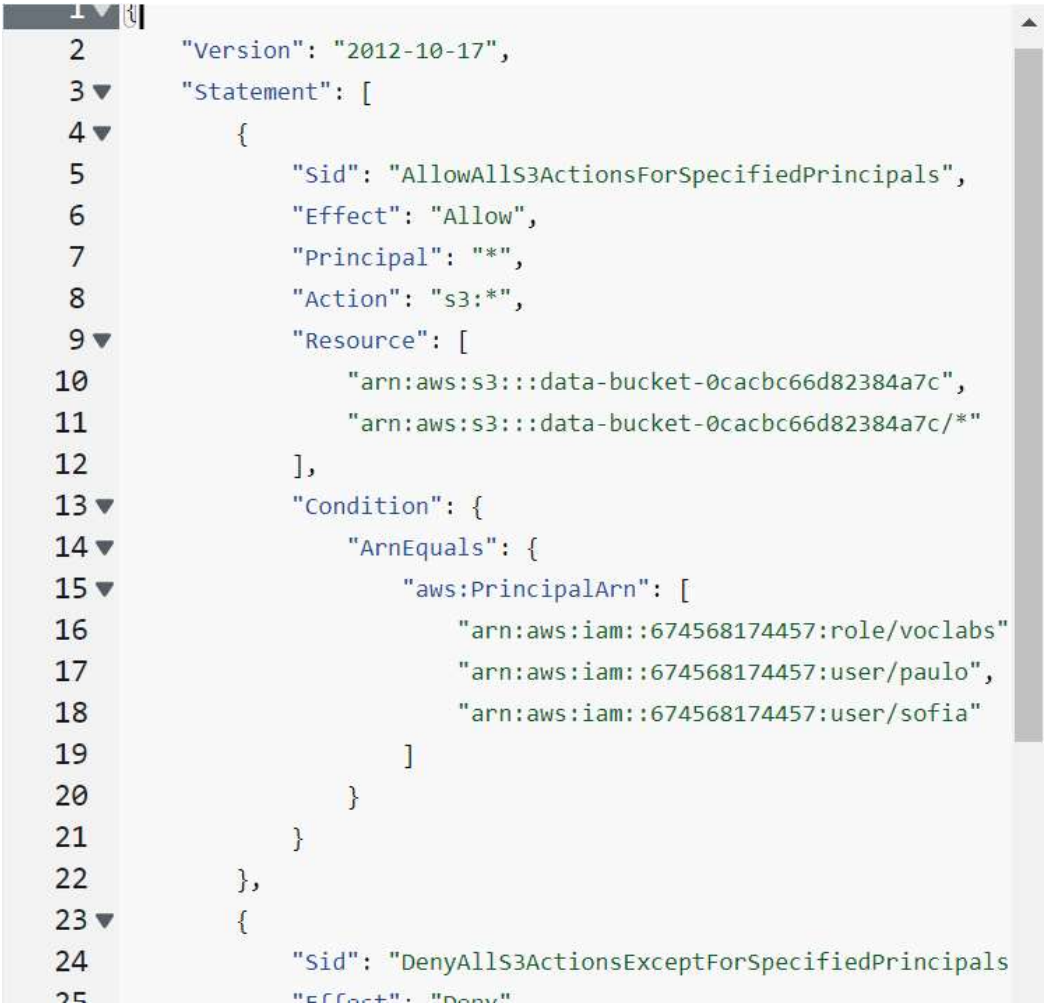
Task 1:

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

We create a new bucket by the name of data-bucket, upload a text file to the bucket and modify the bucket policy according to the specifications.



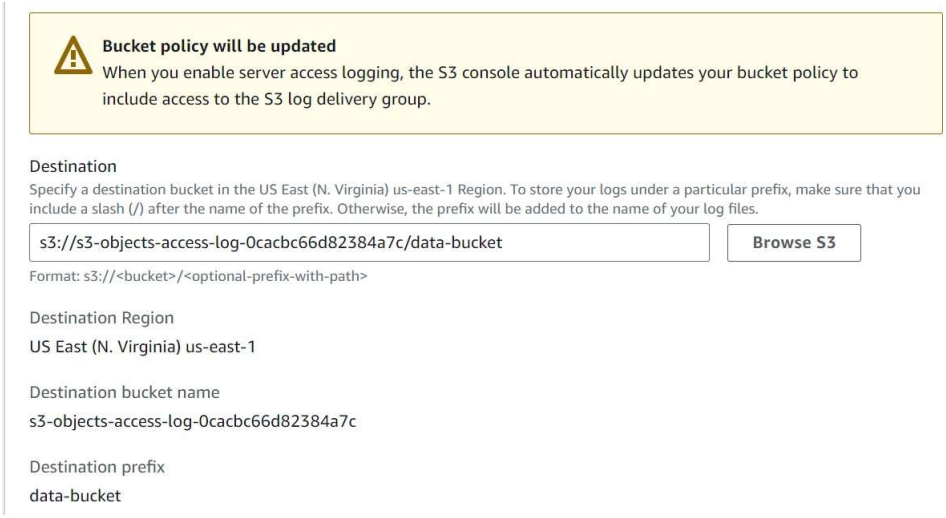
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAllS3ActionsForSpecifiedPrincipals",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::data-bucket-0cacbc66d82384a7c",
                "arn:aws:s3:::data-bucket-0cacbc66d82384a7c/*"
            ],
            "Condition": {
                "ArnEquals": {
                    "aws:PrincipalArn": [
                        "arn:aws:iam::674568174457:role/voclabs"
                        "arn:aws:iam::674568174457:user/paulo",
                        "arn:aws:iam::674568174457:user/sofia"
                    ]
                }
            }
        },
        {
            "Sid": "DenyAllS3ActionsExceptForSpecifiedPrincipals
```

Bucket policy for allow and deny

We now test access from Paulo and Mary users and the access levels are indeed according to the specifications.

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

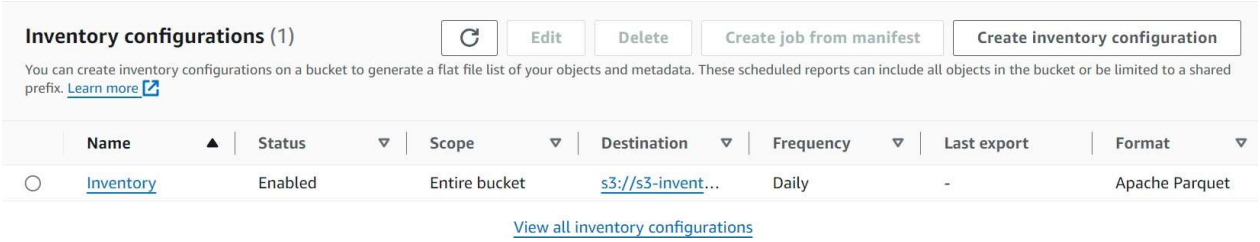
After enabling versioning from the properties tab in the bucket menu, we enable server access logging to the logging bucket already created by the environment and confirm the bucket policy change.



Access logging to the pre-created bucket

Task 1.3: Implement the S3 Inventory feature on a bucket

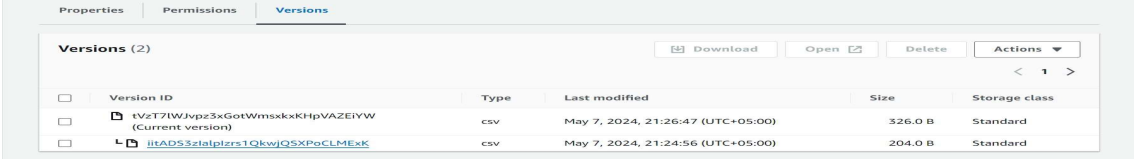
Implement the inventory configuration management setting under the management tab of the data bucket, and direct output to inventory bucket.



inventory management configuration

Task 1.4: Confirm that versioning works as intended

Create the file customers.csv on the computer and log in as Paulo user. Upload the file customers.csv to data-bucket as Paulo user, change it on your computer and upload again to see that there are multiple versions of the file in the bucket.



Multiple versions

Open both versions of the file to generate log data and log out of the Paulo user. Log in to the Mary user and try to access data-bucket, which fails due to permissions.

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

Confirm the log data by accessing one of the objects in the objects access bucket. Create a bucket by the name of athena-results-98765432 and configure this as the result destination from the athena console

Result destination for athena

Paste the provided query into the editor and run it, while observing the results. It generates a new table. Upon previewing the contents of the new table, run the other provided query in the editor and observe the IAM user access results

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

Under the management section of the data-bucket, locate inventory management configuration and select the s3-inventory link.

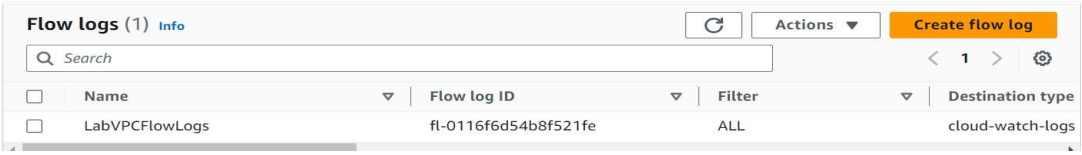
Task 2:

Task 2.1: Review LabVPC and its associated resources

Go to the VPC console and review the LabVPC instance along with its configurations and subnets. Go to the IAM and review the VPCFlowLogsRole and its policy. Afterwards, go to EC2 and observe the details for WebServer.

Task 2.2: Create a VPC flow log

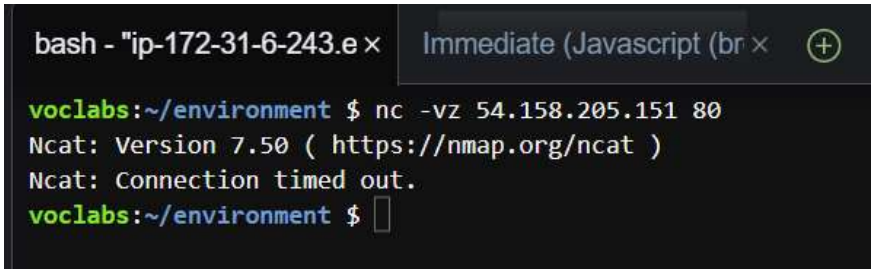
Select LabVPC and create a new flow log according to the specifications.



New flow log

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

Confirm that the WebServer instance's public IP address page does not load. Now, test access through Cloud9 IDE and run a command to check public access



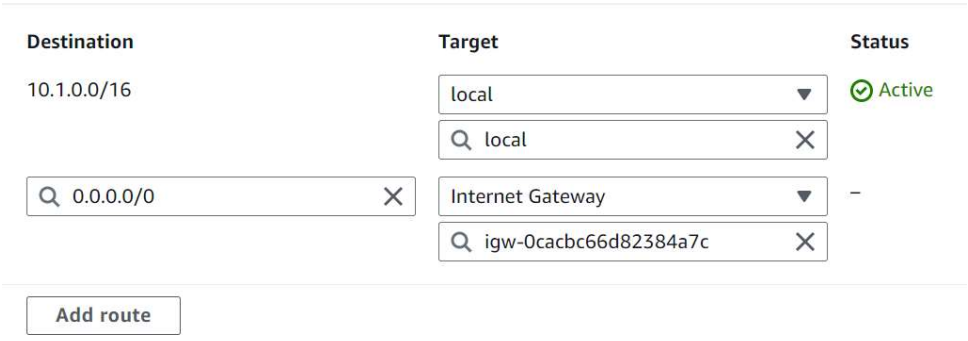
Cloud9Instance command to check access

Repeating this step with port 22 also gives the same result

Inexplicable error: cannot reach log destination. Can't fix because cannot delete flow log.

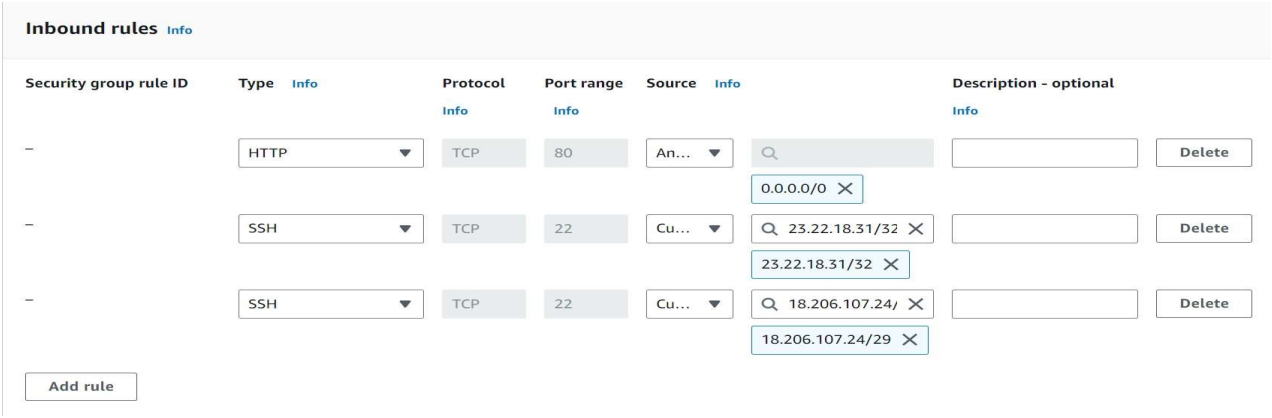
Task 2.4: Configure route table and security group settings

Go to VPC subnets and select WebServerSubnet and edit the route table to add the specified entry.



Adding a new route

Test accessing the public ip address again, which fails again as it is supposed to. Find the Security Group for the WebServer instance and edit the inbound rules as per the specifications. Add an HTTP rule for port 80 access and SSH rule as well. The pubic ipv4 address of the Cloud 9 instance is 23.22.18.31 which is the source, while the destination is the WebServer Security group. Additionally, configure EC2 instance connect as well.



Added inbound rules

Now, we use the nc command to once again test public ip address connectivity from port 22 and it is a success.

```
voclabs:~/environment $  nc -vz 54.158.205.151 22
Ncat: Version 7.50 ( https://nmap.org/ncat )
Ncat: Connected to 54.158.205.151:22.
Ncat: 0 bytes sent, 0 bytes received in 0.01 seconds.
voclabs:~/environment $
```

Test from web browser for HTTP over port 80

Test from Instance Connect

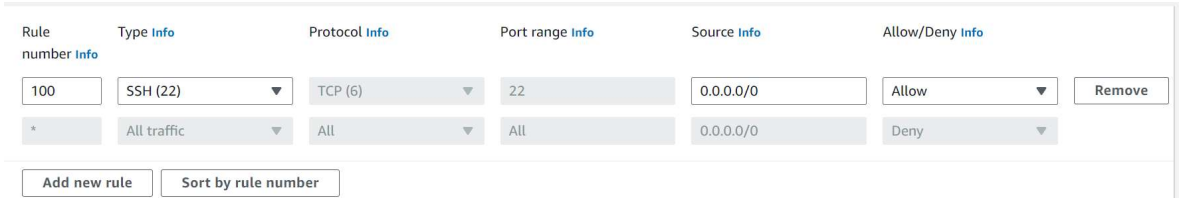
Task 2.5: Secure the WebServerSubnet with a network ACL

Navigating to the network ACL related to the WebServerSubnet. Modify the rule 100 from allow to deny and test access over port 22, which fails as expected. Network ACL overrides Security group inbound rules.

```
voclabs:~/environment $  nc -vz 54.158.205.151 22
Ncat: Version 7.50 ( https://nmap.org/ncat )
Ncat: Connection timed out.
```

Denied due to deny rule

Change the rule again to allow only port 22 access and test again.




```
voclabs:~/environment $  nc -vz 54.158.205.151 22
Ncat: Version 7.50 ( https://nmap.org/ncat )
Ncat: Connected to 54.158.205.151:22.
Ncat: 0 bytes sent, 0 bytes received in 0.01 seconds.
```

Successful because port 22 allowed only

Similarly, add a new rule with number 90 that allows HTTP traffic from anywhere. The web browser access now works as well.

Task 2.6: Review NetworkFirewallVPC and its associated resources

Overview the configurations of FirewallVPC including the network ACL default rule. Now view the WebServer2 instance details and confirm access through port 80 and 22. It is successful.

```
voclabs:~/environment $  nc -vz 44.223.161.213 22
Ncat: Version 7.50 ( https://nmap.org/ncat )
Ncat: Connected to 44.223.161.213:22.
Ncat: 0 bytes sent, 0 bytes received in 0.01 seconds.
```

Now, use Instance Connect and run a command.

```
[ec2-user@webserver2 ~]$ Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
```

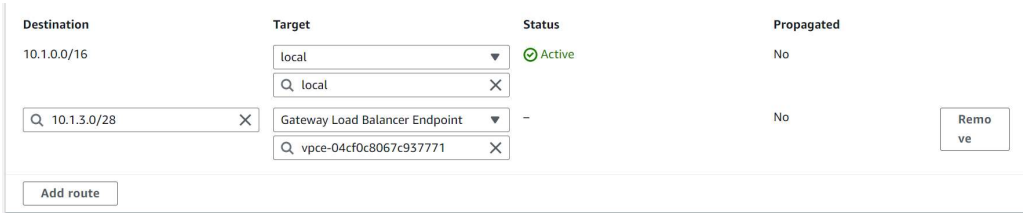
Now, on the web browser test connection over 8080 port. It is also successful

Task 2.7: Create a network firewall

Navigate to the VPC console and create a firewall according to the specifications.

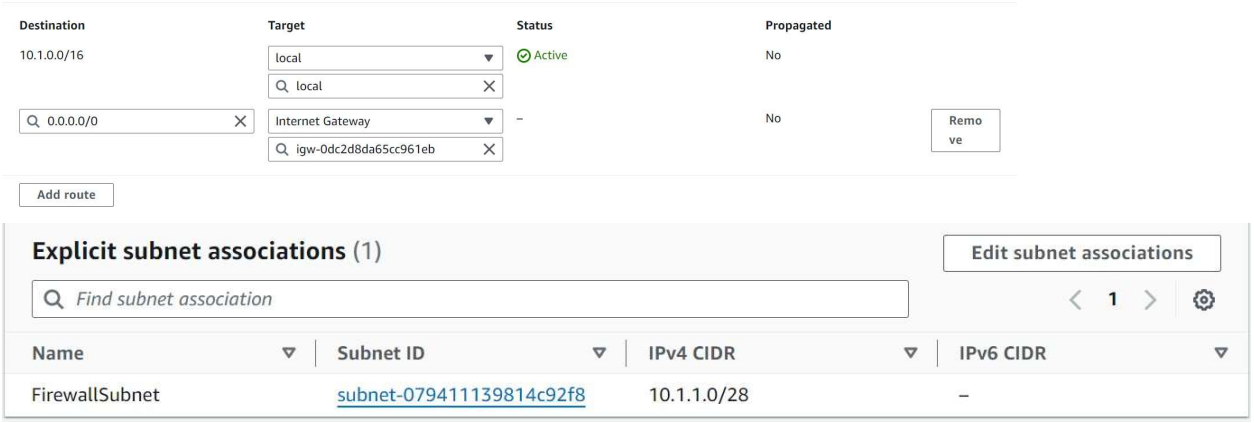
Task 2.8: Create route tables

Creating a new route table under NetworkFirewallVPC and edit it to add a new route that points to WebServer2.



Add an Edge Association so that the table is connected to NetworkFirewallIG

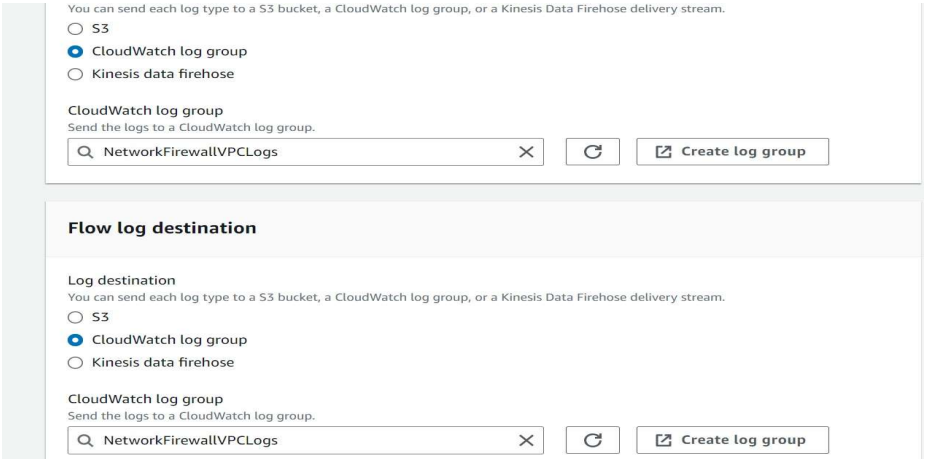
Creating another route table, associate it with FirewallSubnet and add a route to point traffic towards NetworkFirewallIG.



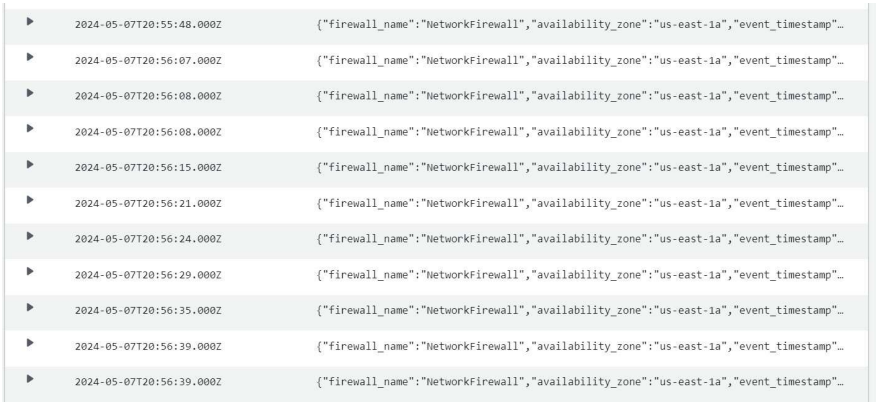
Create another route table for WebServer2 subnet under NetworkFirewallVPC

Task 2.9: Configure logging for the network firewall

Create a CloudWatch log group with 6 month retention settings. Configure alert and flow type logging for the firewall and point them to the cloudwatch group



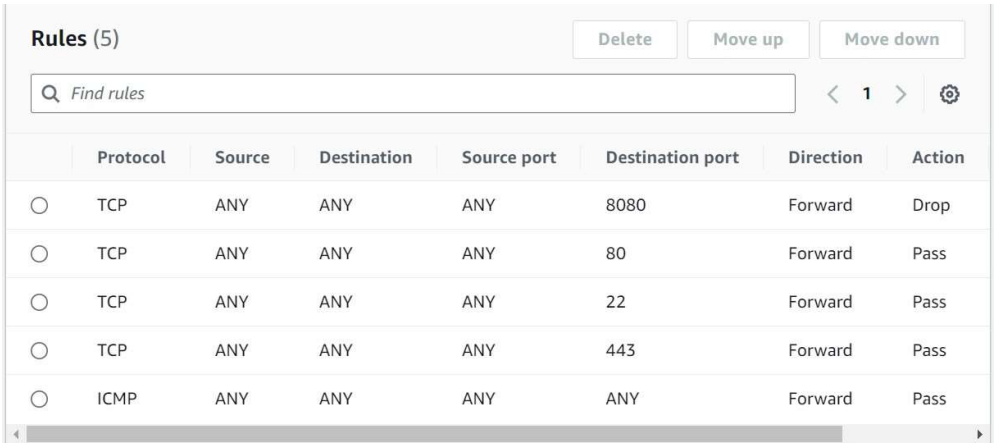
Attempt to access the public IP of WebServer2 and observe the logs generated in the newly created CloudWatch log group.



2024-05-07T20:55:48.000Z	{"firewall_name":"NetworkFirewall","availability_zone":"us-east-1a","event_timestamp"…
2024-05-07T20:56:07.000Z	{"firewall_name":"NetworkFirewall","availability_zone":"us-east-1a","event_timestamp"…
2024-05-07T20:56:08.000Z	{"firewall_name":"NetworkFirewall","availability_zone":"us-east-1a","event_timestamp"…
2024-05-07T20:56:08.000Z	{"firewall_name":"NetworkFirewall","availability_zone":"us-east-1a","event_timestamp"…
2024-05-07T20:56:15.000Z	{"firewall_name":"NetworkFirewall","availability_zone":"us-east-1a","event_timestamp"…
2024-05-07T20:56:21.000Z	{"firewall_name":"NetworkFirewall","availability_zone":"us-east-1a","event_timestamp"…
2024-05-07T20:56:24.000Z	{"firewall_name":"NetworkFirewall","availability_zone":"us-east-1a","event_timestamp"…
2024-05-07T20:56:29.000Z	{"firewall_name":"NetworkFirewall","availability_zone":"us-east-1a","event_timestamp"…
2024-05-07T20:56:35.000Z	{"firewall_name":"NetworkFirewall","availability_zone":"us-east-1a","event_timestamp"…
2024-05-07T20:56:39.000Z	{"firewall_name":"NetworkFirewall","availability_zone":"us-east-1a","event_timestamp"…
2024-05-07T20:56:39.000Z	{"firewall_name":"NetworkFirewall","availability_zone":"us-east-1a","event_timestamp"…

Task 2.10: Configure the firewall policy and test access

Create a stateful rule group and add 5 rules as per the specifications.



Rules (5)

Delete | Move up | Move down

Find rules

1

	Protocol	Source	Destination	Source port	Destination port	Direction	Action
○	TCP	ANY	ANY	ANY	8080	Forward	Drop
○	TCP	ANY	ANY	ANY	80	Forward	Pass
○	TCP	ANY	ANY	ANY	22	Forward	Pass
○	TCP	ANY	ANY	ANY	443	Forward	Pass
○	ICMP	ANY	ANY	ANY	ANY	Forward	Pass

Now, test multiple forms of access to the WebServer2 instance.
Browser access:

Netcat access:

```
voclabs:~/environment $  nc -vz 44.223.161.213 22
Ncat: Version 7.50 ( https://nmap.org/ncat )
Ncat: Connected to 44.223.161.213:22.
Ncat: 0 bytes sent, 0 bytes received in 0.01 seconds.
```

Instance connect and commands:

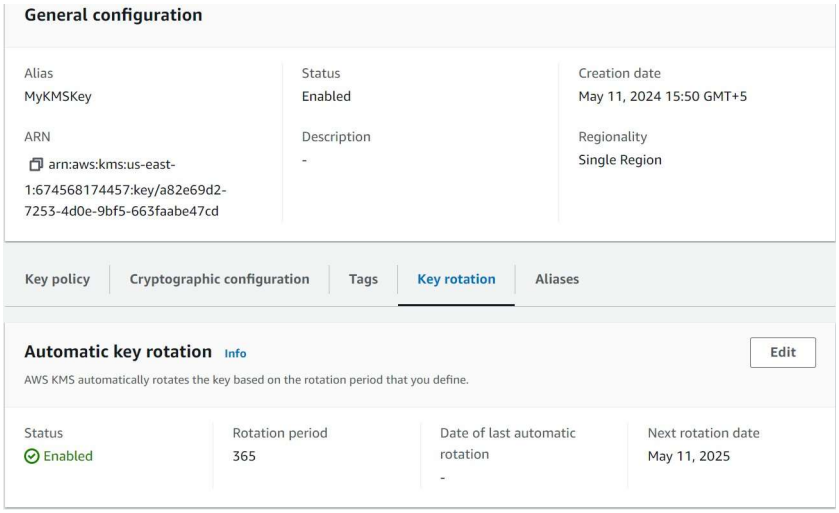
```
[ec2-user@webserver2 ~]$ ping -c 3 www.amazon.com
PING e15316.dsca.akamaiedge.net (23.202.154.76) 56(84) bytes of data.
64 bytes from a23-202-154-76.deploy.static.akamaitechnologies.com (23.202.154.76): icmp_seq=1 ttl=51
64 bytes from a23-202-154-76.deploy.static.akamaitechnologies.com (23.202.154.76): icmp_seq=2 ttl=51
64 bytes from a23-202-154-76.deploy.static.akamaitechnologies.com (23.202.154.76): icmp_seq=3 ttl=51

--- e15316.dsca.akamaiedge.net ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 2.573/3.148/4.138/0.702 ms
[ec2-user@webserver2 ~]$ sudo netstat -tulpn | grep -i listen
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      2155/sshd: /usr/sbi
tcp6       0      0 :::80                   :::*                    LISTEN      3743/httpd
tcp6       0      0 :::22                   :::*                    LISTEN      2155/sshd: /usr/sbi
[ec2-user@webserver2 ~]$
```

And finally, confirm that access through port 8080 is not available.

Task 3:

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

Create an AWS customer managed key, grants permissions to the voclabs role and turn on key rotation



AWS managed key

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

Update the AWS key policy and add a statement to the Principal section under “Allow Use of Key”

```
{
    "Sid": "Allow use of the key",
    "Effect": "Allow",
    "Principal": {
        "AWS": [
            "arn:aws:iam::674568174457:user/sofia",
            "arn:aws:iam::674568174457:role/voclabs"
        ]
    },
```

Allow key usage for Sofia

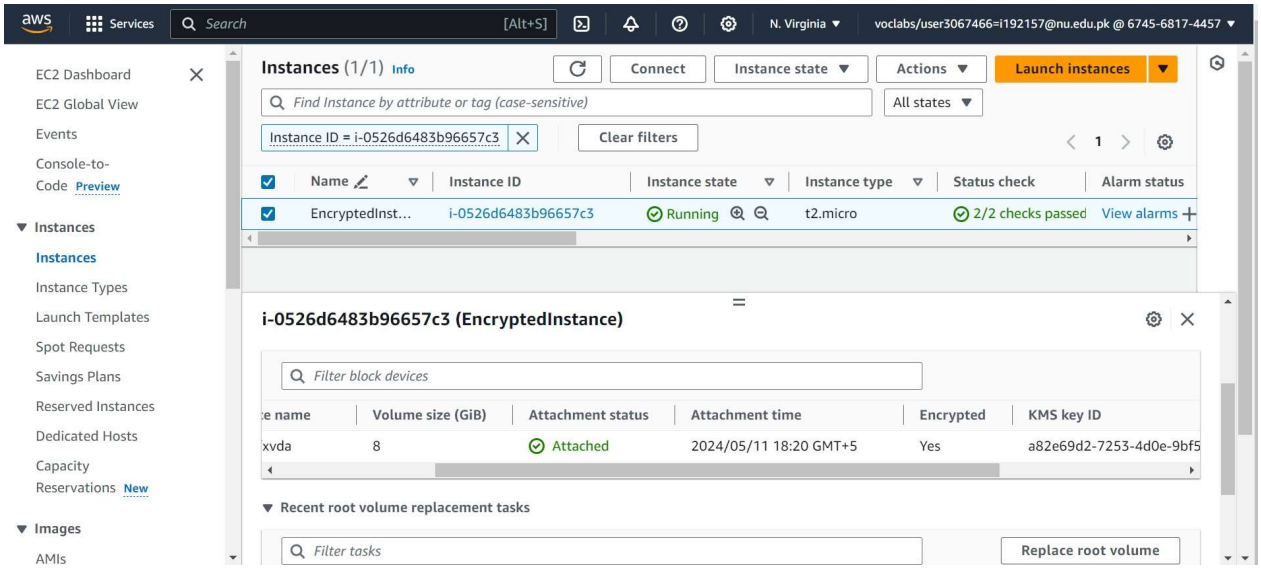
Analyze the given policy

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

Modify data-bucket to use SSE-KMS encryption. Log in as Sofia and upload a csv file to data-bucket. Check to see how the new encryption status on the file and confirm that it can be downloaded by Sofia. Now, upon trying to access the file as Paulo, the access is denied.

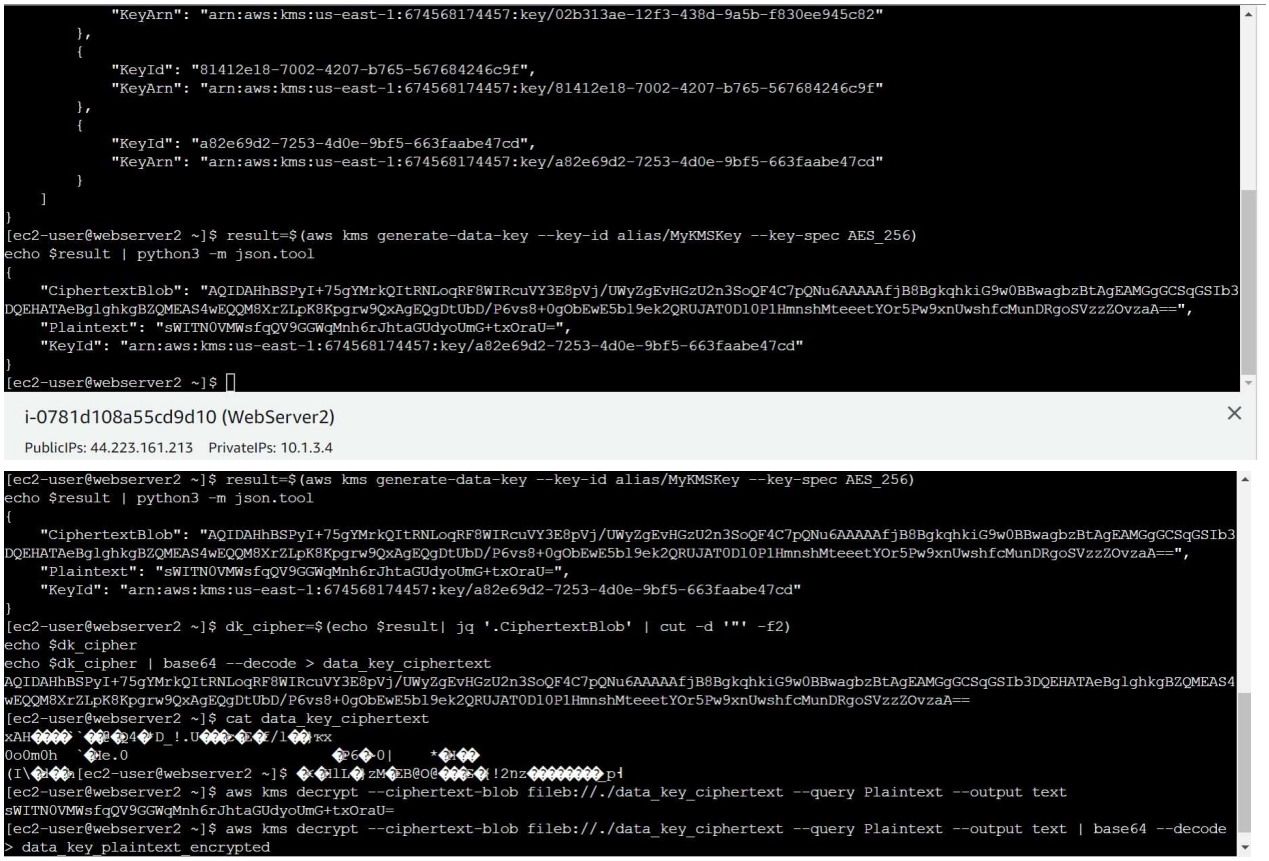
Task 3.4: Use AWS KMS to encrypt the root volume of an EC2 instance

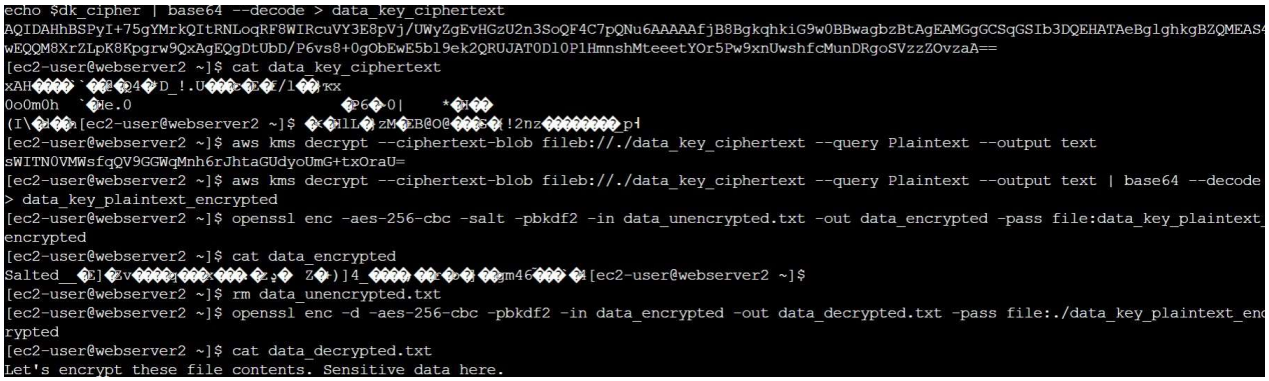
Create an Instance and configure the storage with the KMS key created earlier



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

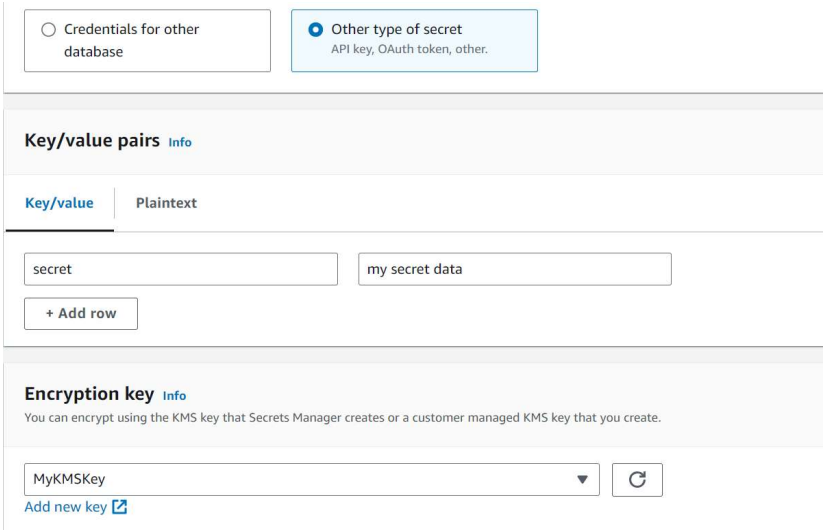
We Instance connect into the WebServer2 instance and create a text file with sample content in it. Afterwards, we run commands to configure the AWS Keys to encrypt this content, view the encrypted content and then decrypt it again to show the original content.





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

Use Secrets Manager to create a secret and encrypt it with our newly created key



Creating a secret and encrypting it using a key

Connect to the WebServer2 instance and fetch the secret

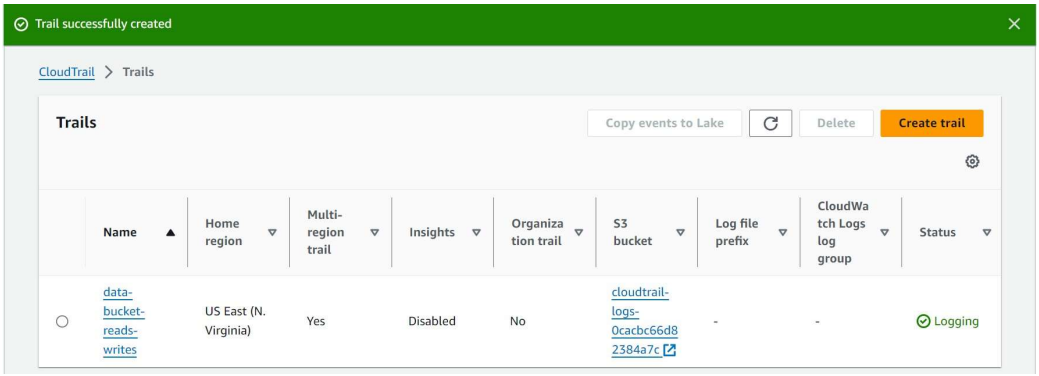
```
[ec2-user@webserver2 ~]$ aws secretsmanager get-secret-value --secret-id mysecret
{
    "ARN": "arn:aws:secretsmanager:us-east-1:674568174457:secret:mysecret-eVkX12",
    "Name": "mysecret",
    "VersionId": "6e649a2d-824b-4008-ac98-0058cc72afdc",
    "SecretString": "{\"secret\":\"my secret data\"}",
    "VersionStages": [
        "AWSCURRENT"
    ],
    "CreatedDate": "2024-05-11T17:11:59.036000+00:00"
```

Fetch secret

Task 4:

Task 4.1: Use CloudTrail to record Amazon S3 API calls

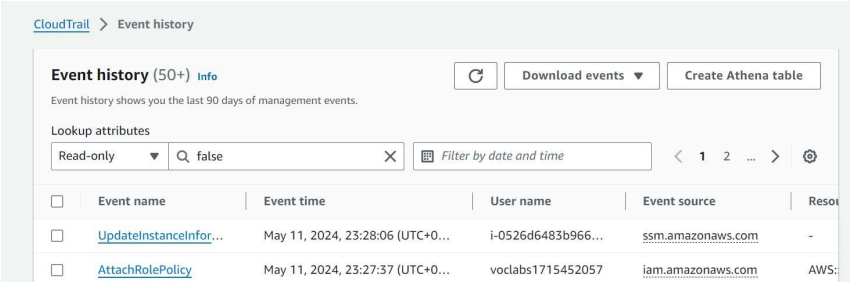
We need to create a Trail through CloudTrail console

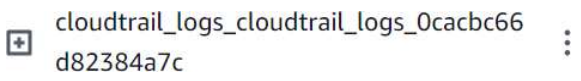


Creating a trail

Upload a csv dataset to data-bucket and open it in the S3 console to generate a cloudtrail log.

In the CloudTrail console, we go to event history and create an Athena table and select the cloudtrail-logs bucket for storage.



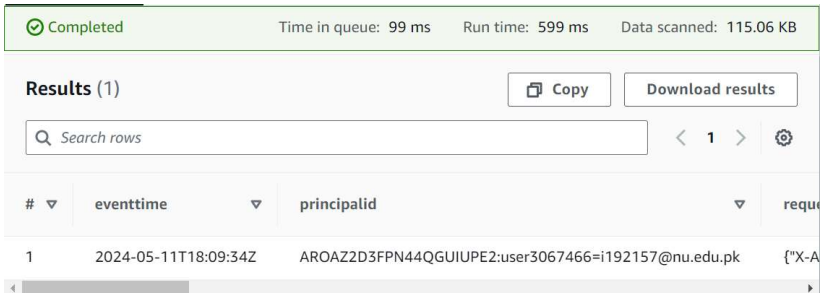


This resulting table and query generates cloudtrail logs in athena

1	1.09	{type=AWSService, principalid=null, arn=null, accountid=null, invokedby=cloudtrail.
2	1.09	{type=AWSService, principalid=null, arn=null, accountid=null, invokedby=cloudtrail.
3	1.09	{type=AWSService, principalid=null, arn=null, accountid=null, invokedby=cloudtrail.
4	1.08	{type=AssumedRole, principalid=AROAZ2D3FPN47CCWCHG77:i-0526d6483b9665
5	1.09	{type=AssumedRole, principalid=AROAZ2D3FPN44QGUIUPE2:user3067466=i19215
6	1.10	{type=AssumedRole, principalid=AROAZ2D3FPN44QGUIUPE2:user3067466=i19215
7	1.10	{type=AssumedRole, principalid=AROAZ2D3FPN44QGUIUPE2:user3067466=i19215
8	1.08	{type=AssumedRole, principalid=AROAZ2D3FPN44QGUIUPE2:user3067466=i19215
9	1.08	{type=AssumedRole, principalid=AROAZ2D3FPN44QGUIUPE2:user3067466=i19215
10	1.09	{type=AWSService, principalid=null, arn=null, accountid=null, invokedby=cloudtrail.

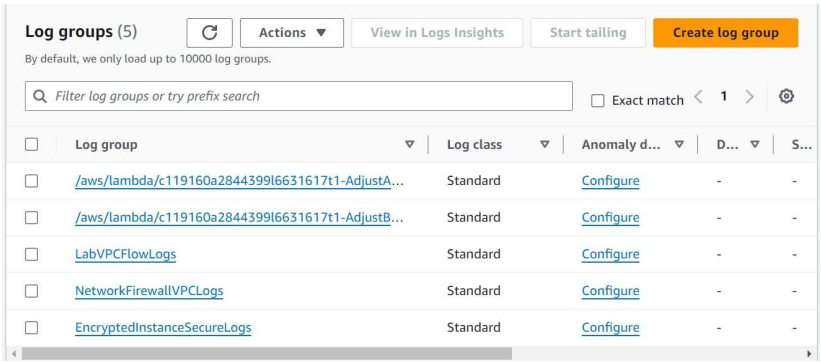
10 lines of logs

Run the provided query with the new table



Task 4.2: Use CloudWatch Logs to monitor secure logs

Create a log group names EncryptedInstanceSecureLogs



Connect to the EncryptedInstance and run commands
Commands to be run for the following tasks:

Install cloudwatch packages and a Linux daemon.
Download a template JSON file for a CloudWatch configuration template
Start CloudWatch agent and confirm it is active and running
View 10 lines of security logs

```
[ec2-user@ip-10-1-3-5 ~]$ sudo tail -f /var/log/secure
May 11 18:47:50 ip-10-1-3-5 sudo: pam_unix(sudo:session): session opened for user root by ec2-user(uid=0)
May 11 18:47:52 ip-10-1-3-5 sudo: pam_unix(sudo:session): session closed for user root
May 11 18:48:12 ip-10-1-3-5 sudo: ec2-user : TTY=pts/0 ; PWD=/home/ec2-user ; USER=root ; COMMAND=/sbin/service#040amazon-cloudwatch-agent#
040status
May 11 18:48:12 ip-10-1-3-5 sudo: pam_unix(sudo:session): session opened for user root by ec2-user(uid=0)
May 11 18:48:12 ip-10-1-3-5 sudo: pam_unix(sudo:session): session closed for user root
May 11 18:48:38 ip-10-1-3-5 sudo: ec2-user : TTY=pts/0 ; PWD=/home/ec2-user ; USER=root ; COMMAND=/bin/cat#040/opt/aws/amazon-cloudwatch-ag
ent/logs/amazon-cloudwatch-agent.log
May 11 18:48:38 ip-10-1-3-5 sudo: pam_unix(sudo:session): session opened for user root by ec2-user(uid=0)
May 11 18:48:38 ip-10-1-3-5 sudo: pam_unix(sudo:session): session closed for user root
May 11 18:48:50 ip-10-1-3-5 sudo: ec2-user : TTY=pts/0 ; PWD=/home/ec2-user ; USER=root ; COMMAND=/bin/tail#040-f#040/var/log/secure
May 11 18:48:50 ip-10-1-3-5 sudo: pam_unix(sudo:session): session opened for user root by ec2-user(uid=0)
```

Download the PEM file from the lab details and upload to the Cloud9IDE before running some commands that perform the following tasks:

Ssh into EncryptedInstance through C9IDE

```
voclabs:~/environment $ ssh -i labsuser.pem ec2-user@34.229.92.206
The authenticity of host '34.229.92.206 (34.229.92.206)' can't be established.
ECDSA key fingerprint is SHA256:yc5xXR3oR+4DnHxufqD4xGLFWyLLp2RW9Xv9qVkaVbM.
ECDSA key fingerprint is MD5:f9:eb:e9:a7:51:7a:69:10:ee:56:37:70:ea:c3:82:94.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '34.229.92.206' (ECDSA) to the list of known hosts.
Last login: Sat May 11 18:44:13 2024 from ec2-18-206-107-29.compute-1.amazonaws.com
   ,     #_
   ~\_  ####_        Amazon Linux 2
  ~~  \_#####\
  ~~     \###|       AL2 End of Life is 2025-06-30.
  ~~       \#/ ___
   ~~       V~' '->
    ~~~         /    A newer version of Amazon Linux is available!
      ~~._.   _/
         _/ _/       Amazon Linux 2023, GA and supported until 2028-03-15.
       _/m/'           https://aws.amazon.com/linux/amazon-linux-2023/

[ec2-user@ip-10-1-3-5 ~]$
```

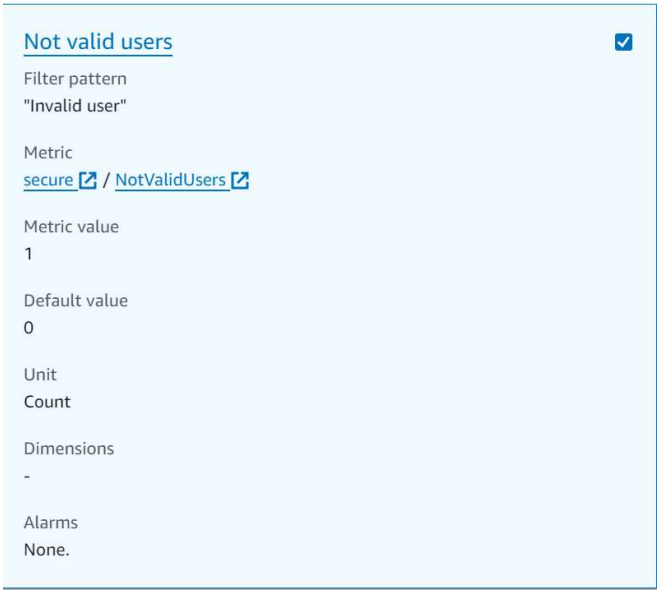
```
May 11 18:59:47 ip-10-1-3-5 sshd[3996]: pam_unix(sshd:session): session opened for user ec2-user by (uid=0)
```

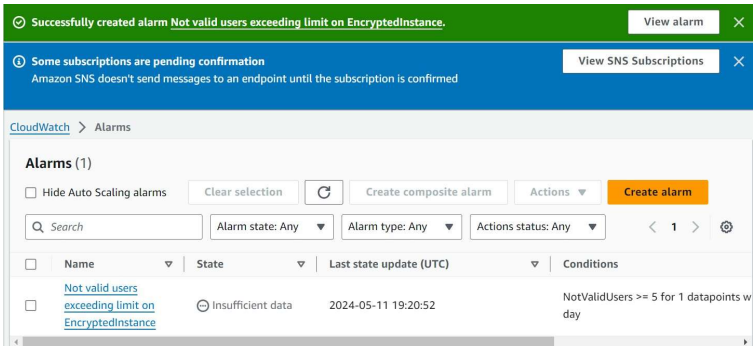
Similarly, initiate a failed ssh by using the wrong username “ubuntu”
Now, confirm log entries into the cloudwatch group

2024-05-11T18:59:47.268Z	May 11 18:59:47 ip-10-1-3-5 sshd[3996]: error: AuthorizedKeysCommand /opt/aws/bin/e…
2024-05-11T18:59:47.268Z	May 11 18:59:47 ip-10-1-3-5 sshd[3996]: Accepted publickey for ec2-user from 23.22.…
2024-05-11T18:59:51.952Z	May 11 18:59:47 ip-10-1-3-5 sshd[3996]: pam_unix(sshd:session): session opened for …
2024-05-11T19:01:34.554Z	May 11 19:01:34 ip-10-1-3-5 sshd[4186]: Received disconnect from 23.22.18.31 port 4…
2024-05-11T19:01:34.554Z	May 11 19:01:34 ip-10-1-3-5 sshd[4186]: Disconnected from 23.22.18.31 port 48092
2024-05-11T19:01:38.951Z	May 11 19:01:34 ip-10-1-3-5 sshd[3996]: pam_unix(sshd:session): session closed for …
2024-05-11T19:01:53.601Z	May 11 19:01:53 ip-10-1-3-5 sshd[4232]: Invalid user ubuntu from 23.22.18.31 port 4…
2024-05-11T19:01:53.601Z	May 11 19:01:53 ip-10-1-3-5 sshd[4232]: input_userauth_request: invalid user ubuntu…
2024-05-11T19:01:57.951Z	May 11 19:01:53 ip-10-1-3-5 sshd[4232]: Connection closed by 23.22.18.31 port 46666…

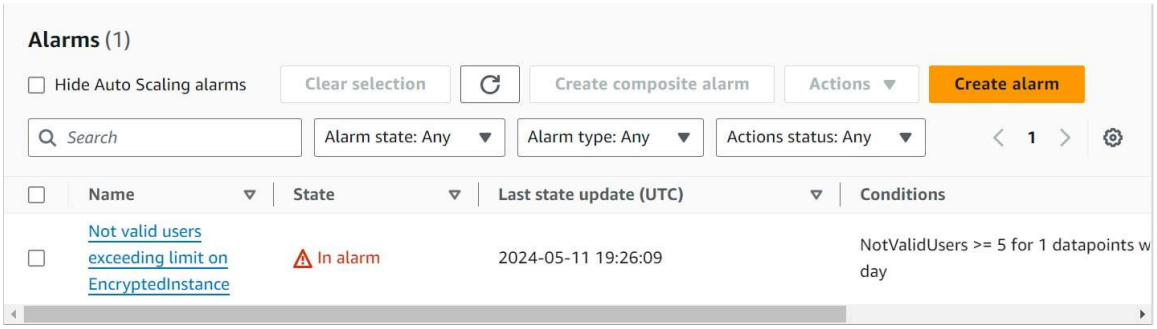
Task 4.3: Create a CloudWatch alarm to send notifications for security incidents

Go to the EncryptedInstanceSecureLogs CloudWatch log group and create a metric filter with some metrics. Also, create a CloudWatch alarm for a 1 day period

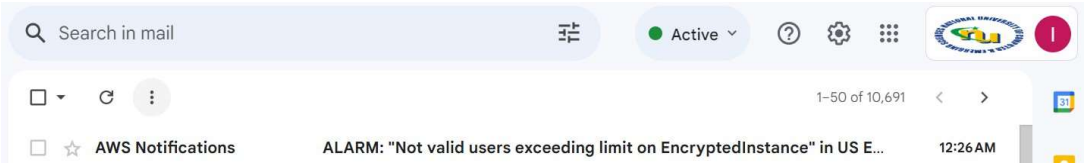




Confirm the subscription through email and test alarm through at least 5 Cloud9IDE invalid ssh requests. This causes the alarm to trigger



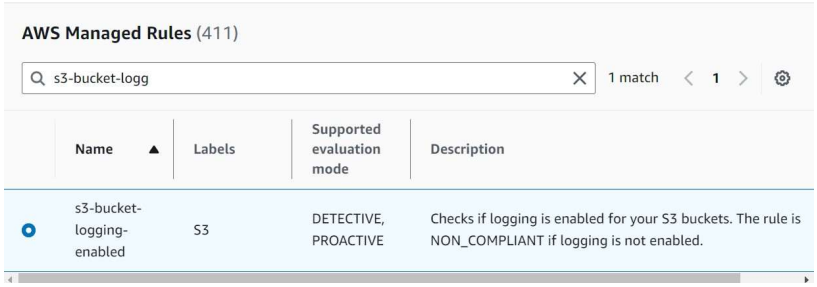
Confirm that the email was received



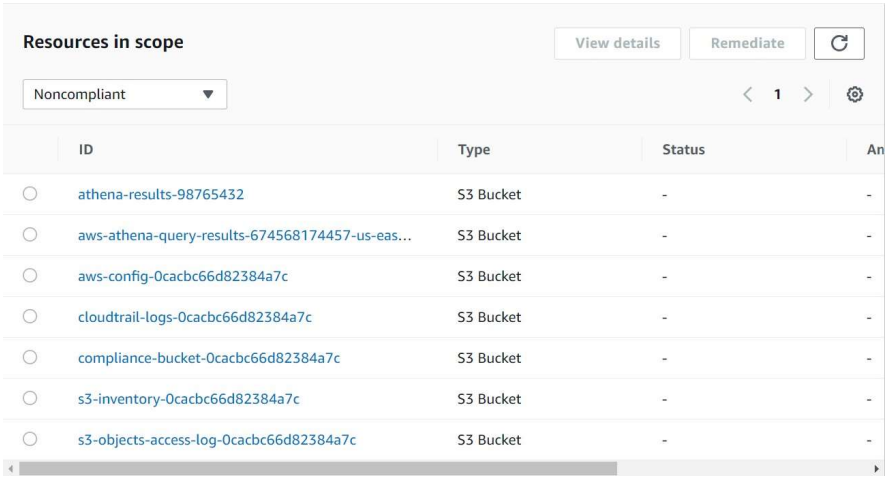
Task 4.4: Configure AWS Config to assess security settings and remediate the configuration of AWS resources

Review roles created by the lab to grants various permissions. Create an S3 bucket.

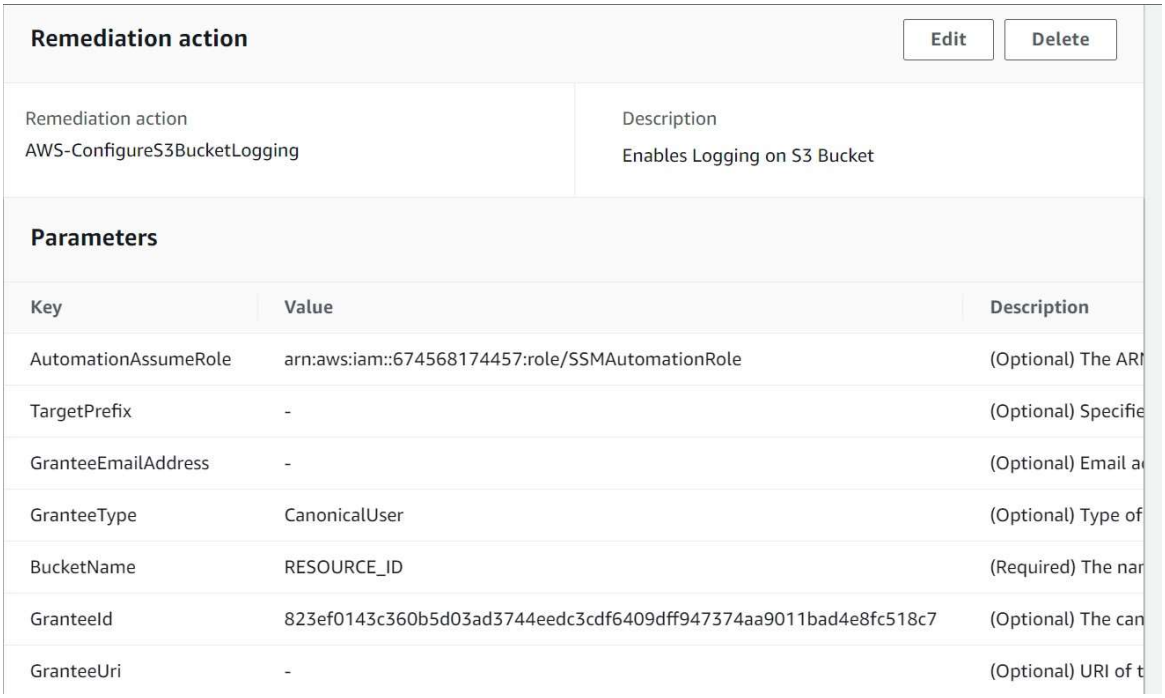
Now, change the object ownership settings of the objects-access-logs bucket to enable ACLs. Go to AWS Config console and set up a config. Then, set up a managed rule.



Noncompliant buckets according to the role:



The compliance bucket is listed under noncompliant due to server access logging being disabled. We configure manual remediation under the AWS Config we created to fix this.



Now, choose compliance-bucket from within the role details and remediate it.

The action was successfully executed.