

# Cloud Computing with AWS

By Adam Cordner, Chris Perry & Evie Snuffle

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

<b>1 Glossary</b>	<b>iv</b>
<b>2 Introduction</b>	<b>1</b>
<b>3 Web App</b>	<b>2</b>
3.1 Software Stack . . . . .	2
3.2 Database Design . . . . .	2
3.3 Interface Design . . . . .	6
<b>4 Virtual Private Cloud and Subnets</b>	<b>9</b>
<b>5 Elastic Cloud Compute</b>	<b>10</b>
5.1 AWS Setup . . . . .	10
5.2 EC2 Login . . . . .	14
5.3 Package Setup . . . . .	15
5.4 Web App Setup . . . . .	16
5.5 systemd Services . . . . .	18
<b>6 Simple Storage Service</b>	<b>19</b>
<b>7 CloudFront</b>	<b>20</b>
<b>8 CloudWatch</b>	<b>21</b>
<b>9 CloudTrail</b>	<b>22</b>
<b>10 Relational Database Service</b>	<b>23</b>
<b>11 Availability Zones</b>	<b>24</b>
<b>12 Elastic Load Balancing</b>	<b>25</b>
<b>13 Security Practices</b>	<b>26</b>
<b>14 Cost Breakdown</b>	<b>27</b>
14.1 Estimated Costs . . . . .	27
14.2 Scaling Up to 10,000 Users . . . . .	27
14.3 Scaling Up to 1 Million Users . . . . .	27
14.4 Scaling Up to 10 Million Users . . . . .	27

<b>15 Testing</b>	<b>28</b>
15.1 Testing EC2 . . . . .	28
15.2 Testing S3 . . . . .	28
15.3 Testing CloudFront . . . . .	28
15.4 Testing RDS . . . . .	29
15.5 Testing CloudWatch . . . . .	30
15.6 Testing CloudTrail . . . . .	30
15.7 Testing ELB . . . . .	30
<b>16 Future Enhancements</b>	<b>31</b>
<b>17 Conclusion</b>	<b>32</b>
<b>References</b>	<b>33</b>
<b>Bibliography</b>	<b>33</b>
<b>A Screenshots</b>	<b>34</b>

# List of Figures

3.1 Database tables overview. . . . .	3
3.2 migrations table. . . . .	3
3.3 users table. . . . .	4
3.4 stories table. . . . .	5
3.5 <i>Digital-Ink</i> home page log in and sign up forms. . . . .	6
3.6 <i>Digital-Ink</i> story creation form. . . . .	7
3.7 <i>Digital-Ink</i> account page. . . . .	8
3.8 <i>Digital-Ink</i> stories page and story view. . . . .	8
5.1 Selection of EC2 OS Image. . . . .	10
5.2 Selection of EC2 Instance. . . . .	11
5.3 Selection of EC2 Keypair. . . . .	11
5.4 Selection of EC2 Networking options. . . . .	12
5.5 Generated EC2 Keypair in the .pem format. . . . .	12
5.6 Selection of EC2 Storage Configuration. . . . .	13
5.7 SSH command to log into EC2 instance. . . . .	14
5.8 Logging into EC2 instance. . . . .	14
5.9 Installing Git. . . . .	15
5.10 Installing Docker. . . . .	16
5.11 Cloning the web app from Github. . . . .	16
5.12 Containers required for the web app being pulled from Docker Hub. . . . .	17
5.13 Creation of tables through php artisan migrate command. . . . .	17
15.1 Gherkin example. . . . .	28
A.1 After Allocating Elastic IP Address . . . . .	34
A.2 Allocating Elastic IP Address . . . . .	34
A.3 Cloning the App . . . . .	35
A.4 CloudWatch Conditions . . . . .	35
A.5 CloudWatch Specify Metric . . . . .	36
A.6 Create Instance - Application and OS Images . . . . .	36
A.7 Create Instance - Configure Storage . . . . .	37
A.8 Create Instance - Instance Type . . . . .	37
A.9 Create Instance - Name & Tags . . . . .	38
A.10 Create Instance - Network Settings . . . . .	38
A.11 Creating Key Pair . . . . .	39
A.12 Digital Ink . . . . .	39

A.13 Docker Compose . . . . .	40
A.14 Edit Instance - Network Settings . . . . .	40
A.15 Installing Docker using Package Manager . . . . .	41
A.16 Installing Docker using Package Manager (In Progress) . . . . .	41
A.17 Installing Git . . . . .	42
A.18 Starting Docker systemd Service . . . . .	42
A.19 Instances . . . . .	43
A.20 Launching Instance . . . . .	43
A.21 Log In with Key Pair . . . . .	44
A.22 Selecting a VPC Configuration . . . . .	44
A.23 Successfully Initiated Instance . . . . .	45
A.24 VPC Successfully Created . . . . .	45
A.25 VPC with Public and Private Subnets, Loading . . . . .	46
A.26 VPC with Public and Private Subnets . . . . .	46
A.27 Your VPCs . . . . .	47

# 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
- **CPU** — Central Processing Unit
- **DB** — Database
- **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
- **LAMP** — Linux, Apache, MySQL, PHP
- **ML** — Machine Learning

- **nslookup** — Name Server Lookup
- **DIG** — Domain Information Groper
- **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
- **SNS** — Simple Notification Service
- **SQL** — Structured Query Language
- **SSH** — Secure Shell
- **VPC** — Virtual Private Cloud
- **VPN** — Virtual Private Network

# 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 (Lee and Ware, 2003). The web app is hosted within a Docker container (Anderson, 2015) 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 (Fielding and Kaiser, 1997). MySQL is an open-source relational database management system (Widenius, Axmark, and Arno, 2002) 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 (Lerdorf et al., 2002). 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: `users`, `stories`, and `migrations`.

The `migrations` table (see Figure 3.2) contains records which correspond to the migrations within the Laravel web app. These migrations contain the scripts required to automatically generate the `users` and `stories` tables in SQL. It contains the following three columns:

- `id`: the unique ID for each migration.
- `migration`: points to the scripts used to create tables.

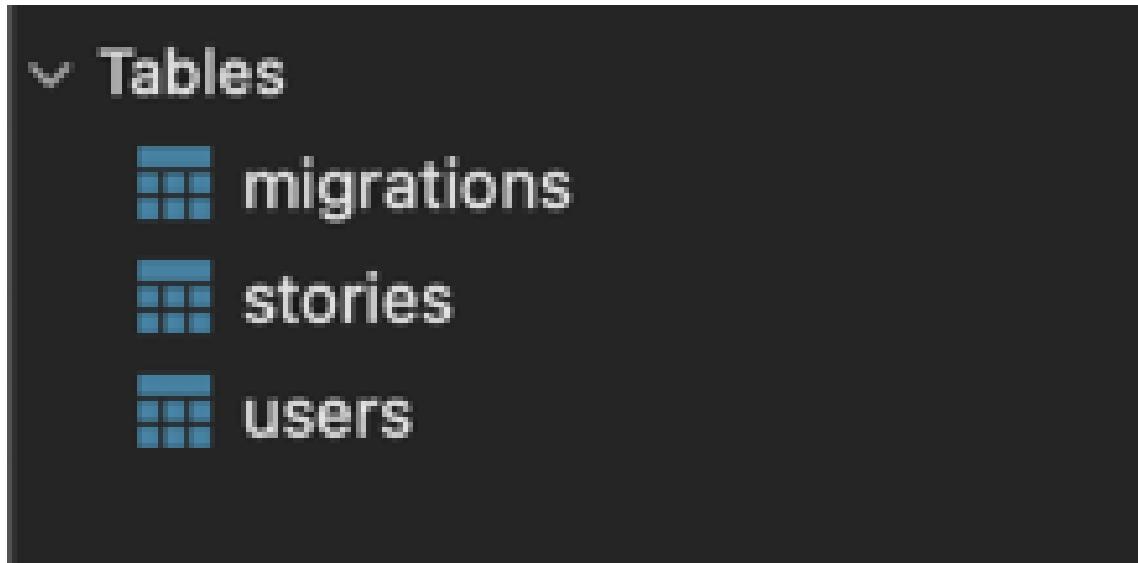


Figure 3.1: Database tables overview.

- `batch`: how many times the script has been ran.

<code>id</code>	<code>migration</code>	<code>batch</code>
1	<code>2014_10_12_000000_create_users_table</code>	1
4	<code>2020_03_13_105916_create_stories_table</code>	1

Figure 3.2: `migrations` table.

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

- `id`: the unique ID for each user account.
- `name`: the name associated with user account.
- `email`: the unique email used to log in.
- `password`: the password used to log in, encrypted with 184 bit hashing by Bcrypt (Laravel, 2022b).
- `remember_token`: keeps the user logged in if they select "Remember me".
- `created_at`: records what date and time the user account was first created at.
- `updated_at`: records what date and time the user account was last updated at.

The screenshot shows the MySQL Workbench interface with a database connection named 'MySQL 8.0.28 : TLSv1.2 : digital-ink : group4 : users'. The 'SQL' tab is selected. Below the tabs, there are two tables: 'stories' and 'users'. The 'users' table is displayed as a grid with the following data:

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...	mDUIGRIiVhyEi6...	2022-05-10 15:50:27	2022-05-10 15:50:27		
2	evie	^~^@snugg.ie	\$2y\$10\$vNRmMN/IRSIYTrCUJEPu44jkVi0r5...	Mu1WdyOz1e8...	2022-05-10 15:51:29	2022-05-10 15:51:29		

Figure 3.3: users table.

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

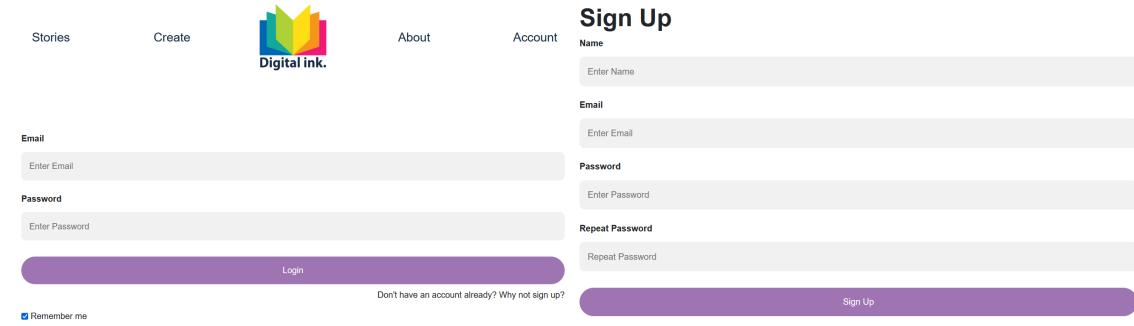
- `id`: the unique ID for each story.
- `author_id`: the unique ID associated with the user who created the story.
- `title`: the title associated with the story.
- `genre`: the genre associated with the story, which can be one of eight different genres.
- `blurb`: a brief description of the story.
-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

<code>id</code>	<code>author_id</code>	<code>title</code>	<code>genre</code>	<code>blurb</code>	<code>content</code>
2	1 →	Group 4 Loves AWS	Romance	Meet Group 4 and their love for AWS!	Group 4 loves using AWS' cloud features to host Di...
3	2 →	cheese savoury	Action and Adventure	nice and simple	on malted granary
<code>cover_image</code> <code>file_upload</code> <code>published</code> <code>created_at</code> <code>updated_at</code>					
<a href="#">./storage/cover_images/7537329101652200606.j...</a>		NULL		1 2022-05-10 16:36:46	2022-05-10 16:36:46
<a href="#">images/digital-ink-logo.png</a>		NULL		1 2022-05-10 16:38:51	2022-05-10 16:38:51

Figure 3.4: `stories` table.

### 3.3 Interface Design

The design of the web app was created using Blade, a powerful templating engine (Laravel, 2022a). 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 `users` table in the database.

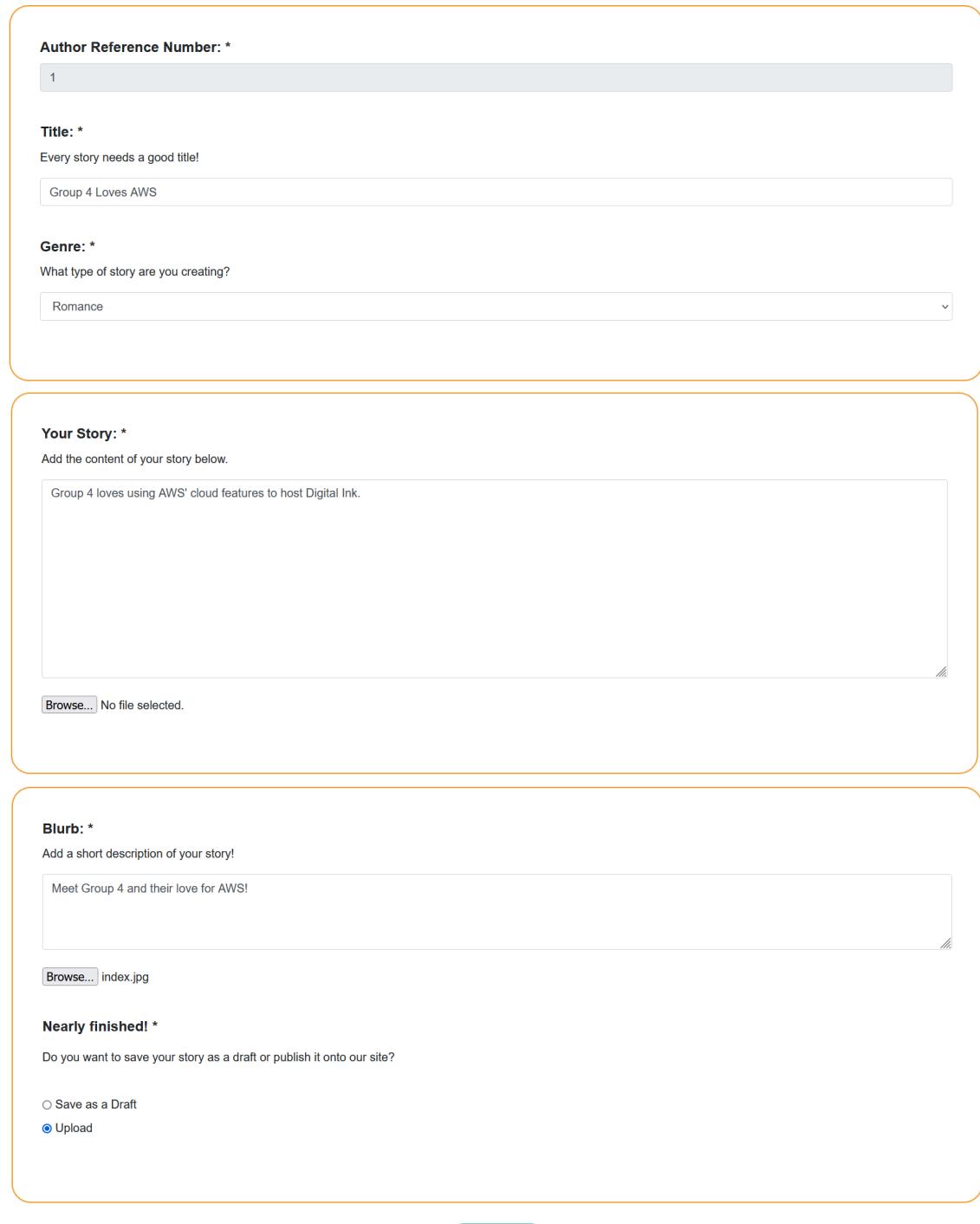


The figure shows the Digital-Ink home page with two side-by-side forms: a 'Login' form on the left and a 'Sign Up' form on the right. The top navigation bar includes links for 'Stories', 'Create', 'About', and 'Account'. The Digital-Ink logo is centered above the forms. The 'Login' form fields are 'Email' (placeholder 'Enter Email') and 'Password' (placeholder 'Enter Password'). Below these is a purple 'Login' button. A link 'Don't have an account already? Why not sign up?' is positioned between the two forms. The 'Sign Up' form fields are 'Name' (placeholder 'Enter Name'), 'Email' (placeholder 'Enter Email'), 'Password' (placeholder 'Enter Password'), and 'Repeat Password' (placeholder 'Repeat Password'). Below these is a purple 'Sign Up' button. A 'Remember me' checkbox is located at the bottom left of the page.

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 `stories` table.

## Create your story!



The figure shows a three-panel interface for creating a story. The top panel is titled 'Create your story!' and contains fields for 'Author Reference Number' (1), 'Title' (Every story needs a good title! Group 4 Loves AWS), and 'Genre' (Romance). The middle panel is titled 'Your Story:' and contains a text area with placeholder text ('Group 4 loves using AWS' cloud features to host Digital Ink.') and a file upload field ('Browse... No file selected'). The bottom panel is titled 'Blurb:' and contains a text area with placeholder text ('Meet Group 4 and their love for AWS!') and a file upload field ('Browse... index.jpg'). At the bottom right is a 'Nearly finished!' section with radio buttons for 'Save as a Draft' (unchecked) and 'Upload' (checked). A 'Complete' button is at the bottom center.

**Author Reference Number:** \*

1

**Title:** \*

Every story needs a good title!  
Group 4 Loves AWS

**Genre:** \*

What type of story are you creating?  
Romance

**Your Story:** \*

Add the content of your story below.

Group 4 loves using AWS' cloud features to host Digital Ink.

Browse... No file selected.

**Blurb:** \*

Add a short description of your story!

Meet Group 4 and their love for AWS!

Browse... index.jpg

**Nearly finished!** \*

Do you want to save your story as a draft or publish it onto our site?

Save as a Draft  
 Upload

Complete

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 `stories` table or removes a record from it.

The screenshot shows a user interface for managing published stories. At the top, a green header bar displays the message "Yay! Your story has been published!". Below this, a title "Hi Adam!" is shown. The main content area is titled "Here are your published stories:" and contains a table with three columns: "TITLE", "GENRE", and "ACTIONS". The table has one row of data: "Group 4 Loves AWS" under "TITLE", "Romance" under "GENRE", and two buttons under "ACTIONS": "Edit" (highlighted with a green oval) and "Delete" (highlighted with a red oval).

TITLE	GENRE	ACTIONS
Group 4 Loves AWS	Romance	<a href="#">Edit</a> <a href="#">Delete</a>

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.

The screenshot shows the "Stories" page and a detailed view of a story. The top navigation bar includes a search bar with placeholder text "Enter a title to search" and a "Search" button. The main content area displays a table with four columns: "AUTHOR ID", "TITLE", "GENRE", and "BLURB". Two stories are listed: "Group 4 Loves AWS" by author ID 1 (Romance, blurb: "Meet Group 4 and their love for AWS!") and "cheese savoury" by author ID 2 (Action and Adventure, blurb: "nice and simple"). Below the table, a specific story is viewed for "Group 4 Loves AWS". It shows a thumbnail image of a person, the title "Group 4 Loves AWS", the author "Written by: 1", the genre "Genre: Romance", and a blurb: "Group 4 loves using AWS' cloud features to host Digital Ink.". A "Back" button is located at the bottom left of the story view.

AUTHOR ID	TITLE	GENRE	BLURB
1	Group 4 Loves AWS	Romance	Meet Group 4 and their love for AWS!
2	cheese savoury	Action and Adventure	nice and simple

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

## **Chapter 4**

# **Virtual Private Cloud and Subnets**

Amazon Virtual Private Cloud (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 (Amazon Web Services (AWS), 2022b). A VPC can have multiple assigned subnets, which are a range of IP addresses accessible in the VPC.

# Chapter 5

# Elastic Cloud Compute

## 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 (Amazon Web Services (AWS), 2022a). 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.

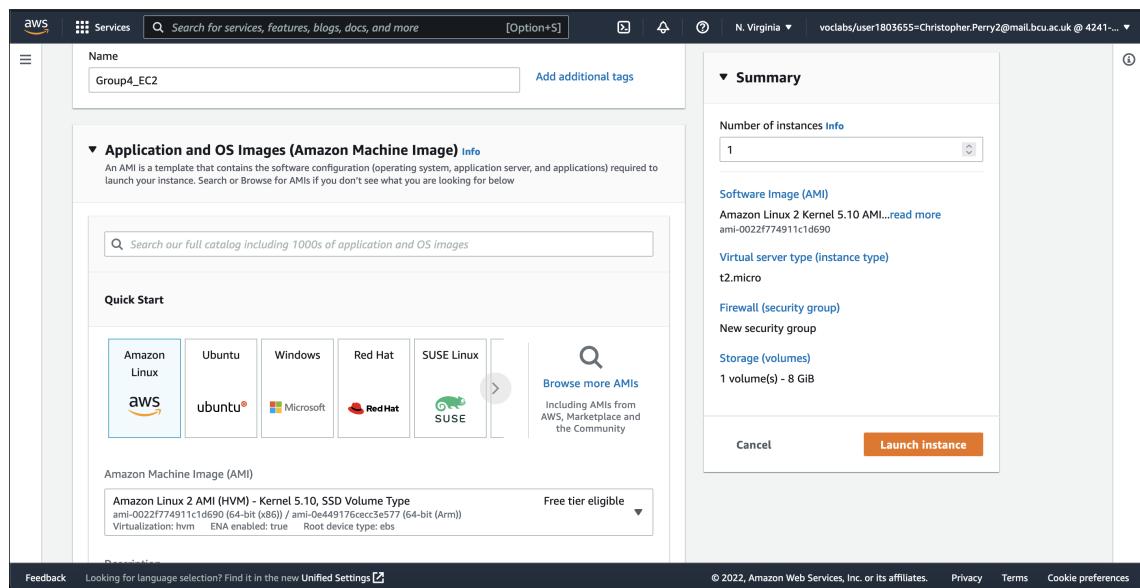


Figure 5.1: Selection of EC2 OS Image.

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). The selection of this can be found in Figure 5.2.

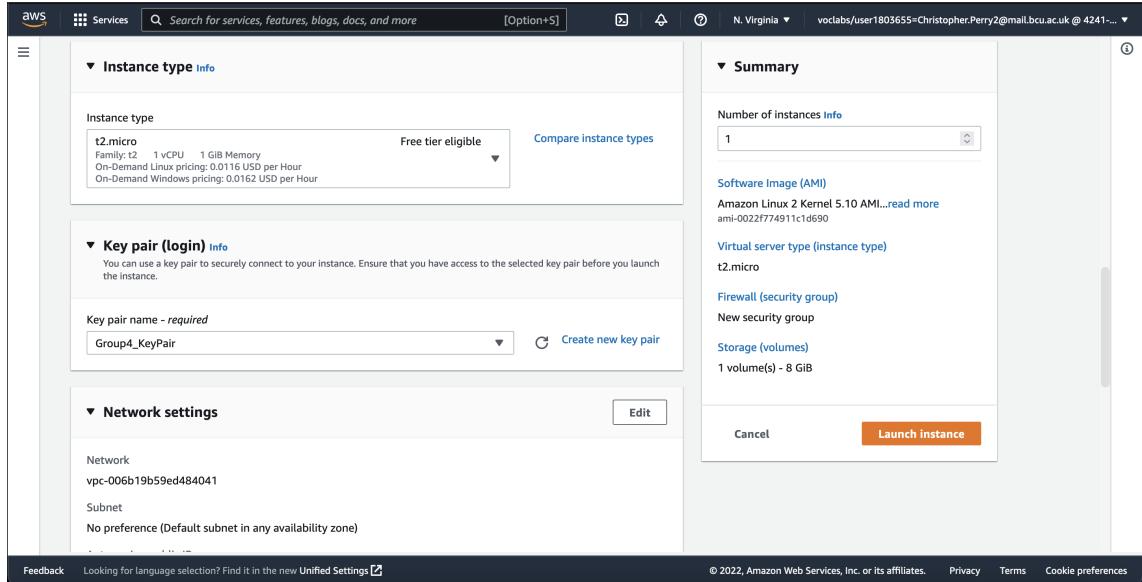


Figure 5.2: Selection of EC2 Instance.

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 DockerHub, which will allow a Laravel instance to be initialised, as discussed in Section 3.

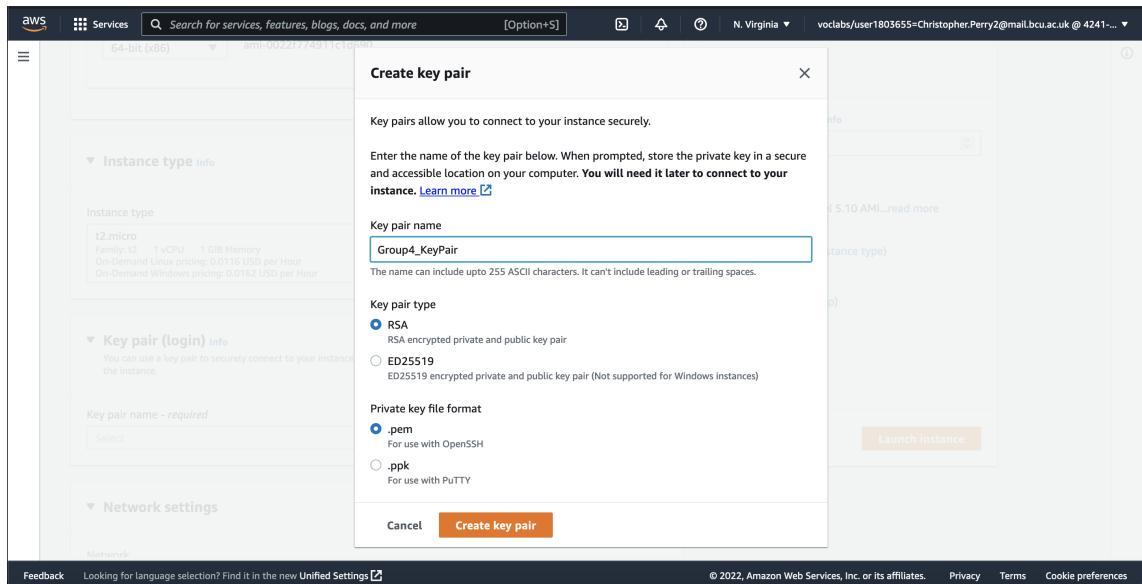


Figure 5.3: Selection of EC2 Keypair.

This process can be seen in Figure 5.3.

The instance is assigned the VPC created in Section 4, where it is assigned a subnet in the same availability zone of us-east-1.

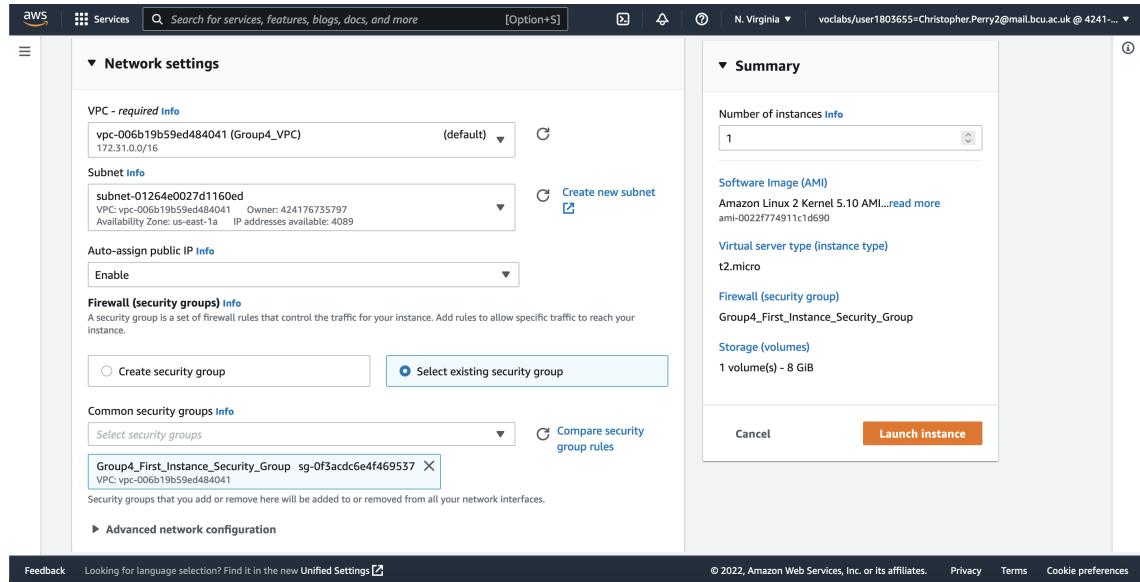


Figure 5.4: Selection of EC2 Networking options.

A screenshot of a terminal window on a Mac OS X desktop. The title bar says 'chris@Christophers-MacBook-Pro:~/Desktop/Keypair'. The command entered is 'ssh -i Group4\_KeyPair.pem ec2-user@52.45.13.111'. The terminal shows the command prompt and the user's name.

Figure 5.5: Generated EC2 Keypair in the .pem format.

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

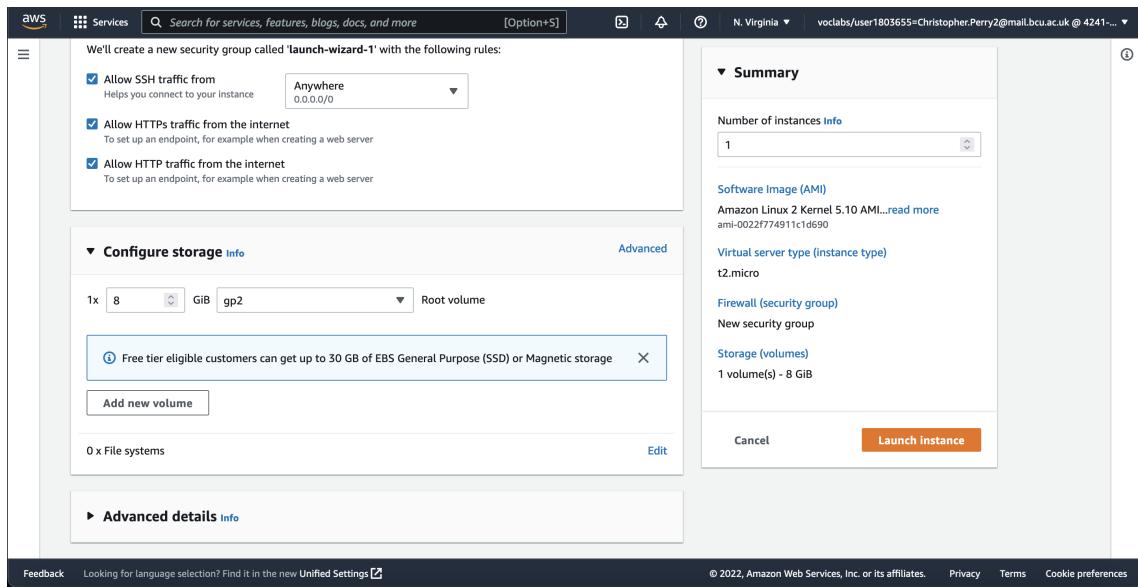


Figure 5.6: Selection of EC2 Storage Configuration.

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.

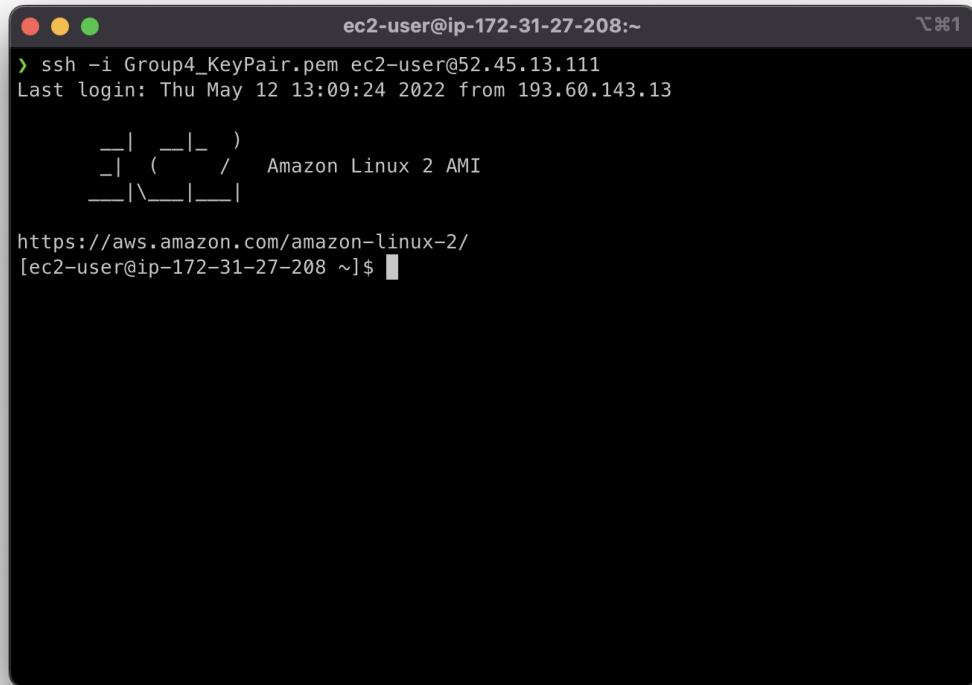
## 5.2 EC2 Login

The EC2 instance `group4-ec2` is now live, and the webapp can be loaded onto it. The instance is first logged in to through the use of the `ssh` command, followed by the `-i` argument to specify an identify file, which was generated earlier, and then the public ipv4 address of the instance.

```
ssh -i ~/Desktop/Group4_KeyPair.pem ec2-user@52.45.13.111
```

Figure 5.7: SSH command to log into EC2 instance.

This command can seen being executed in Figure 5.7.



A screenshot of a terminal window titled "ec2-user@ip-172-31-27-208:~". The window shows the command `ssh -i Group4_KeyPair.pem ec2-user@52.45.13.111` being run, followed by the output: "Last login: Thu May 12 13:09:24 2022 from 193.60.143.13". Below this, the Amazon Linux 2 AMI logo is displayed, consisting of a stylized tree or root system icon. The final line of output is `https://aws.amazon.com/amazon-linux-2/ [ec2-user@ip-172-31-27-208 ~]$`.

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 `yum install git`.

```
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-L-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/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          1/7
  Installing : git-core-doc-2.32.0-1.amzn2.0.1.noarch        2/7
  Installing : 1:perl-Error-0.17020-2.amzn2.noarch         3/7
  Installing : 1:emacs-filesystem-27.2-4.amzn2.0.1.noarch   4/7
  Installing : perl-TermReadKey-2.30-20.amzn2.0.2.x86_64    5/7
  Installing : git-2.32.0-1.amzn2.0.1.x86_64               6/7
  Verifying  : perl-Error-0.17020-2.amzn2.0.1.noarch         7/7
  Verifying  : git-core-2.32.0-1.amzn2.0.1.x86_64            1/7
  Verifying  : git-core-doc-2.32.0-1.amzn2.0.1.noarch         2/7
  Verifying  : 1:perl-Error-0.17020-2.amzn2.noarch          3/7
  Verifying  : 1:emacs-filesystem-27.2-4.amzn2.0.1.noarch    4/7
  Verifying  : perl-TermReadKey-2.30-20.amzn2.0.2.x86_64     5/7
  Verifying  : git-2.32.0-1.amzn2.0.1.x86_64                6/7
  Verifying  : git-core-2.32.0-1.amzn2.0.1.x86_64            7/7
Installed:
  git.x86_64 0:2.32.0-1.amzn2.0.1

Dependency Installed:
  emacs-filesystem.noarch 1:27.2-4.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 `yum install docker` is executed to install Docker as a result.

```
[ec2-user@ip-172-31-27-208 ~]$ sudo yum update
Loaded plugins: extras_suggestions, langpacks, 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, langpacks, 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: runc >= 1.0.0 for package: docker-20.10.13-2.amzn2.x86_64
--> Processing Dependency: libcgroup >= 0.40.rcl5.15 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 libcgroup.x86_64 0:0.41-21.amzn2 will be installed
-->> Package pigz.x86_64 0:2.3.4-1.amzn2.0.1 will be installed
-->> Package runc.x86_64 0:1.0.3-2.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 dependencies:
containerd x86_64 1.4.13-2.amzn2.0.1 amzn2extra-docker 23 M
libcgroup x86_64 0.41-21.amzn2 amzn2-core 66 k
pigz x86_64 2.3.4-1.amzn2.0.1 amzn2-core 81 k
runc x86_64 1.0.3-2.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/N]: [■]
```

Figure 5.10: Installing Docker.

## 5.4 Web App Setup

The web app is firstly cloned from its repository via the `git clone` command, and a new `digital-ink` folder is made to store the contents.

```
ec2-user@ip-172-31-27-208:~$ git clone https://github.com/ChrisP99/digital-ink
.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 `cd` command is used to move into the digital-ink folder, and the web app is subsequently launched through the `docker-compose up -d` to launch the web app as a detached Docker container. Relevant containers that are required to be downloaded from the `dockerfile` are then pulled.

```
phpmyadmin Pulling
: 5eb5h503b376 Extracting      27.53MB/31.37MB          5.95
: 8b1ad84cf101 Download complete                         5.75
: 38c937dadeb7 Download complete                         5.75
: 6a2f1dc96e59 Download complete                         5.75
: f8c3f82c39d4 Download complete                         5.75
: 90fcf6462bd8c Download complete                         5.75
[+] Running 0/319 Download complete                      5.75
F phpmyadmin Pulling
: 5eb5h503b376 Extracting      27.53MB/31.37MB          6.00
: 8b1ad84cf101 Download complete                         5.85
: 38c937dadeb7 Download complete                         5.85
: 6a2f1dc96e59 Download complete                         5.85
: f8c3f82c39d4 Download complete                         5.85
: 90fcf6462bd8c Download complete                         5.85
[+] Running 7/319 Download complete                      5.85
F phpmyadmin Pulling
: 5eb5h503b376 Pull complete                           13.05
: 8b1ad84cf101 Pull complete                           7.55
: 38c937dadeb7 Extracting [=====] 32.31MB/91.6MB    7.55
: 6a2f1dc96e59 Download complete                         12.85
: f8c3f82c39d4 Download complete                         12.85
: 90fcf6462bd8c Download complete                         12.85
[+] Running 7/319 Download complete                      12.85
F db Pulling
: 4eb654d6f7df59 Pull complete                           12.85
: 268554d6fe96 Download complete                         12.85
: 6c29fa0d4492 Download complete                         12.85
: 73e23c50a259 Download complete                         12.85
: 81ac1c3c96fc2 Download complete                         12.85
: b6baa3e623949 Download complete                         12.85
: dac5d67fd59 Download complete                         12.85
: f4d468666d9c36 Download complete                         12.85
: 443ab0e14c80 Download complete                         12.85
: 56e049224795 Download complete                         12.85
: 213e66cdcf56 Download complete                         12.85
: 99b9a4731108 Download complete                         12.85
F db Pulling
: 4eb654d6f7df59 Pull complete                           13.05
: 96e2a2b2711b Pull complete                           6.55
: 8aa3c0c50660 Pull complete                           8.05
: 9ef24f6c30 Pull complete                            9.45
: 76a886310f64 Pull complete                           9.85
: 15b23ec3f5f0 Extracting [=====] 13.11MB/14.06MB    10.15
: ae55dc337dc9 Download complete                         12.75
: a4c4c43adf52 Download complete                         12.75
: c6cab33e8f91 Download complete                         12.75
: 2ef1c4f2c43f6 Download complete                         12.75
: 2e5ee322af48 Download complete                         12.75
```

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

The result of this command launches 3 containers:

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

At the minute, the web app is live through the `digital-ink` container, and is using a local version of MySQL as a database, stored within the `mysqlDocker` container. The database has no tables, but can be populated through the use of Laravel. The container is firstly accessed through `docker exec app bash`, and the database is populated with tables with `php artisan migrate`. 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@de0e13abddff12:/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_000000_create_password_resets_table
Migrated: 2014_10_12_000000_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@de0e13abddff12:/srv/app#
```

Figure 5.13: Creation of tables through `php artisan migrate` command.

The subsequent output of this command can be found in Figure 5.13

## **5.5 systemd Services**

systemd services was used to ensure the automatic starting of the digital-ink web application

EVIE FILL THIS OUT YOU SILLY LLAMA

## **Chapter 6**

# **Simple Storage Service**

# **Chapter 7**

## **CloudFront**

# **Chapter 8**

## **CloudWatch**

# **Chapter 9**

## **CloudTrail**

# **Chapter 10**

## **Relational Database Service**

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**

## **Chapter 13**

# **Security Practices**

# **Chapter 14**

## **Cost Breakdown**

- 14.1 Estimated Costs**
- 14.2 Scaling Up to 10,000 Users**
- 14.3 Scaling Up to 1 Million Users**
- 14.4 Scaling Up to 10 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 (Santos and Vilain, 2018). 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 15.1.

```
Scenario: ...
  Given ...
  When ...
  Then ...
```

Figure 15.1: Gherkin example.

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

```
Scenario: Accessing instance through SSH with .pem file private key.
  Given ...
  When ...
  Then ...

Scenario: Accessing web app through EC2 domain name.
  Given ...
  When ...
  Then ...
```

### 15.2 Testing S3

```
Scenario: Accessing web app image through S3 domain name.
  Given ...
  When ...
  Then ...
```

### 15.3 Testing CloudFront

```
Scenario: Accessing web app image through CloudFront domain name.  
    Given ...  
    When ...  
    Then ...  
  
Scenario: Accessing web app image through CloudFront domain name in another region.  
    Given ...  
    When ...  
    Then ...  
  
Scenario: Accessing web app image through CloudFront domain name in another IP address.  
    Given ... (NSLookup test)  
    When ...  
    Then ...
```

## 15.4 Testing RDS

```
Scenario: Creating user information through the web app.  
    Given ...  
    When ...  
    Then ...  
  
Scenario: Creating story information through the web app.  
    Given ...  
    When ...  
    Then ...  
  
Scenario: Reading user information from the database into the web app.  
    Given ...  
    When ...  
    Then ...  
  
Scenario: Reading story information from the database into the web app.  
    Given ...  
    When ...  
    Then ...  
  
Scenario: Updating user information in the database through the web app.  
    Given ...  
    When ...  
    Then ...  
  
Scenario: Updating story information in the database through the web app.  
    Given ...  
    When ...  
    Then ...  
  
Scenario: Deleting user information in the database through the web app.  
    Given ...  
    When ...  
    Then ...
```

```
Scenario: Deleting story information in the database through the web app.  
  Given ...  
  When ...  
  Then ...
```

## 15.5 Testing CloudWatch

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

```
Scenario:  
  Given ...  
  When ...  
  Then ...
```

## 15.6 Testing CloudTrail

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

```
Scenario:  
  Given ...  
  When ...  
  Then ...
```

## 15.7 Testing ELB

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

## **Chapter 16**

# **Future Enhancements**

# **Chapter 17**

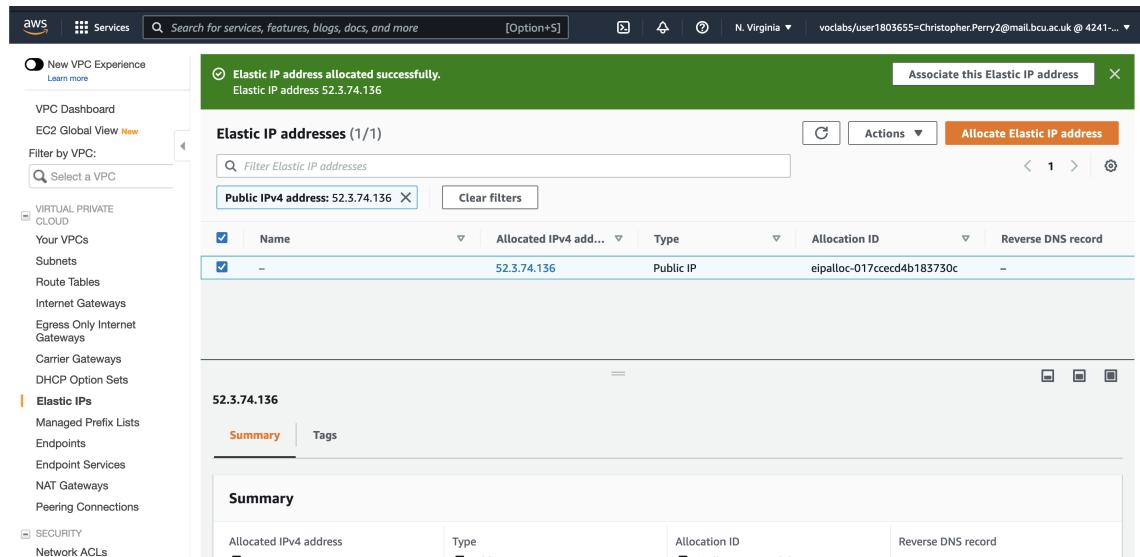
## **Conclusion**

# References

- Amazon Web Services (AWS) (2022a). *What is Amazon EC2?* AVAILABLE AT: <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html>.
- (2022b). *What is Amazon VPC?* AVAILABLE AT: <https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html>.
- Anderson, C. (2015). “Docker [software engineering]”. In: *IEEE Software* 32.3, pp. 102–c3. AVAILABLE AT: <https://doi.org/10.1109/MS.2015.62>.
- Fielding, R. T. and G. Kaiser (1997). “The Apache HTTP server project”. In: *IEEE Internet Computing* 1.4, pp. 88–90. AVAILABLE AT: <https://doi.org/10.1109/4236.612229>.
- Laravel (2022a). *Blade Templates*. AVAILABLE AT: <https://laravel.com/docs/9.x/blade>.
- (2022b). *Hashing*. AVAILABLE AT: <https://laravel.com/docs/9.x/hashing>.
- Lee, J. and B. Ware (2003). *Open Source Web Development with LAMP: Using Linux, Apache, MySQL, Perl, and PHP*. Addison-Wesley Professional. AVAILABLE AT: <https://isbnsearch.org/isbn/9780201770612>.
- Lerdorf, R. et al. (2002). *Programming PHP*. " O'Reilly Media, Inc.". AVAILABLE AT: <https://isbnsearch.org/isbn/9781565926103>.
- Santos, E. C. dos and P. Vilain (2018). “Automated acceptance tests as software requirements: An experiment to compare the applicability of fit tables and gherkin language”. In: *International Conference on Agile Software Development*. Springer, pp. 104–119. AVAILABLE AT: [https://doi.org/10.1007/978-3-319-91602-6\\_7](https://doi.org/10.1007/978-3-319-91602-6_7).
- Widenius, M., D. Axmark, and K. Arno (2002). *MySQL reference manual: documentation from the source*. " O'Reilly Media, Inc.". AVAILABLE AT: <https://isbnsearch.org/isbn/9780596002657>.

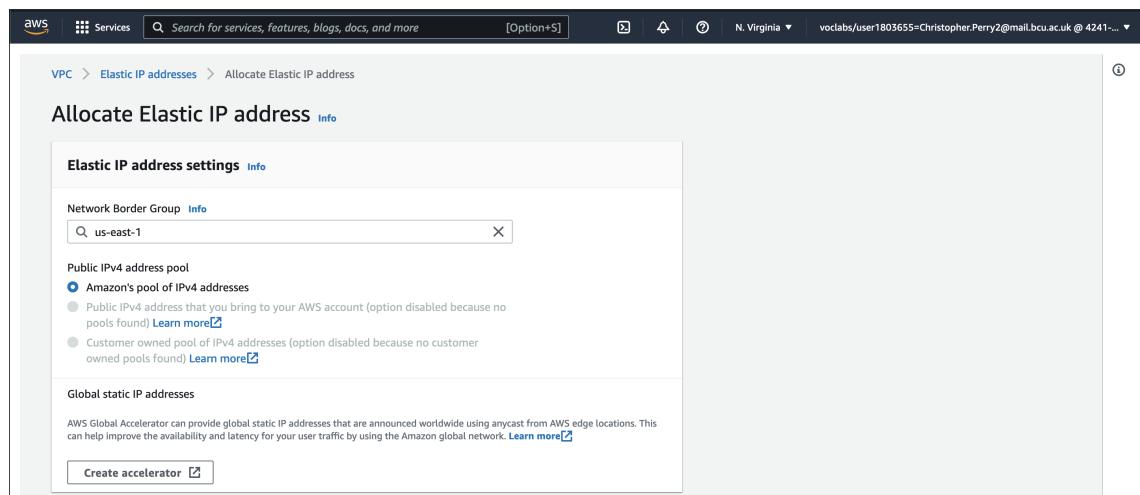
# Appendix A: Screenshots

I am an appendix, please be kind.



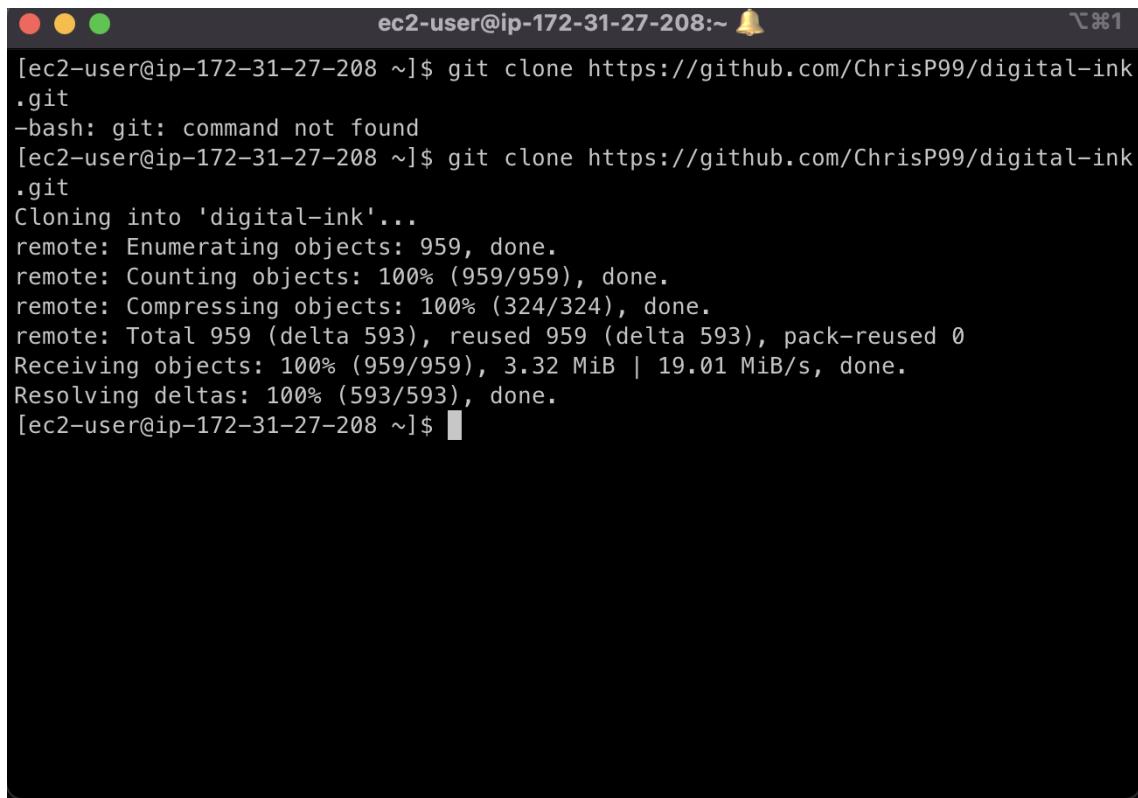
The screenshot shows the AWS VPC Elastic IP addresses page. At the top, a green banner displays the message "Elastic IP address allocated successfully." followed by the allocated IP address "52.3.74.136". Below the banner, the main interface shows a table of "Elastic IP addresses (1/1)". The table has columns for Name, Allocated IPv4 add..., Type, Allocation ID, and Reverse DNS record. One row is listed with the values: Name (checkbox selected), Allocated IPv4 add... (52.3.74.136), Type (Public IP), Allocation ID (eipalloc-017ccecd4b183730c), and Reverse DNS record (empty). Below the table, a summary card for the IP address 52.3.74.136 provides details such as Allocated IPv4 address, Type, Allocation ID, and Reverse DNS record.

Figure A.1: After Allocating Elastic IP Address



The screenshot shows the "Allocate Elastic IP address" settings page. The "Elastic IP address settings" section includes a "Network Border Group" dropdown set to "us-east-1". Under "Public IPv4 address pool", the "Amazon's pool of IPv4 addresses" option is selected. There is also a note about public IPv4 addresses being disabled because no pools were found. The "Global static IP addresses" section contains a note about AWS Global Accelerator and a "Create accelerator" button.

Figure A.2: Allocating Elastic IP Address



```
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 A.3: Cloning the App

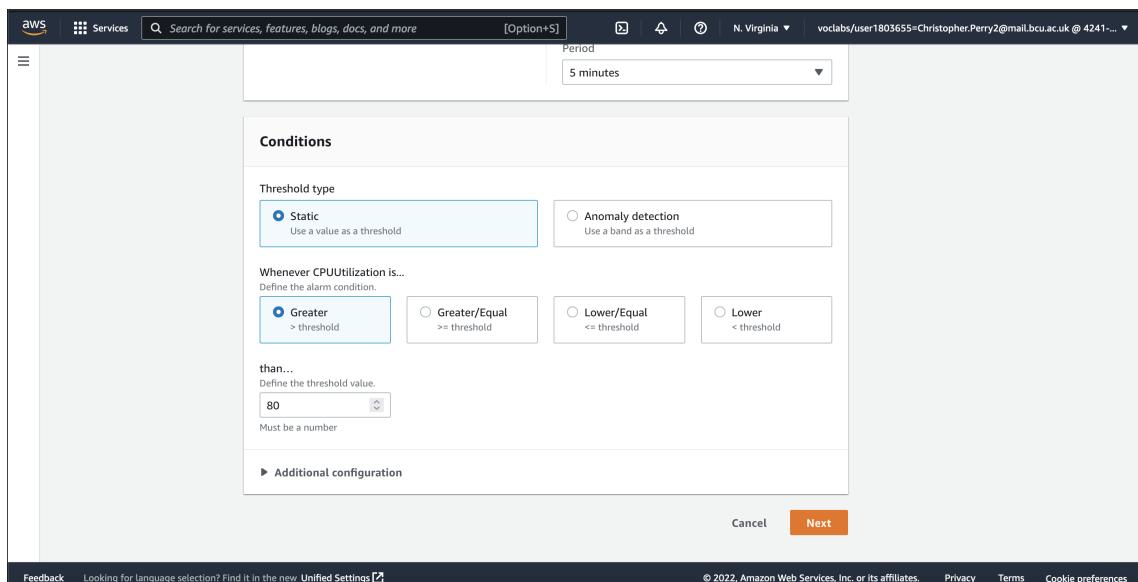


Figure A.4: CloudWatch Conditions

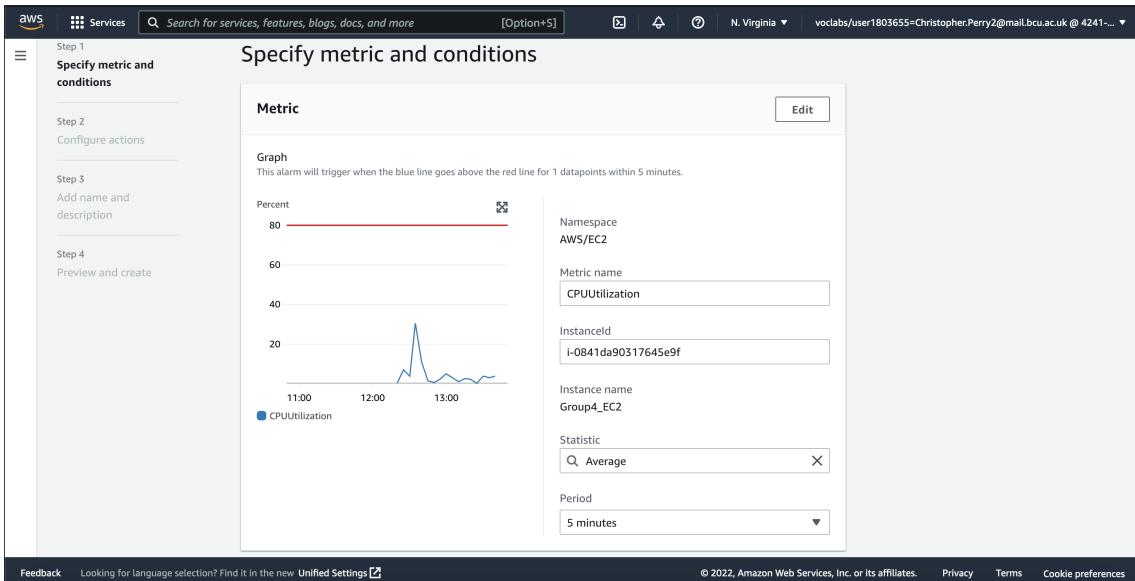


Figure A.5: CloudWatch Specify Metric

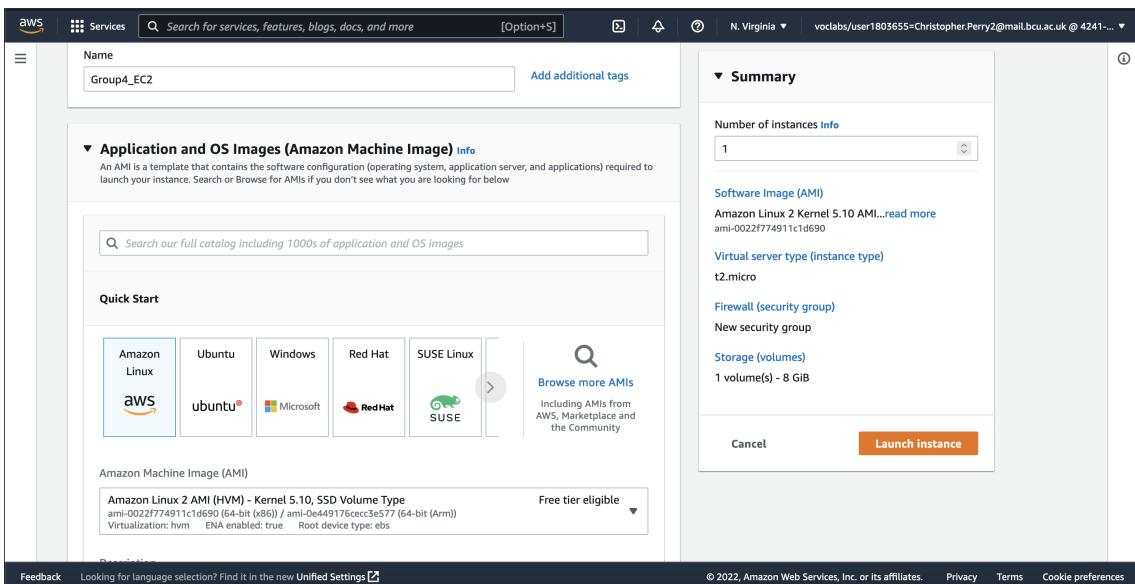


Figure A.6: Create Instance - Application and OS Images

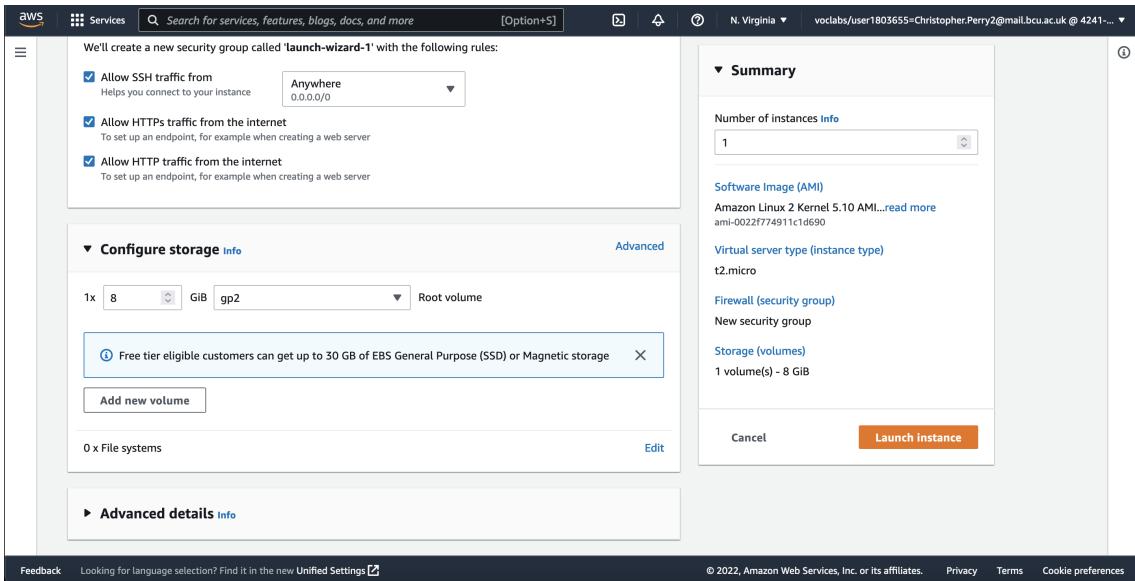


Figure A.7: Create Instance - Configure Storage

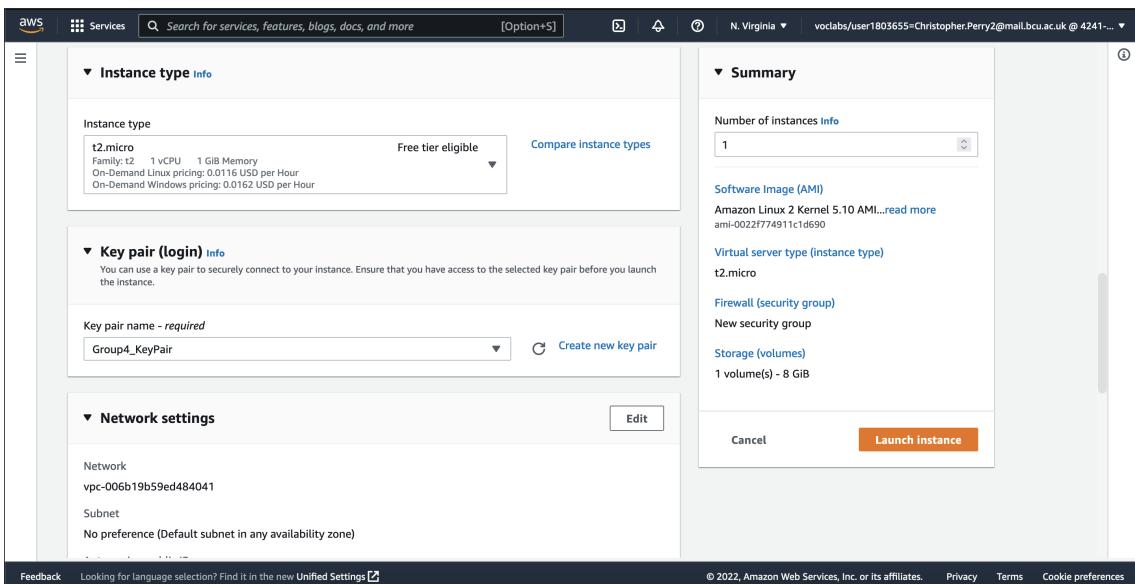


Figure A.8: Create Instance - Instance Type

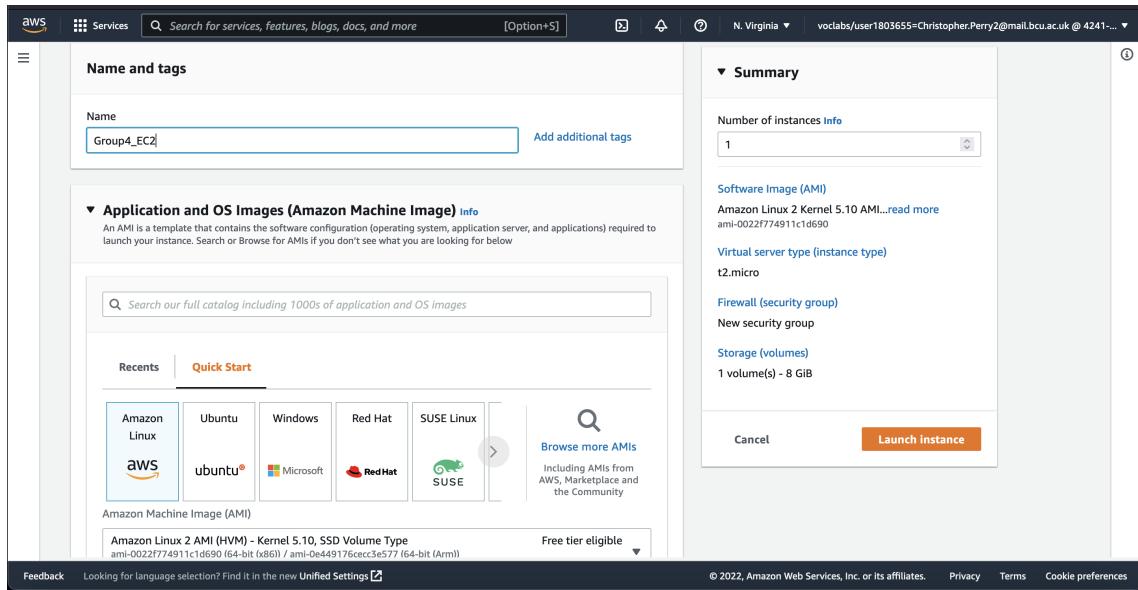


Figure A.9: Create Instance - Name & Tags

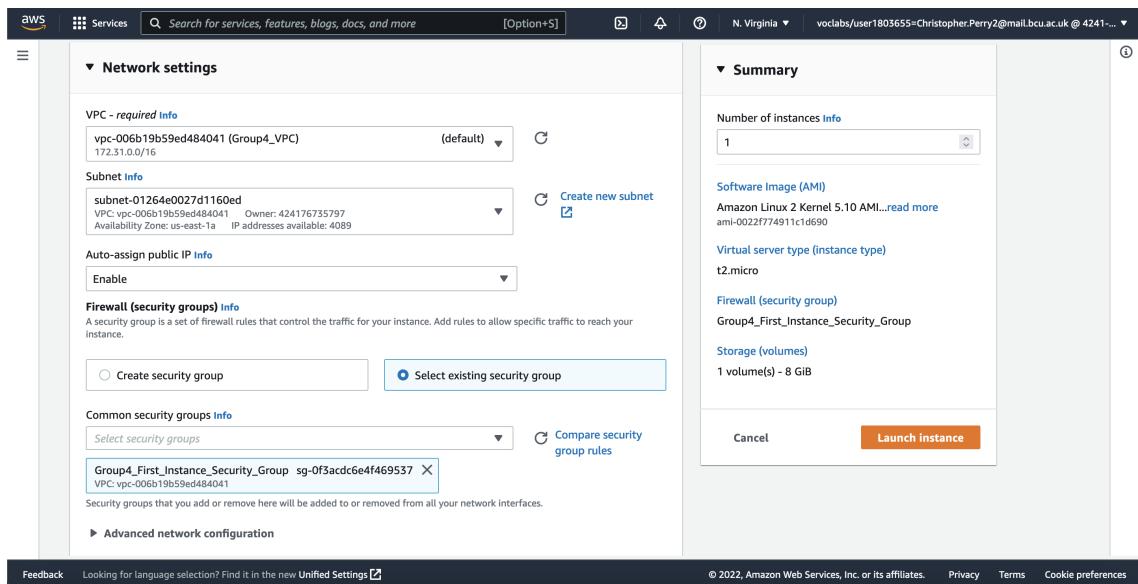


Figure A.10: Create Instance - Network Settings

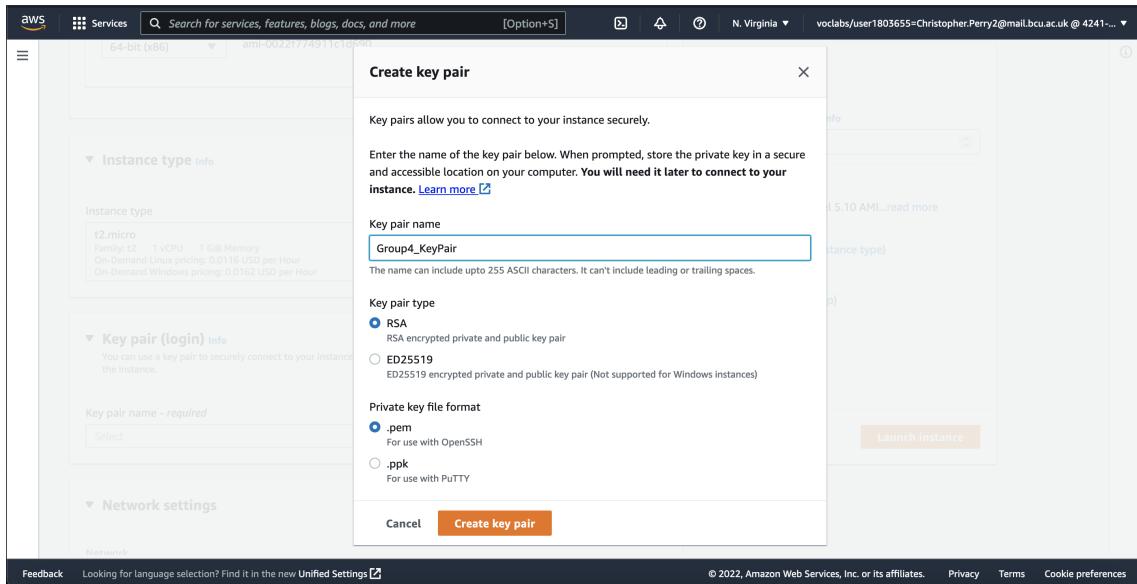


Figure A.11: Creating Key Pair

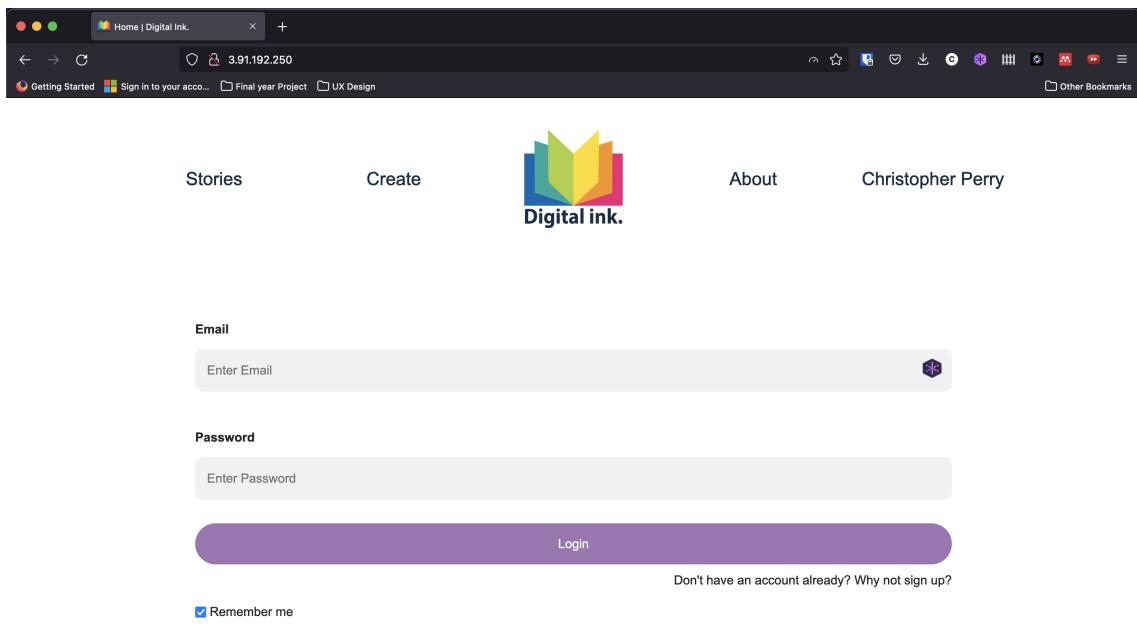


Figure A.12: Digital Ink

```
ec2-user@ip-172-31-27-208:~/digital-ink

: phpmyadmin Pulling
: 5eb5b503b376 Extracting      27.53MB/31.37MB          5.9s
: 8b1a0d84cf101 Download complete                         5.7s
: 38c937addeb7 Download complete                         5.7s
: 6a2f1dc96e59 Download complete                         5.7s
: f8c3f82c39d4 Download complete                         5.7s
: 90fc6462b088 Download complete                         5.7s
[+] Running 0/319 Download complete
: phpmyadmin Pulling
: 5eb5b503b376 Extracting      27.53MB/31.37MB          6.0s
: 8b1a0d84cf101 Download complete                         5.8s
: 38c937addeb7 Download complete                         5.8s
: 6a2f1dc96e59 Download complete                         5.8s
: f8c3f82c39d4 Download complete                         5.8s
: 90fc6462b088 Download complete                         5.8s
[+] Running 7/319 Download complete
: phpmyadmin Pulling
: 5eb5b503b376 Pull complete
: 8b1a0d84cf101 Pull complete
: 38c937addeb7 Extracting      [=====] 32.31MB/91.6MB
: 6a2f1dc96e59 Download complete                         12.8s
: f8c3f82c39d4 Download complete                         12.8s
: 90fc6462b088 Download complete                         12.8s
: c670d99116c9 Download complete                         12.8s
: 268554d6fe96 Download complete                         12.8s
: 6c29fa0d4492 Download complete                         12.8s
: 73c23c0a259 Download complete                         12.8s
: 81ac13c96fc2 Download complete                         12.8s
: b60a3e6c23949 Download complete                         12.8s
: dac5dd67fd59 Download complete                         12.8s
: fd46866d9c36 Download complete                         12.8s
: 443a80ef4c80 Download complete                         12.8s
: 5e0049224f95 Download complete                         12.8s
: 213e66cdf7f56 Download complete                         12.8s
: 9b9b44731108 Download complete                         12.8s
[+] db Pulling
: 15bb3e15f562 Pull complete
: 96c2ab037a1b Pull complete
: 8aa3ac85066b Pull complete
: ac7e524fc98 Pull complete
: f6a88631064f Pull complete
: 15bb3e3cf1f50 Extracting      [=====] 13.11MB/14.06MB
: ae65dc337dc8 Download complete                         12.7s
: ad4c4c43adaf5f2 Download complete                         12.7s
: c6cab33e8ff1 Download complete                         12.7s
: 2e1cf2c43f16 Download complete                         12.7s
: 2e5ee322aaf48 Download complete                         12.7s
```

Figure A.13: Docker Compose

The screenshot shows the AWS Launch Wizard interface for creating a new Amazon Linux 2 instance. The 'Network settings' section includes a subnet (vpc-006b19b59ed484041) and security group rules for SSH, HTTP, and HTTPS traffic. The 'Summary' section shows one instance, the AMI (Amazon Linux 2 Kernel 5.10 AMI), and the instance type (t2.micro). The 'Configure storage' section shows a root volume of 8 GiB. At the bottom, there are 'Cancel' and 'Launch instance' buttons.

aws Services Search for services, features, blogs, docs, and more [Option+Shift]

☰ N. Virginia v vclabs/user1803655=Christopher.Perry2@mail.bcu.ac.uk @ 4241...

**▼ Network settings**

Network  
vpc-006b19b59ed484041

Subnet  
No preference (Default subnet in any availability zone)

Auto-assign public IP  
Enable

Security groups (Firewall) [Info](#)  
We'll create a new security group called 'launch-wizard-1' with the following rules:

Allow SSH traffic from Anywhere  
Helps you connect to your instance 0.0.0.0/0

Allow HTTPs traffic from the internet  
To set up an endpoint, for example when creating a web server

Allow HTTP traffic from the internet  
To set up an endpoint, for example when creating a web server

**▼ Summary**

Number of instances [Info](#)  
1

Software Image (AMI)  
Amazon Linux 2 Kernel 5.10 AMI...read more  
ami-0022f774911c1d690

Virtual server type (instance type)  
t2.micro

Firewall (security group)  
New security group

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

[Cancel](#) [Launch instance](#)

**▼ Configure storage** [Info](#) Advanced

1x 8 GiB gp2 Root volume

Feedback Looking for language selection? Find it in the new [Unified Settings](#)

© 2022, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Figure A.14: Edit Instance - Network Settings

```
[ec2-user@ip-172-31-27-208 ~]$ sudo yum update
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core
[ec2-user@ip-172-31-27-208 ~]$ sudo yum install docker docker-compose
Loaded plugins: extras_suggestions, langpacks, 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: runc >= 1.0.0 for package: docker-20.10.13-2.amzn2.x86_64
--> Processing Dependency: libcgroup >= 0.40.rcl5-15 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 libcgroup.x86_64 0:0.41-21.amzn2 will be installed
--> Package pigz.x86_64 0:2.3.4-1.amzn2.0.1 will be installed
--> Package runc.x86_64 0:1.0.3-2.amzn2 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====


| Package                                                                                               | Arch   | Version            | Repository        | Size  |
|-------------------------------------------------------------------------------------------------------|--------|--------------------|-------------------|-------|
| <hr/>                                                                                                 |        |                    |                   |       |
| Installing:                                                                                           |        |                    |                   |       |
| docker                                                                                                | x86_64 | 20.10.13-2.amzn2   | amzn2extra-docker | 40 M  |
| Installing for dependencies:                                                                          |        |                    |                   |       |
| containerd                                                                                            | x86_64 | 1.4.13-2.amzn2.0.1 | amzn2extra-docker | 23 M  |
| libcgroup                                                                                             | x86_64 | 0.41-21.amzn2      | amzn2-core        | 66 k  |
| pigz                                                                                                  | x86_64 | 2.3.4-1.amzn2.0.1  | amzn2-core        | 81 k  |
| runc                                                                                                  | x86_64 | 1.0.3-2.amzn2      | amzn2extra-docker | 3.0 M |
| <hr/>                                                                                                 |        |                    |                   |       |
| Transaction Summary                                                                                   |        |                    |                   |       |
| <hr/>                                                                                                 |        |                    |                   |       |
| Install 1 Package (<4 Dependent packages)                                                             |        |                    |                   |       |
| Total download size: 67 M                                                                             |        |                    |                   |       |
| Installed size: 280 M                                                                                 |        |                    |                   |       |
| Is this ok [y/N]:                                                                                     |        |                    |                   |       |
| snuffle   ↵   □ ~   % ssh + fish + bash + zsh + fish + fish + fish + fish + fish + fish   □ spacedust |        |                    |                   |       |
| 54% ━━━━━━   5.2 GB ━━━━━━   0 18% ━━   ⏳ 5-04, 1:31 PM                                               |        |                    |                   |       |


```

Figure A.15: Installing Docker using Package Manager

```
Resolving Dependencies
--> Running transaction check
--> Package docker.x86_64 0:20.10.13-2.amzn2 will be installed
--> Processing Dependency: runc >= 1.0.0 for package: docker-20.10.13-2.amzn2.x86_64
--> Processing Dependency: libcgroup >= 0.40.rcl5-15 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 libcgroup.x86_64 0:0.41-21.amzn2 will be installed
--> Package pigz.x86_64 0:2.3.4-1.amzn2.0.1 will be installed
--> Package runc.x86_64 0:1.0.3-2.amzn2 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====


| Package                                                                                               | Arch   | Version            | Repository        | Size                     |
|-------------------------------------------------------------------------------------------------------|--------|--------------------|-------------------|--------------------------|
| <hr/>                                                                                                 |        |                    |                   |                          |
| Installing:                                                                                           |        |                    |                   |                          |
| docker                                                                                                | x86_64 | 20.10.13-2.amzn2   | amzn2extra-docker | 40 M                     |
| Installing for dependencies:                                                                          |        |                    |                   |                          |
| containerd                                                                                            | x86_64 | 1.4.13-2.amzn2.0.1 | amzn2extra-docker | 23 M                     |
| libcgroup                                                                                             | x86_64 | 0.41-21.amzn2      | amzn2-core        | 66 k                     |
| pigz                                                                                                  | x86_64 | 2.3.4-1.amzn2.0.1  | amzn2-core        | 81 k                     |
| runc                                                                                                  | x86_64 | 1.0.3-2.amzn2      | amzn2extra-docker | 3.0 M                    |
| <hr/>                                                                                                 |        |                    |                   |                          |
| Transaction Summary                                                                                   |        |                    |                   |                          |
| <hr/>                                                                                                 |        |                    |                   |                          |
| Install 1 Package (<4 Dependent packages)                                                             |        |                    |                   |                          |
| Total download size: 67 M                                                                             |        |                    |                   |                          |
| Installed size: 280 M                                                                                 |        |                    |                   |                          |
| Is this ok [D/y/N]: y                                                                                 |        |                    |                   |                          |
| Downloading packages:                                                                                 |        |                    |                   |                          |
| (1/5): libcgroup-0.41-21.amzn2.x86_64.rpm                                                             |        |                    |                   | 1 66 kB 00:00:00         |
| (2/5): pigz-2.3.4-1.amzn2.0.1.x86_64.rpm                                                              |        |                    |                   | 1 81 kB 00:00:00         |
| (3/5): containerd-1.4.13-2.amzn2.0.1.x86_64.rpm                                                       |        |                    |                   | 1 23 MB 00:00:00         |
| (4/5): docker-20.10.13-2.amzn2.x86_64.rpm                                                             |        |                    |                   | 1 40 MB 00:00:00         |
| (5/5): runc-1.0.3-2.amzn2.x86_64.rpm                                                                  |        |                    |                   | 1 3.0 MB 00:00:00        |
| <hr/>                                                                                                 |        |                    |                   |                          |
| Total                                                                                                 |        |                    |                   | 66 MB/s   67 MB 00:00:01 |
| Running transaction check                                                                             |        |                    |                   |                          |
| Running transaction test                                                                              |        |                    |                   |                          |
| Transaction test succeeded                                                                            |        |                    |                   |                          |
| Running transaction                                                                                   |        |                    |                   |                          |
| Installing : runc-1.0.3-2.amzn2.x86_64                                                                |        |                    |                   | 1/5                      |
| Installing : containerd-1.4.13-2.amzn2.0.1.x86_64                                                     |        |                    |                   | 2/5                      |
| Installing : libcgroup-0.41-21.amzn2.x86_64                                                           |        |                    |                   | 3/5                      |
| Installing : pigz-2.3.4-1.amzn2.0.1.x86_64                                                            |        |                    |                   | 4/5                      |
| Installing : docker-20.10.13-2.amzn2.x86_64 [##### <td></td> <td></td> <td></td> <td>5/5]</td>        |        |                    |                   | 5/5]                     |
| snuffle   ↵   □ ~   % ssh + fish + bash + zsh + fish + fish + fish + fish + fish + fish   □ spacedust |        |                    |                   |                          |
| 74% ━━━━━━   5.3 GB ━━━━━━   0 18% ━━   ⏳ 5-04, 1:31 PM                                               |        |                    |                   |                          |


```

Figure A.16: Installing Docker using Package Manager (In Progress)

```
git-core x86_64 2.32.0-1.amzn2.0.1 amzn2-core 2.7 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:
=====
(G/7): emacs-fs/filesystem-27.2-2.x86_64.amzn2.0.1.noarch.rpm | 67 kB 00:00:00
(G/7): git-2.32.0-1.amzn2.0.1.x86_64.rpm | 126 kB 00:00:00
(G/7): git-core-doc-2.32.0-1.amzn2.0.1.noarch.rpm | 2.7 MB 00:00:00
(G/7): perl-Error-0.17020-2.amzn2.noarch.rpm | 32 kB 00:00:00
(G/7): perl-Git-2.32.0-1.amzn2.0.1.noarch.rpm | 43 kB 00:00:00
(G/7): git-core-2.32.0-1.amzn2.0.1.x86_64.rpm | 4.8 MB 00:00:00
(G/7): perl-TermReadKey-2.30-20.amzn2.0.2.x86_64.rpm | 31 kB 00:00:00
=====
29 MB/s | 7.8 MB 00:00:00

Total
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
=====
Installing : git-core-2.32.0-1.amzn2.0.1.x86_64 1/7
Installing : git-core-doc-2.32.0-1.amzn2.0.1.noarch 2/7
Installing : 1:perl-Error-0.17020-2.amzn2.noarch 3/7
Installing : 1:emacs-fs/filesystem-27.2-2.x86_64.amzn2.0.1.noarch 4/7
Installing : perl-TermReadKey-2.30-20.amzn2.0.2.x86_64 5/7
Installing : perl-Git-2.32.0-1.amzn2.0.1.noarch 6/7
Installing : git-core-2.32.0-1.amzn2.0.1.x86_64 7/7
Verifying : perl-TermReadKey-2.30-20.amzn2.0.2.x86_64 1/7
Verifying : git-core-doc-2.32.0-1.amzn2.0.1.noarch 2/7
Verifying : perl-Git-2.32.0-1.amzn2.0.1.noarch 3/7
Verifying : 1:emacs-fs/filesystem-27.2-2.x86_64.amzn2.0.1.noarch 4/7
Verifying : git-2.32.0-1.amzn2.0.1.x86_64 5/7
Verifying : git-core-2.32.0-1.amzn2.0.1.x86_64 6/7
Verifying : 1:perl-Error-0.17020-2.amzn2.noarch 7/7

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

Dependency Installed:
emacs-fs/filesystem.noarch 1:27.2-4.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 ~]$ █
& snuffleupug □ ⟲ ls ssh • fish • bash • zsh • fish • fish • fish • fish -fslsh | spacedust |
```

Figure A.17: Installing Git

Figure A.18: Starting Docker systemd Service

The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with links like EC2 Dashboard, EC2 Global View, Events, Tags, Limits, Instances (with sub-links for Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Scheduled Instances, Capacity Reservations), and Images (with sub-links for AMIs and AMI Catalog). The main content area has a header 'Instances (2) Info' with buttons for Connect, Actions, and Launch Instances. A search bar is at the top. Below it is a table with columns: Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IP. Two rows are listed: 'Group4\_EC2' (terminated, t2.micro, no alarms, us-east-1d) and 'Group4\_EC2' (running, t2.micro, no alarms, us-east-1d). At the bottom, there's a 'Select an instance' section and a footer with copyright information.

Figure A.19: Instances

The screenshot shows the 'Launching instance' step in the AWS EC2 instance launch process. It includes a progress bar at 80% completion, a 'Launch initiation' status, and a 'Details' link. A message at the top says 'You've been opted into the new launch experience. Find out more about this experience or send us feedback. You can still return to the previous version by opting-out.' There's also an 'Opt-out to the old experience' button. The footer is identical to Figure A.19.

Figure A.20: Launching Instance

```

ec2-user@ip-172-31-27-208:~/digital-ink
logout
Connection to 3.91.192.250 closed.
> ssh -i ~/Desktop/Group4_KeyPair.pem ec2-user@3.91.192.250
Last login: Wed May  4 13:21:09 2022 from 193.60.143.12
  _\|_ _\|_
 _\| (   /  Amazon Linux 2 AMI
  _\|_\|_|_|
https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-27-208 ~]$ cd digital-ink
[ec2-user@ip-172-31-27-208 digital-ink]$ cd ..
[ec2-user@ip-172-31-27-208 ~]$ sudo mv digital-ink digital-ink.old
[ec2-user@ip-172-31-27-208 ~]$ git clone https://github.com/ChrisP99/digital-ink.git
Cloning into 'digital-ink'...
remote: Enumerating objects: 9909, done.
remote: Counting objects: 100% (9909/9909), done.
remote: Compressing objects: 100% (6382/6382), done.
remote: Total 9909 (delta 3080), reused 9863 (delta 3034), pack-reused 0
Receiving objects: 100% (9909/9909), 17.59 MiB | 14.30 MiB/s, done.
Resolving deltas: 100% (3080/3080), done.
Updating files: 100% (8963/8963), done.
[ec2-user@ip-172-31-27-208 ~]$ cd digital-ink
[ec2-user@ip-172-31-27-208 digital-ink]$ sudo /usr/local/bin/docker-compose up -d
[*] Running 3/3
  • Container digital-ink-db-1  Running
  • Container phpmyadmin  Running
  • Container digital-ink-app-1 Started
[ec2-user@ip-172-31-27-208 digital-ink]$ sudo /usr/local/bin/docker-compose down
[*] Running 4/4
  • Container phpmyadmin  Removed
  • Container digital-ink-app-1 Removed
  • Container digital-ink-db-1 Removed
  • Network digital-ink_default Removed
[ec2-user@ip-172-31-27-208 digital-ink]$ sudo /usr/local/bin/docker-compose up -d
[*] Running 4/4
  • Network digital-ink_default Created
  • Container digital-ink-db-1 Started
  • Container phpmyadmin  Started
  • Container digital-ink-app-1 Started
[ec2-user@ip-172-31-27-208 digital-ink]$ sudo /usr/local/bin/docker-compose exec app bash
root@c565ab9c37ff:/srv/app# php artisan migrate
Nothing to migrate.
root@c565ab9c37ff:/srv/app# 

```

Figure A.21: Log In with Key Pair

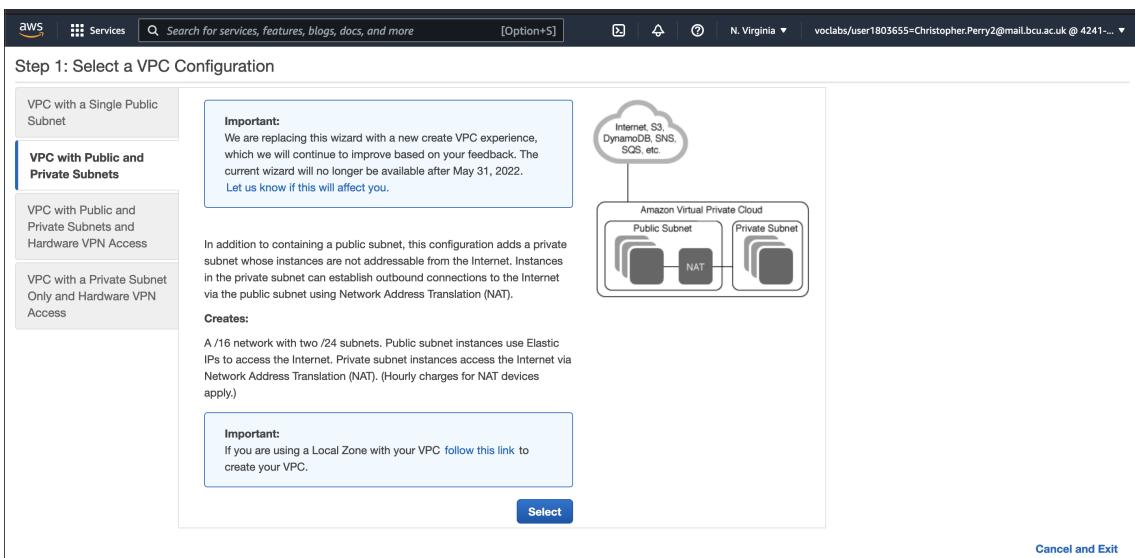


Figure A.22: Selecting a VPC Configuration

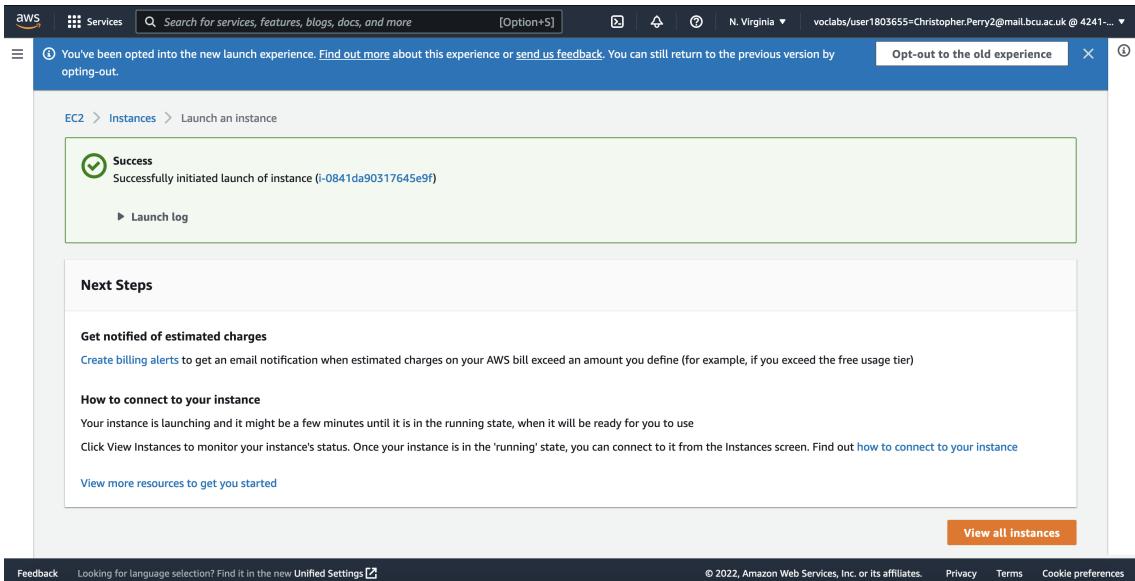


Figure A.23: Successfully Initiated Instance

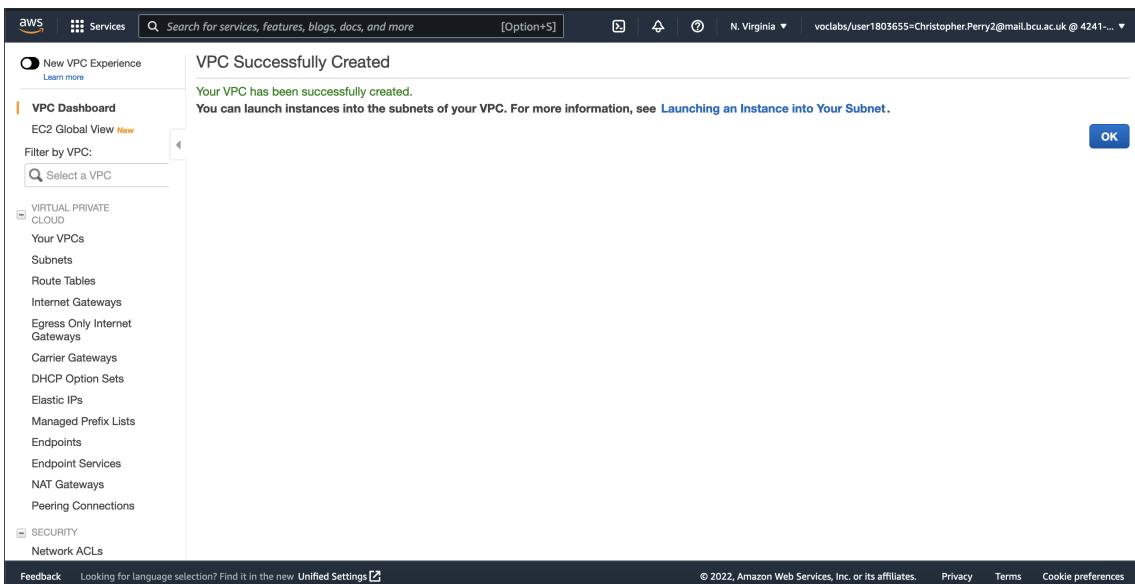


Figure A.24: VPC Successfully Created

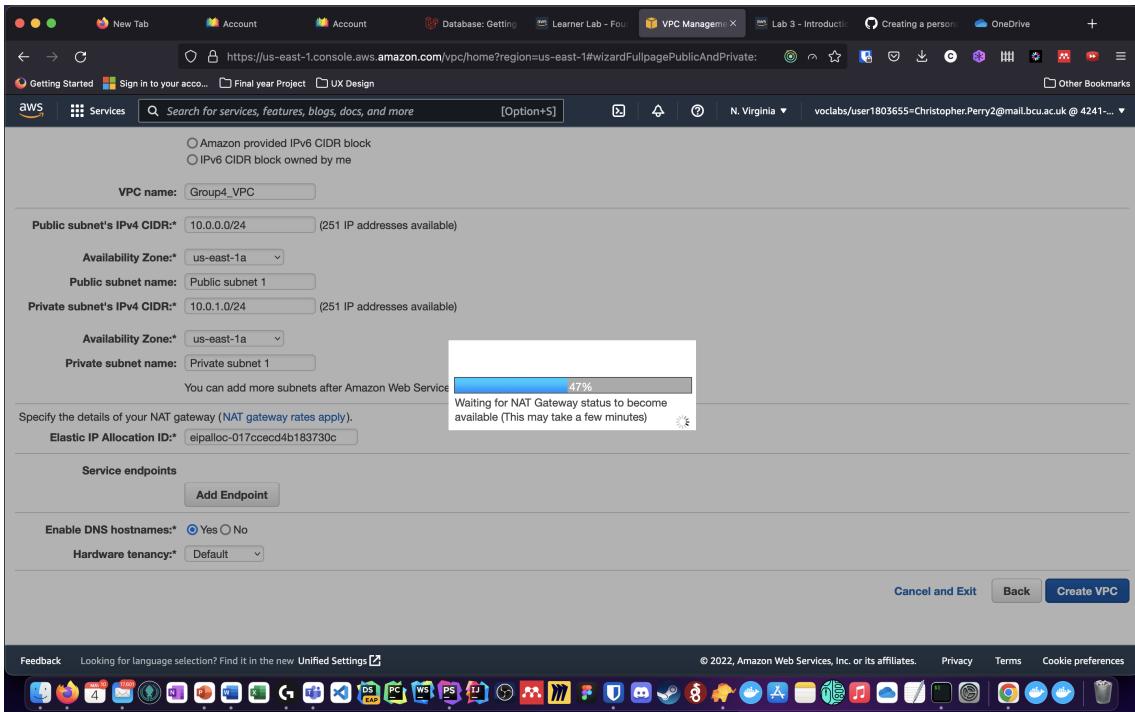


Figure A.25: VPC with Public and Private Subnets, Loading

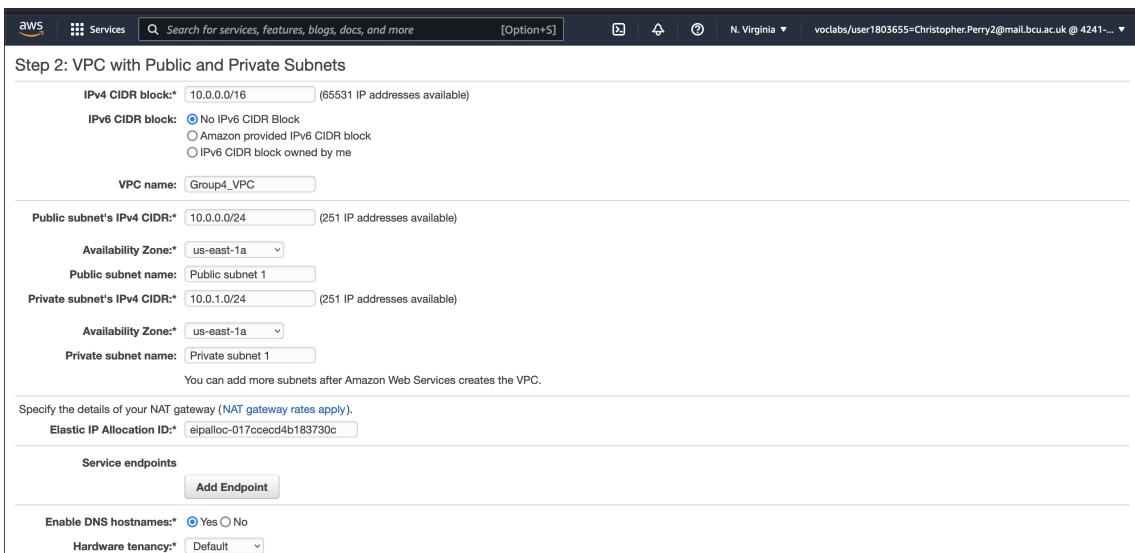


Figure A.26: VPC with Public and Private Subnets

Your VPCs (2) info

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	-

vpc-07657585bc0e3b3b5

Details    CIDs    Flow logs    Tags

**Details**

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

Feedback    Looking for language selection? Find it in the new Unified Settings [?](#)

Figure A.27: Your VPCs