates the user has typed the command 'ssh -i Group4_KeyPair.pem ec2-user@52.45.13.111'. The response shows the last login details: 'Last login: Thu May 12 13:09:24 2022 from 193.60.143.13'. Following this, there is a stylized logo consisting of vertical and horizontal lines forming a face-like shape. Below the logo, the URL 'https://aws.amazon.com/amazon-linux-2/' is displayed, followed by the prompt '[ec2-user@ip-172-31-27-208 ~]\$'.

Screenshot of a web browser showing a login page for "Digital ink.".

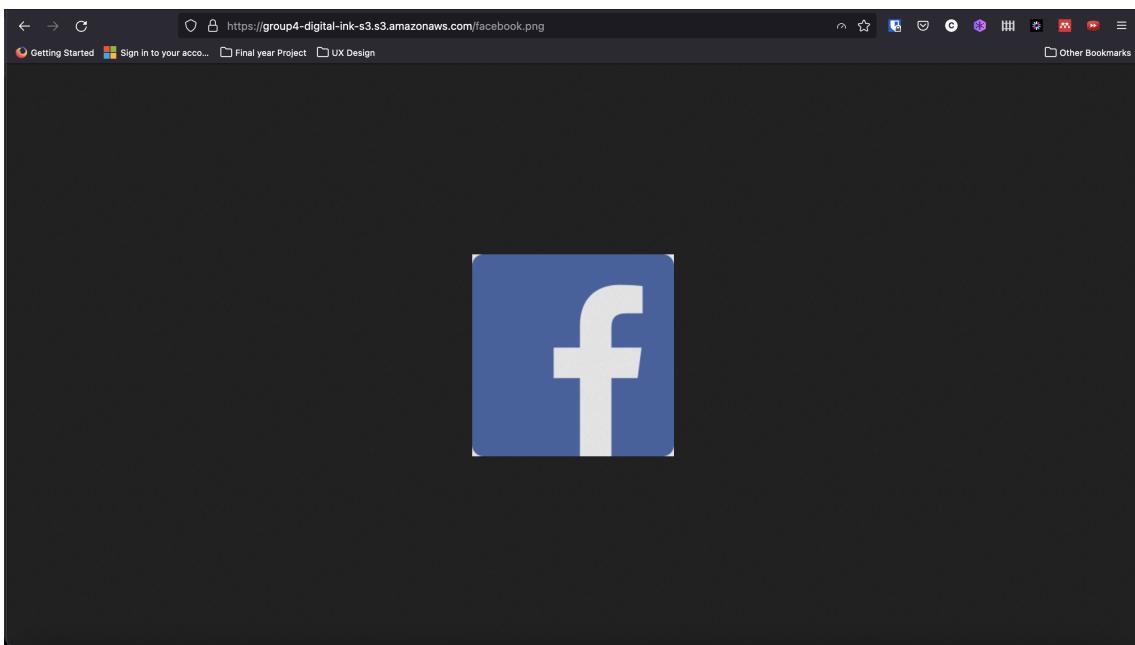
The browser address bar shows the URL: `ec2-52-45-13-111.compute-1.amazonaws.com`. The bookmarks bar includes links for "Getting Started", "Sign in to your acco...", "Final year Project", "UX Design", and "Other Bookmarks".

The page header features navigation links: Stories, Create, Digital ink. logo, About, and Account.

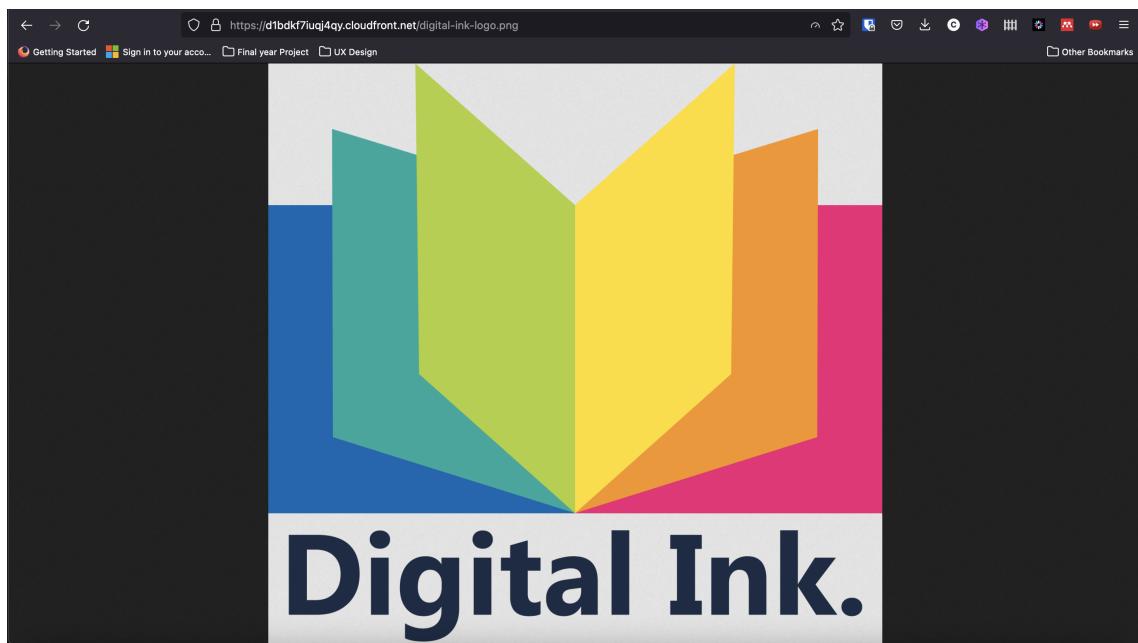
The main form area contains fields for Email and Password, both with placeholder text ("Enter Email" and "Enter Password"). A purple "Login" button is centered below the fields. To the right of the "Login" button is a link: "Don't have an account already? Why not sign up?". Below the "Email" field is a "Remember me" checkbox.

[Sign In](#)

15.2 Testing S3



15.3 Testing CloudFront



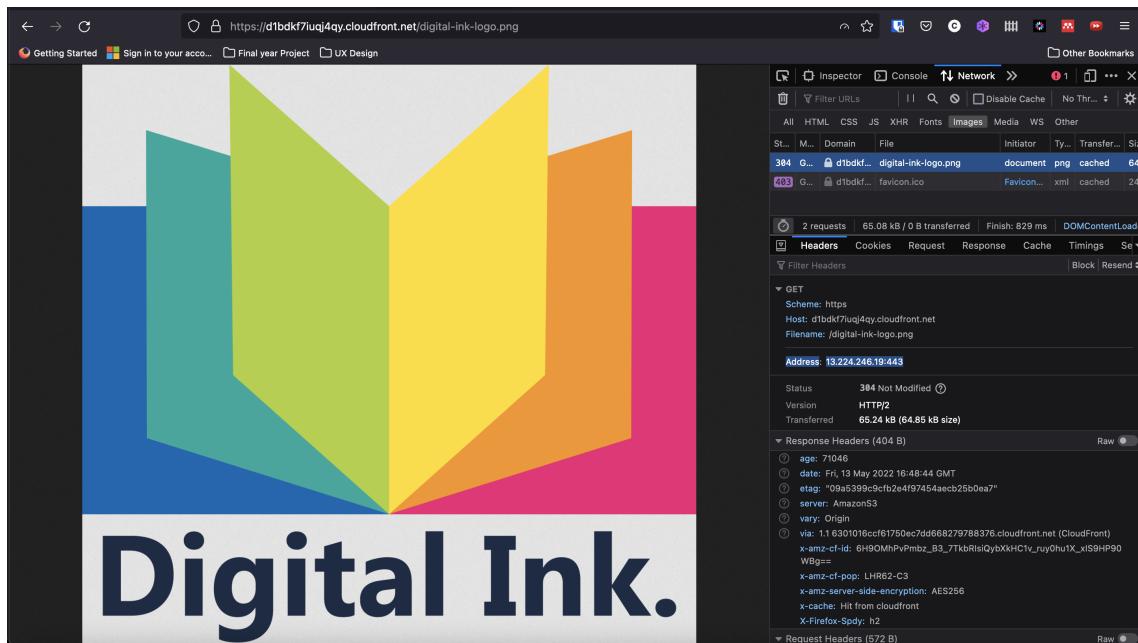


Figure 15.1: Digital Ink image whilst connected to UK IP.

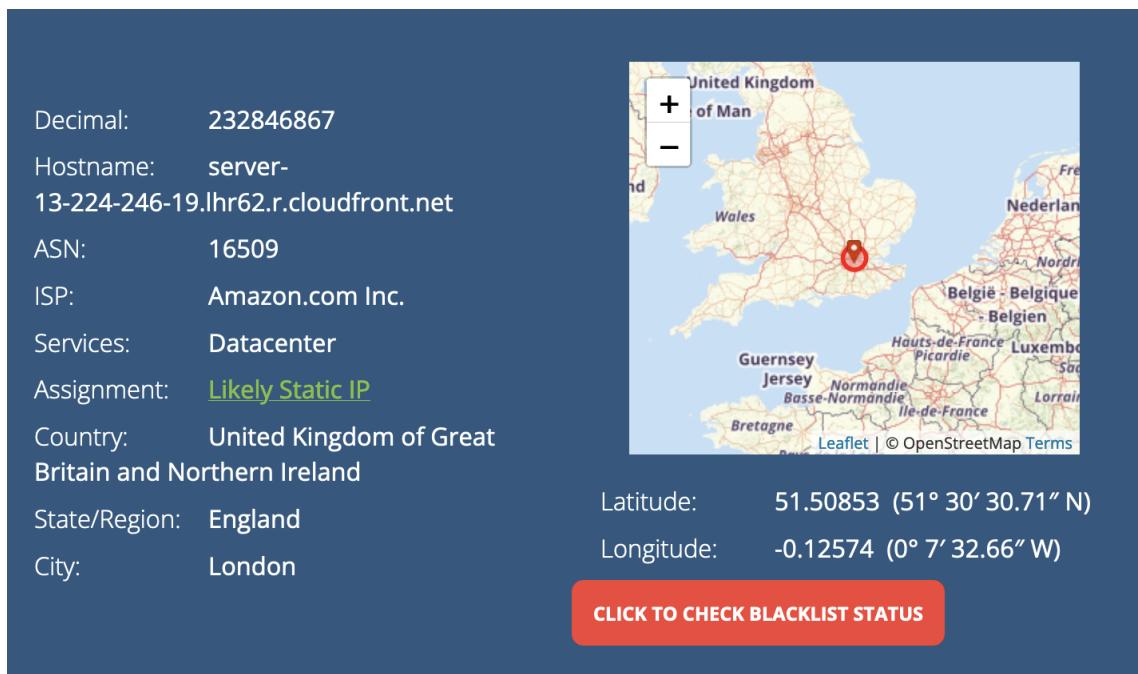


Figure 15.2: UK IP Location.

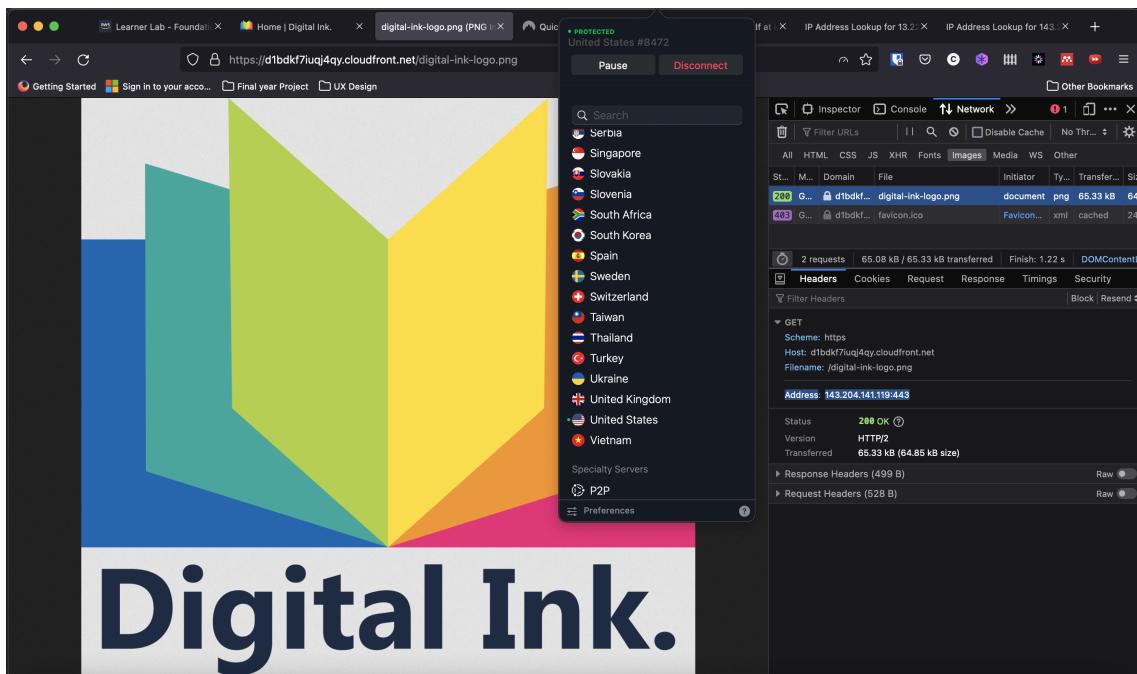


Figure 15.3: Digital Ink image whilst connected to US IP.

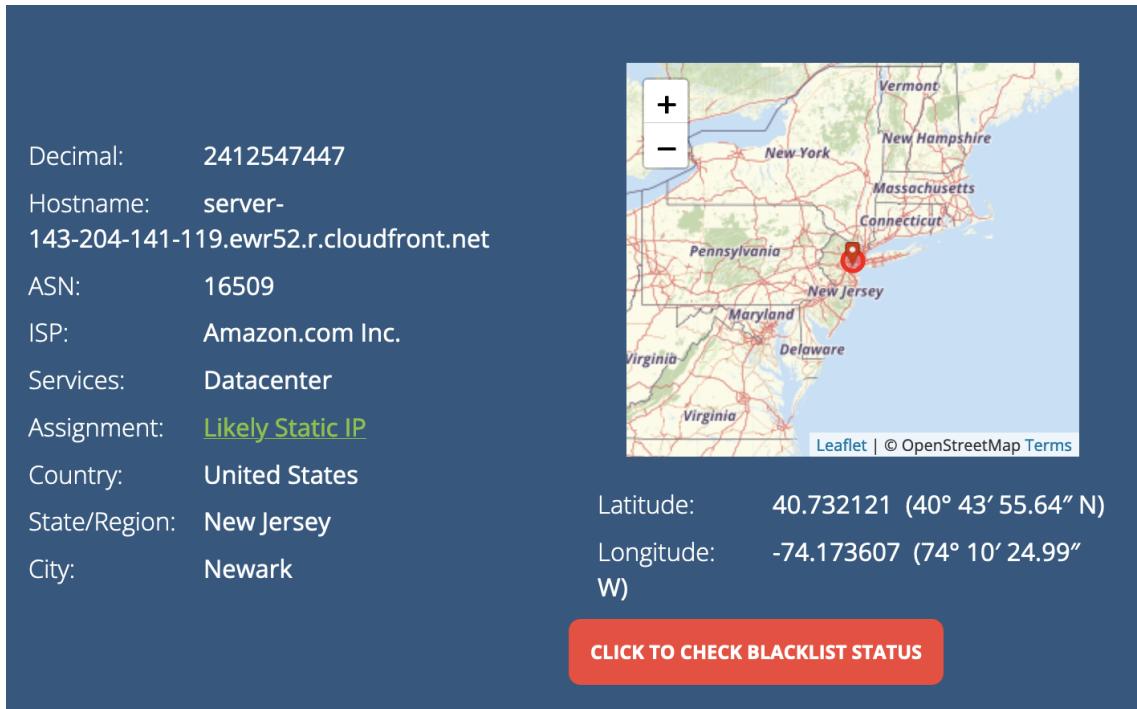


Figure 15.4: US IP Location.

15.4 Testing RDS

15.5 Testing CloudWatch

(One test for each of the metrics we set up.)

15.6 Testing CloudTrail

(One test for each of the metrics we set up.)

15.7 Testing ELB

(Test for turning off instance 1. Test for turning off instance 2.)

Chapter 16

Future Enhancements

An SSL/TLS certificate could be added, as it was not possible to do this from within the lab tutorial due to a lack of permissions from the root AWS account. By adding a certificate we would be able to encrypt the connections to our EC2 instance and increase the security of our application for all users.

We would like to add more roles and users to our IAM, but were limited by the amount that we were able to access.

If we were to do the project again, we would use AWS Elastic Beanstalk as it allows automatic orchestration of all the resources needed for a LAMP stack style application such as ours, it also has support for Docker which is a containerization technology that we had used in our application project.

Chapter 17

Conclusion

[heading=bibintoc]

Appendix A: Additional Screenshots

TODO: I am an appendix, please be kind.