Project overview and objectives

In this project, you're challenged to use familiar AWS services, as well as AWS services that might be new to you, to create resources in AWS and to implement security on them. Throughout various AWS Academy courses, you have completed hands-on labs. You have used different AWS services and features to build a variety of solutions.

In many parts of the project, step-by-step guidance is *not* provided. This is intentional. These specific sections of the project are meant to challenge you to practice skills that you have acquired throughout your learning experiences prior to this project. In some cases, you might be challenged to use resources to independently learn new skills.

By the end of this project, you should be able to do the following:

- Secure access to objects in an Amazon Simple Storage Service (Amazon S3) bucket.
- Secure network access to your virtual network.
- Encrypt data at rest by using AWS Key Management Service (AWS KMS) on an Amazon Elastic Block Store (Amazon EBS) volume.
- Manage encryption keys by using AWS KMS.
- Create a monitoring and incident response system by using Amazon CloudWatch and AWS Config.

If you ever want to reset the lab environment to the original state, before you started the lab, use the **Reset** option above these instructions. Note that this won't reset your budget. **Warning:** If you reset the environment, you will *permanently delete* everything that you have created or stored in this AWS account.

AWS service restrictions

In this lab environment, access to AWS services and service actions are restricted to the ones that are needed to complete the lab instructions. You might encounter errors if you attempt to access other services or perform actions beyond the ones that are described in this lab.

Scenario

In this project, you work as a cloud security specialist at AnyCompany Financial bank.

The company has locations throughout the country. The company provides checking and savings accounts, credit cards, loans, and investment products. All products require the use of personally identifiable information (PII), such as account numbers, contact information, and personal IDs. Your manager, the Director of IT, has tasked you to ensure the security of the company's resources in the AWS Cloud.

Key areas of the AWS infrastructure that need to be secured include S3 buckets, the network that hosts web servers, and keys that are used to encrypt data.

Solution requirements

This project is split into different phases, and each phase is designed to address one or more solution requirements. The solution must meet the following requirements:

- R1 - Design (phase 2)
- R2 - Optimize cost (phase 2)
- R3 - Restrict access (phases 1, 2, and 3)
- R4 - Enforce compliance (phases 3 and 4)
- R5 - Encrypt data (phase 3)
- R6 - Withstand penetration testing (phase 2)
- R7 - Monitor and log user activity (phases 1 and 4)
- R8 - Generate alert and change management notifications (phase 4)

Approach

The following table provides an overview of the phases of this project.

Phase	Detail	Solution requirement(s)

1	Securing data in Amazon S3 - The solution you build will implement security features including bucket policies, object versioning, inventory logging, and object-level logging.	R3, R7
2	Securing VPCs - The solution you build will implement security features including VPC flow logs, route table configurations, network ACLs, and network firewalls.	R1, R2, R3, R6
3	Securing AWS resources by using AWS KMS - The solution you build will implement security features including using a KMS customer managed key to encrypt data stored in S3, to encrypt an EC2 instance EBS volume, to encrypt data using KMS envelope encryption, and to encrypt data stored in Secrets Manager.	R3, R5
4	Monitoring and logging - The solution you build will implement security features available in CloudTrail, CloudWatch, and AWS Config to log and monitor activity in your account.	R4, R7, R8

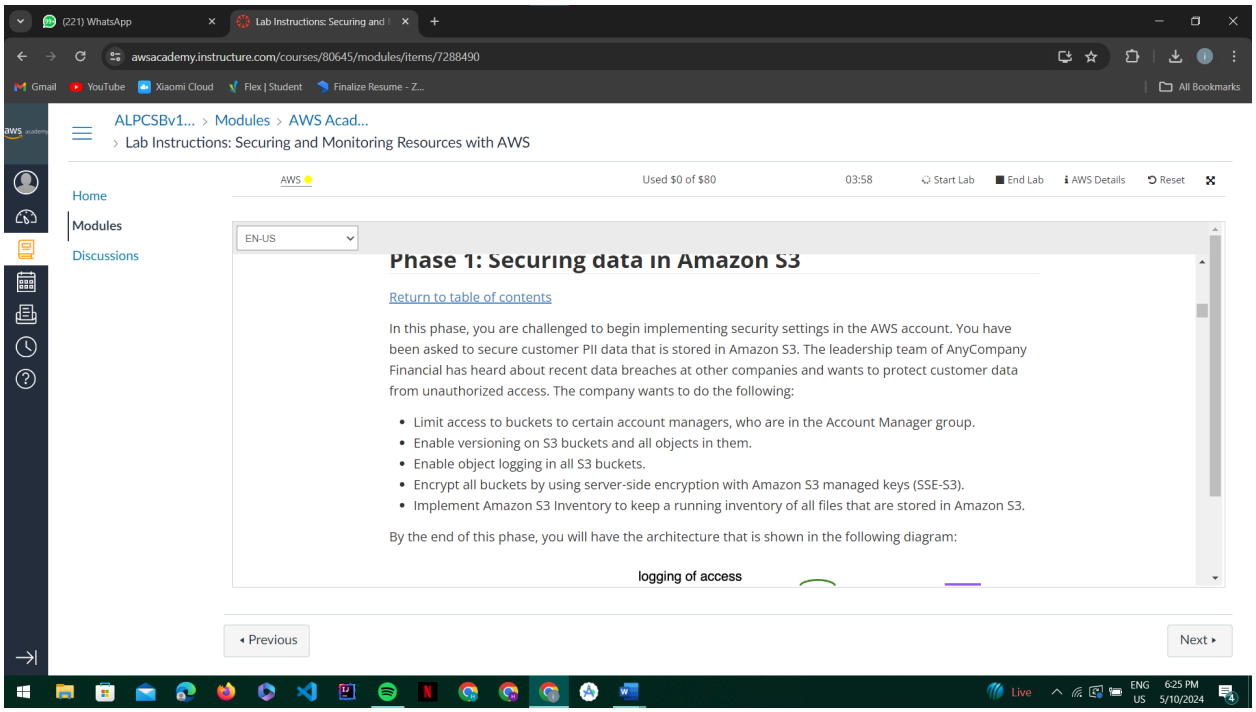
Phase 1: Securing data in Amazon S3

Return to table of contents

In this phase, you are challenged to begin implementing security settings in the AWS account. You have been asked to secure customer PII data that is stored in Amazon S3. The leadership team of AnyCompany Financial has heard about recent data breaches at other companies and wants to protect customer data from unauthorized access. The company wants to do the following:

- Limit access to buckets to certain account managers, who are in the Account Manager group.
- Enable versioning on S3 buckets and all objects in them.
- Enable object logging in all S3 buckets.
- Encrypt all buckets by using server-side encryption with Amazon S3 managed keys (SSE-S3).
- Implement Amazon S3 Inventory to keep a running inventory of all files that are stored in Amazon S3.

By the end of this phase, you will have the architecture that is shown in the following diagram:

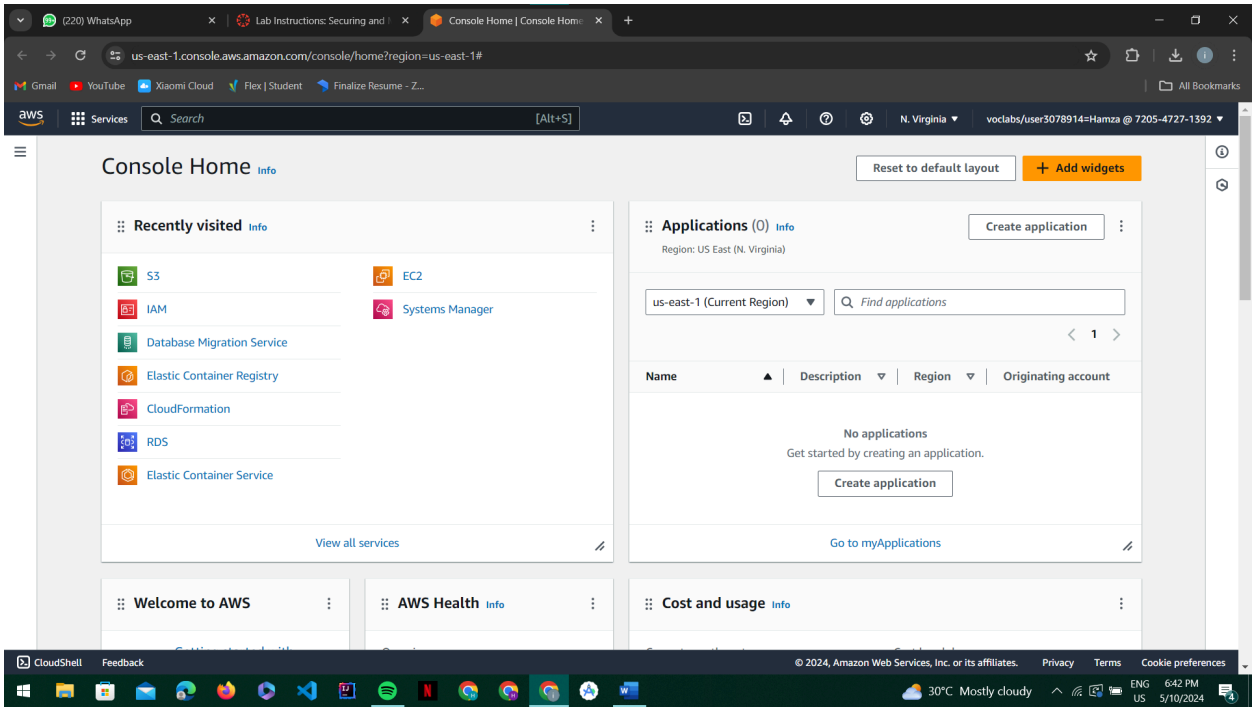


Task 1.1: Create a bucket, apply a bucket policy, and test access

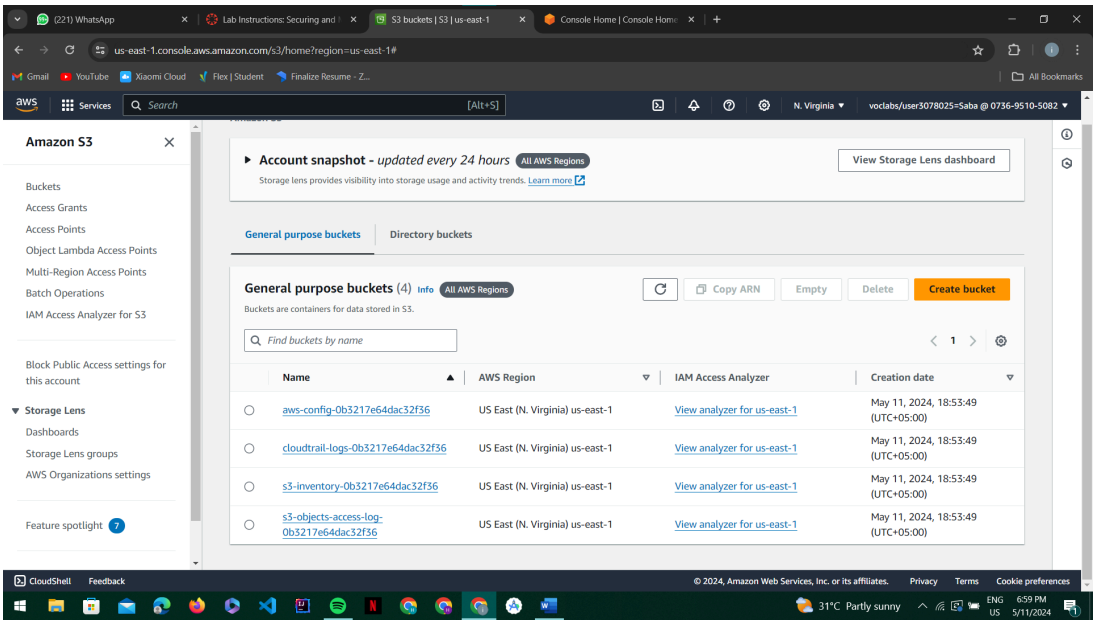
In this task, you are challenged to create a bucket, apply a bucket policy to it, and then test whether Paulo and Mary can access the bucket.

1. After you connect to the AWS Management Console, notice that you're logged in with the *voclabs* IAM role. You can see this in the top-right corner of the console. **Tip**: For most tasks and steps, you will perform actions with this *voclabs* role. However, in some steps, you will log in as a test user (Mary, Paulo, or Sofia).

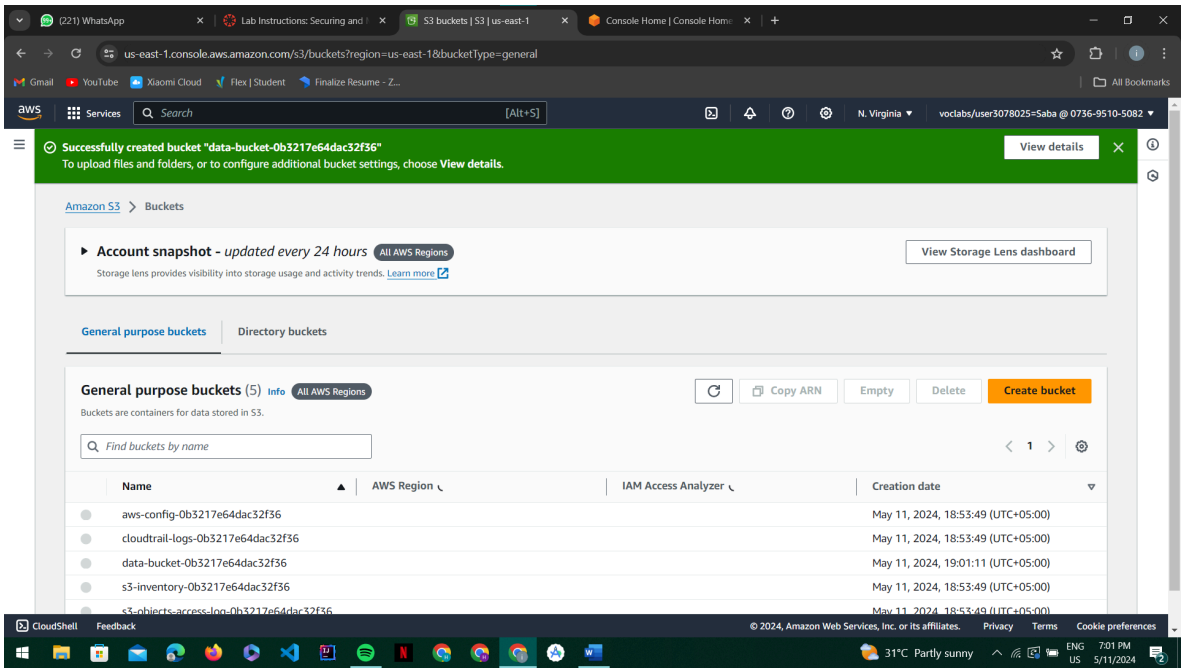
When you aren't sure which user you are logged in as, verify the name that is displayed in this area of the console.



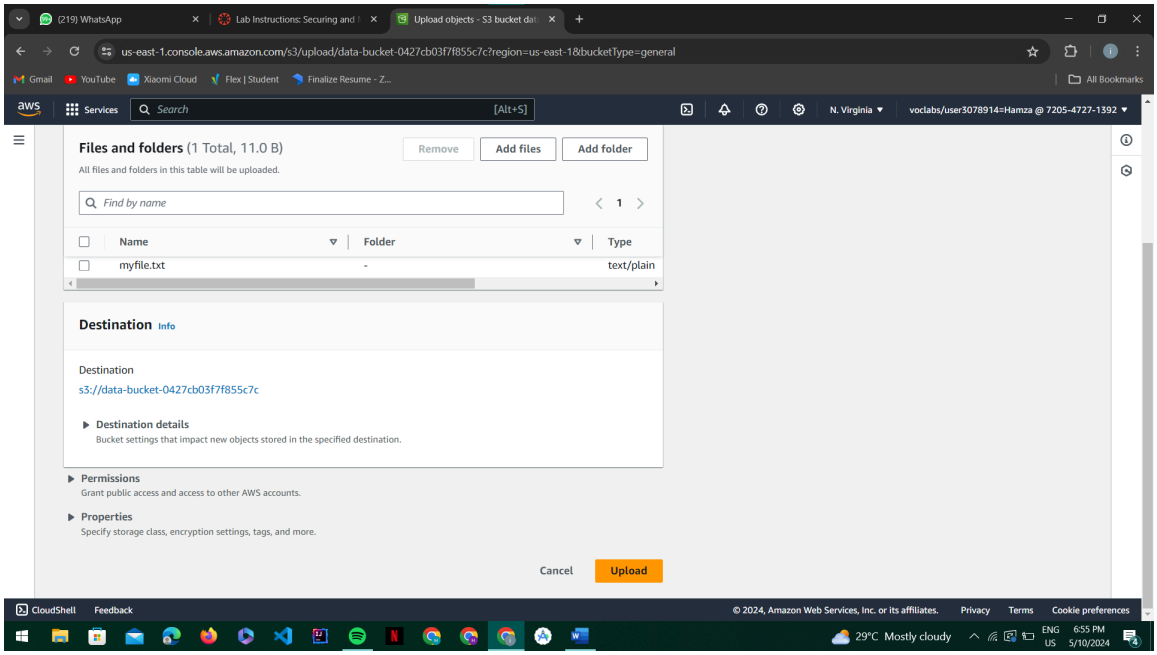
2. In the Amazon S3 console, notice that a few buckets have already been created for you. Copy the unique ID from the bucket names

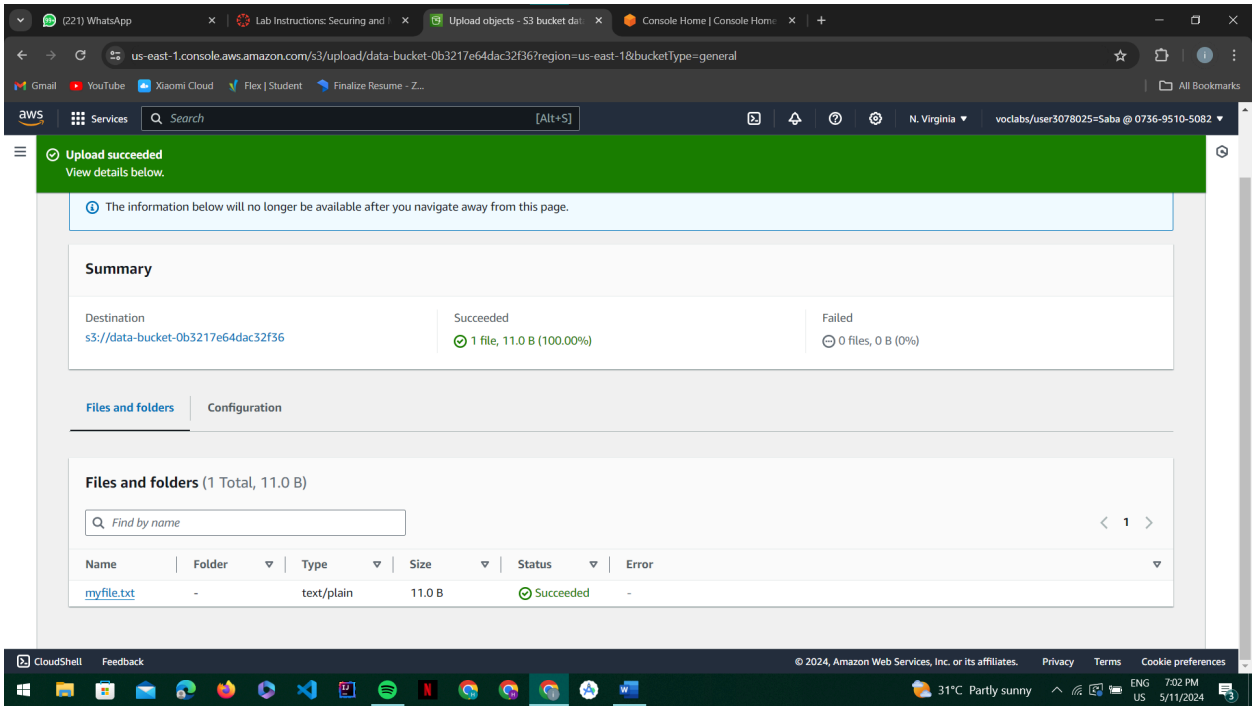


3. Create a new bucket named `data-bucket-<unique-ID>` in *us-east-1*, where `<unique-ID>` is the value that you copied from the name of the existing bucket. Accept all the default bucket settings.

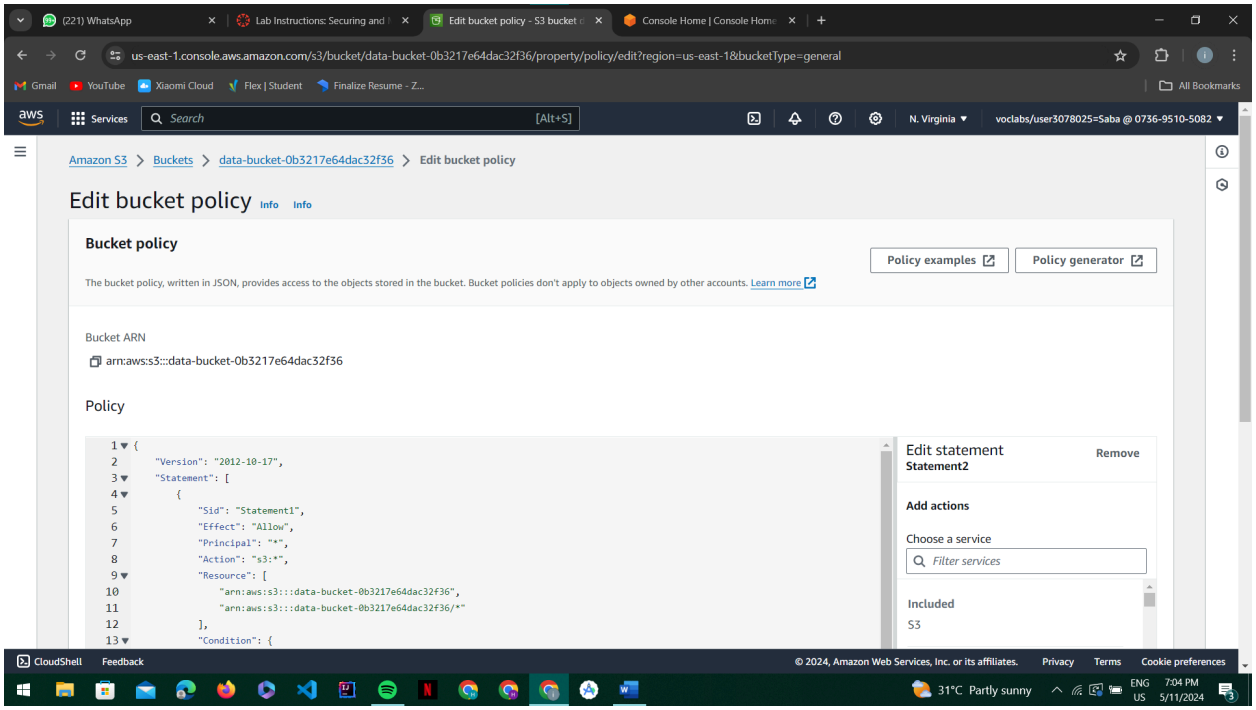


4. Upload an object to the *data-bucket*.
For example, create a file named *myfile.txt*, write "hello world" in the file, save it, and then upload it.





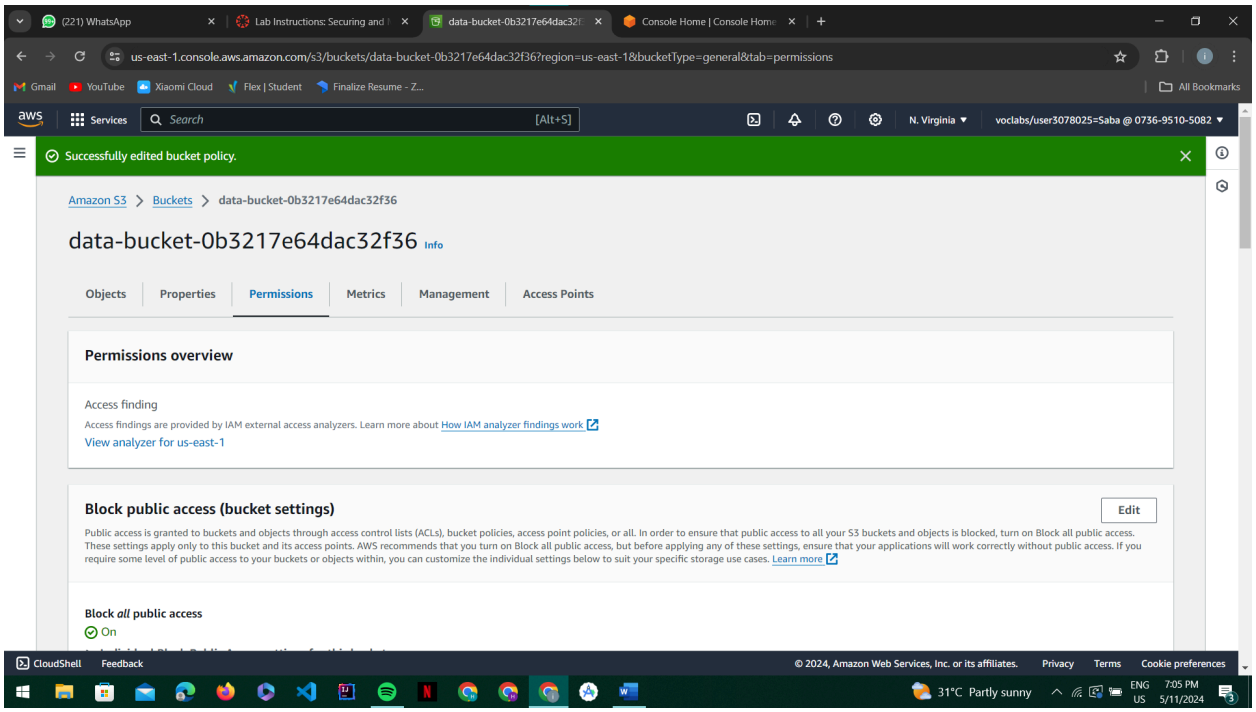
5. Author a bucket policy for the *data-bucket* and apply it to the bucket. The bucket policy should include two statements:
    - The first statement should do the following:
        - *Allow* all Amazon S3 service actions for *all* principals for *data-bucket* and all objects in it.
        - Include a condition that the ARN equals the ARN for the *voclabs* IAM role, the *paulo* IAM user, or the *sofia* IAM user.
    - The second statement should do the following:
        - *Deny* all Amazon S3 service actions for *all* principals for *data-bucket* and all objects in it.
        - Include a condition that the ARN *doesn't* equal the ARN for the *voclabs* IAM role, the *paulo* IAM user, or the *sofia* IAM user.



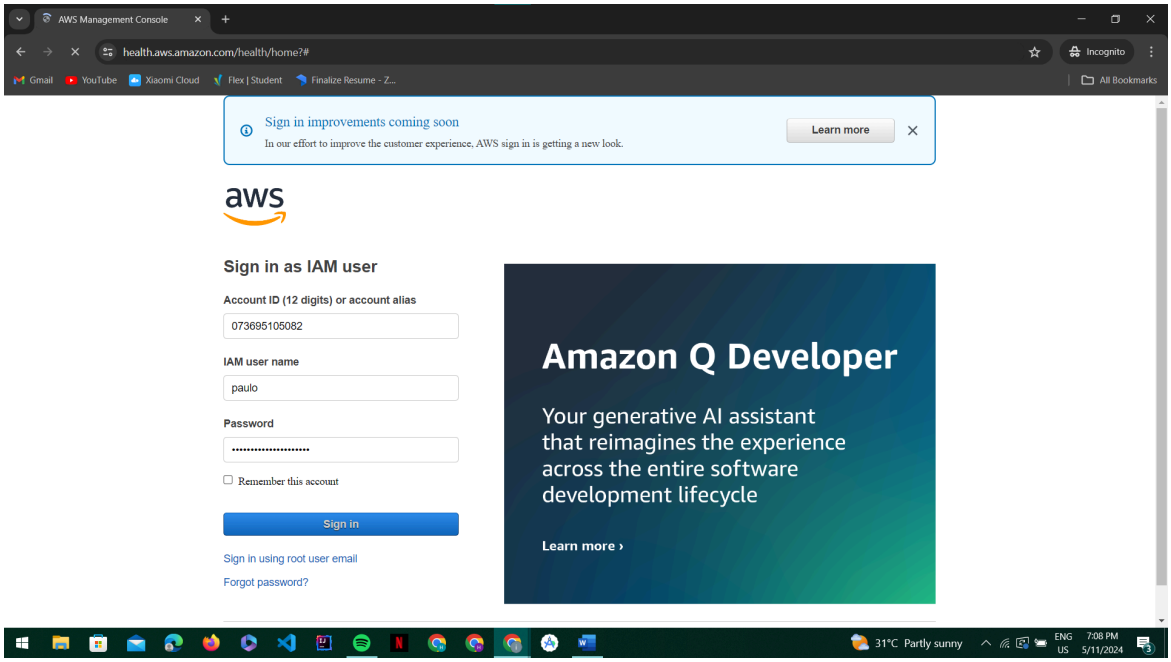

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Statement1",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::data-bucket-0b3217e64dac32f36",
                "arn:aws:s3:::data-bucket-0b3217e64dac32f36/*"
            ],
            "Condition": {
                "ArnEquals": {
```

```
                "aws:PrincipalArn": [
                    "arn:aws:iam::073695105082:user/paulo",
                    "arn:aws:iam::073695105082:user/sofia",
                    "arn:aws:iam::073695105082:role/voclabs"
                ]
            }
        }
    },
    {
        "Sid": "Statement2",
        "Effect": "Deny",
        "Principal": "*",
        "Action": "s3:*",
        "Resource": [
            "arn:aws:s3:::data-bucket-0b3217e64dac32f36",
            "arn:aws:s3:::data-bucket-0b3217e64dac32f36/*"
        ],
        "Condition": {
            "ArnNotEquals": {
                "aws:PrincipalArn": [
                    "arn:aws:iam::073695105082:user/sofia",
                    "arn:aws:iam::073695105082:user/paulo",
                    "arn:aws:iam::073695105082:role/voclabs"
                ]
            }
```

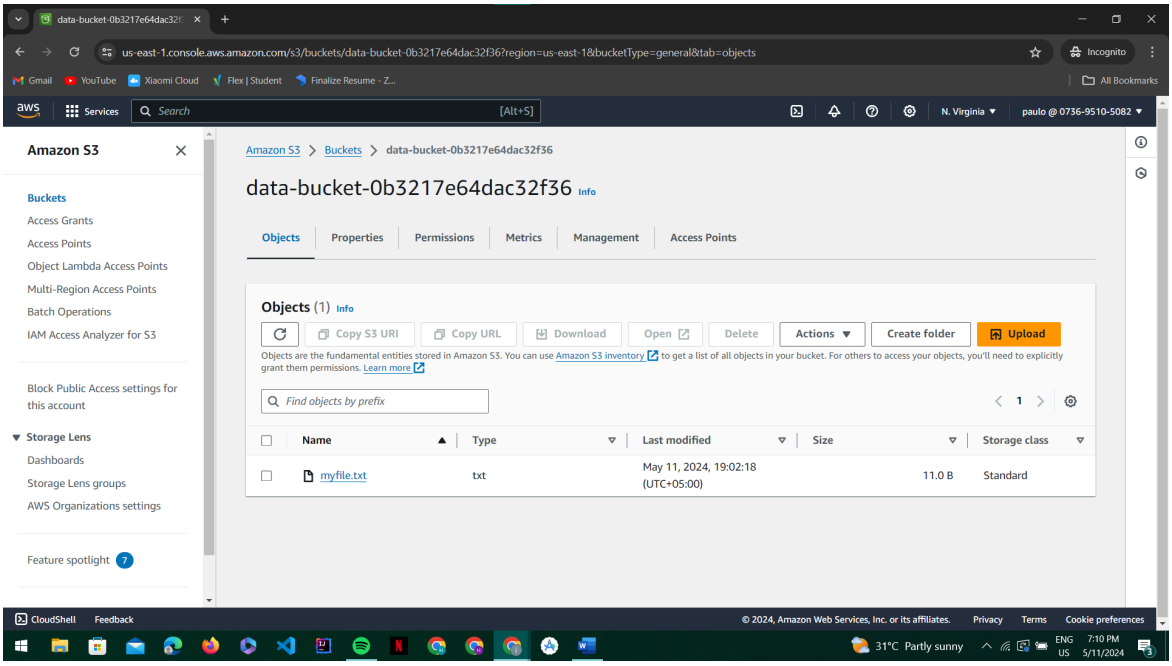
```
            }
        }
    ]
}
```



6. Verify the S3 access level of both the paulo and mary user logins. To do this:
    - In an incognito or private browser window, log in to the AWS Management Console as the *paulo* IAM user.

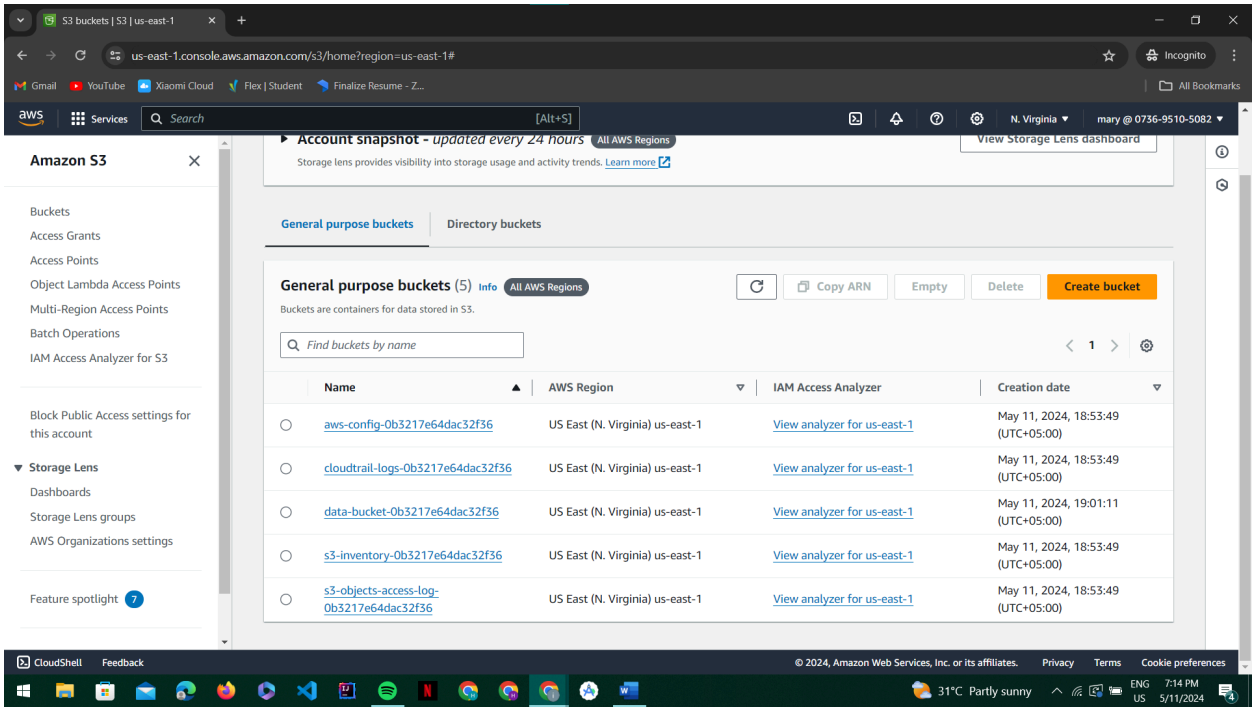


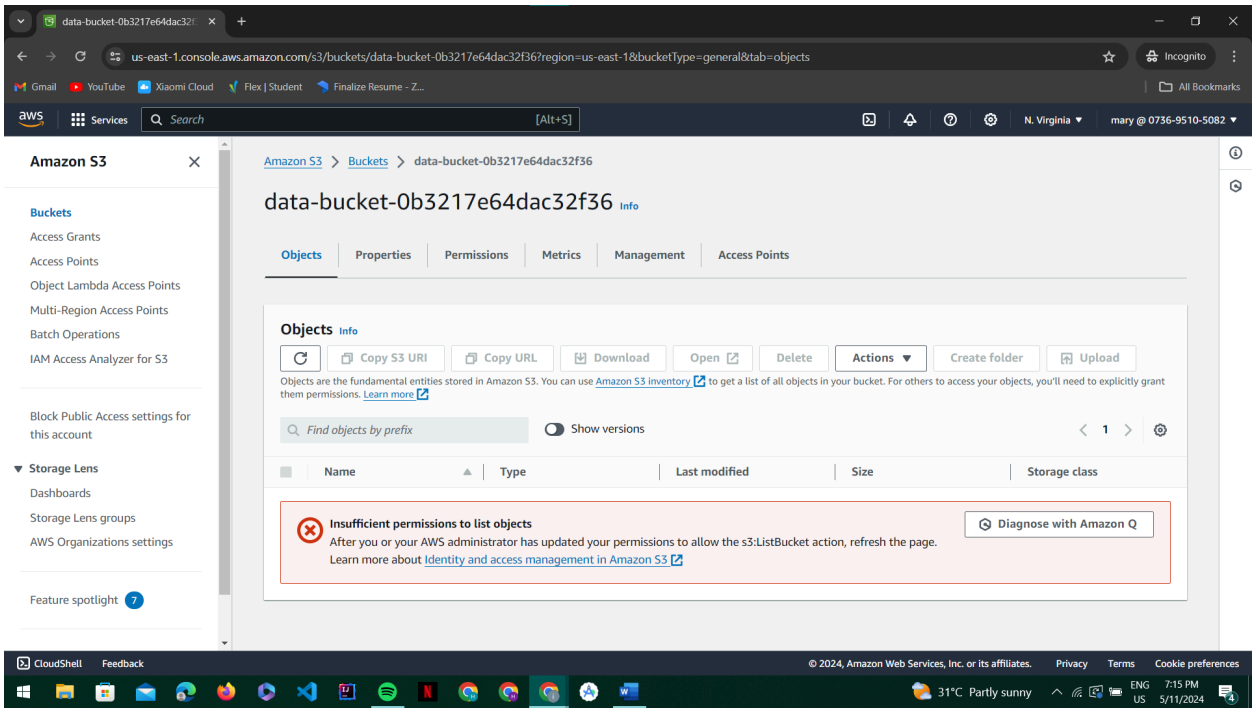
AWS Details above these lab instructions.



- Test the *paulo* user's access to Amazon S3:
  - Verify he can access the *data-bucket* and download objects from it.
  - Verify he can access the contents of any other bucket in the account.
- Test the *Mary* user's access to Amazon S3:

- Verify she sees that four of the buckets show Access status *Buckets and objects not public*, however the **data-bucket** shows Access status *Error*.





- Verify that when she accesses any of the buckets, she sees an "Insufficient permissions to list objects" error.

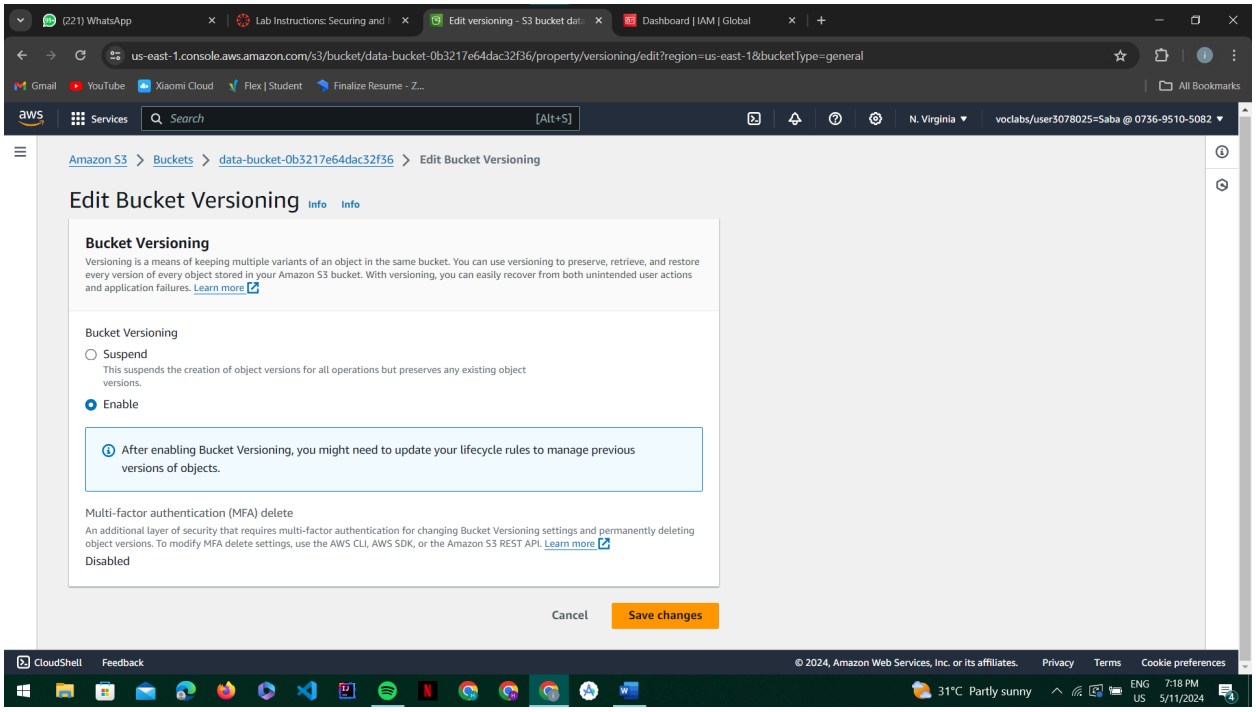
- When you are finished testing, log the *mary* user out of the console.

Task 1.2: Enable versioning and object-level logging on a bucket

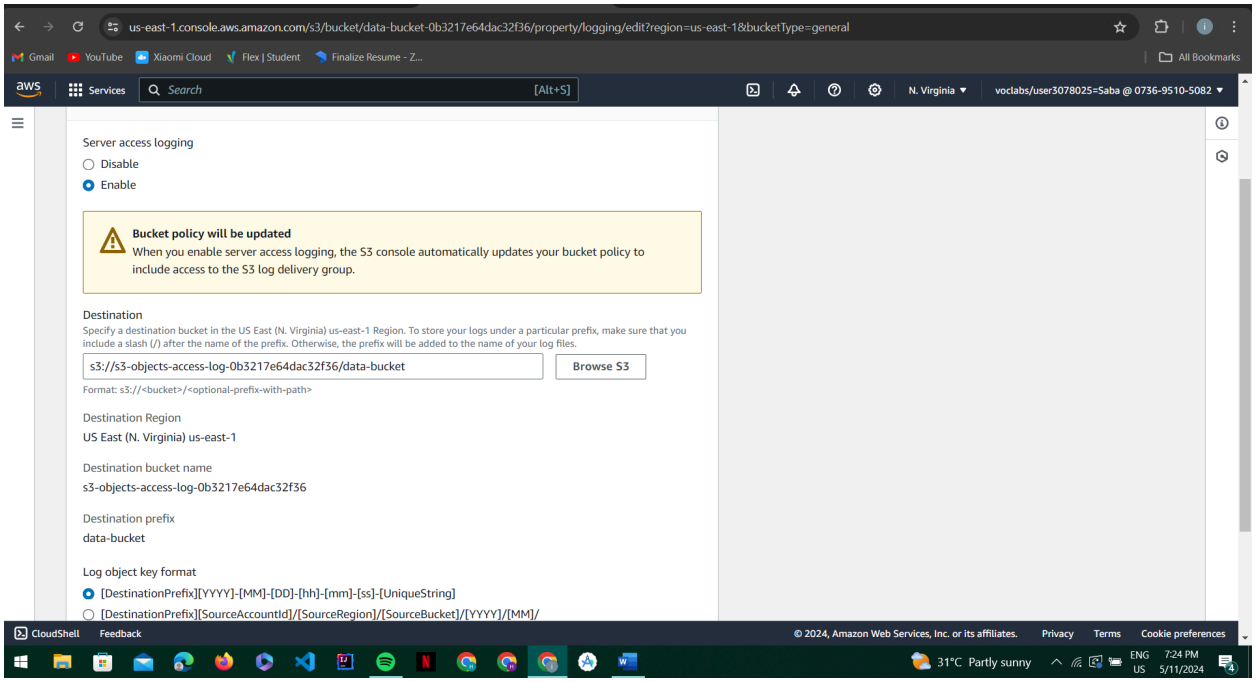
In this task, you are challenged to enable versioning and object-level logging on the *data-bucket*. With versioning enabled, you can track all changes to objects that are stored in the bucket and revert any object to a previous version if needed.

1. Ensure that you are using the browser session where you're logged in to the console as the *voclabs* IAM role.

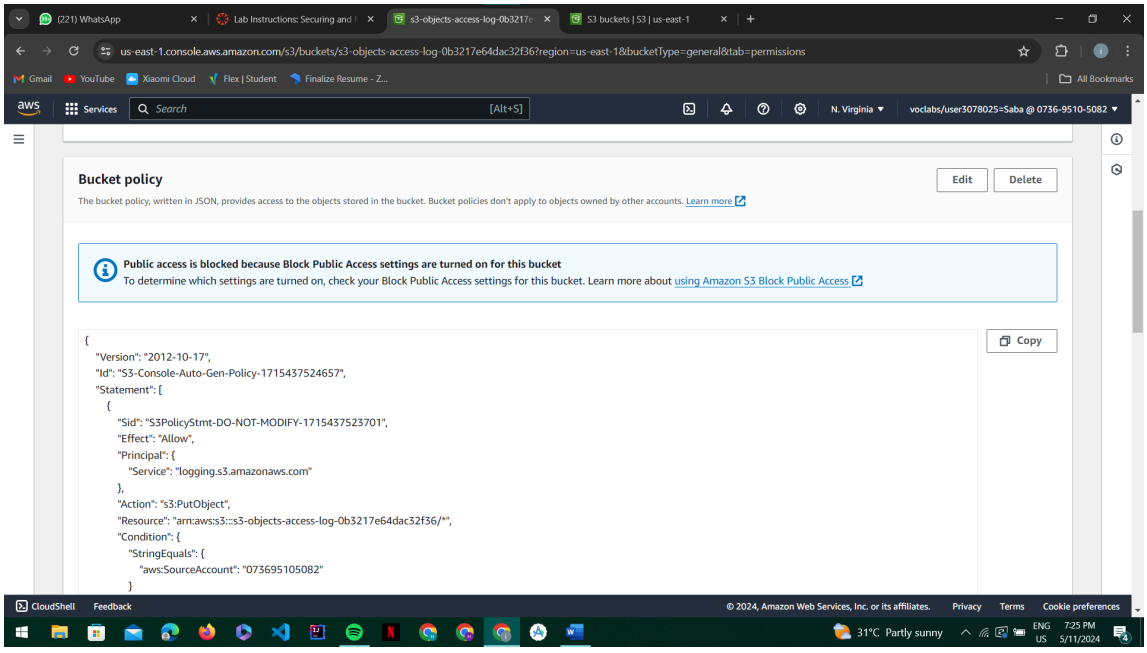
2. Enable versioning on the *data-bucket*.



3. Enable server access logging on the *data-bucket*
   Configure the logs to be written to the *s3-objects-access-log* bucket.
   Add `/data-bucket` as a prefix to the end of the **Target bucket** path (it doesn't need to match the actual *data-bucket* name).



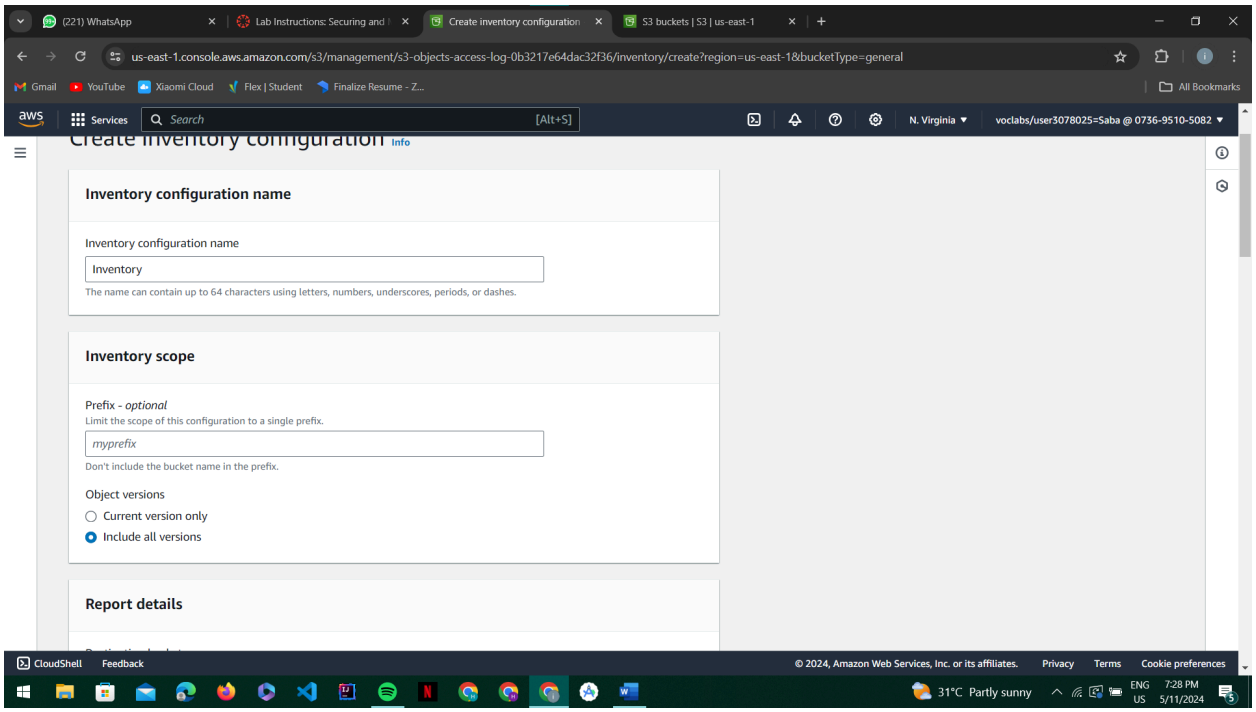
4. Verify that the bucket policy for the *s3-objects-access-log* bucket now includes a statement to allow the Amazon S3 logging service to write objects to the bucket.

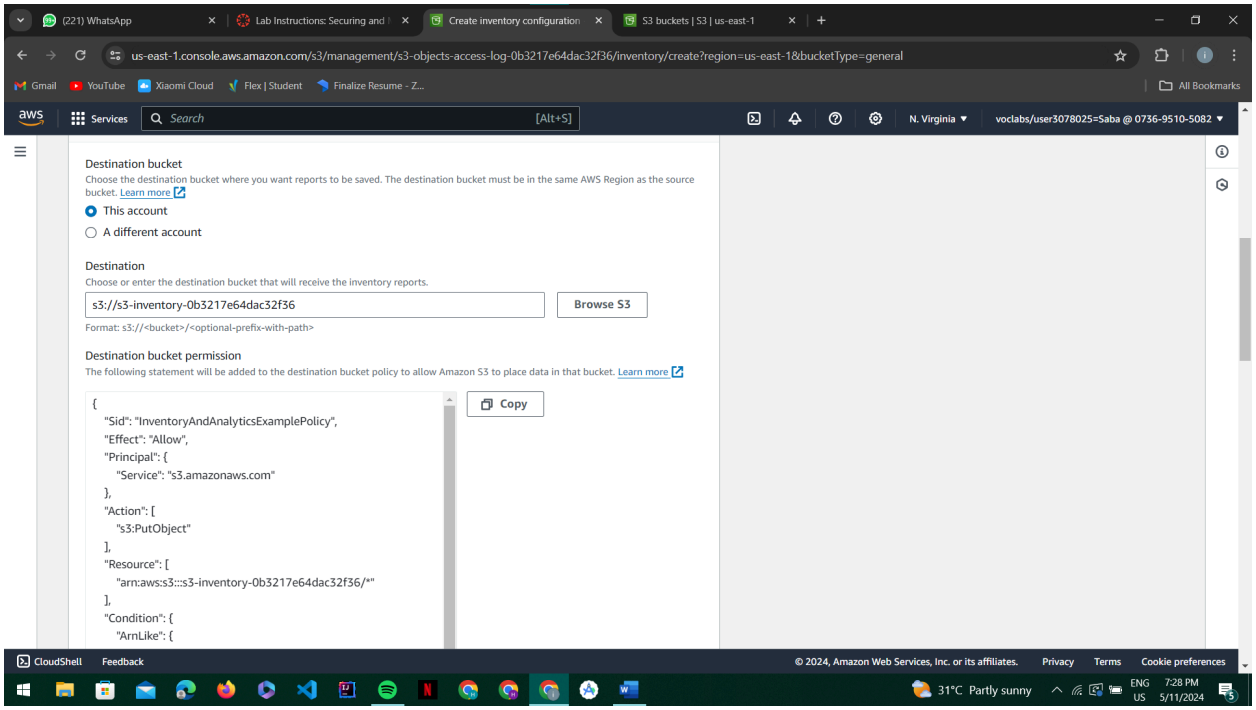


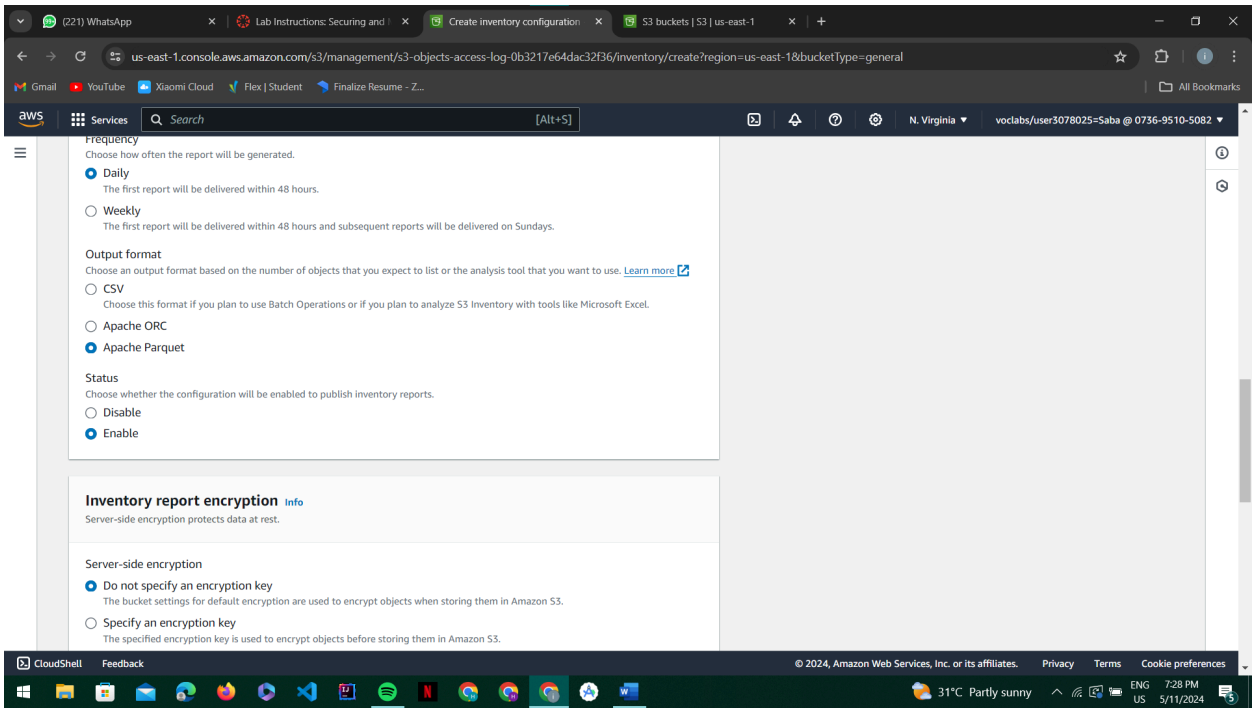
Task 1.3: Implement the S3 Inventory feature on a bucket

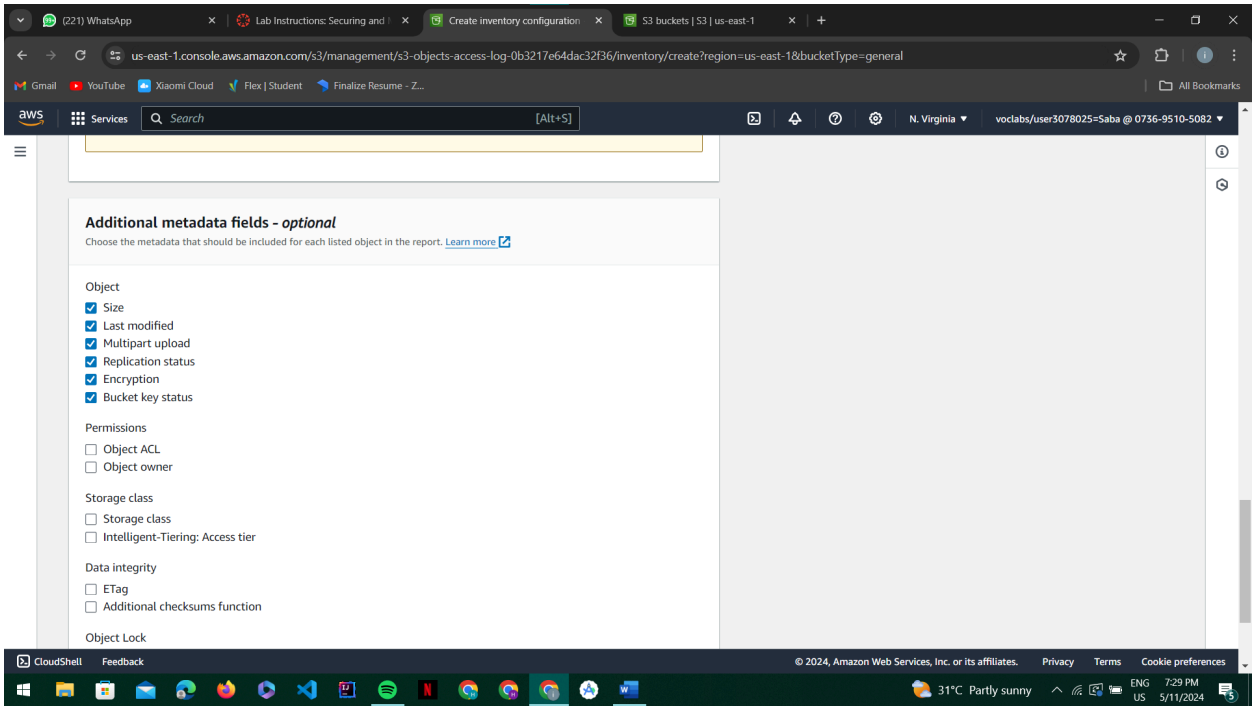
In this task, you are challenged to enable the S3 Inventory feature to monitor changes to objects that are stored in an S3 bucket. S3 Inventory provides a scheduled report of the metadata and object-level changes to your S3 objects and buckets. By using the feature, you can track changes to the stored objects and detect potential security incidents.

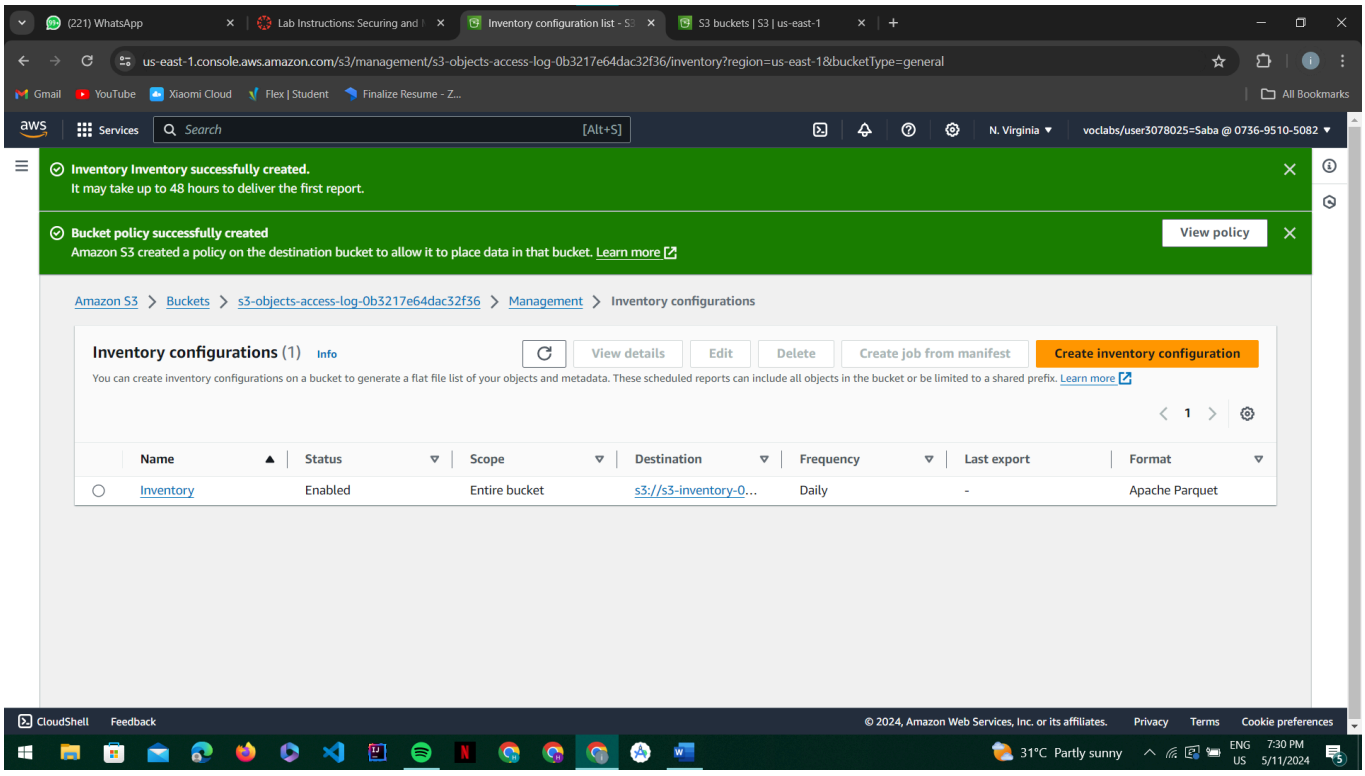
1. Enable the S3 Inventory feature on the *data-bucket*.
   Tip: You can do this in the bucket's **Management** tab.
   Use the following settings:
   - Use `Inventory` as the inventory configuration name.
   - Include all object versions.
   - Set the *s3-inventory* bucket as the destination for report details.
   - Set *Apache Parquet* as the output format.
   - Select all six additional metadata *Object* fields.




Destination bucket
Choose the destination bucket where you want reports to be saved. The destination bucket must be in the same AWS Region as the source bucket. Learn more
This account
A different account
Destination
Choose or enter the destination bucket that will receive the inventory reports.
s3://s3-inventory-0b3217e64dac32f36
Browse S3
Format: s3://<bucket>/<optional-prefix-with-path>
Destination bucket permission
The following statement will be added to the destination bucket policy to allow Amazon S3 to place data in that bucket. Learn more
Copy
{
"Sid": "InventoryAndAnalyticsExamplePolicy",
"Effect": "Allow",
"Principal": {
"Service": "s3.amazonaws.com"
},
"Action": [
"s3:PutObject"
],
"Resource": [
"arn:aws:s3:::s3-inventory-0b3217e64dac32f36/*"
],
"Condition": {
"ArnLike": {
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Frequency
Choose how often the report will be generated.
Daily
The first report will be delivered within 48 hours.
Weekly
The first report will be delivered within 48 hours and subsequent reports will be delivered on Sundays.
Output format
Choose an output format based on the number of objects that you expect to list or the analysis tool that you want to use. Learn more
CSV
Choose this format if you plan to use Batch Operations or if you plan to analyze S3 Inventory with tools like Microsoft Excel.
Apache ORC
Apache Parquet
Status
Choose whether the configuration will be enabled to publish inventory reports.
Disable
Enable
Inventory report encryption Info
Server-side encryption protects data at rest.
Server-side encryption
Do not specify an encryption key
The bucket settings for default encryption are used to encrypt objects when storing them in Amazon S3.
Specify an encryption key
The specified encryption key is used to encrypt objects before storing them in Amazon S3.
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Create inventory configuration
us-east-1.console.aws.amazon.com/s3/management/s3-objects-access-log-0b3217e64dac32f36/inventory/create?region=us-east-1&bucketType=general
Additional metadata fields - optional
Choose the metadata that should be included for each listed object in the report. Learn more
Object
Size
Last modified
Multipart upload
Replication status
Encryption
Bucket key status
Permissions
Object ACL
Object owner
Storage class
Storage class
Intelligent-Tiering: Access tier
Data integrity
ETag
Additional checksums function
Object Lock
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

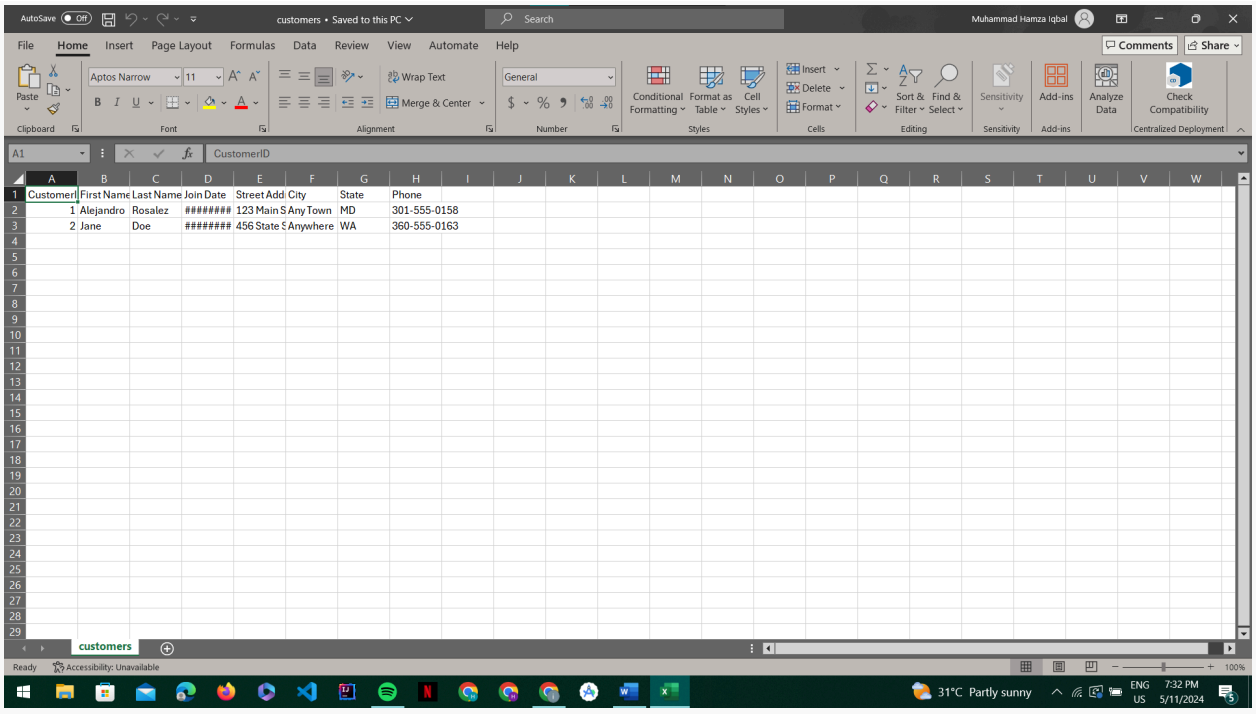

Inventory Inventory successfully created.
It may take up to 48 hours to deliver the first report.
Bucket policy successfully created
Amazon S3 created a policy on the destination bucket to allow it to place data in that bucket. Learn more
View policy
Amazon S3 > Buckets > s3-objects-access-log-0b3217e64dac32f36 > Management > Inventory configurations
Inventory configurations (1) Info
View details
Edit
Delete
Create job from manifest
Create inventory configuration
You can create inventory configurations on a bucket to generate a flat file list of your objects and metadata. These scheduled reports can include all objects in the bucket or be limited to a shared prefix. Learn more
Name
Status
Scope
Destination
Frequency
Last export
Format
Inventory
Enabled
Entire bucket
s3://s3-inventory-0...
Daily
-
Apache Parquet
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Task 1.4: Confirm that versioning works as intended

In this task, you are challenged to access the AWS account as the *paulo* user and upload an object to the *data-bucket*. Then, you are challenged to confirm that versioning is enabled on the object. You are also challenged to test access as the *mary* user. In the next task, you will analyze the object-level logs to see the actions that you took as different users in this task.

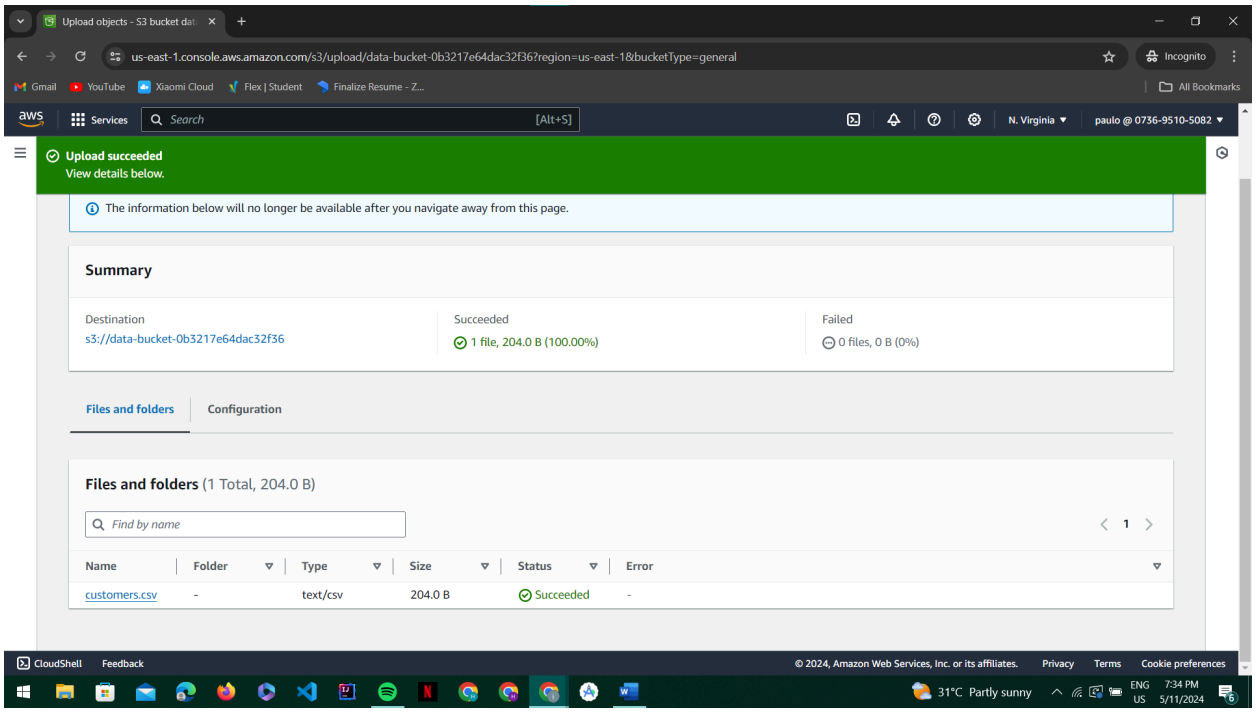
1. On your computer, create a new file named `customers.csv`. Then, copy the following text to the file and save the changes.

   ```
   CustomerID,First Name,Last Name,Join Date,Street Address,City,State,Phone
   1,Alejandro,Rosalez,12/12/2013,123 Main St.,Any Town,MD,301-555-0158
   2,Jane,Doe,10/5/2014,456 State St.,Anywhere,WA,360-555-0163
   ```

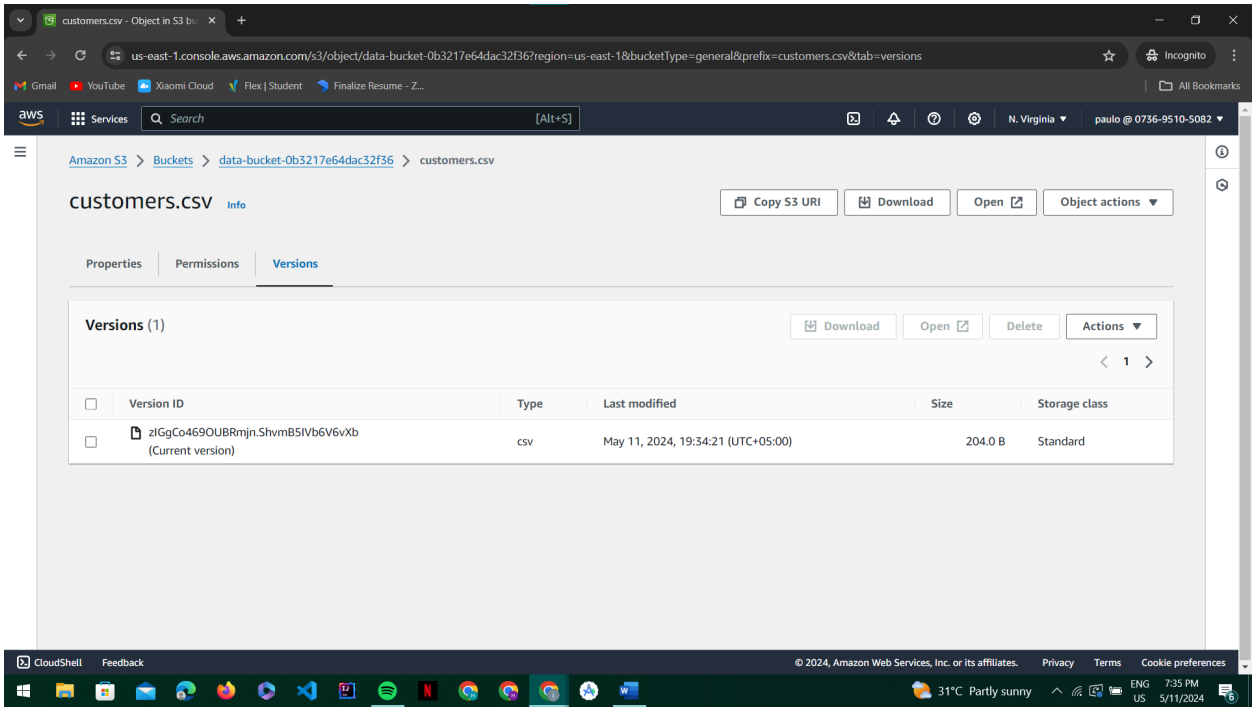


2. Log in to the AWS account as the *paulo* user and upload the *customers.csv* file to the *data-bucket*.
   After you upload the file, check the server-side encryption settings that are

applied to the object. It should be encrypted with Amazon S3 managed keys (SSE-S3).

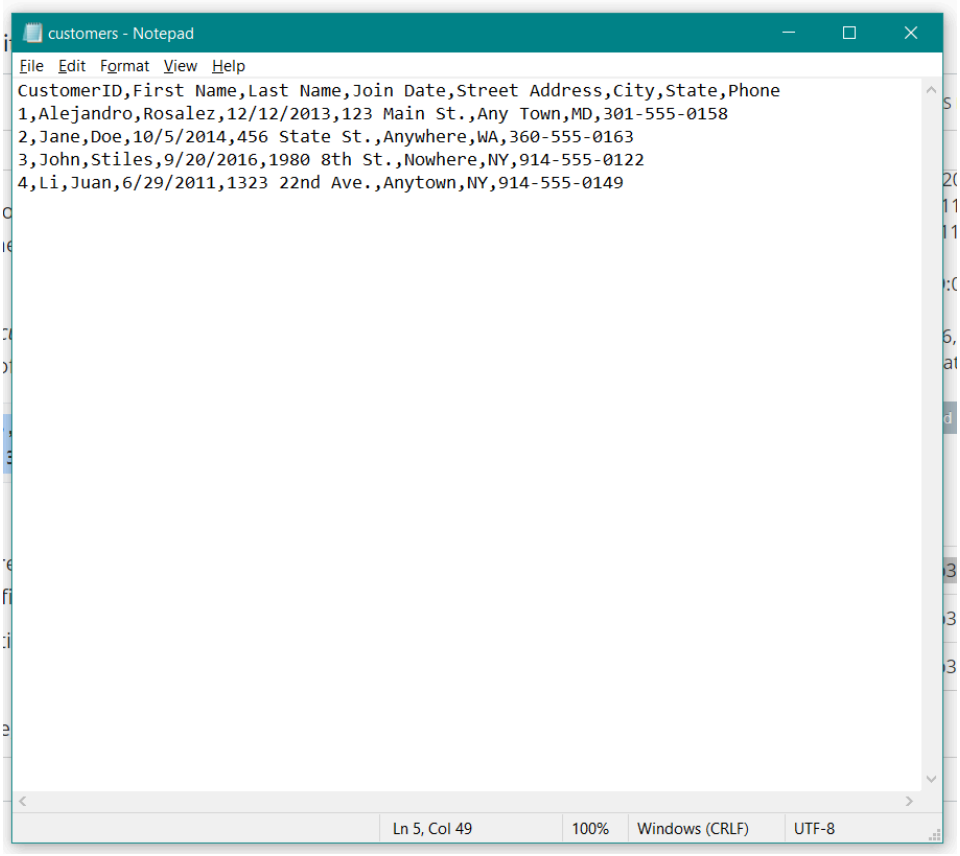


3. Analyze how many versions of customer.csv exist by navigating to the *customers.csv* details page and choosing the **Versions** tab.

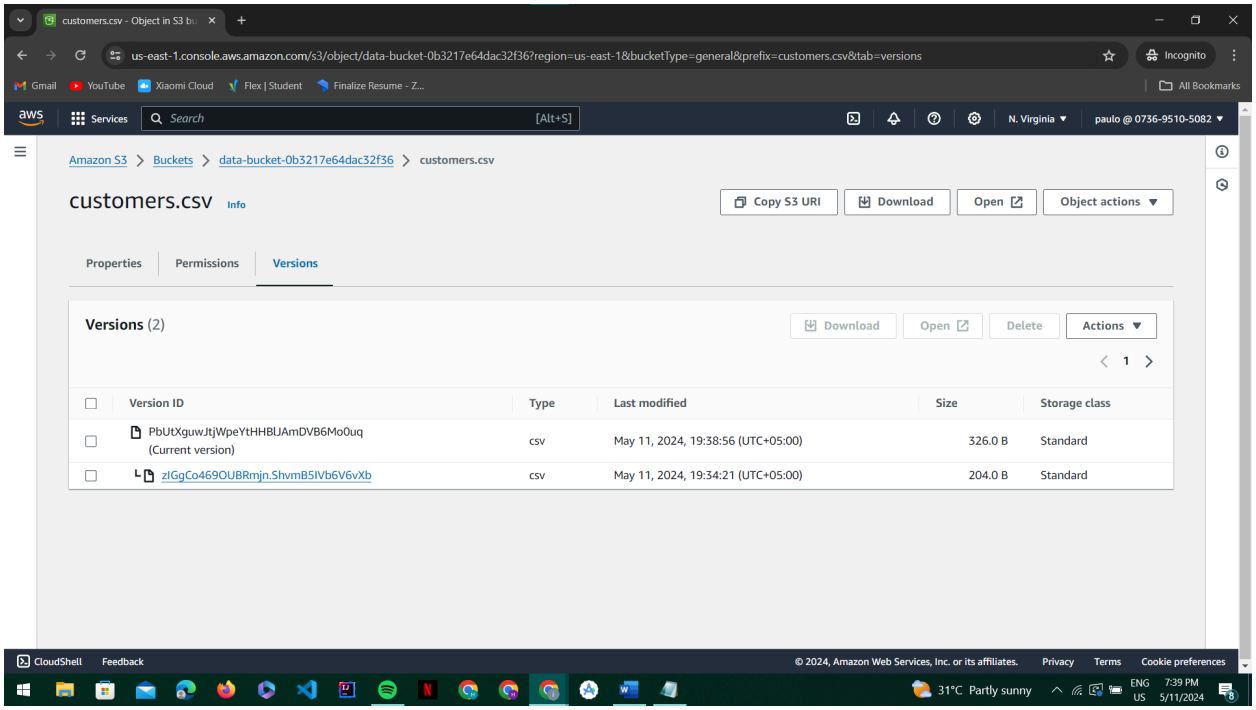


4. On your computer, edit the *customers.csv* file and add more data to it. For example, add the following two rows of data at the bottom of the file.

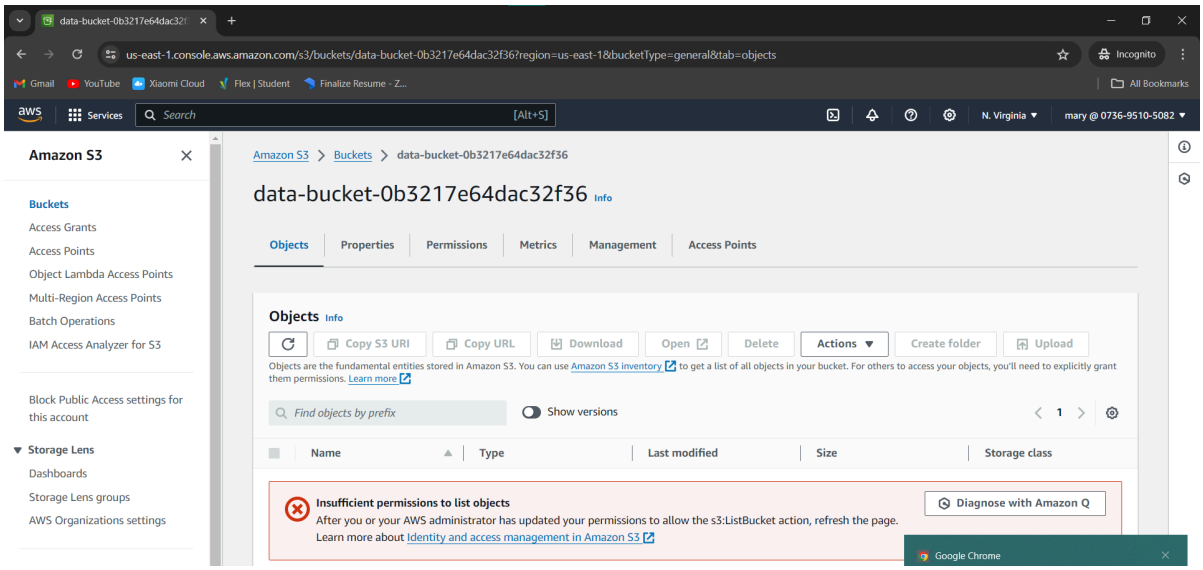
   ```
   3,John,Stiles,9/20/2016,1980 8th St.,Nowhere,NY,914-555-0122
   4,Li,Juan,6/29/2011,1323 22nd Ave.,Anytown,NY,914-555-0149
   ```



5. In the browser window where you're logged in as the *paulo* user, upload the new version of the *customers.csv* file to the *data-bucket*.
   After you upload the file, notice the versions that are available.



6. From the Amazon S3 console, open the *current version* of the *customers.csv* file to confirm that it contains all four rows of records.
7. Log in as the *mary* user and take action in the Amazon S3 console to generate log events.
    - Log the *paulo* user out of the console, and then log in as the *mary* user.
    - Try to access objects in the *data-bucket*.
      As you saw earlier, the *mary* user cannot access this bucket or its contents.



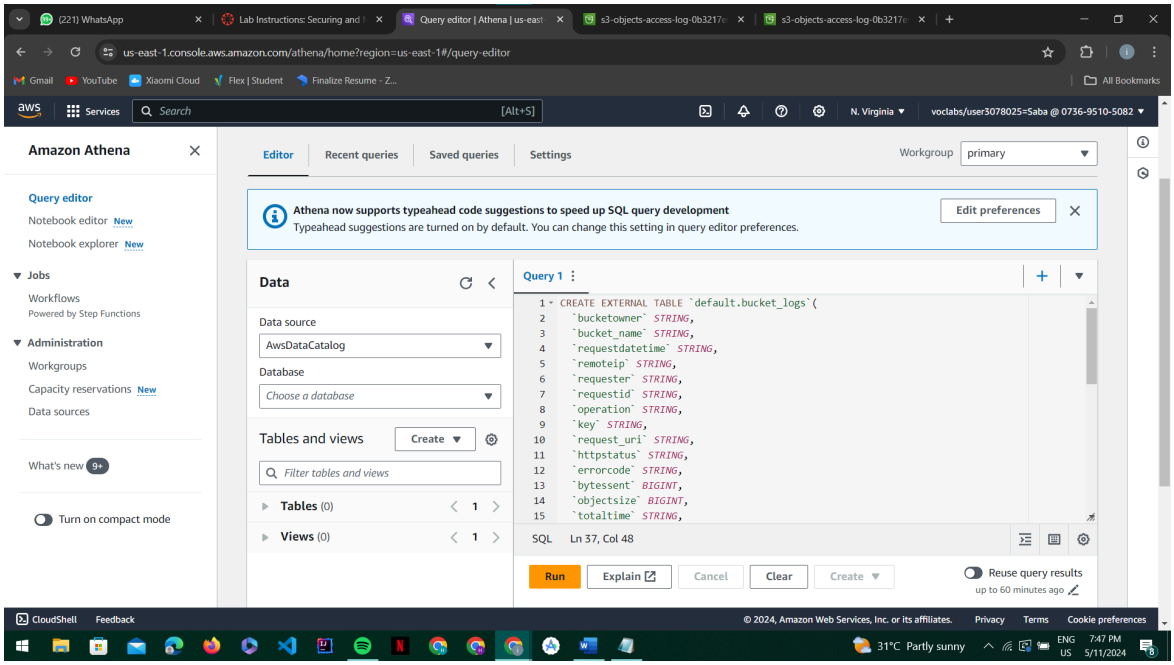
Task 1.5: Confirm object-level logging and query the access logs by using Athena

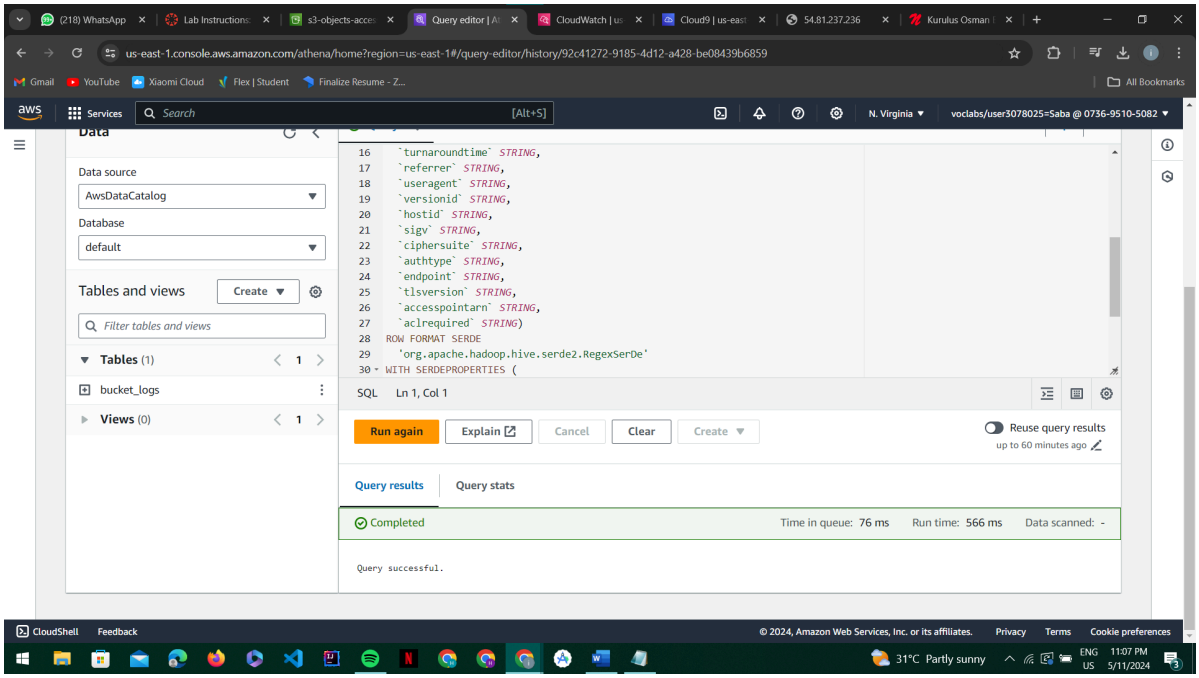
In this task, you are challenged to confirm the S3 object-level logging you enabled earlier is successfully writing log data to S3. You will also use Athena to query these logs.

1. In the browser session where you're logged in to the console as the *voclabs* role, observe the objects that are stored in the *s3-objects-access-log* bucket. Download one of the objects, open it, and review the contents.

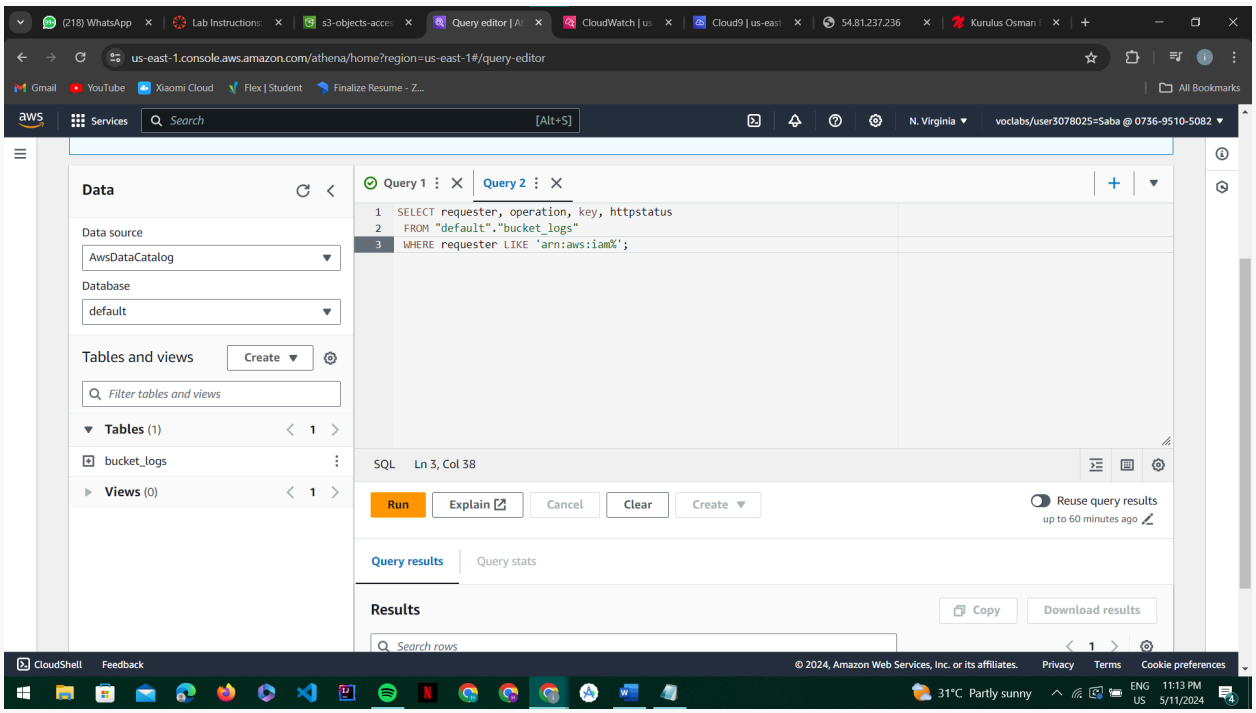
2. Create an Athena table from the access logs. To do this:
    - Create an S3 bucket named `athena-results-<random-number>` where `<random-number>` is some random number. Accept the default bucket settings.
    - In the Athena console open the query editor.
    - Before running queries, set the athena-results bucket to be the location for query results.
      Note: consult the Amazon Athena documentation for details as needed.
    - Then, in the **Editor** tab paste the following query into the query area:
    - ```
      CREATE EXTERNAL TABLE `default.bucket_logs`(
        `bucketowner` STRING,
        `bucket_name` STRING,
        `requestdatetime` STRING,
        `remoteip` STRING,
        `requester` STRING,
        `requestid` STRING,
        `operation` STRING,
        `key` STRING,
        `request_uri` STRING,
        `httpstatus` STRING,
        `errorcode` STRING,
        `bytessent` BIGINT,
        `objectsize` BIGINT,
        `totaltime` STRING,
        `turnaroundtime` STRING,
        `referrer` STRING,
        `useragent` STRING,
        `versionid` STRING,
        `hostid` STRING,
        `sigv` STRING,
      ```
      ```

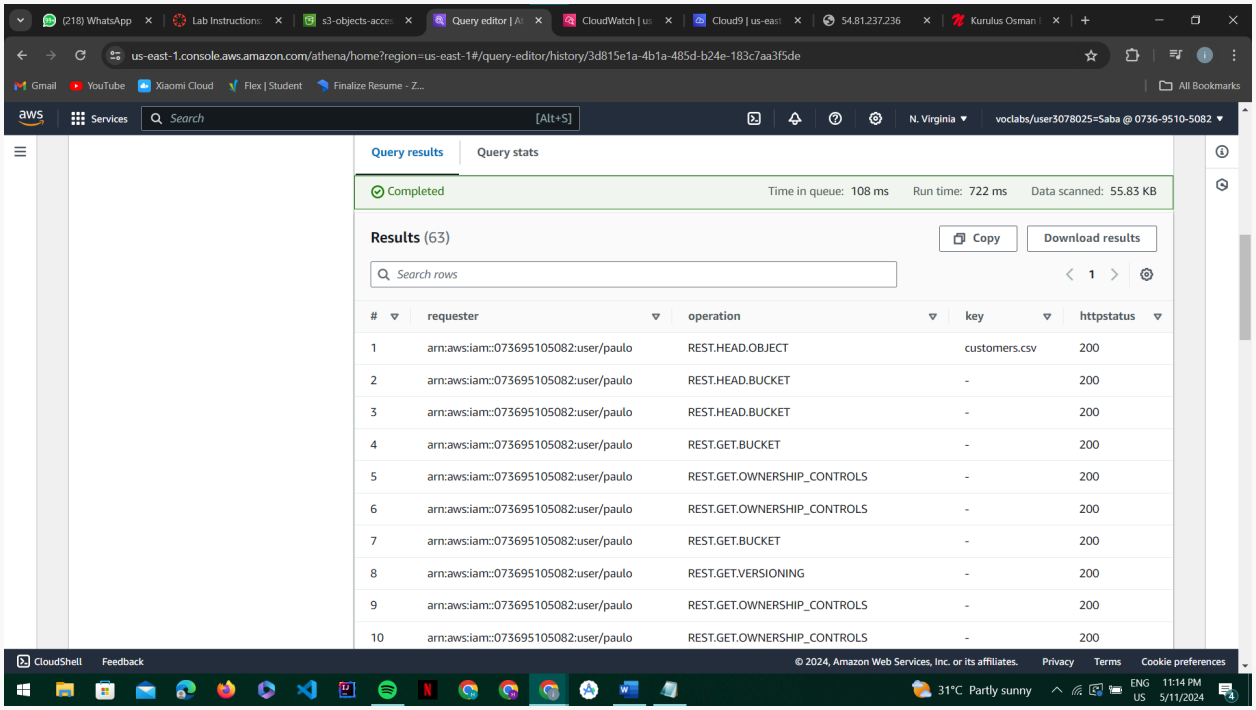
```
 `ciphersuite` STRING,
 `authtype` STRING,
 `endpoint` STRING,
 `tlsversion` STRING,
 `accesspointarn` STRING,
 `aclrequired` STRING)
ROW FORMAT SERDE
 'org.apache.hadoop.hive.serde2.RegexSerDe'
 WITH SERDEPROPERTIES (
 'input.regex'='([^ ]*) ([^ ]*) \\[(.*?)\\] ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) (\"[^\"]*\"|-)
(-|[0-9]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) (\"[^\"]*\"|-) ([^ ]*)(?: ([^ ]*) ([^ ]*) ([^ ]*)
([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*))?.*$')
 STORED AS INPUTFORMAT
 'org.apache.hadoop.mapred.TextInputFormat'
 OUTPUTFORMAT
 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION
 's3://s3-objects-access-log-<UNIQUE-ID>/'
```





3. Query the table to discover access details.
    - Preview the contents of the **bucket_logs** table.Run the following query.
    - ```
      SELECT requester, operation, key, httpstatus
       FROM "default"."bucket_logs"
       WHERE requester LIKE 'arn:aws:iam%';
      ```


      ```



Task 1.6: Review the S3 Inventory report by using S3 Select

Recall that in task 1.3, you created an inventory configuration for the *data-bucket*. In this task, you are challenged to access and review the S3 Inventory report by using Amazon S3 Select.

Important: It can take up to 48 hours for AWS to deliver the first inventory report. If you don't yet see data, you might need to skip this task and return to it later.

1. In the Amazon S3 console, on the *data-bucket* details page, choose the **Management** tab. In the **Inventory configurations** section, choose the **Destination** link for the **Inventory** configuration.

2. Locate the .parquet file in the inventory bucket, and use S3 Select to query the inventory and output the query in JSON format.
   You should see at least three entries: one entry for the first file that you uploaded (in task 1.1), and additional entries for each time you uploaded a new version of the customers.csv file.

Phase 2: Securing VPCs

After you secure the data in Amazon S3, the leadership team for AnyCompany Financial wants you to focus on securing the *network* in the AWS Cloud that hosts the company's critical applications. They are aware of recent network security incidents and want to ensure that their network is protected from unauthorized access and attacks. Your task is to secure the virtual private clouds (VPCs) for the company's web servers.

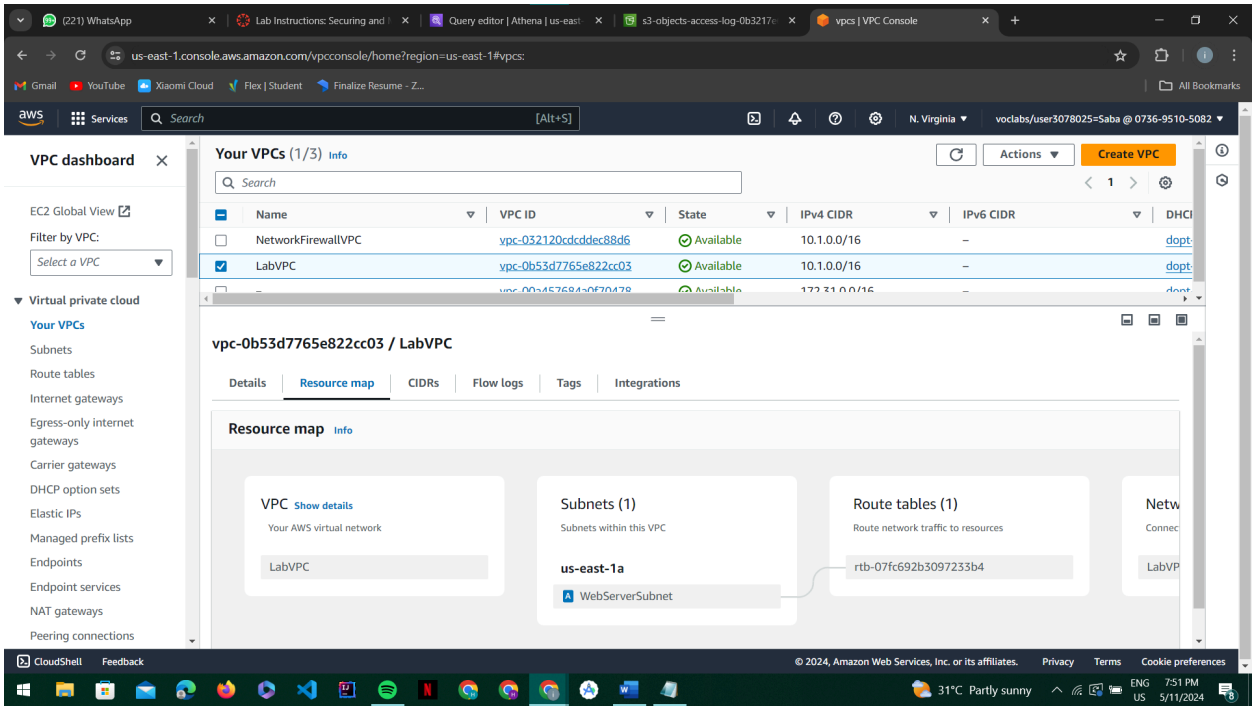
An inexperienced employee of AnyCompany Financial created the LabVPC and the WebServer instance that pre-exist in your lab project environment. The employee made some mistakes and the result is that the network is not properly configured. In tasks 2.1 through 2.4 you will analyze the existing configurations and make updates to correct the network configuration.

The following diagram shows the resources that already exist in the lab environment and which you will use for the first five tasks in this phase:

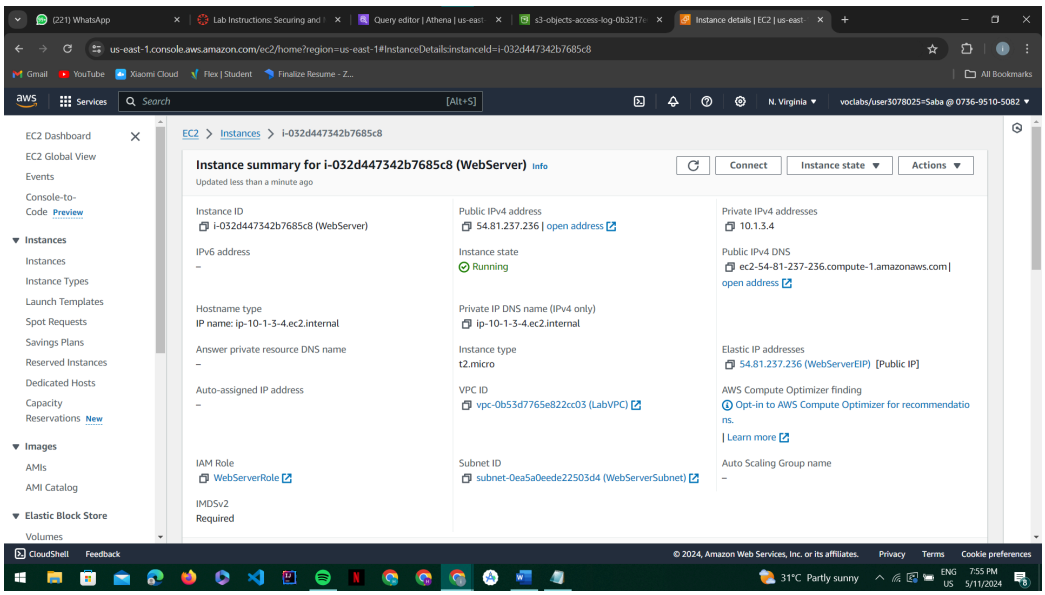
Task 2.1: Review LabVPC and its associated resources

In this task, you will familiarize yourself with resources that already exist in the lab environment.

1. In the Amazon VPC console, verify that you are observing the *us-east-1* Region, and analyze the resource map of the existing *LabVPC*.
   Notice that it contains the following resources:
   - A subnet named *WebServerSubnet*.
   - A route table. This is the main route table that is created by default for any VPC.
   - An internet gateway.
2. Notice that the subnet isn’t currently routed to the internet gateway.



3. In the IAM console, locate the *VPCFlowLogsRole* IAM role. Review the permissions that are granted to this role.
4. In the Amazon EC2 console, observe the details for the *WebServer* instance. Notice public IPv4 address is assigned, An IAM role is attached.
    - A security group named *Webserver Security Group* is associated.
    - The instance runs in a subnet named *WebServer Subnet*, which is in *LabVPC*.

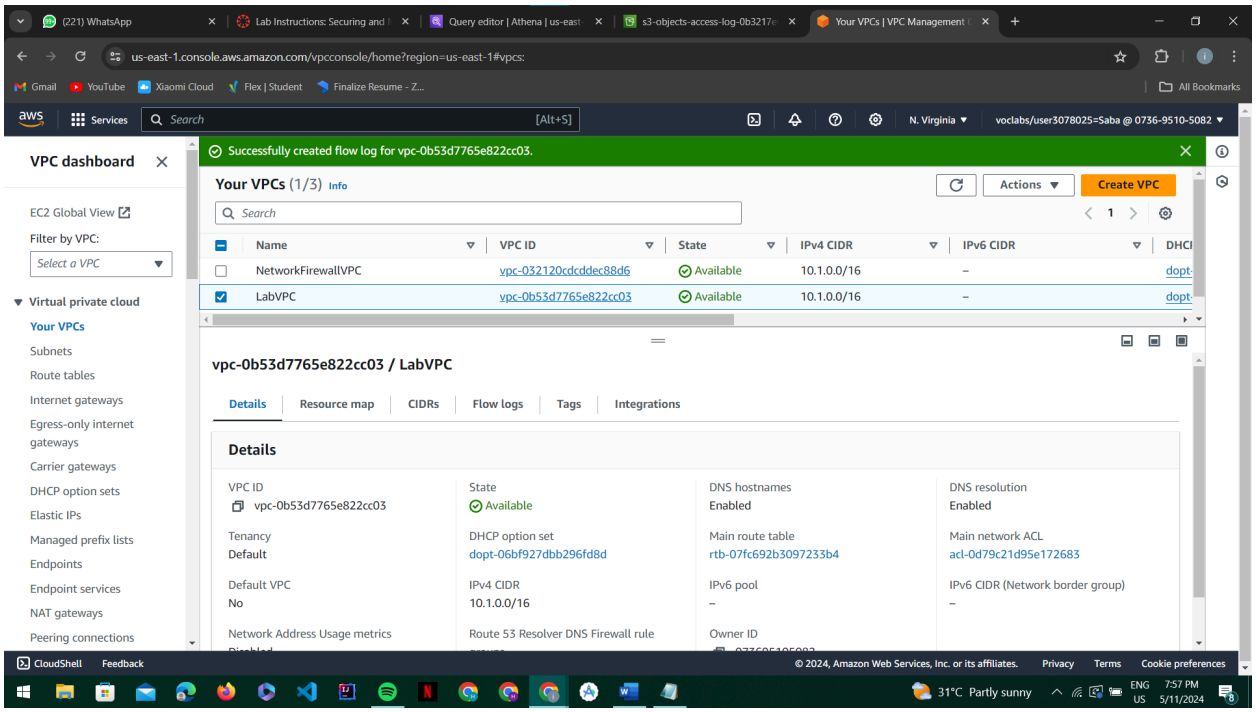


Task 2.2: Create a VPC flow log

In this task, you are challenged to create a VPC flow log for *LabVPC*. The VPC Flow Logs feature can help you understand how inbound and outbound traffic flows through the VPC. The feature also provides monitoring information that will provide insights for how to secure the web server, subnets, and VPC in later tasks.

By the end of this task, you will have configured the architecture that is shown in the following diagram:

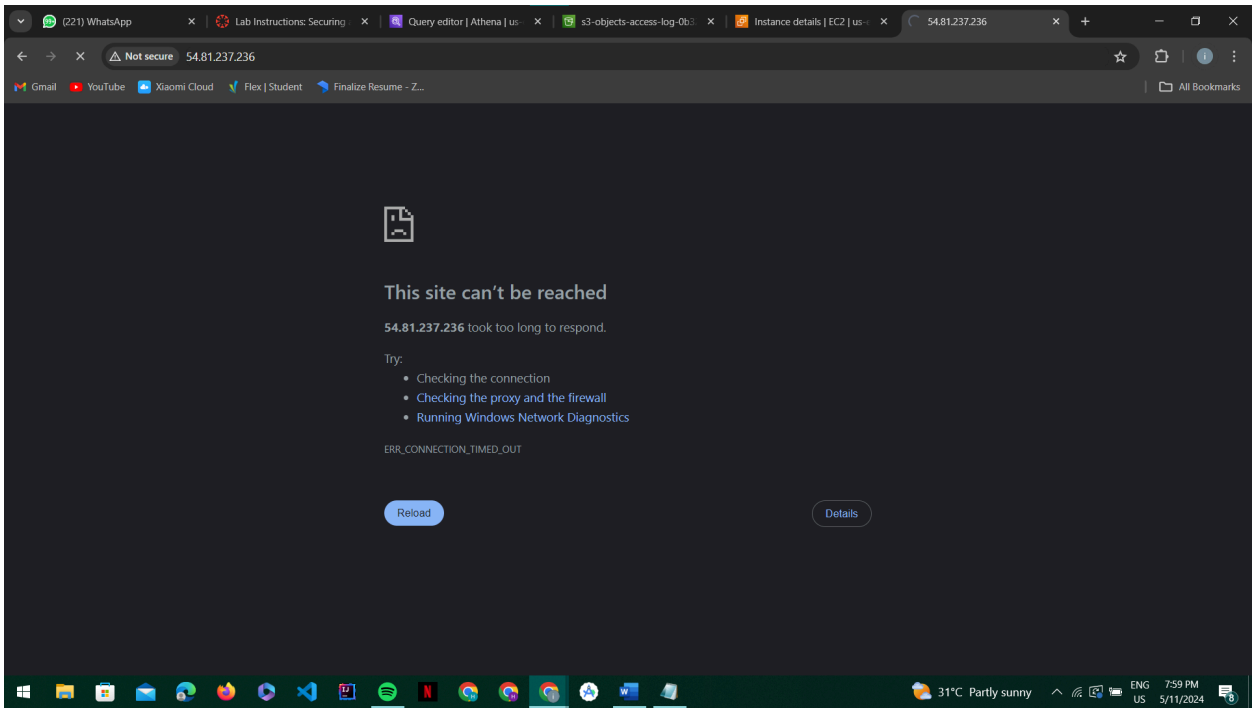
1. Create a VPC flow log for *LabVPC*.
    - Name the flow log `LabVPCFlowLogs`.
    - Configure it to capture all types of traffic, and send all log data to CloudWatch Logs with a maximum aggregation interval of **1 minute**.
    - When you create the flow log, define a new destination log group named `LabVPCFlowLogs`. Have it use the IAM role that you observed in the previous task.



Task 2.3: Access the WebServer instance from the internet and review VPC flow logs in CloudWatch

In this task, you are challenged to use your web browser to test access to the *WebServer* EC2 instance over port 80 (HTTP). You are also challenged to test access to port 22 (SSH) by using the *netcat* command, which will test whether inbound traffic is allowed. Then, you will be challenged to review the VPC flow log to see how these attempts were recorded.

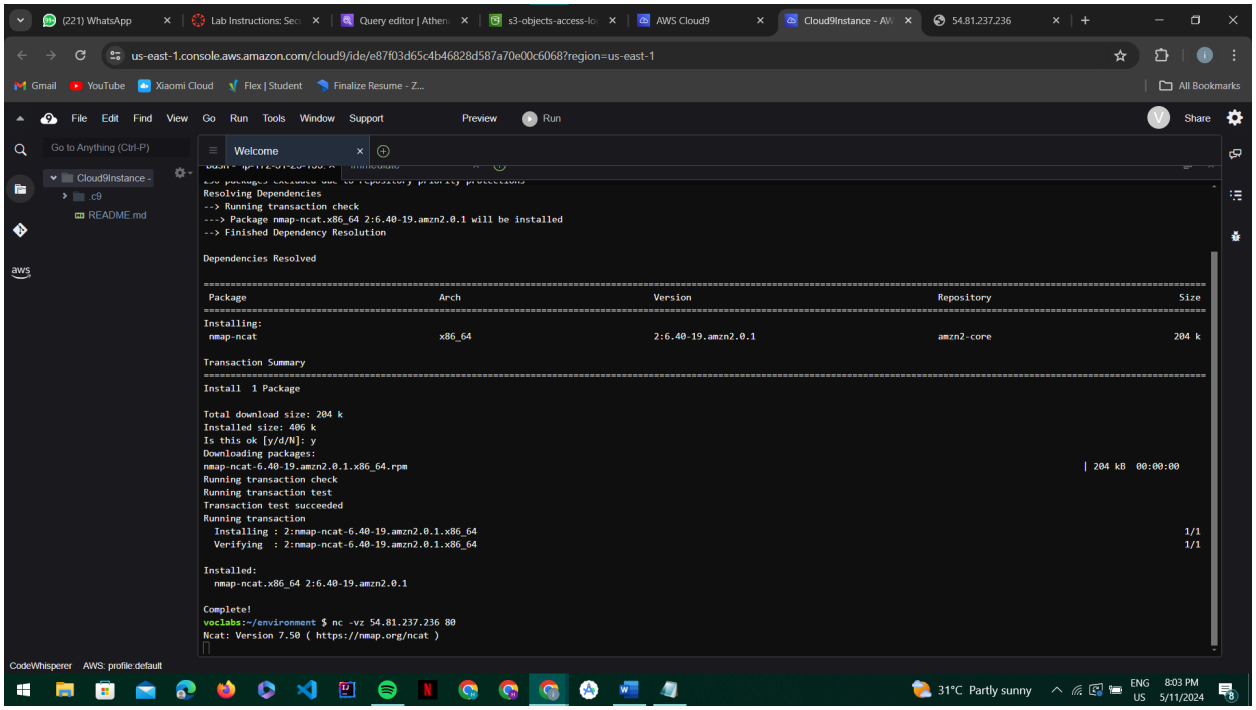
1. In a new browser tab, try to load the WebServer instance's public IP address at `http://<WebServer-public-IP>`
   The page doesn't load. This is expected. Recall that an inexperienced employee of AnyCompany Financial made some mistakes and the result is that the network is not properly configured.



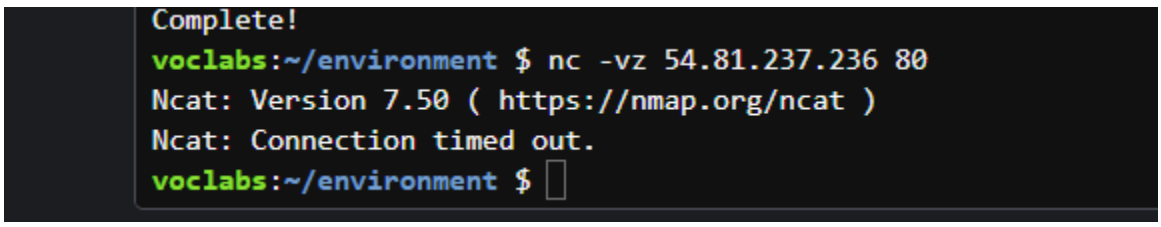
2. Use the AWS Cloud9 **Cloud9Instance** IDE to test access to the *WebServer* instance.
   Note: Because you are testing access to the *public* IPv4 address, your test is effectively happening *from* the internet.
   - To try to access the instance over the standard HTTP port (80), run the following command, replacing the placeholder with the correct value:

     ```
     nc -vz <WebServer-public-IP> 80
     ```

   - After some time passes, the connection fails or times out.



3. Next, try to access the same instance on standard SSH port (22), again using the `nc` command.
   This connection also fails or times out.



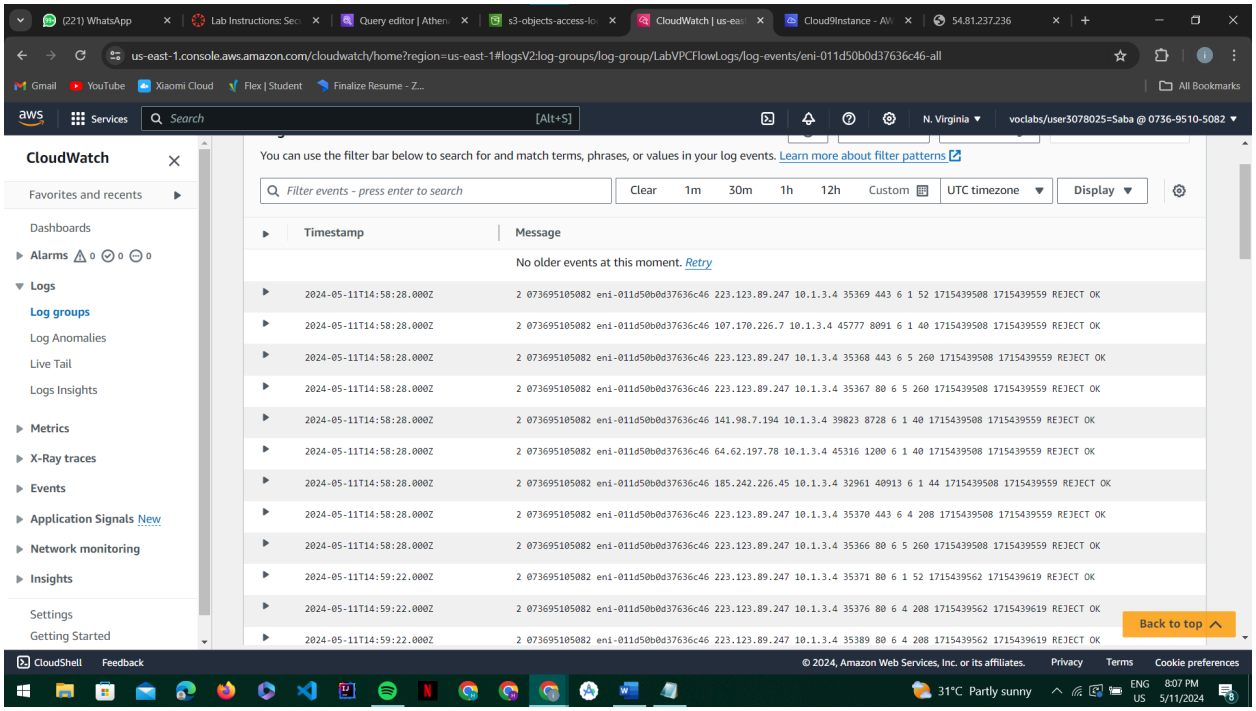
4. In the CloudWatch console, look at the log stream for the *LabVPCFlowLogs* log group to verify logs were generated by your actions in the previous step.
   Notice that the *Message* field for all entries indicates "REJECT".

5. Filter the log entries to display only the events that originated from the public IP address of the AWS Cloud9 instance that you used to run commands earlier in the task.

```
curl http://169.254.169.254/latest/meta-data/public-ipv4
```

The list of log events is filtered to show the netcat scans that you ran against the *WebServer* instance from the AWS Cloud9 terminal. Each log entry indicates the port

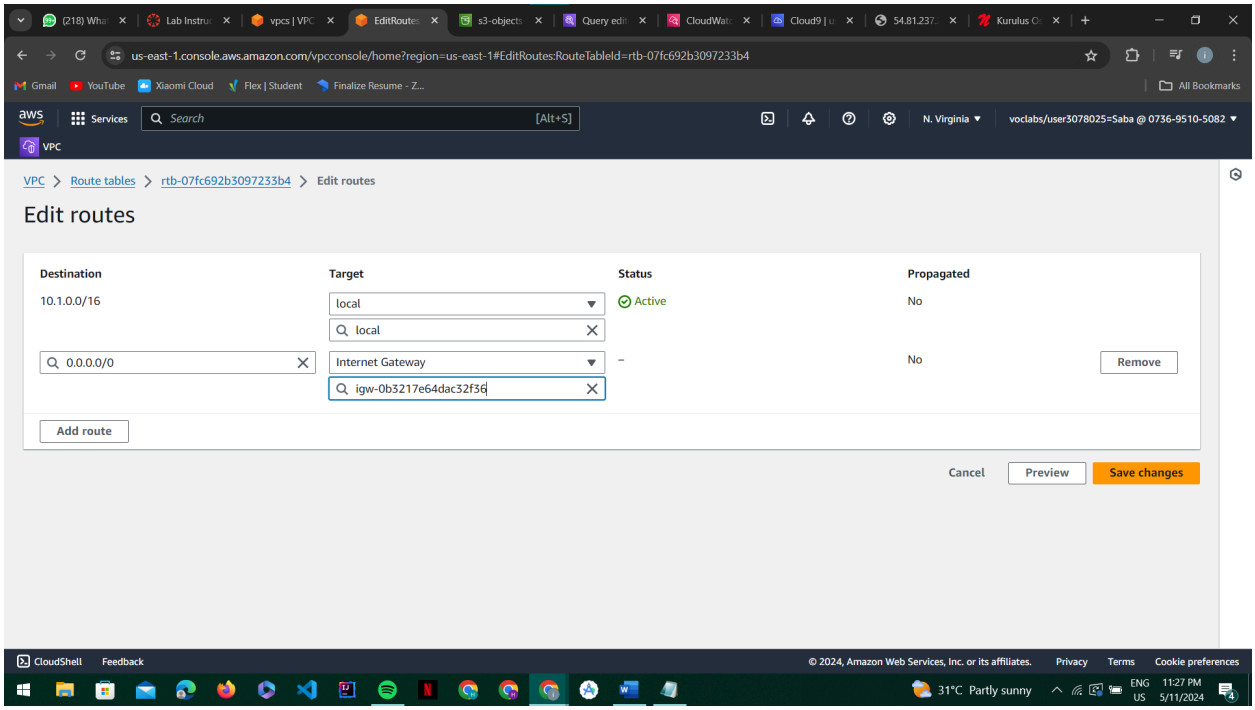
number (for example, 80 or 22) that the attempt was made on. For information about the log format, see Flow Log Record Examples in the *Amazon VPC User Guide*.
Optional: Filter the log entries by the IP address of your computer. You can find your IP address on a website such as www.whatismyip.com. If you filter by this IP address, you will see entries that are related to your attempt to load the webpage that is hosted on the *WebServer* instance.

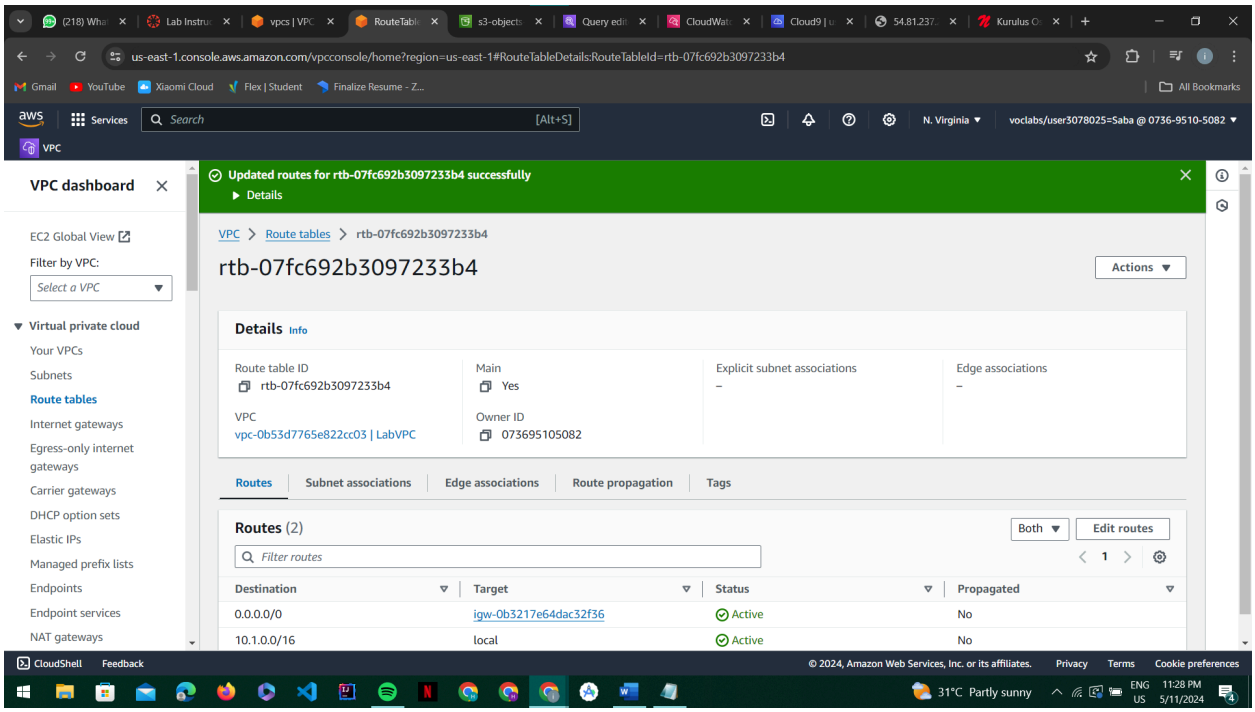


Task 2.4: Configure route table and security group settings

In this task, you are challenged to create a route for traffic from the internet to access the *WebServerSubnet* through an internet gateway. This will allow inbound HTTP traffic to be directed to the *WebServer* instance. You will also be challenged to modify the security group that is associated with the *WebServer* instance to allow inbound traffic on ports 22 (SSH) and 80 (HTTP)

1. Modify the route table that is associated with the *WebServerSubnet*. Add a second route to the route table so that traffic to and from the internet (0.0.0.0/0) is routed to the existing *LabVPCIG* internet gateway.

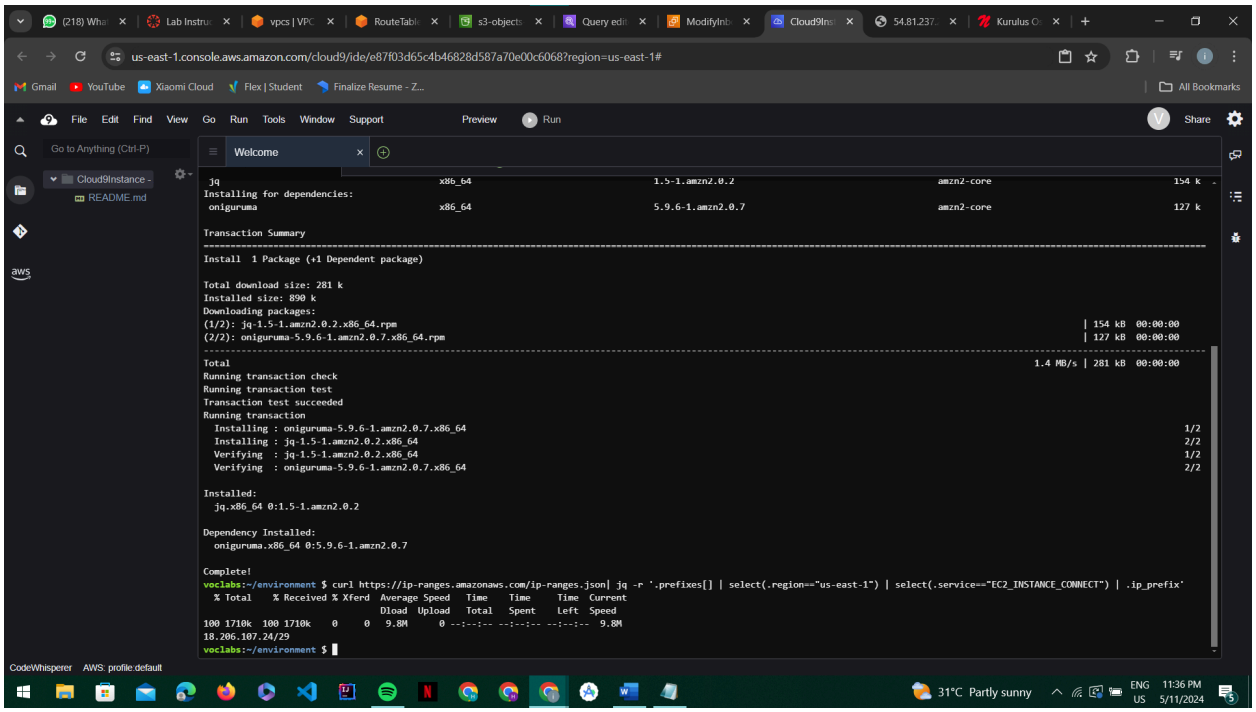


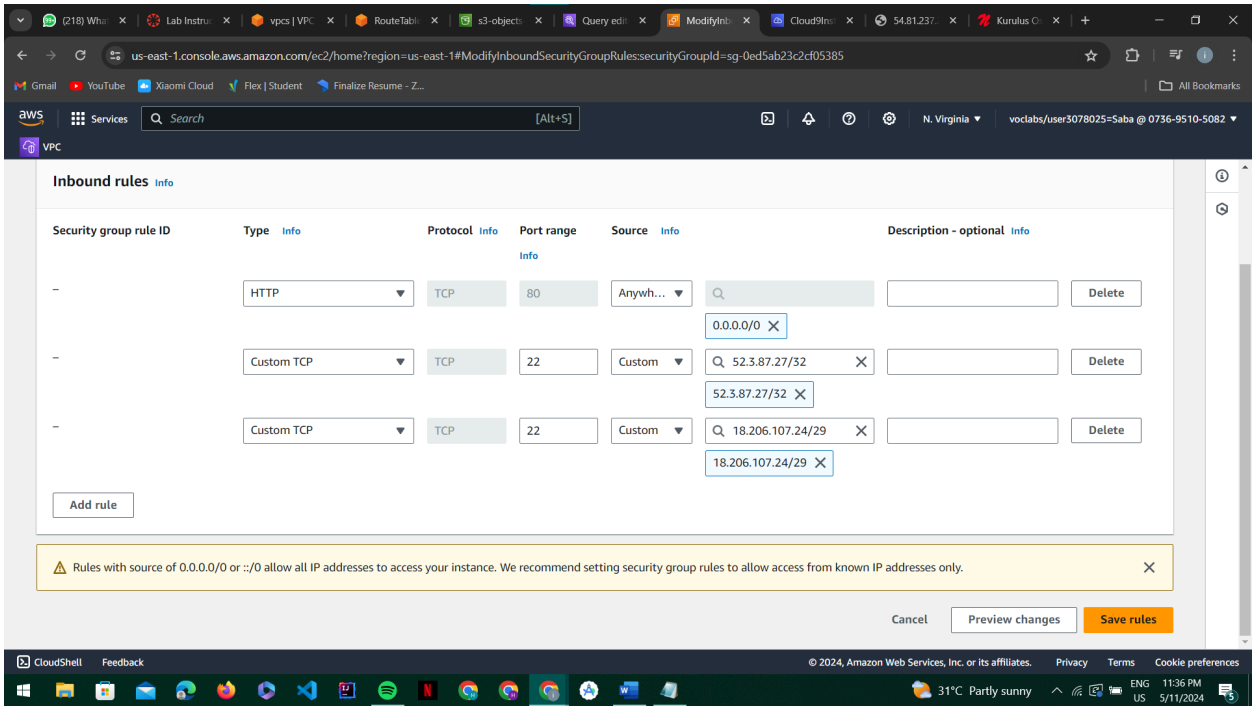


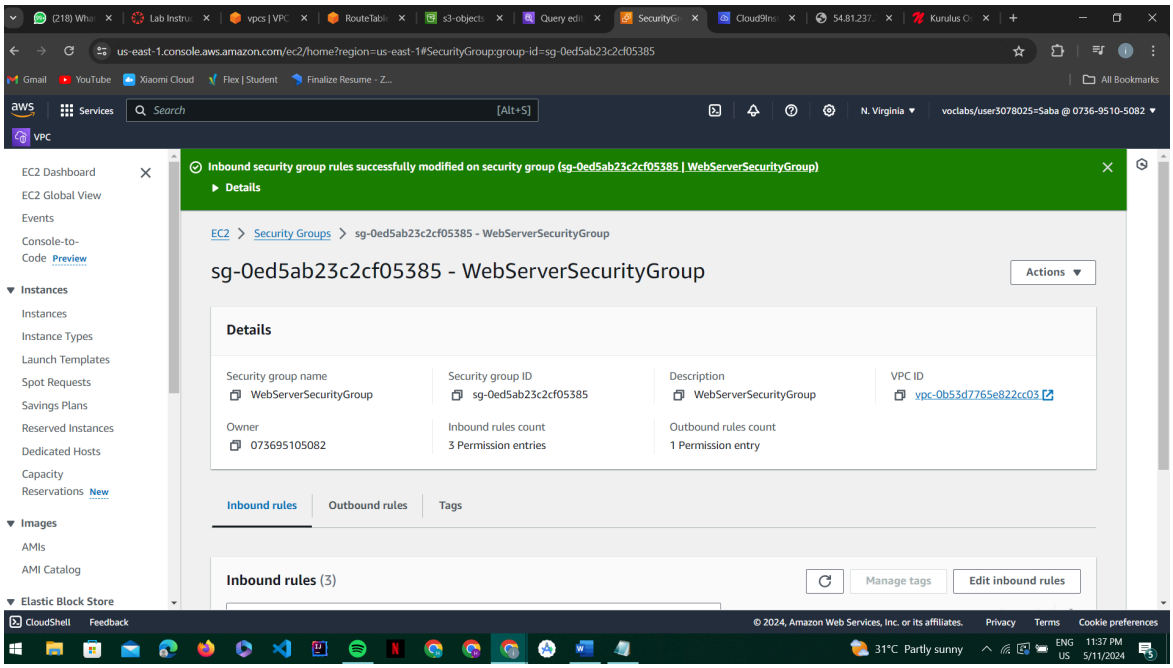
2. Test access from port 80 to the *WebServer* instance again. You can use the `nc` command in the AWS Cloud9 IDE or try to load the following address in a web browser: `http://<WebServer-public-IP>`

The connection times out or fails.

3. Modify the inbound rules for the security group that is associated with the *WebServer* instance so that the instance's webpage will be accessible from the internet and so that certain SSH connections to it will be permitted. To do this:
    - Delete any inbound rules that already exist, unless they allow traffic to ports 22 or 80.
    - Add an inbound rule for HTTP traffic. The rule should allow traffic from *anywhere* to *TCP port 80*.
    - Add an inbound rule to allow SSH access to the *WebServer* instance from the AWS Cloud9 instance. Set the *source* for the rule to match the public IP address of the AWS Cloud9 instance. (You retrieved this IP address in an earlier task.)
    - Add an inbound rule to allow SSH access to the *WebServer* instance so that you can use EC2 Instance Connect to connect to the instance.
      To discover the IP address range to use as the *source* for this rule, run the following commands in your AWS Cloud9 IDE:
    - `sudo yum install -y jq`
      `curl https://ip-ranges.amazonaws.com/ip-ranges.json| jq -r '.prefixes[] | select(.region=="us-east-1") | select(.service=="EC2_INSTANCE_CONNECT") | .ip_prefix'`



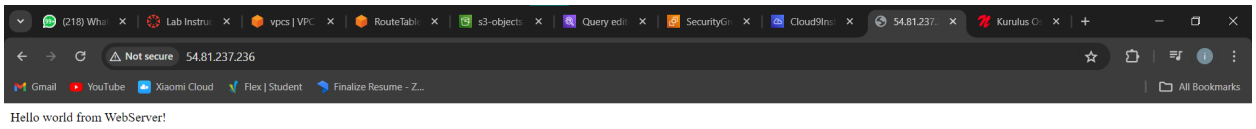




You should now have three inbound rules defined for this security group.

4. Use the `nc` command to test access from port 22 to the *WebServer* instance again.
   The test should be successful.

5. Use your web browser to test access from port 80 to the webpage on the *WebServer* instance.
   The test should be successful. The message "Hello world from WebServer!" displays.



Hello world from WebServer!

6. Use EC2 Instance Connect to access the *WebServer* instance.
   After you connect, run the following command: `ping -c 3 www.amazon.com`
   This command tests whether the instance can access the internet. (In this case, you test whether the instance can access the public Amazon.com website.). If you receive 64 bytes of data in return *three* times, then the test was successful, and the *WebServer* instance can access the internet.
   Congratulations! You were successfully able to troubleshoot network connections in this task.

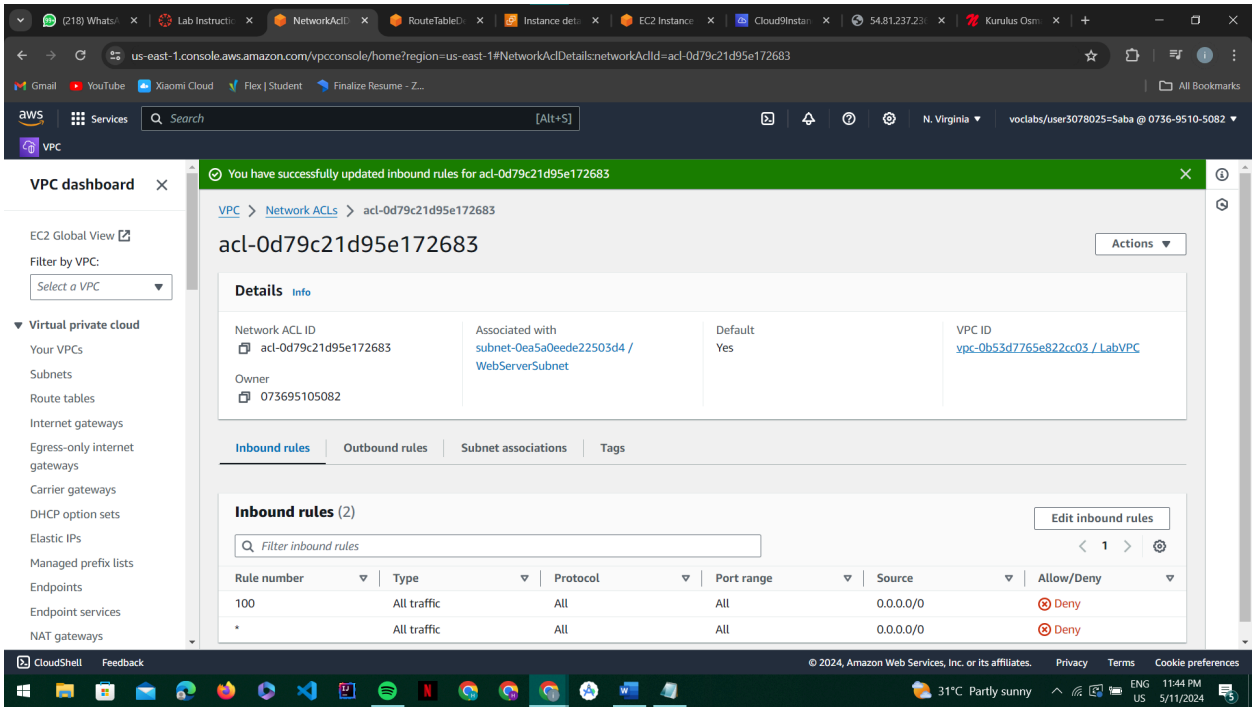
```
[ec2-user@webserver ~]$ ping -c 3 www.amazon.com
PING d3ag4hukkh62yn.cloudfront.net (18.154.233.37) 56(84) bytes of data.
64 bytes from server-18-154-233-37.iad55.r.cloudfront.net (18.154.233.37): icmp_seq=1 ttl=249 time=1.26 ms
64 bytes from server-18-154-233-37.iad55.r.cloudfront.net (18.154.233.37): icmp_seq=2 ttl=249 time=1.11 ms
64 bytes from server-18-154-233-37.iad55.r.cloudfront.net (18.154.233.37): icmp_seq=3 ttl=249 time=1.12 ms

--- d3ag4hukkh62yn.cloudfront.net ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 1.111/1.164/1.261/0.068 ms
[ec2-user@webserver ~]$
```

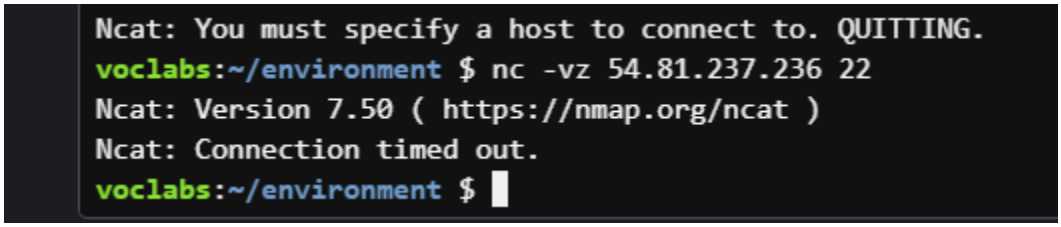
Task 2.5: Secure the WebServerSubnet with a network ACL

In this task, you are challenged to configure a network access control list (ACL) to secure the subnet where the web server is running. The network ACL will provide an additional layer of security beyond the security group that you already configured.

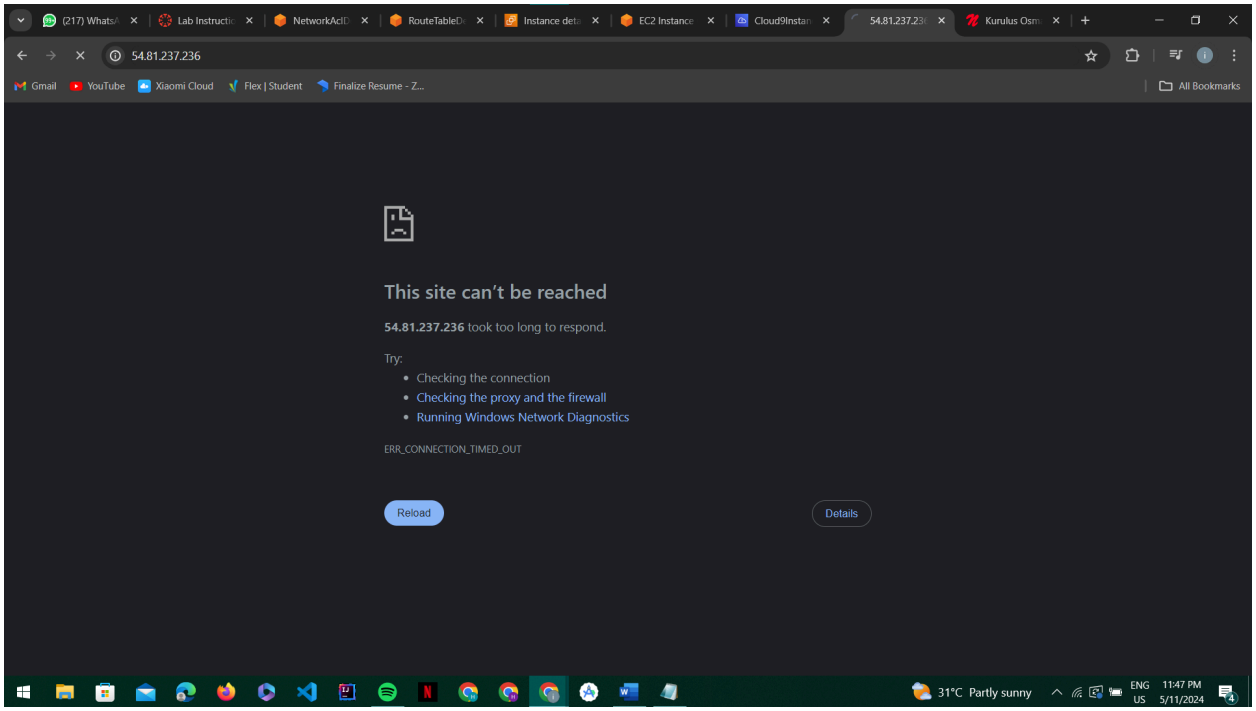
1. In the Amazon VPC console, navigate to the details page for the network ACL that is associated with the *WebServerSubnet* in *LabVPC*.



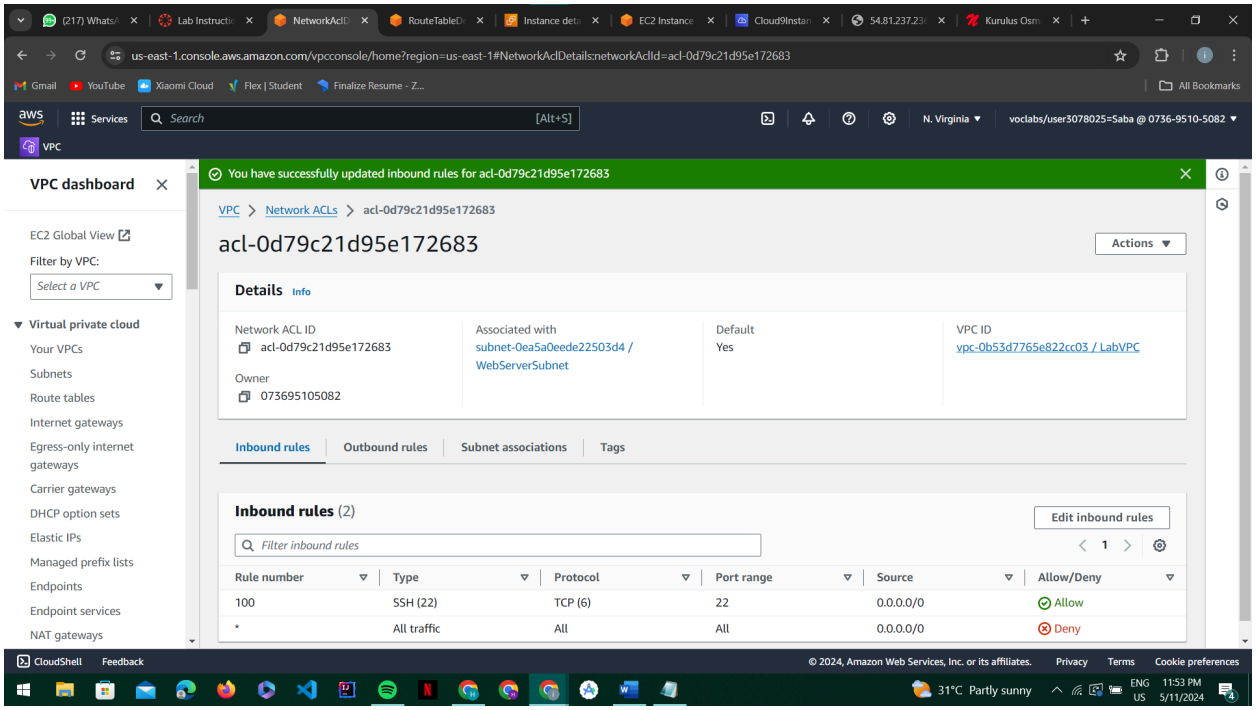
2. To test whether you can access port 22 on the *WebServer* instance, modify the *100* rule from *Allow* to *Deny*. Then, run the `nc` command.
   The connection fails or times out, which indicates that access is no longer allowed. Recall that this test *was successful* in step 4 of the previous task.

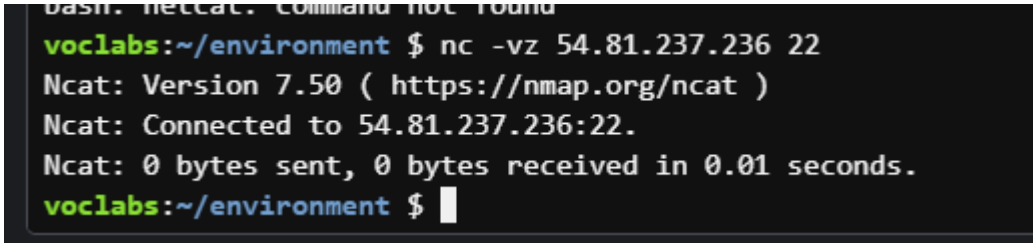


3. Test whether you can access the webpage at `http://<WebServer-public-IP>` where `<WebServer-public-IP>` is the public IPv4 address of the *WebServer* instance.
   It should fail for the same reason.

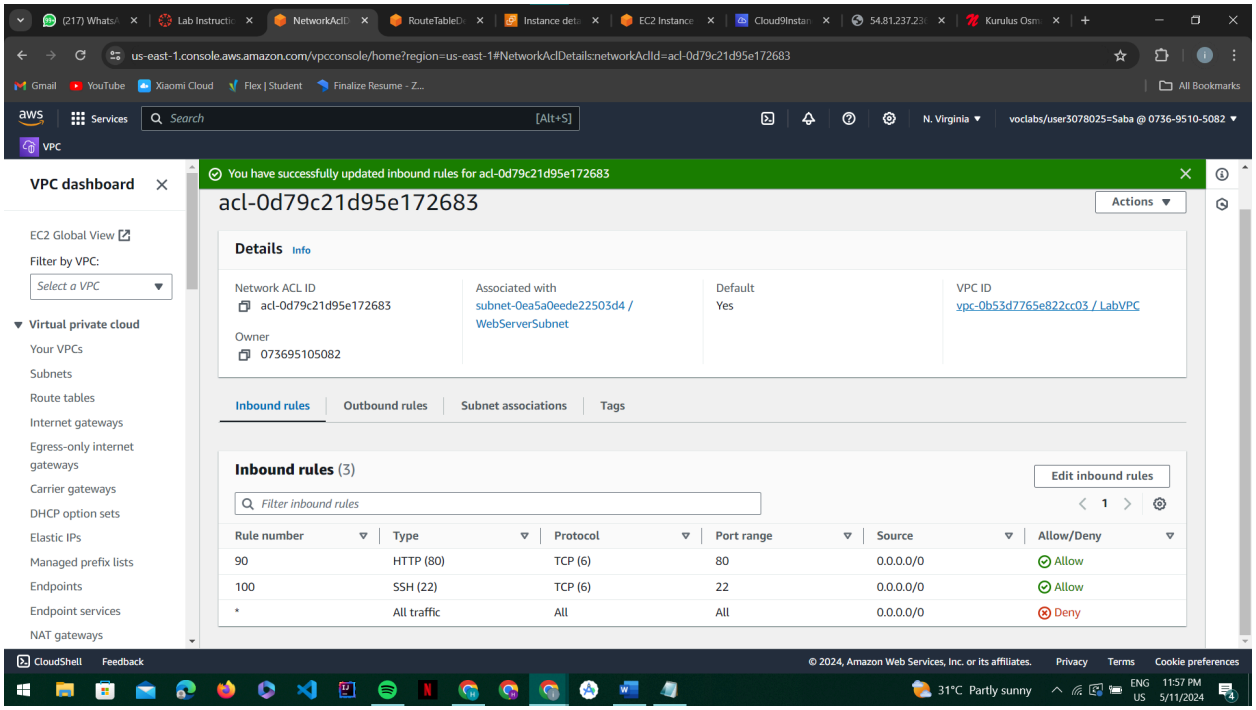


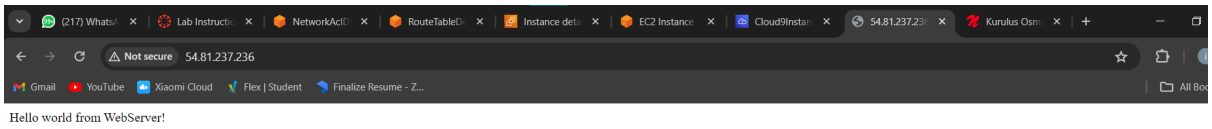
4. Modify rule number 100 to *allow* connections on port 22 only. Then, use netcat to confirm whether you can access port 22.
   The test should be successful.





5. Edit the inbound rules for the network ACL again. Add a *new* rule that allows *HTTP* traffic from *anywhere*. Use *90* as the rule number.
   Then, confirm whether HTTP traffic is allowed by trying to access the webpage at `http://<WebServer-public-IP>` where `<WebServer-public-IP>` is the public IPv4 address of the *WebServer* instance.
   You should see the "Hello world from WebServer!" message.



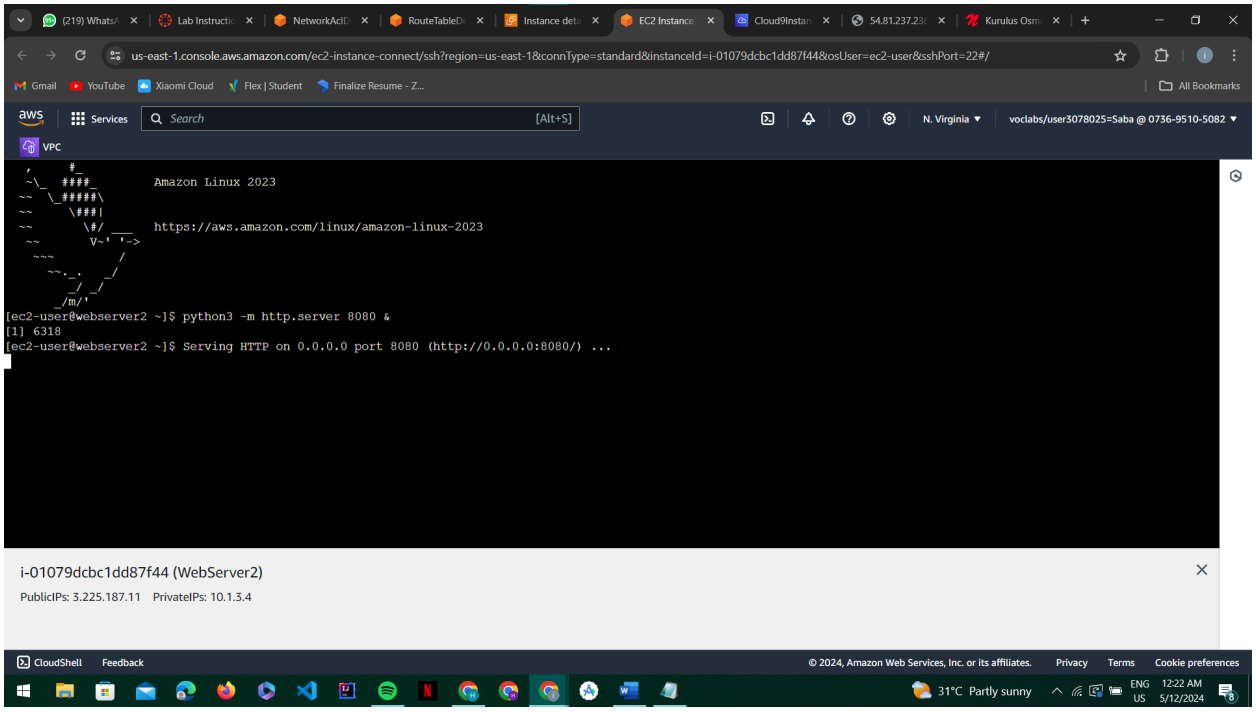


Task 2.6: Review NetworkFirewallVPC and its associated resources

In this phase so far, you worked to secure *LabVPC* by updating a route table, network ACL, and security group.

1. Observe the existing NetworkFirewallVPC resources and configurations.
    - In the VPC console select **NetworkFirewallVPC**, and choose the **Resource map** tab. Notice that the VPC contains the following:
        - A *WebServer2Subnet* subnet. (*LabVPC* contained a subnet named *WebServerSubnet*.)

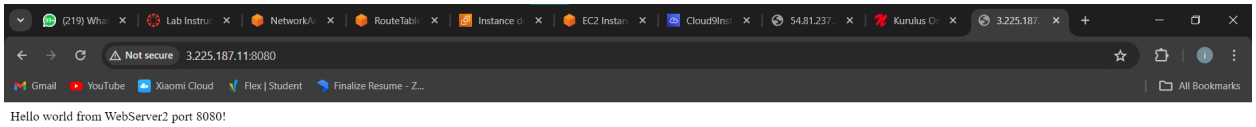
        - A second subnet, named *FirewallSubnet*. (*LabVPC* had only one subnet).
        - An internet gateway, named *NetworkFirewallIG*. The default (main) route table is already configured to route traffic between *WebServer2Subnet* and the internet.
    - Still in the Amazon VPC console, observe the network ACL settings for *NetworkFirewallVPC*.
      The default inbound rules exist. Rule 100 allows all traffic from all sources.
    - In the Amazon EC2 console, observe the settings for the *WebServer2* instance and copy the public IPv4 address to use in the next step.
      Observe that this instance runs in *WebServer2Subnet*.
      Notice that inbound traffic is allowed from anywhere (0.0.0.0/0) to ports 80 (HTTP), 22 (SSH), and 8080.



2. Confirm access to the WebServer2 instance on ports 80 and 22.
    - Verify HTTP access using a browser.
      You should see the "Hello world from WebServer2!" message.
    - Verify SSH access by using the `nc` command in the AWS Cloud9 terminal.
      The attempt should be successful.

3. Start an additional website that runs on *WebServer2* port 8080 and test access.
    - Use EC2 Instance Connect to connect to the *WebServer2* instance.

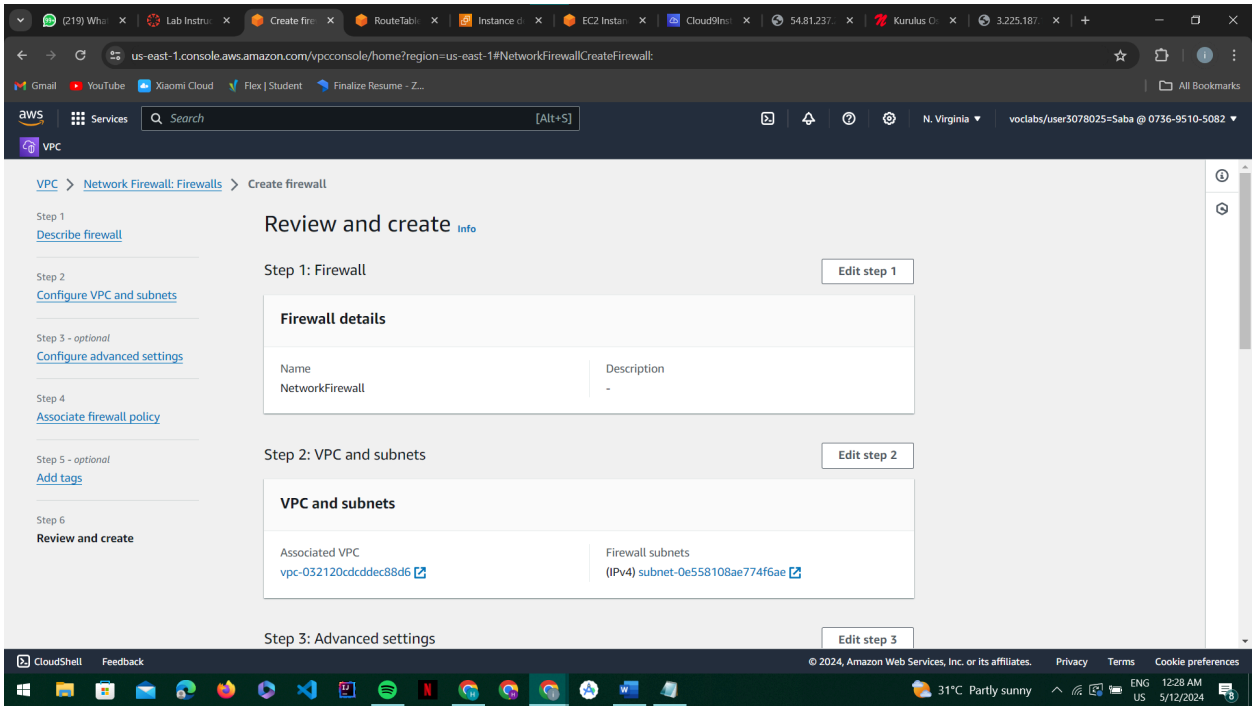
- After you connect, run the following command.
- `python3 -m http.server 8080 &`
- After running the command, press Enter to return to the prompt. Keep the EC2 Instance Connect session open.
- In a separate browser tab, try to load `http://<WebServer2-public-IP>:8080` The attempt should be successful, and you should see the "Hello world from WebServer2 port 8080!" message.

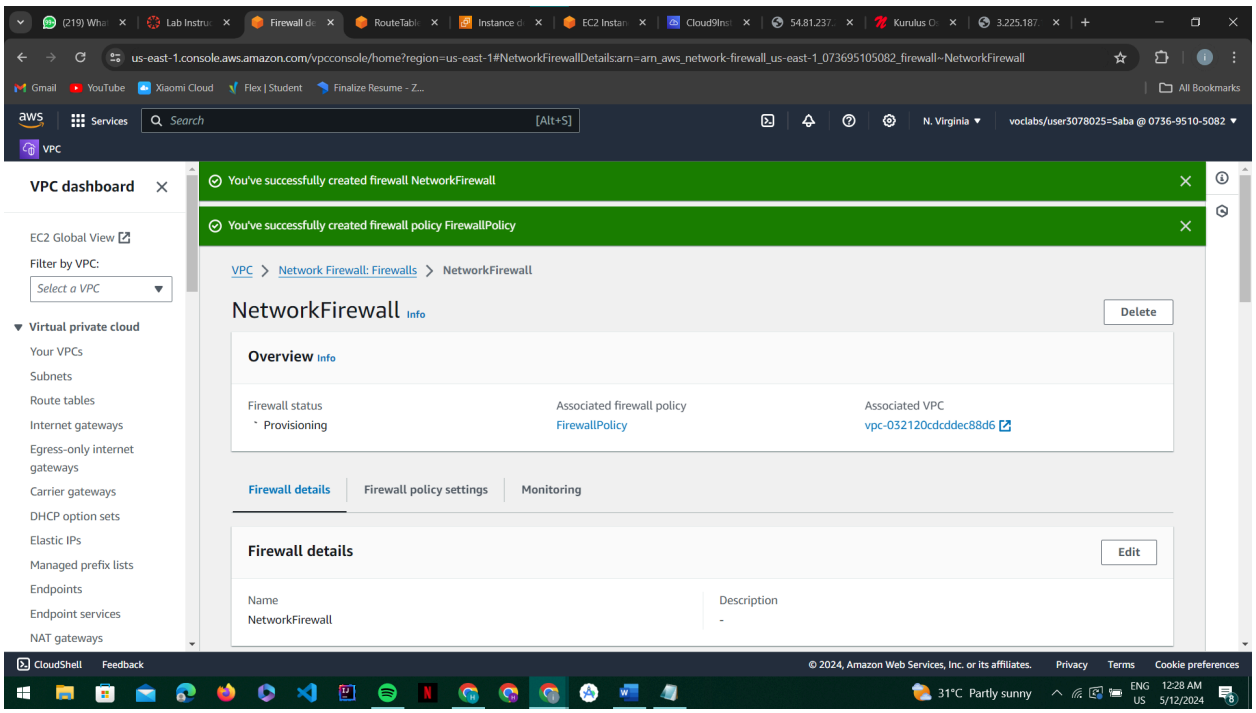


Task 2.7: Create a network firewall

In this task, you are challenged to create a network firewall for the *NetworkFirewallVPC*, which you haven't yet used in this project.

1. In the Amazon VPC console, create a firewall named `NetworkFirewall`.
   - Associate it with *NetworkFirewallVPC*. Create the firewall in the *us-east-1a* Availability Zone for *FirewallSubnet* and IP address type *IPv4*.
   - Name the firewall policy `FirewallPolicy`
   - Under **Delete protection** remove the check for **Enable**. Also for **Subnet change protection** remove the check for **Enable**.
   - Wait for the firewall to attain the status of *Ready* before you continue to the next step. The process takes approximately 3 minutes to complete.

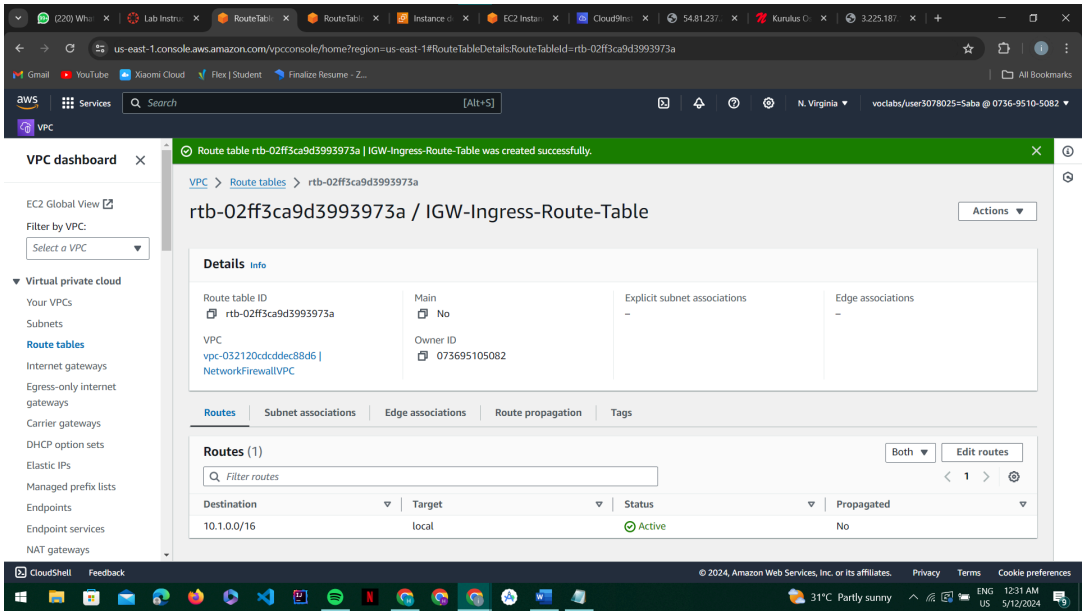

us-east-1.console.aws.amazon.com/vpcconsole/home?region=us-east-1#NetworkFirewallCreateFirewall:
VPC
VPC > Network Firewall: Firewalls > Create firewall
Step 1
Describe firewall
Step 2
Configure VPC and subnets
Step 3 - optional
Configure advanced settings
Step 4
Associate firewall policy
Step 5 - optional
Add tags
Step 6
Review and create
Review and create Info
Step 1: Firewall
Edit step 1
Firewall details
Name
NetworkFirewall
Description
-
Step 2: VPC and subnets
Edit step 2
VPC and subnets
Associated VPC
vpc-032120cdcddec88d6
Firewall subnets
(IPv4) subnet-0e558108ae774f6ae
Step 3: Advanced settings
Edit step 3
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


us-east-1.console.aws.amazon.com/vpcconsole/home?region=us-east-1#NetworkFirewallDetails:arn=arn_aws_network-firewall_us-east-1_073695105082_firewall~NetworkFirewall
VPC dashboard
EC2 Global View
Filter by VPC:
Select a VPC
Virtual private cloud
Your VPCs
Subnets
Route tables
Internet gateways
Egress-only internet gateways
Carrier gateways
DHCP option sets
Elastic IPs
Managed prefix lists
Endpoints
Endpoint services
NAT gateways
You've successfully created firewall NetworkFirewall
You've successfully created firewall policy FirewallPolicy
VPC > Network Firewall: Firewalls > NetworkFirewall
NetworkFirewall Info
Delete
Overview Info
Firewall status
Provisioning
Associated firewall policy
FirewallPolicy
Associated VPC
vpc-032120cdcddec88d6
Firewall details
Firewall policy settings
Monitoring
Firewall details
Edit
Name
NetworkFirewall
Description
-
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

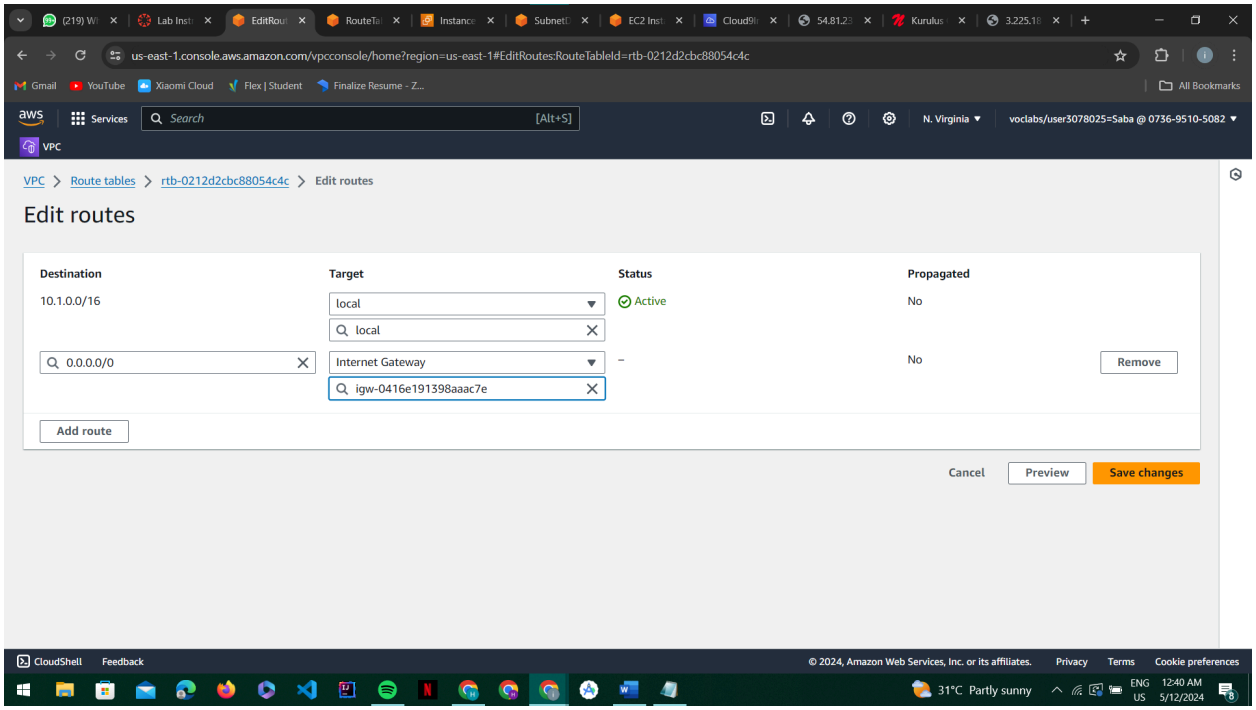
Task 2.8: Create route tables

In this task, you are challenged to create and configure three new route tables, including one for each subnet in the *NetworkFirewallVPC* and one to handle inbound (ingress) traffic for the internet gateway in *NetworkFirewallVPC*.

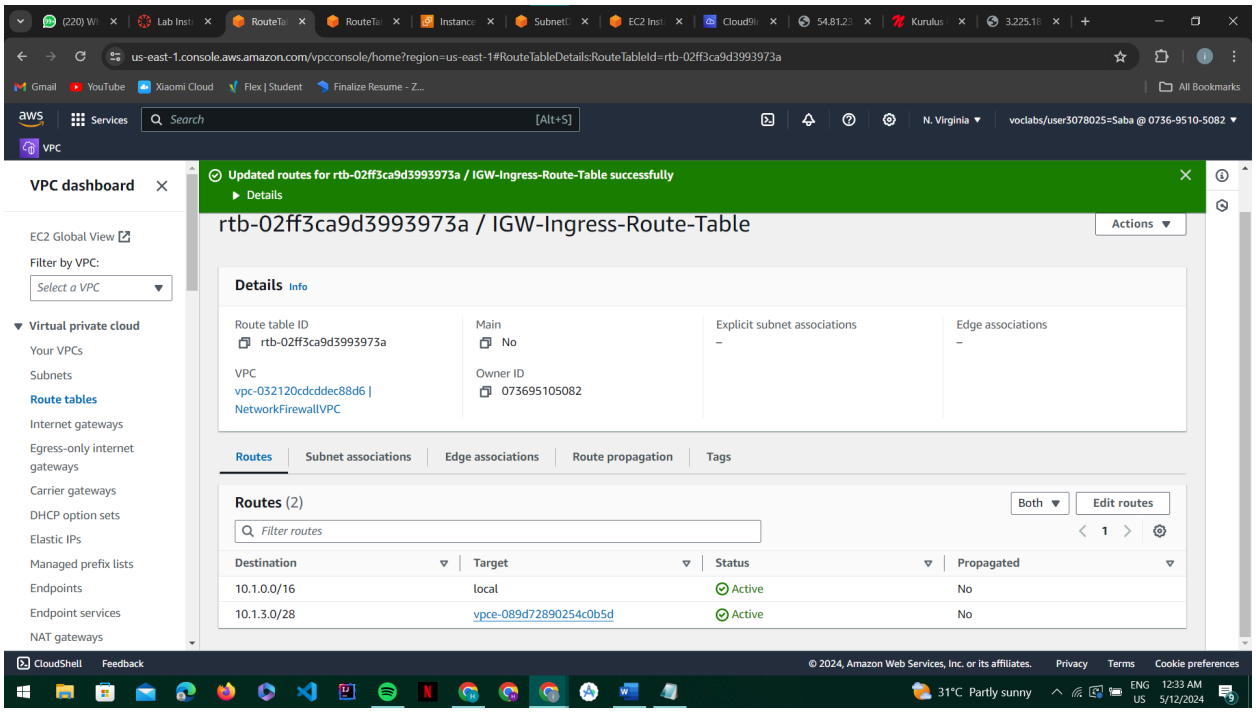
1. Create a new route table named `IGW-Ingress-Route-Table` in the *NetworkFirewallVPC*.



2. Edit the route table to add a new route.
    - Set the *destination* to be the CIDR block of the subnet that the *WebServer2* instance runs in.
    - Set the *target* to the only *Gateway Load Balancer Endpoint* that is available.

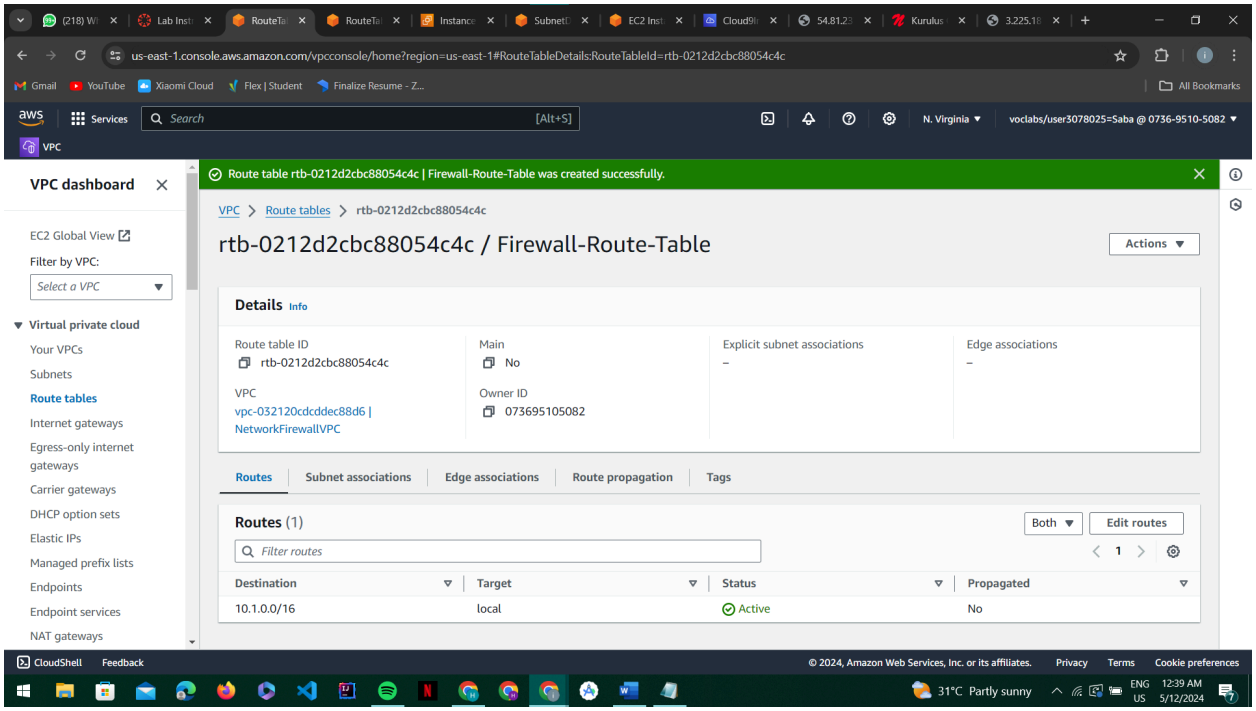


3. Add an edge association to the *IGW-Ingress-Route-Table* so that the *NetworkFirewallIG* internet gateway is associated with the route table.

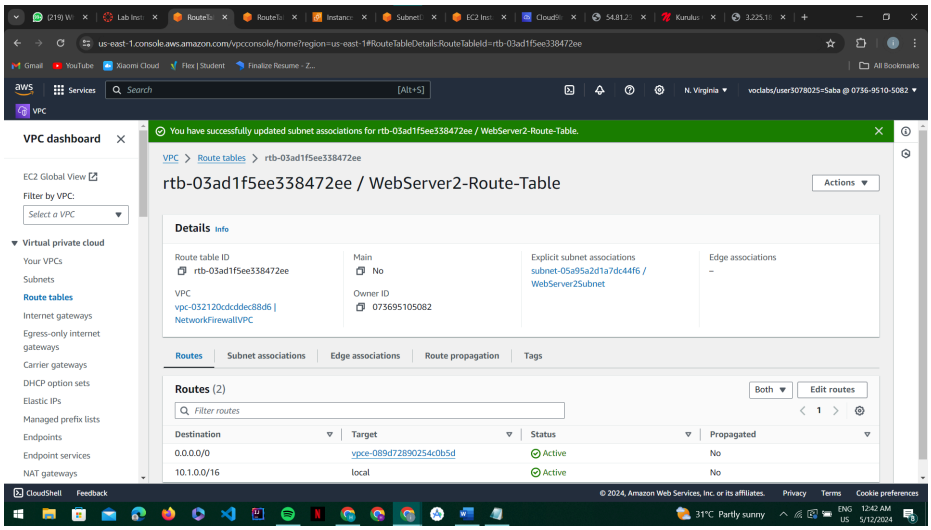


4. Create another route table in *NetworkFirewallVPC* for the *FirewallSubnet*.

- Name the route table `Firewall-Route-Table`
- Add a route so that `0.0.0.0/0` traffic is routed to the *NetworkFirewallIG*.
- Create an explicit association between the *FirewallSubnet* subnet and the route table.



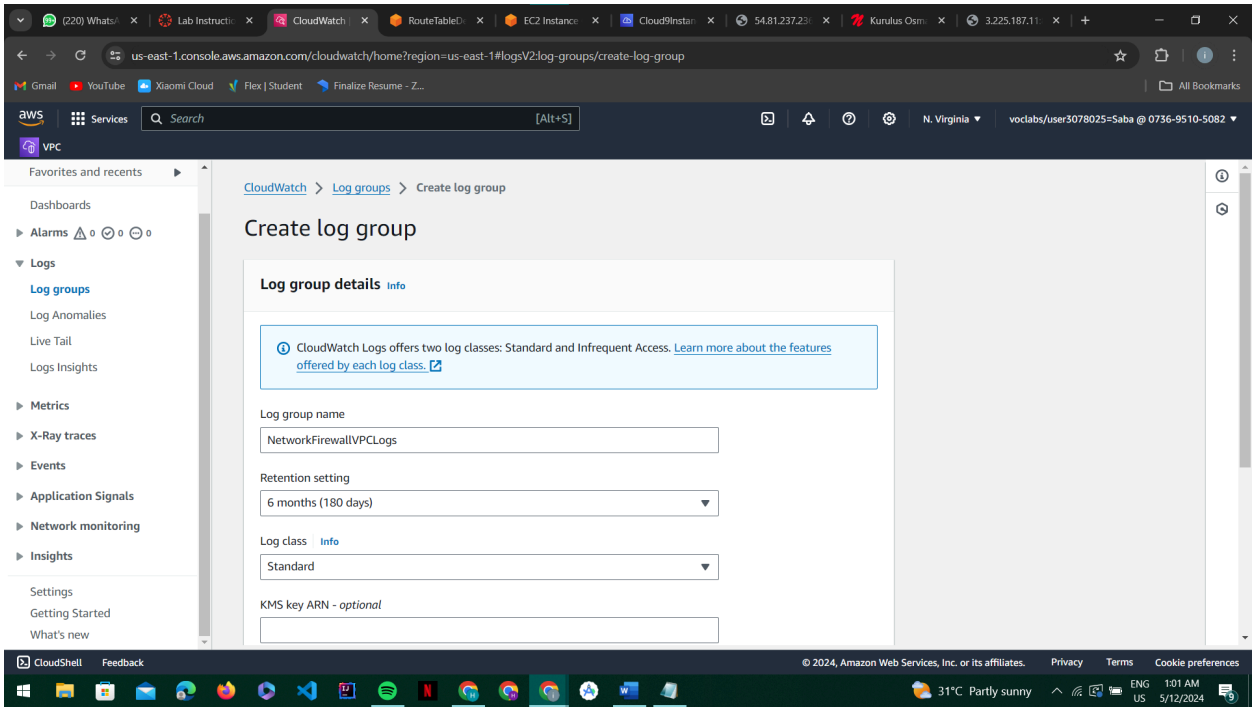
5. Create another route table in *NetworkFirewallVPC* for the *Webserver2Subnet*.
   - Name the route table `WebServer2-Route-Table`
   - Add a route so that `0.0.0.0/0` traffic is routed to the *Gateway Load Balancer Endpoint*.



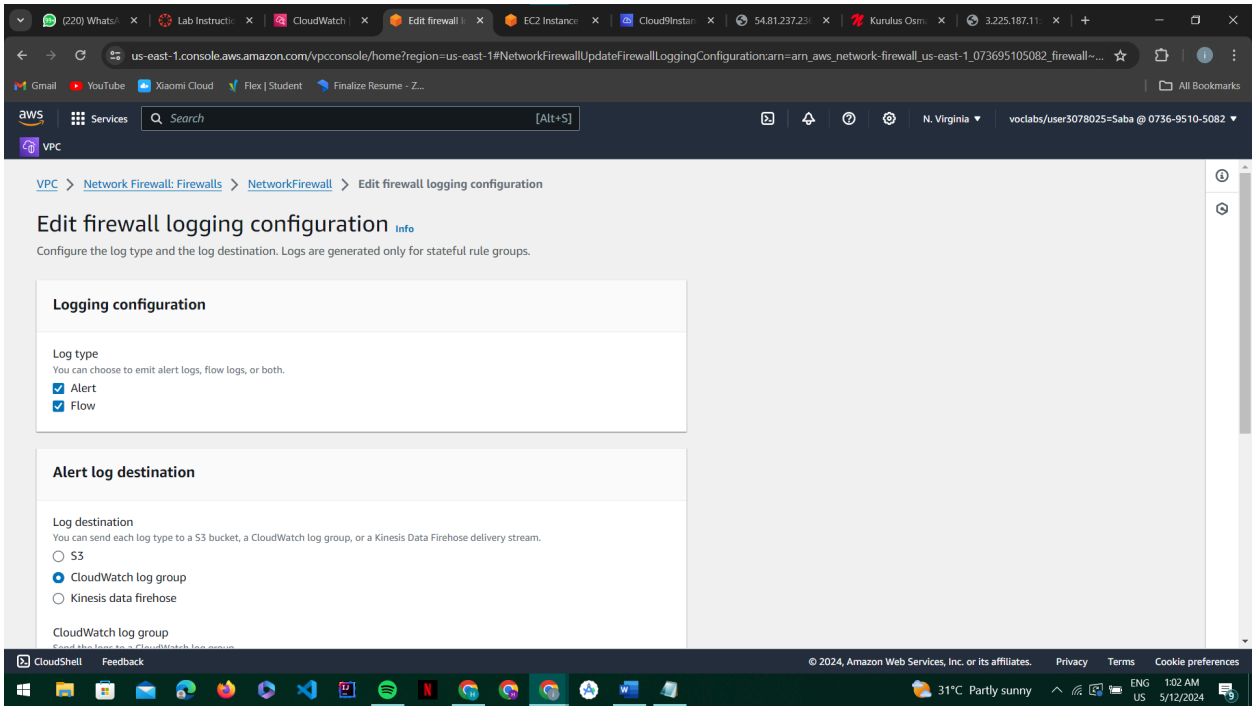
Task 2.9: Configure logging for the network firewall

In this task, you are challenged to configure logging for the network firewall so that you can analyze details of network traffic requests.

1. Create a CloudWatch log group named `NetworkFirewallVPCLogs` with a retention setting of *6 months*.

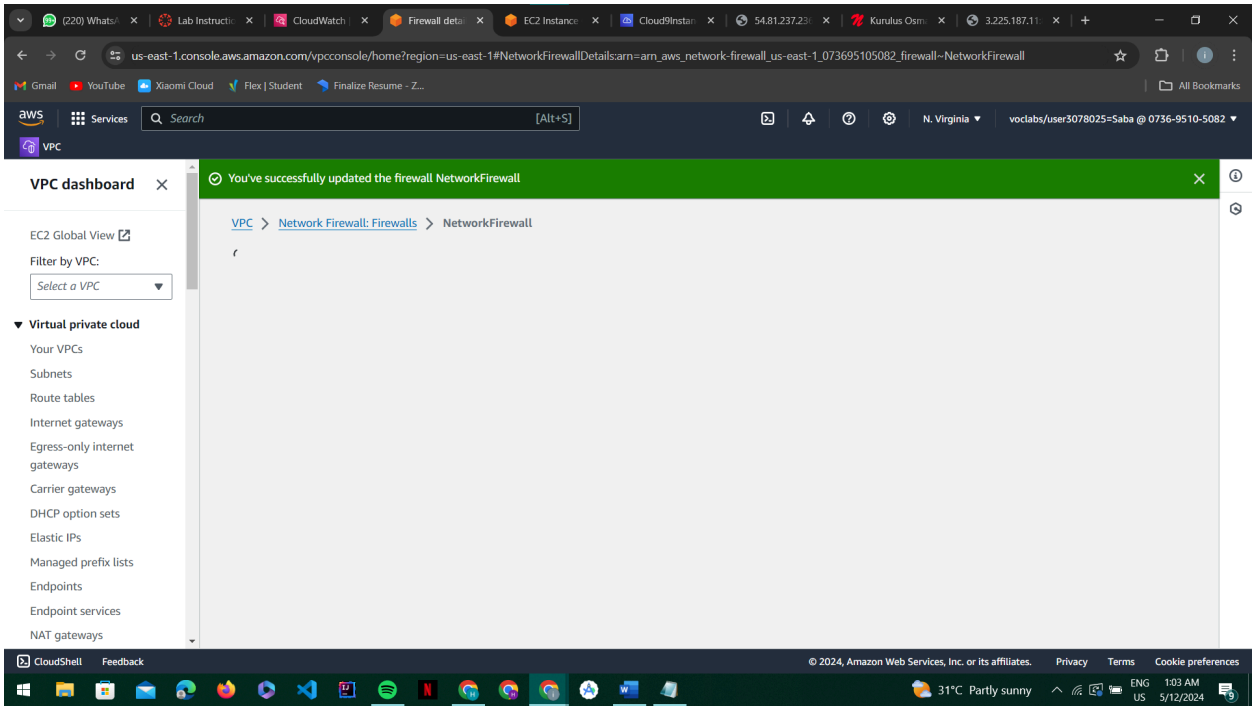


2. In the settings for the *NetworkFirewall* that you created, browse to the **Firewall details** area, and configure both **Alert** and **Flow** type logging. Set the log destination for both types of logging to use the *NetworkFirewallVPCLogs* CloudWatch log group.



3. Test the logging configuration by attempting to load *WebServer2* instance's website on port 80 in a new browser tab.

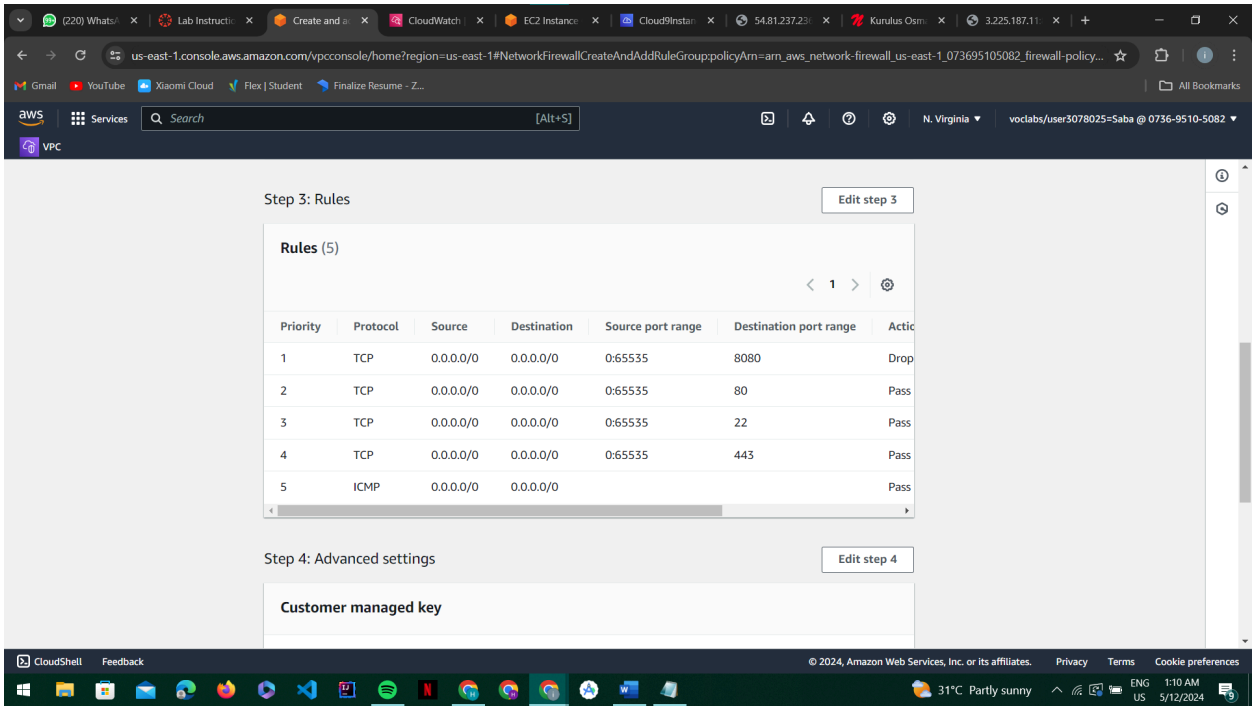
4. In the CloudWatch console, look for log events in the *NetworkFirewallVPCLogs* log group. Filter the log events by your public IP address to find the log entries that resulted from the test that you ran in the previous step.



Task 2.10: Configure the firewall policy and test access

When you first created the network firewall, you defined an empty policy. By default, the empty policy blocks all traffic, which is why your attempt to load the *WebServer2* website in the previous task failed.

1. Navigate to the details page for the *NetworkFirewall* and begin to create the rule group.
    - In the navigation pane of the Amazon VPC console, choose **Firewalls**.
    - Choose the name link for **NetworkFirewall**.
    - In the **Overview** section at the top of the page, under **Step 2**, choose **Add rule groups** > **Create stateful rule group**.

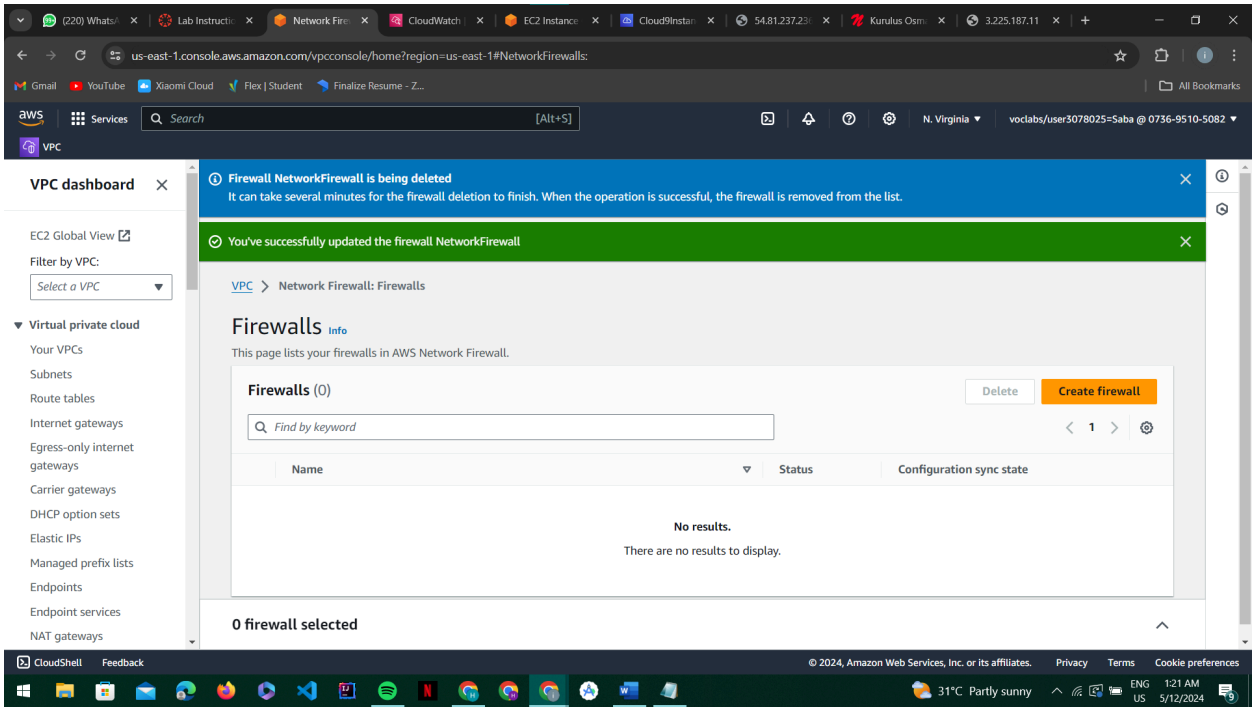


2. Configure the stateful rule group with the name `NetworkFirewallVPCRuleGroup` and a capacity of *100*. Use the other default settings.
    - Configure the rule group to have five rules with the following requirements:
        - First rule: Use the *TCP* protocol. Set the destination port to *8080*, and set the action to *Drop*.
        - Second rule: Use the *TCP* protocol. Set the destination port to *80*, and set the action to *Pass*.
        - Third rule: Use the *TCP* protocol. Set the destination port to *22*, and set the action to *Pass*.
        - Fourth rule: Use the *TCP* protocol. Set the destination port to *443*, and set the action to *Pass*.
        - Fifth rule: Use the *ICMP* protocol. Set the destination port to *any*, and set the action to *Pass*.
    - After you configure the rules, choose **Create and add to policy**.

3. Test the firewall rules.
    - In a new browser tab, load the webpage hosted on WebServer2 using the public IP address.
      The attempt should succeed because the firewall policy now contains a rule that allows traffic on TCP port 80.
    - In the AWS Cloud9 IDE, use the netcat command to test SSH (port 22) access to *WebServer2*.
      The attempt should succeed.

- Use EC2 Instance Connect to connect to *WebServer2*. The attempt should succeed. After you connect, run the following commands in the session:
- `ping -c 3 www.amazon.com`
  `sudo netstat -tulpn | grep -i listen`
-

4. In the CloudWatch console, observe the network firewall log entries that were created by your tests in the previous step.
   Congratulations! You successfully implemented AWS Network Firewall to secure a VPC!
   Note: Make sure to remove all resources associated with this section.
   - Remove edge association on IGW-Ingress-RouteTable
   - Delete IGW-Ingress-RouteTable
   - Remove subnet association on Firewall-Route-Table
   - Delete Firewall-Route-Table
   - Remove subnet association on WebServer2-Route-Table
   - Delete WebServer2-Route-Table
   - Remove logging config from the NetworkFirewall
   - Delete NetworkFirewall
   - Delete firewall policy
   - Delete firewall rule group
   - Delete CloudWatch logs (optional)



Cost estimate to secure a VPC with a network firewall

AnyCompany Financial has asked you to provide an estimate of the cost to implement a network firewall in the us-east-1 Region. The company has provided the following information about the planned usage.

Criteria	Estimate per month
EC2 - WebServer2	1 On-demand (Shared Instance, 100% usage)
VPC - NetworkFirewallVPC	1 IPAM
Inbound data transfer from (internet free)	100 GB
Outbound data transfer to (US East, N. Virginia) - outbound not Intra-Region	1 TB
Network firewall - number of endpoints	1
Network firewall - usage per endpoint	30 days
Network firewall - data processed per month	2.0 TB

Use the AWS Pricing Calculator and the information from the previous table to create a detailed pricing estimate for the network firewall.

After you create the estimate, export it:

- Choose **Export**, and then choose to export as a .csv or .pdf file.
- When prompted, choose **OK**, and download the file.
- If required by your educator, add information about the cost estimate to your presentation.

Phase 3: Securing AWS resources by using AWS KMS

The legal department at AnyCompany Financial received notice from the U.S. Federal Deposit Insurance Corporation (FDIC) that the company needs to encrypt sensitive information, such as PII, credit card numbers, and social security numbers. The legal department contacted your manager, the director of IT, and they tasked you to implement encryption and tokenization to meet regulatory compliance standards.

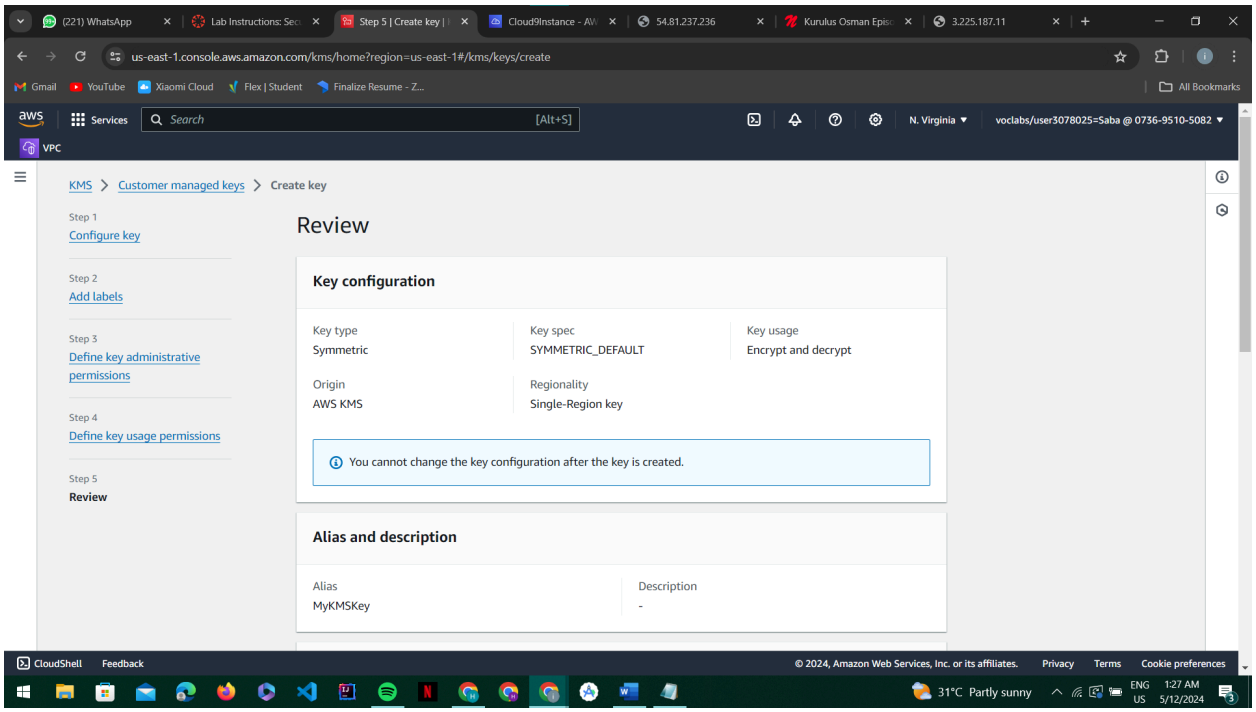
Specifically, the company wants to do the following:

- Implement a mechanism to create and manage AWS KMS customer managed keys.
- Use the managed key solution to encrypt data that is stored in Amazon S3 and on the EBS root volume of an EC2 instance.
- Use the customer managed keys to encrypt secrets that are stored in Secrets Manager.

Task 3.1: Create a customer managed key and configure key rotation

In this task, you will create an AWS KMS customer managed key. You will then configure automatic key rotation on the key. In later tasks, you will be challenged to use this key to protect data that is stored in your account.

1. Create an AWS KMS customer managed key with the alias (name) of `MyKMSKey`. Keep the default settings *except* grant the *Key administrator* and *Key user* permissions to the *voclabs* role.



2. Configure AWS KMS key rotation on the new key so that it is automatically rotated every year.

Task 3.2: Update the AWS KMS key policy and analyze an IAM policy

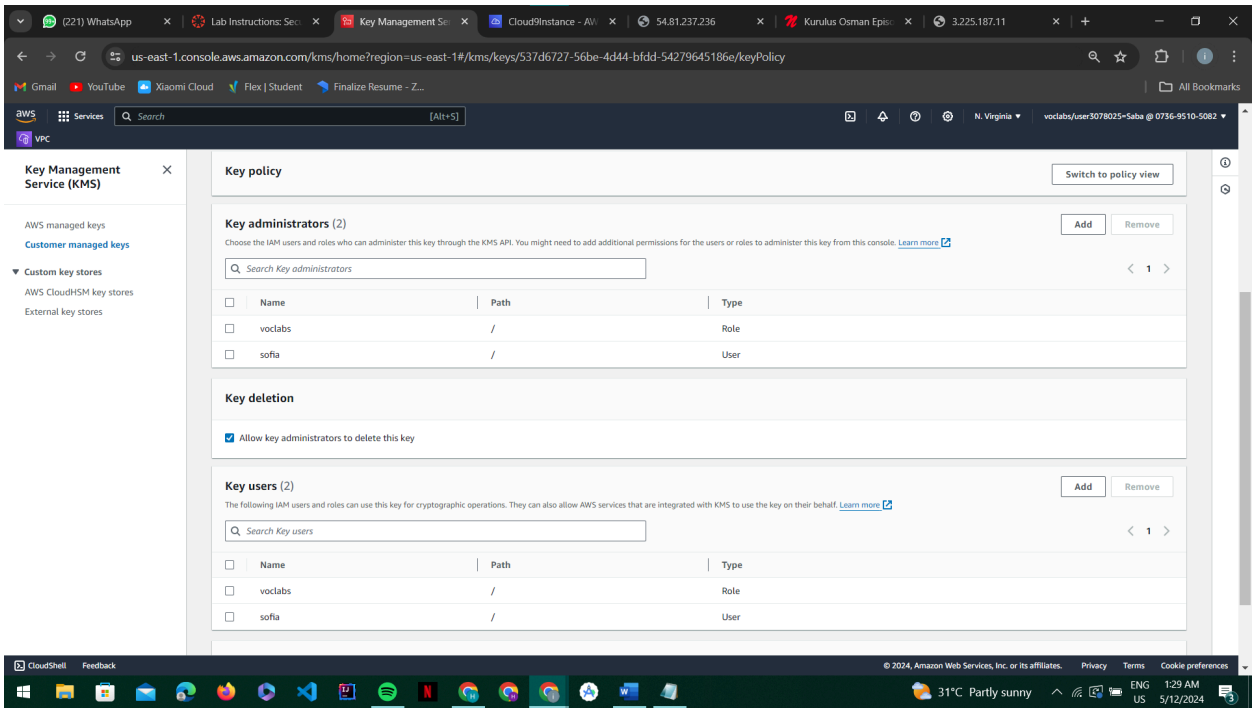
Return to table of contents

In this task, you are challenged to modify the policy of the AWS KMS key that you created so that the *sofia* user will be authorized to use the key. You are also challenged to analyze the IAM policy that controls what the *sofia* user can do in the AWS account.

1. Modify the key policy for *MyKMSKey* so that it allows the *sofia* IAM user to use the key.
   Tip: The existing key policy contains a statement with a statement ID (SID) value

of *Allow use of the key*. Add an additional principal to this statement so that the principal allows both the *voclabs* IAM role and the *sofia* IAM user to use the key. After you modify the key policy, the code *inside* the `Principal {},` element should match the following (where ACCOUNT-NUMBER is the actual account number):

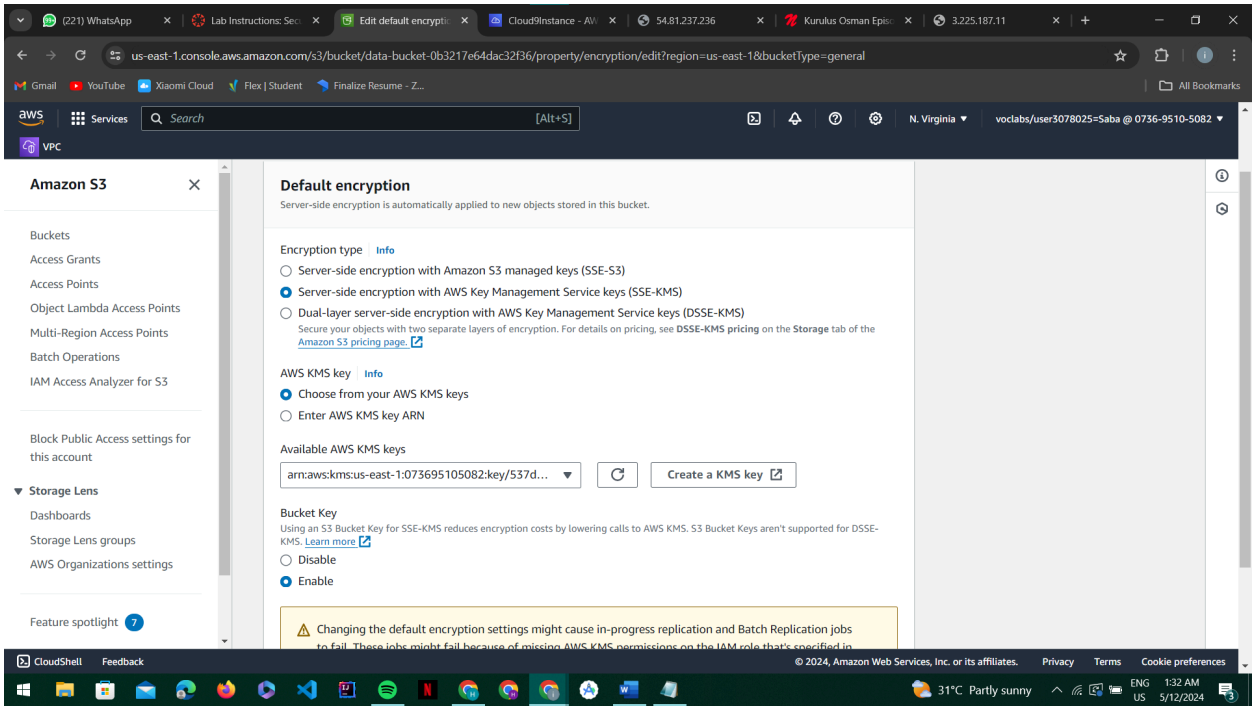
2. `"AWS": [`
   `"arn:aws:iam::ACCOUNT-NUMBER:role/voclabs",`
   `"arn:aws:iam::ACCOUNT-NUMBER:user/sofia"`
   `]`
3. Analyze the *PolicyForFinancialAdvisors* IAM policy that exists in your account.



Task 3.3: Use AWS KMS to encrypt data in Amazon S3

In this task, you are challenged to use the AWS KMS key that you created to encrypt an object in the *data-bucket* S3 bucket. You will then test access to the object.

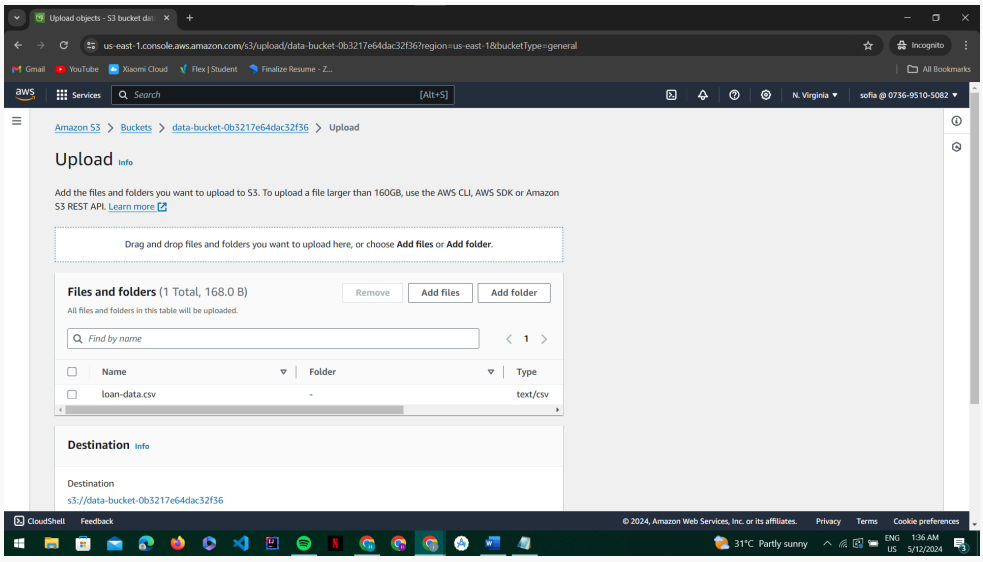
1. Modify the encryption settings on the *data-bucket* S3 bucket that you created in phase 1 so that the bucket uses *SSE-KMS* encryption.



2. On your computer, create a new file named `loan-data.csv` that you can use to test encryption. After creating the file in a text editor, paste the following data into the file and save the changes.

```
date,description,amount,principal,interest
2023-01-14,payment,1000.00,845.52,154.48
2022-12-22,payment,1021.52,742.80,278.72
2022-11-15,payment,1000.00,855.27,144.73
```

3. In an incognito or private browser window, log in to the AWS Management Console as the *sofia* IAM user.



4. While logged in as the *sofia* user, try to open or download the *loan-data.csv* file.
   The attempt should succeed.

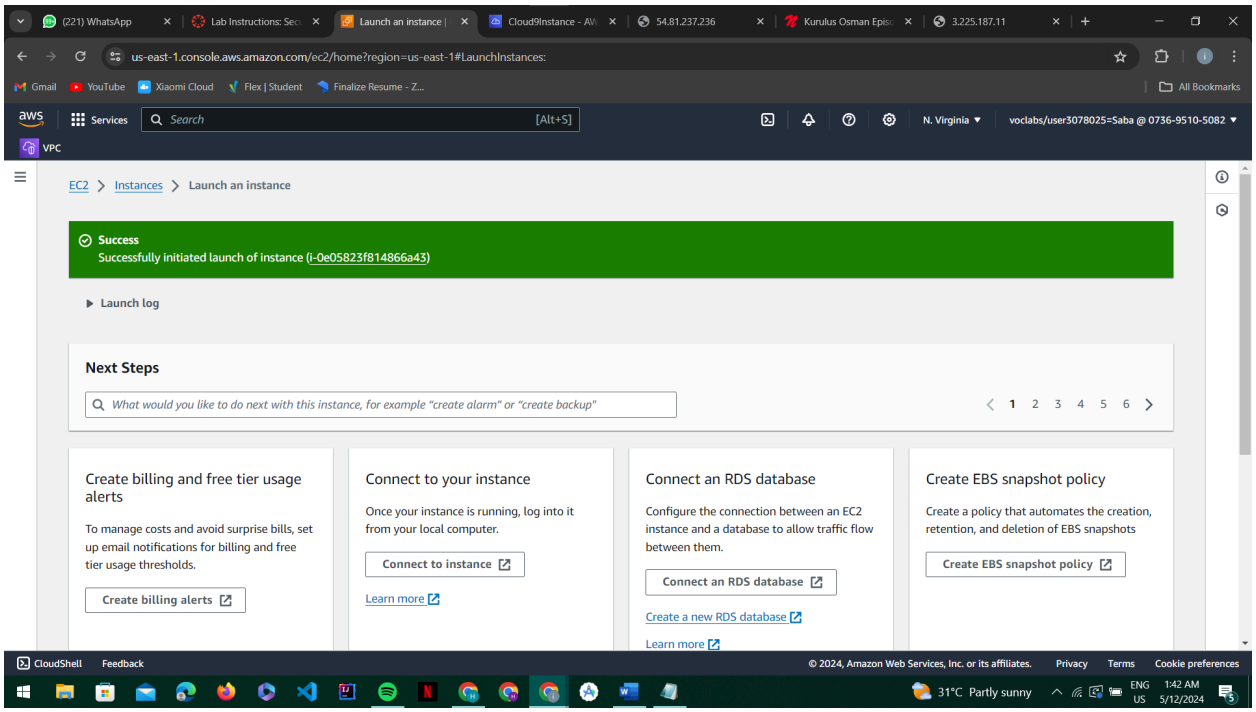
5. Sign the *sofia* user out of the console, and then sign in as the *paulo* user. Then, try to open or download the *loan-data.csv* object.
   The attempt should fail.
   - When done testing, log out and exit the incognito browser tab.

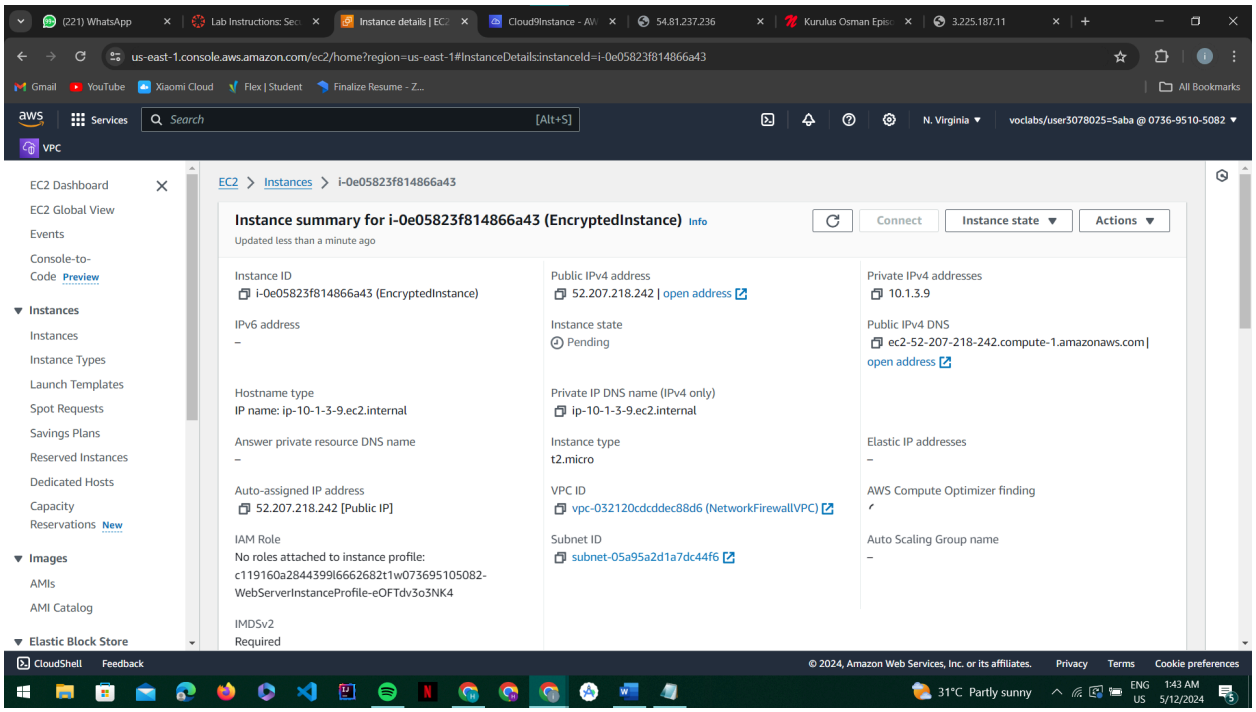
Task 3.4: Use AWS KMS to encrypt the root volume of an EC2 instance

In this task, you are challenged to use the AWS KMS key again, but now you will use it to encrypt the root volume of a new EC2 instance.

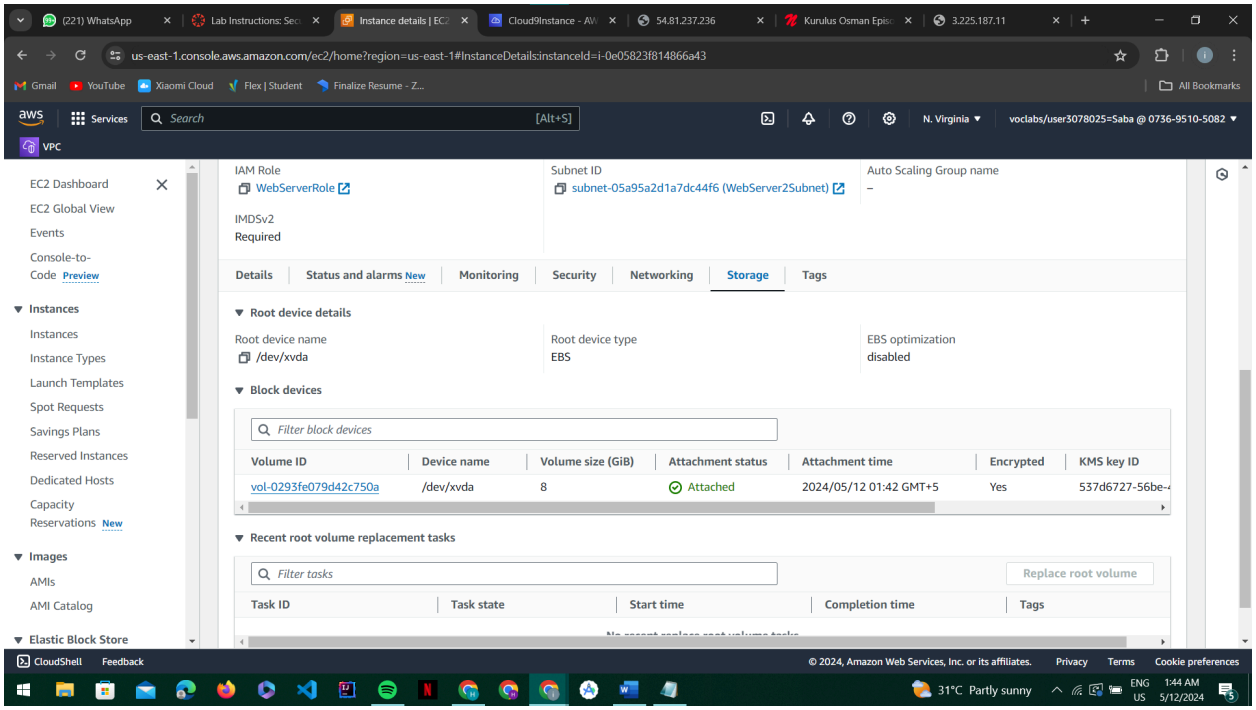
1. While logged in with the *voclabs* role, create a new EC2 instance. Keep the default settings except for the following:
   - Name the instance `EncryptedInstance`
   - Choose *Amazon Linux 2 AMI* as the AMI and *t2.micro* as the instance type.
   - For key pair, choose *vockey*.
   - Edit the network settings and deploy it to the *NetworkFirewallVPC* in *WebServerSubnet2*. Use the existing *WebServer2SecurityGroup*.
   - In the *Configure storage* settings, choose *Advanced* and choose to *encrypt* the *AMI root* volume by using *MyKMSKey*.

- In the *Advanced details*, assign *WebServerInstanceProfile* as the IAM instance profile.





2. On the details page for *EncryptedInstance*, choose the **Storage** tab and verify that the instance root volume is encrypted.



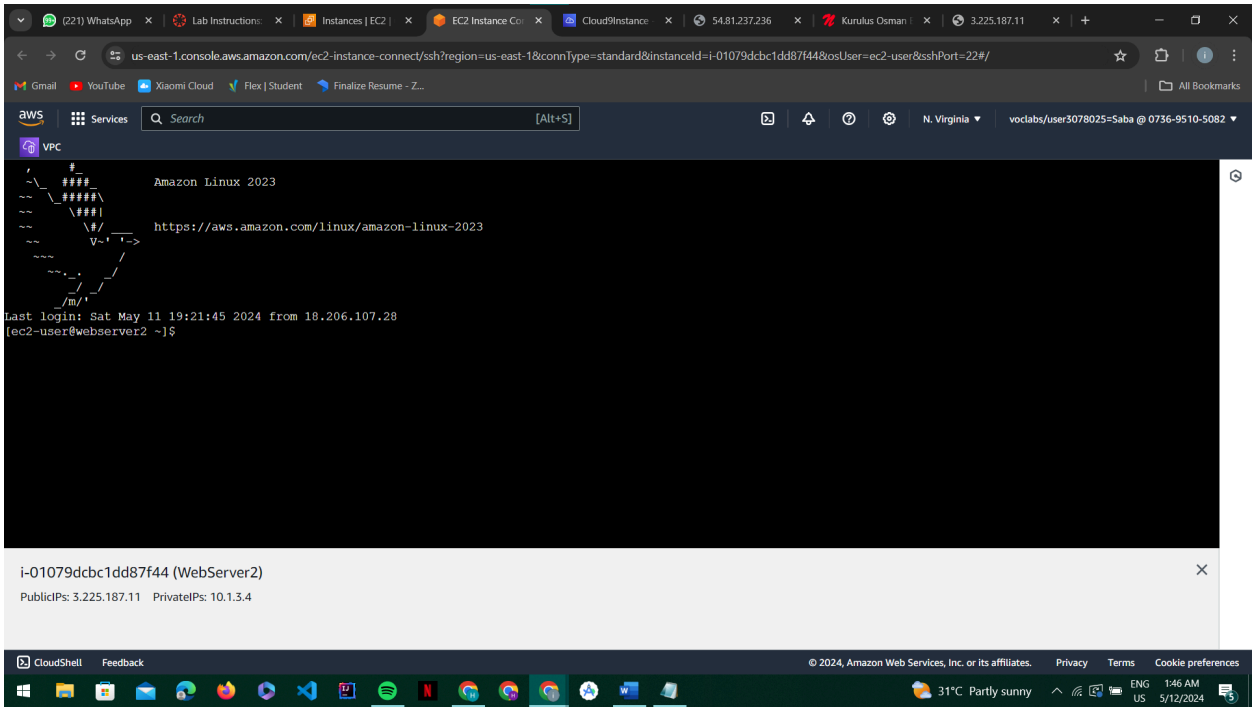
Task 3.5: Use AWS KMS envelope encryption to encrypt data in place

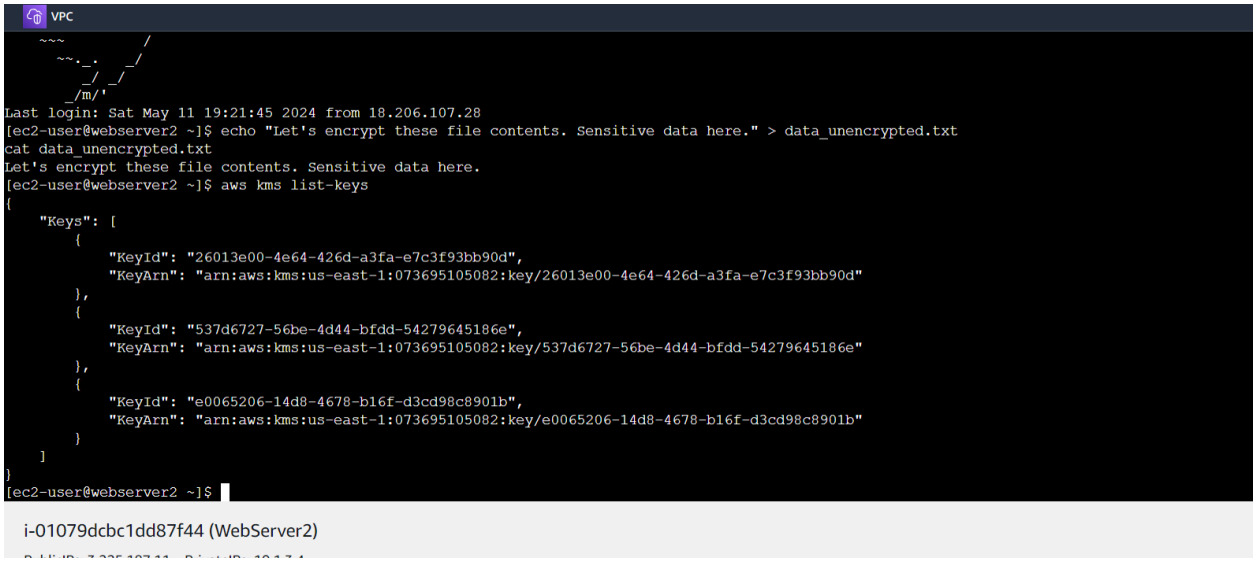
In this task, you are challenged to use the AWS Command Line Interface (AWS CLI) to encrypt data in place by using the AWS KMS key. You are also challenged to see how to decrypt the encrypted data. For convenience, you will use the *WebServer2* instance from phase 2 to complete this task.

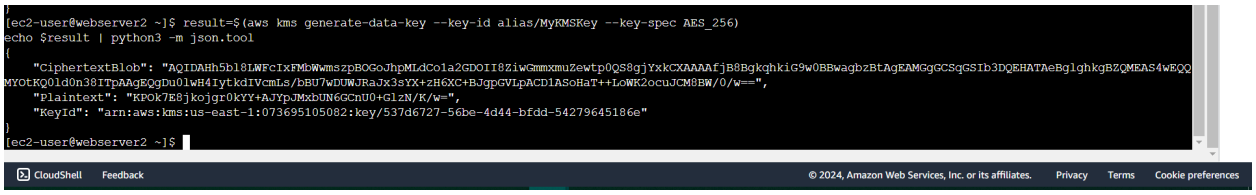
1. Use EC2 Instance Connect to connect to *WebServer2*.
2. To create a file that you will later try to encrypt, run the following commands in the EC2 Instance Connect session:
3. `echo "Let's encrypt these file contents. Sensitive data here." > data_unencrypted.txt`
   `cat data_unencrypted.txt`
4. Use the AWS CLI to generate a data key from the AWS KMS key.
   In the *WebServer2* EC2 Instance Connect session, run the following command to test access to AWS KMS:
   - `aws kms list-keys`
   - The command returns a list of AWS KMS keys in the account.

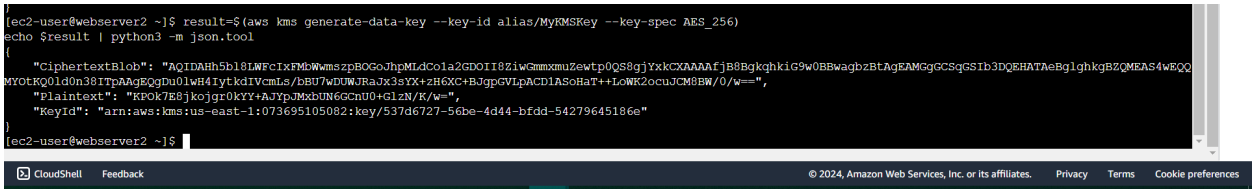
- Next, run the following commands to generate a *data key* for the MyKMSKey, save it to a bash variable, and then echo the data key details in pretty JSON format:

```
result=$(aws kms generate-data-key --key-id alias/MyKMSKey --key-spec AES_256)
echo $result | python3 -m json.tool
```









5. Save the data key to disk.
    - To export the *CiphertextBlob* value from the result and save it to a text file in base64-encoded format, run the following commands:

```
dk_cipher=$(echo $result| jq '.CiphertextBlob' | cut -d "" -f2)
echo $dk_cipher
echo $dk_cipher | base64 --decode > data_key_ciphertext
```

    - To see that the data key is stored in encrypted (not human-readable) format, run the following command:

```
cat data_key_ciphertext
```

- **Tip:** You can regenerate the plaintext version of the data key from the base64-encoded ciphertext at anytime, as demonstrated by running the following command:

```
aws kms decrypt --ciphertext-blob fileb://./data_key_ciphertext --query Plaintext --output text
```

- Run the previous command again but also save the result to a file in base64-encoded format:

```
aws kms decrypt --ciphertext-blob fileb://./data_key_ciphertext --query Plaintext --output text | base64 --decode > data_key_plaintext_encrypted
```

-

6. Use the data key to encrypt the file that you created earlier.
   - To encrypt the *data_unencrypted.txt* file, which you created in the previous step, run the following command:

```
openssl enc -aes-256-cbc -salt -pbkdf2 -in data_unencrypted.txt -out data_encrypted -pass file:data_key_plaintext_encrypted
```

   - To try to read the file, run the following command:

```
cat data_encrypted
```

   - It's not human-readable, unfortunately.
   - To delete the unencrypted version of the file, run the following command:

```
rm data_unencrypted.txt
```

7. Decrypt the file to prove that the data is retrievable.
   - To decrypt the data and save it as a new file named *data_decrypted*, run the following command:

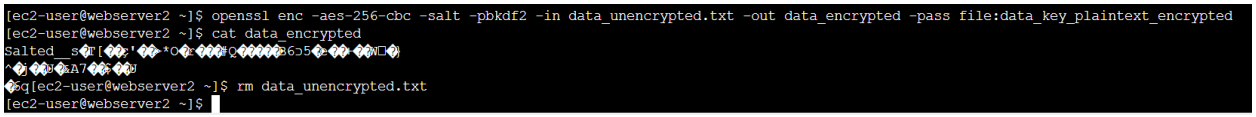
```
openssl enc -d -aes-256-cbc -pbkdf2 -in data_encrypted -out data_decrypted.txt -pass file:./data_key_plaintext_encrypted
```

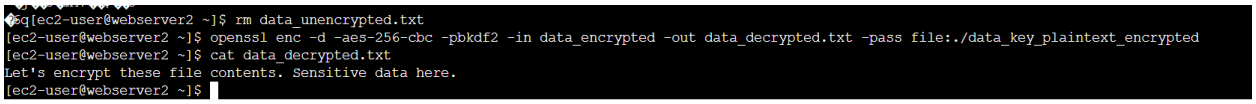
- To print the results to ensure the data is now readable, run the following command:

```
cat data_decrypted.txt
```

- You should see the text from the original *data_unencrypted.txt* file. You have now seen how it's possible to use envelope encryption to encrypt data at rest.

To see that the data key is stored in encrypted (not human-readable) format, run the following command:

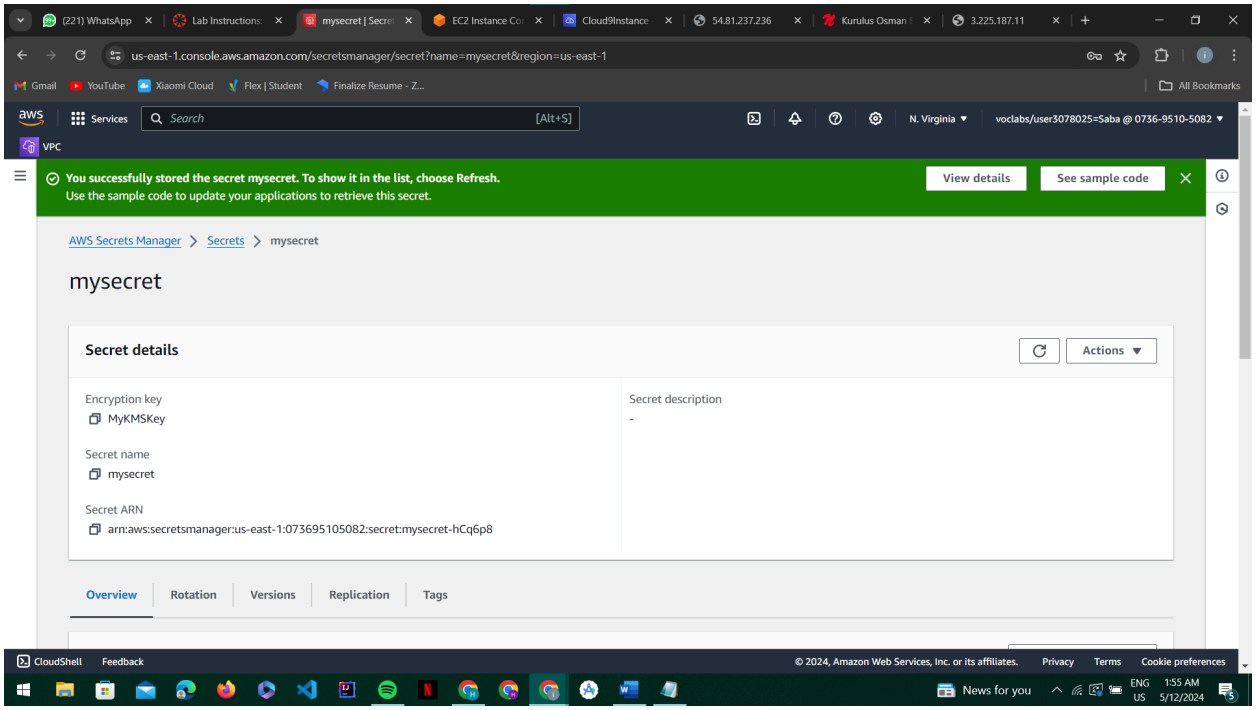




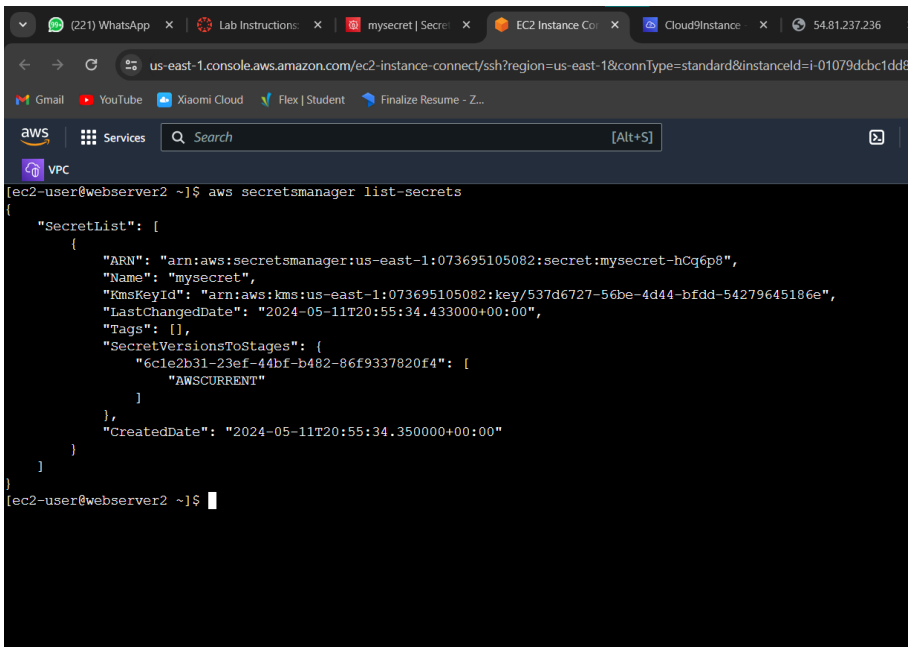
Task 3.6: Use AWS KMS to encrypt a Secrets Manager secret

In this task, you are challenged to create a key-value pair (a *secret*), which you will encrypt with your AWS KMS key and store in Secrets Manager. You are then challenged to verify that you can retrieve the secret by using the AWS CLI.

1. In the Secrets Manager console, create a new secret of type *Other type of secret*.
    - Give it a key of `secret` with the value `my secret data`.
    - Encrypt it with your *MyKMSKey* and name the secret `mysecret`.
      Tip: Notice that the console provides sample code that you could use to access this secret later by using a programming language.



2. Use EC2 Instance Connect to connect to the *WebServer2* instance, and then use the AWS CLI to retrieve the secret.
   Tip: Start by invoking the `aws secretsmanager list-secrets` command. Then, use the `aws secretsmanager get-secret-value` command to actually retrieve the secret.
   You should see the secret's key-value pair data returned in the results.



Phase 4: Monitoring and logging

The leadership team at AnyCompany Financial received reports of a new security breach at one of its biggest competitors. The company wants to ensure that it's prepared to detect and respond to any future security incidents. The director of IT has asked you to implement a monitoring and logging solution to detect security incidents so that the company can respond promptly.

The solution needs to do the following:

- Track all API calls to S3 buckets.
- Monitor application logs.
- Notify team members in case of security incidents.
- Monitor AWS resource configurations and automatically modify configurations that are out of compliance.

The following table describes the tasks that you will implement in this phase.

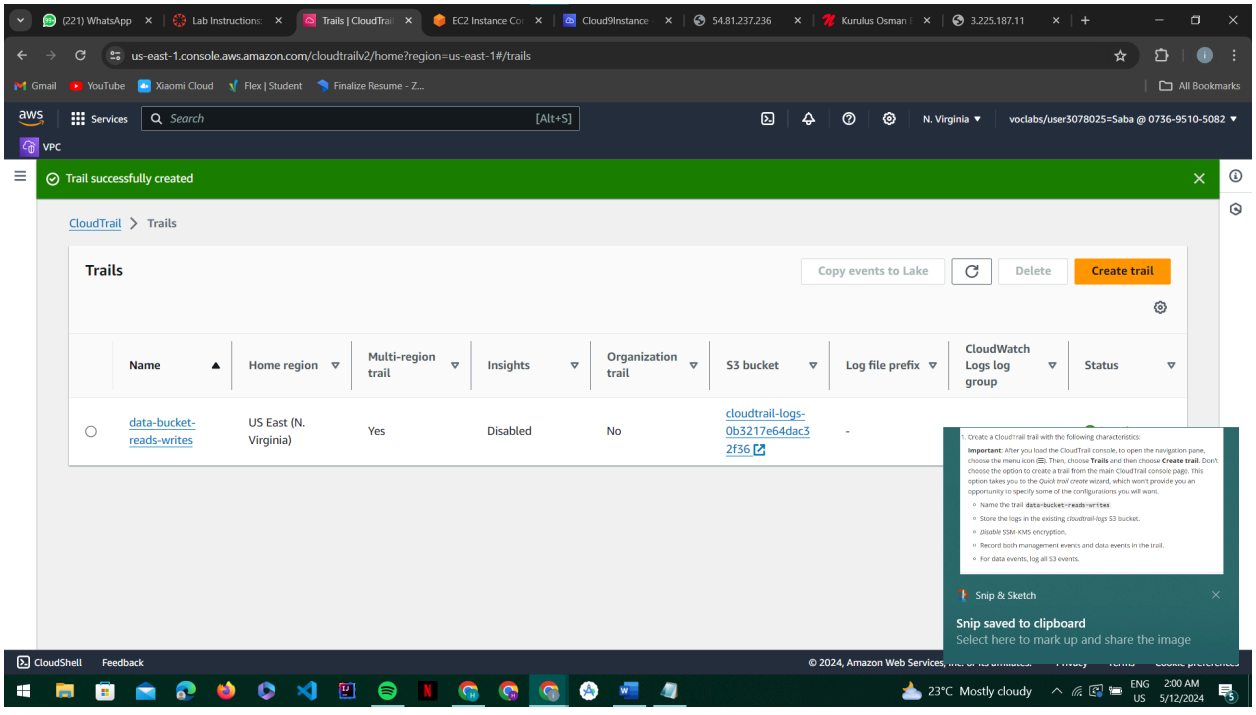
Numbered Task	Detail
1	Create a trail in AWS CloudTrail to record Amazon S3 API calls.
2	Configure CloudWatch Logs to monitor the *WebServer* instance's authentication logs.
3	Create a CloudWatch alarm to notify team members when access to the *WebServer* instance is attempted.
4	Use AWS Config to ensure that S3 buckets have object logging enabled when they are created.

Task 4.1: Use CloudTrail to record Amazon S3 API calls

In this task, you are challenged to use CloudTrail to record API calls that are made to Amazon S3 buckets. This information will provide an audit trail to track when S3 objects are created, modified, or read.

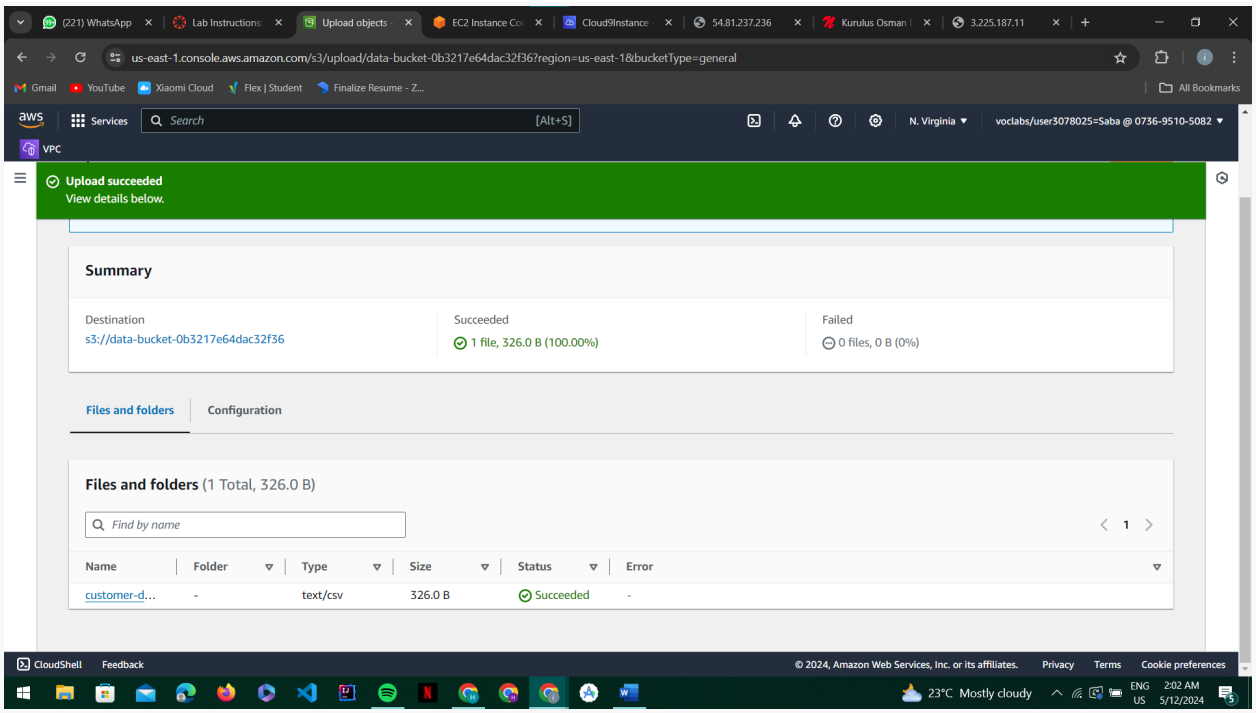
By the end of this task, you will have configured the architecture that is shown in the following diagram:

1. Create a CloudTrail trail with the following characteristics:
   Important: After you load the CloudTrail console, to open the navigation pane, choose the menu icon (). Then, choose **Trails** and then choose **Create trail**. Don't choose the option to create a trail from the main CloudTrail console page. This option takes you to the *Quick trail create* wizard, which won't provide you an opportunity to specify some of the configurations you will want.
   - Name the trail `data-bucket-reads-writes`
   - Store the logs in the existing *cloudtrail-logs* S3 bucket.
   - *Disable* SSM-KMS encryption.
   - Record both management events and data events in the trail.
   - For data events, log all S3 events.



2. On your computer, create a file named `customer-data.csv`. Then, in a text editor, paste the following data into the file and save the changes.

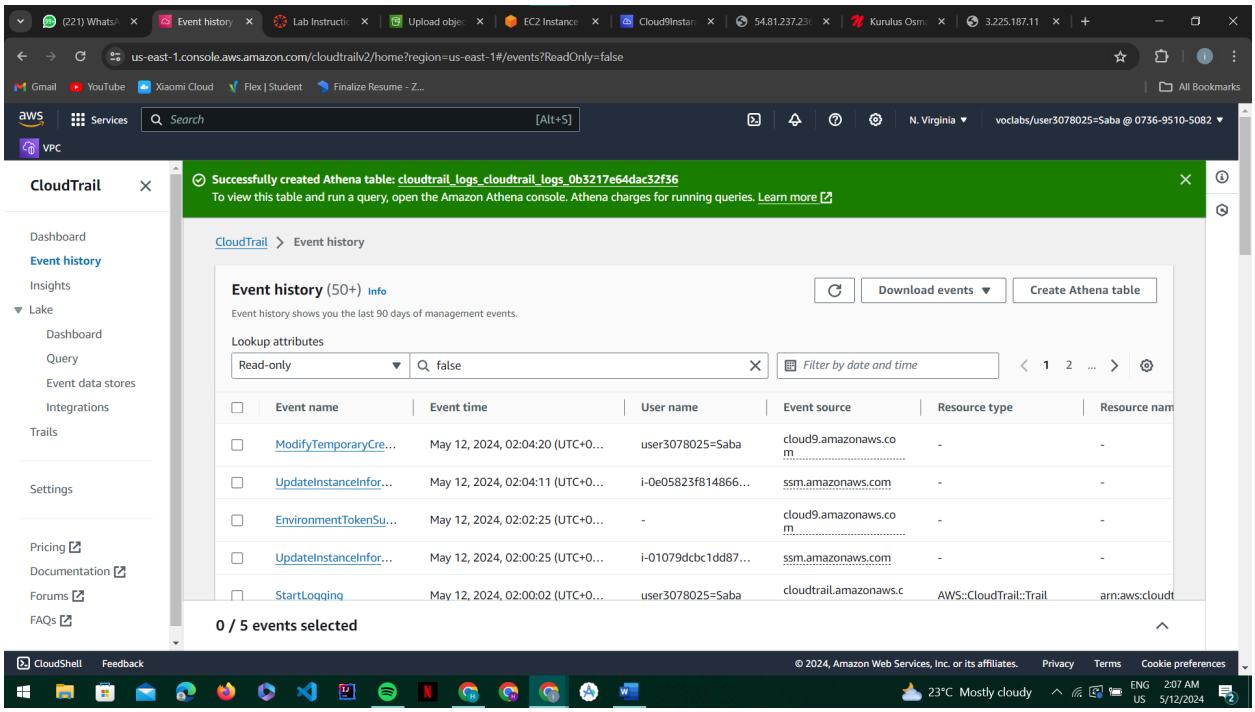
```
CustomerID,First Name,Last Name,Join Date,Street Address,City,State,Phone
1,Alejandro,Rosalez,12/12/2013,123 Main St.,Any Town,MD,301-555-0158
2,Jane,Doe,10/5/2014,456 State St.,Anywhere,WA,360-555-0163
3,John,Stiles,9/20/2016,1980 8th St.,Nowhere,NY,914-555-0122
4,Li,Juan,6/29/2011,1323 22nd Ave.,Anytown,NY,914-555-0149
```



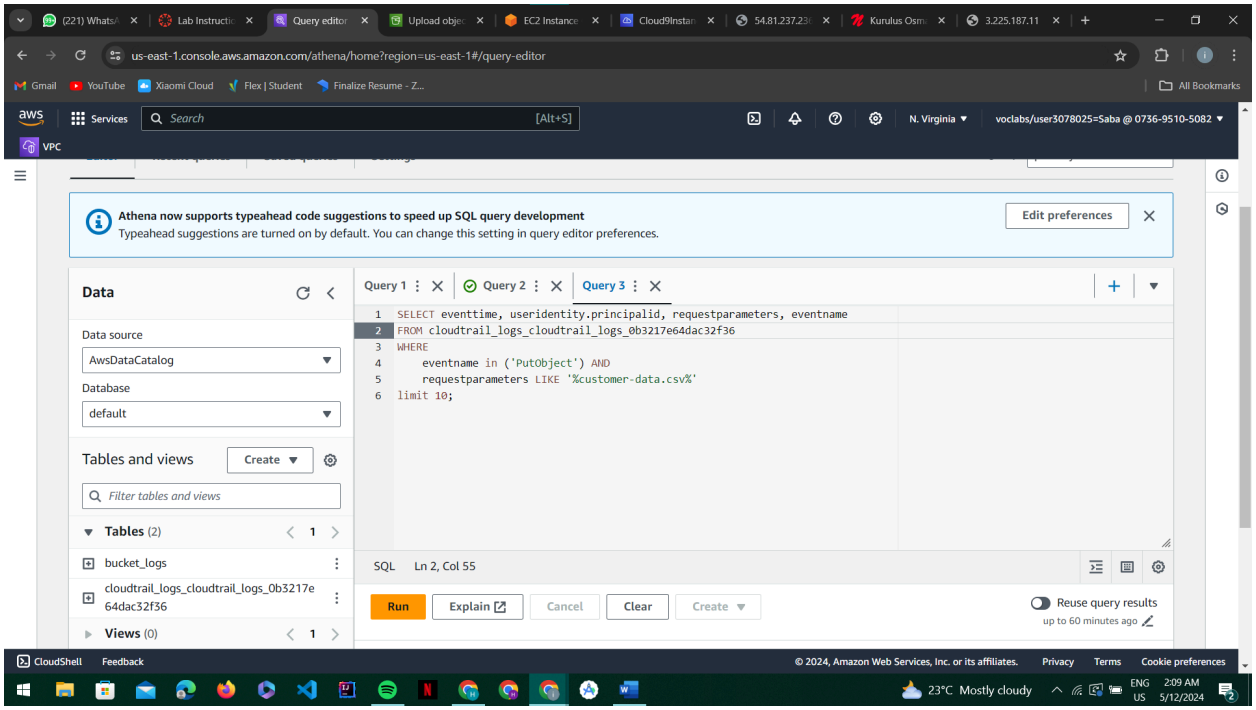
3. Upload the *customer-data.csv* file to *data-bucket*.
   After you upload the file, open the file in the Amazon S3 console.
   Note: Your actions from this step will create entries in the CloudTrail trail, which will be important for the next steps.

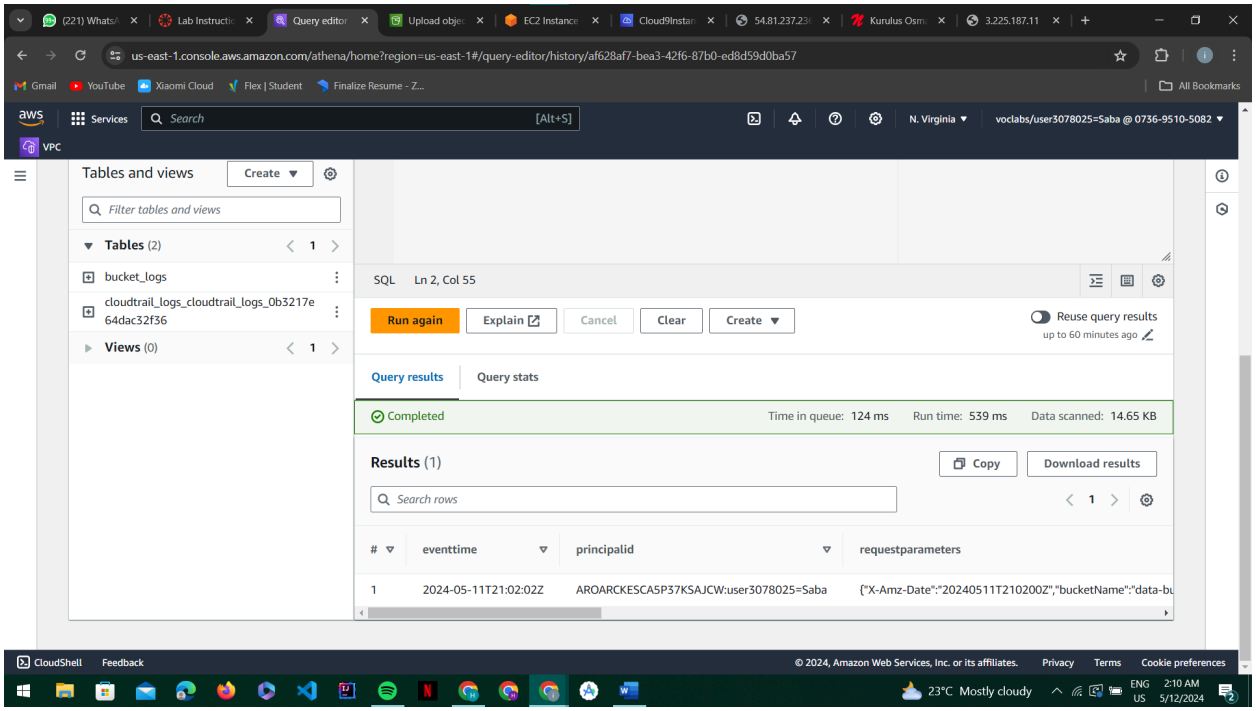
4. Use the CloudTrail console to create an Athena table that describes the format of the data in the *cloudtrail-logs* S3 bucket.
   - Consult the CloudTrail documentation for how to accomplish this.
   - Before creating the table, be sure to configure the **Storage location** to be the **cloudtrail-logs** bucket.
     Analysis: A *CREATE EXTERNAL TABLE* table query is generated in Athena. When you run the query in the next step, the table will be created. The table will define the structure of the data that is being written to the

cloudtrail-logs S3 bucket so that you can later query it. You will use Athena in this task to analyze CloudTrail logs, in the same way that you used Athena to analyze S3 object-level logging in phase 1.



5. Create and run an Athena query to retrieve the CloudTrail event log data for when you uploaded the *customer-data.csv* file to Amazon S3.
    - In the Athena console, *preview* the cloudtrail_logs table that should now exist since you created it in the previous step.
      Verify that 10 rows of data appear in the **Results** area.
      Important: Typically, CloudTrail delivers events within 5 minutes of an API call. You might need to wait a few minutes and then try to preview the table again before it returns results. Verify that data is returned in the preview before continuing to the next step.
    - Run query shown below into a new query tab after replacing `<table name>` with the correct table name:
    - ```
      SELECT eventtime, useridentity.principalid, requestparameters, eventname
      FROM <table name>
      WHERE
        eventname in ('PutObject') AND
        requestparameters LIKE '%customer-data.csv%'
      limit 10;
      ```
    - The following image provides an example of the results:
      ```





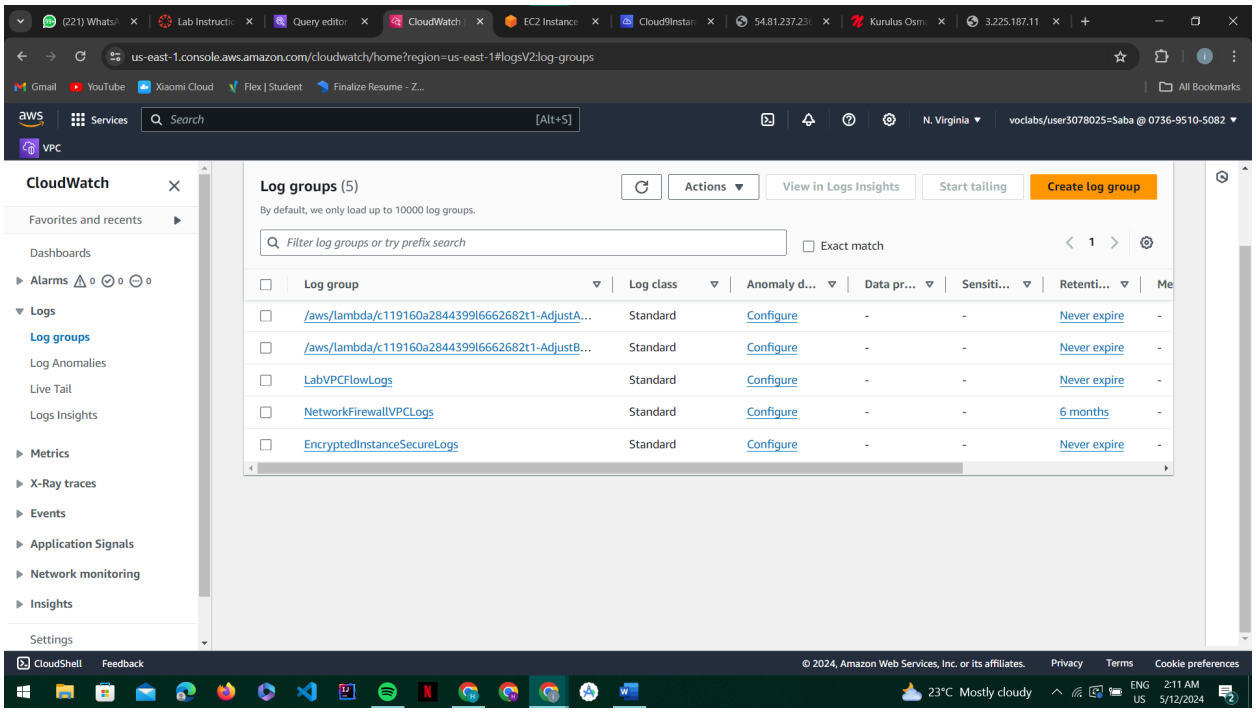
Task 4.2: Use CloudWatch Logs to monitor secure logs

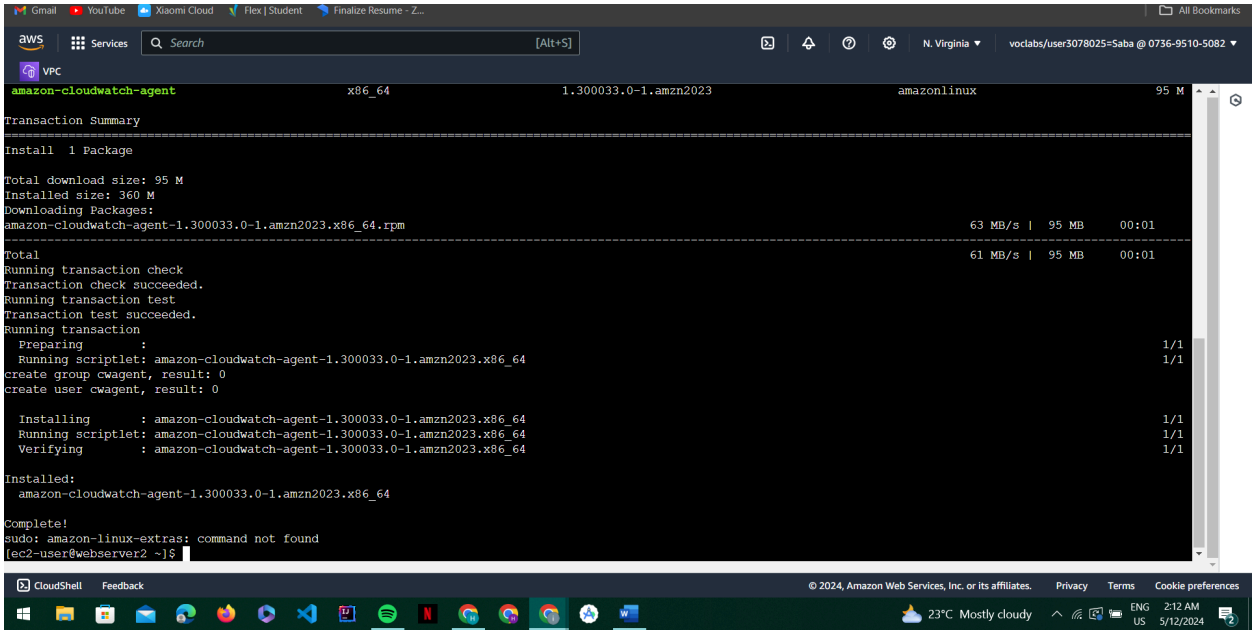
AnyCompany Financial has been working with a vendor to design and manage their website. The vendor has requested SSH access to design and manage the website. Company policy doesn't allow direct access through SSH to the production web servers, but access can be provided to a development web server named *EncryptedInstance*.

1. Create a CloudWatch log group named `EncryptedInstanceSecureLogs` with all default settings.

2. Use EC2 Instance Connect to connect to *EncryptedInstance*.
   After you connect, run the following commands to install the CloudWatch agent and a Linux daemon named collectd, which the CloudWatch agent will use:

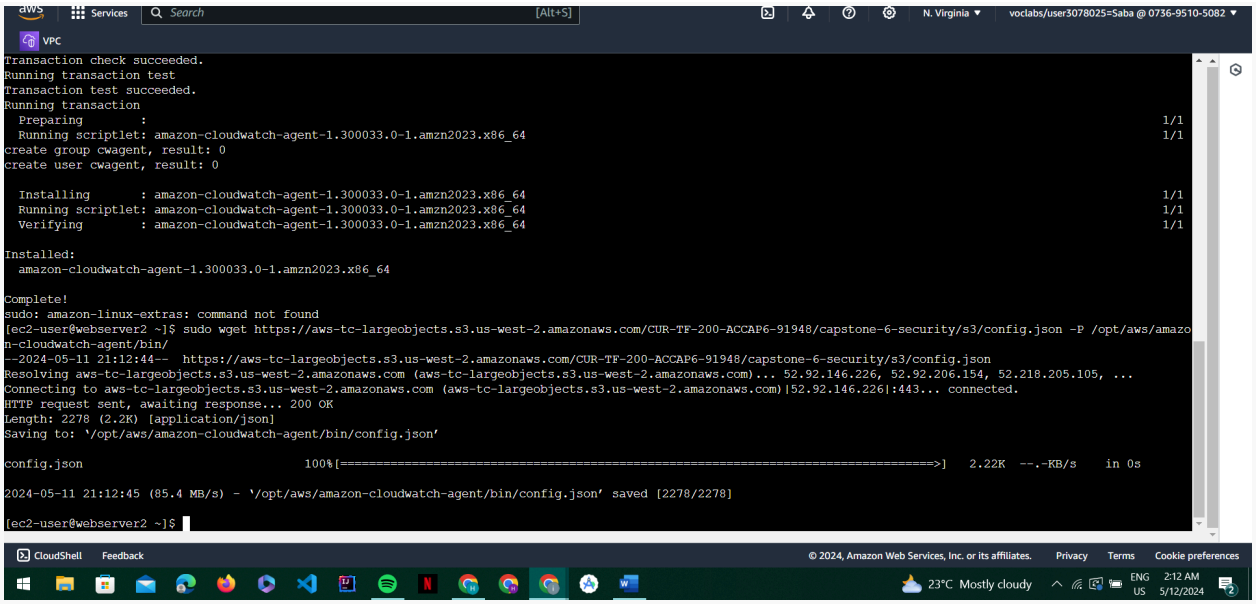
```
sudo yum install -y amazon-cloudwatch-agent
sudo amazon-linux-extras install -y collectd
```

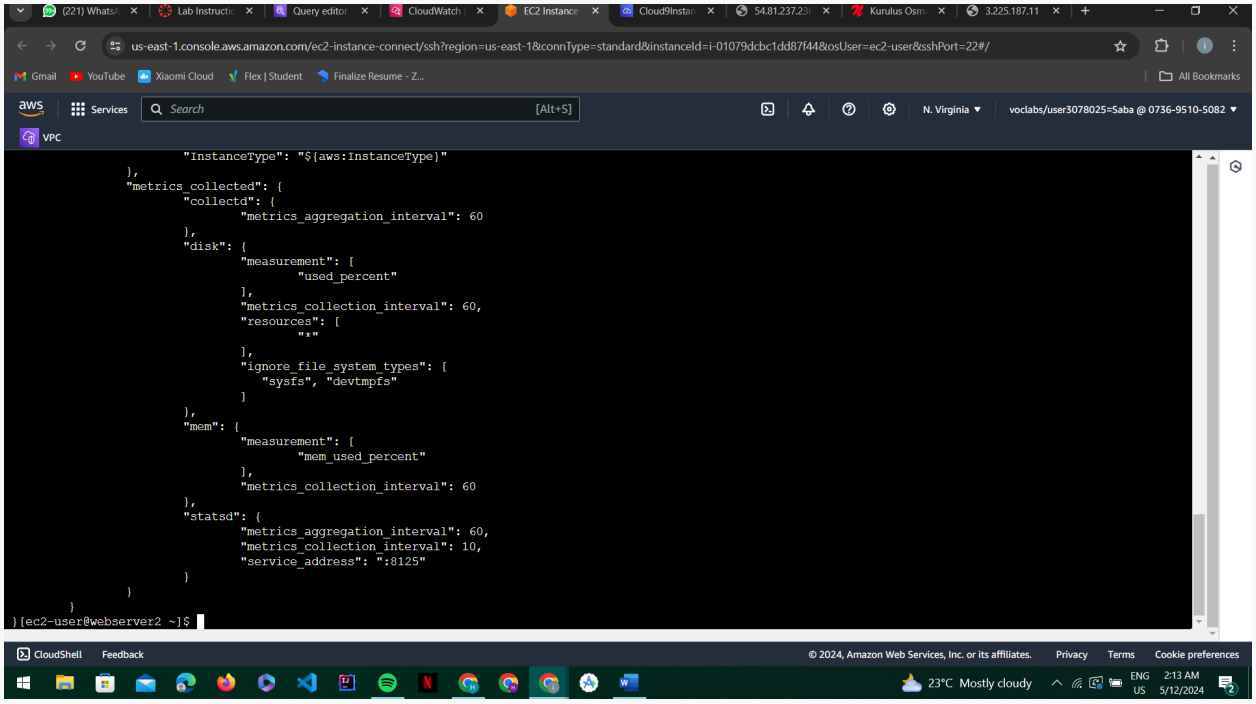




The agent and the collectd software should be successfully installed.

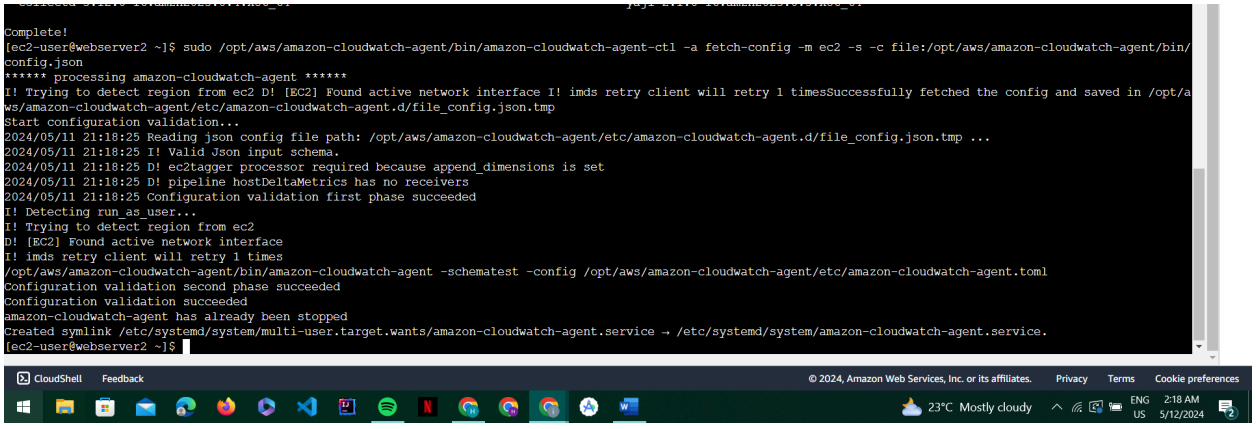
3. Download and configure a JSON file that provides configuration details for the CloudWatch agent.
    - To download the template file, run the following command in the EC2 Instance Connect session:
    - `sudo wget https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-200-ACCAP6-91948/capstone-6-security/s3/config.json -P /opt/aws/amazon-cloudwatch-agent/bin/`
    - To print out the file template so that you can see what it specifies, run the following command:
    - `sudo cat /opt/aws/amazon-cloudwatch-agent/bin/config.json`

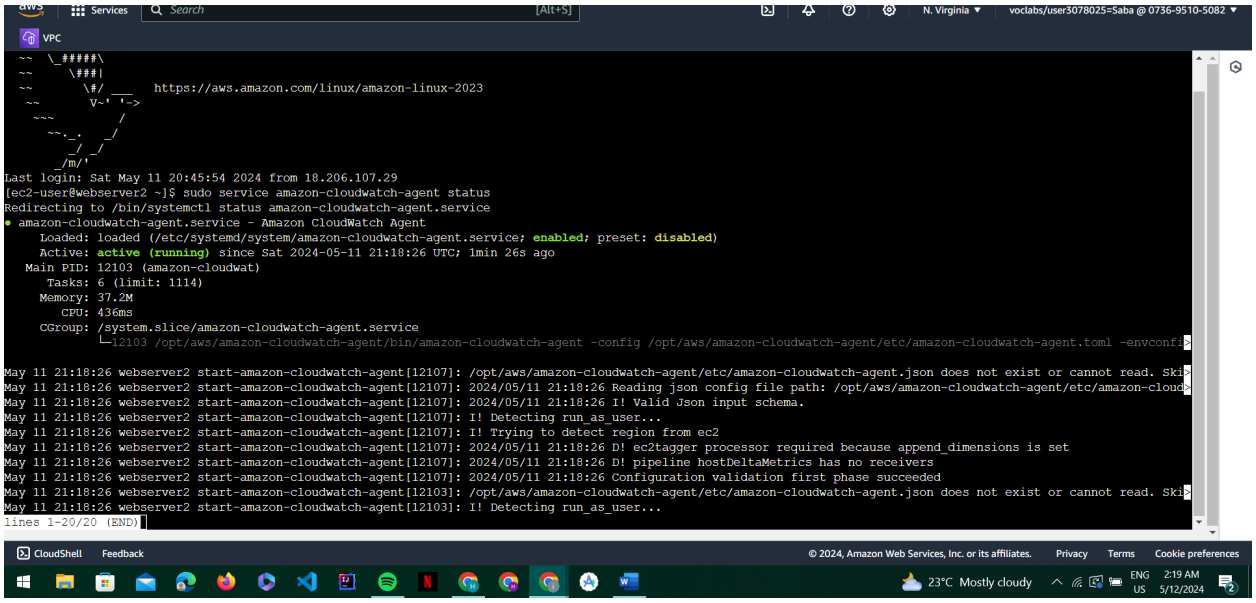




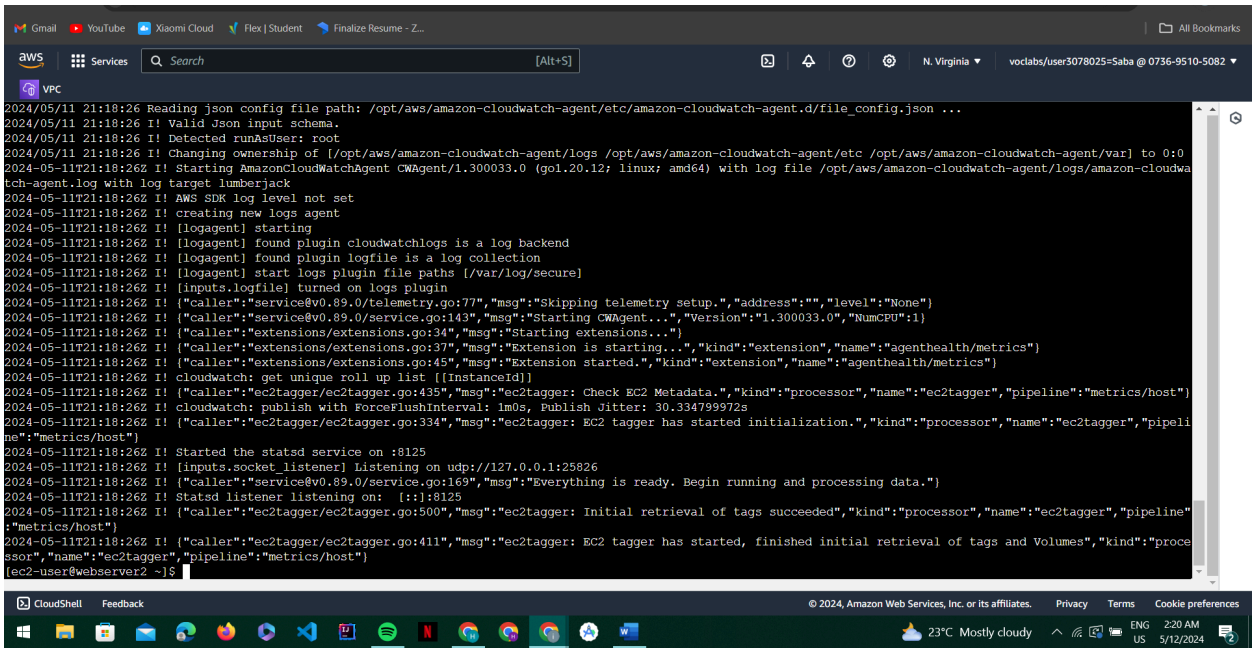
4. Start the CloudWatch agent, and confirm that it is running.
    - To start the agent, run the following command:
    - sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m ec2 -s -c file:/opt/aws/amazon-cloudwatch-agent/bin/config.json
    - To verify the CloudWatch agent status, run the following command:
    - sudo service amazon-cloudwatch-agent status
    - The output should indicate that the agent is *active* and *running*.

- To confirm that the CloudWatch agent is able to reach the CloudWatch service in your AWS account, run the following command:
- `sudo cat /opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-agent.log`
- The output should include a line toward the end that says "[logagent] piping log from EncryptedInstanceSecureLogs/EncryptedInstanceSecureLogs-...(/var/log/secure) to cloudwatchlogs...".
- To actively tail the */var/log/secure* file so that you can monitor the logs that will be created in the next step, run the following command:
- `sudo tail -f /var/log/secure`
- Keep the EC2 Instance Connect tab open, and continue to the next step.

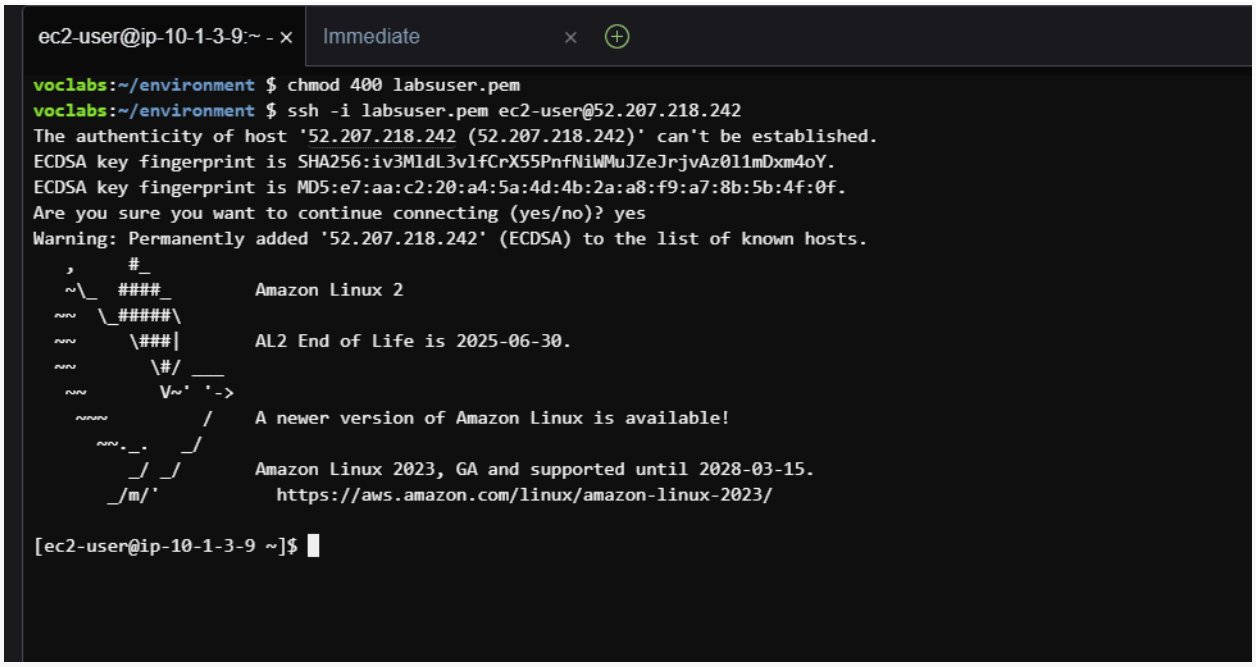




```
[ec2-user@webserver2 ~]$ sudo cat /opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-agent.log
2024/05/11 21:18:26 I! Config has been translated into TOML /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.toml
2024/05/11 21:18:26 D! config [agent]
  collection_jitter = "0s"
  debug = false
  flush_interval = "1s"
  flush_jitter = "0s"
  hostname = ""
  interval = "60s"
  logfile = "/opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-agent.log"
  logtarget = "lumberjack"
  metric_batch_size = 1000
  metric_buffer_limit = 10000
  omit_hostname = false
  precision = ""
  quiet = false
```



5. Create some security logs by successfully connecting and then failing to connect to the *EncryptedInstance* over SSH from your AWS Cloud9 IDE.
    - Download the PEM file from the **AWS Details** link above these lab instructions.
    - Upload the PEM file from your computer to your AWS Cloud9 IDE by using the **File** > **Upload Local Files** option.
      Tip: Before you run the next command, you might want to arrange your browser tabs so that you can see the EC2 Instance Connect session that you still have open from the previous step.
    - In the AWS Cloud9 bash terminal, run the following commands. Replace `EncryptedInstance-public-IP` with the public IPv4 address of the *EncryptedInstance*:
    - chmod 400 labsuser.pem
      ssh -i labsuser.pem ec2-user@EncryptedInstance-public-IP



- When prompted if you are sure that you want to connect, enter `yes`
  The connection should succeed. In the EC2 Instance Connect session, you should see that lines were recorded in the */var/log/secure* file when you ran the previous commands.
- In the AWS Cloud9 bash terminal, run the following commands to disconnect from the SSH session and then try to connect with a username that isn't valid. Replace `EncryptedInstance-public-IP` with the public IPv4 address of the *EncryptedInstance*:
- exit
  ssh -i labsuser.pem ubuntu@EncryptedInstance-public-IP
- The connection attempt fails with a "Permission denied" message.
  Analysis: You tried to connect to the instance with the username *ubuntu*, which doesn't exist on this Amazon Linux instance. This results in a failed connection attempt, which is also recorded in */var/log/secure*. (You should be able to see this where you are running the *tail* command in the EC2 Instance Connect session for *EncryptedInstance*.) Recall that the CloudWatch agent should be forwarding the secure file logs to the CloudWatch log group. You will confirm that next.

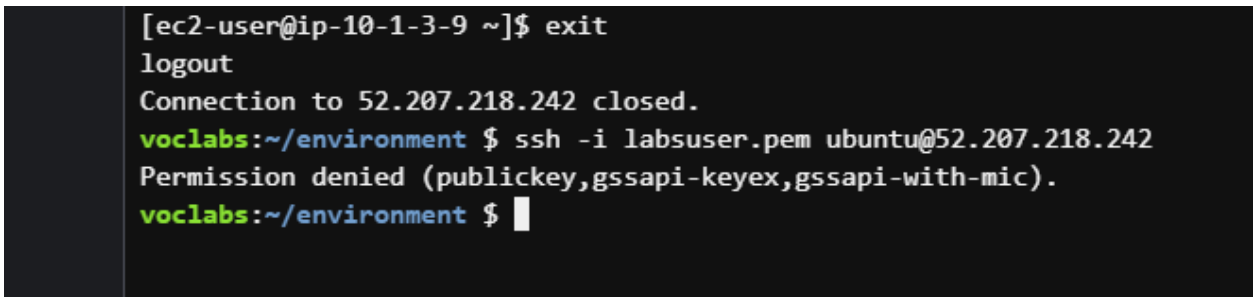
6. Verify that the secure logs are being written to the CloudWatch log group.
   - Locate the *EncryptedInstanceSecureLogs* CloudWatch log group.
   - Open the latest log stream.
     You should find that the SSH actions that you took have been logged. For

example, you should see log events similar to "Accepted publickey for ec2-user from..." and "Invalid user ubuntu from...". Expand the entries to view details.
Note: The result set that you see may also include many log entries of failed log attempts not initiated by you. This is not uncommon.

Task 4.3: Create a CloudWatch alarm to send notifications for security incidents

AnyCompany Financial wants security team members in IT to be notified when attempts to access the *EncryptedInstance* through SSH are denied. In this task, you will create a CloudWatch alarm to notify these team members when such an incident occurs

1. Go to the *EncryptedInstanceSecureLogs* CloudWatch log group, and create a metric filter with the following characteristics:
    - Filter pattern: `"Invalid user"` (Be sure to include the quotation marks.)
    - Filter name: `Not valid users`
    - Metric namespace: `secure`
    - Metric name: `NotValidUsers`
    - Metric value: `1`
    - Default value: `0`
    - Unit: **Count**



2. Create a CloudWatch alarm from the metric filter that you just created.
    - On the details page for the *EncryptedInstanceSecureLogs* log group, on the **Metric filters** tab, select the check box for the **Not valid users** metric filter.
    - Create a CloudWatch alarm with the following characteristics:
        - Period: *1 day*

        - Condition: Whenever NotValidUsers is *greater than or equal to 5*.
        - Notification: Create a new Amazon Simple Notification Service (Amazon SNS) topic named `Not_valid_users_exceeding_limit` and have notifications emailed to you. (Use an email address that you have access to.)
    - Name the alarm `Not valid users exceeding limit on EncryptedInstance` with the following description: `Not valid access attempts over SSH to the EncryptedInstance server have exceeded 4 in the last 24 hours.`
      Important: When you first create the alarm state will show *Insufficient data*, and the **Actions** column will show a warning. The warning appears because the alarm sends a message to an Amazon Simple Notification Service (Amazon SNS) topic with an endpoint (the email address) that hasn't yet been confirmed. The alarm won't work as expected until the endpoint is confirmed.

3. Go to the inbox for the email address that you entered in the previous step. You should see an email from AWS Notifications. In the email, choose the **Confirm subscription** link.

4. To test the alarm, return to the AWS Cloud9 IDE. In the bash terminal, run at least five invalid SSH access attempts to connect to the *EncryptedInstance* public IP address over SSH.

5. In the CloudWatch console, in the latest log stream for the *EncryptedInstanceSecureLogs* log group, filter the log events for `Invalid user`.
   Verify that at least five events are returned and that they all occurred within the past 24 hours. This indicates that the alarm threshold has been met.

6. In the Alarms area of the CloudWatch console, confirm that the *Invalid users* alarm has a state of *In alarm*.
   Check your email to confirm that you received a notification that the alarm threshold was crossed.