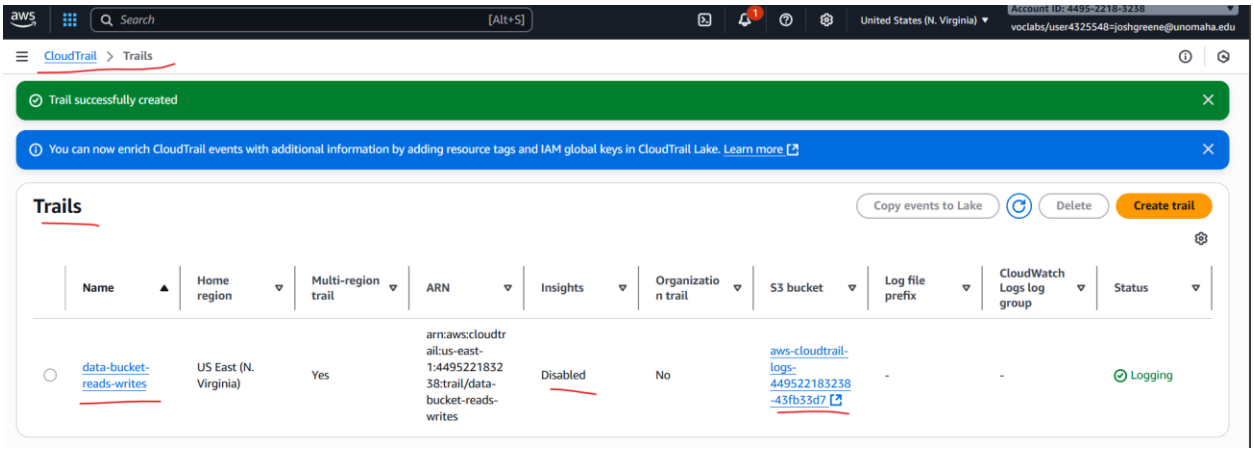
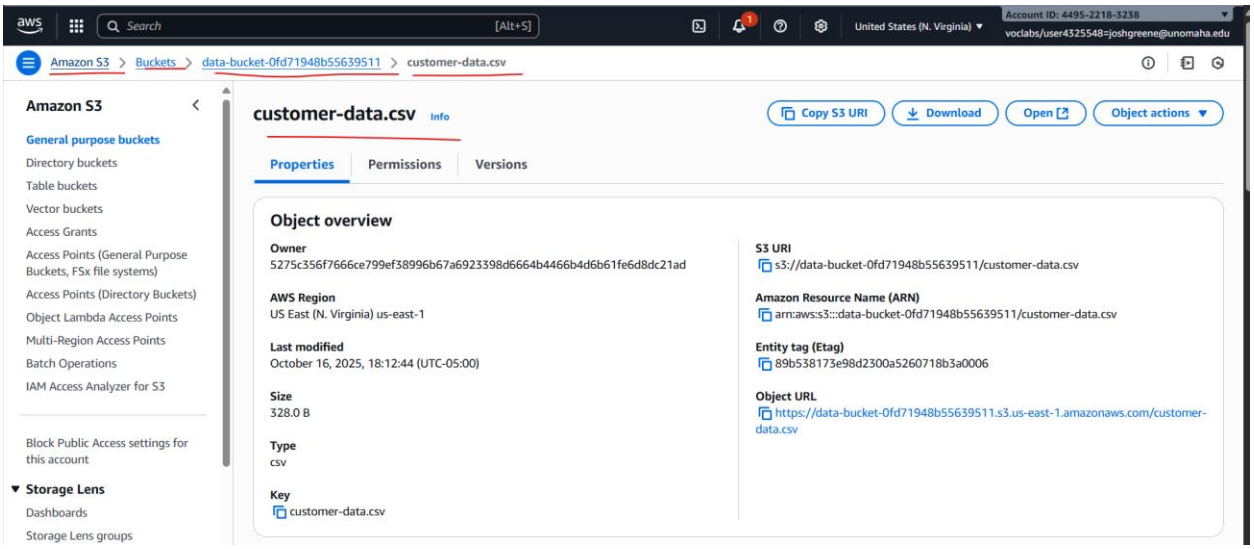
Monitoring and Logging with AWS CloudTrail, CloudWatch, and AWS Config

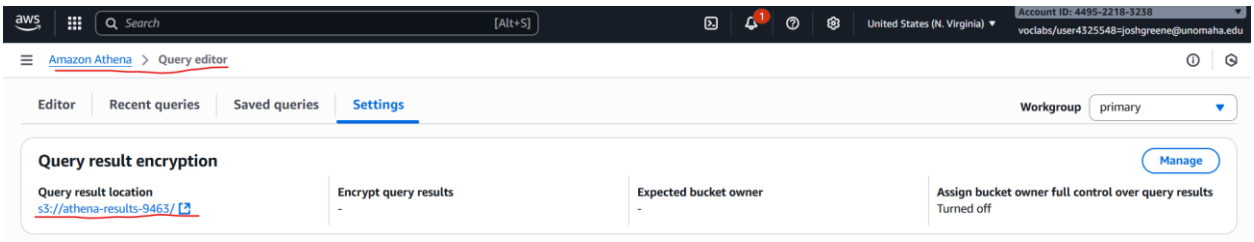
Task 1.1: Use CloudTrail to record Amazon S3 API calls



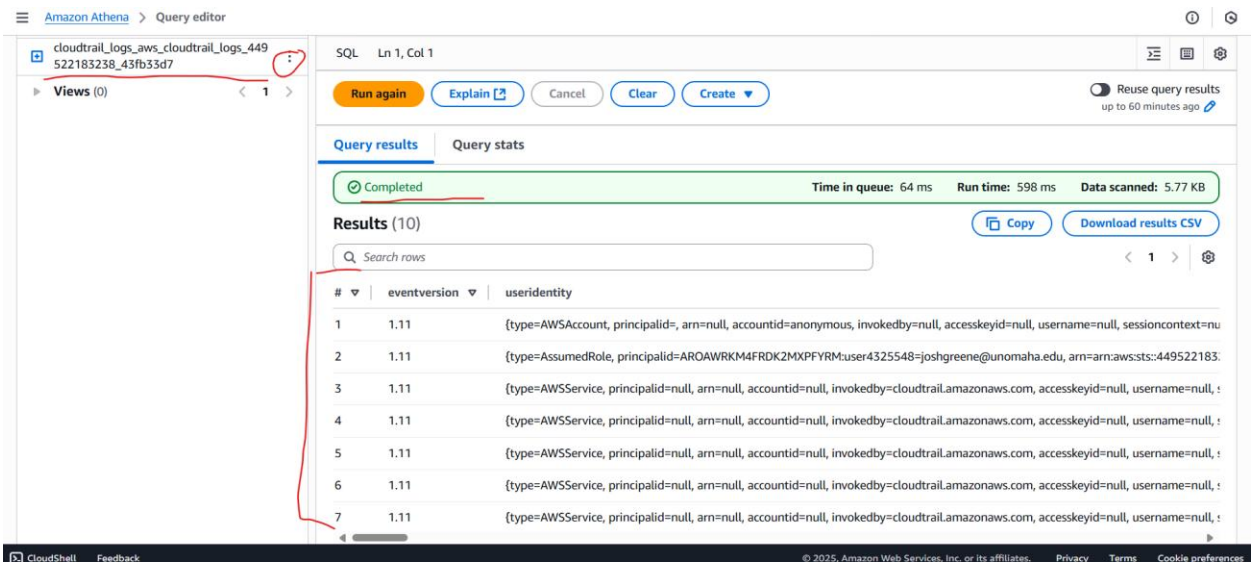
I made a CloudTrail trail that writes logs to cloudtrail-logs, with management and S3 data events turned on so puts/gets on S3 will be captured.



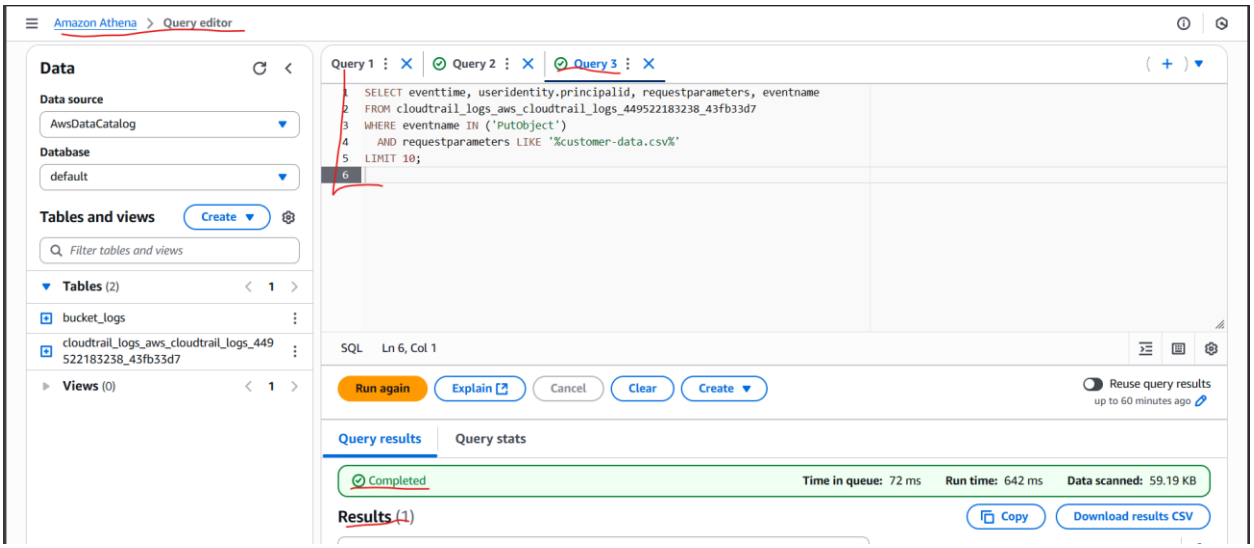
I uploaded customer-data.csv to data-bucket and opened it, which creates PutObject and GetObject events that CloudTrail will log.



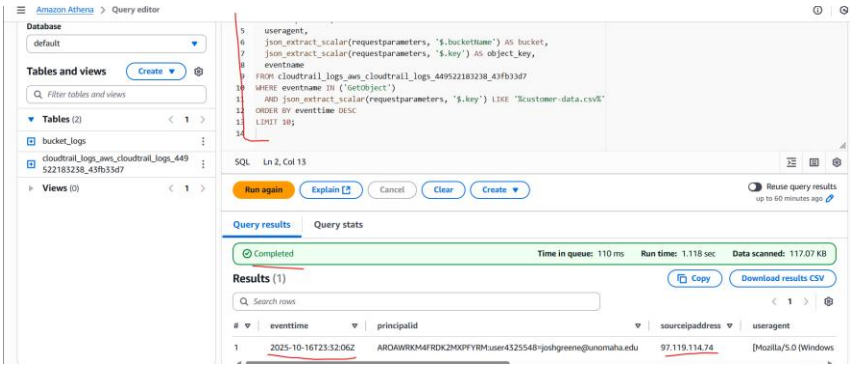
I used CloudTrail's built-in helper to create an Athena table over the cloudtrail-logs bucket so I can query CloudTrail JSON logs with SQL.



Here I previewed my CloudTrail table in Athena to make sure logs from my earlier S3 actions had been delivered. Once I saw data appear in the preview results, I knew my Athena table was correctly reading log files from the CloudTrail S3 bucket.

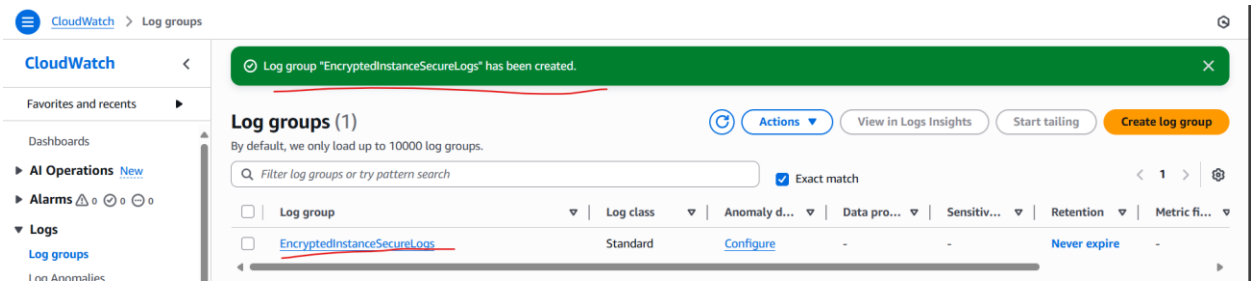


Here I ran a query in Athena to find the PutObject event for when I uploaded *customer-data.csv* to my S3 bucket. The results showed the user who performed the action, the time it occurred, and other request details like the bucket and file name.

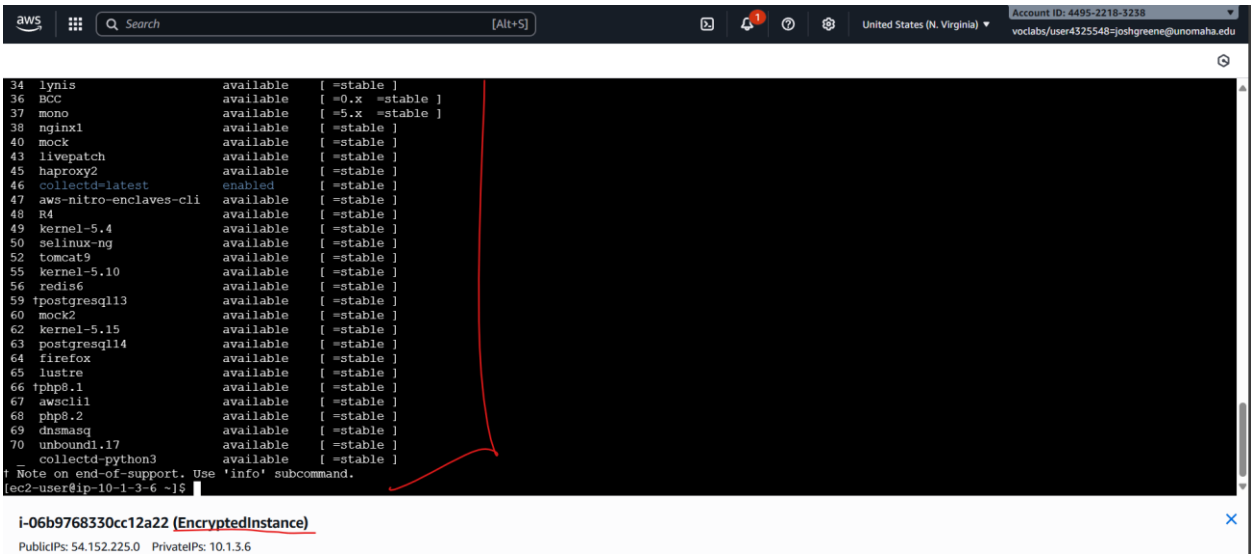


Here I created and ran a query to find the GetObject event, which records when the *customer-data.csv* file was opened or downloaded. The results showed who accessed the file, from which IP address, and which browser or client was used. Reviewing these results confirmed that CloudTrail successfully logged both upload and access activity, giving a complete record of when the file was added, who retrieved it, and how it was accessed.

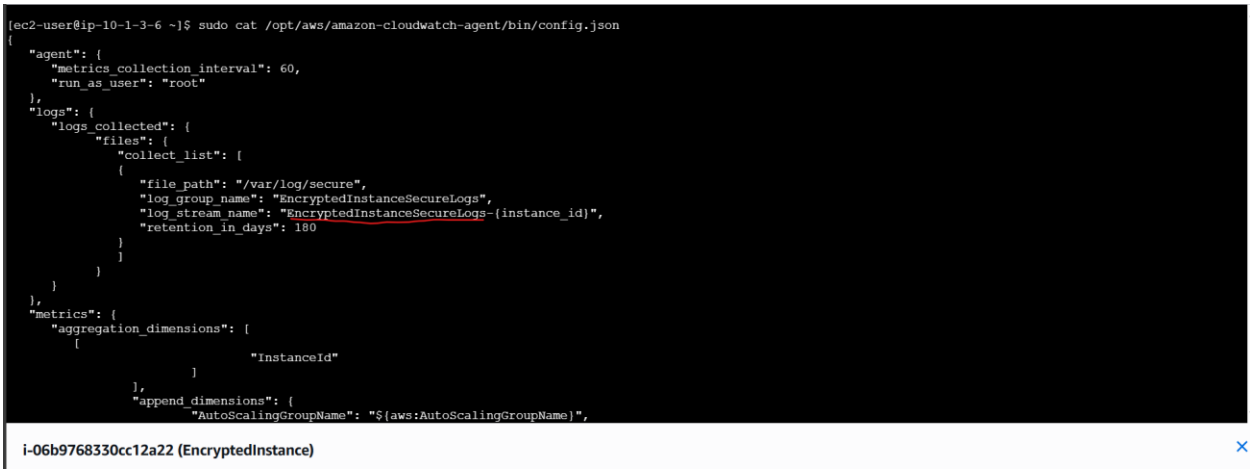
Task 1.2: Use CloudWatch Logs to monitor secure logs



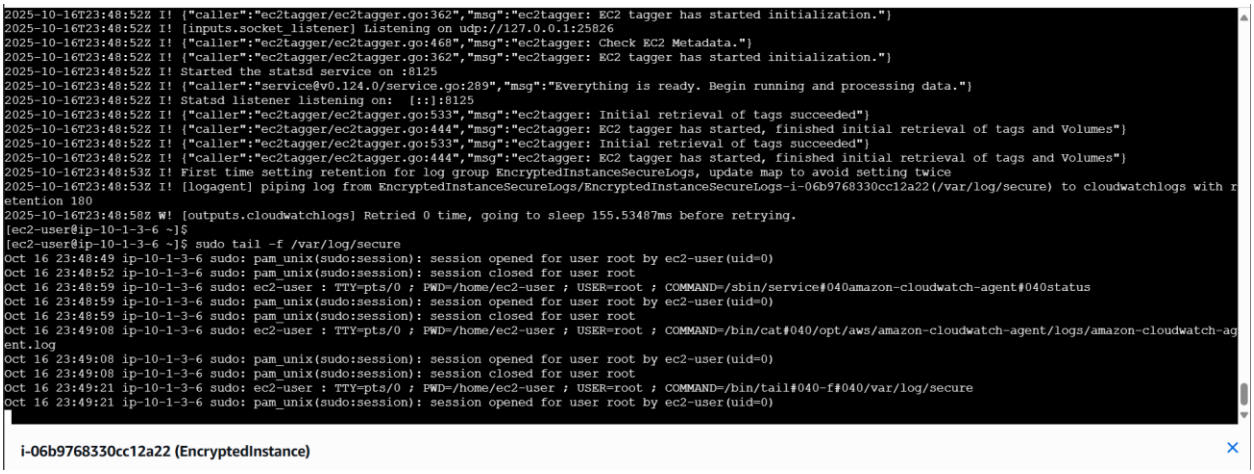
Here I made a new CloudWatch log group called *EncryptedInstanceSecureLogs*. This is where system and SSH logs from my EncryptedInstance will be sent and stored for monitoring.



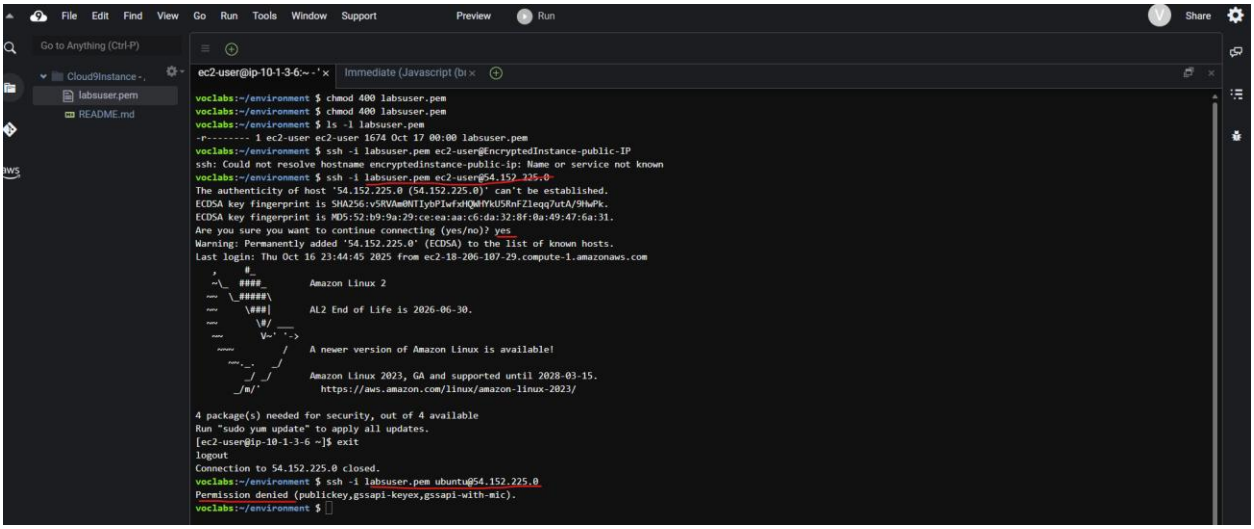
I connected to my EncryptedInstance and installed both the CloudWatch agent and collectd, which are needed for sending log data from the instance to CloudWatch.



Here I downloaded the CloudWatch agent configuration file. It tells the agent to forward security log entries from /var/log/secure on my EC2 instance to the *EncryptedInstanceSecureLogs* group in CloudWatch.

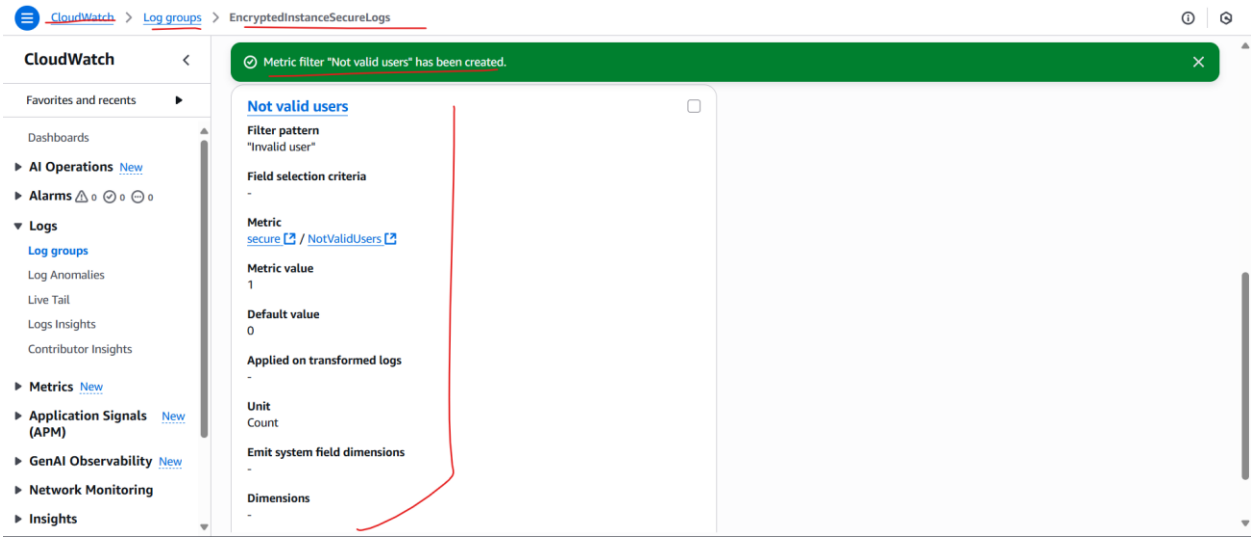


I started the CloudWatch agent, confirmed it was active, and verified that it was piping secure log entries to CloudWatch. I used the tail -f command to watch the live /var/log/secure file so I could see logs appear as actions occur.

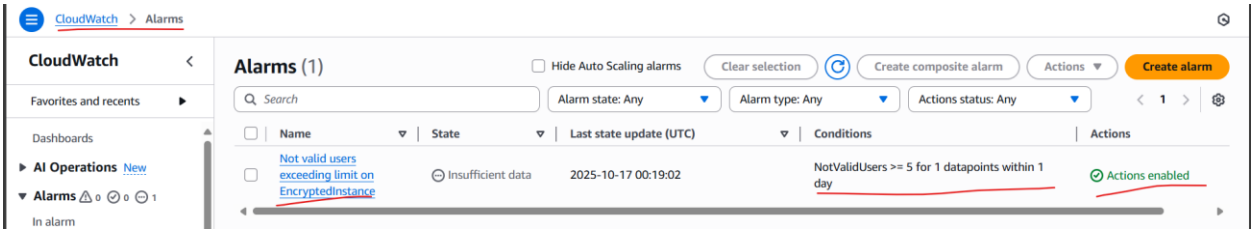


I used my PEM key to SSH into my EncryptedInstance at 54.152.225.0, confirmed a successful login, then disconnected and tried again with the wrong username to generate a failed login. Both actions created security logs that the CloudWatch agent is now forwarding to my *EncryptedInstanceSecureLogs* group in CloudWatch.

Task 1.3: Create a CloudWatch alarm to send notifications for security incidents

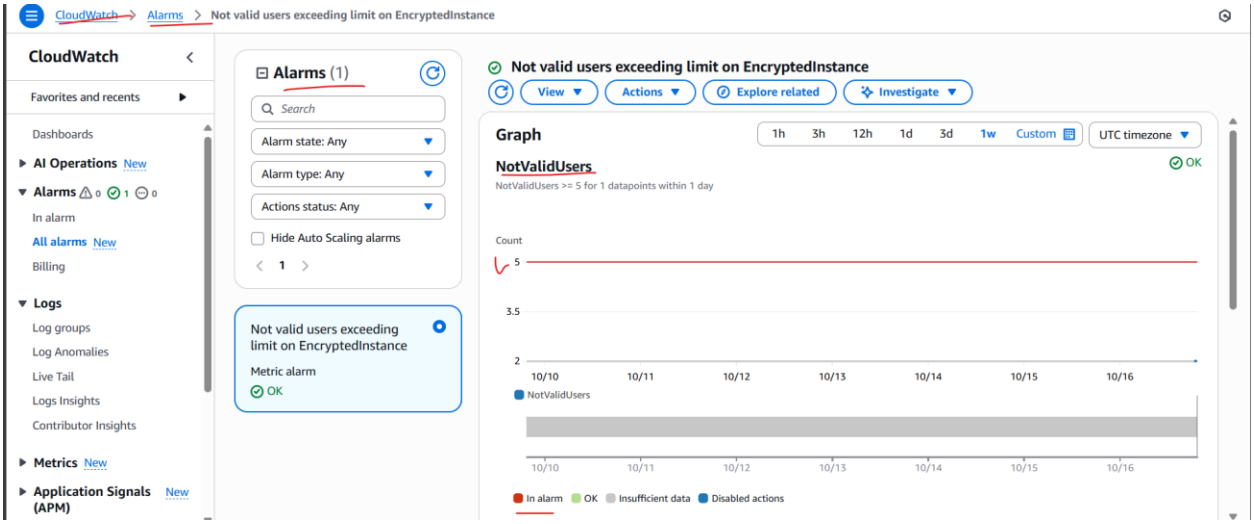


Here I created a metric filter in CloudWatch that searches the secure logs for the phrase *"Invalid user"*. Each time that message appears, it records a metric value of 1 under the secure namespace, which will let me trigger alarms when too many failed SSH logins occur.



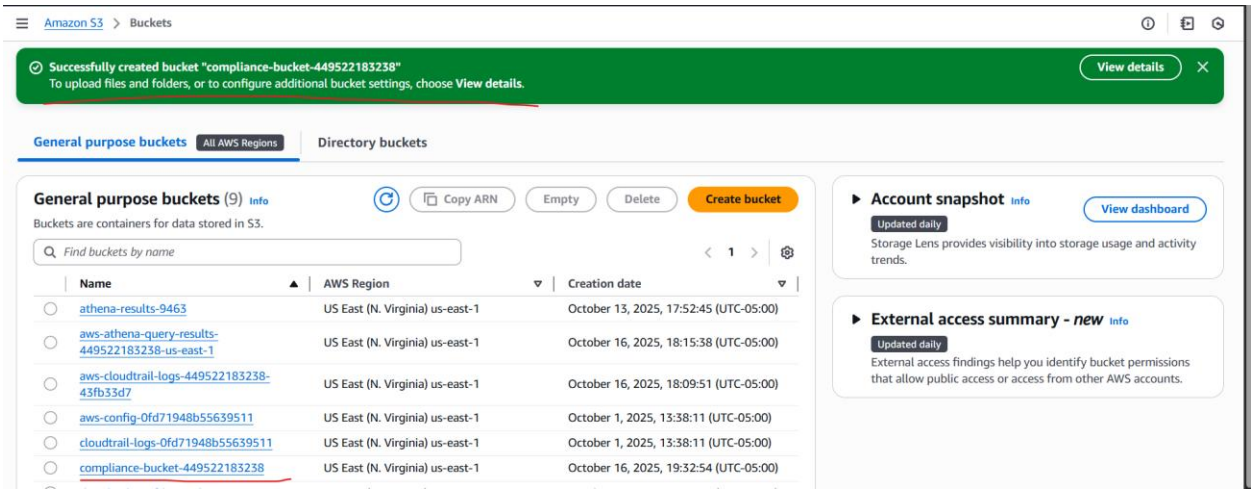
Here I created a CloudWatch alarm that watches the *NotValidUsers* metric. If five or more invalid SSH logins happen in one day, it changes to *In alarm* state and sends an SNS notification email to alert the security team. I also confirmed my SNS email subscription so the CloudWatch alarm can actually send me notifications when triggered.

I purposely made five failed SSH logins using the ubuntu username to generate *Invalid user* messages in the instance's secure log so the alarm metric increases and crosses the threshold.

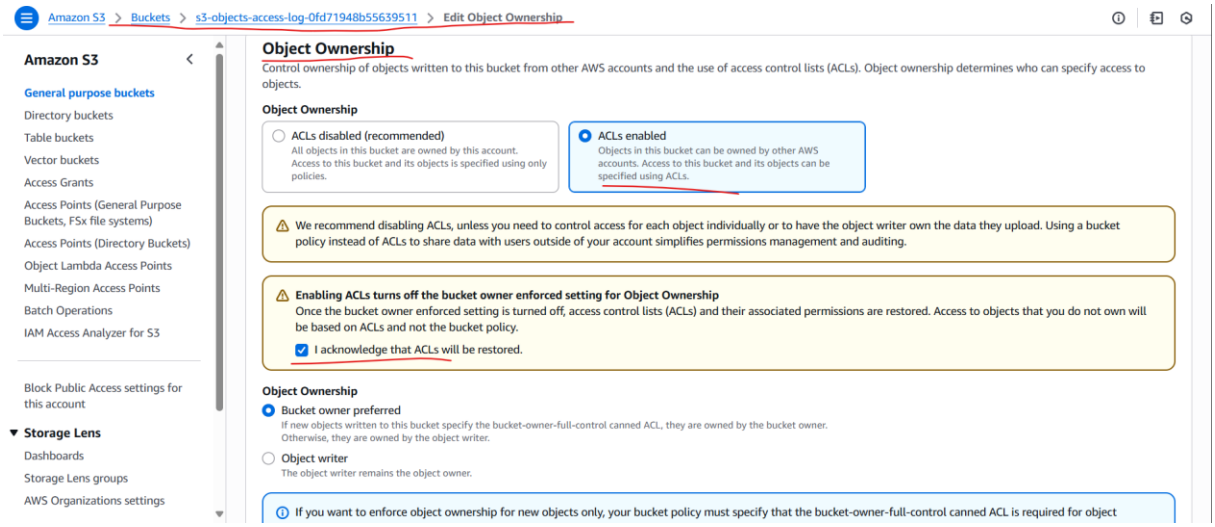


I confirmed that my failed SSH attempts appeared in the secure log, the *Not valid users* alarm switched to *In alarm* state, and an SNS email notification was delivered to my inbox. This proves the alerting system works as intended.

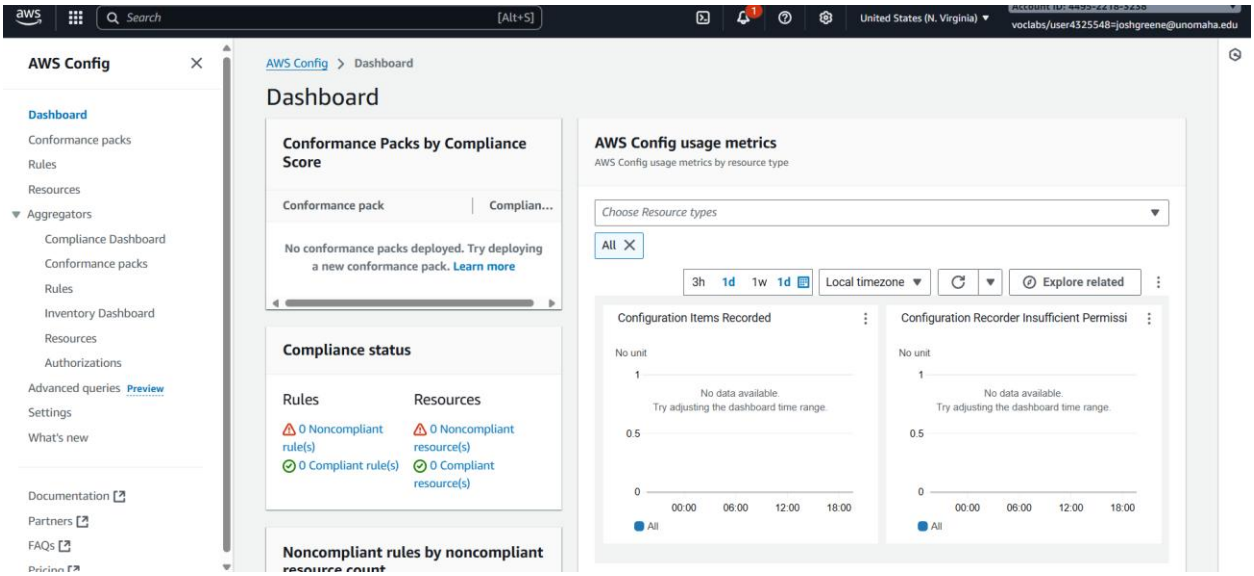
Task 1.4: Configure AWS Config to assess security settings and remediate the configuration of AWS resources



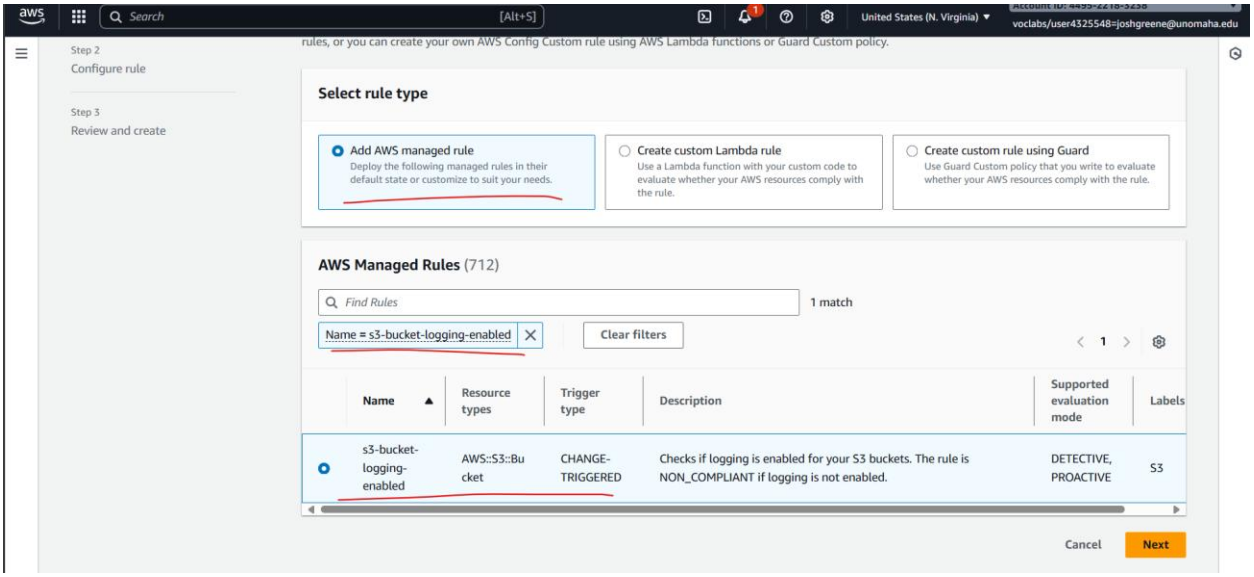
I created a new bucket named *compliance-bucket-449522183238* in us-east-1 with ACLs enabled so it can be checked for logging compliance.



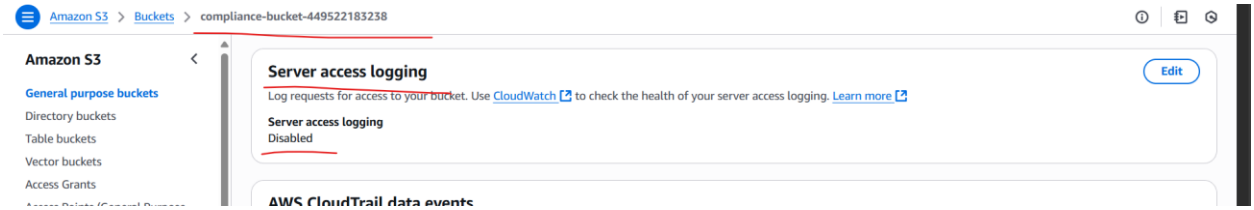
I re-enabled ACLs on the log bucket so AWS Config's remediation automation can write access logs successfully.



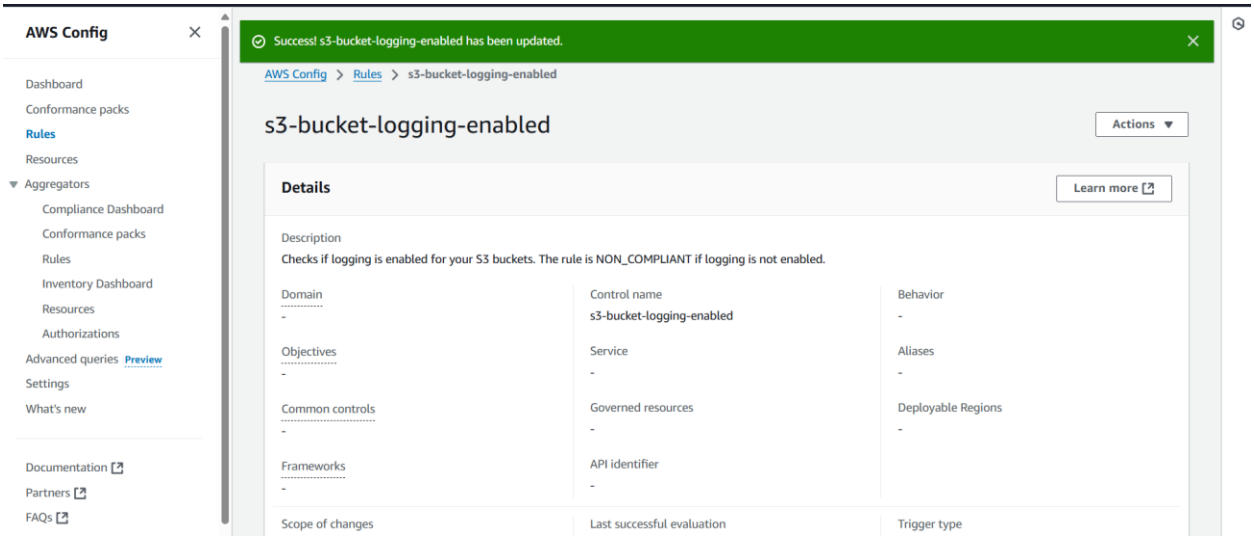
I enabled AWS Config to record and track changes for all resources in the region using the predefined role and log bucket.



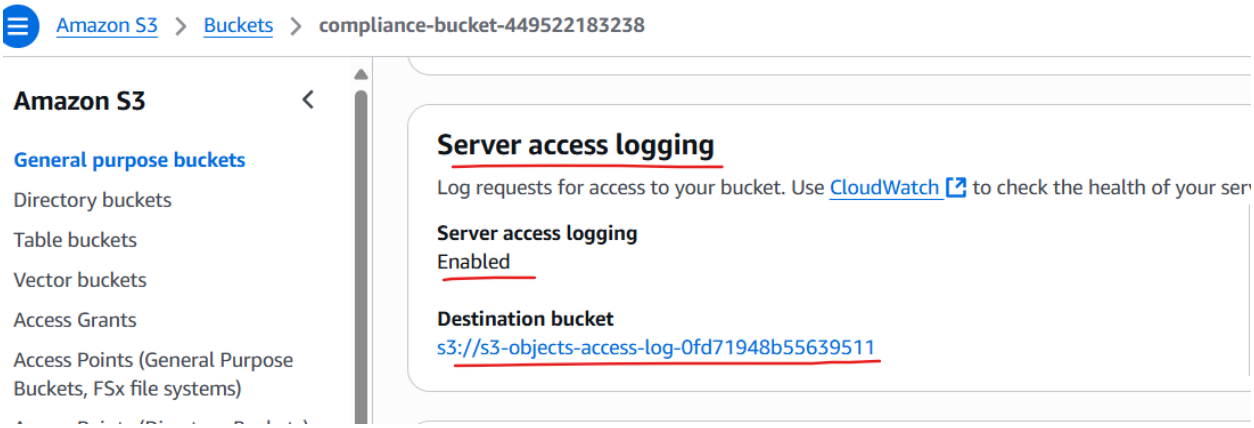
I added a managed rule that checks if each S3 bucket has server access logging turned on.



AWS Config flagged my compliance bucket as noncompliant since logging was off, confirming that the rule works.



I set up manual remediation using the AWS Systems Manager runbook AWS-ConfigureS3BucketLogging to automatically turn on access logging for any noncompliant bucket.



After setting up the remediation rule in AWS Config, I went back to the rule details page and scrolled down to the section that listed all the buckets being checked. I found my compliance-bucket listed as noncompliant and clicked the option to remediate it. Once the remediation started, I waited a short time and refreshed the page until the status changed to show it was compliant. From there, I went to the S3 console, opened my compliance-bucket, and checked under the Properties tab. When I scrolled down to server access logging, it showed as enabled and pointed to my logging bucket named s3-objects-access-log-449522183238. This confirmed that the remediation worked and that AWS Config successfully turned on logging for the bucket automatically.

Project Summary:

In this project, I set up AWS Config to automatically check and fix security settings for S3 buckets to meet company compliance standards. I started by reviewing the IAM roles and existing S3 buckets that AWS Config and Systems Manager would use to monitor and remediate issues. Then I created a new bucket called compliance-bucket-449522183238, which AWS Config would evaluate. I also made sure ACLs were enabled on the logging bucket so AWS would be able to write logs into it later. After that, I set up AWS Config to record all resources in the region and added the managed rule called s3-bucket-logging-enabled, which checks if server access logging is turned on for S3 buckets.

Once the rule was active, I confirmed that my compliance bucket showed up as noncompliant because logging wasn't yet enabled. I then configured a manual remediation action that used the AWS-ConfigureS3BucketLogging automation document. This allowed AWS to automatically fix any bucket that didn't have logging enabled by turning it on and sending logs to my log bucket. After running the remediation, I verified in the S3 console that my compliance bucket was now compliant, with server access logging successfully enabled. This showed that AWS Config and the remediation action were working together to automatically enforce security standards.