

AWS Cloud Labs

Portfolio Samples

Will Kittredge

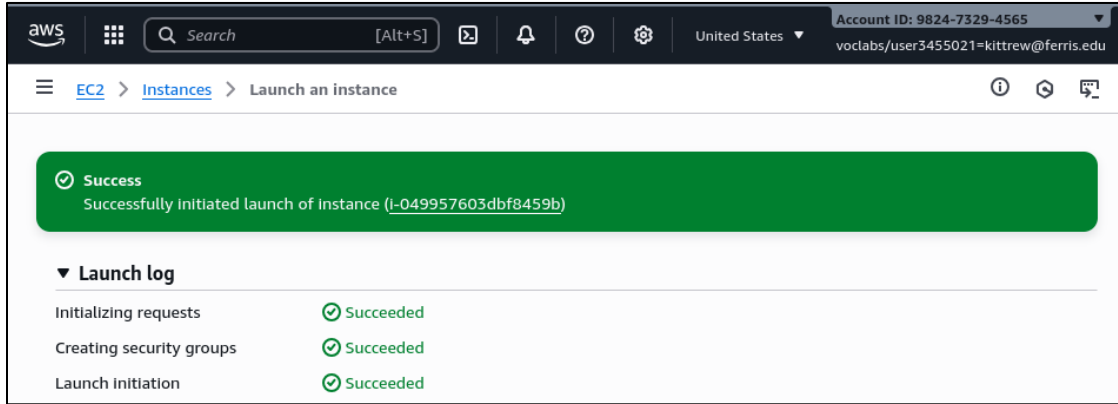
TABLE OF CONTENTS

Lab 3: Introduction to Amazon EC2..... 3
 Screenshots 3
 Synopsis 6
 Lab 4: Working with EBS..... 7
 Screenshots 7
 Synopsis 10
 Challenge Lab 4: Creating a Static Website 11
 Screenshots 11
 Synopsis 17
 Lab 6: Scale and Load Balance your Architecture 18
 Screenshots 18
 Synopsis 23
 Challenge Lab 6: Migrating a Database..... 24
 Screenshots 24
 Synopsis 27
 Challenge Lab 7: VPC Networking Environment 28
 Screenshots 28
 Synopsis 32
 Lab 8: Create a VPC Peering Connection..... 33
 Screenshots 33
 Synopsis 36
 Challenge Lab 10: Scalable/Available Café 37
 Screenshots 37
 Synopsis 42

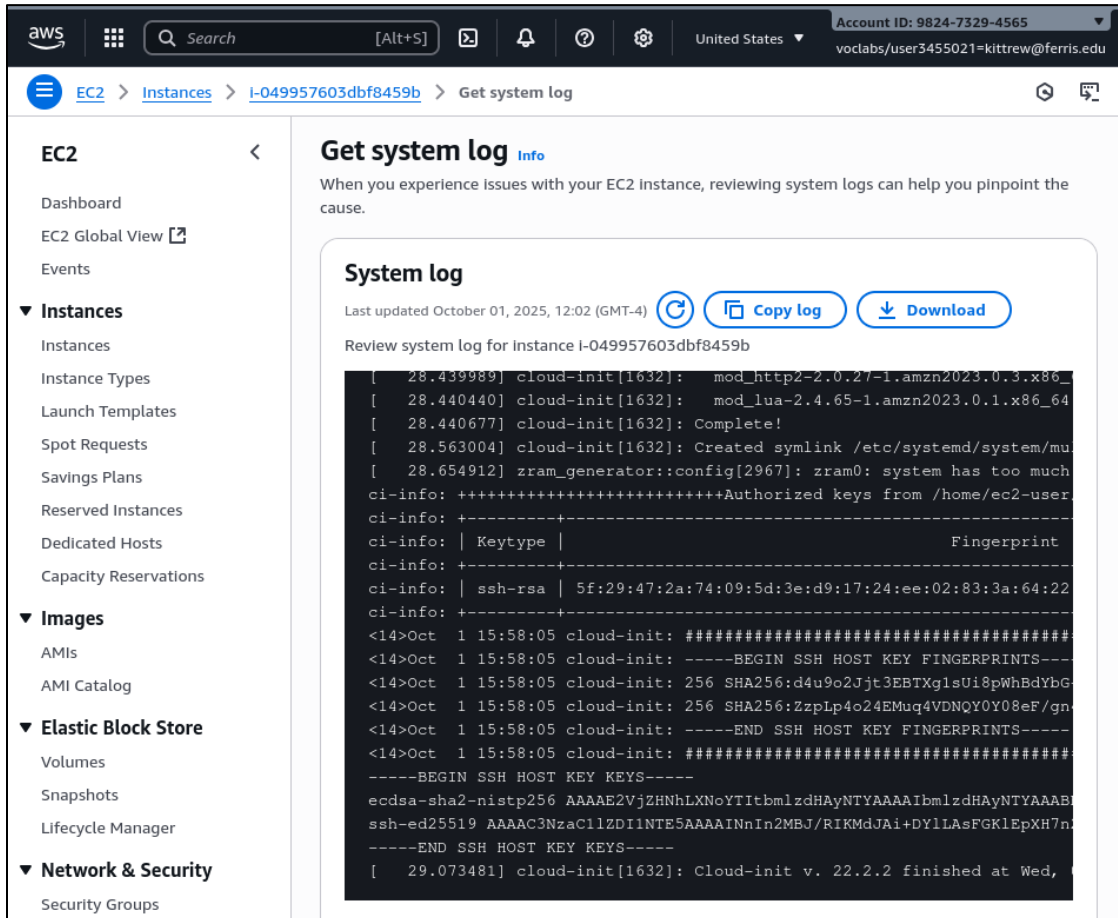
LAB 3: INTRODUCTION TO AMAZON EC2

SCREENSHOTS

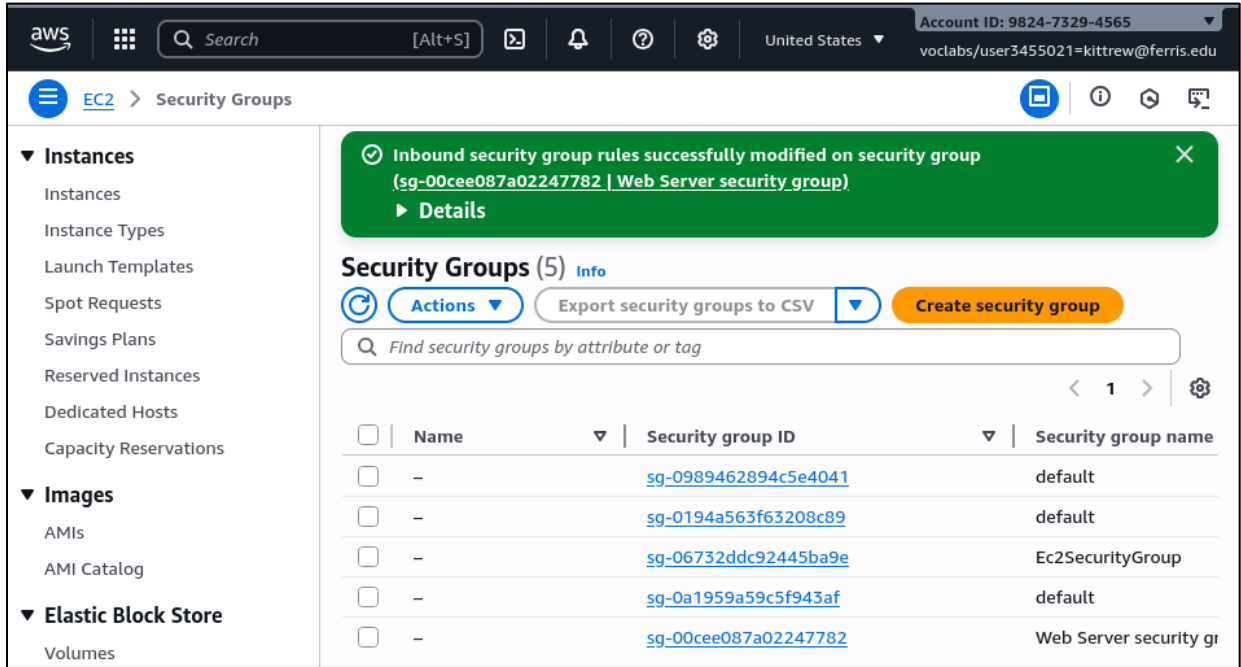
Launched EC2 Instance



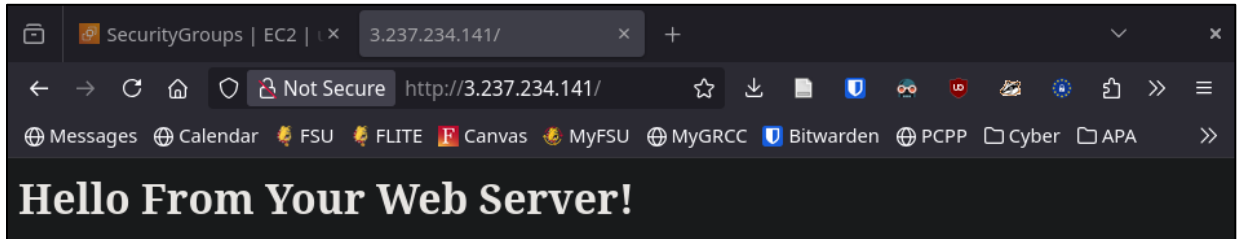
Checked System Log



Modified Security Group Rules



Tested Connection to EC2 Instance



Modified/Resized Volume

The screenshot shows the AWS console interface. At the top, there's a navigation bar with the AWS logo, a search bar, and account information (Account ID: 9824-7329-4565, user: voclabs/user3455021=kittrew@ferris.edu). The left sidebar shows the 'Instances' menu. The main content area features a blue notification banner: 'Requested volume modification for volume vol-07dd311d7f604f1a7. The volume is being modified.' Below this, the 'Volumes (1)' section is active, showing a search bar and a table of volumes. The table has columns for Name, Volume ID, Type, Size, and IOPS. One volume is listed: 'vol-07dd311d7f604f1a7' of type 'gp3', size '8 GiB', and '3000' IOPS.

Name	Volume ID	Type	Size	IOPS
	vol-07dd311d7f604f1a7	gp3	8 GiB	3000

Tested Stop Protection

The screenshot shows the AWS console interface. At the top, there's a navigation bar with the AWS logo, a search bar, and account information (Account ID: 9824-7329-4565, user: voclabs/user3455021=kittrew@ferris.edu). The left sidebar shows the 'EC2' menu. The main content area features a red error banner: 'Failed to stop the instance i-049957603dbf8459b. The instance 'i-049957603dbf8459b' may not be stopped. Modify its 'disableApiStop' instance attribute and try again.' Below this, the 'Instances (1/2)' section is active, showing a search bar and a table of instances. The table has columns for Name, Instance ID, Instance state, and Instance type. Two instances are listed: 'Bastion Host' (Instance ID: i-0a2c7119e40d31412, state: Running, type: t2.micro) and 'Web Server' (Instance ID: i-049957603dbf8459b, state: Running, type: t2.small).

Name	Instance ID	Instance state	Instance type
Bastion Host	i-0a2c7119e40d31412	Running	t2.micro
Web Server	i-049957603dbf8459b	Running	t2.small

SYNOPSIS

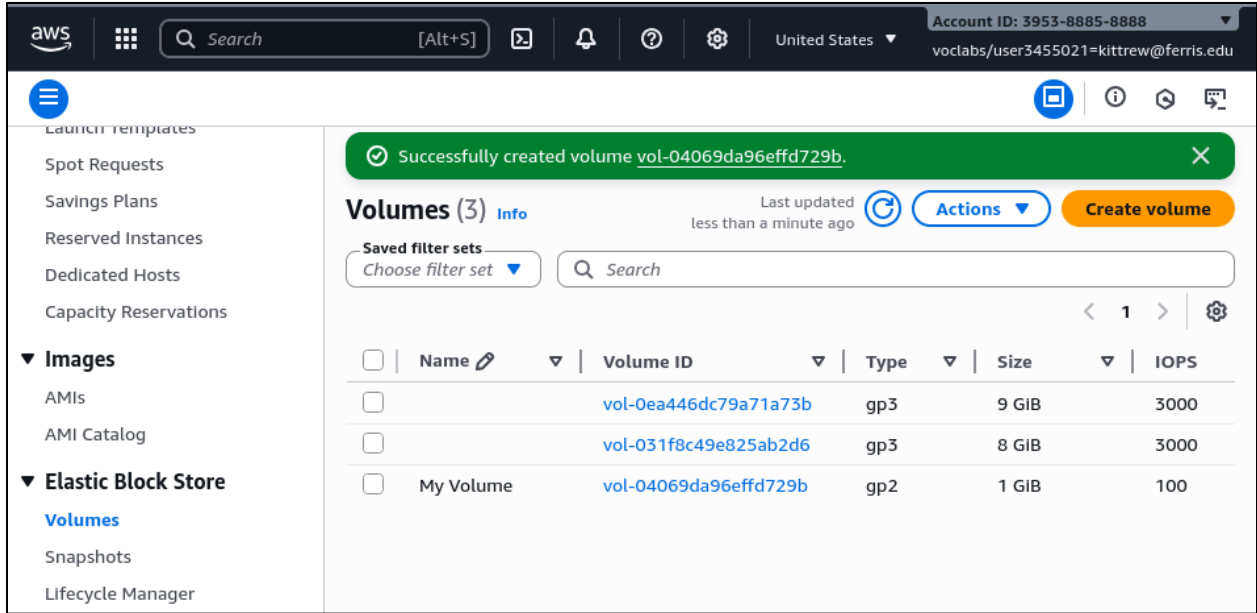
The main purpose of this lab was to provide an introduction to the Amazon Elastic Compute Cloud (EC2) service within the AWS environment. The instructions cover basic (yet fundamental) tasks like launching and monitoring instances, configuring network settings, and modifying resource allocations.

Since we had already done a few other labs that leveraged EC2 instances prior to this one, this was not actually our first time using the service. However, previous labs did not go into nearly as many EC2-specific details compared to this one – any of their EC2-related instructions usually just served as a way to prepare the lab environment for upcoming steps that were more pertinent to lab’s focus. This lab, although relatively short and simple, was useful because it put that information into one package for us to familiarize ourselves with. After having completed the lab, I’m confident in my ability to create and deploy simple EC2 virtual machine configurations from the setup wizard. Since the process is conceptually similar to creating a virtual machine instance in other environments that I’m familiar with (e.g., locally on your own computer), it felt relatively easy to get my bearings once I understood Amazon’s terminology. When it comes to deploying and monitoring more advanced virtual machine configurations, I think that I still need more practice using EC2 before I become confident in my ability to leverage its features appropriately/effectively.

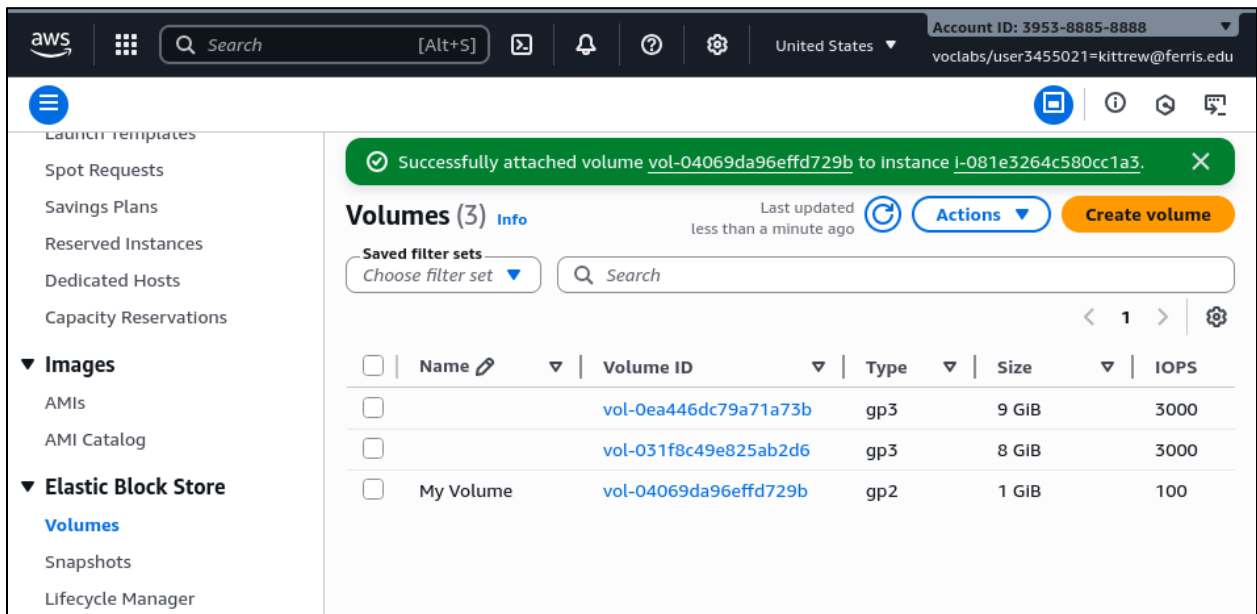
LAB 4: WORKING WITH EBS

SCREENSHOTS

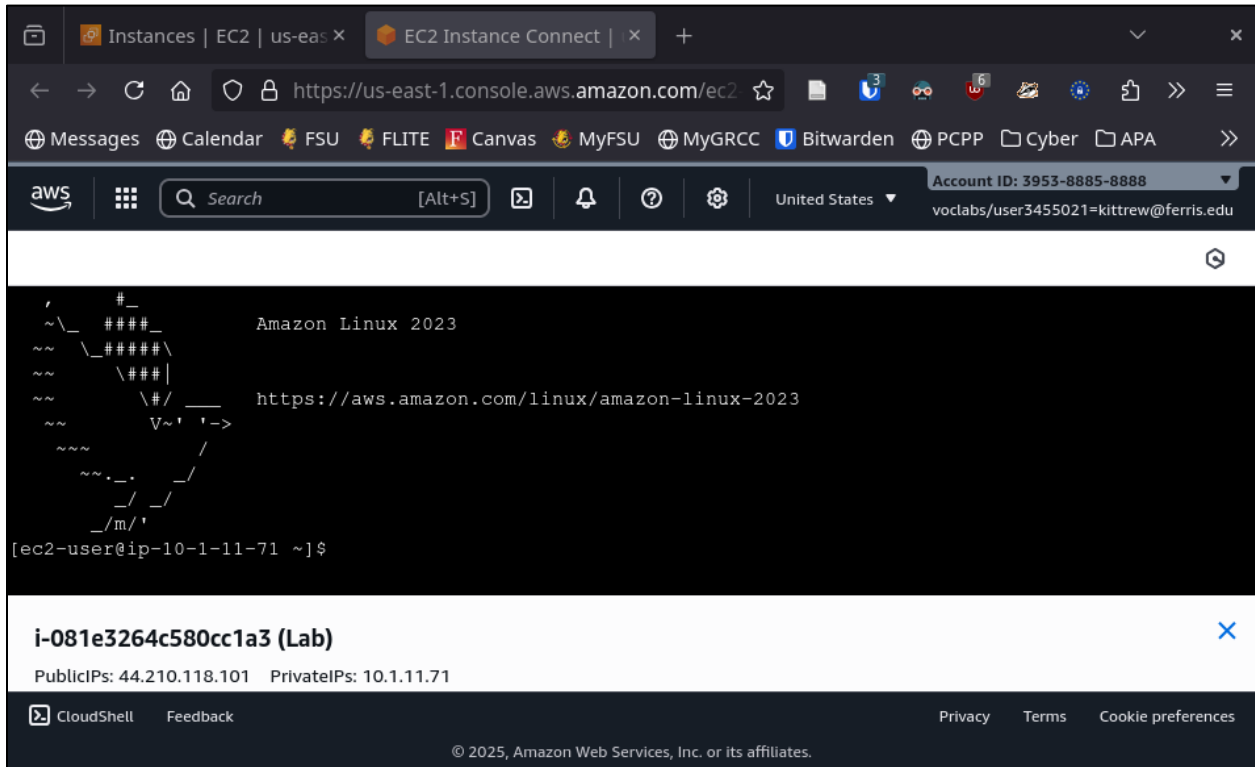
Created New Volume



Attached New Volume



Connected to EC2 Instance



Created Filesystem, Mounted Volume, and Wrote Data

```
[ec2-user@ip-10-1-11-71 ~]$ cat /etc/fstab
#
UUID=de5c5998-22a2-4734-85d8-c8f972ff40e9 / xfs defaults,noatime 1 1
UUID=3021-32B7 /boot/efi vfat defaults,noatime,uid=0,gid=0,umask=0077,shortname=winnt,x
-systemd.automount 0 2
/dev/sdf /mnt/data-store ext3 defaults,noatime 1 2
[ec2-user@ip-10-1-11-71 ~]$ df -h
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        4.0M   0  4.0M   0% /dev
tmpfs           475M   0  475M   0% /dev/shm
tmpfs           190M  440K  190M   1% /run
/dev/xvda1      8.0G  1.6G  6.4G  21% /
tmpfs           475M   0  475M   0% /tmp
/dev/xvda128    10M  1.3M  8.7M  13% /boot/efi
tmpfs           95M   0   95M   0% /run/user/1000
/dev/xvdf       975M   60K  924M   1% /mnt/data-store
[ec2-user@ip-10-1-11-71 ~]$ sudo sh -c "echo some text has been written > /mnt/data-store/file.txt"
[ec2-user@ip-10-1-11-71 ~]$ cat /mnt/data-store/file.txt
some text has been written
[ec2-user@ip-10-1-11-71 ~]$
```

Removed Data (After Snapshot)

```
[ec2-user@ip-10-1-11-71 ~]$ sudo rm /mnt/data-store/file.txt
[ec2-user@ip-10-1-11-71 ~]$ ls /mnt/data-store/
lost+found
[ec2-user@ip-10-1-11-71 ~]$
```

Restored Snapshot to Volume and Attached to Instance

The screenshot shows the AWS Management Console interface. At the top, there is a navigation bar with the AWS logo, a search bar, and account information (Account ID: 3953-8885-8888). A green notification banner at the top of the main content area states: "Successfully attached volume vol-03ebd57edbf4098d9 to instance i-081e3264c580cc1a3." Below this, the "Volumes (4)" section is displayed, showing a table of volumes. The table has columns for Name, Volume ID, Type, Size, and IOPS. The "Restored Volume" is highlighted in the table.

Name	Volume ID	Type	Size	IOPS
	vol-0ea446dc79a71a73b	gp3	9 GiB	3000
Restored Volume	vol-03ebd57edbf4098d9	gp2	1 GiB	100
	vol-031f8c49e825ab2d6	gp3	8 GiB	3000
My Volume	vol-04069da96effd729b	gp2	1 GiB	100

Tested Snapshot

```
[ec2-user@ip-10-1-11-71 ~]$ sudo rm /mnt/data-store/file.txt
[ec2-user@ip-10-1-11-71 ~]$ ls /mnt/data-store/
lost+found
[ec2-user@ip-10-1-11-71 ~]$ sudo mkdir /mnt/data-store2
[ec2-user@ip-10-1-11-71 ~]$ sudo mount /dev/sdg /mnt/data-store2
[ec2-user@ip-10-1-11-71 ~]$ ls /mnt/data-store2
file.txt lost+found
[ec2-user@ip-10-1-11-71 ~]$
```

SYNOPSIS

This lab served as an introduction to using the Elastic Block Store (EBS) for Amazon EC2 instances. Instructions covered basic tasks such as creating and attaching volumes, making and utilizing snapshots, as well as creating filesystems on the EBS volumes that we made (and mounting them).

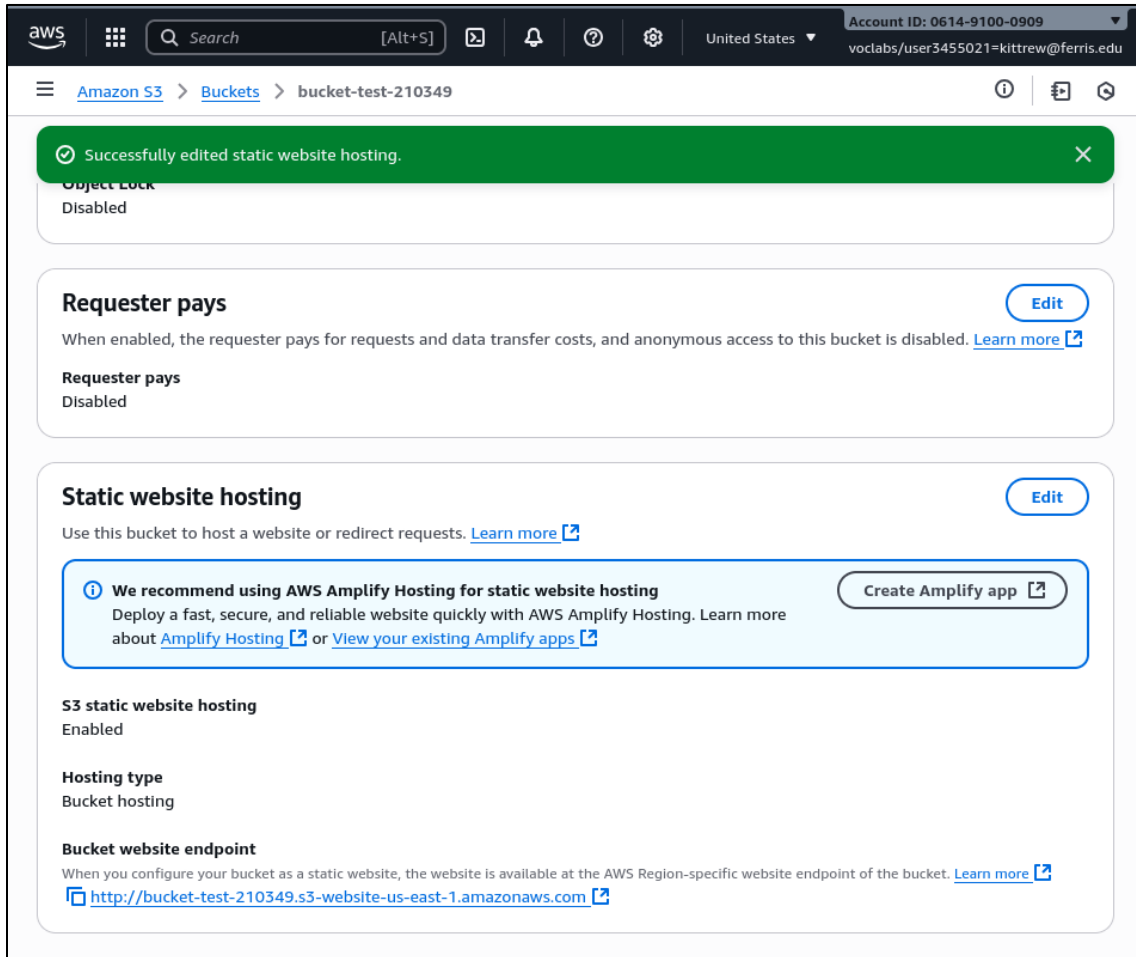
Generally, I found this lab to be relatively simple and straightforward compared to some of the other ones. However, the concepts that it covers are undoubtedly foundational when it comes to managing virtual machines/resources. I would argue that the ability to easily create and deploy snapshots (as well as the wrangling of virtual hardware, such as disks) is one of the primary technical benefits of utilizing virtualization and cloud technology – so it's important that we actually know how to perform those actions in AWS. Like with other concepts we've covered in previous labs (e.g., launching EC2 instances, configuring firewall rules), I found it easy to understand how to manage volumes and snapshots using EBS due to prior experience with other virtual environments.

After having completed the lab, I feel like I have most of the skills that I would need to create a reliable simple environment in AWS – such as moving a homelab with a few test VMs into the cloud. We know how to launch instances from the EC2 lab, how to manage their volumes and snapshots from this lab, how to create the desired networking configuration(s) from the VPC lab, and various supporting concepts from the rest of the labs, assignments, and course content.

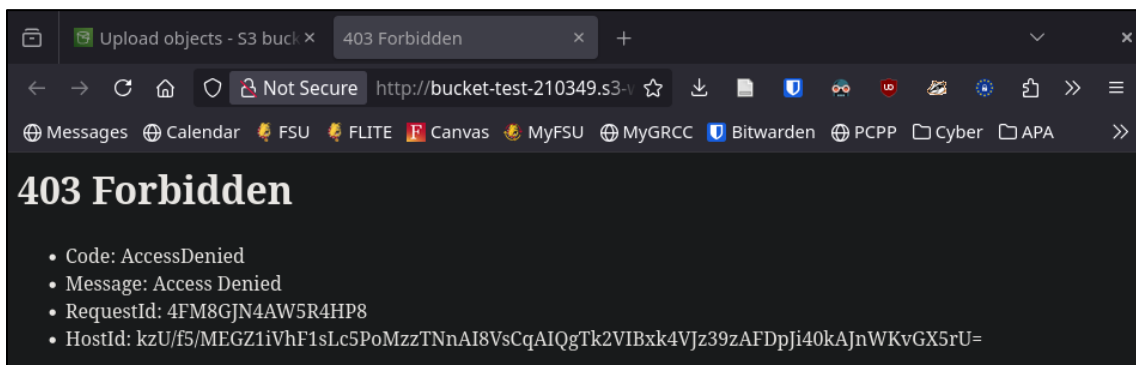
CHALLENGE LAB 4: CREATING A STATIC WEBSITE

SCREENSHOTS

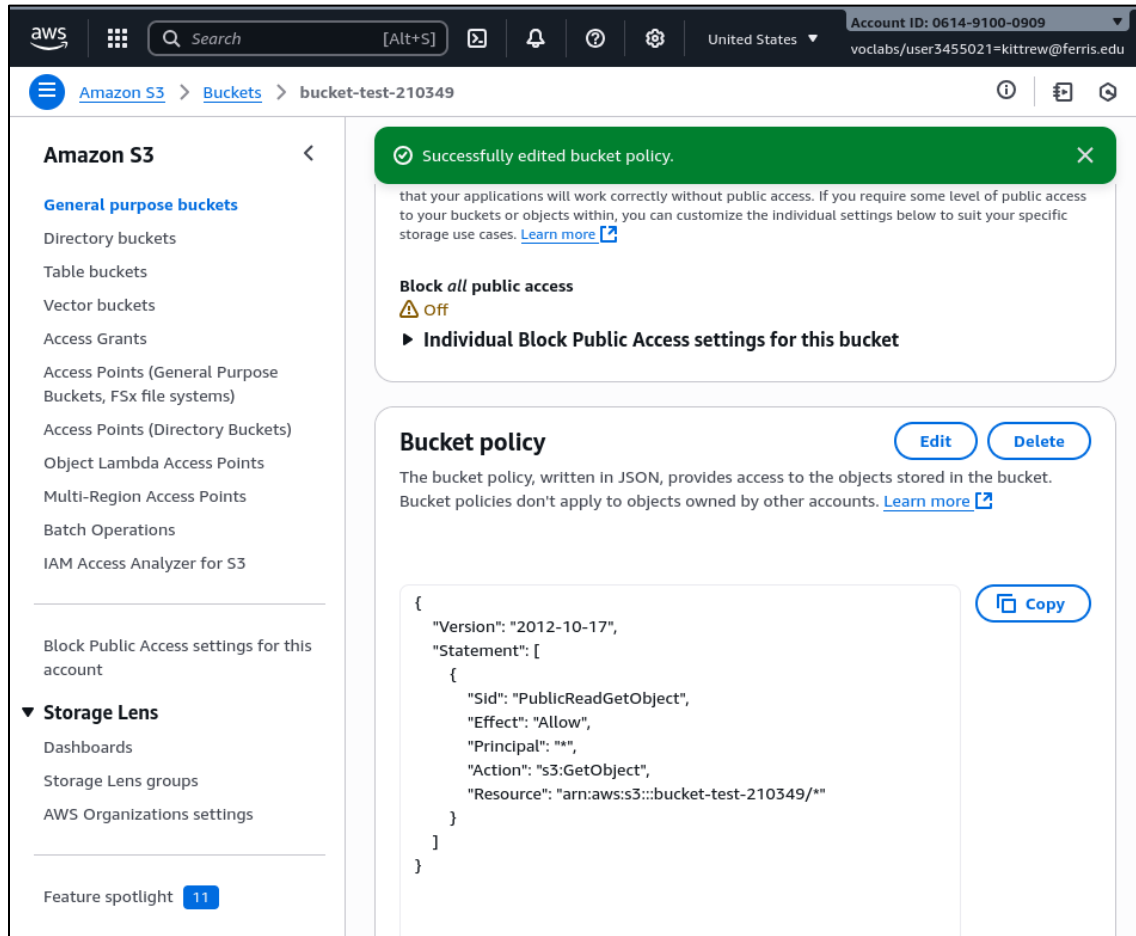
Created Bucket and Enabled Static Website Hosting



Tested Connection after Uploading Files



Updated Bucket Policy





Tested Connection with Updated Policy

bucket-test-210349 - S3 x Welcome to the Café! x

Not Secure http://bucket-test-210349.s3- ☆


Messages Calendar FSU FLITE Canvas MyFSU MyGRCC Bitwarden PCPP Cyber APA

Café




The Café offers an assortment of delicious and delectable pastries and coffees that will put a smile on your face. From cookies to croissants, tarts and cakes, each treat is specially prepared to excite your tastebuds and brighten your day!


Frank bakes a rich variety of cookies. Try them all!



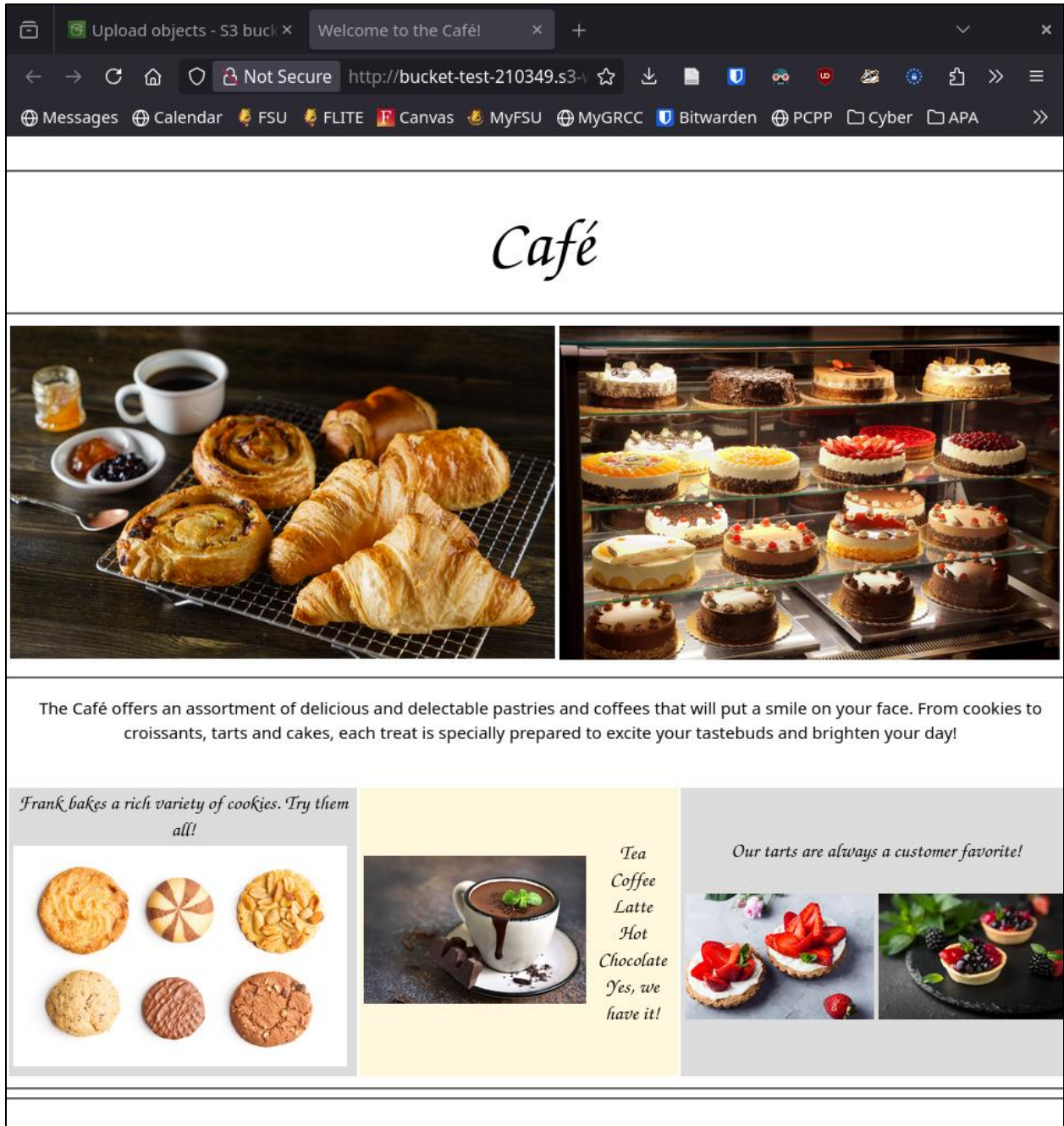
*Tea
Coffee
Latte
Hot
Chocolate
Yes, we have it!*



Our tarts are always a customer favorite!



Modified `index.html` File after Enabling Versioning



The image shows a browser window with the URL `http://bucket-test-210349.s3-`. The website content includes:

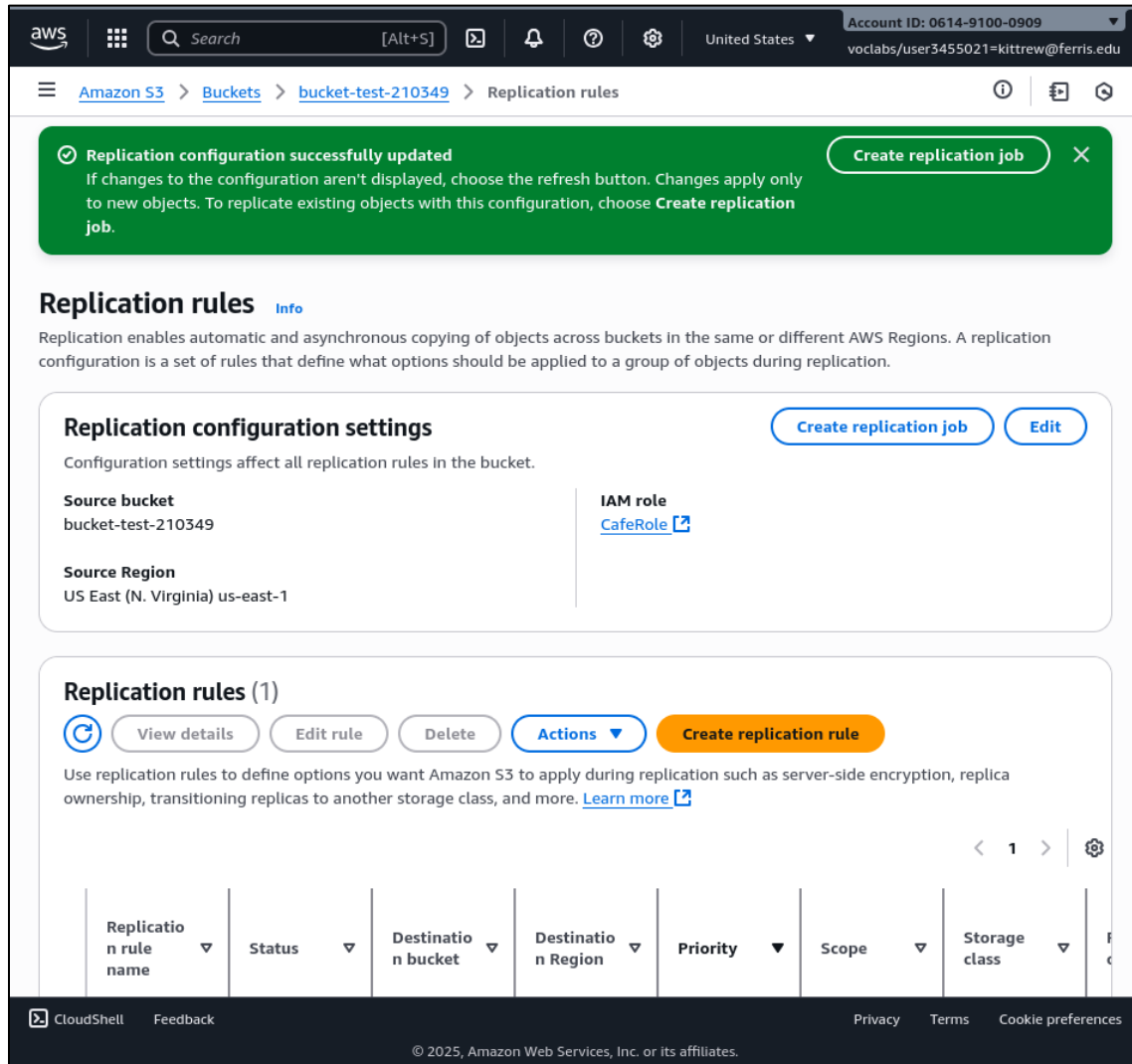
- A header with the word *Café* in a cursive font.
- Two side-by-side images: the left one shows a coffee cup, a glass of orange juice, and a tray of pastries (croissants and cinnamon rolls); the right one shows a display case filled with various cakes and tarts.
- A paragraph of text: "The Café offers an assortment of delicious and delectable pastries and coffees that will put a smile on your face. From cookies to croissants, tarts and cakes, each treat is specially prepared to excite your tastebuds and brighten your day!"
- A grid of three promotional panels:
 - Left panel: *Frank bakes a rich variety of cookies. Try them all!* with an image of six different cookies.
 - Middle panel: *Tea Coffee Latte Hot Chocolate Yes, we have it!* with an image of a coffee cup.
 - Right panel: *Our tarts are always a customer favorite!* with an image of several tarts.

Created Lifecycle Rules

The screenshot shows the AWS console interface for the 'Lifecycle configuration' page of a bucket named 'bucket-test-210349'. The page title is 'Lifecycle configuration'. Below the title, there is a brief explanation: 'To manage your objects so that they are stored cost effectively throughout their lifecycle, configure their lifecycle. A lifecycle configuration is a set of rules that define actions that Amazon S3 applies to a group of objects. Lifecycle rules run once per day.' Below this, it states 'Default minimum object size for transitions: All storage classes 128K'. The main section is titled 'Lifecycle rules (2)' and includes buttons for 'View details', 'Edit', 'Delete', 'Actions', and 'Create lifecycle rule'. A search bar is present with the placeholder text 'Find lifecycle rules by name'. Below the search bar is a table with the following columns: 'Lifecycle...', 'Status', 'Scope', 'Current ...', 'Noncurrent versions actions', and 'Expired'. Two rules are listed in the table:

Lifecycle...	Status	Scope	Current ...	Noncurrent versions actions	Expired
S3IA-old-objects	Enabled	Entire bucket	-	Transition to Standard-IA	-
delete-old-objects	Enabled	Entire bucket	-	Permanently delete	-

Enabled Cross-Region Replication



SYNOPSIS

This lab was similar to the previous challenge lab we completed where we were tasked with creating a dynamic website. The required files were provided to us (i.e., we did not have to actually structure our own webpage using HTML/CSS), so the challenge was knowing how to properly utilize S3 buckets and their features to create the desired outcome. This primarily involved configuring the bucket correctly, modifying permissions and policies, and enabling versioning.

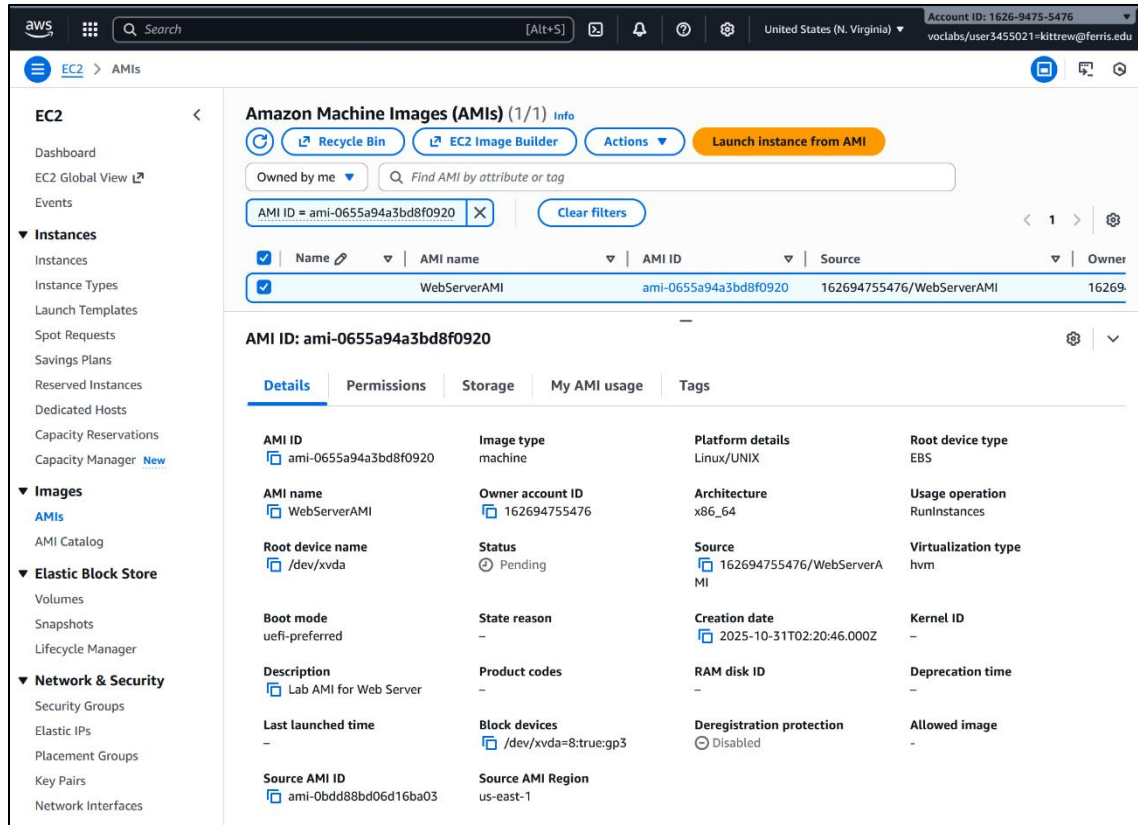
I encountered a few minor issues with this lab, but I was able to progress through the rest of the steps without problems after 10-15 minutes of research and troubleshooting. Creating and applying the bucket policy gave me the most troubles, as I was initially unable to get Amazon to accept the policy I provided. There were also a few quirks that I had to familiarize myself with when it comes to using S3 buckets. In one case, I thought I had completed uploading all of the website files to the bucket I created. I didn't discover until later that file uploads are not automatically initiated after you select a file or folder from your OS with the file picker – you have to scroll to the bottom of the webpage and manually start the upload(s).

After completing the lab, I feel that I could set up a relatively simple static webpage using AWS in a reasonable amount of time and without needing to rely heavily on documentation. The lab content was also applicable to me personally, as I have been slowly chipping away at a personal project that involves creating a static site to host my portfolio and other work that I would like to share publicly.

LAB 6: SCALE AND LOAD BALANCE YOUR ARCHITECTURE

SCREENSHOTS

Created AMI for Auto Scaling



Created Load Balancer

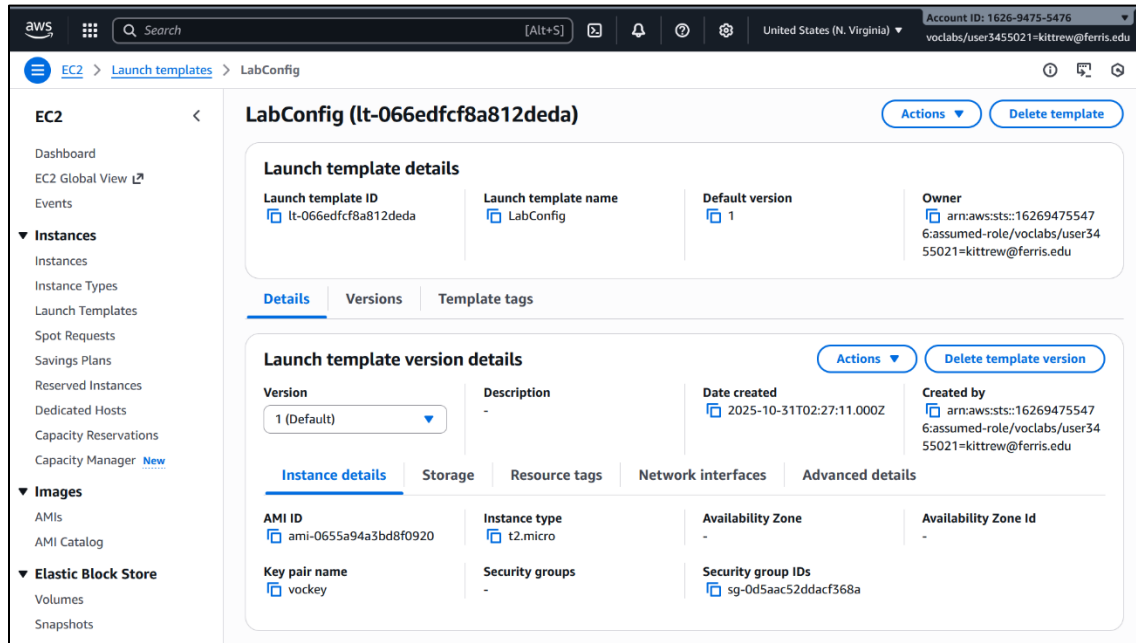
The screenshot displays the AWS Management Console interface for a newly created Elastic Load Balancing (ELB) instance. At the top, a green notification banner states: "Successfully created load balancer: LabELB. It might take a few minutes for your load balancer to fully set up and route traffic. Targets will also take a few minutes to complete the registration process and pass initial health checks." The main content area is titled "LabELB" and includes a "Details" section with the following information:

- Load balancer type:** Application
- Status:** Provisioning
- VPC:** vpc-0596c79e8cbac923b
- Load balancer IP address type:** IPv4
- Scheme:** Internet-facing
- Hosted zone:** Z35XDOTRQ7X7K
- Availability Zones:** subnet-034d3b55f99339b5e (us-east-1b use1-az4), subnet-001d67611d1c07836 (us-east-1a use1-az2)
- Date created:** October 30, 2025, 22:25 (UTC-04:00)
- Load balancer ARN:** arn:aws:elasticloadbalancing:us-east-1:162694755476:loadbalancer/app/LabELB/dad1f363bcd385ce
- DNS name:** LabELB-1863551978.us-east-1.elb.amazonaws.com (A Record)

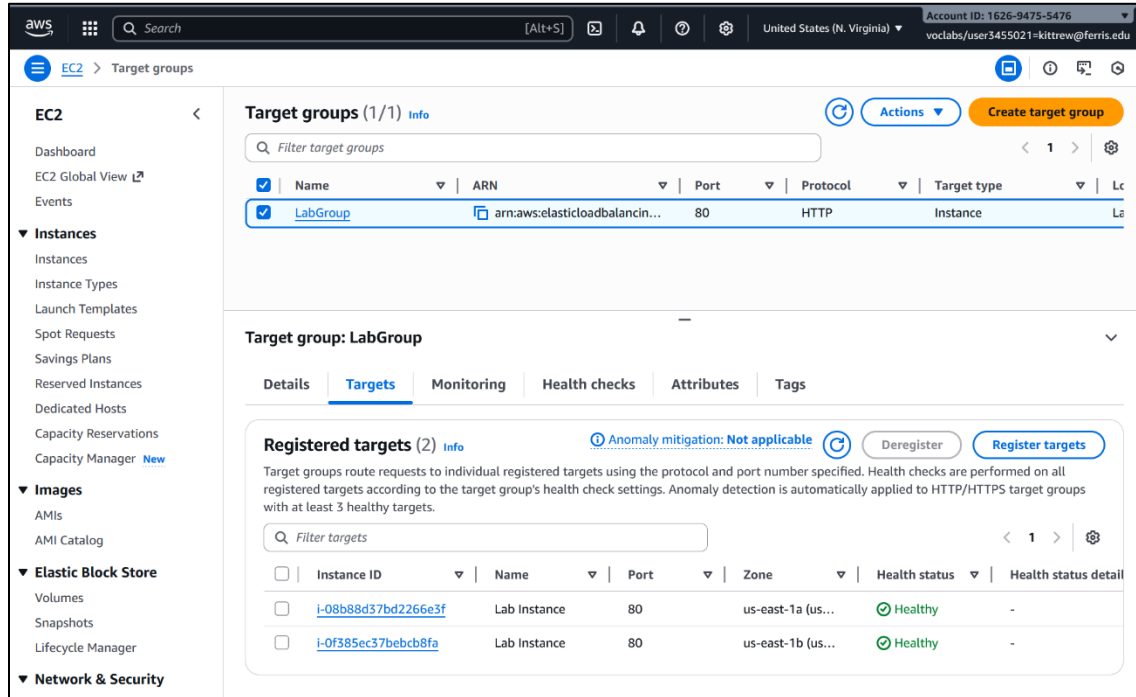
Below the details, the "Listeners and rules" tab is active, showing one listener configuration:

Protocol:Port	Default action	Rules	ARN	Security policy
HTTP:80	Forward to target group LabGroup (1 (100%)) Target group stickiness: Off	1 rule	ARN	Not applicable

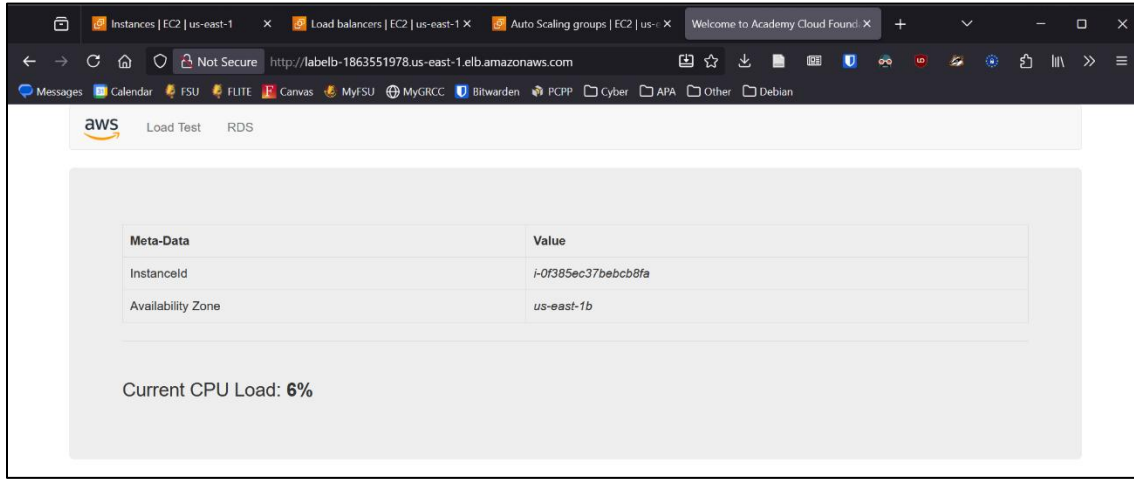
Created Launch Template



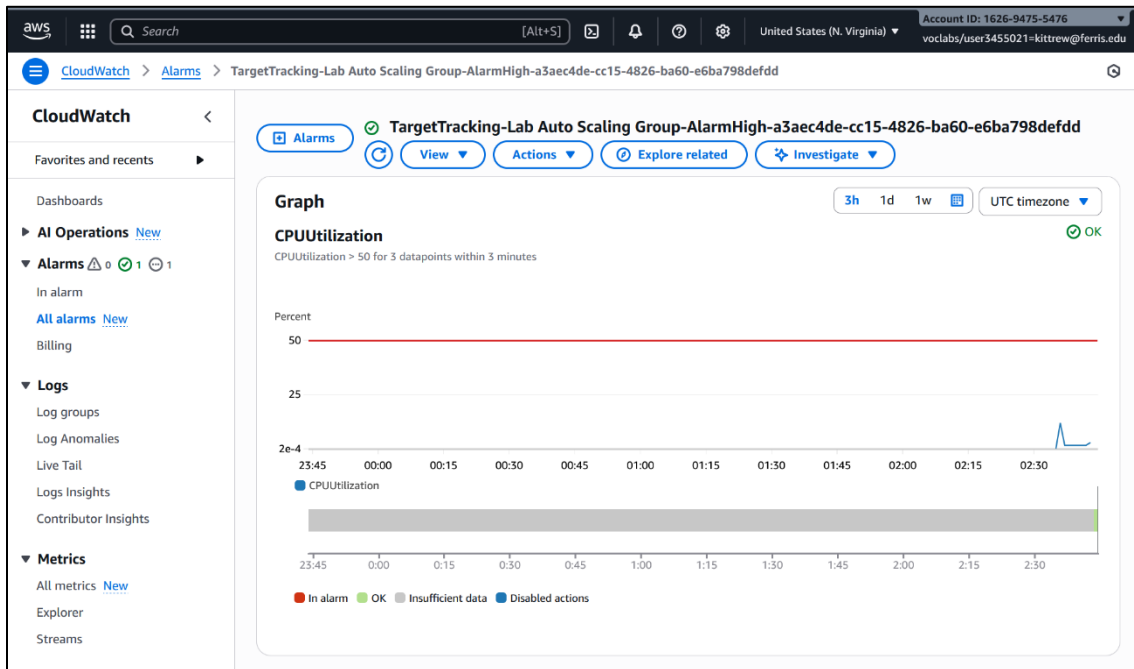
Verified Health of Auto Scaling Instances



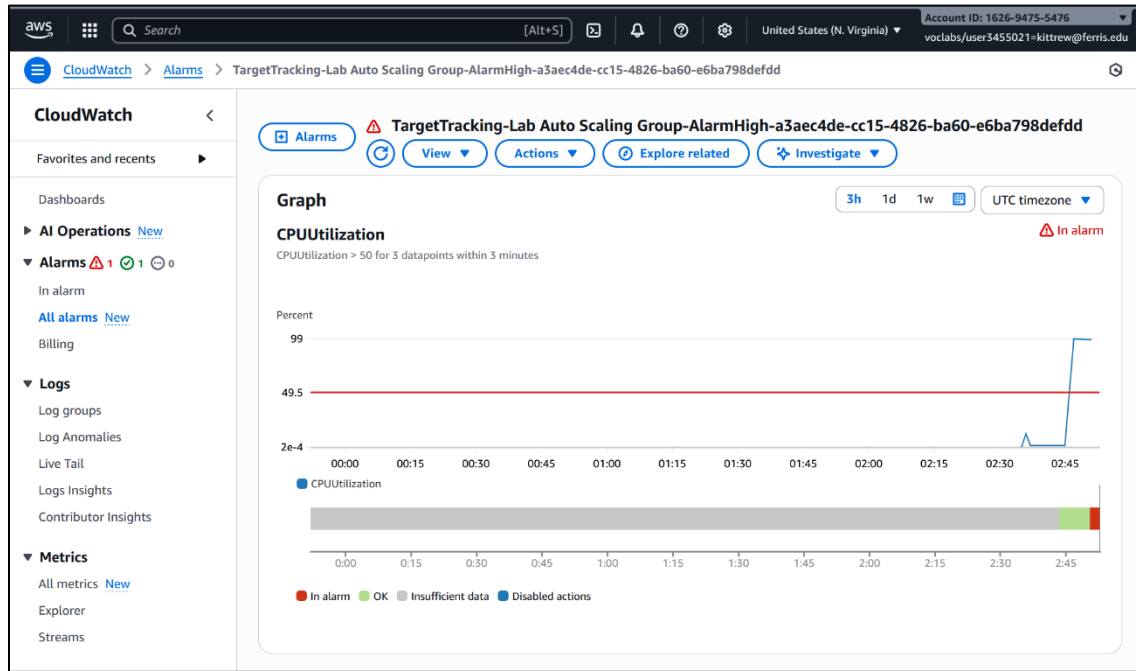
Reached Application



Checked Alarm Status (OK)



Checked Alarm Status (In Alarm)



Observed Automatic Creation of Additional Instances

The screenshot shows the AWS EC2 console "Instances" page. There are 6 instances listed, all in a "Running" state. The table below summarizes the instance details:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status
Lab Instance	i-0aad3f367e0a9419	Running	t2.micro	2/2 checks passed	View alarms +
Bastion Host	i-03aef0e8a91a3000e	Running	t2.micro	2/2 checks passed	View alarms +
Lab Instance	i-0f385ec37bebcb8fa	Running	t2.micro	2/2 checks passed	View alarms +
Lab Instance	i-018723f5b1a7d7be4	Running	t2.micro	2/2 checks passed	View alarms +
Lab Instance	i-08b88d37bd2266e3f	Running	t2.micro	2/2 checks passed	View alarms +
Web Server 1	i-0dc75627908c57cdc	Running	t2.micro	2/2 checks passed	View alarms +

SYNOPSIS

For this lab, we were tasked with using Elastic Load Balancing and Auto Scaling in order to automatically scale and load balance from one EC2 instance to multiple. The steps primarily involved creating a machine image template, creating and configuring a load balancer, launch template, and auto scaling group, monitoring performance, and observing automatic scaling after a CPU load test.

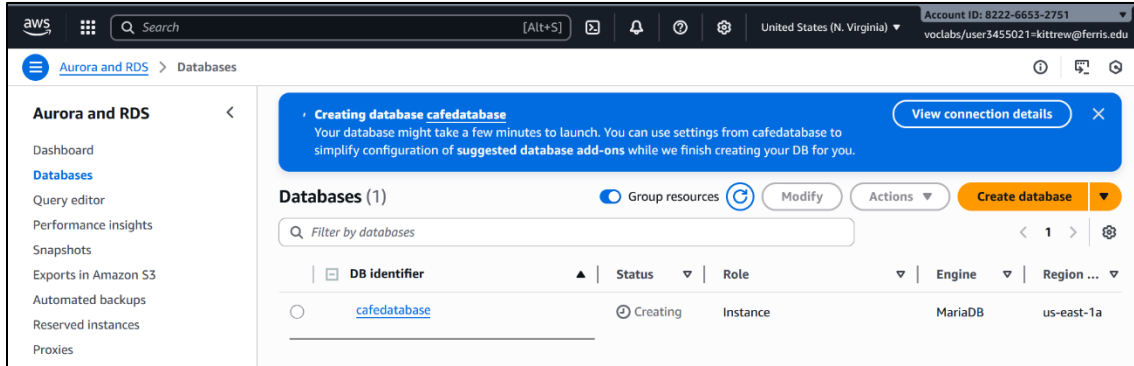
Unfortunately, I ran into an issue when following the launch template and auto scaling group instructions in task 3 where AWS failed to create the auto scaling policy after I finished the setup wizard – it claimed that I had insufficient permissions for some reason. This caused the expected alarms to not appear within CloudWatch, so I had to re-create the auto scaling policy in order to get this portion of the steps to work. I was able to manually edit the auto scaling group so that I could create another policy without needing to delete the entire group and start over. I suspect that this configuration might have been *slightly* wrong due to how long it took CloudWatch to pick up some of the metrics, but it ended up working as intended after a few minutes of waiting for it to respond to my activity (e.g., initiating the CPU load test).

In the absence of any strange permissions issues, I feel confident in my ability to create simple load balancing configurations in AWS with these tools after completing the lab. There are more configurations and setup wizards to complete than I had initially expected, but I found that these were relatively easy to follow once I read the instructions and understood what the purpose of each change that we made was.

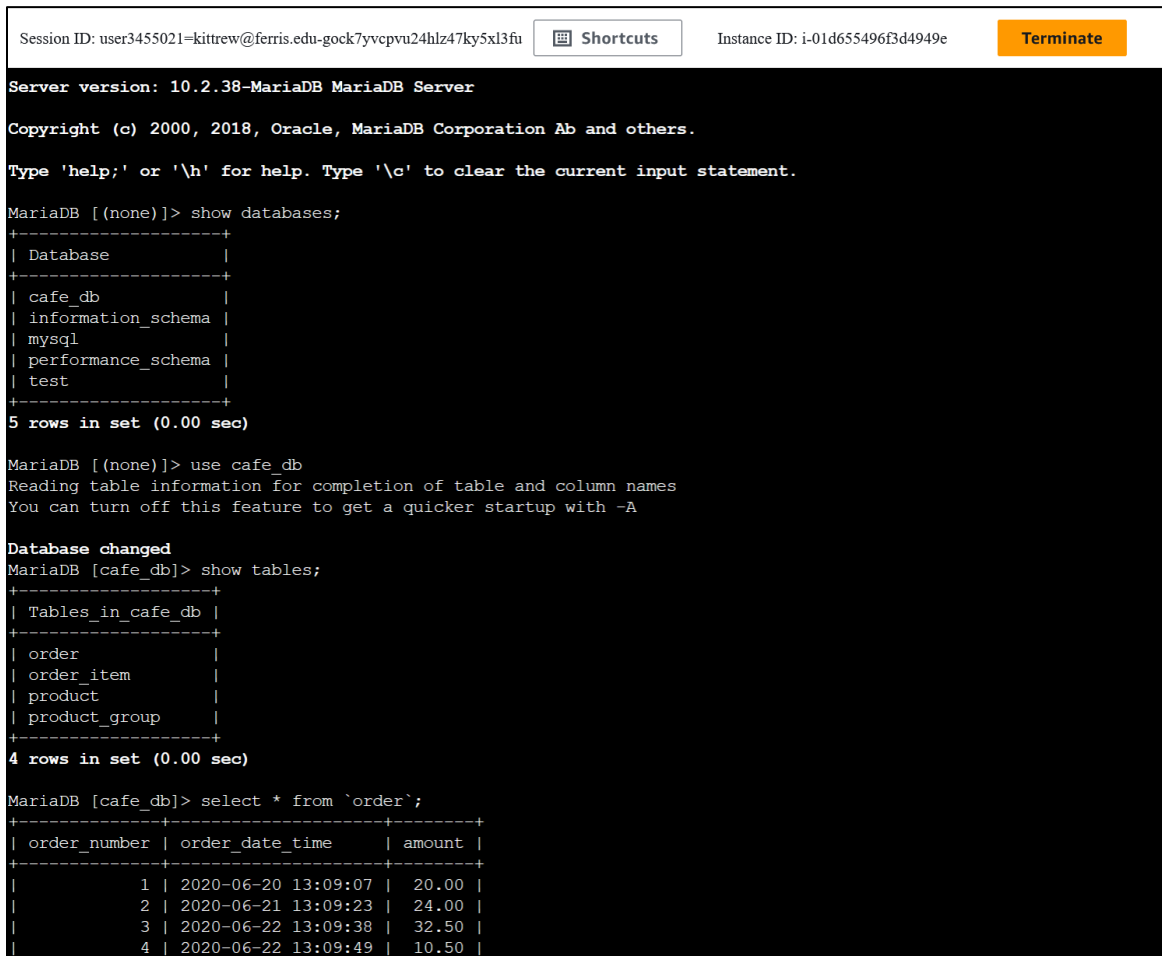
CHALLENGE LAB 6: MIGRATING A DATABASE

SCREENSHOTS

Created Database



Checked Database Content



Dumped Database Content

```

Session ID: user3455021=kittrew@ferris.edu-gock7yvcpvu24hlz47ky5xl3fu Shortcuts Instance ID: i-01d655496f3d4949e Terminate

[ec2-user@cafeserver ~]$ mysqldump --databases cafe_db -u root -p > CafeDbDump.sql
Enter password:
[ec2-user@cafeserver ~]$ ls
CafeDbDump.sql
[ec2-user@cafeserver ~]$
    
```

Imported Database Content to RDS

```

Session ID: user3455021=kittrew@ferris.edu-gock7yvcpvu24hlz47ky5xl3fu Shortcuts Instance ID: i-01d655496f3d4949e Terminate

[ec2-user@cafeserver ~]$ mysql -u admin -p --host cafedatabase.cz3drz8rna5u.us-east-1.rds.amazonaws.com
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 45
Server version: 11.4.5-MariaDB-log managed by https://aws.amazon.com/rds/

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> show databases;
+-----+
| Database |
+-----+
| cafe_db |
| information_schema |
| innodb |
| mysql |
| performance_schema |
| sys |
+-----+
6 rows in set (0.00 sec)

MariaDB [(none)]> use cafe_db;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [cafe_db]> show tables;
+-----+
| Tables_in_cafe_db |
+-----+
| order |
| order_item |
| product |
| product_group |
+-----+
4 rows in set (0.00 sec)

MariaDB [cafe_db]> select * from `order`;
+-----+-----+-----+
| order_number | order_date_time | amount |
+-----+-----+-----+
| 1 | 2020-06-20 13:09:07 | 20.00 |
| 2 | 2020-06-21 13:09:23 | 24.00 |
| 3 | 2020-06-22 13:09:38 | 32.50 |
| 4 | 2020-06-22 13:09:49 | 10.50 |
| 5 | 2020-06-24 13:10:02 | 42.00 |
| 6 | 2020-06-25 13:10:20 | 14.00 |
| 7 | 2020-06-25 13:10:30 | 35.00 |
| 8 | 2020-06-27 13:10:36 | 6.00 |
| 9 | 2020-06-28 13:10:46 | 18.00 |
| 10 | 2020-07-02 13:10:58 | 18.00 |
| 11 | 2020-07-03 13:11:08 | 19.50 |
| 12 | 2020-07-04 13:11:17 | 9.00 |
+-----+-----+-----+
    
```

Stopped Database on EC2 Instance

```
Session ID: user3455021=kittrew@ferris.edu-gock7yvcpvu24hlz47ky5xl3fu Shortcuts Instance ID: i-01d655496f3d4949e Terminate

[ec2-user@cafeserver ~]$ sudo service mariadb status
Redirecting to /bin/systemctl status mariadb.service
● mariadb.service - MariaDB 10.2 database server
   Loaded: loaded (/usr/lib/systemd/system/mariadb.service; enabled; vendor preset: disabled)
   Drop-In: /usr/lib/systemd/system/mariadb.service.d
            └─tokudb.conf
   Active: inactive (dead) since Tue 2025-10-21 18:24:35 UTC; 9s ago
   Process: 3717 ExecStartPost=/usr/libexec/mysql-check-upgrade (code=exited, status=0/SUCCESS)
   Process: 3646 ExecStart=/usr/libexec/mysqld --basedir=/usr $MYSQLD_OPTS $WSREP_NEW_CLUSTER (code=exited, status=0/SUCCESS)
   Process: 3513 ExecStartPre=/usr/libexec/mysql-prepare-db-dir %n (code=exited, status=0/SUCCESS)
   Process: 3489 ExecStartPre=/usr/libexec/mysql-check-socket (code=exited, status=0/SUCCESS)
   Main PID: 3646 (code=exited, status=0/SUCCESS)
   Status: "MariaDB server is down"

Oct 21 17:33:32 ip-10-0-0-134.ec2.internal mysql-check-upgrade[3717]: ERROR: ld.so: object '/usr/lib64/libjemalloc.so.1' from LD_PRELOAD ...ored.
Oct 21 17:33:32 ip-10-0-0-134.ec2.internal mysql-check-upgrade[3717]: ERROR: ld.so: object '/usr/lib64/libjemalloc.so.1' from LD_PRELOAD ...ored.
Oct 21 17:33:32 ip-10-0-0-134.ec2.internal mysql-check-upgrade[3717]: ERROR: ld.so: object '/usr/lib64/libjemalloc.so.1' from LD_PRELOAD ...ored.
Oct 21 17:33:32 ip-10-0-0-134.ec2.internal mysql-check-upgrade[3717]: ERROR: ld.so: object '/usr/lib64/libjemalloc.so.1' from LD_PRELOAD ...ored.
Oct 21 17:33:32 ip-10-0-0-134.ec2.internal mysql-check-upgrade[3717]: ERROR: ld.so: object '/usr/lib64/libjemalloc.so.1' from LD_PRELOAD ...ored.
Oct 21 17:33:32 ip-10-0-0-134.ec2.internal mysql-check-upgrade[3717]: ERROR: ld.so: object '/usr/lib64/libjemalloc.so.1' from LD_PRELOAD ...ored.
Oct 21 17:33:32 ip-10-0-0-134.ec2.internal mysql-check-upgrade[3717]: ERROR: ld.so: object '/usr/lib64/libjemalloc.so.1' from LD_PRELOAD ...ored.
Oct 21 17:33:32 ip-10-0-0-134.ec2.internal systemd[1]: Started MariaDB 10.2 database server.
Oct 21 18:24:34 cafeserver systemd[1]: Stopping MariaDB 10.2 database server...
Oct 21 18:24:35 cafeserver systemd[1]: Stopped MariaDB 10.2 database server.
Hint: Some lines were ellipsized, use -l to show in full.
[ec2-user@cafeserver ~]$
```

Tested Functionality

The screenshot shows a web browser window with the URL `http://3.218.164.144/cafes/orderHistory.php`. The page displays the Café application with a navigation menu (Home, Menu, Order History) and an Order History section. The history shows three orders:

Order Number	Date	Time	Total Amount
26	2025-10-21	18:29:39	\$2.00
25	2025-10-21	18:00:47	\$1.50
24	2020-07-28	13:14:07	\$35.00

Item	Price	Quantity	Amount
Donut	\$1.00	2	\$2.00

Item	Price	Quantity	Amount
Croissant	\$1.50	1	\$1.50

Item	Price	Quantity	Amount
Strawberry Blueberry Tart	\$3.50	4	\$14.00
Strawberry Tart	\$3.50	3	\$10.50
Latte	\$3.50	3	\$10.50

Order number 25 is from the test at the beginning of the lab before the migration, while order number 26 is from the test at the end of the lab.

SYNOPSIS

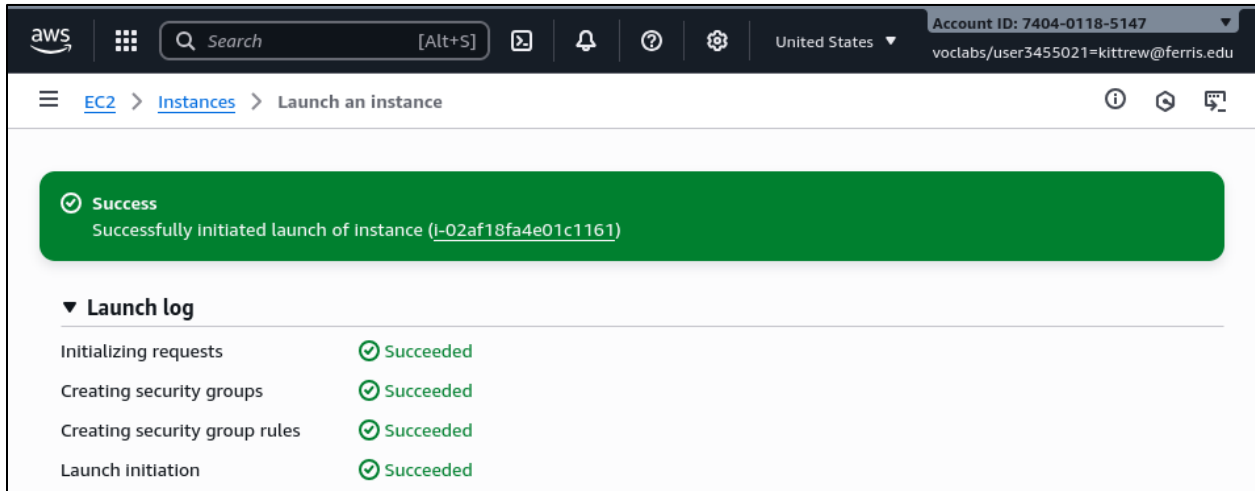
For this challenge lab, we were tasked with migrating from a database hosted on an EC2 instance to Amazon RDS, the service for relational databases in the cloud. The main challenge was in understanding how the initial database configuration worked for the café application, and then knowing which additional steps (i.e., ones not given in the instructions) we would need to complete to switch to RDS and make the café application work with the new configuration.

Although I have some experience with databases, I was admittedly not confident in my abilities at the start of this lab. The database security class was one of the first classes that I completed in the ISI program, and I had not worked with any database technology since then (apart from brief command line interactions like when we show the tables in this lab). Fortunately, I didn't find this to be a problem because most of the technical details were abstracted away with the database creation wizard. I'm becoming better at developing a sense for which configuration options and services might need to be investigated when I encounter problems in AWS, so I didn't find it difficult to get the café application working once I finished the database migration. There were a few minutes at the end of the lab where the menu page initially didn't work, but I quickly discovered that this was because I had forgotten to update the database username and password values in the AWS secrets manager. As soon as I updated these values, the page began working and I was able to successfully submit an order and view it in the history.

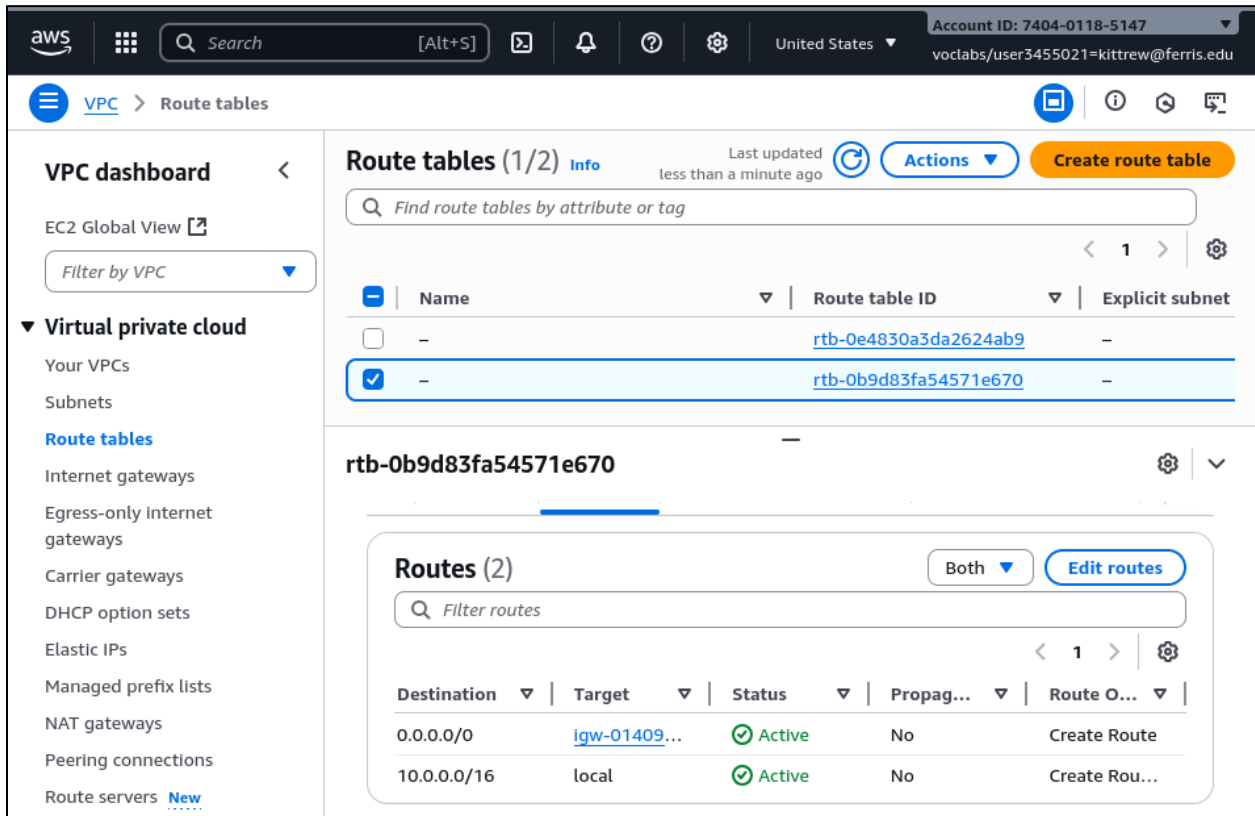
CHALLENGE LAB 7: VPC NETWORKING ENVIRONMENT

SCREENSHOTS

Launched Bastion Host



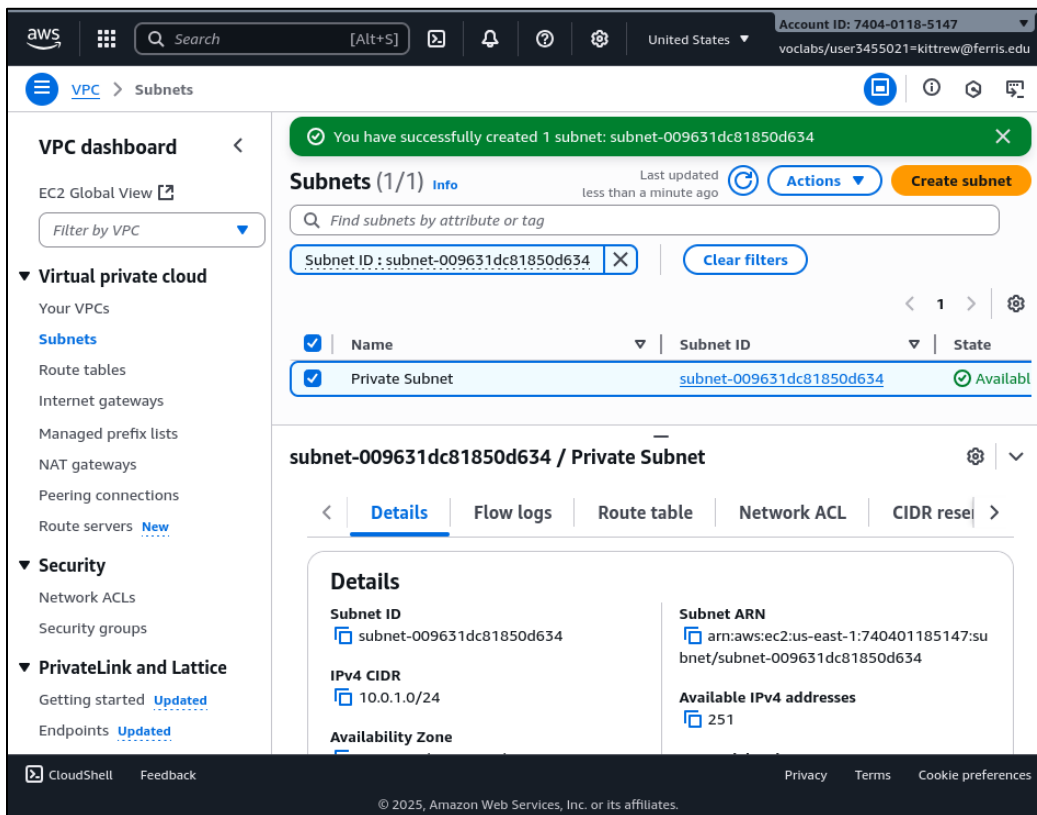
Updated Route Table with Internet Gateway



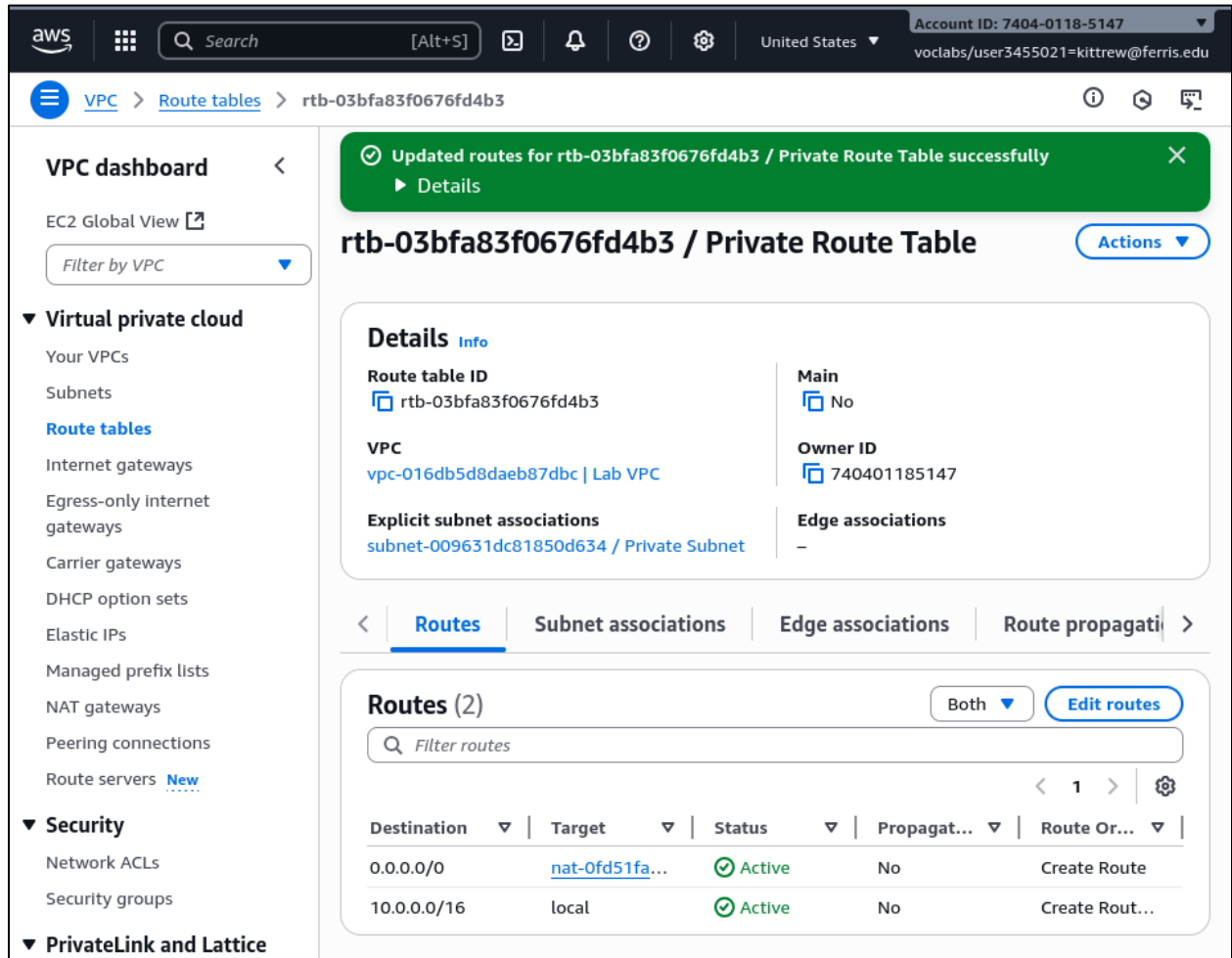
Successfully Connected to Bastion Host

```
(ssh) ~/
:will@abscissa:~/ $ ssh -i Downloads/labsuser.pem ec2-user@ec2-3-238-18-213.compute-1.amazonaws.com
#_
~\_ #####_ Amazon Linux 2023
~\~\_#####\_
~\~\_###|
~\~\_#/ --- https://aws.amazon.com/linux/amazon-linux-2023
~\~\_V~' '->
~\~\_ - - -
~\~\_ - / - - -
~\~\_ - /m/ '
[ec2-user@ip-10-0-0-203 ~]$
```

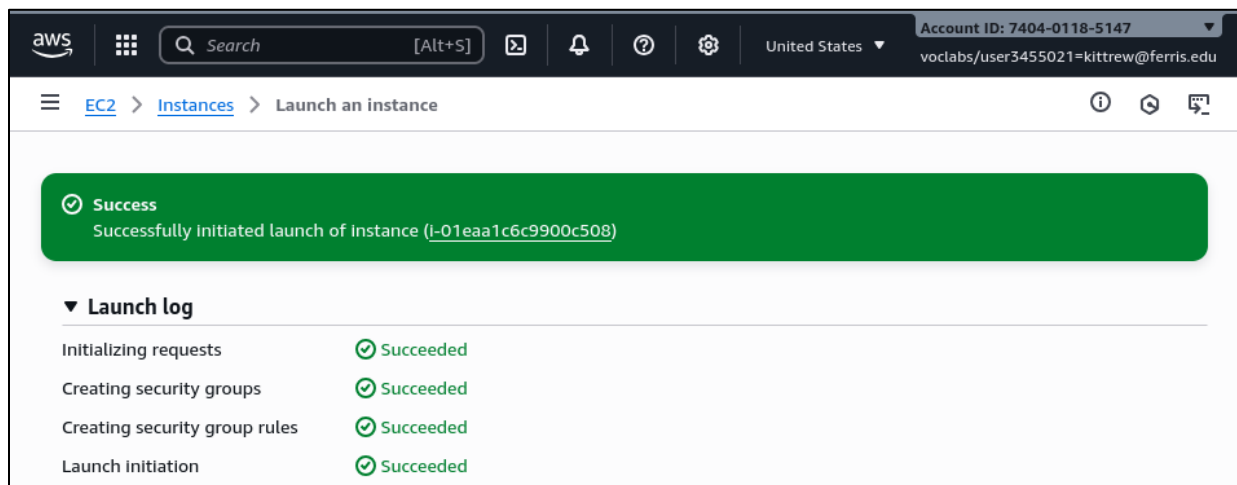
Created Private Subnet



Updated Private Subnet Route Table with NAT Gateway



Launched EC2 Instance in Private Subnet



Connected to Host in Private Subnet via SSH Passthrough

```
(ssh) ~/
:will@abscissa:~/ $ ssh -A ec2-user@ec2-3-238-18-213.compute-1.amazonaws.com
#_
~\_ #####_ Amazon Linux 2023
~~ \_#####\
~~ \###|
~~ \#/ --- https://aws.amazon.com/linux/amazon-linux-2023
~~ V~' '->
~~~~
~~~.~.~
-/_ -/_
-/_m/'

Last login: Tue Sep 30 12:01:55 2025 from 76.112.150.225
[ec2-user@ip-10-0-0-203 ~]$ ssh ec2-user@10.0.1.219
The authenticity of host '10.0.1.219 (10.0.1.219)' can't be established.
ED25519 key fingerprint is SHA256:1UmeY0Fu1N3sHXie+E0vbDyA1lZ0rRzMUNZUgeh26lY.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.1.219' (ED25519) to the list of known hosts.
#_
~\_ #####_ Amazon Linux 2023
~~ \_#####\
~~ \###|
~~ \#/ --- https://aws.amazon.com/linux/amazon-linux-2023
~~ V~' '->
~~~~
~~~.~.~
-/_ -/_
-/_m/'

[ec2-user@ip-10-0-1-219 ~]$
```

Tested Connection to Internet from Private Subnet Host

```
(ssh) ~/
[ec2-user@ip-10-0-1-219 ~]$ ping 8.8.8.8 -c 4
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data:
64 bytes from 8.8.8.8: icmp_seq=1 ttl=112 time=1.57 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=112 time=1.71 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=112 time=1.66 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=112 time=1.11 ms

--- 8.8.8.8 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 1.106/1.511/1.708/0.239 ms
[ec2-user@ip-10-0-1-219 ~]$
```

SYNOPSIS

During this challenge lab, we were tasked with applying our knowledge of the EC2 and VPC services within AWS to create a virtual environment that meets the specifications we were given. Some portions of the lab were guided, while others did not offer complete instructions and required additional research or experimentation.

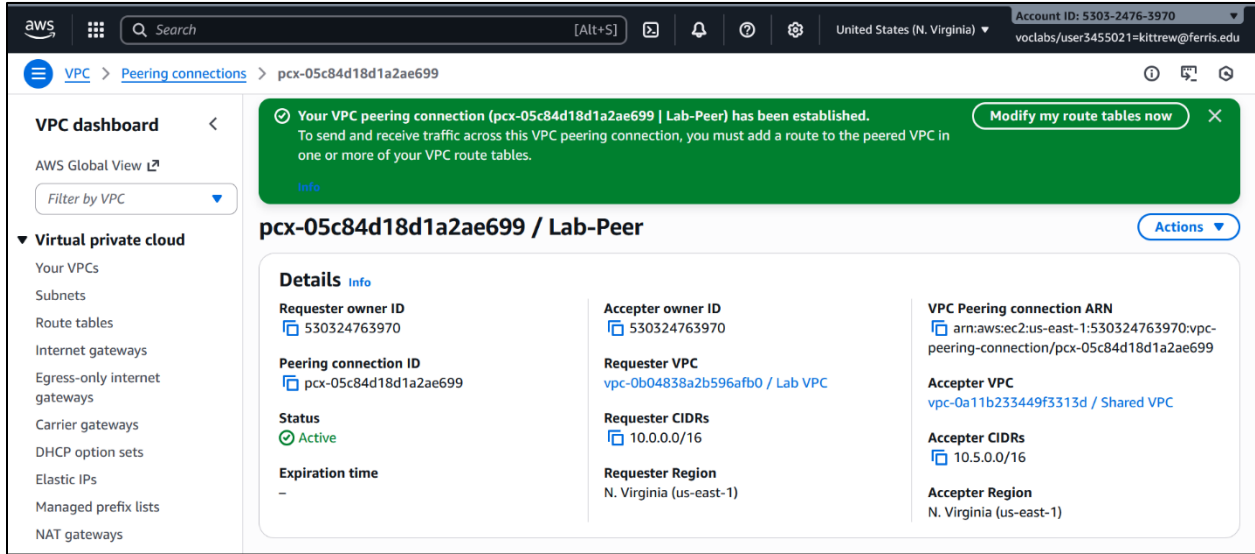
I ran into a few issues, but I think that the lab went well for the most part. Before I was able to use any of the `ssh-add` commands that were in the instructions, I had to run `eval `ssh-agent -s`` first – which would then allow the `ssh-add` command(s) to complete without any error. I also discovered that the `-K` flag the instructions told me to use was incorrect for my version of Linux (Debian 13). After making the SSH connection to a host, I had a minor problem where my `TERM` was unrecognized, but this could be solved quickly with `export TERM=xterm`. At the end of the lab, I was able to create the network ACL to deny ICMP traffic between the private subnet and the EC2 instance but was unable to get Amazon to apply the settings. I'm not entirely sure what was happening since I think that my method was correct based on their documentation, but I kept receiving permissions errors followed by network connection errors to the AWS webpage itself (maybe something went wrong with my connection?). Had the settings been applied, I think they would have worked correctly.

Overall, I feel more confident in my ability to leverage EC2 and VPC services to create environments as needed without requiring specific instructions. It was nice to get more hands-on practice with using the services, and it was validating to find that I could usually predict roughly what steps were going to come next in the instructions even if I didn't guess every detail correctly.

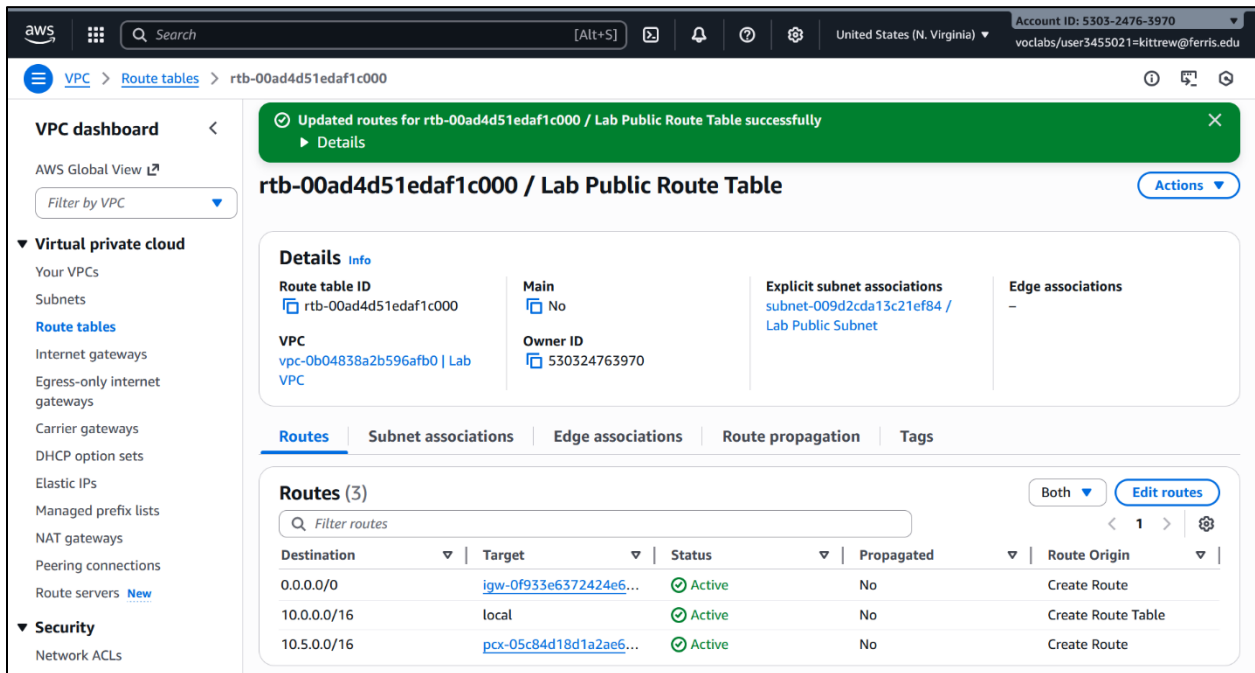
LAB 8: CREATE A VPC PEERING CONNECTION

SCREENSHOTS

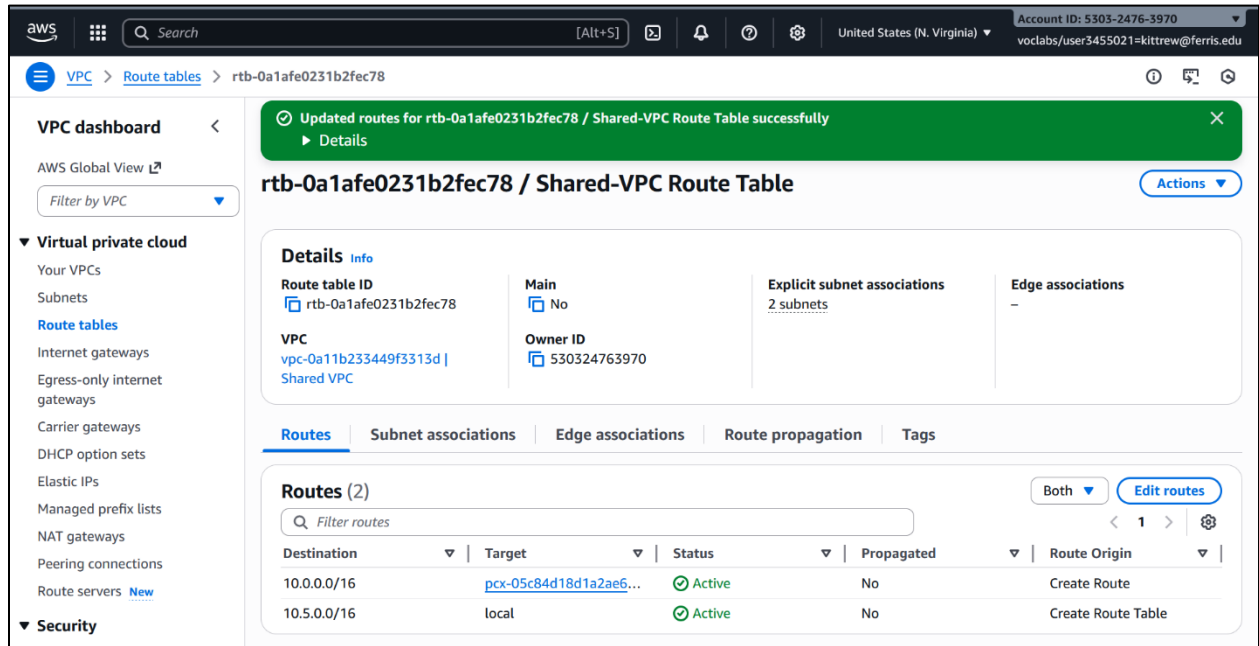
Created and Accepted VPC Peering Connection



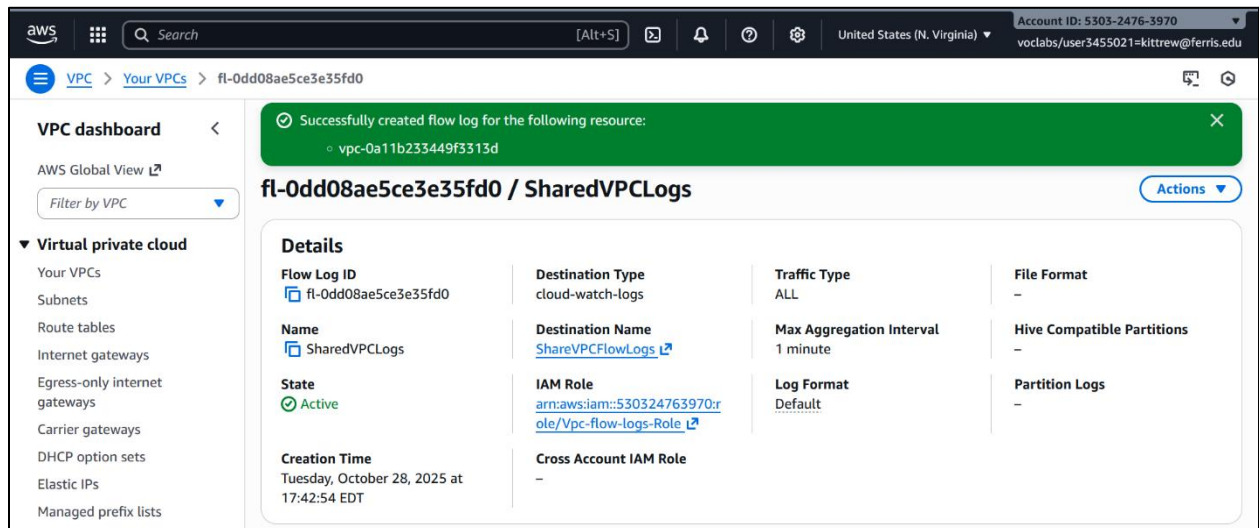
Updated Lab Public Route Table



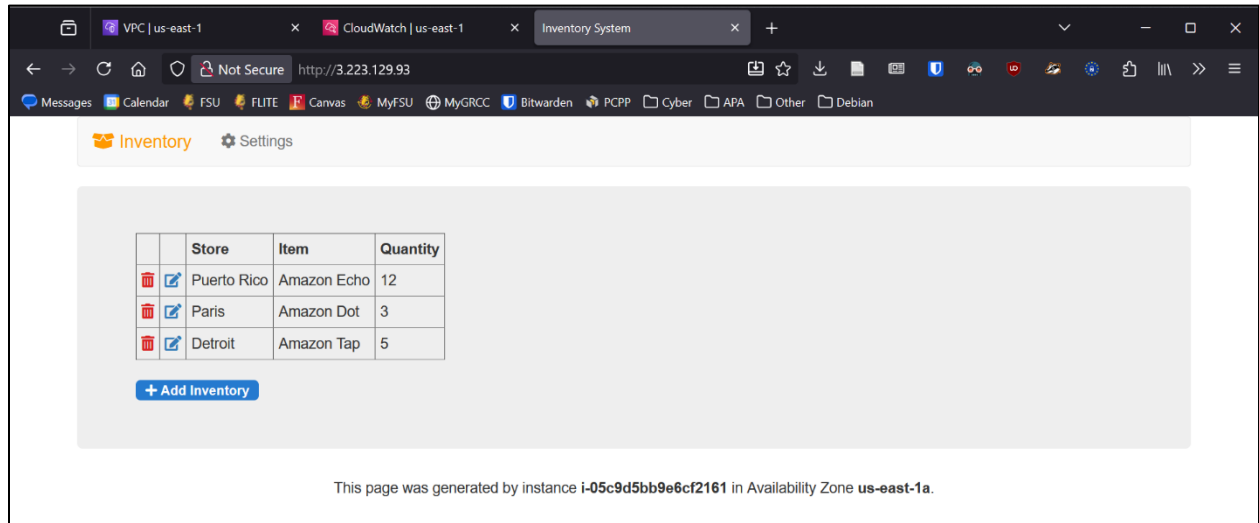
Updated Shared-VPC Route Table



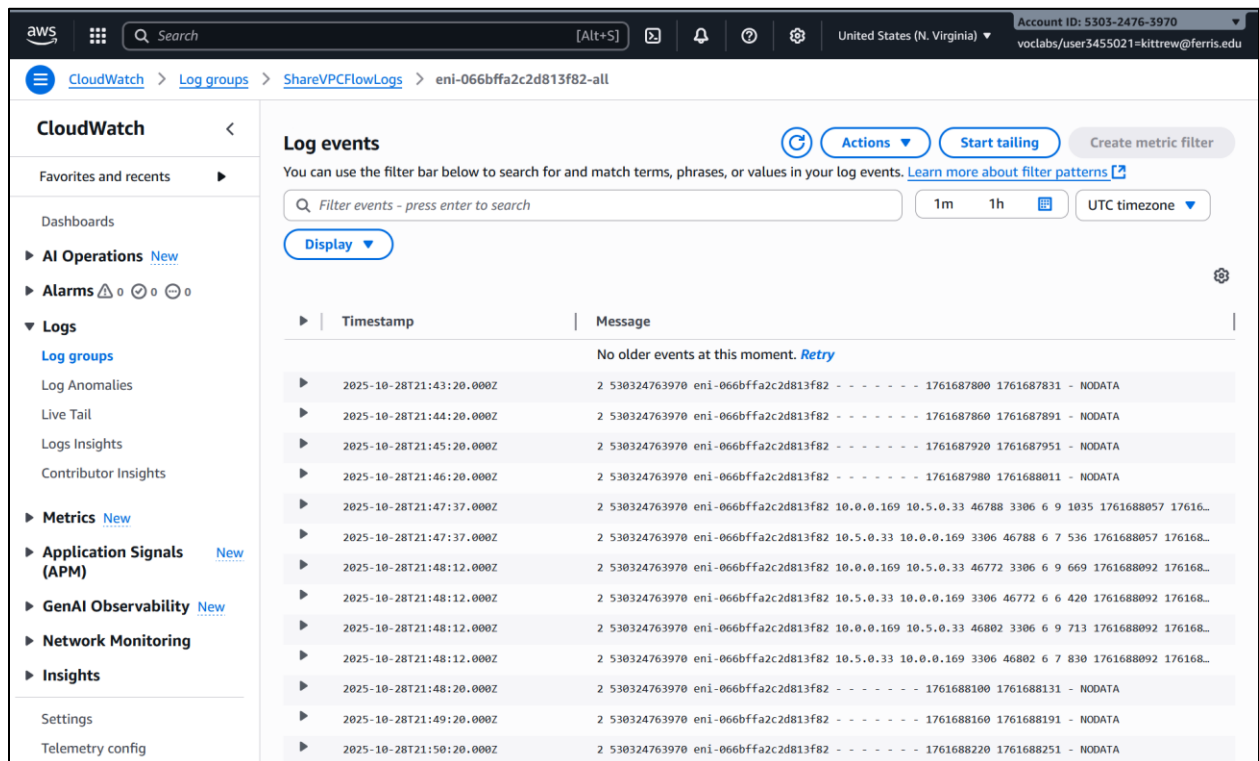
Created VPC Flow Log



Tested VPC Peering Connection



Viewed Flow Logs



SYNOPSIS

In this lab, we created a VPC peering connection between two pre-existing virtual private clouds in our environment. In addition to creating the connection itself, we configured route tables to ensure that traffic is delivered properly between VPCs and enabled logging so that we could have insight into data on the network. At the end of the lab, we tested the configuration and checked the logs to ensure that everything had functioned correctly.

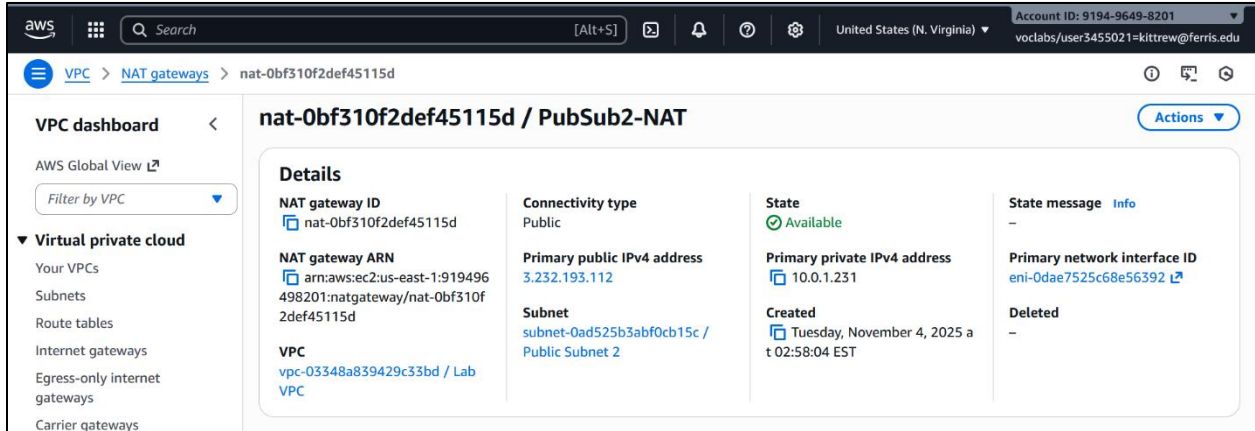
Despite the lab being guided, I don't feel confident in my ability to create configurations that use VPC peering connections in an effective way. I didn't have trouble following any of the steps, but I also don't think I understand what the correct use case/application of VPC peering is. Of course, I know that this feature allows me to connect two VPCs (even between different accounts and regions), but I don't understand why this configuration was beneficial to the environment that we created in our lab – as both VPCs belonged to the same account in the same region. Perhaps I'm reading into it too much and it was only for demonstration purposes, but my understanding is that we could have put the database in a private subnet within a single VPC and then used security controls to achieve effectively the same result.

Otherwise, I feel like I have the foundations to use VPC peering effectively once I develop a better understanding of when it should be used and why it might be desirable over a different configuration.

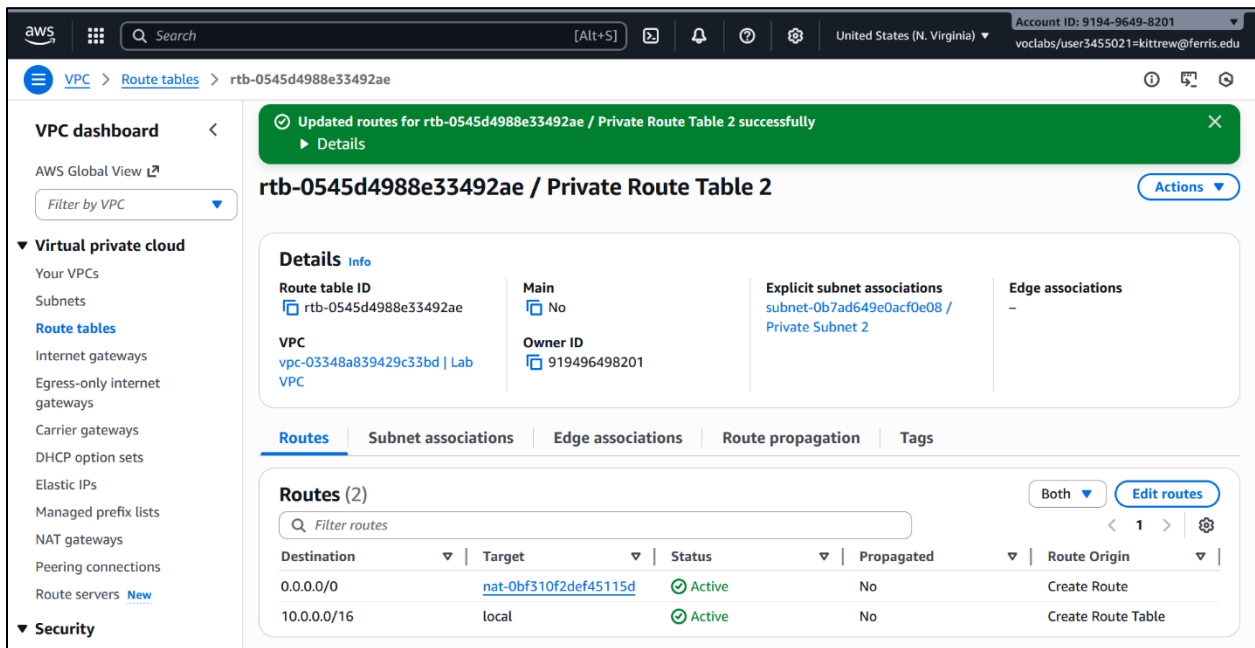
CHALLENGE LAB 10: SCALABLE/AVAILABLE CAFÉ

SCREENSHOTS

Created NAT Gateway



Modified Private Subnet 2 Route Table



Created Launch Template

The screenshot shows the AWS Management Console interface for a Launch Template. The breadcrumb navigation is EC2 > Launch templates > LabLaunchTemplate. The page title is 'LabLaunchTemplate (lt-083c1ac1647d35766)'. There are 'Actions' and 'Delete template' buttons. The 'Launch template details' section includes:

Launch template ID	Launch template name	Default version	Owner
lt-083c1ac1647d35766	LabLaunchTemplate	1	arn:aws:sts::919496498201:assumed-role/voclabs/user3455021=kittrew@ferris.edu

Below this is a 'Launch template version details' section for version 1 (Default), with a 'Date created' of 2025-11-04T08:05:20.000Z. There are 'Actions' and 'Delete template version' buttons. The 'Instance details' section includes:

AMI ID	Instance type	Availability Zone	Availability Zone Id
ami-07457e911436d3692	t2.micro	-	-
Key pair name	Security groups	Security group IDs	
LaunchTemplateKeypair	-	sg-05d781ef497fd413f	

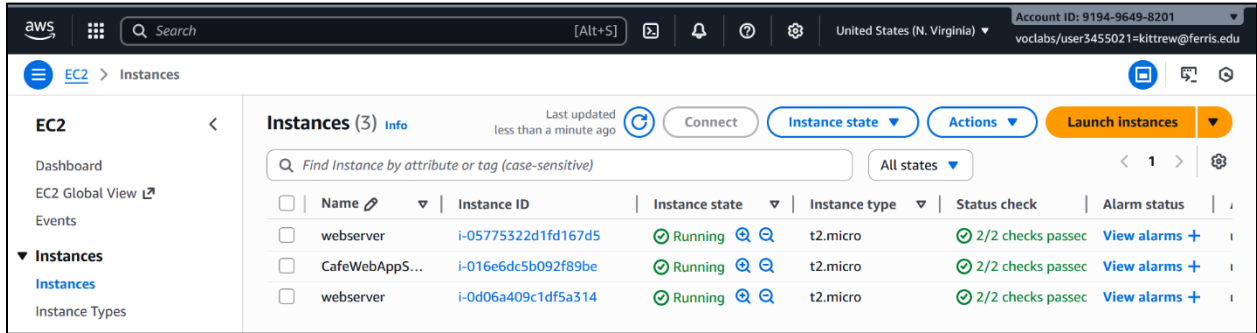
Created Auto Scaling Group

The screenshot shows the AWS Management Console interface for an Auto Scaling Group. The breadcrumb navigation is EC2 > Auto Scaling groups > LabAutoScalingGroup. The page title is 'LabAutoScalingGroup'. There is an 'Edit' button. The 'LabAutoScalingGroup Capacity overview' section includes:

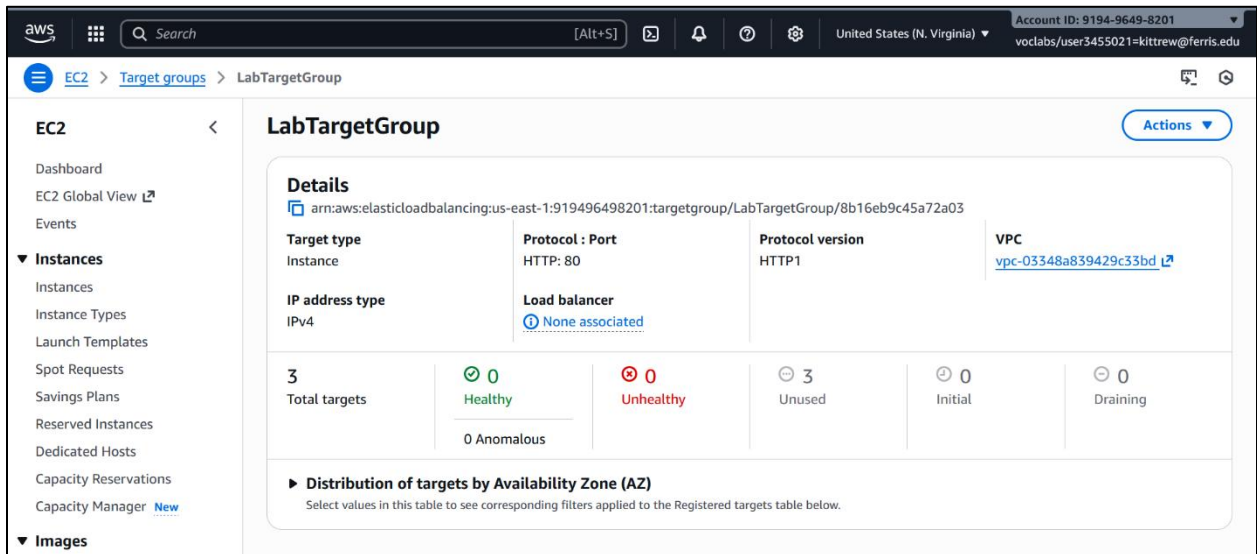
Desired capacity	Scaling limits (Min - Max)	Desired capacity type	Status
2	2 - 6	Units (number of instances)	Updating capacity

The 'Date created' is Tue Nov 04 2025 03:10:30 GMT-0500 (Eastern Standard Time).

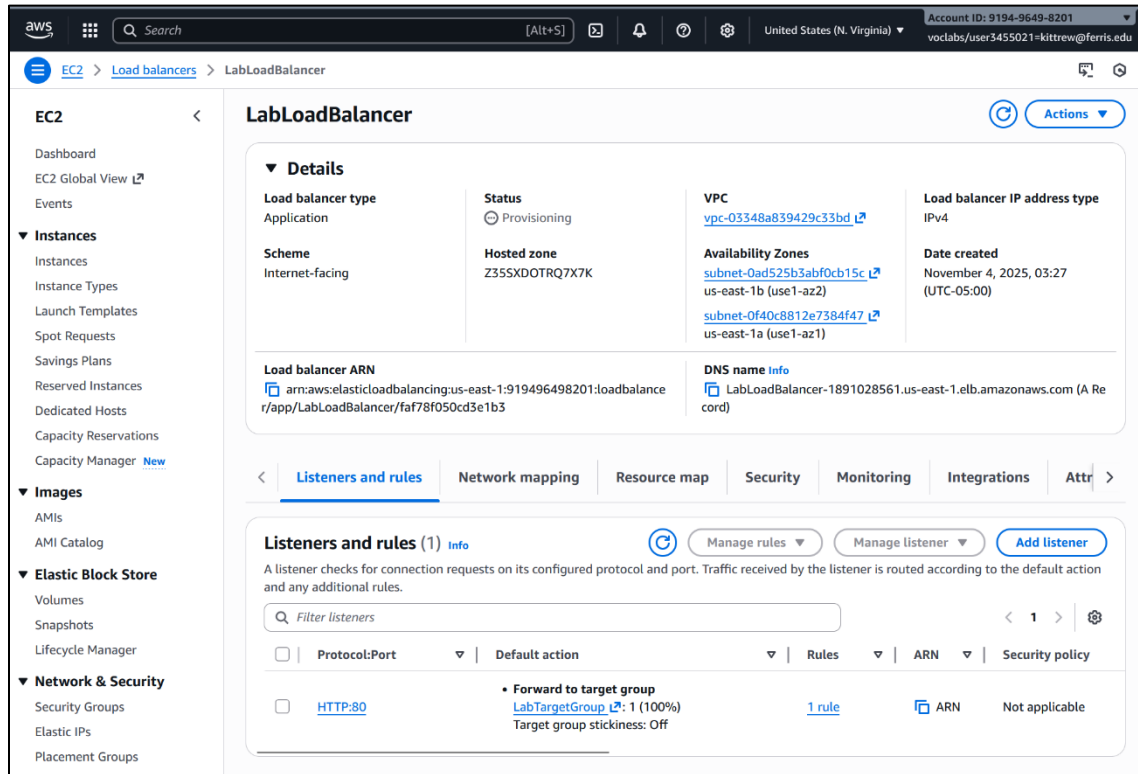
Observed Instance Creation



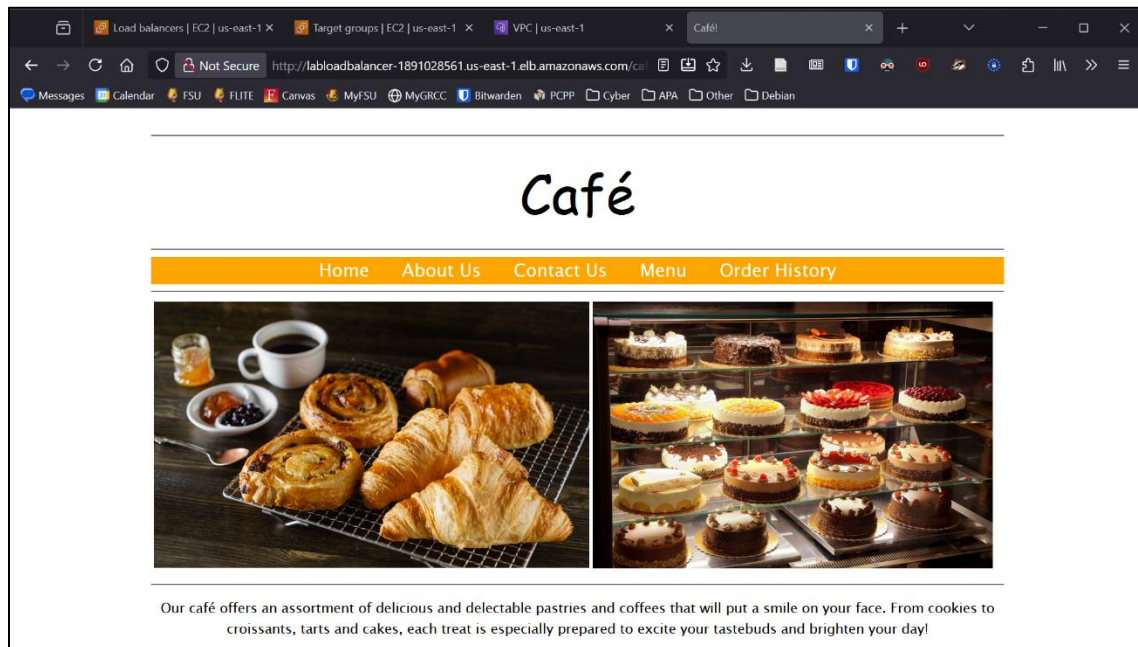
Created Target Group



Created Load Balancer



Reached Café Application



Started Stress Test

```
sh-4.2$ stress --cpu 1 --timeout 600
stress: info: [3444] dispatching hogs: 1 cpu, 0 io, 0 vm, 0 hdd
```

Observed Auto Scaling

The screenshot shows the AWS Management Console interface for the EC2 Instances page. The top navigation bar includes the AWS logo, a search bar, and the region 'United States (N. Virginia)'. The main content area displays 'Instances (7)' with a table listing the following data:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status
webserv	i-05775322d1fd167d5	Running	t2.micro	2/2 checks passed	View alarms
webserv	i-0952e660b69f5c317	Running	t2.micro	2/2 checks passed	View alarms
webserv	i-072b51601c234b8f2	Running	t2.micro	2/2 checks passed	View alarms
CafeWebAppS...	i-016e6dc5b092f89be	Running	t2.micro	2/2 checks passed	View alarms
webserv	i-0d06a409c1df5a314	Running	t2.micro	2/2 checks passed	View alarms
webserv	i-061223c326f9f0cd4	Running	t2.micro	2/2 checks passed	View alarms
webserv	i-001a158706b8b978f	Running	t2.micro	2/2 checks passed	View alarms

SYNOPSIS

In this challenge lab, we were tasked with using load balancing and auto scaling to create a scalable and highly available architecture for the café web application. Before we could begin configuring auto scaling, the existing network configuration had to be inspected and updated to facilitate the desired multi-AZ environment. Tasks related to automatic scaling setup were completed afterwards, and a stress test was initiated to confirm that automatic scaling worked.

Most of the lab went smoothly, but I experienced some minor issues towards the end. Like in my previous lab, AWS failed to make the scaling policy after I completed the setup wizard for the auto scaling group. This time, however, policy creation failed silently. I discovered the issue during the final lab task where we observe more EC2 instances being automatically deployed after noticing that nothing had changed about 5 minutes into the CPU stress test. I had to modify the auto scaling group to re-create the automatic scaling policy, which fixed this portion of the issue. The `stress --cpu 1 --timeout 600` command had also completed by this time, which caused the CloudWatch alarm to never be triggered (preventing any automatic scaling). Automatic scaling worked as expected after restarting the stress test and waiting for a few minutes.

I feel more confident that I can use scalability and availability features in AWS with less guidance after completing this lab and troubleshooting the issue that I experienced. With more practice, I would feel comfortable attempting to use scalability and availability features in combination with some of the others that we've covered (e.g., database services).