

# Cloud Computing with AWS

By Adam Cordner, Chris Perry & Evie Snuffle

A descriptive 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</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Web App</b>	<b>2</b>
<b>3</b>	<b>VPC and Subnets</b>	<b>7</b>
<b>4</b>	<b>EC2</b>	<b>8</b>
<b>5</b>	<b>S3</b>	<b>11</b>
<b>6</b>	<b>CloudFront</b>	<b>12</b>
<b>7</b>	<b>CloudWatch</b>	<b>13</b>
<b>8</b>	<b>CloudTrail</b>	<b>14</b>
<b>9</b>	<b>Server and Database</b>	<b>15</b>
<b>10</b>	<b>Availability Zones</b>	<b>16</b>
<b>11</b>	<b>Elastic Load Balancing</b>	<b>17</b>
<b>12</b>	<b>Security Practices</b>	<b>18</b>
<b>13</b>	<b>Cost Breakdown</b>	<b>19</b>
13.1	Estimated Costs . . . . .	19
13.2	Scaling Up to 10,000 Users . . . . .	19
13.3	Scaling Up to 1 Million Users . . . . .	19
13.4	Scaling Up to 10 Million Users . . . . .	19
<b>14</b>	<b>Testing</b>	<b>20</b>
14.1	Testing EC2 . . . . .	20
14.2	Testing S3 . . . . .	20
14.3	Testing CloudFront . . . . .	20
14.4	Testing Server and Database . . . . .	20
14.5	Testing CloudWatch . . . . .	20
<b>15</b>	<b>Future Enhancements</b>	<b>21</b>
<b>16</b>	<b>Conclusion</b>	<b>22</b>

<b>A Screenshots</b>	<b>23</b>
<b>Bibliography</b>	<b>23</b>

# List of Figures

2.1 Database tables overview . . . . .	2
2.2 migrations table . . . . .	3
2.3 users table . . . . .	3
2.4 stories table . . . . .	4
2.5 <i>Digital-Ink</i> home page log in and sign up forms . . . . .	4
2.6 <i>Digital-Ink</i> story creation form . . . . .	5
2.7 <i>Digital-Ink</i> account page . . . . .	6
2.8 <i>Digital-Ink</i> stories page and story view . . . . .	6
4.1 Selection of EC2 OS Image . . . . .	8
4.2 Selection of EC2 Instance . . . . .	9
4.3 Selection of EC2 Storage Configuration . . . . .	9
A.1 After Allocating Elastic IP Address . . . . .	23
A.2 Allocating Elastic IP Address . . . . .	23
A.3 Cloning the App . . . . .	24
A.4 CloudWatch Conditions . . . . .	24
A.5 CloudWatch Specify Metric . . . . .	25
A.6 Create Instance - Application and OS Images . . . . .	25
A.7 Create Instance - Configure Storage . . . . .	26
A.8 Create Instance - Instance Type . . . . .	26
A.9 Create Instance - Name & Tags . . . . .	27
A.10 Create Instance - Network Settings . . . . .	27
A.11 Creating Key Pair . . . . .	28
A.12 Digital Ink . . . . .	28
A.13 Docker Compose . . . . .	29
A.14 Edit Instance - Network Settings . . . . .	29
A.15 Installing Docker using Package Manager . . . . .	30
A.16 Installing Docker using Package Manager (In Progress) . . . . .	30
A.17 Installing Git . . . . .	31
A.18 Starting Docker systemd Service . . . . .	31
A.19 Instances . . . . .	32
A.20 Launching Instance . . . . .	32
A.21 Log In with Key Pair . . . . .	33
A.22 Selecting a VPC Configuration . . . . .	33
A.23 Successfully Initiated Instance . . . . .	34
A.24 VPC Successfully Created . . . . .	34

A.25 VPC with Public and Private Subnets, Loading . . . . .	35
A.26 VPC with Public and Private Subnets . . . . .	35
A.27 Your VPCs . . . . .	36

# Chapter 1

## 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 2

## Web App

This chapter of the report will detail the local design and development of the *Digital Ink* web application.

*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 (?). The web app is hosted within a Docker container (?) 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 (?). MySQL is an open-source relational database management system (?) 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 (?). Additionally, all developers involved have prior experience with PHP.

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 2.1, the database which this web app uses consists of three tables: `users`, `stories`, and `migrations`.

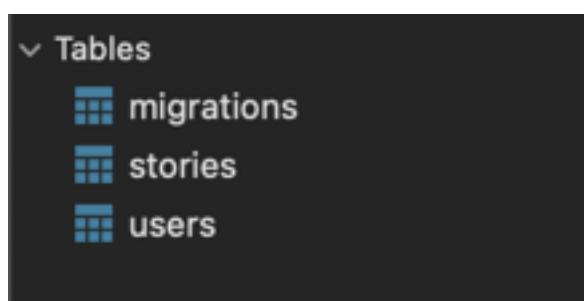


Figure 2.1: Database tables overview.

The `migrations` table (see Figure 2.2) 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.
- `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 2.2: `migrations` table.

The `users` table (see Figure 2.3) 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 (?).
- `remember_token`: keeps the user logged into the device if the user selects "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.

stories							users		
<code>id</code>	<code>name</code>	<code>email</code>	<code>password</code>	<code>remember_token</code>	<code>created_at</code>	<code>updated_at</code>			
1	Adam	adam.cordner@mail.bcu.ac.uk	\$2y\$10\$ISS.3eqO3o0dRXaK8sMy.eBqUapgD...	mDUIGRIiVhyEl6...	2022-05-10 15:50:27	2022-05-10 15:50:27			
2	evie	^~@snugg.ie	\$2y\$10\$vVNRmMN/RSIYTrCUjEPu44jkVI0r5...	Mu1WdyOz1e8...	2022-05-10 15:51:29	2022-05-10 15:51:29			

Figure 2.3: `users` table.

The `stories` table (see Figure 2.4) contains all the information about user-created stories, and it contains the following 11 columns:

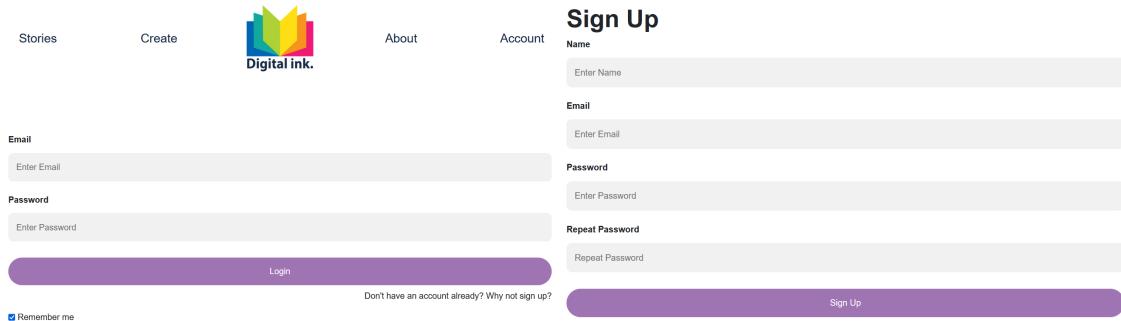
- `id`: the unique ID for each story.
- `author_id`: the unique ID associated with the user who created the story.
- `title`: the title associated with the story.
- `genre`: the genre associated with the story, which can be one of eight different genres.
- `blurb`: a brief description of the story.
- `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>	
<code>./storage/cover_images/7537329101652200606.j... images/digital-ink-logo.png</code>	NULL	1	2022-05-10 16:36:46	2022-05-10 16:36:46	
	NULL	1	2022-05-10 16:38:51	2022-05-10 16:38:51	

Figure 2.4: stories table.

The design of the web app was created using Blade, a powerful templating engine (?). When the user initially accesses the web app, they are able to log in or sign up. This can be seen in Figure 2.5. When a user has created an account, a record is written to the `users` table in the database.



The screenshot shows the Digital-Ink home page. At the top, there is a navigation bar with links for 'Stories', 'Create', 'About', and 'Account'. Below the navigation is a logo for 'Digital ink.' featuring a stylized yellow and green graphic. The main area contains two forms side-by-side. On the left is the 'Login' form, which includes fields for 'Email' (with placeholder 'Enter Email') and 'Password' (with placeholder 'Enter Password'). Below these fields is a purple 'Login' button. To the right of the login form is a 'Sign Up' form titled 'Sign Up'. It includes fields for 'Name' (placeholder 'Enter Name'), 'Email' (placeholder 'Enter Email'), 'Password' (placeholder 'Enter Password'), and 'Repeat Password' (placeholder 'Repeat Password'). Below these fields is a purple 'Sign Up' button. At the bottom center of the page, there is a link 'Don't have an account already? Why not sign up?' and a 'Remember me' checkbox.

Figure 2.5: 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 2.6. Once a story has been created, it is written to the `stories` table.

## Create your story!

The form consists of three stacked sections, each enclosed in an orange border:

- Author Reference Number:** A text input field containing the value "1".
- Title:** A text input field containing the value "Every story needs a good title!". Below it is a dropdown menu containing the value "Group 4 Loves AWS".
- Genre:** A dropdown menu containing the value "Romance".
- Your Story:** A text area containing the value "Group 4 loves using AWS' cloud features to host Digital Ink." Below it is a file input field labeled "Browse..." with the value "No file selected."
- Blurb:** A text area containing the value "Meet Group 4 and their love for AWS!". Below it is a file input field labeled "Browse..." with the value "index.jpg".
- Nearly finished!** A section with the label "Do you want to save your story as a draft or publish it onto our site?". It contains two radio button options:
  - Save as a Draft
  - Upload

A blue "Complete" button is located at the bottom right of the third section.

Figure 2.6: *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 2.7. From here, a story can be edited or deleted, which either updates a record in the `stories` table or removes a record from it.

## Hi Adam!

Yay! Your story has been published!

Here are your published stories:

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

Figure 2.7: *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 2.8.

**Stories**

Enter a title to search

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

**Group 4 Loves AWS**



Written by: 1    Genre: Romance

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

[Back](#)

Figure 2.8: *Digital-Ink* stories page and story view.

# **Chapter 3**

## **VPC and Subnets**

# Chapter 4

## EC2

After the configuration of the VPC and subnets was completed, the initial deployment of the web app began through setting up EC2. This AWS service allows for scalable computing capacity through the use of a virtual computing environment hosted in the cloud (?). The web app will be stored on an EC2 instance of Amazon Linux, known as Amazon machine images (AMIs), which wil then be launched through a docker container stored on the app.

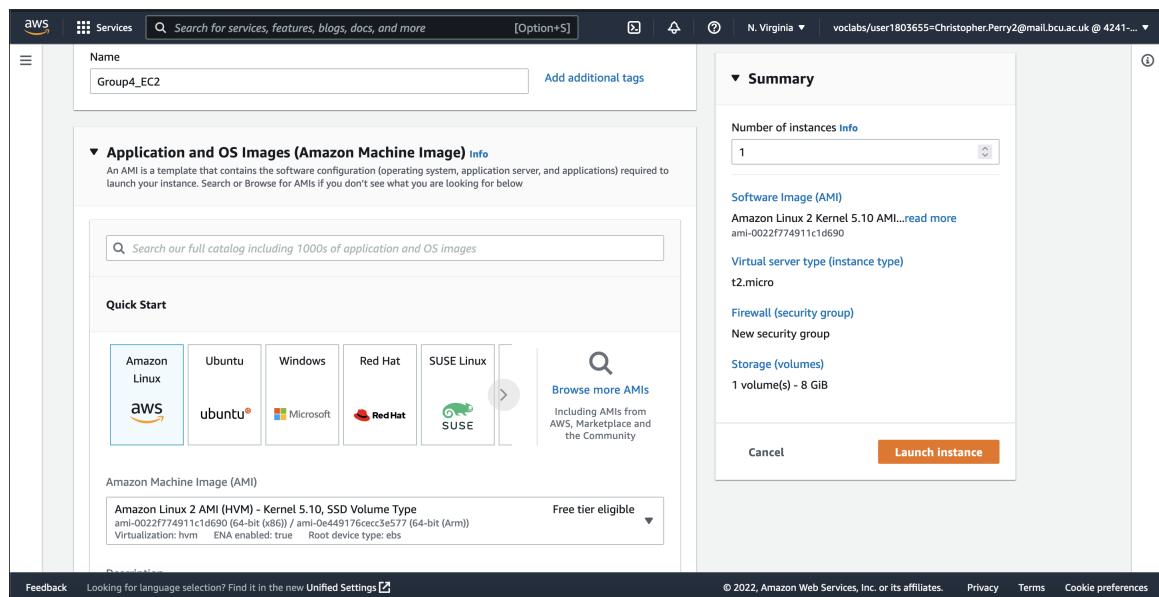


Figure 4.1: Selection of EC2 OS Image.

Figure 4.1 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 4.2.

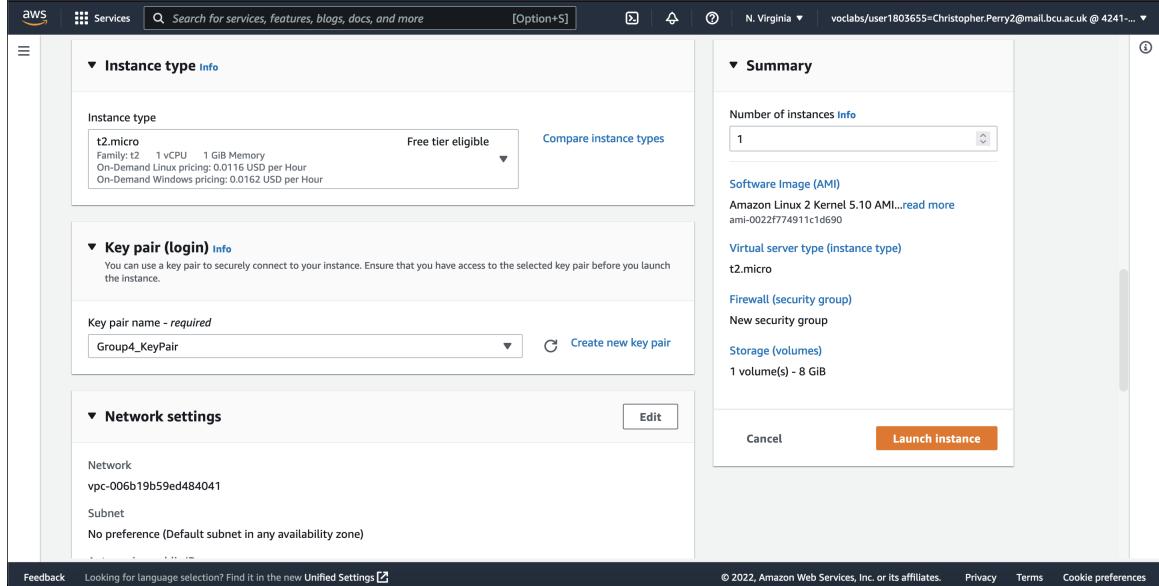


Figure 4.2: Selection of EC2 Instance.

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.

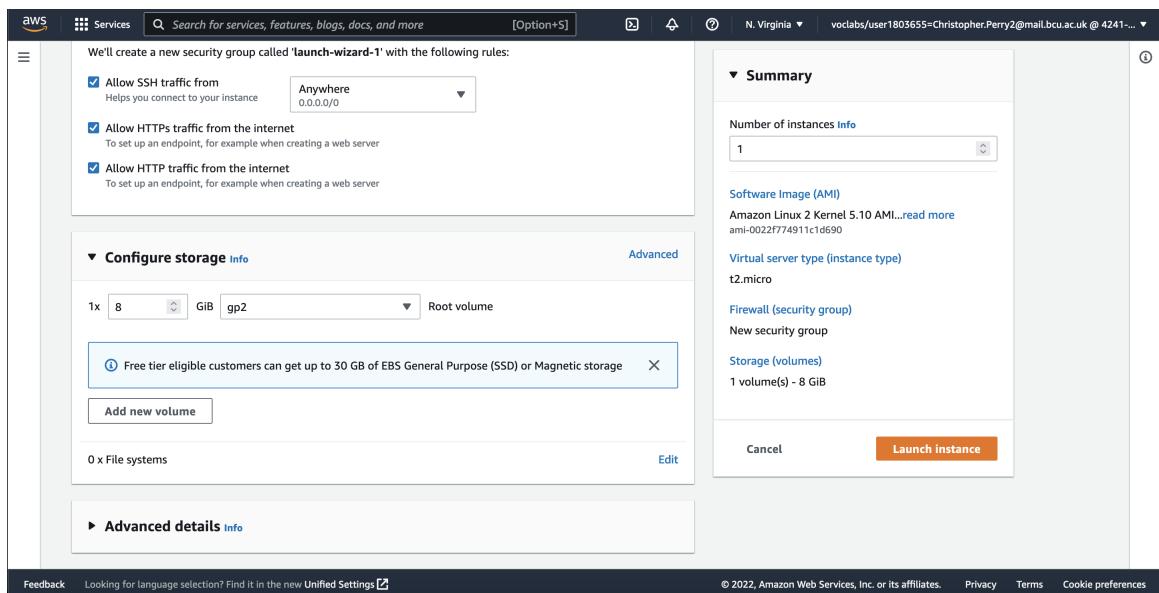


Figure 4.3: Selection of EC2 Storage Configuration.

The selection of these options can be found in Figure 4.3. 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.

Once the AMI and storage options were selected, 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.

# **Chapter 5**

**S3**

# **Chapter 6**

## **CloudFront**

# **Chapter 7**

## **CloudWatch**

# **Chapter 8**

## **CloudTrail**

## **Chapter 9**

# **Server and Database**

# **Chapter 10**

## **Availability Zones**

## **Chapter 11**

# **Elastic Load Balancing**

## **Chapter 12**

# **Security Practices**

# **Chapter 13**

## **Cost Breakdown**

- 13.1 Estimated Costs**
- 13.2 Scaling Up to 10,000 Users**
- 13.3 Scaling Up to 1 Million Users**
- 13.4 Scaling Up to 10 Million Users**

# **Chapter 14**

## **Testing**

**14.1 Testing EC2**

**14.2 Testing S3**

**14.3 Testing CloudFront**

**14.4 Testing Server and Database**

**14.5 Testing CloudWatch**

## **Chapter 15**

# **Future Enhancements**

# **Chapter 16**

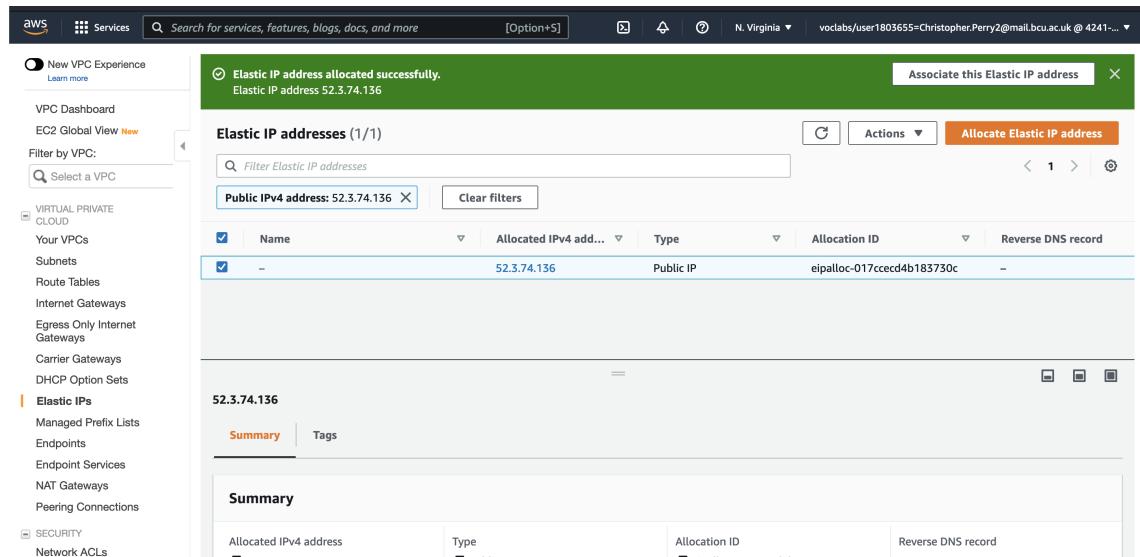
## **Conclusion**

# Bibliography

- Amazon Web Services (AWS). *What is amazon ec2?* Available from: <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html>.
- Anderson, C., 2015. Docker [software engineering]. *IEEE Software*, 32(3), pp.102–c3. Available from: <http://doi.org/10.1109/MS.2015.62>.
- Fielding, R.T. and Kaiser, G., 1997. The apache http server project. *IEEE Internet Computing*, 1(4), pp.88–90. Available from: <http://doi.org/10.1109/4236.612229>.
- Laravel, 2022a. *Blade templates*. Available from: <https://laravel.com/docs/9.x/blade>.
- Laravel, 2022b. *Hashing*. Available from: <https://laravel.com/docs/9.x/hashing>.
- Lee, J. and Ware, B., 2003. *Open source web development with lamp: Using linux, apache, mysql, perl, and php*. Addison-Wesley Professional.
- Lerdorf, R., Tatroe, K., Kaehms, B. and McGredy, R., 2002. *Programming php*. " O'Reilly Media, Inc.".
- Widenius, M., Axmark, D. and Arno, K., 2002. *Mysql reference manual: documentation from the source*. " O'Reilly Media, Inc.".

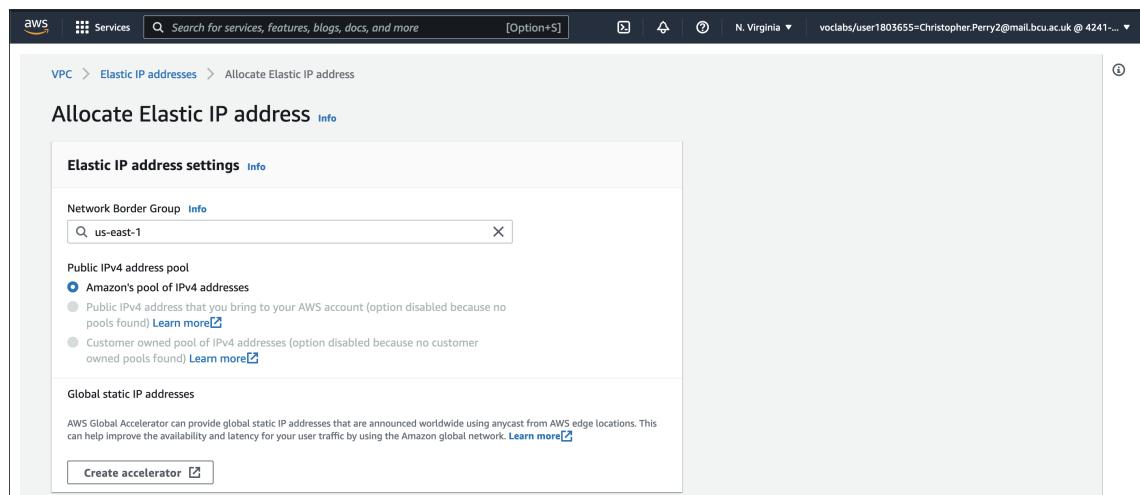
# Appendix A: Screenshots

I am an appendix, please be kind.



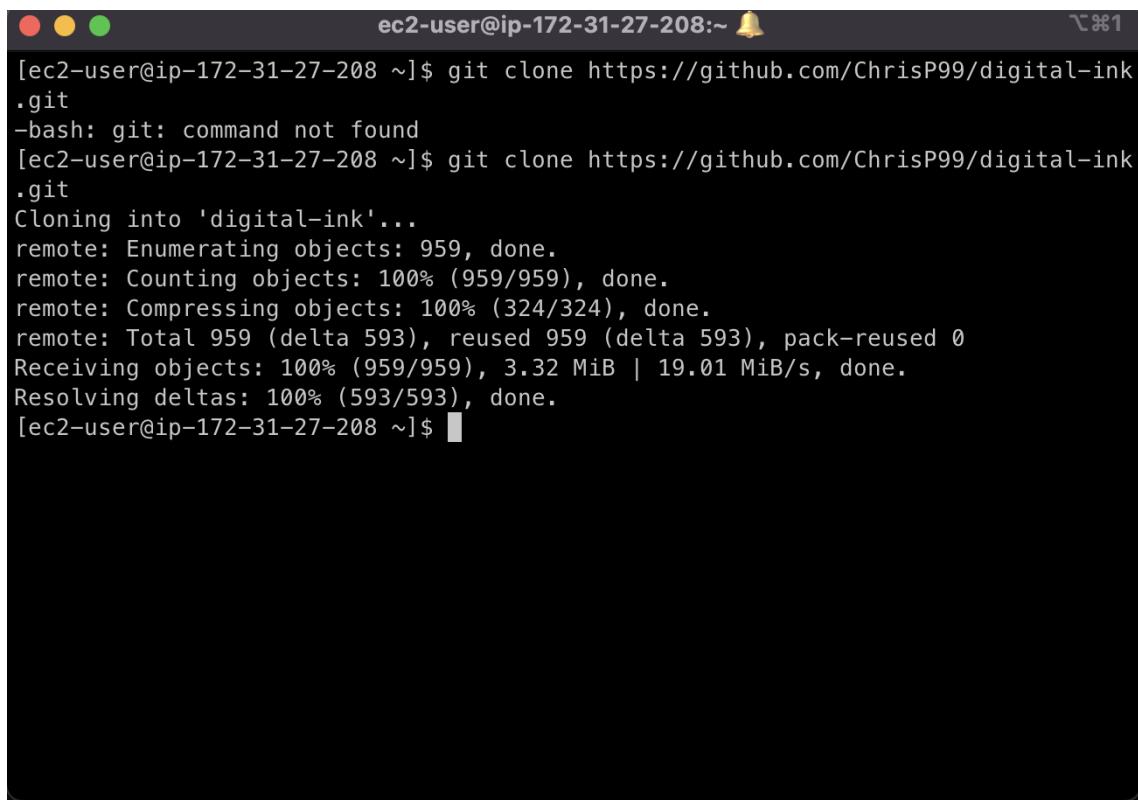
The screenshot shows the AWS VPC Elastic IP addresses page. At the top, a green banner displays the message "Elastic IP address allocated successfully." followed by the allocated IP address "52.3.74.136". Below the banner, the main interface shows a table of "Elastic IP addresses (1/1)". The table has 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 are also options for "Public IPv4 address that you bring to your AWS account" and "Customer owned pool of IPv4 addresses", both of which are disabled. 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

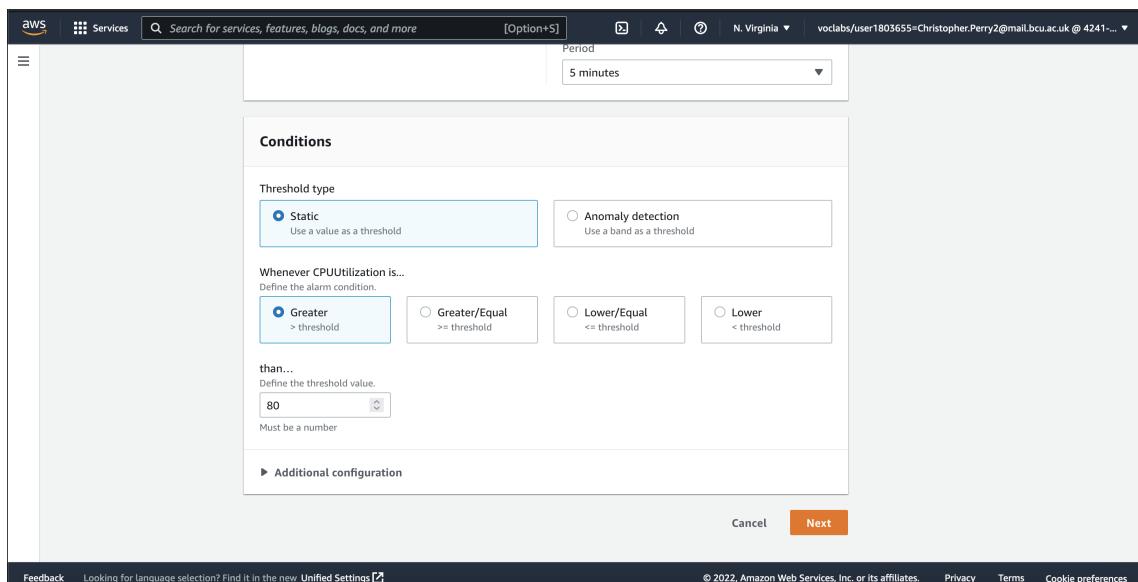


Figure A.4: CloudWatch Conditions

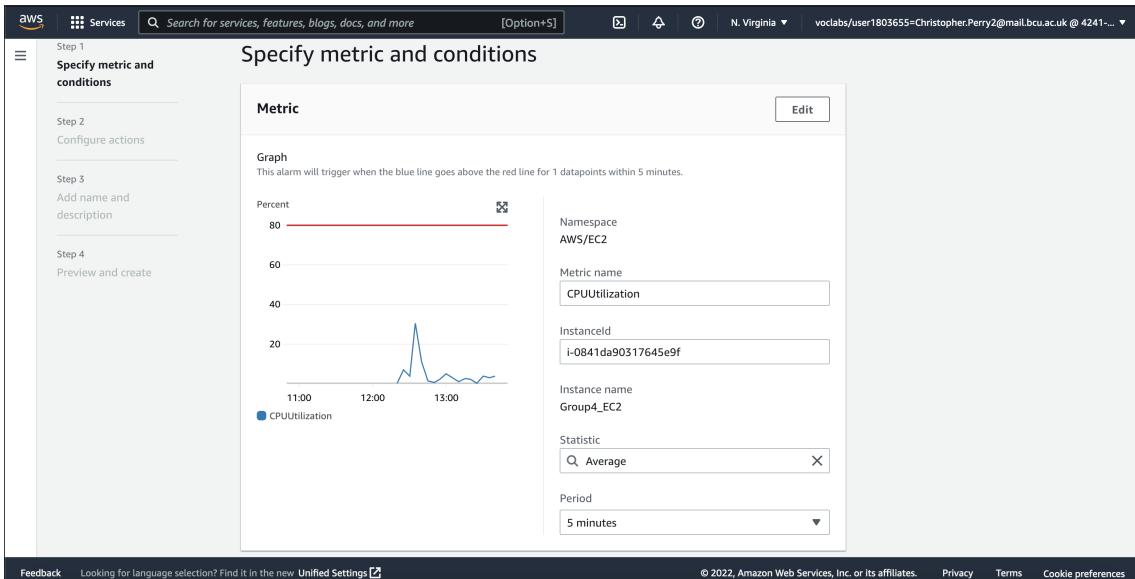


Figure A.5: CloudWatch Specify Metric

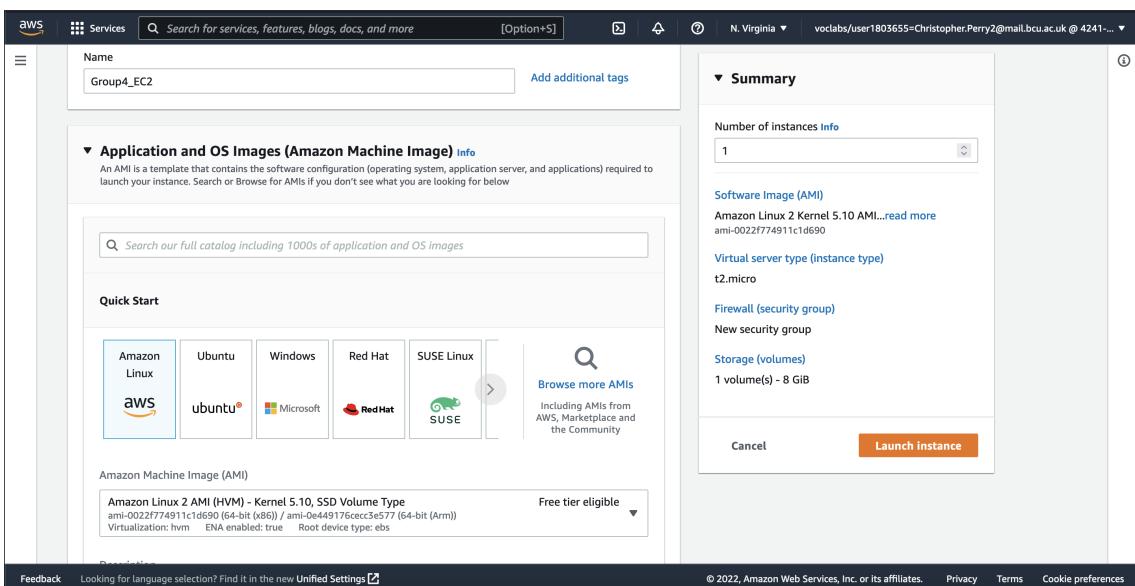


Figure A.6: Create Instance - Application and OS Images

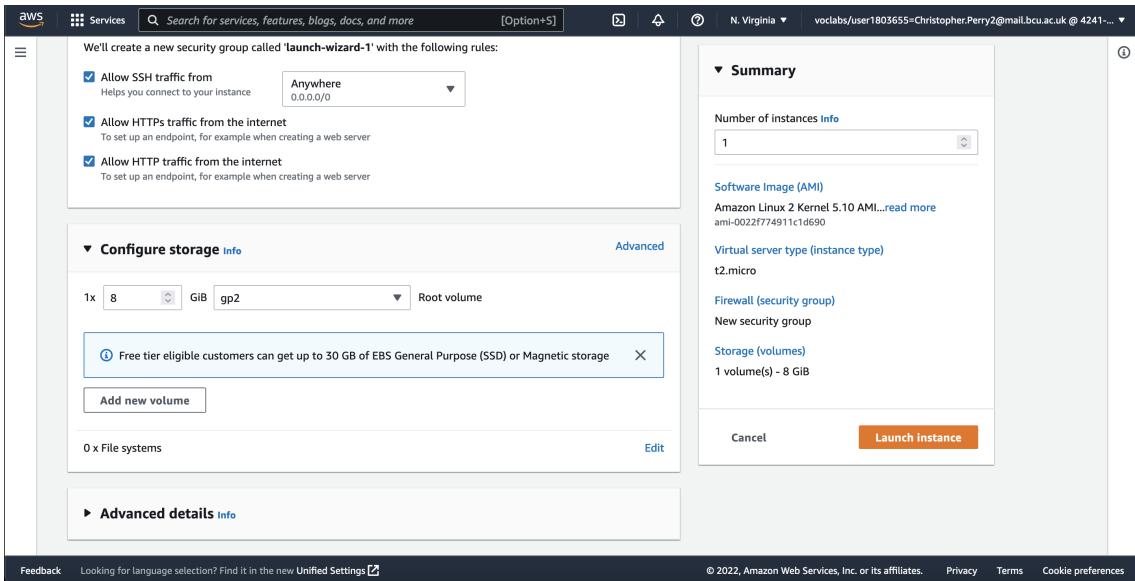


Figure A.7: Create Instance - Configure Storage

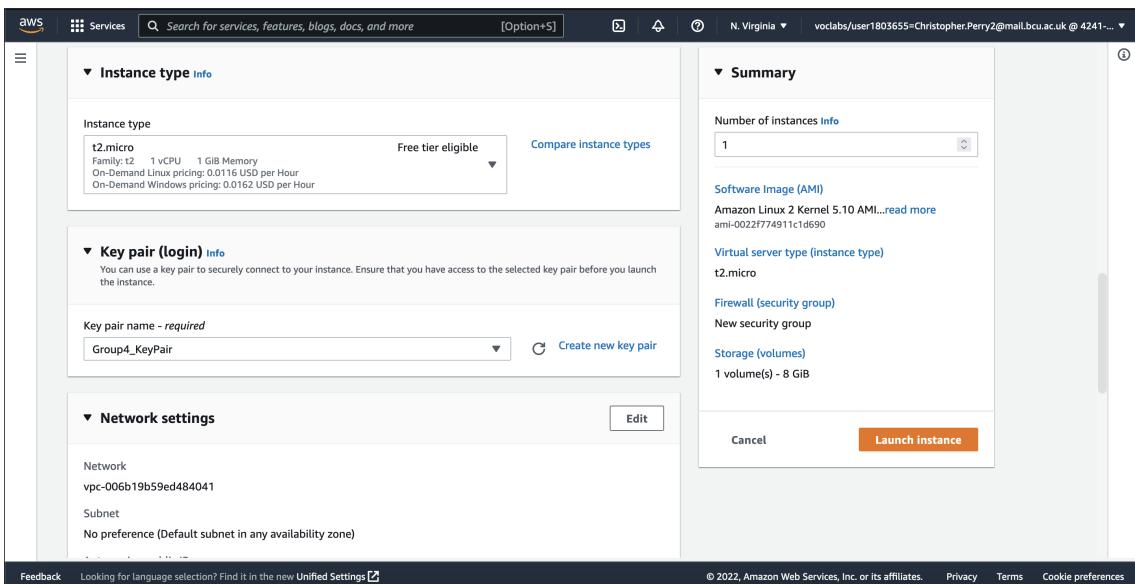


Figure A.8: Create Instance - Instance Type

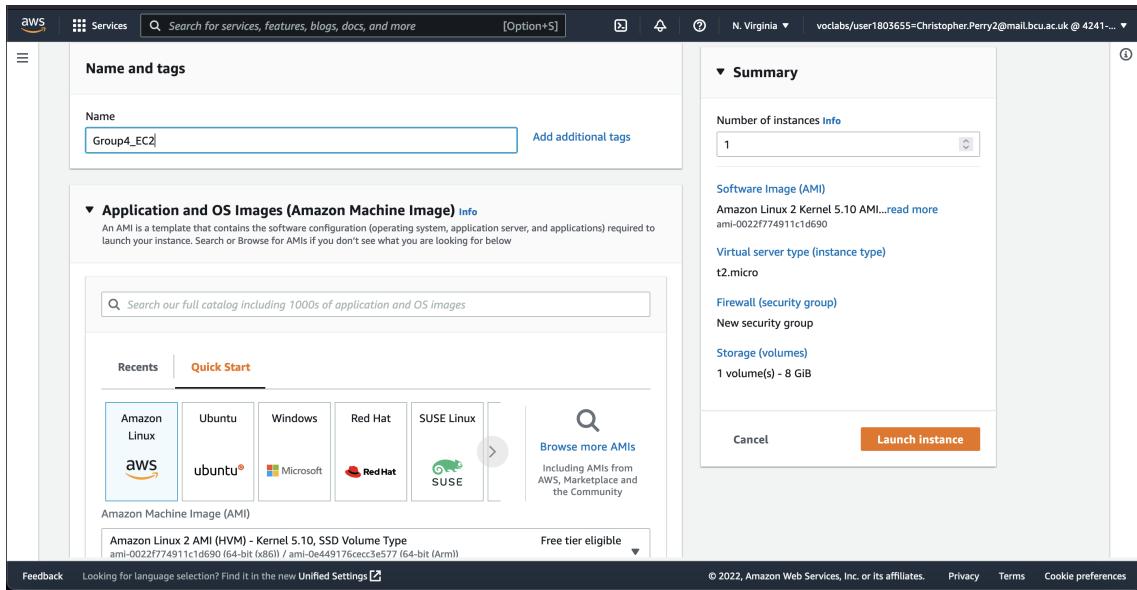


Figure A.9: Create Instance - Name & Tags

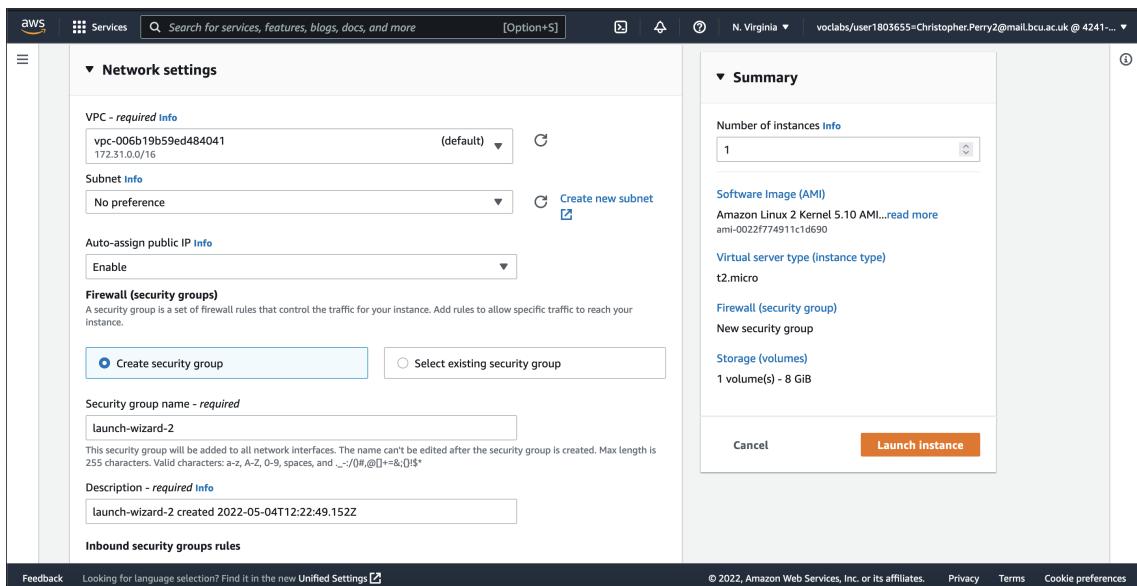


Figure A.10: Create Instance - Network Settings

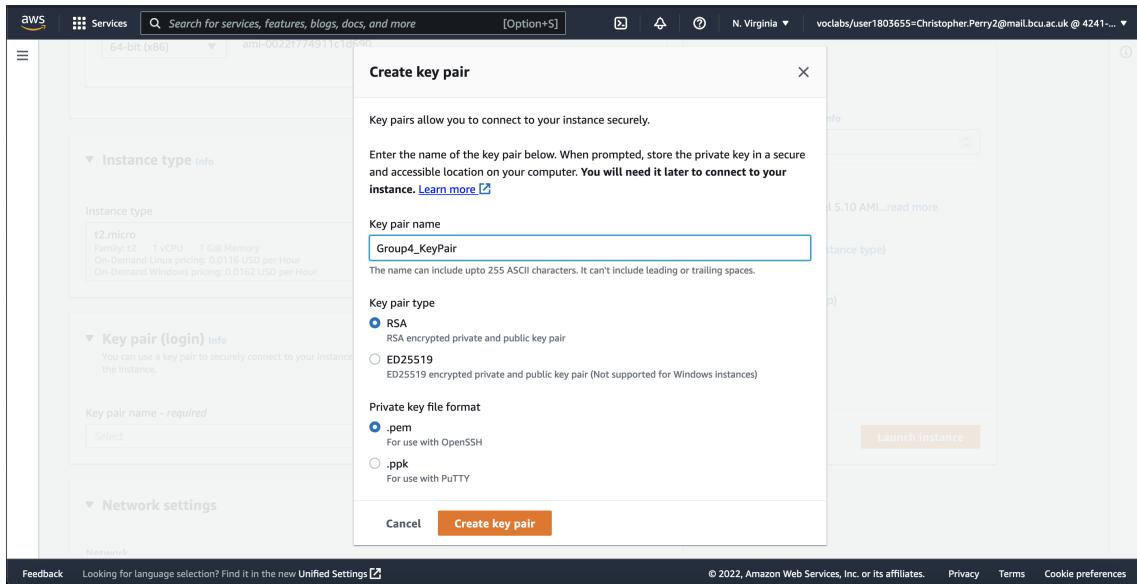


Figure A.11: Creating Key Pair

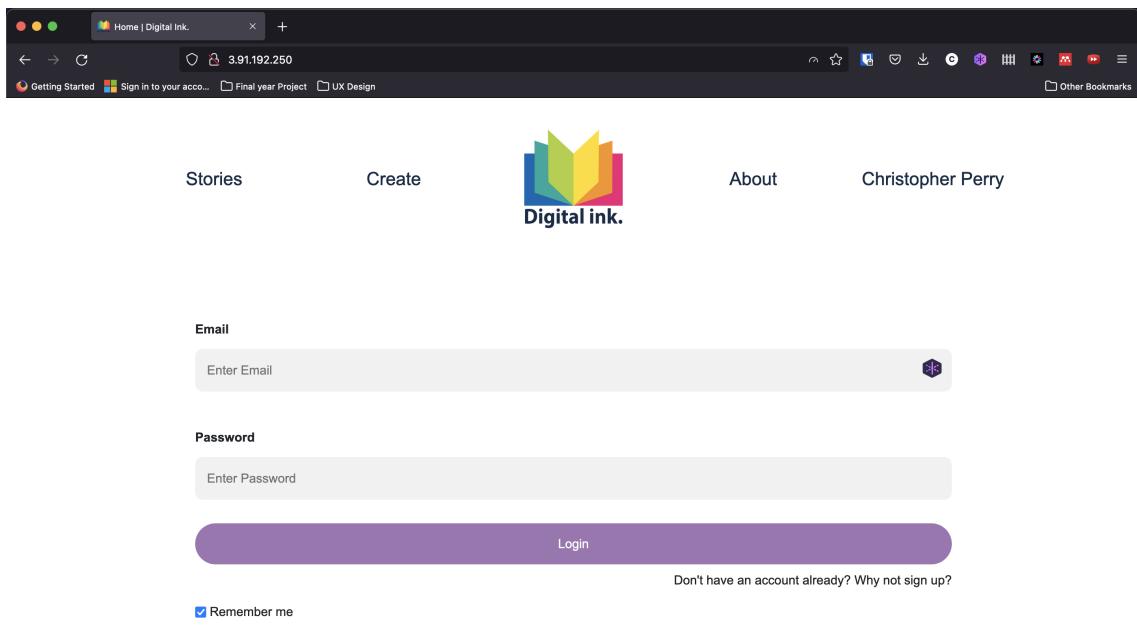


Figure A.12: Digital Ink

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

: phpmypadmin 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      5.7s
: phpmypadmin 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      5.8s
: phpmypadmin Pulling
: 5eb5b503b376 Pull complete          7.9s
: 8b1a0d84cf101 Pull complete          7.9s
: 38c937addeb7 Extracting   [=====] 32.31MB/91.6MB      12.8s
: 6a2f1dc96e59 Download complete          12.8s
: f8c3f82c39d4 Download complete          12.8s
: 90fc6462b088 Download complete          12.8s
[+] Running 7/319 Download complete      12.8s
: phpmypadmin Pulling
: 5eb5b503b376 Pull complete          12.8s
: 8b1a0d84cf101 Pull complete          12.8s
: 38c937addeb7 Extracting   [=====] 32.31MB/91.6MB      12.8s
: 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          13.0s
: 96c2ab37a1b Pull complete          8.5s
: 8aa3ac85066b Pull complete          9.8s
: ac7e524fc98 Pull complete          9.4s
: f6a88631064f Pull complete          9.8s
[+] 15bb3e15f562 Extracting   [=====] 13.11MB/14.06MB      10.1s
: ae65dc337dc8 Download complete          12.7s
: a4d4c43adaf5f2 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
| 3.7 kB  00:00:00
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 |
|=====|
| docker       | x86_64 | 20.10.13-2.amzn2 | amzn2extra-docker | 40 M |
|=====|
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 [D/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 |
|=====|
| docker       | x86_64 | 20.10.13-2.amzn2 | amzn2extra-docker | 40 M |
|=====|
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 [D/y/N]: y
Downloading packages:
(1/5): libcgroup-0.41-21.amzn2.x86_64.rpm          | 66 kB  00:00:00
(2/5): pigz-2.3.4-1.amzn2.0.1.x86_64.rpm          | 81 kB  00:00:00
(3/5): containerd-1.4.13-2.amzn2.0.1.x86_64.rpm  | 23 MB  00:00:00
(4/5): docker-20.10.13-2.amzn2.x86_64.rpm         | 40 MB  00:00:00
(5/5): runc-1.0.3-2.amzn2.x86_64.rpm              | 3.0 MB  00:00:00
|=====|
Total
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     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 various navigation options like EC2 Dashboard, EC2 Global View, Events, Tags, Limits, Instances (with sub-options like Instances, Instance Types, Launch Templates, etc.), and Images (with AMIs and AMI Catalog). The main content area has a title 'Instances (2) Info' and a table with two rows. The first row shows 'Group4\_EC2' with instance ID 'i-08b5532d59930e0b1', state 'Terminated', type 't2.micro', and other details. The second row shows 'Group4\_EC2' with instance ID 'i-0841da90317645e9f', state 'Running', type 't2.micro', and other details. There are buttons for 'Connect', 'Actions', and 'Launch instances'.

Figure A.19: Instances

The screenshot shows the 'Launching instance' step of the AWS EC2 instance launch process. It includes a progress bar at 80% completion, a 'Details' link, and a note to wait while launching. At the top, there's a message about opting into the new launch experience with an 'Opt-out to the old experience' button.

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

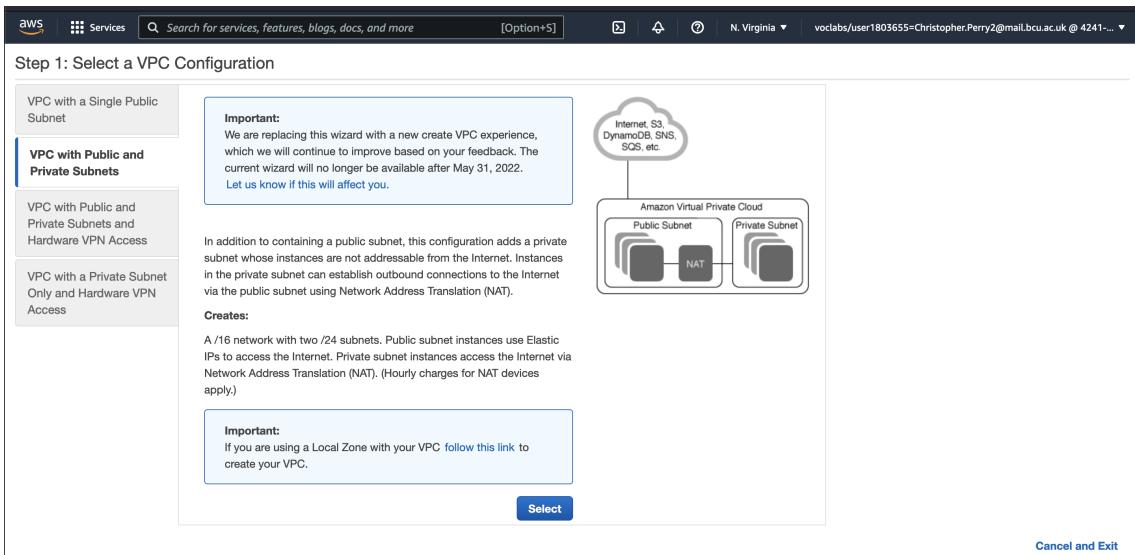


Figure A.22: Selecting a VPC Configuration

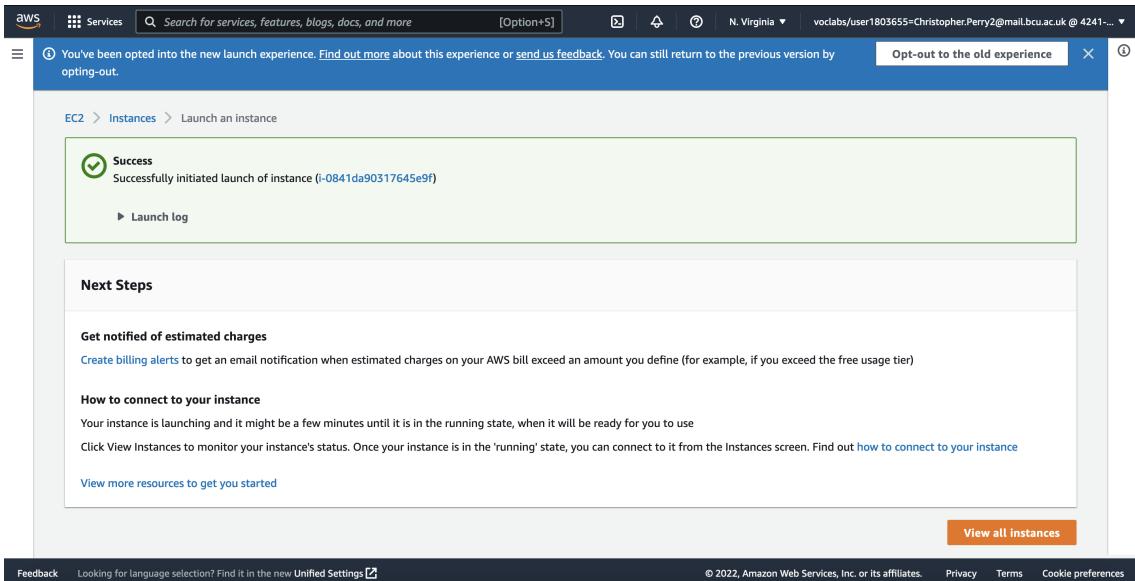


Figure A.23: Successfully Initiated Instance

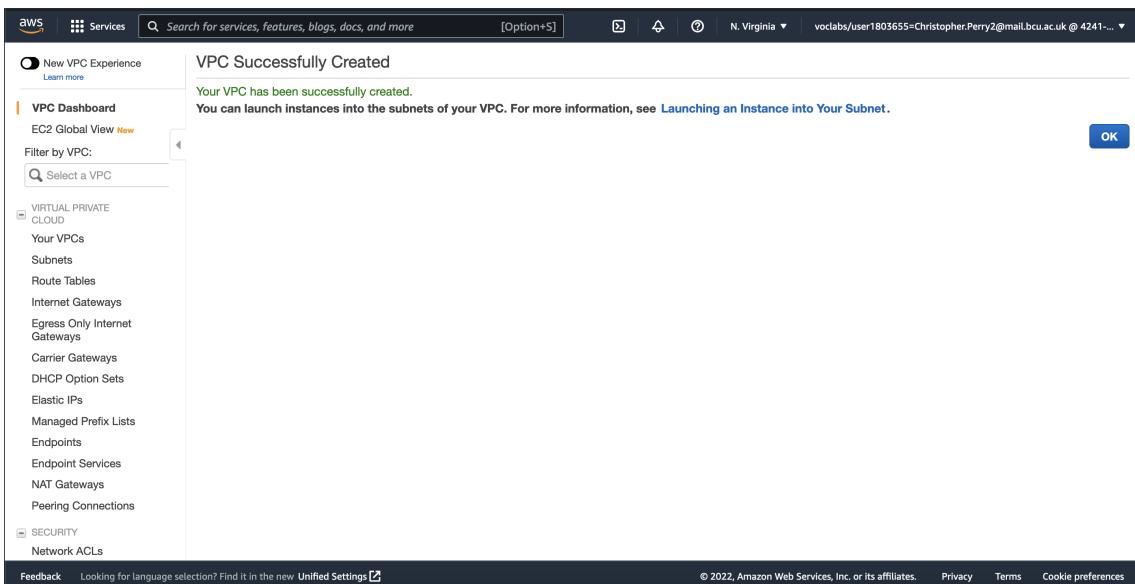


Figure A.24: VPC Successfully Created

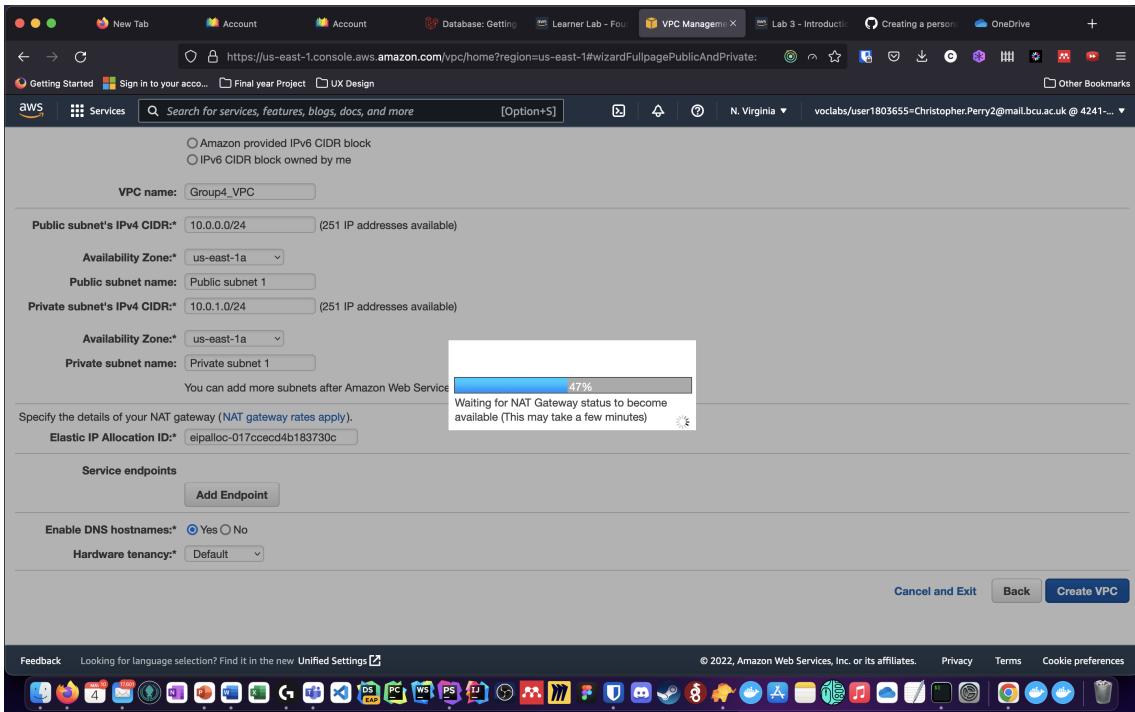


Figure A.25: VPC with Public and Private Subnets, Loading

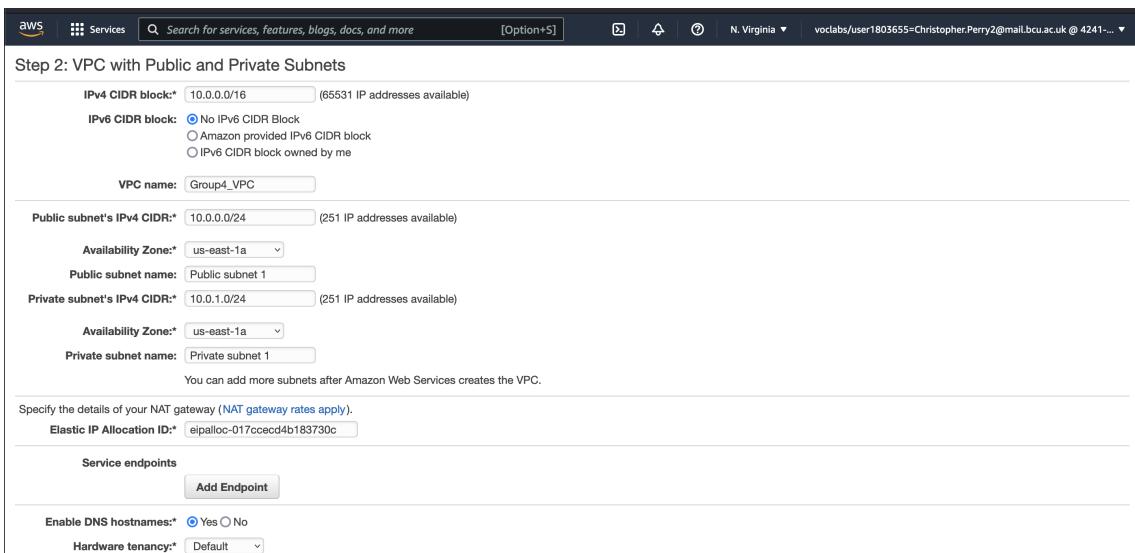


Figure A.26: VPC with Public and Private Subnets

The screenshot shows the AWS VPCs page with the following details:

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	-

**Details for VPC ID: vpc-07657585bc0e3b3b5**

Details	CIDRs	Flow logs	Tags
<b>Details</b>			
VPC ID vpc-07657585bc0e3b3b5	State Available	DNS hostnames Disabled	DNS resolution Enabled

Figure A.27: Your VPCs