Cloud Computing with AWS

By Adam Cordner, Chris Perry & Evie Snuggle

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.

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`.

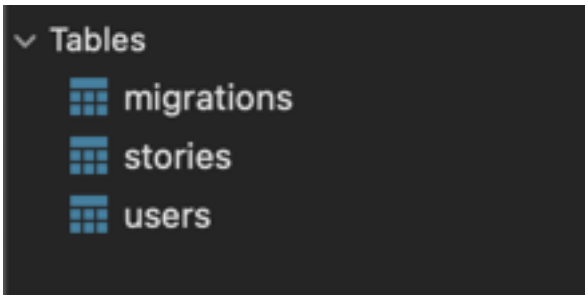

Figure 2.1: Database tables overview.

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.

id	migration	batch
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 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.

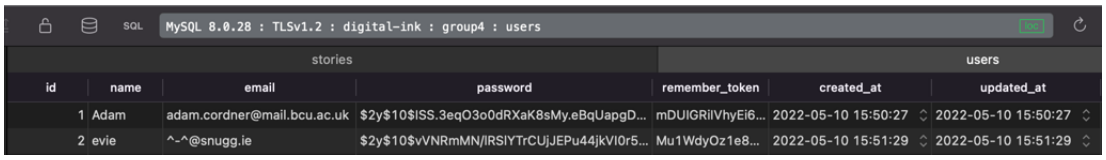

Figure 2.3: `users` table.

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.

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.

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.

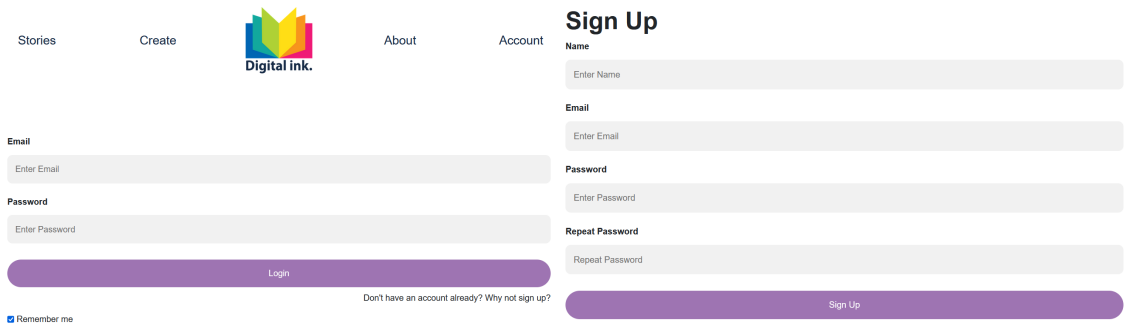

Figure 2.5: *Digital-Ink* home page log in and sign up forms.

Once a user is signed in, they can create a story. Creating a story requires the user to entire 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.

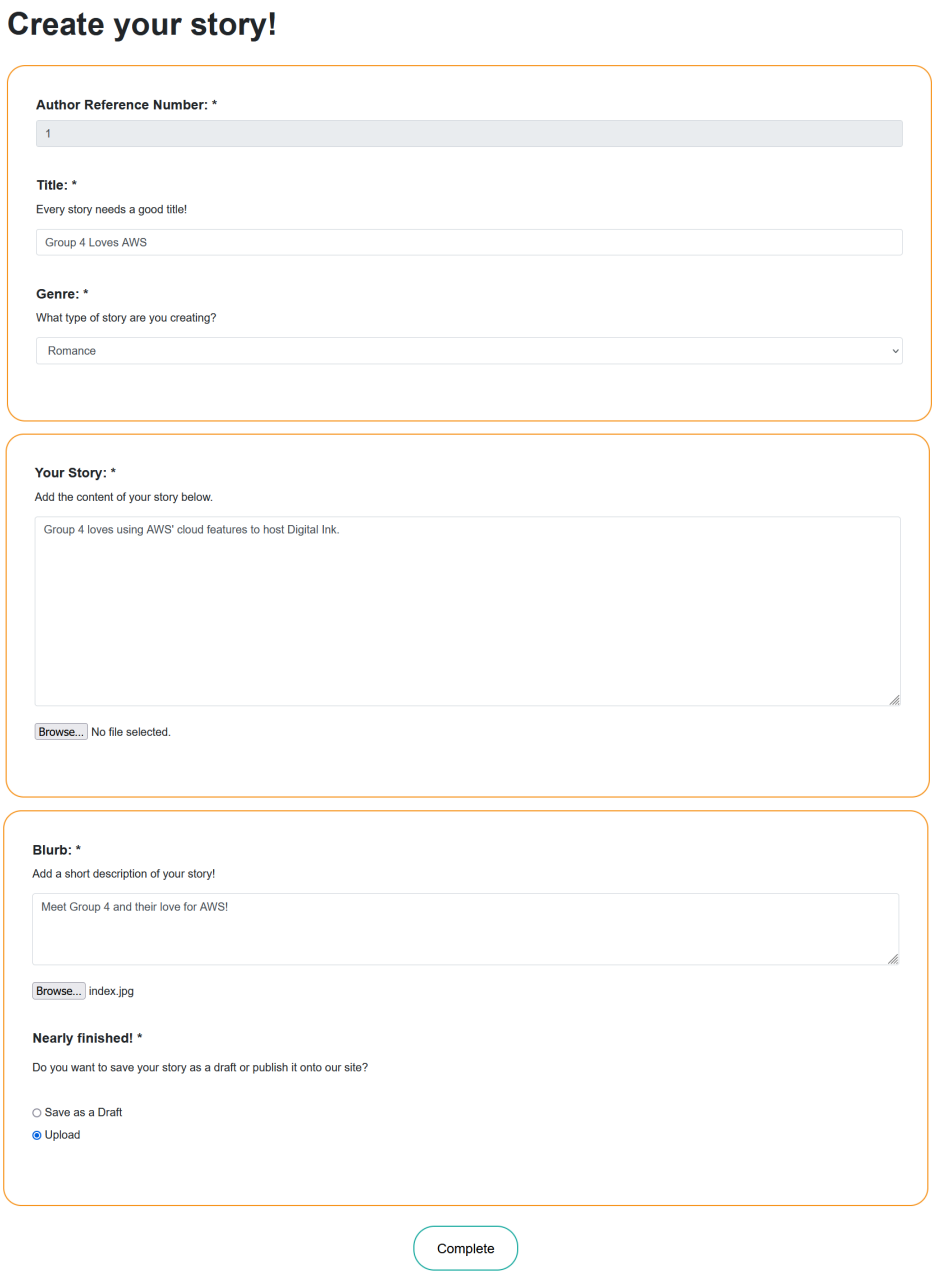

Figure 2.6: *Digital-Ink* story creation form.

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.

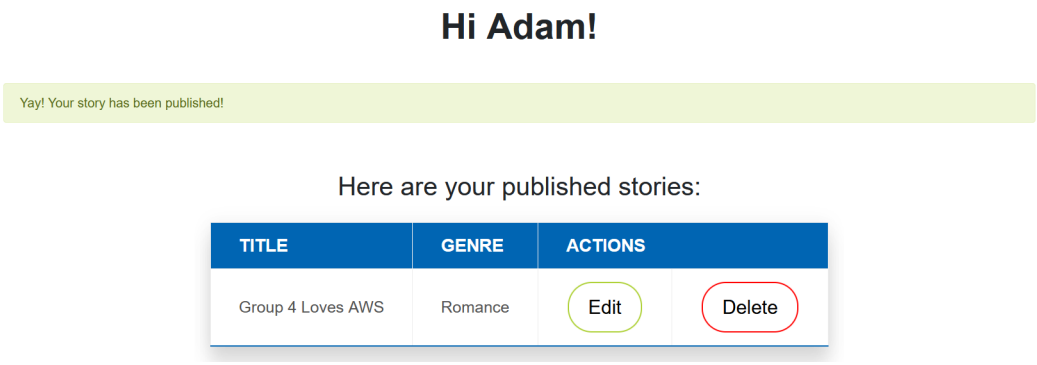

Figure 2.7: *Digital-Ink* account page.

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.

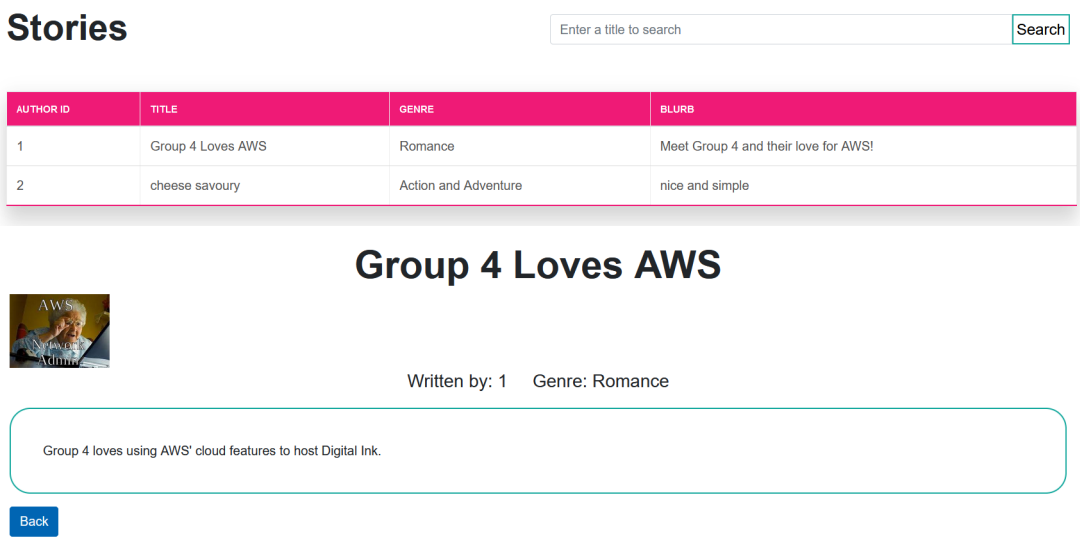

Figure 2.8: *Digital-Ink* stories page and story view.

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.

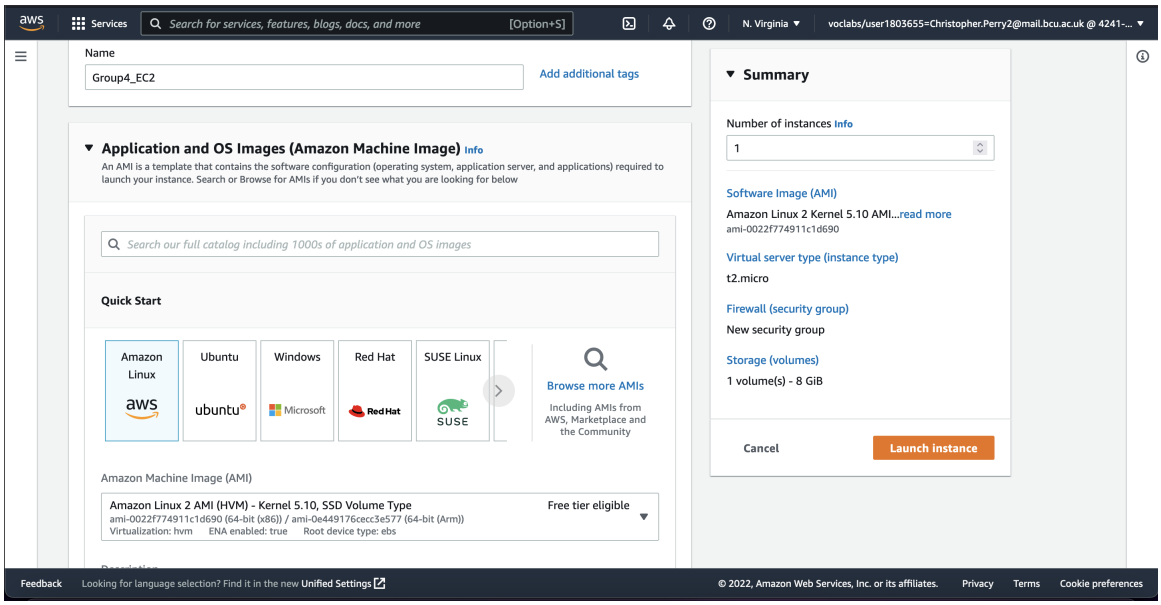

Figure 4.1: Selection of EC2 OS Image.

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 specfic 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.

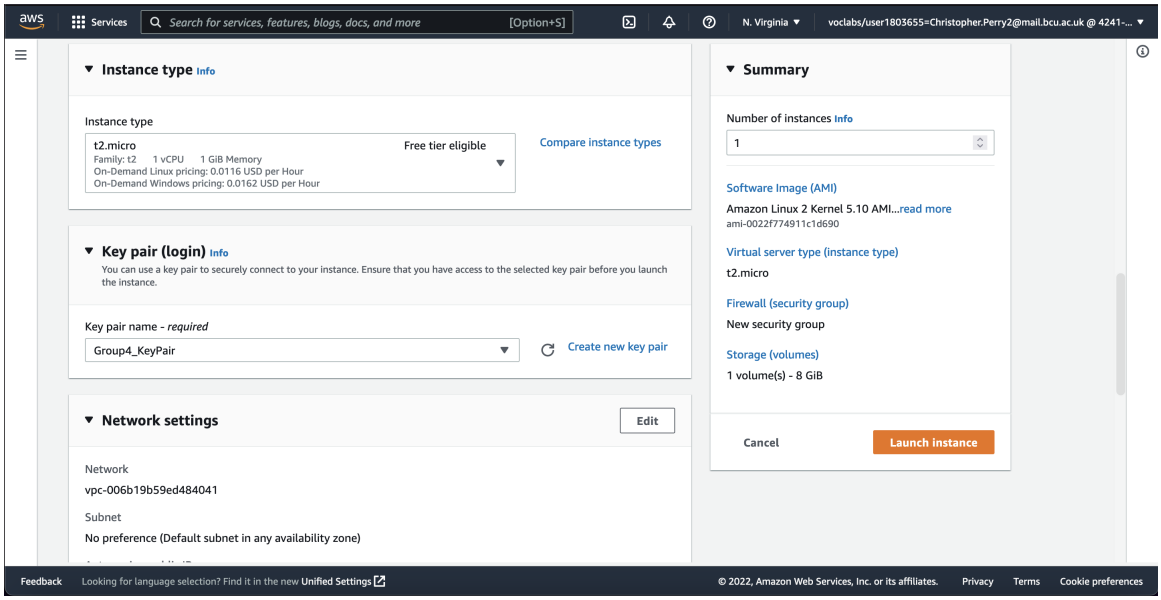

Figure 4.2: Selection of EC2 Instance.

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.

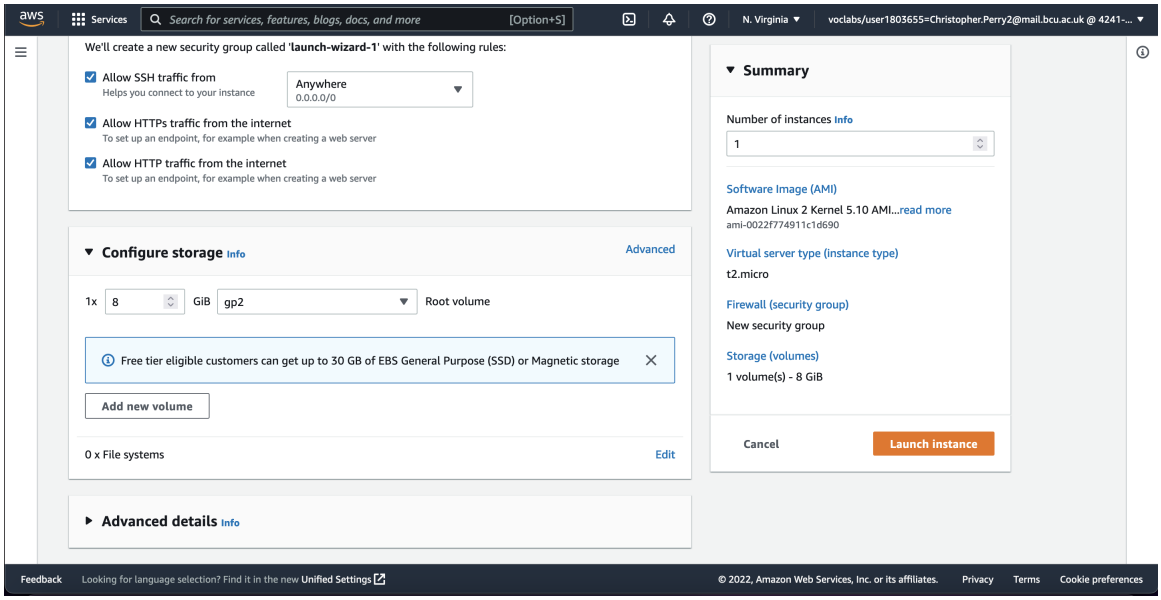

Figure 4.3: Selection of EC2 Storage Configuration.

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.

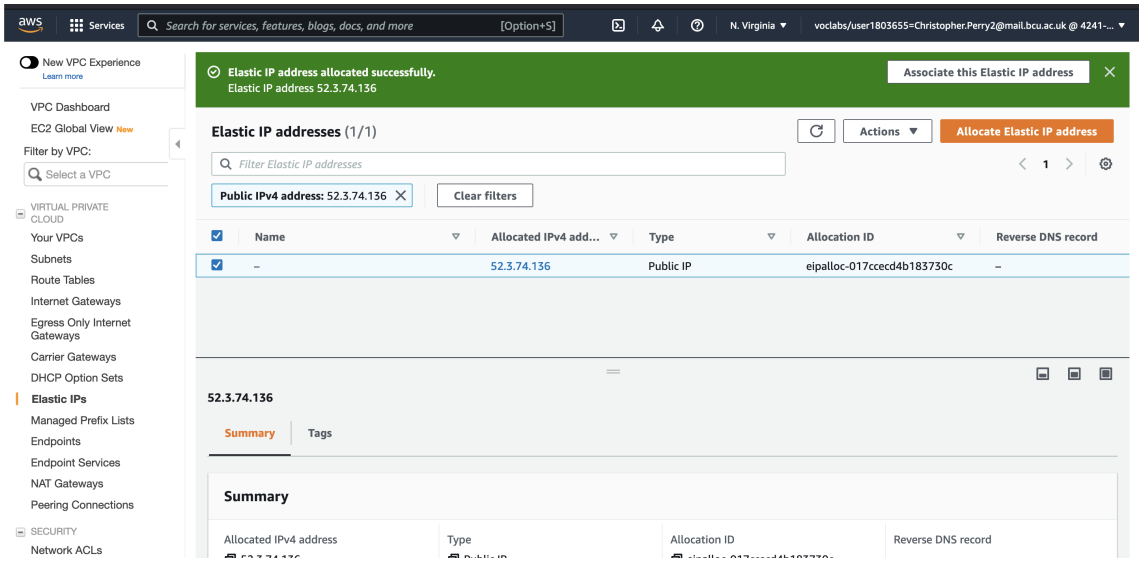

Figure A.1: After Allocating Elastic IP Address

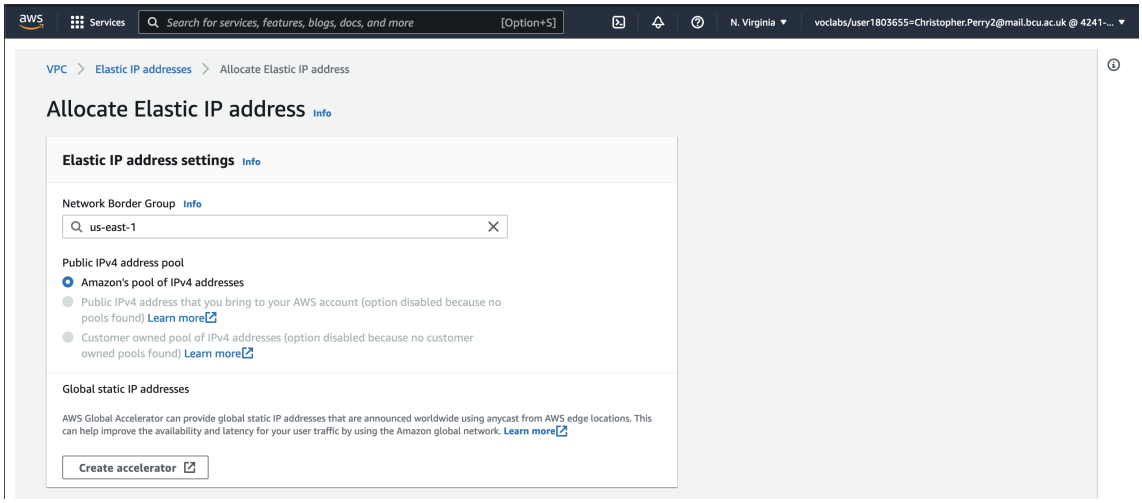

Figure A.2: Allocating Elastic IP Address

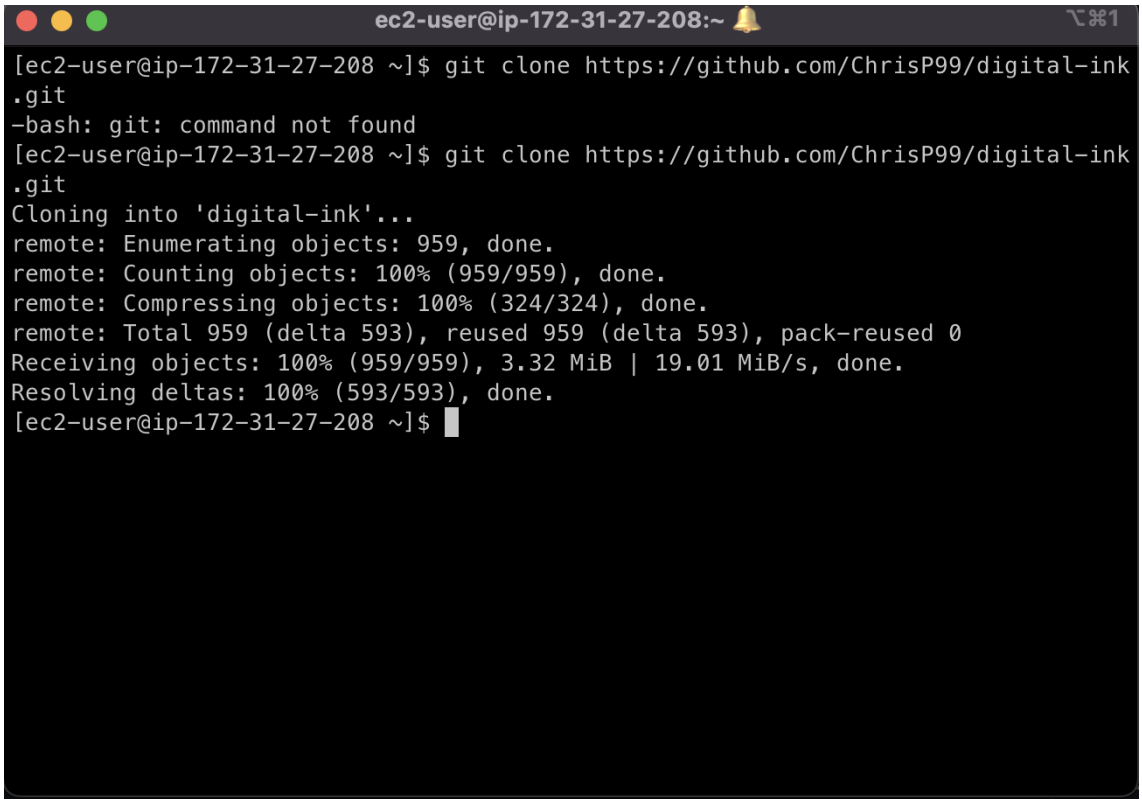

Figure A.3: Cloning the App

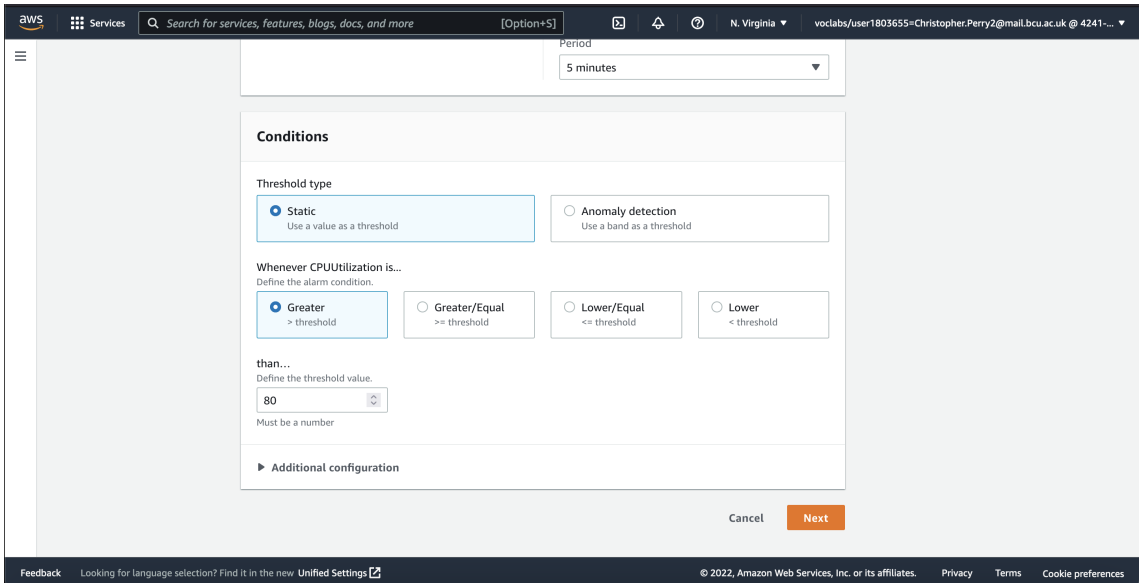

Figure A.4: CloudWatch Conditions

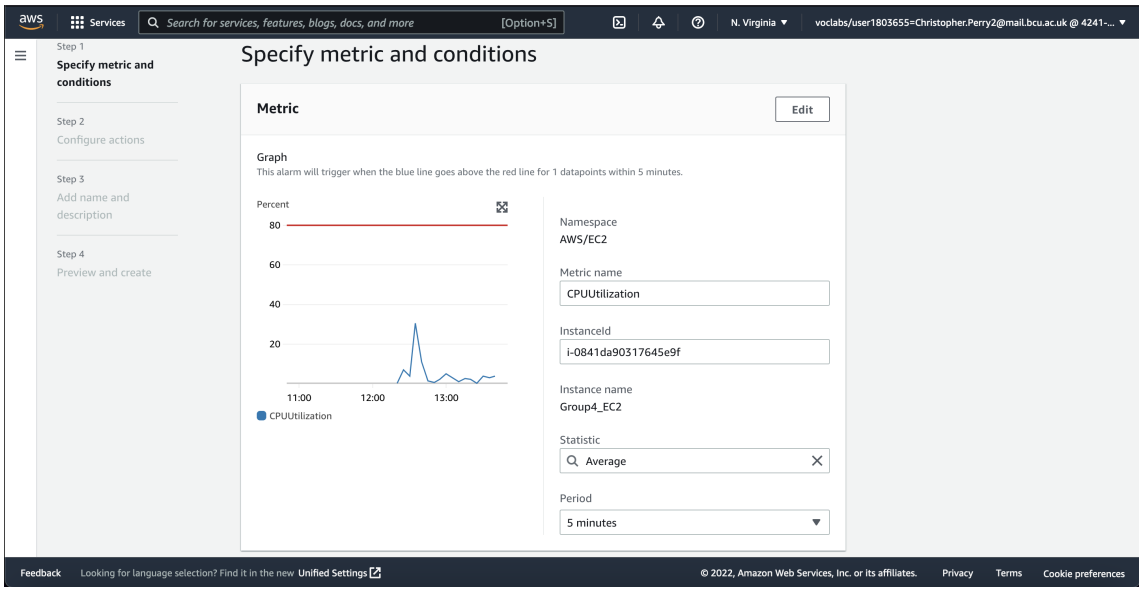

Figure A.5: CloudWatch Specify Metric

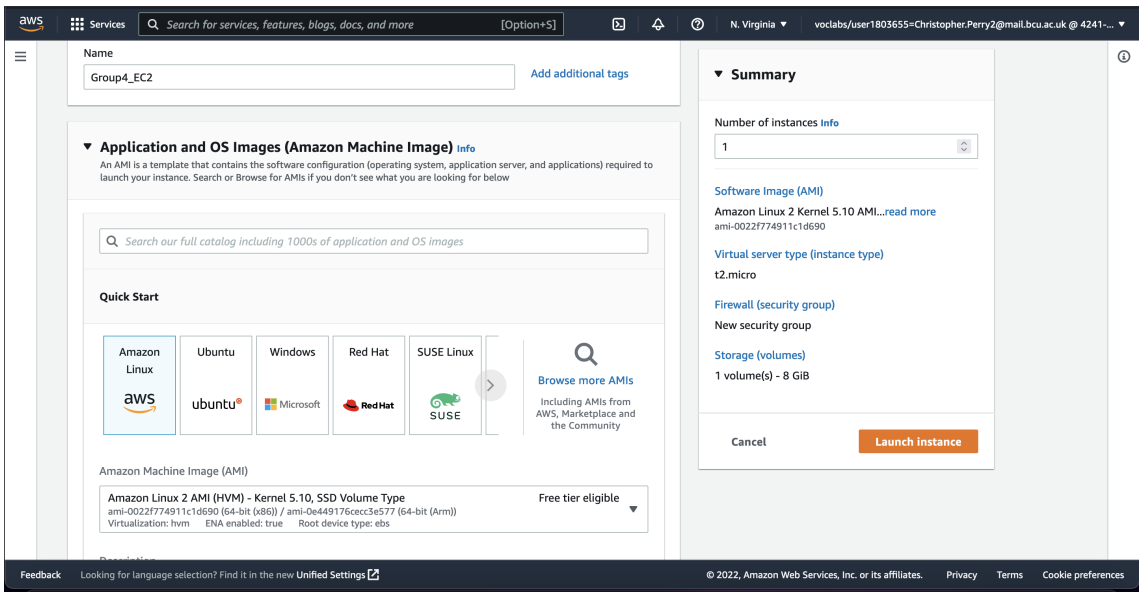

Figure A.6: Create Instance - Application and OS Images

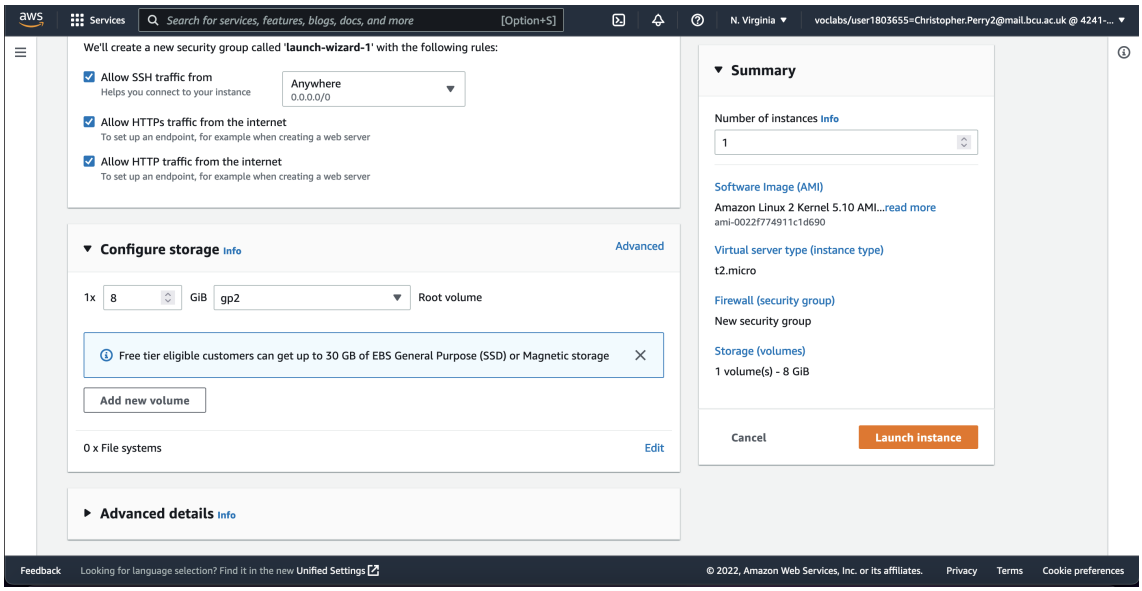

Figure A.7: Create Instance - Configure Storage

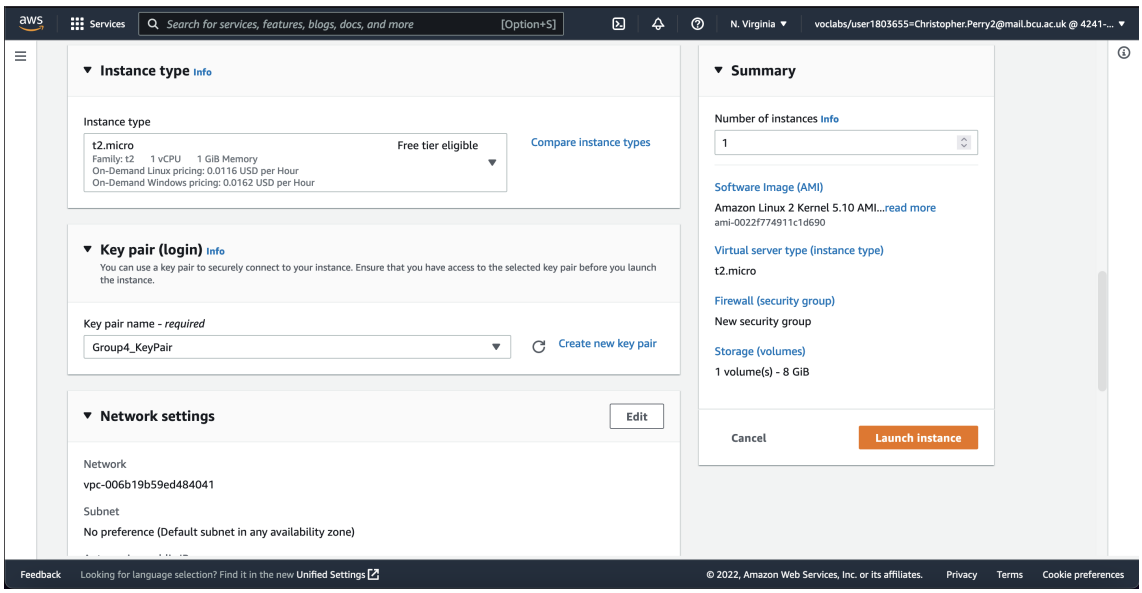

Figure A.8: Create Instance - Instance Type

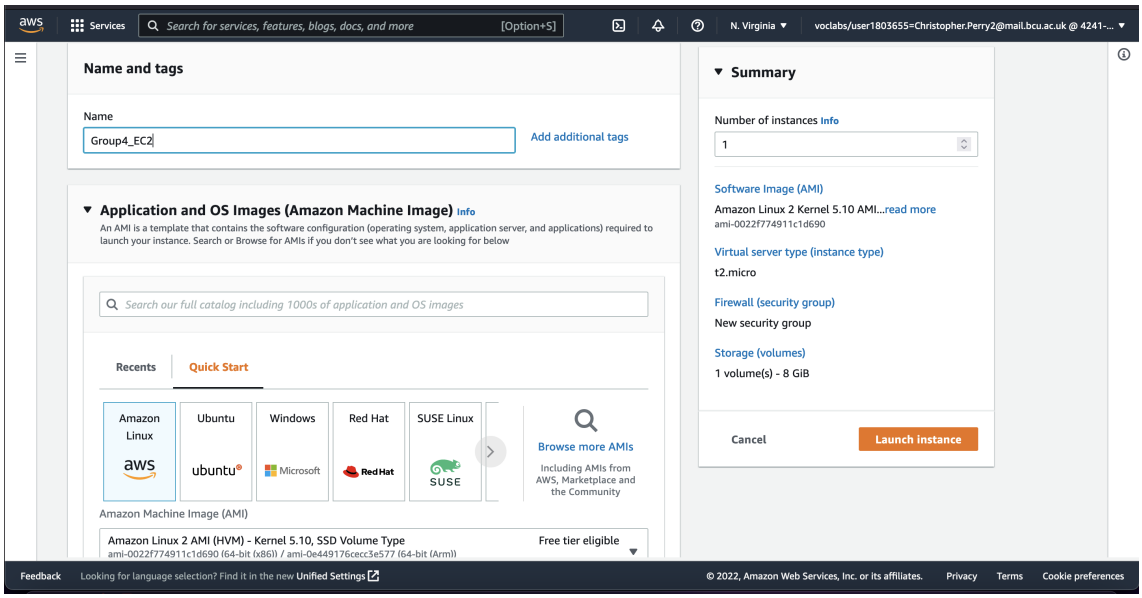

Figure A.9: Create Instance - Name & Tags

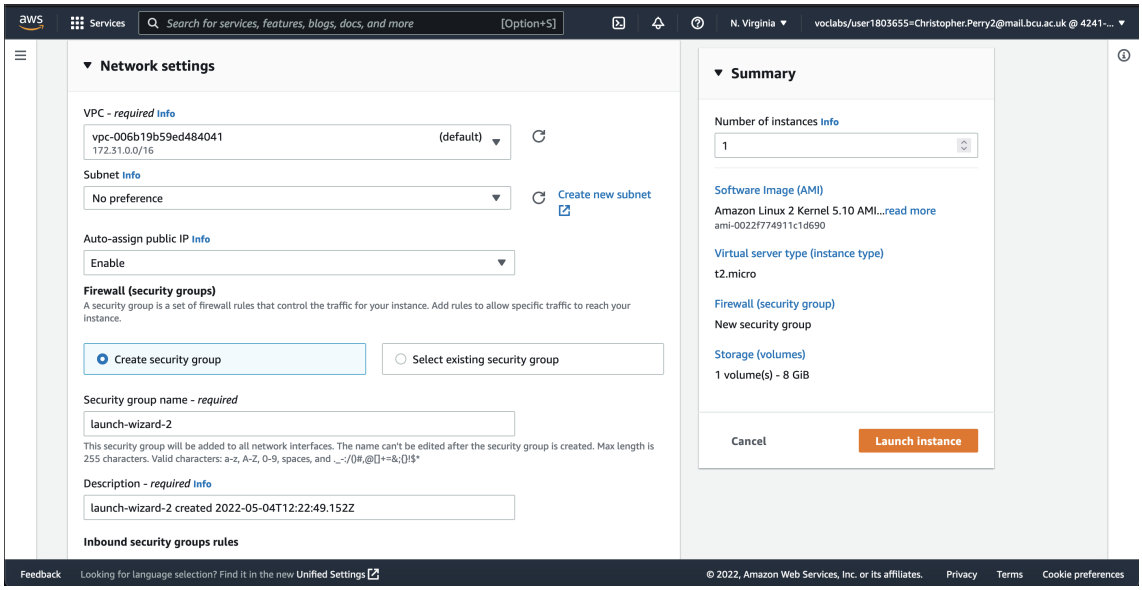

Figure A.10: Create Instance - Network Settings

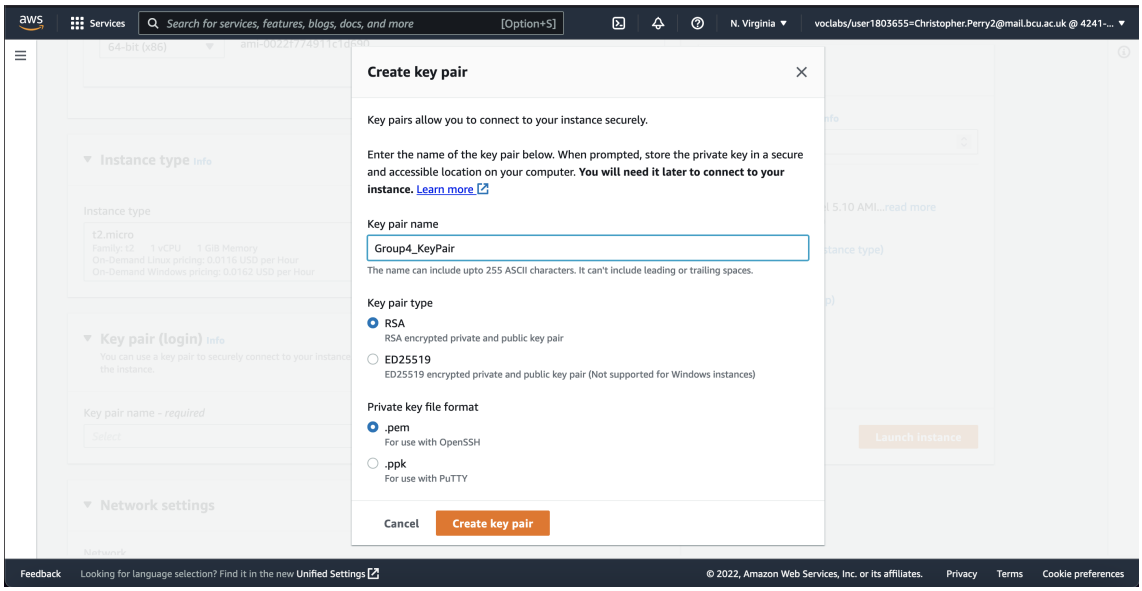

Figure A.11: Creating Key Pair

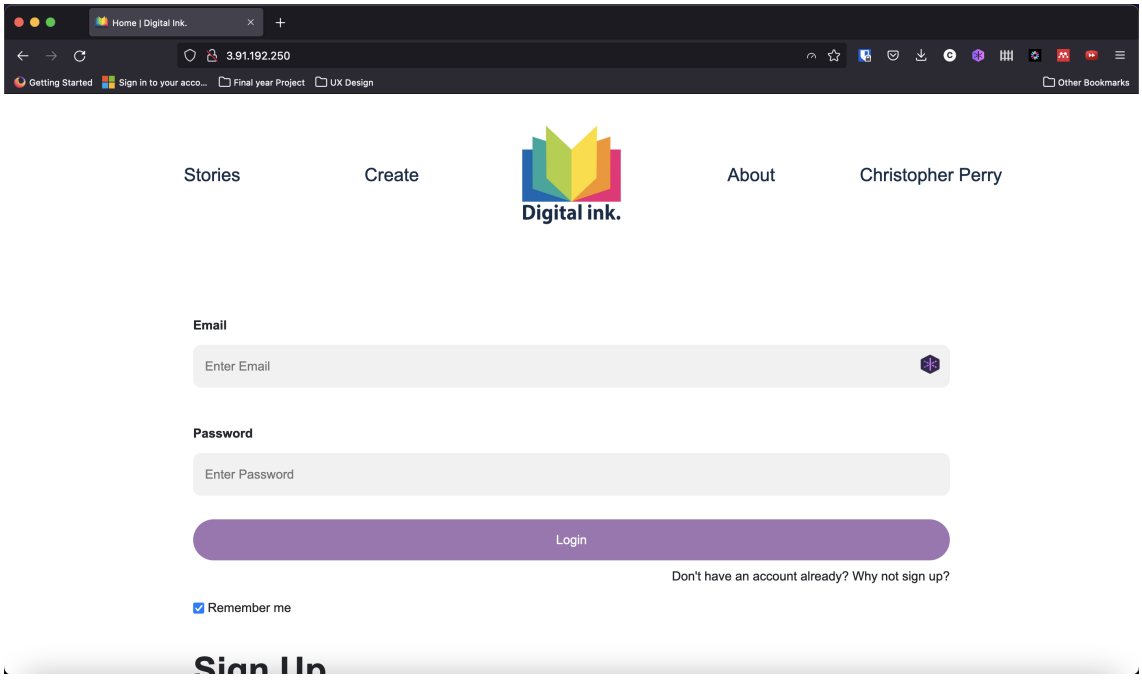

Figure A.12: Digital Ink

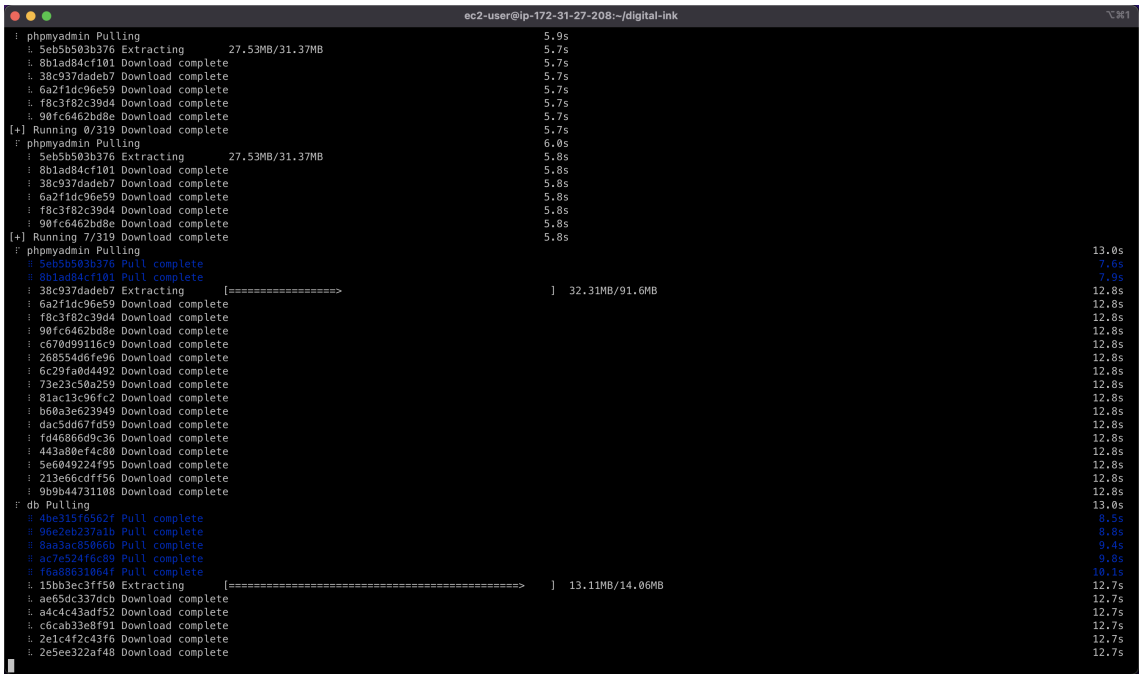

Figure A.13: Docker Compose

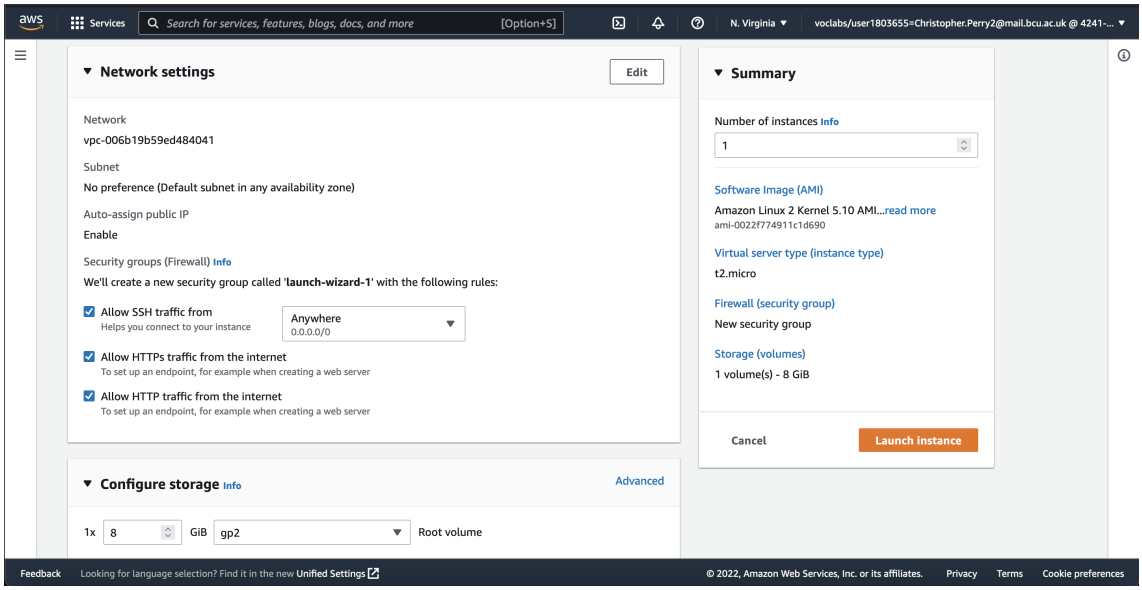

Figure A.14: Edit Instance - Network Settings

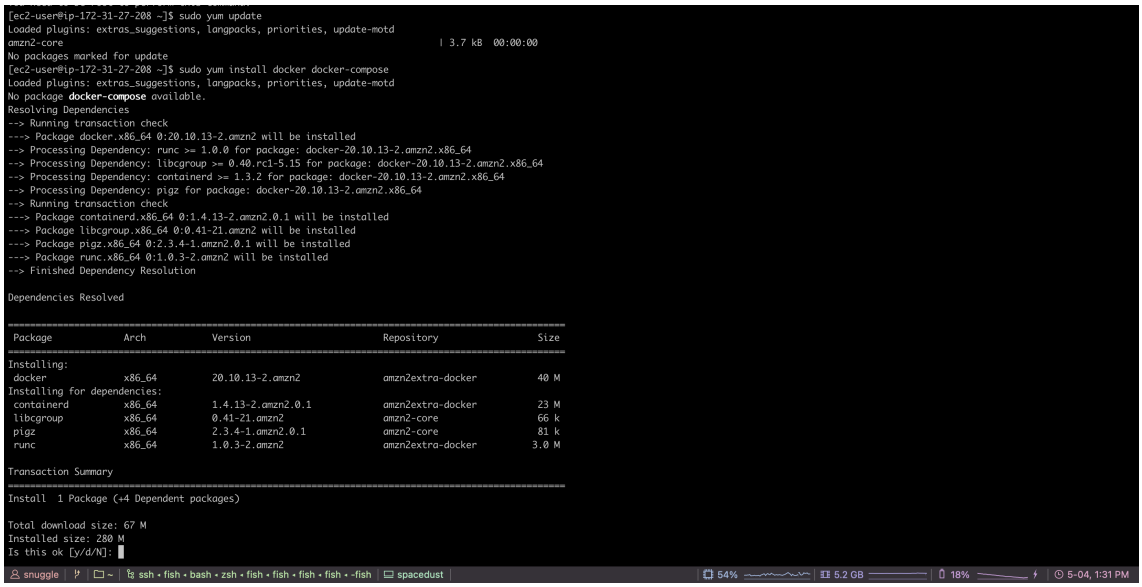

Figure A.15: Installing Docker using Package Manager

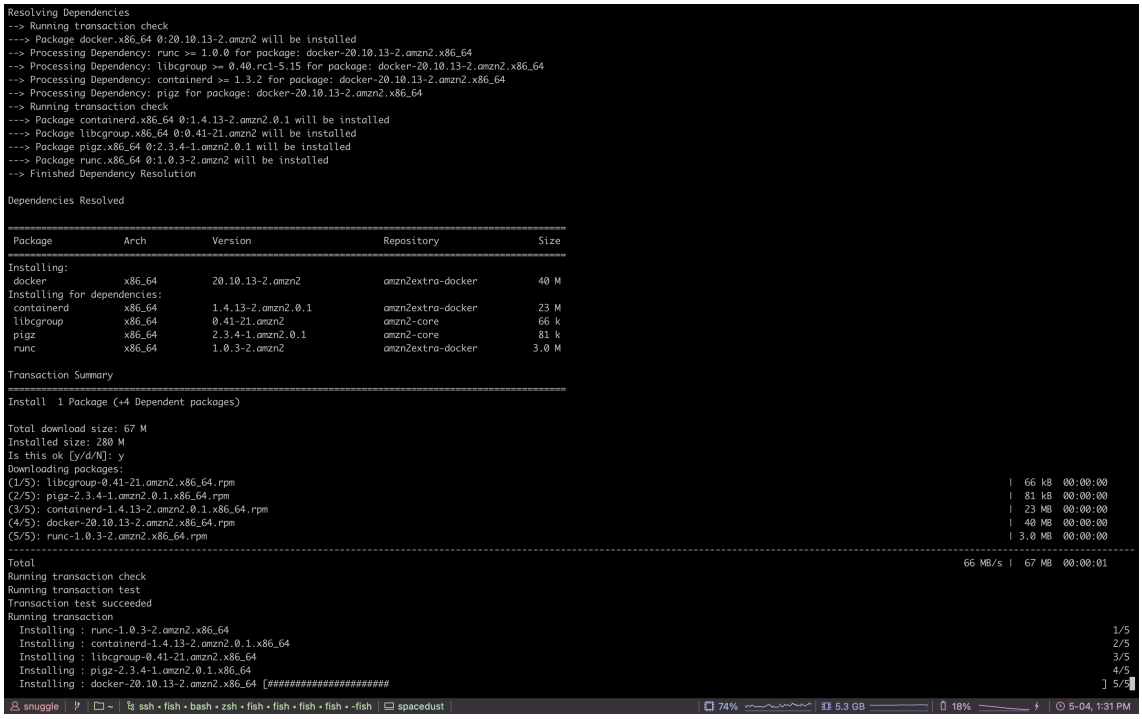

Figure A.16: Installing Docker using Package Manager (In Progress)

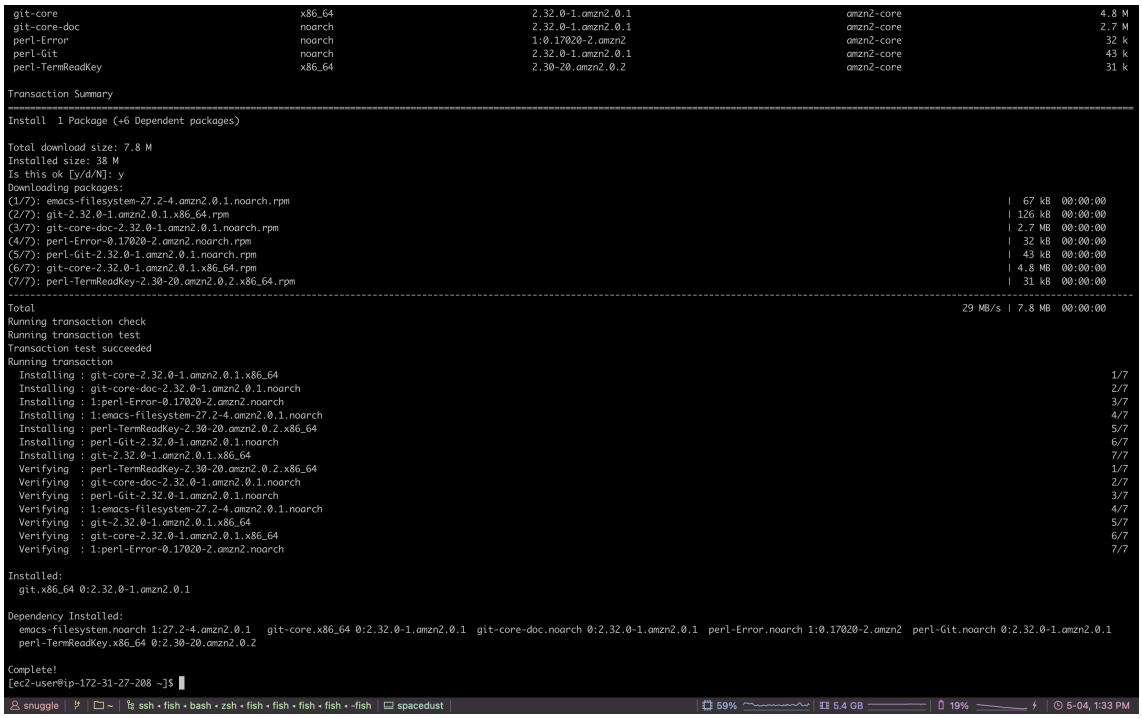

Figure A.17: Installing Git

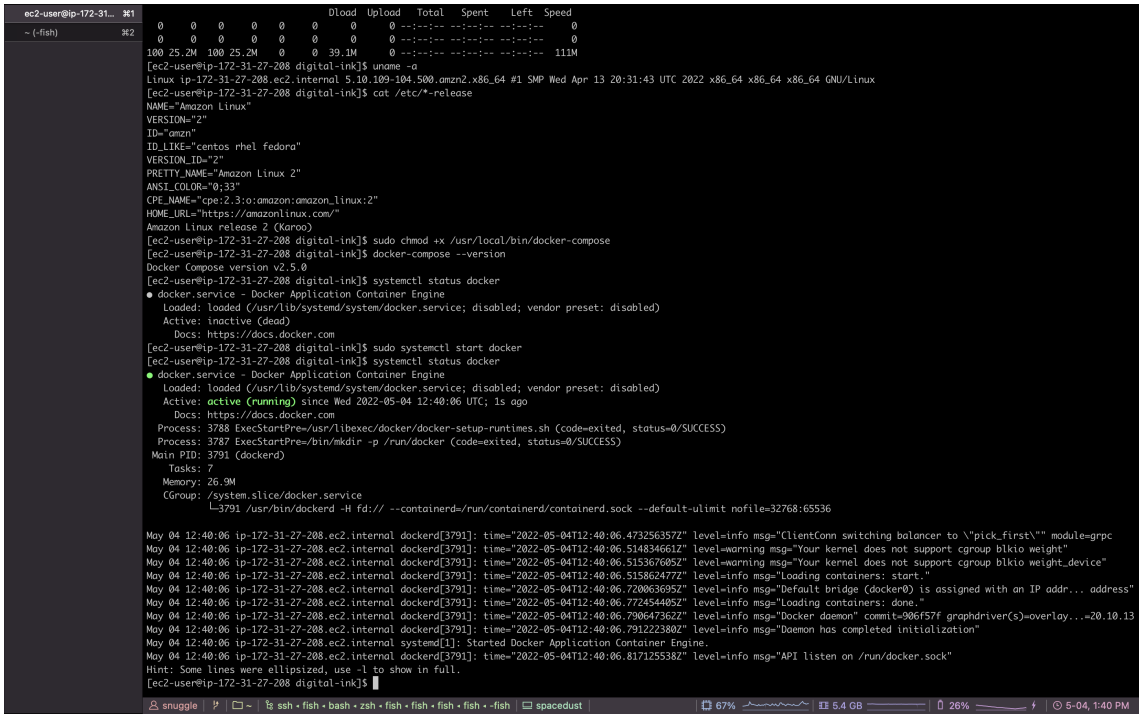

Figure A.18: Starting Docker systemd Service

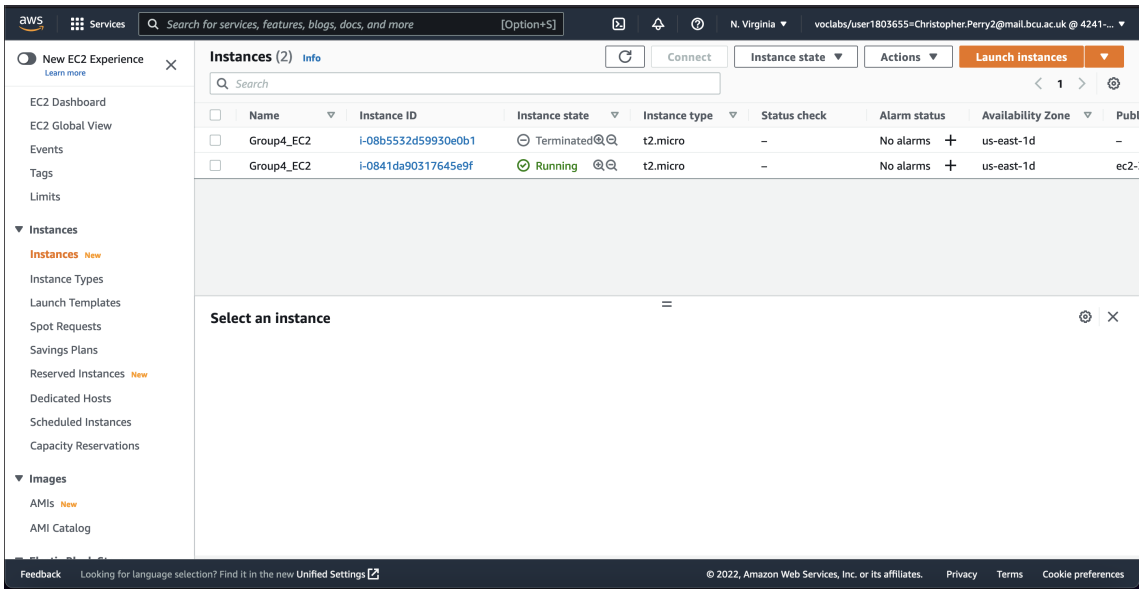

Figure A.19: Instances

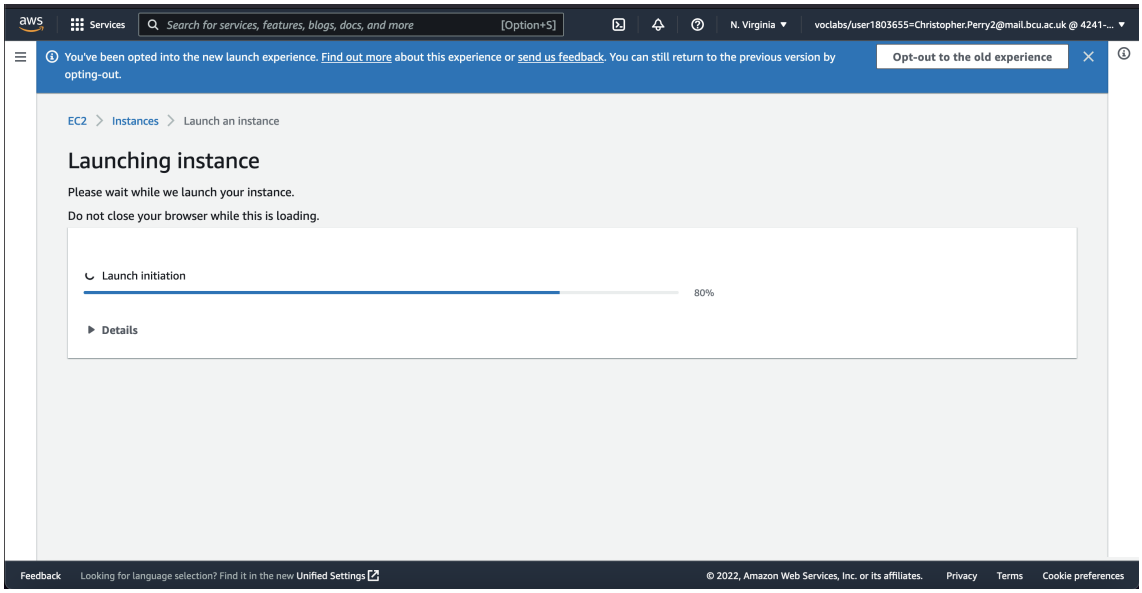

Figure A.20: Launching Instance

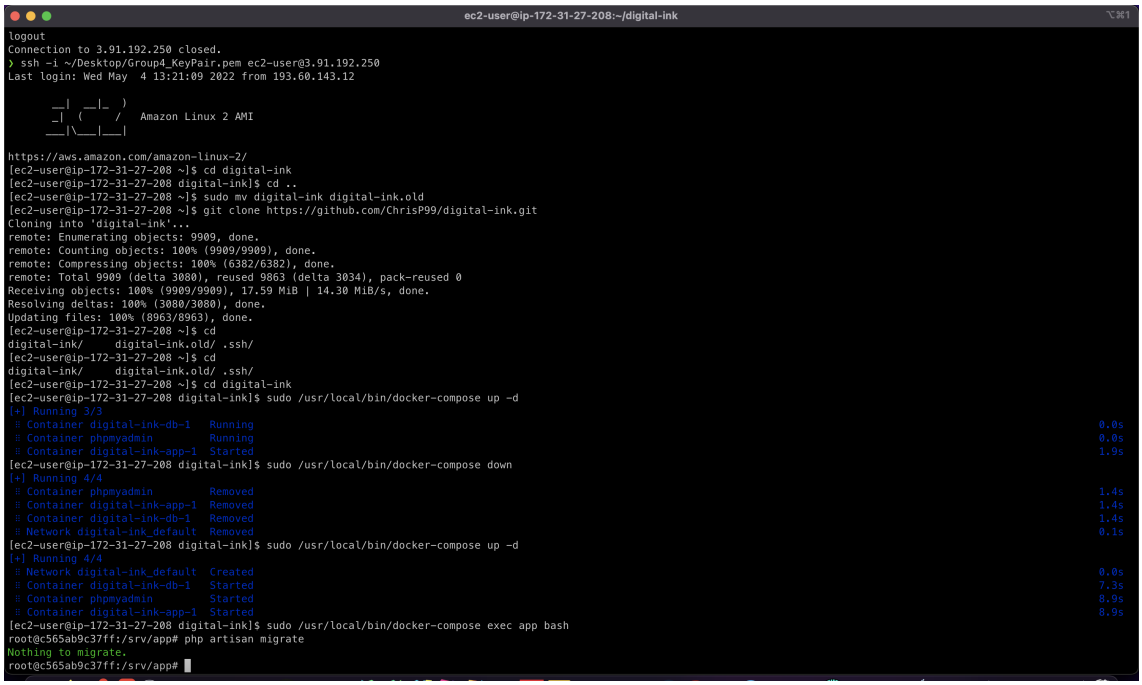

Figure A.21: Log In with Key Pair

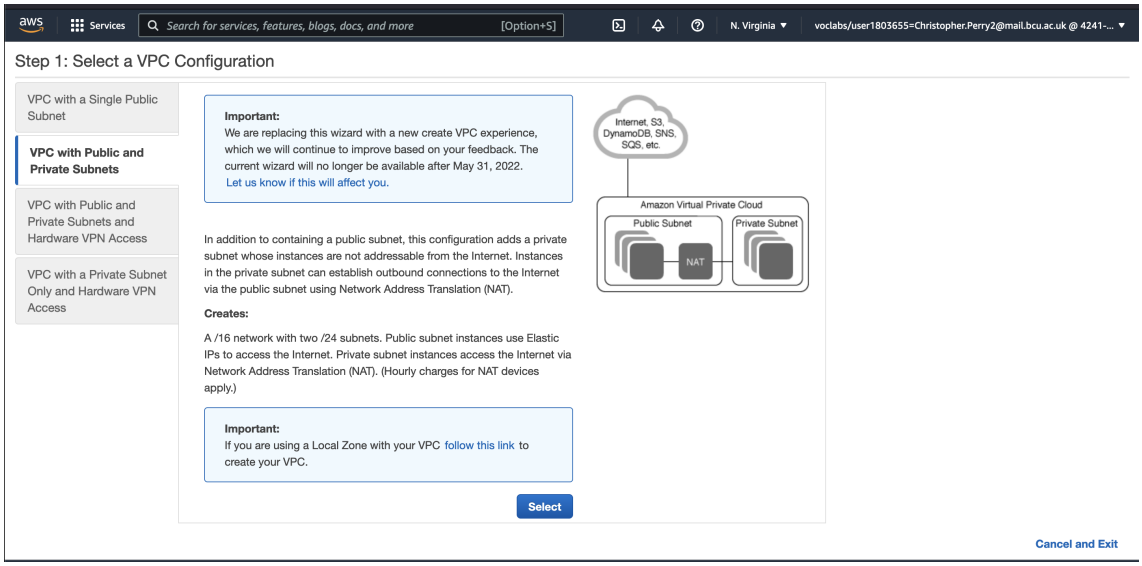

Figure A.22: Selecting a VPC Configuration

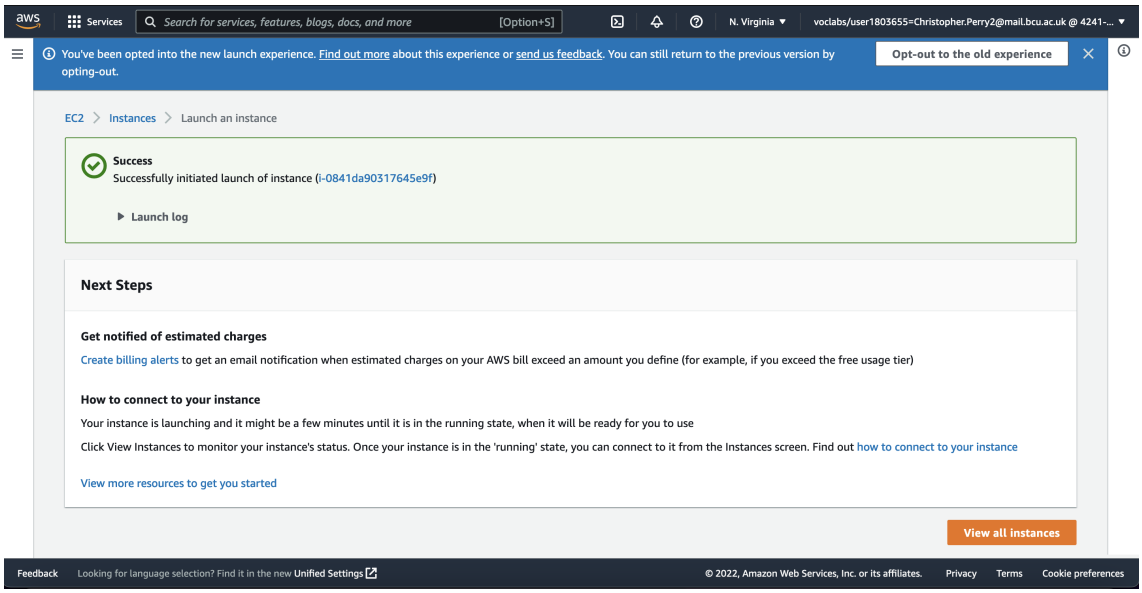

Figure A.23: Successfully Initiated Instance

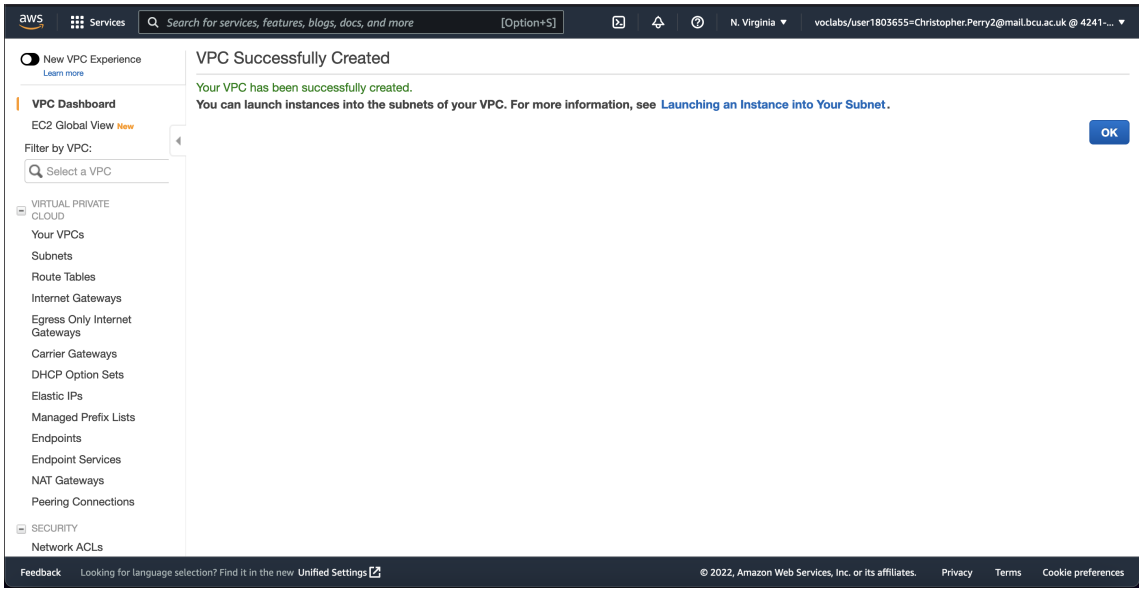

Figure A.24: VPC Successfully Created

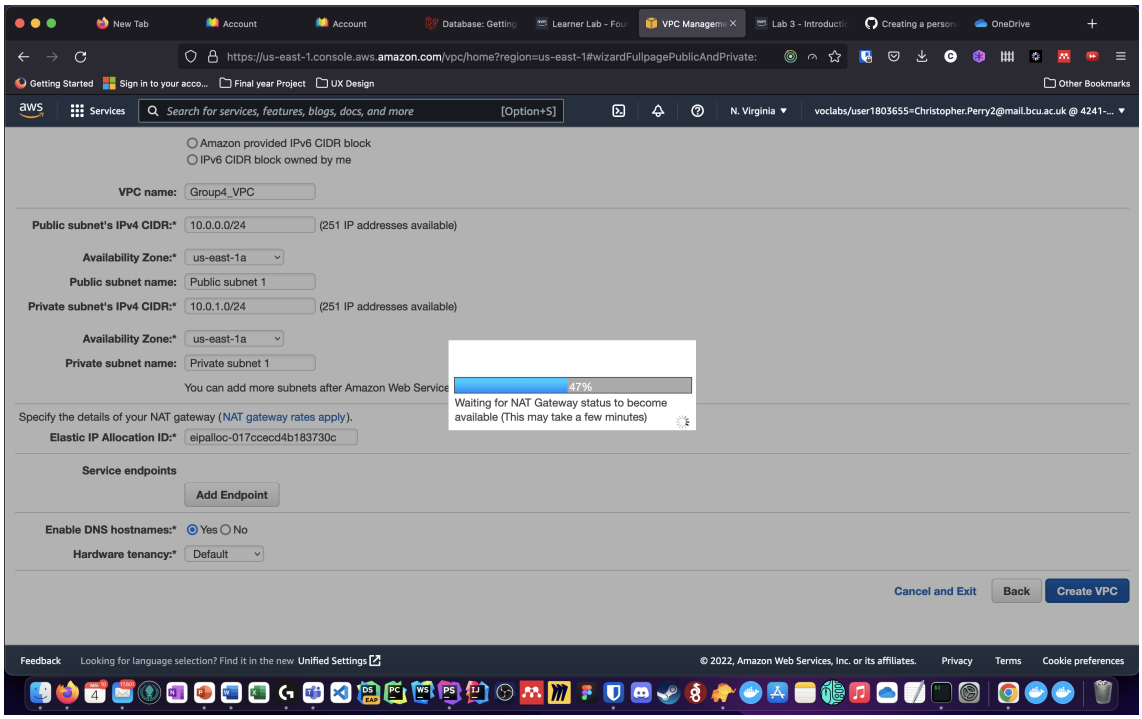

Figure A.25: VPC with Public and Private Subnets, Loading

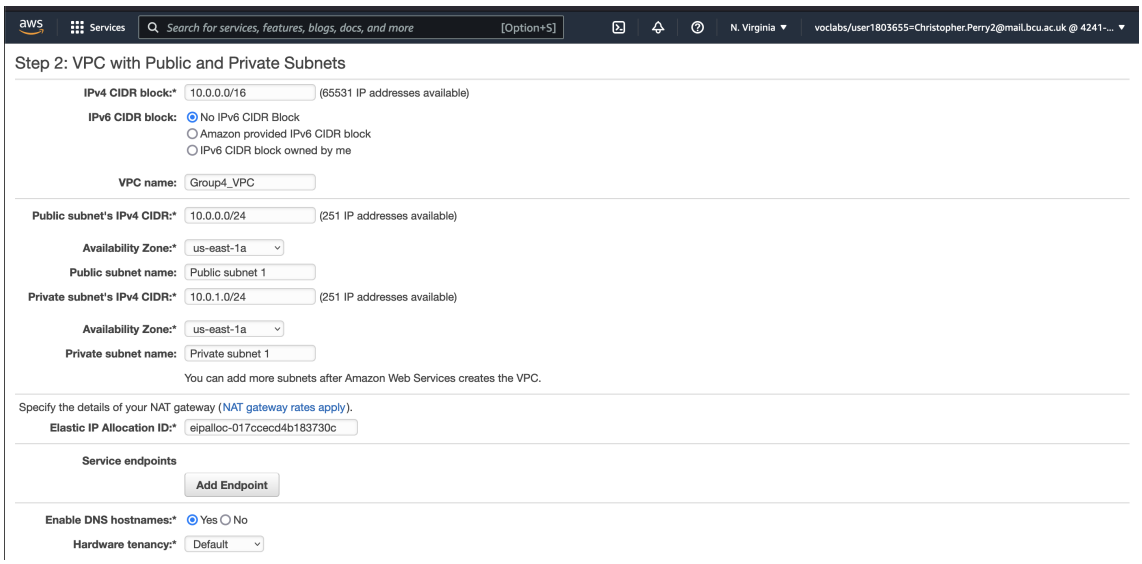

Figure A.26: VPC with Public and Private Subnets

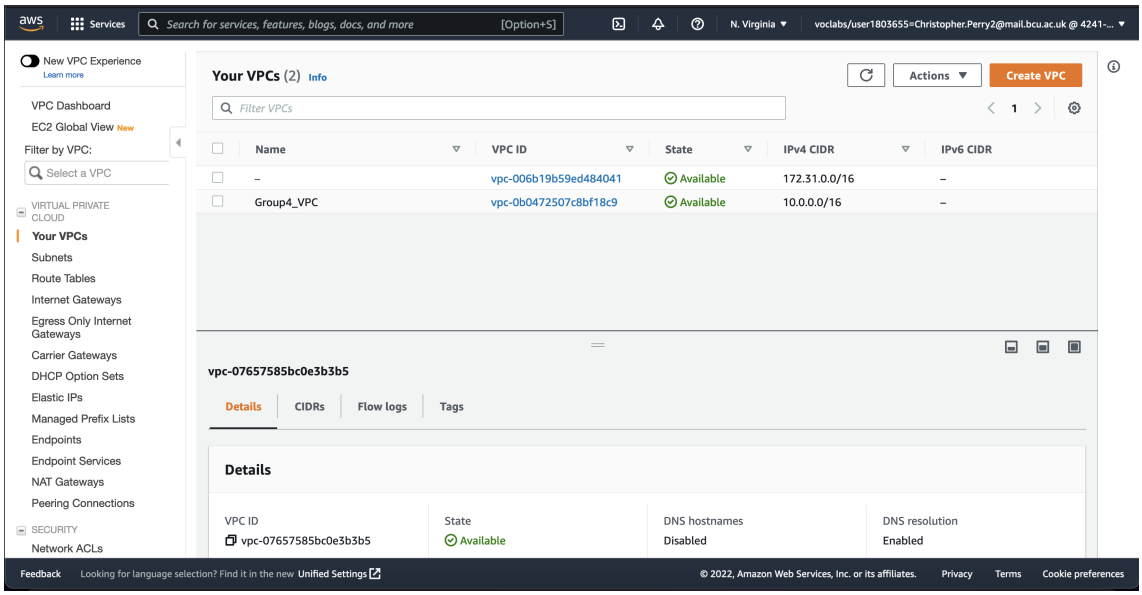

Figure A.27: Your VPCs