

# 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**

# **List of Figures**

# 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. 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.

### **2.1 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 (?). 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.

## 2.2 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 Structured Query Language (SQL) is used to access the database. As shown in Figure ??, the database which this web app uses consists of three tables: `users`, `stories`, and `migrations`.

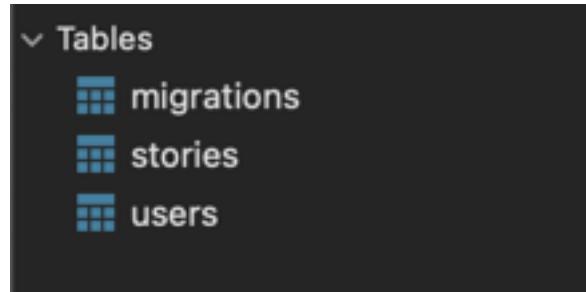


Figure 2.1: Database tables overview.

The `migrations` table (see Figure ??) 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	2014_10_12_000000_create_users_table	1
4	2020_03_13_105916_create_stories_table	1

Figure 2.2: `migrations` table.

The `users` table (see Figure ??) 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.

The screenshot shows a MySQL Workbench interface with a database named `digital-link` and a schema named `group4`. The current table is `users`. The table has seven columns: `id`, `name`, `email`, `password`, `remember_token`, `created_at`, and `updated_at`. The data for the two rows is as follows:

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.3eqO3o0dRxalK8sMy.eBqUapgD...	mDUIGRIIVhEl6...	2022-05-10 15:50:27	2022-05-10 15:50:27
2	evie	^~@snugg.ie	\$2y\$10\$vVNRmMN/lRSIYTrCUjJEPu44jkVI0r5...	Mu1WdyOz1e8...	2022-05-10 15:51:29	2022-05-10 15:51:29

Figure 2.3: `users` table.

The `stories` table (see Figure ??) 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.

id	author_id	title	genre	blurb	content
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	
cover_image	file_upload	published	created_at	updated_at	
..storage/cover_images/7537329101652200606.j...	NULL	1	2022-05-10 16:36:46	2022-05-10 16:36:46	
images/digital-ink-logo.png	NULL	1	2022-05-10 16:38:51	2022-05-10 16:38:51	

Figure 2.4: stories table.

## 2.3 Interface Design

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 ???. 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. On the left, 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 book icon. The main area contains two forms side-by-side. The left form is for 'Login' and includes fields for 'Email' (with placeholder 'Enter Email') and 'Password' (with placeholder 'Enter Password'). Below these fields is a purple 'Login' button. At the bottom of this section is a link 'Don't have an account already? Why not sign up?' and a 'Remember me' checkbox. The right form is for 'Sign Up' and 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.

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 ???. Once a story has been created, it is written to the `stories` table.

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

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 ???.

## Create your story!

**Author Reference Number:** \*

**Title:** \*

Every story needs a good title!

**Genre:** \*

What type of story are you creating?

**Your Story:** \*

Add the content of your story below.

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

No file selected.

**Blurb:** \*

Add a short description of your story!

Meet Group 4 and their love for AWS!

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 2.6: *Digital-Ink* story creation form.

**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.

**Stories**

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.

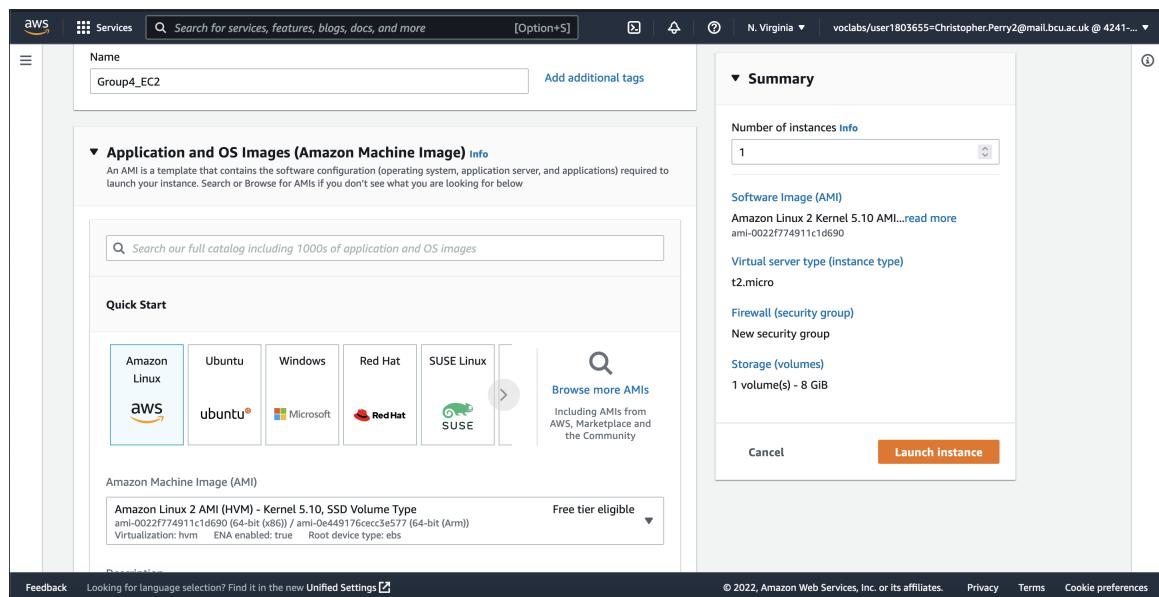


Figure 4.1: Selection of EC2 OS Image.

Figure ?? 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 ??.

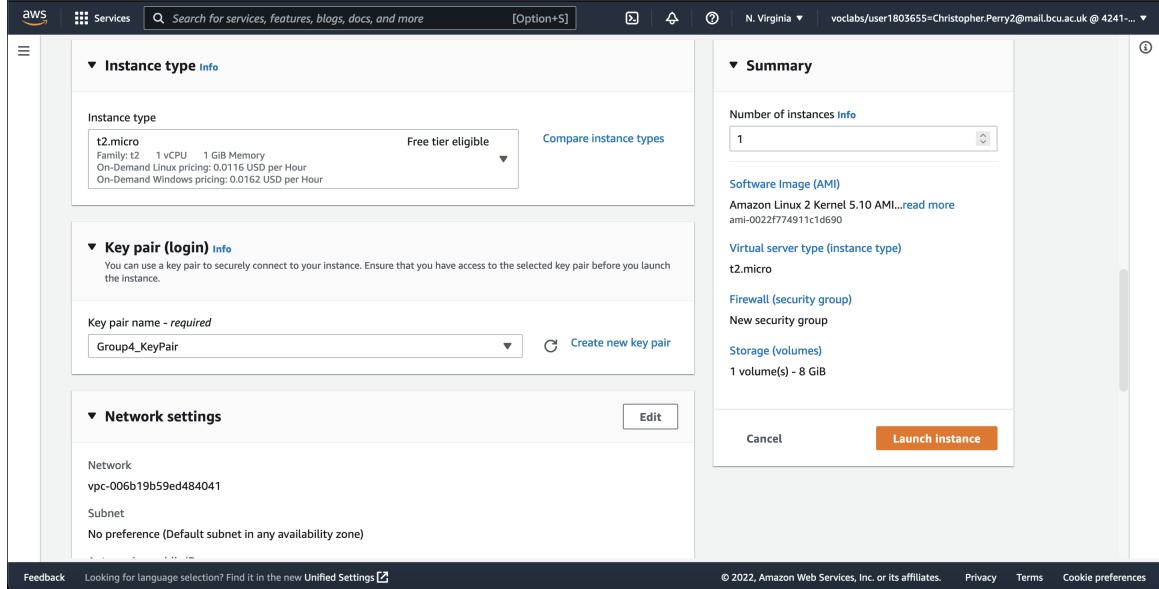


Figure 4.2: Selection of EC2 Instance.

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.

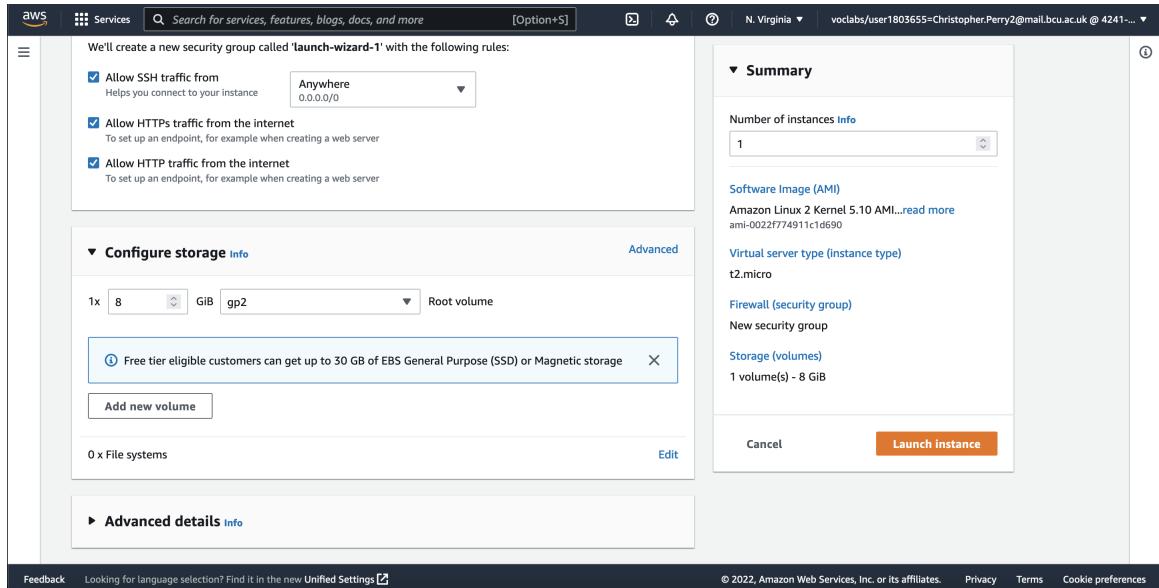


Figure 4.3: Selection of EC2 Storage Configuration.

The selection of these options can be found in Figure ?? . 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**

# **Relational Database Service**

Amazon's RBS 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 RBS instance, an 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 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**

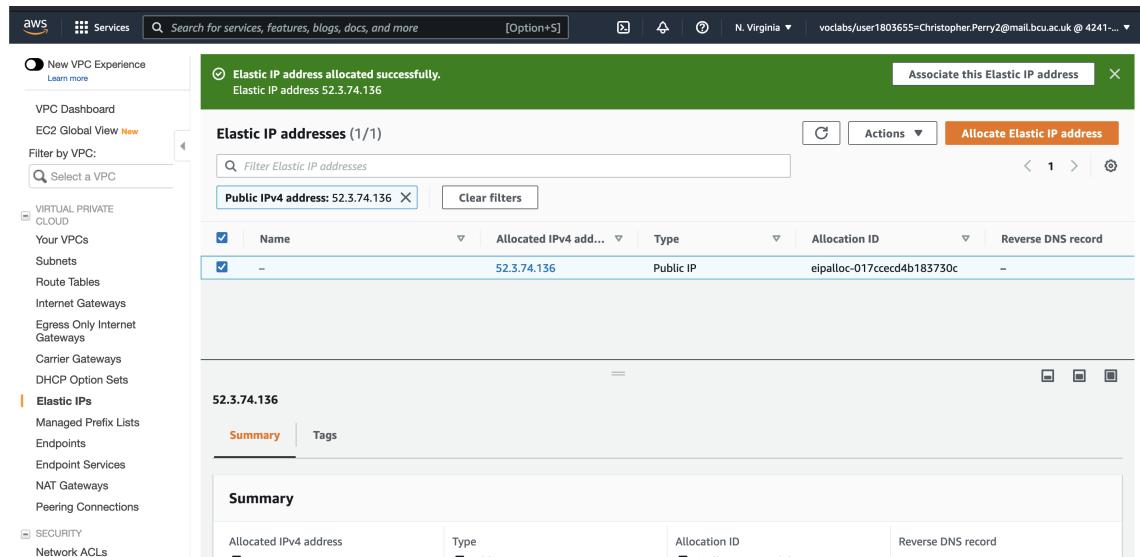
## **Conclusion**

# 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.".

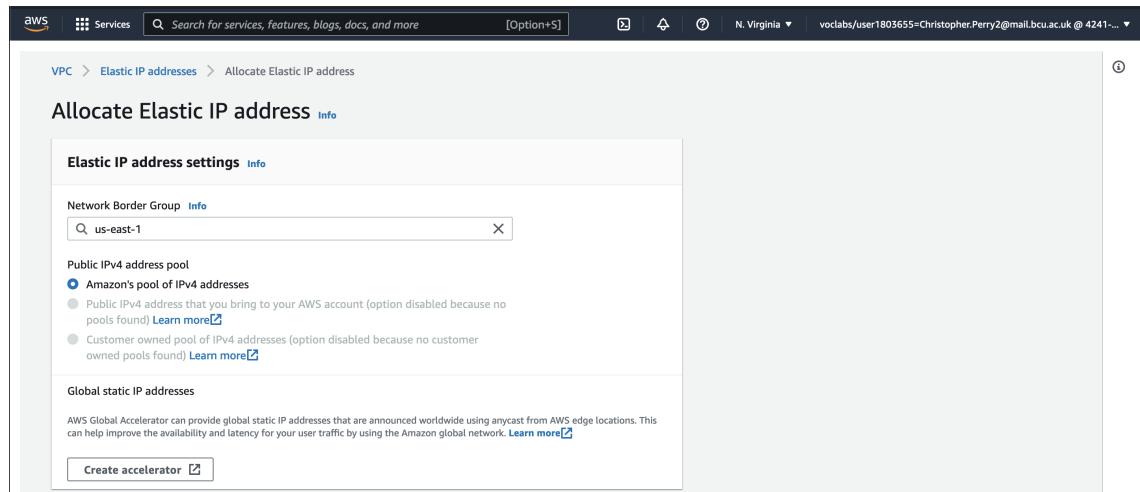
# Appendix A: Screenshots

I am an appendix, please be kind.



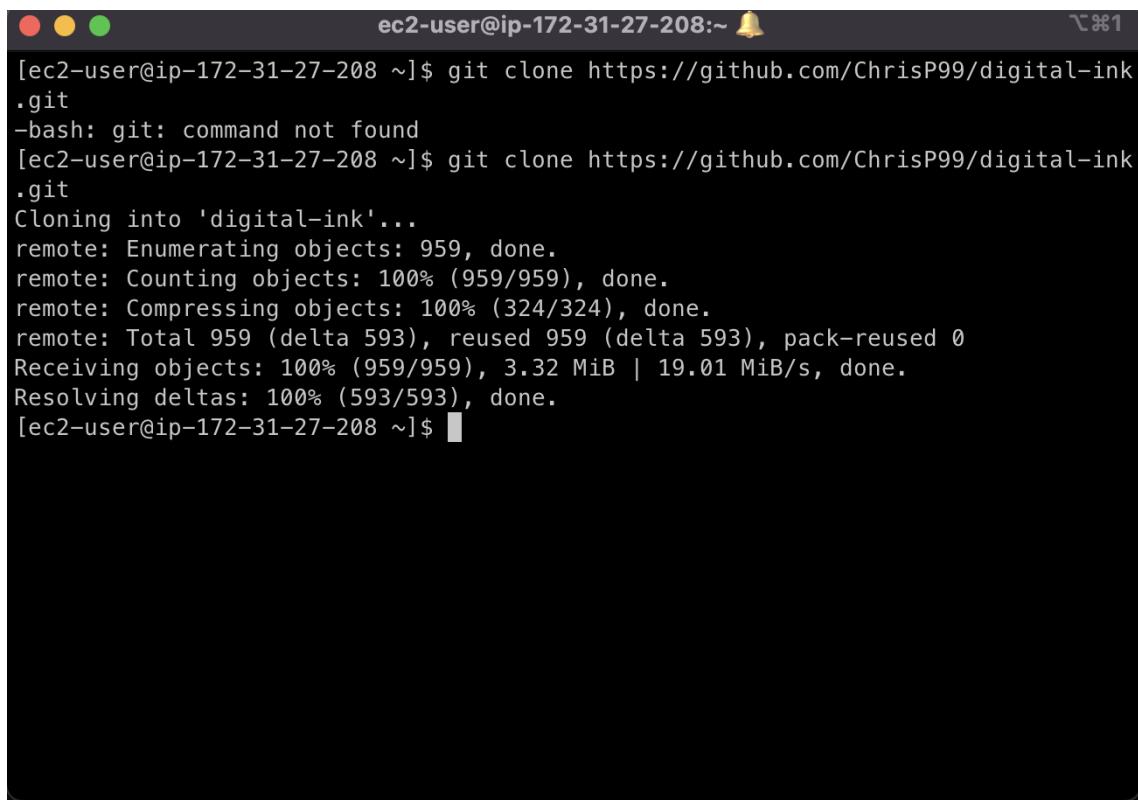
The screenshot shows the AWS VPC Elastic IP addresses page. At the top, a green banner displays the message "Elastic IP address allocated successfully." followed by the allocated IP address "52.3.74.136". Below the banner, the main interface shows a table of "Elastic IP addresses (1/1)". The table has 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, there is a summary section for the IP address 52.3.74.136, which includes tabs for Summary and Tags, and a detailed view of the summary information.

Figure A.1: After Allocating Elastic IP Address



The screenshot shows the "Allocate Elastic IP address" settings page. The top navigation bar includes "VPC > Elastic IP addresses > Allocate Elastic IP address". The main form is titled "Elastic IP address settings" and contains sections for "Network Border Group" (set to "us-east-1") and "Public IPv4 address pool". Under "Public IPv4 address pool", the "Amazon's pool of IPv4 addresses" option is selected. There is also a note about "Global static IP addresses" and a "Create accelerator" button at the bottom.

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

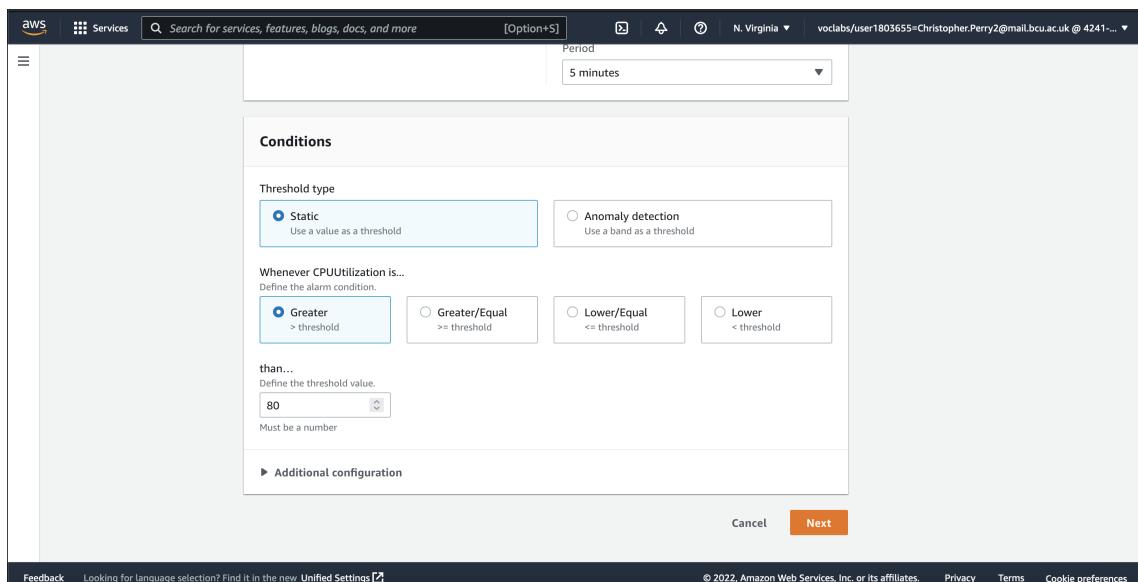


Figure A.4: CloudWatch Conditions

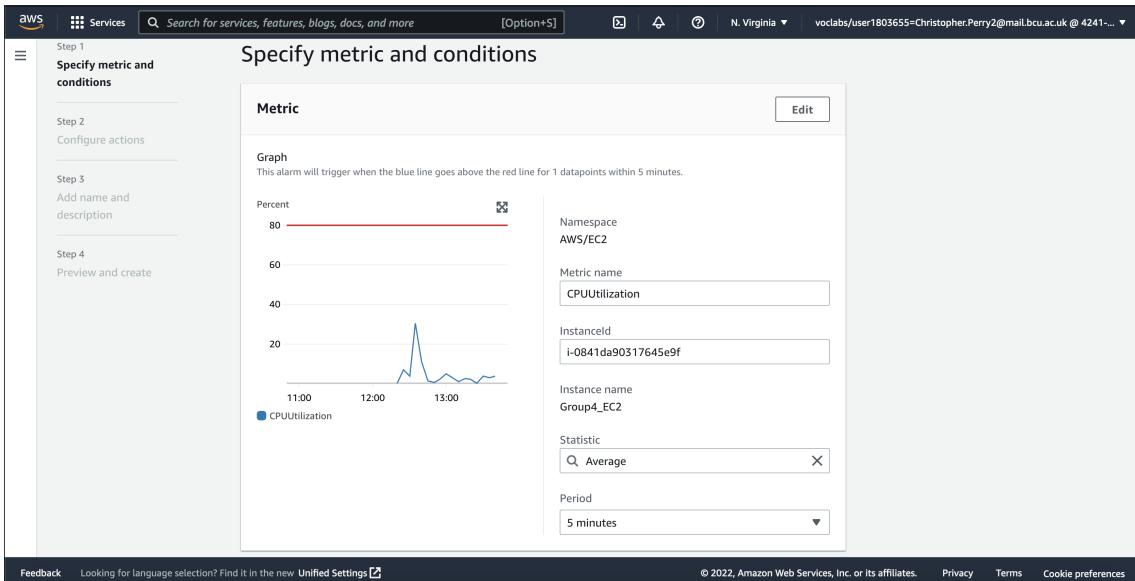


Figure A.5: CloudWatch Specify Metric

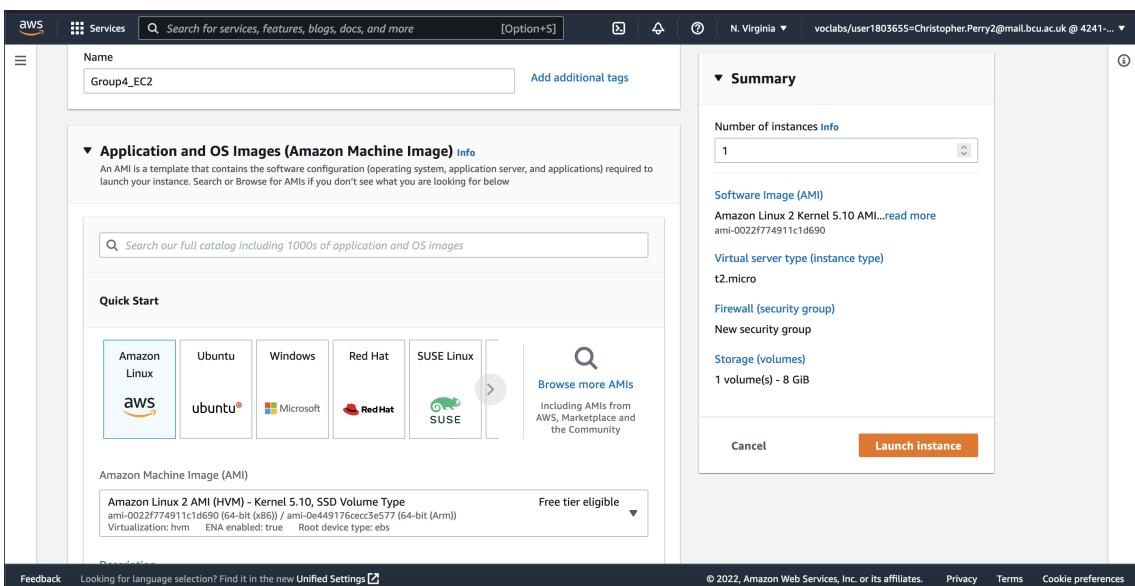


Figure A.6: Create Instance - Application and OS Images

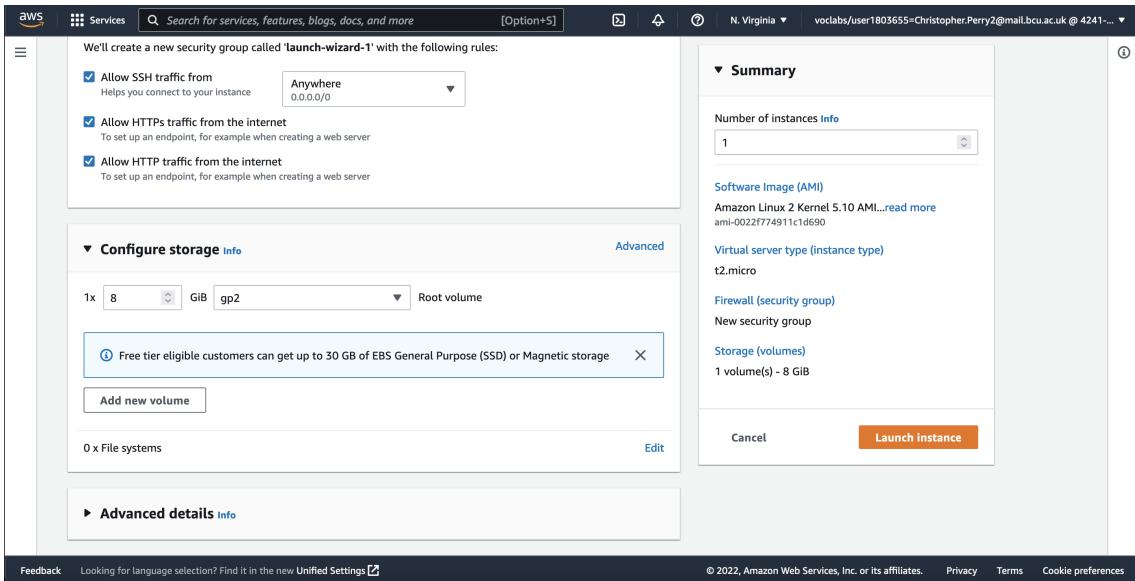


Figure A.7: Create Instance - Configure Storage

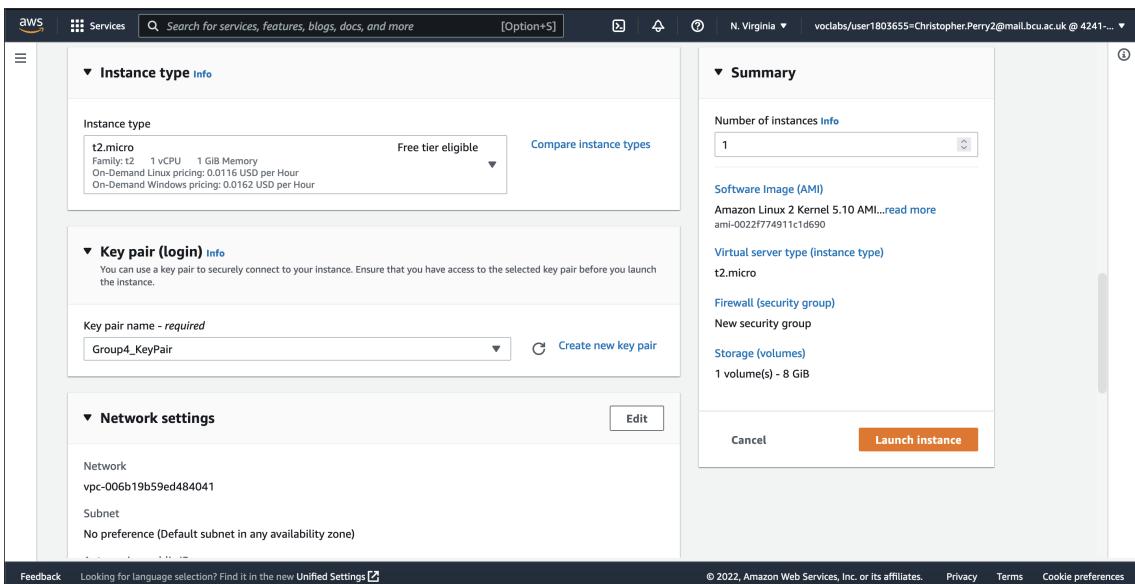


Figure A.8: Create Instance - Instance Type

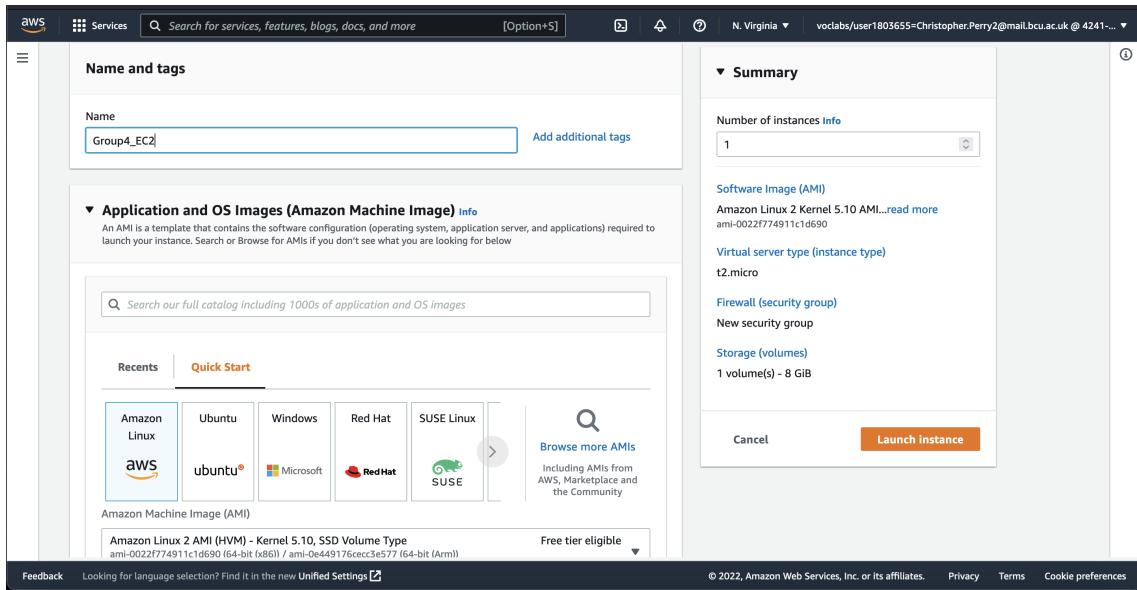


Figure A.9: Create Instance - Name & Tags

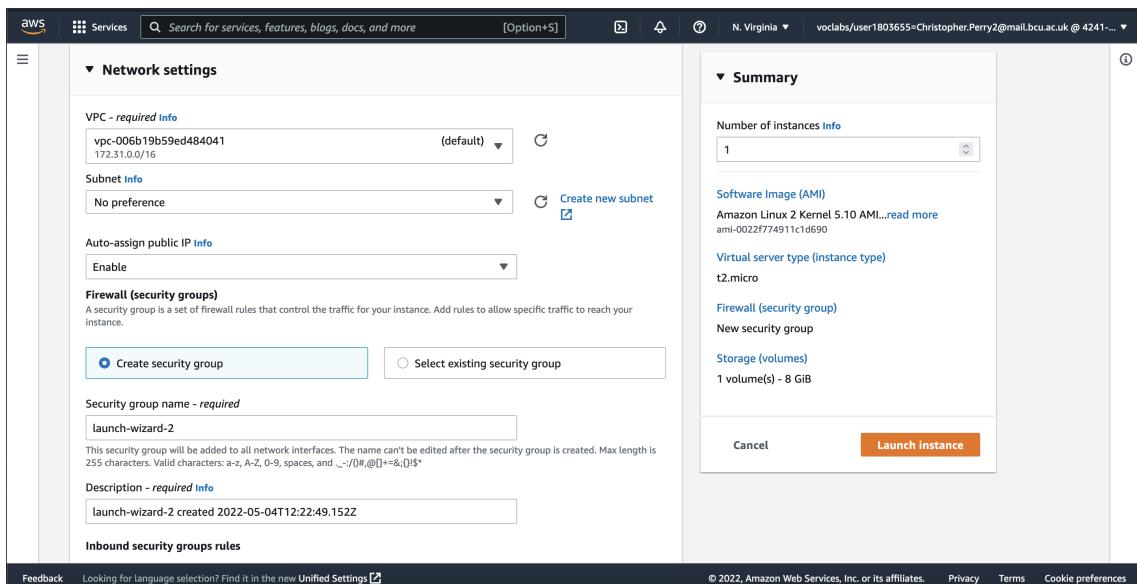


Figure A.10: Create Instance - Network Settings

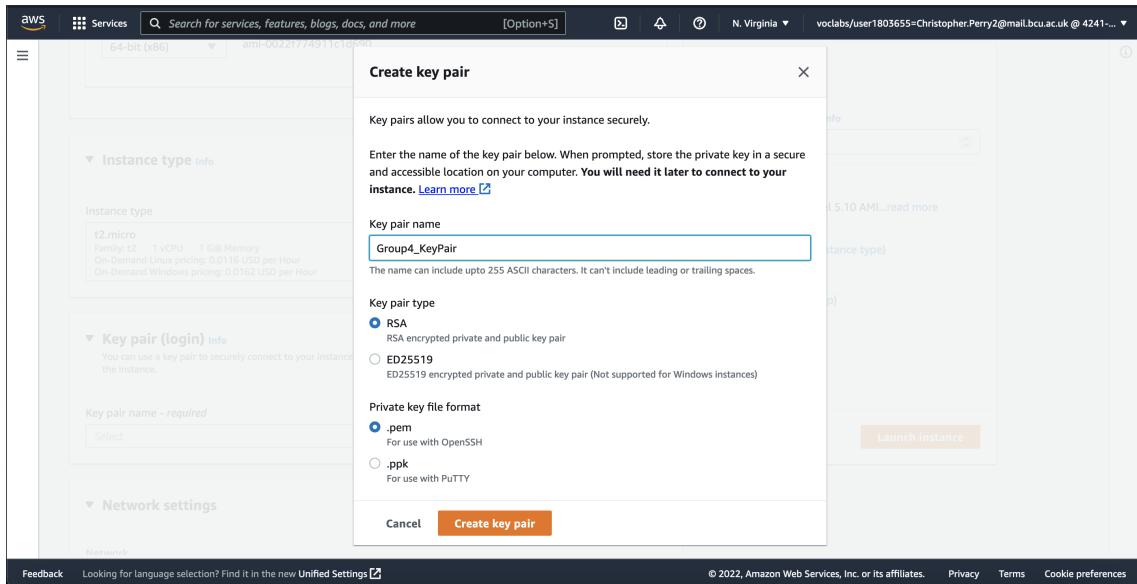


Figure A.11: Creating Key Pair

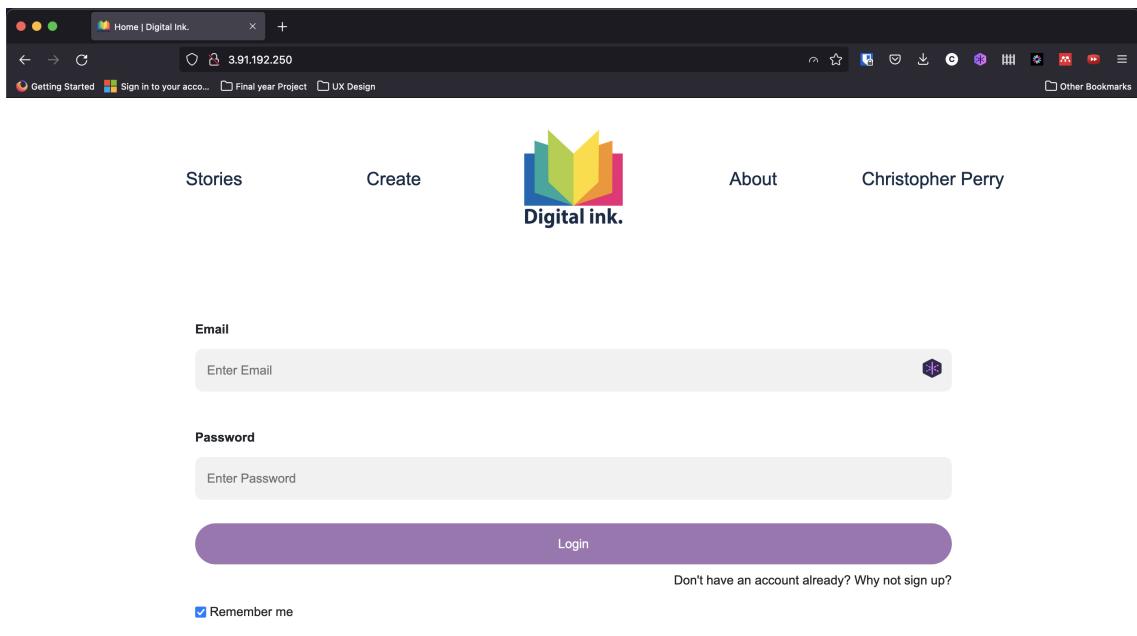


Figure A.12: Digital Ink

```
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
: ad4c4c43ad5f52 Download complete                         12.7s
: c6cab33e8ff1 Download complete                         12.7s
: 2e1cf2c43f16 Download complete                         12.7s
: 2e5ee322aaf48 Download complete                         12.7s
```

Figure A.13: Docker Compose

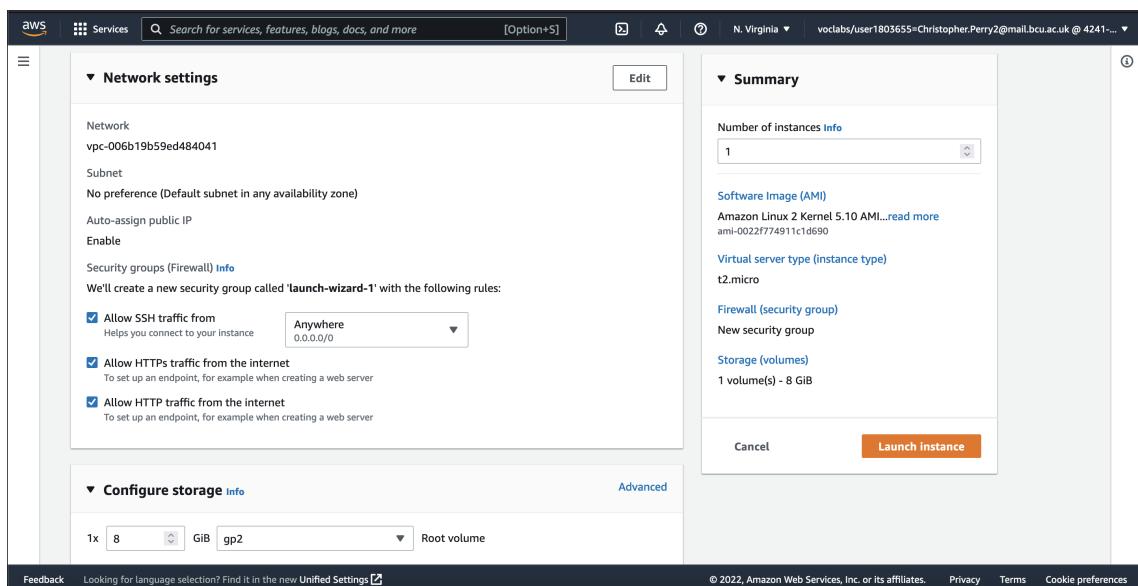


Figure A.14: Edit Instance - Network Settings

```
[ec2-user@ip-172-31-27-208 ~]$ sudo yum update
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core
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 [D/y/N]: [ ] | 54% | 5.2 GB | 0 18% | ① 5-04, 1:31 PM | snuffle | ↻ | □ ~ | % ssh + fish + bash + zsh + fish + fish + fish + fish + fish + fish + spacedust |
```

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 |
|=====|=====|=====|=====|=====|
| 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
  66 MB/s | 67 MB 00:00:01
| 74% | 5.3 GB | 0 18% | ① 5-04, 1:31 PM | snuffle | ↻ | □ ~ | % ssh + fish + bash + zsh + fish + fish + fish + fish + fish + fish + spacedust |
```

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

```

git-core          x86_64      2.32.0-1.amzn2.0.1           amzn2-core          4.8 M
git-core-doc     noarch      2.32.0-1.amzn2.0.1           amzn2-core          2.7 M
perl-Error       noarch      1:0.17020-2.amzn2           amzn2-core          32 k
perl-Git         noarch      2.32.0-1.amzn2.0.1           amzn2-core          43 k
perl-TermReadKey x86_64      2.30-28.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-2.30-28.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
  Installing : git-core-doc-2.32.0-1.amzn2.0.1.noarch
  Installing : 1:emacs-filesystem-27.2-4.amzn2.0.1.noarch
  Installing : perl-TermReadKey-2.30-28.amzn2.0.2.x86_64
  Installing : perl-Git-2.32.0-1.amzn2.0.1.noarch
  Installing : git-2.32.0-1.amzn2.0.1.x86_64
  Verifying  : git-core-2.32.0-1.amzn2.0.1.noarch
  Verifying  : perl-TermReadKey-2.30-28.amzn2.0.2.x86_64
  Verifying  : perl-Git-2.32.0-1.amzn2.0.1.noarch
  Verifying  : 1:emacs-filesystem-27.2-4.amzn2.0.1.noarch
  Verifying  : git-2.32.0-1.amzn2.0.1.x86_64
  Verifying  : git-core-2.32.0-1.amzn2.0.1.x86_64
  Verifying  : 1:perl-Error-0.17020-2.amzn2.noarch
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 0:2.30-28.amzn2.0.2

Complete!
[ec2-user@ip-172-31-27-208 ~]$ snuffle | ▶ | □ ~ | ⟲ ssh + fish + bash + zsh + fish + fish + fish + fish + fish + fish + spacedust | 59% ↗ 5.4 GB ————— | 19% ↘ 5-04, 1:33 PM

```

Figure A.17: Installing Git

```

ec2-user@ip-172-31... #1
          Load  Upload Total Spent Left Speed
~ (-fish) #2 0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0
100 25.2M 100 25.2M 0   0 39.1M 0   0   0   0   0   0
[ec2-user@ip-172-31-27-208 digital-in]$ uname -a
Linux ip-172-31-27-208.ec2.internal 5.10.109-104.500.amzn2.x86_64 #1 SMP Wed Apr 13 20:31:43 UTC 2022 x86_64 x86_64 x86_64 GNU/Linux
[ec2-user@ip-172-31-27-208 digital-in]$ cat /etc/*-release
NAME="Amazon Linux"
VERSION="2"
ID="amzn"
ID_LIKE="centos rhel fedora"
VERSION_ID="2"
PRETTY_NAME="Amazon Linux 2"
ANSI_COLOR="0;33"
CPE_NAME="cpe:2.3:o:amazon:amazon_linux:2"
HOME_URL="https://amazonlinux.com/"
Amazon Linux 2 (Xenon)
[ec2-user@ip-172-31-27-208 digital-in]$ sudo chmod +x /usr/local/bin/docker-compose
[ec2-user@ip-172-31-27-208 digital-in]$ docker-compose --version
Docker Compose version v2.5.0
[ec2-user@ip-172-31-27-208 digital-in]$ systemctl status docker
● docker.service - Docker Application Container Engine
  Loaded: loaded (/usr/lib/systemd/system/docker.service; disabled; vendor preset: disabled)
  Active: inactive (dead)
    Docs: https://docs.docker.com
[ec2-user@ip-172-31-27-208 digital-in]$ sudo systemctl start docker
[ec2-user@ip-172-31-27-208 digital-in]$ systemctl status docker
● docker.service - Docker Application Container Engine
  Loaded: loaded (/usr/lib/systemd/system/docker.service; disabled; vendor preset: disabled)
  Active: active (running) since Wed 2022-05-04 12:40:06 UTC; 1s ago
    Docs: https://docs.docker.com
  Process: 3788 ExecStartPre=/usr/libexec/docker/docker-setup-runtimes.sh (code=exited, status=0/SUCCESS)
  Main PID: 3791 (dockerd)
    Tasks: 7
   Memory: 26.9M
      CGroup: /system.slice/docker.service
              └─3791 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock --default-ulimit nofile=32768:65536

May 04 12:40:06 ip-172-31-27-208.ec2.internal dockerd[3791]: time="2022-05-04T12:40:06+00:00" level=info msg="ClientConn switching balancer to \"pick_first\""
May 04 12:40:06 ip-172-31-27-208.ec2.internal dockerd[3791]: time="2022-05-04T12:40:06+00:00" level=warning msg="Your kernel does not support cgroup blkio weight"
May 04 12:40:06 ip-172-31-27-208.ec2.internal dockerd[3791]: time="2022-05-04T12:40:06+00:00" level=warning msg="Your kernel does not support cgroup blkio weight_device"
May 04 12:40:06 ip-172-31-27-208.ec2.internal dockerd[3791]: time="2022-05-04T12:40:06+00:00" level=info msg="Loading containers: start"
May 04 12:40:06 ip-172-31-27-208.ec2.internal dockerd[3791]: time="2022-05-04T12:40:06+00:00" level=info msg="Default bridge (docker0) is assigned with an IP address... address"
May 04 12:40:06 ip-172-31-27-208.ec2.internal dockerd[3791]: time="2022-05-04T12:40:06+00:00" level=info msg="Loading containers: done."
May 04 12:40:06 ip-172-31-27-208.ec2.internal dockerd[3791]: time="2022-05-04T12:40:06+00:00" level=info msg="Docker daemon" commit="906f57f graphdriver(s)=overlay..=20.10.13"
May 04 12:40:06 ip-172-31-27-208.ec2.internal dockerd[3791]: time="2022-05-04T12:40:06+00:00" level=info msg="Daemon has completed initialization"
May 04 12:40:06 ip-172-31-27-208.ec2.internal systemd[1]: Started Docker Application Container Engine.
May 04 12:40:06 ip-172-31-27-208.ec2.internal dockerd[3791]: time="2022-05-04T12:40:06+00:00" level=info msg="API listen on /run/docker.sock"
Hint: Some lines were ellipsized, use -l to show in full.
[ec2-user@ip-172-31-27-208 digital-in]$

snuffle | ▶ | □ ~ | ⟲ ssh + fish + bash + zsh + fish + fish + fish + fish + fish + spacedust | 67% ↗ 5.4 GB ————— | 26% ↘ 5-04, 1:40 PM

```

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 (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Scheduled Instances, Capacity Reservations, Images, AMIs, and AMI Catalog. The main content area has a title 'Instances (2) Info' with a search bar and a table. The table lists two instances: 'Group4\_EC2' (terminated, t2.micro, us-east-1d) and 'Group4\_EC2' (running, t2.micro, us-east-1d). At the bottom, there's a modal titled 'Select an instance'.

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 'Launch initiation' section, 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.

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

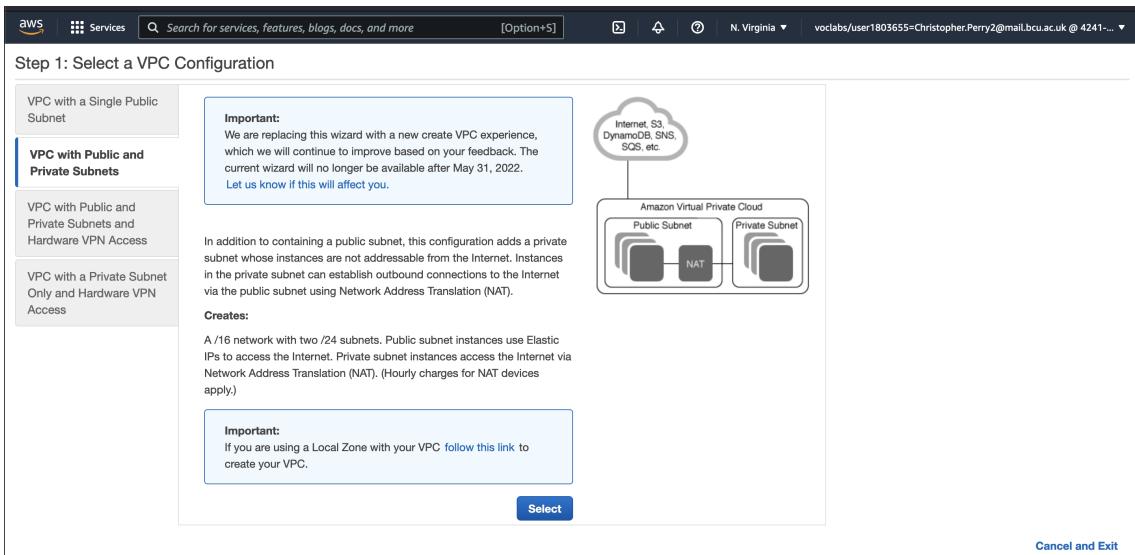


Figure A.22: Selecting a VPC Configuration

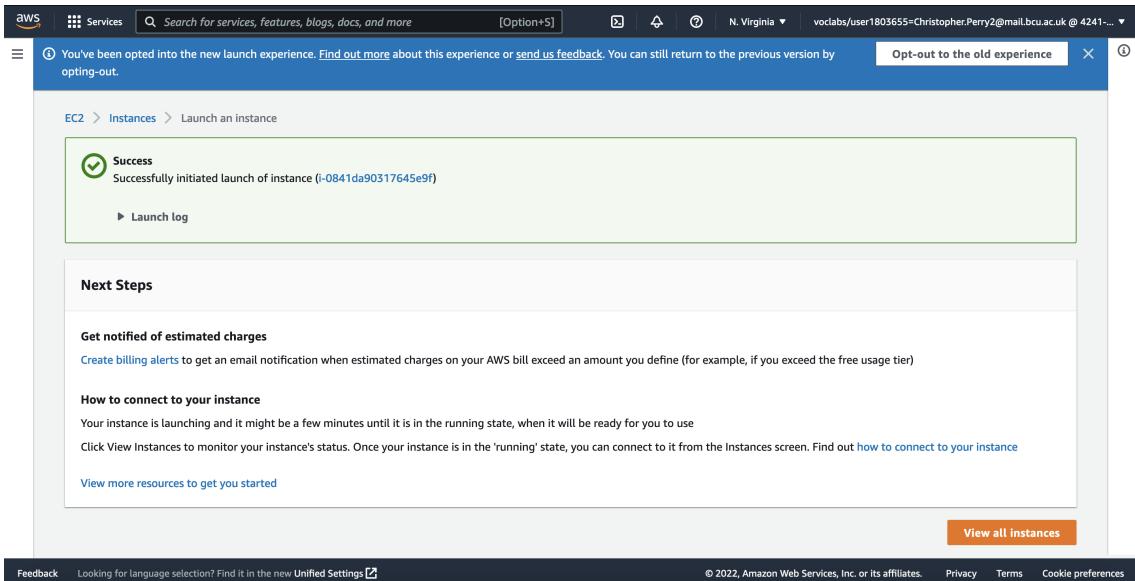


Figure A.23: Successfully Initiated Instance

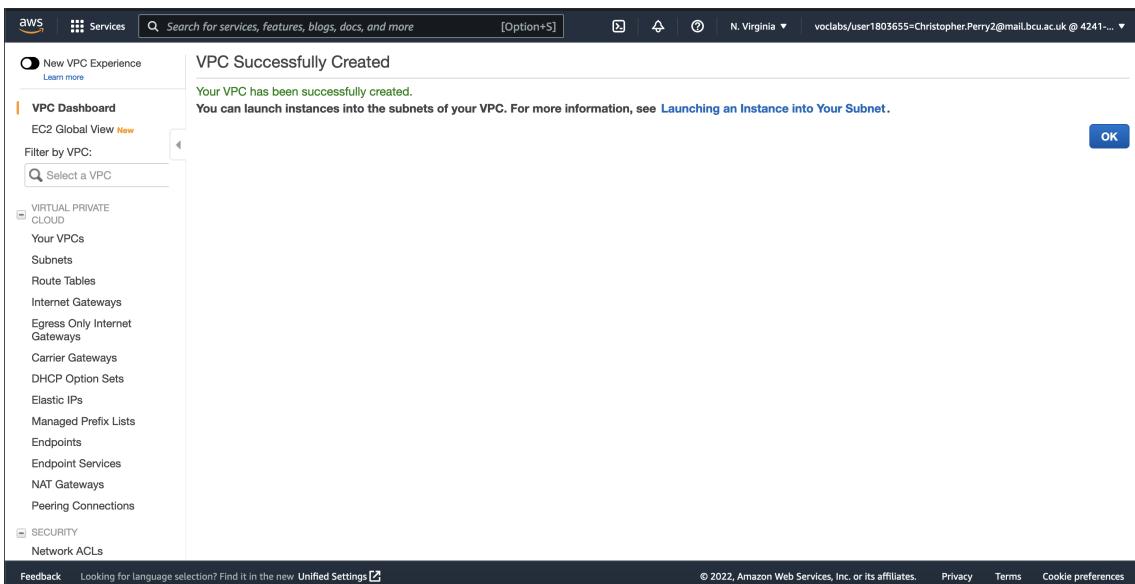


Figure A.24: VPC Successfully Created

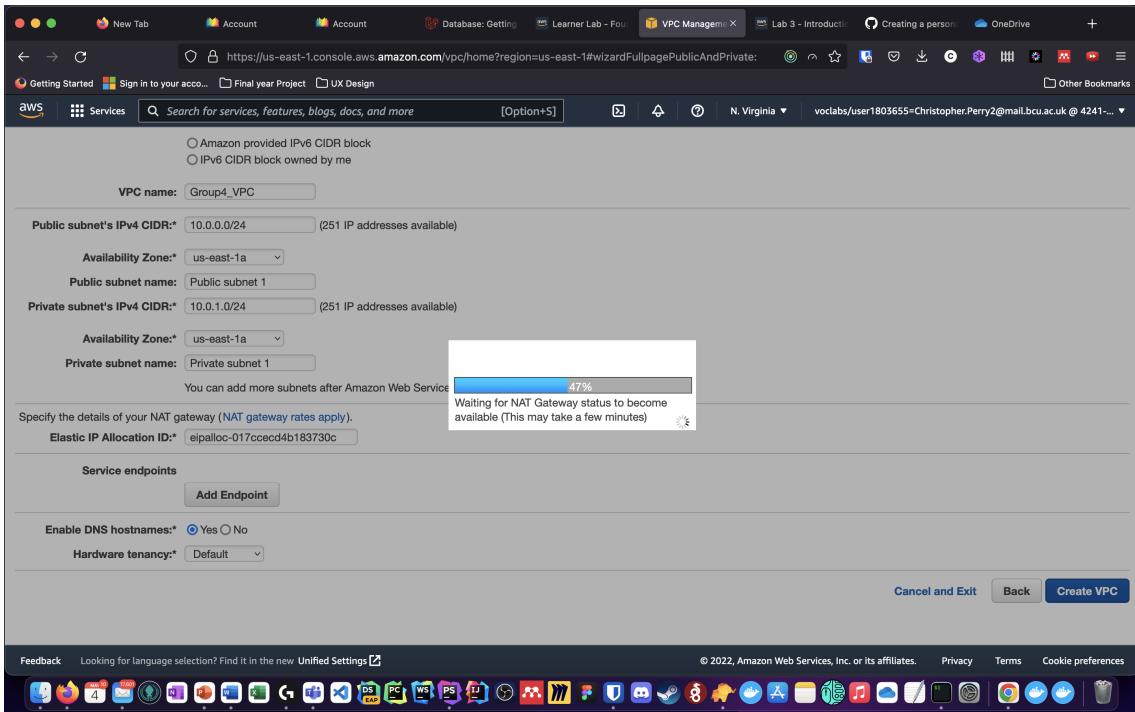


Figure A.25: VPC with Public and Private Subnets, Loading

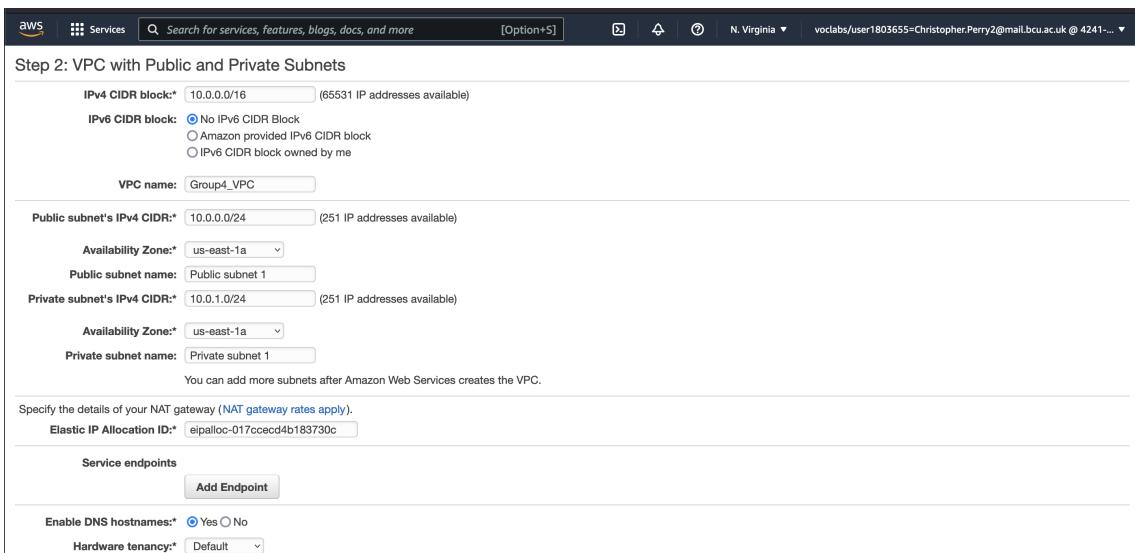


Figure A.26: VPC with Public and Private Subnets

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**

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

Feedback: Looking for language selection? Find it in the new Unified Settings.

Figure A.27: Your VPCs