SOC 2 Readiness & Compliance Remediation Project
Executive Summary

Objective:
The objective of this SOC 2 Readiness and Remediation project is to assess the organisation's current cloud and operational environment against SOC 2 Trust Services Criteria, identify areas of non-compliance, and provide a prioritised remediation plan. This initiative ensures the company is prepared for a successful SOC 2 audit and demonstrates commitment to security, availability, confidentiality, processing integrity, and privacy.

Why This Matters:
SOC 2 compliance is not just a regulatory checkbox — it's a competitive differentiator that builds customer trust and enables access to enterprise-level contracts. Failing an audit can lead to lost opportunities, reputational damage, and increased security risk. By addressing potential gaps proactively, the company can minimise audit findings, avoid costly delays, and strengthen its security posture.

Business Value Delivered:

- Strengthened security posture aligned to SOC 2 criteria.
- Increased client trust and credibility, enabling enterprise deals.
- Reduced audit readiness time and costs.
- Mitigation of security and compliance risks.
- Improved internal processes and governance.

Approach:

1. Conducted an initial review of systems and configurations to identify SOC 2-relevant gaps.
2. Documented misconfigurations, missing controls, and audit risks through screenshots and technical notes.
3. Mapped findings against SOC 2 Trust Services Criteria.
4. Developed a remediation plan with clear steps, prioritisation, and responsible parties.
5. Prepared supporting evidence and documentation to demonstrate compliance to auditors.

Key Outcomes:

• Identified and documented all observed gaps and issues impacting SOC 2 readiness, each with corresponding remediation steps.

• Created a clear remediation plan with evidence for each issue.

• Produced a deliverable that can be provided directly to clients and auditors.

• Enabled a faster, smoother audit process.

Strategic Alignment:
This project directly supports the company's strategic goal of securing high-value enterprise contracts by meeting

SOC 2 compliance requirements. It also aligns with the organisation's long-term commitment to security, operational excellence, and regulatory compliance.

Next Steps for Client Engagement:

1. Review documented findings with key stakeholders.
2. Begin prioritised remediation plan as outlined in the report.
3. Conduct post-remediation validation checks and evidence collection.
4. Engage a SOC 2 audit firm for readiness assessment or Type 1 audit.
5. Establish ongoing compliance monitoring and reporting to ensure sustained adherence.

Step 1 – Create Insecure Security Group via AWS CLI

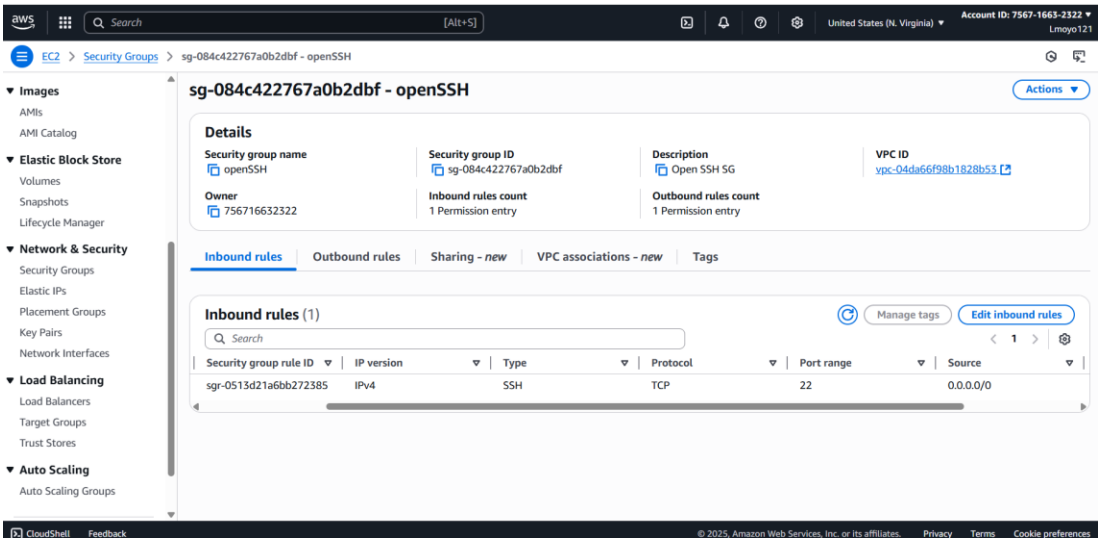
The screenshot shows the creation of a new EC2 security group named SOC2-Insecure-SG using the AWS CLI. The description explicitly states "SOC2 demo – open to world" to indicate this was intentionally created as a misconfigured resource for demonstration purposes. This step establishes a baseline insecure configuration that violates SOC 2 access control principles.

Step 2 – Add Open-to-World Inbound Rule via AWS CLI

The screenshot displays the authorisation of an inbound rule on the SOC2-Insecure-SG security group, allowing TCP traffic on port 22 (SSH) from any IPv4 address (0.0.0.0/0). This configuration represents a critical security risk under SOC 2, as it allows unrestricted internet access to administrative services. The change was deliberately applied to showcase non-compliance detection

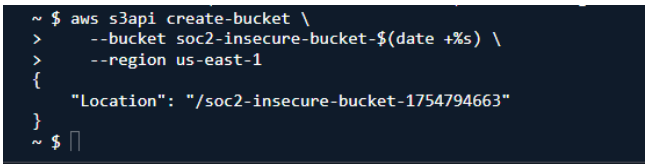
Step 3 – Security Group with Unrestricted SSH in AWS Console

The screenshot from the EC2 Security Groups console confirms the presence of an inbound SSH rule (TCP port 22) with a source of 0.0.0.0/0. This visual evidence verifies that the misconfiguration applied in Step 2 is active. Under SOC 2, such an open access rule would be flagged as a severe security vulnerability requiring immediate remediation.



Step 4 – Create Insecure S3 Bucket via AWS CLI

The screenshot shows the creation of an S3 bucket named using the format soc2-insecure-bucket-<timestamp> via the AWS CLI. This bucket was intentionally created without secure naming conventions or access restrictions to serve as a misconfigured storage resource for the SOC 2 demo. Such a bucket, if left unsecured, can lead to confidentiality breaches.



Step 5 – Disable Public Access Block Settings on S3 Bucket

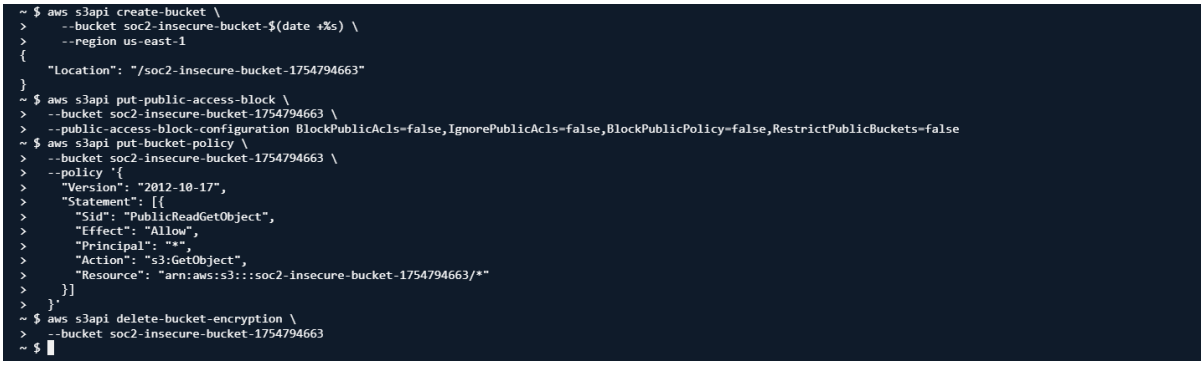
The screenshot displays AWS CLI commands that explicitly disable all public access block settings (BlockPublicAcls, IgnorePublicAcls, BlockPublicPolicy, RestrictPublicBuckets) for the newly created bucket. This deliberate change allows public ACLs and bucket policies to be applied, creating a high-risk configuration in violation of SOC 2 confidentiality and privacy principles.

Step 6 - Apply Public Read Policy to S3 Bucket

The screenshot shows the attachment of a bucket policy granting public read access (s3:GetObject) to all principals (Principal: *) for every object in the bucket. This misconfiguration, combined with the disabled public access blocks from Step 6, results in unrestricted public access to stored data — a severe breach of SOC 2 security and confidentiality requirements, demonstrated intentionally for this exercise.

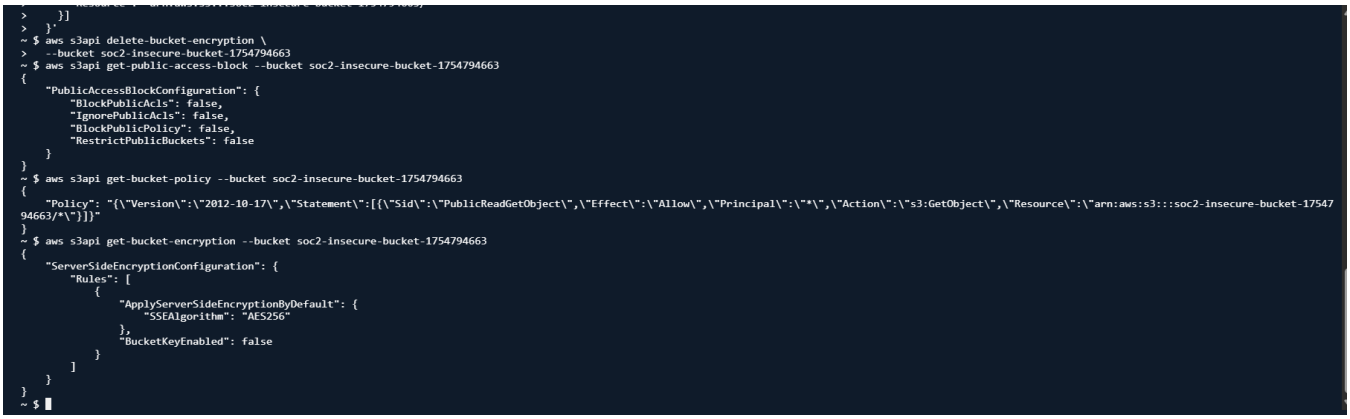
Step 7 – Remove Encryption from Publicly Accessible S3 Bucket

The screenshot shows AWS CLI commands used to further weaken the security of the previously created soc2-insecure-bucket by disabling server-side encryption. This was done after applying a public-read policy and disabling all public access block settings. Removing encryption from a bucket already configured for unrestricted public access maximises the risk of unauthorised data exposure, breaching SOC 2 confidentiality and processing integrity requirements. This misconfiguration was intentional for demonstration purposes.



Step 8 – Verify Insecure Bucket Configuration via AWS CLI

Used AWS CLI commands to check the public access block configuration, bucket policy, and encryption status of the soc2-insecure-bucket. The output confirmed that all public access blocks were disabled, a public-read policy was in place, and server-side encryption (AES256) was disabled. This verification step provided evidence of the deliberate misconfiguration for SOC 2 audit demonstration.



Step 9 - Re-enable Public Access Block on S3 Bucket

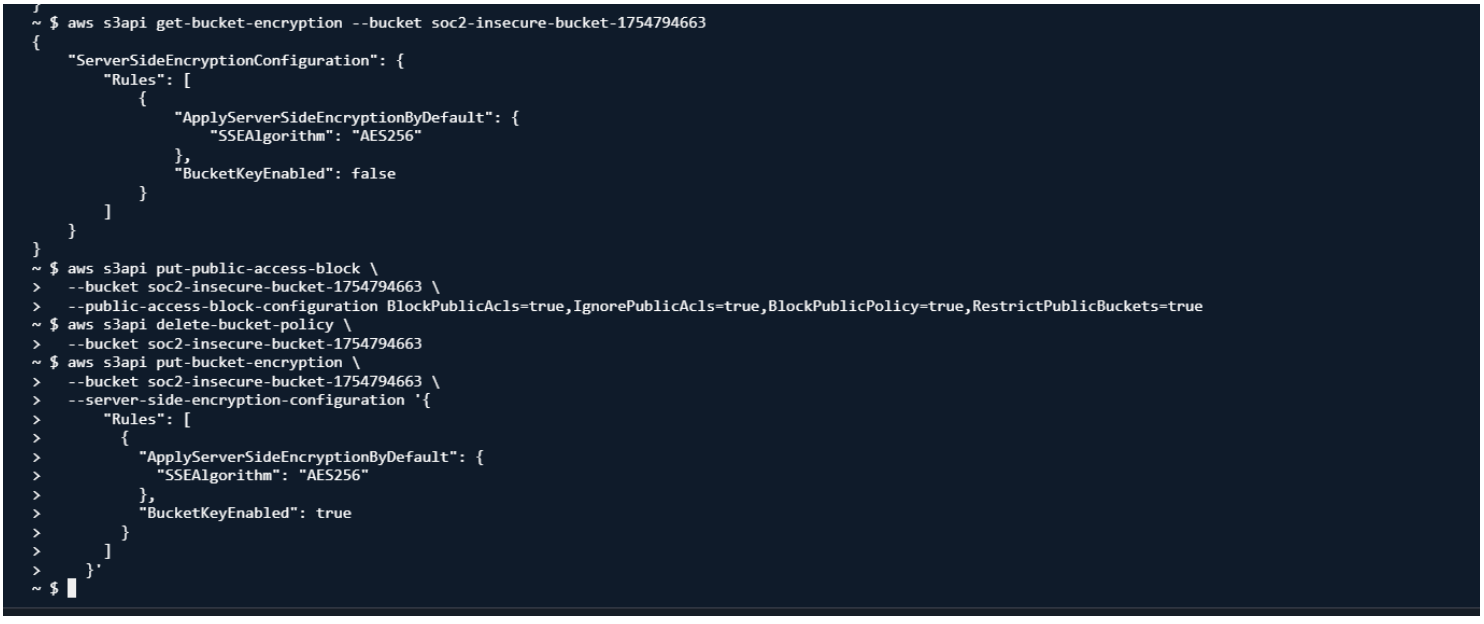
Executed the AWS CLI command to re-enable all public access block settings (BlockPublicAcls, IgnorePublicAcls, BlockPublicPolicy, RestrictPublicBuckets) for the insecure S3 bucket. This marked the start of the remediation phase, demonstrating how SOC 2 violations can be addressed by applying stricter access controls.

Step 10 - Remove Public Read Bucket Policy

Deleted the public-read bucket policy from the S3 bucket using the AWS CLI. This action ensured that no anonymous or unauthorised principal could retrieve objects from the bucket, restoring compliance with SOC 2 confidentiality requirements.

Step 11– Re-enable Server-Side Encryption on S3 Bucket

Reapplied AES256 server-side encryption to the insecure S3 bucket via AWS CLI. Enabling encryption ensured that all data at rest in the bucket was protected, aligning with SOC 2 requirements for safeguarding sensitive information.



Step 12– Confirm Public Access Block and Encryption Status

Verified the public access block configuration and server-side encryption status after remediation. The output confirmed that all public access blocks were active, and encryption was enabled with a bucket key, providing evidence of restored SOC 2 compliance.

Step 13 - Attempt to Retrieve Deleted Bucket Policy

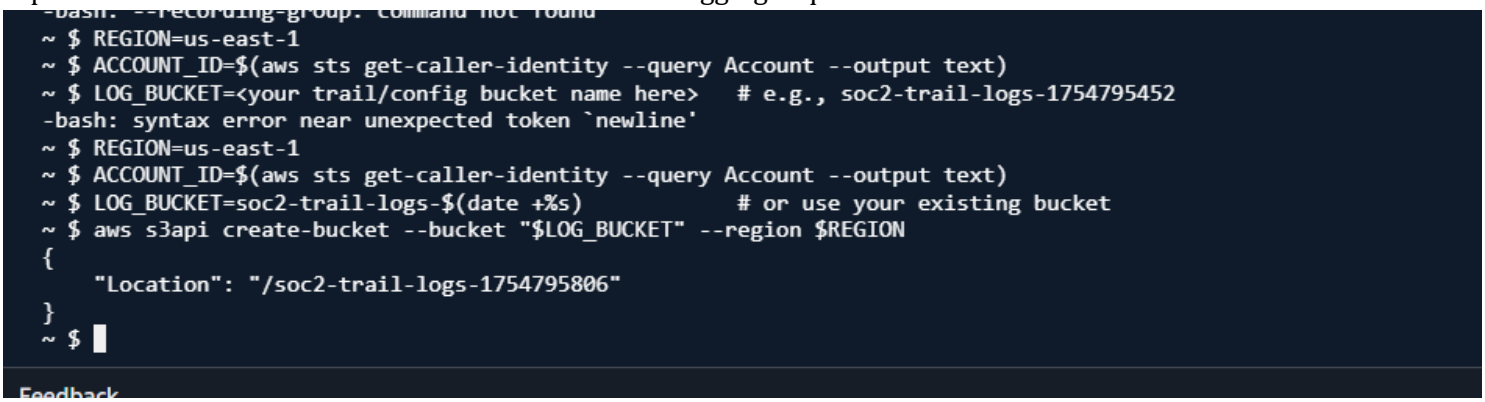
Ran a command to retrieve the bucket policy after deletion, which returned an error confirming that the policy no longer existed. This served as proof that the insecure public-read access path had been fully removed from the bucket.

Step 14 - Create CloudTrail Log Bucket Policy

Created an S3 bucket policy allowing AWS CloudTrail to write logs into a designated log bucket. The policy granted specific permissions to the cloudtrail.amazonaws.com service for s3:GetBucketAcl and s3:PutObject, establishing the storage foundation for audit trails as required under SOC 2.

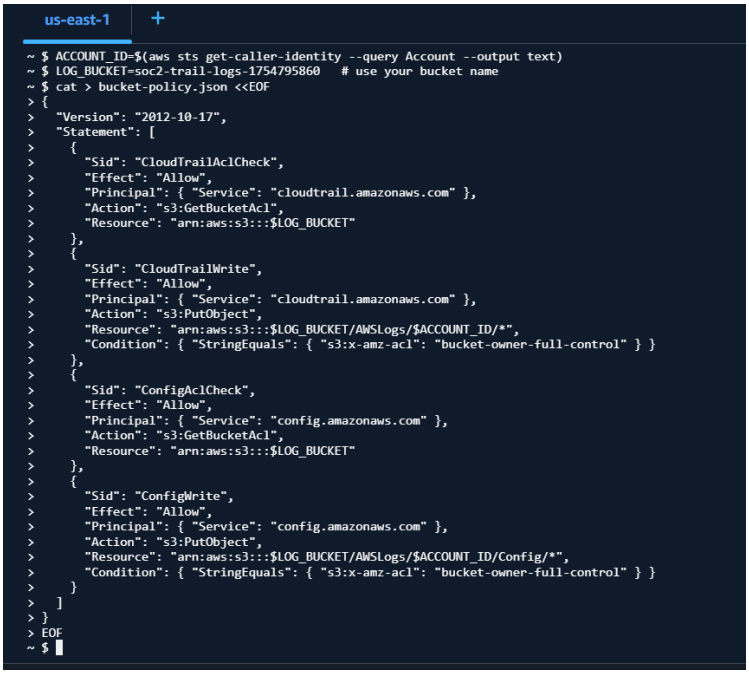
Step 15 – Create and Start CloudTrail

Using the AWS CLI, created a new multi-region CloudTrail named SOC2Trail and associated it with the designated log bucket. Verified that logging was successfully started, ensuring that all account activity across regions would be captured for audit evidence in accordance with SOC 2 logging requirements.



Step 16 – Create New CloudTrail Log Bucket

Provisioned a new S3 bucket named soc2-trail-logs-<timestamp> for CloudTrail logs using the AWS CLI. This step demonstrated the secure creation of a dedicated logging bucket, a critical element in SOC 2's requirement for immutable and tamper-proof audit logging.

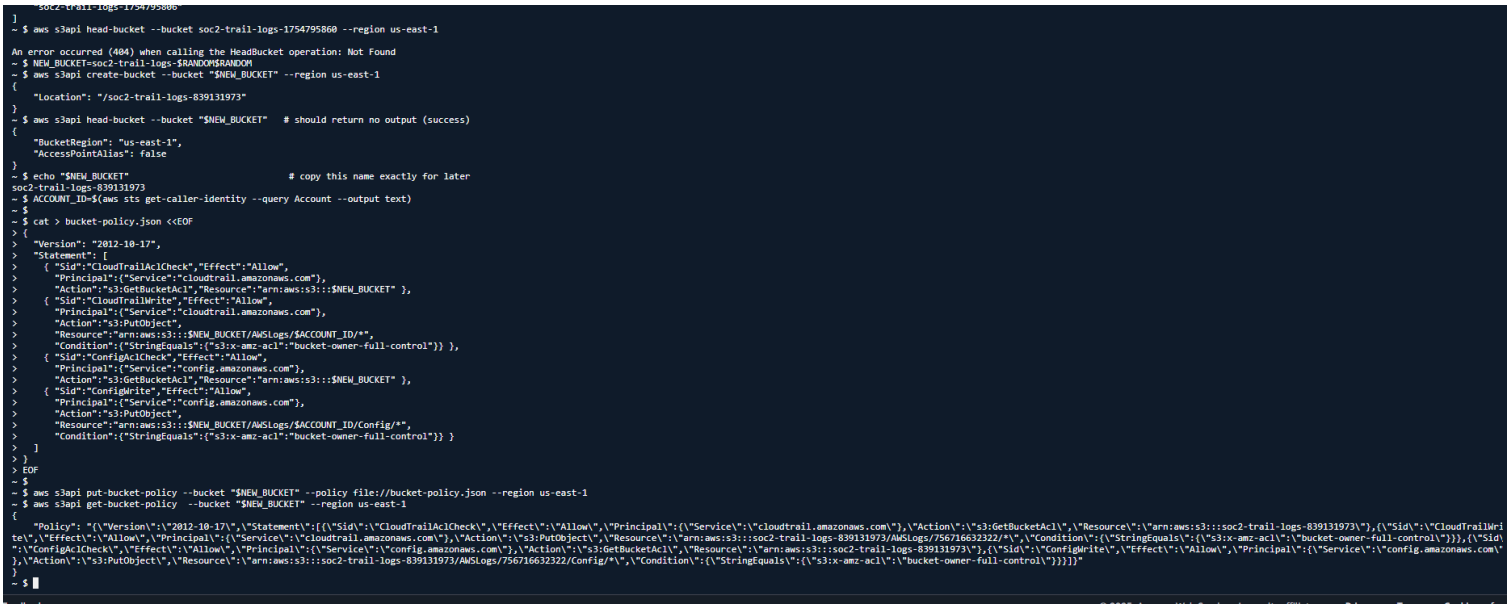


Step 17 – Create Combined CloudTrail and AWS Config Bucket Policy

Created a bucket policy granting both cloudtrail.amazonaws.com and config.amazonaws.com services the necessary s3:GetBucketAcl and s3:PutObject permissions to store logs in the soc2-trail-logs bucket. This unified logging policy ensured that both audit trails and configuration snapshots could be stored in a single secure location, meeting SOC 2 evidence collection requirements

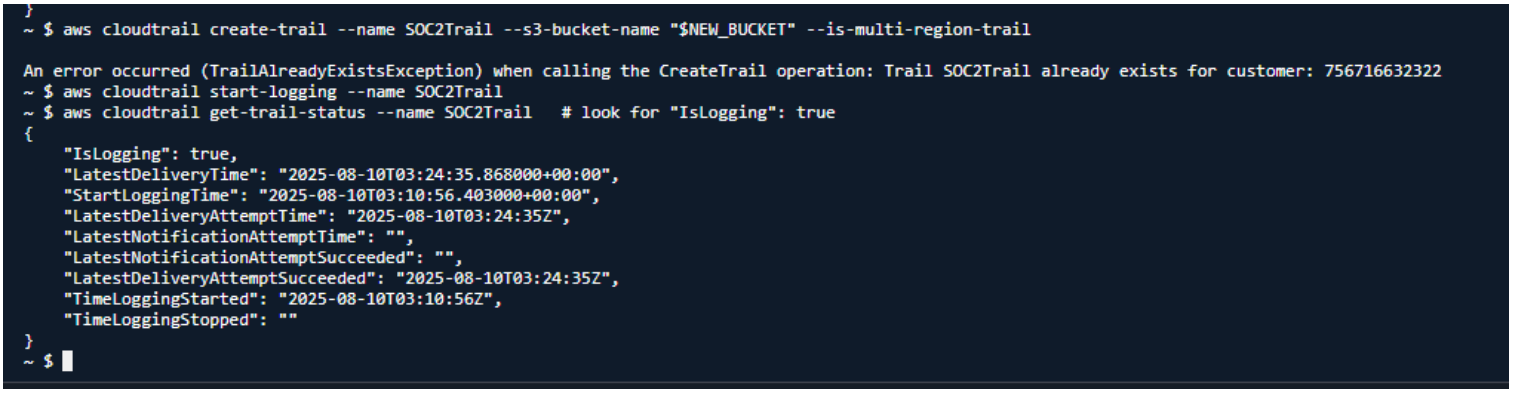
Step 18 – Provision New Trail Logs Bucket and Apply Policy

Verified the absence of the previous log bucket and created a new unique S3 bucket for CloudTrail logs. Generated a dynamic bucket policy with updated resource ARNs, granting appropriate logging permissions, and applied it to the bucket. This step demonstrated how to securely provision a fresh log storage location for SOC 2 monitoring.



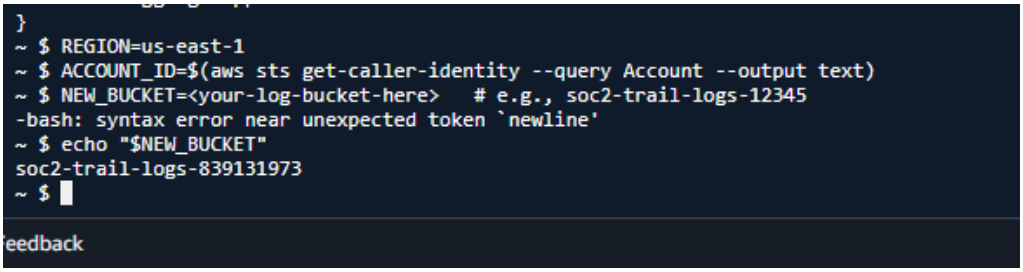
Step 19 - Attempt to Create CloudTrail Using Existing Name

Tried to create a new CloudTrail named SOC2Trail linked to the new log bucket but received a TrailAlreadyExistsException as the trail was already active. Verified trail status, confirming logging was enabled, ensuring continuous audit trail capture as required by SOC 2.



Step 20 – Generate and Display New Trail Bucket Name

Executed AWS CLI commands to generate and display a unique S3 bucket name for storing trail logs. This confirmed the use of a properly randomised, unique identifier for secure logging infrastructure.



Step 21 – Create AWS Config Configuration Recorder

Created a recorder.json configuration to enable AWS Config recording for all supported and global resource types, associating it with the AWS Config service role. This configuration allowed for comprehensive resource tracking in alignment with SOC 2's monitoring and change management controls.

```
~ $ cat > recorder.json <<EOF
> {
>   "name": "default",
>   "roleARN": "arn:aws:iam::${ACCOUNT_ID}:role/aws-service-role/config.amazonaws.com/AWSServiceRoleForConfig",
>   "recordingGroup": { "allSupported": true, "includeGlobalResourceTypes": true }
> }
> EOF
~ $
~ $ aws configservice put-configuration-recorder --configuration-recorder file://recorder.json
~ $
```

Step 22 – Configure AWS Config Delivery Channel

Created a delivery.json file specifying the new S3 bucket for AWS Config data storage and applied it as the AWS Config delivery channel. This ensured that all configuration history and compliance snapshots would be stored securely for SOC 2 evidence purposes.

```
~ $ cat > recorder.json <<EOF
> {
>   "name": "default",
>   "roleARN": "arn:aws:iam::${ACCOUNT_ID}:role/aws-service-role/config.amazonaws.com/AWSServiceRoleForConfig",
>   "recordingGroup": { "allSupported": true, "includeGlobalResourceTypes": true }
> }
> EOF
~ $
~ $ aws configservice put-configuration-recorder --configuration-recorder file://recorder.json
~ $ cat > delivery.json <<EOF
> { "name": "default", "s3BucketName": "${NEW_BUCKET}" }
> EOF
~ $
~ $ aws configservice put-delivery-channel --delivery-channel file://delivery.json
~ $
```

Feedback

Step 23 – Start AWS Config Recording and Verify Status

Enabled the AWS Config recorder and verified its operational status via AWS CLI, confirming it was actively recording with a SUCCESS state. This step demonstrated operational evidence of SOC 2 control monitoring being in place.

```
~ $ aws configservice put-delivery-channel --delivery-channel file://delivery.json
~ $ aws configservice start-configuration-recorder --configuration-recorder-name default
~ $ aws configservice describe-configuration-recorder-status
{
    "ConfigurationRecordersStatus": [
        {
            "arn": "arn:aws:config:us-east-1:756716632322:configuration-recorder/default/7lwdob4ncgtcdu97",
            "name": "default",
            "lastStartTime": "2025-08-05T06:56:29.418000+00:00",
            "recording": true,
            "lastStatus": "SUCCESS",
            "lastStatusChangeTime": "2025-08-10T03:14:39.765000+00:00"
        }
    ]
}
~ $
```

Step 24 – Create SOC 2 Conformance Pack Template

Drafted a cis-mini.yaml conformance pack template containing AWS Config rules for key SOC 2 controls: enabling CloudTrail, enforcing MFA on the root account, prohibiting public S3 buckets, requiring server-side encryption on S3, and restricting SSH access. This template served as an automated compliance check mechanism.

```
~ $ # 1) Create a local conformance pack template (YAML)
~ $ cat > cis-mini.yaml <<'EOF'
> Resources:
>   CloudTrailEnabled:
>     Type: AWS::Config::ConfigRule
>     Properties:
>       ConfigRuleName: cloudtrail-enabled
>       Source:
>         Owner: AWS
>         SourceIdentifier: CLOUD_TRAIL_ENABLED
>
>   RootAccountMFAEnabled:
>     Type: AWS::Config::ConfigRule
>     Properties:
>       ConfigRuleName: root-account-mfa-enabled
>       Source:
>         Owner: AWS
>         SourceIdentifier: ROOT_ACCOUNT_MFA_ENABLED
>
>   S3PublicReadProhibited:
>     Type: AWS::Config::ConfigRule
>     Properties:
>       ConfigRuleName: s3-bucket-public-read-prohibited
>       Source:
>         Owner: AWS
>         SourceIdentifier: S3_BUCKET_PUBLIC_READ_PROHIBITED
>
>   S3BucketServerSideEncryptionEnabled:
>     Type: AWS::Config::ConfigRule
>     Properties:
>       ConfigRuleName: s3-bucket-server-side-encryption-enabled
>       Source:
>         Owner: AWS
>         SourceIdentifier: S3_BUCKET_SERVER_SIDE_ENCRYPTION_ENABLED
>
>   IncomingSSHDisabled:
>     Type: AWS::Config::ConfigRule
>     Properties:
>       ConfigRuleName: restricted-ssh
>       Source:
>         Owner: AWS
>         SourceIdentifier: INCOMING_SSH_DISABLED
> EOF
~ $
~ $ # 2) Deploy the pack from the local file
~ $ aws configservice put-conformance-pack \
>   --conformance-pack-name SOC2-MINI \
>   --template-body file://cis-mini.yaml
{
    "ConformancePackArn": "arn:aws:config:us-east-1:756716632322:conformance-pack/SOC2-MINI/conformance-pack-uyli01pus"
}
~ $
~ $ # 3) Check status until CREATE_COMPLETE
```

Step 25 – Deploy SOC 2 Conformance Pack

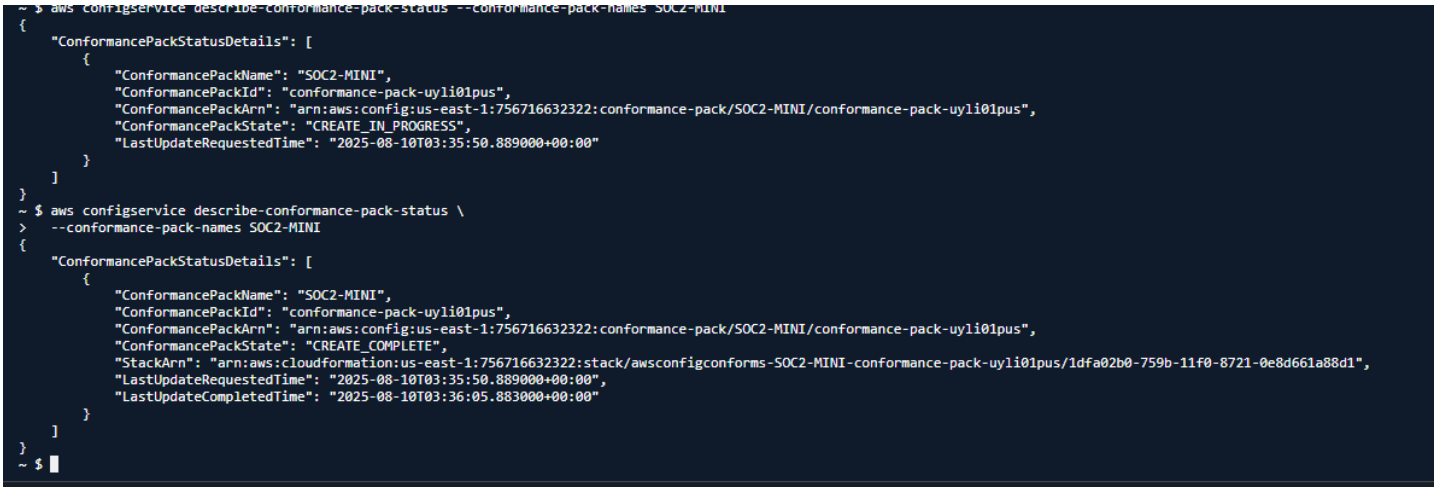
Deployed the conformance pack to the AWS account using the AWS CLI, initiating the creation of compliance checks for all the included rules. This provided a structured, automated method to continuously evaluate SOC 2-related controls.

```
~ $
~ $ # 3) Check status until CREATE_COMPLETE
~ $ aws configservice describe-conformance-pack-status --conformance-pack-names SOC2-MINI
{
    "ConformancePackStatusDetails": [
        {
            "ConformancePackName": "SOC2-MINI",
            "ConformancePackId": "conformance-pack-uyli01pus",
            "ConformancePackArn": "arn:aws:config:us-east-1:756716632322:conformance-pack/SOC2-MINI/conformance-pack-uyli01pus",
            "ConformancePackState": "CREATE_IN_PROGRESS",
            "LastUpdateRequestedTime": "2025-08-10T03:35:50.889000+00:00"
        }
    ]
}
~ $
```

eedback

Step 26 – Verify Conformance Pack Deployment Status

Checked the status of the SOC2-MINI conformance pack until it reached CREATE_COMPLETE. This confirmed that the automated compliance evaluation framework was active and ready to provide SOC 2 control compliance insights.

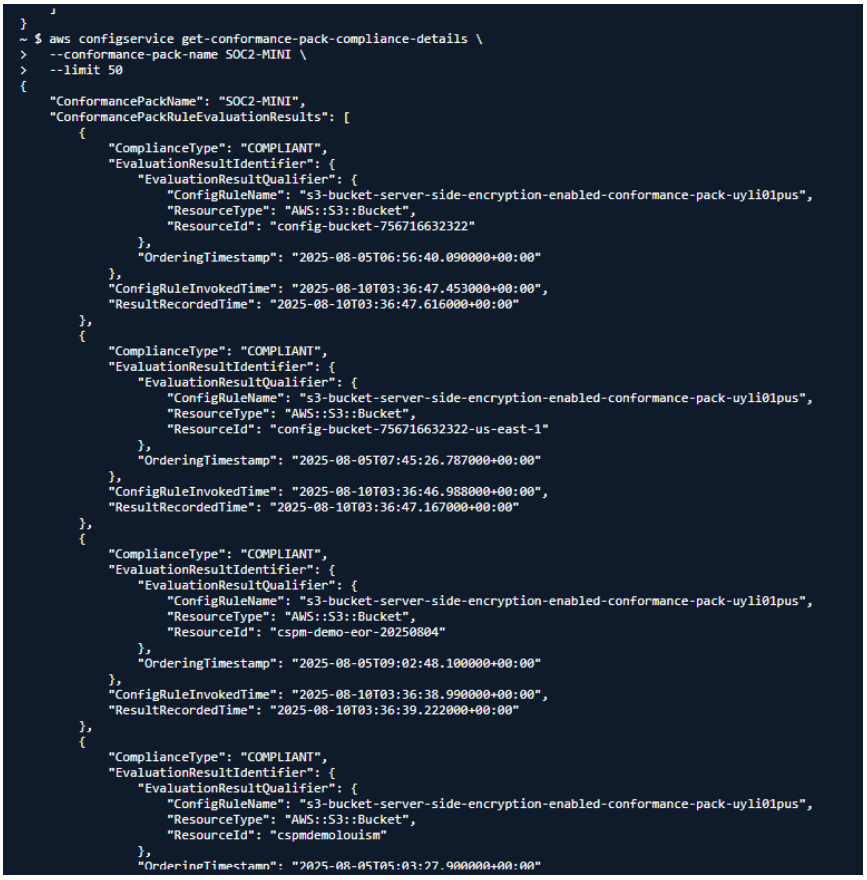


Step 27 – Confirm Conformance Pack Deployment and Compliance Summary

Queried AWS Config to verify that the SOC2-MINI conformance pack had reached the CREATE_COMPLETE status. Retrieved the compliance summary, which showed the pack as NON_COMPLIANT. This result confirmed that one or more resources were violating SOC 2-related controls, aligning with the intentional misconfigurations created earlier in the demo.

Step 28 – View All Compliance Results

Checked the compliance status of all resources in the SOC2-MINI conformance pack. Results showed all S3 buckets were compliant, while two EC2 security groups failed due to unrestricted SSH access.



```
}
~ $ aws configservice get-conformance-pack-compliance-details \
>   --conformance-pack-name SOC2-MINI \
>   --limit 50
{
    "ConformancePackName": "SOC2-MINI",
    "ConformancePackRuleEvaluationResults": [
        {
            "ComplianceType": "COMPLIANT",
            "EvaluationResultIdentifier": {
                "EvaluationResultQualifier": {
                    "ConfigRuleName": "s3-bucket-server-side-encryption-enabled-conformance-pack-uyli01pus",
                    "ResourceType": "AWS::S3::Bucket",
                    "ResourceId": "config-bucket-756716632322"
                },
                "OrderingTimestamp": "2025-08-05T06:56:40.090000+00:00"
            },
            "ConfigRuleInvokedTime": "2025-08-10T03:36:47.453000+00:00",
            "ResultRecordedTime": "2025-08-10T03:36:47.616000+00:00"
        },
        {
            "ComplianceType": "COMPLIANT",
            "EvaluationResultIdentifier": {
                "EvaluationResultQualifier": {
                    "ConfigRuleName": "s3-bucket-server-side-encryption-enabled-conformance-pack-uyli01pus",
                    "ResourceType": "AWS::S3::Bucket",
                    "ResourceId": "config-bucket-756716632322-us-east-1"
                },
                "OrderingTimestamp": "2025-08-05T07:45:26.787000+00:00"
            },
            "ConfigRuleInvokedTime": "2025-08-10T03:36:46.988000+00:00",
            "ResultRecordedTime": "2025-08-10T03:36:47.167000+00:00"
        },
        {
            "ComplianceType": "COMPLIANT",
            "EvaluationResultIdentifier": {
                "EvaluationResultQualifier": {
                    "ConfigRuleName": "s3-bucket-server-side-encryption-enabled-conformance-pack-uyli01pus",
                    "ResourceType": "AWS::S3::Bucket",
                    "ResourceId": "cspm-demo-eor-20250804"
                },
                "OrderingTimestamp": "2025-08-05T09:02:48.100000+00:00"
            },
            "ConfigRuleInvokedTime": "2025-08-10T03:36:38.990000+00:00",
            "ResultRecordedTime": "2025-08-10T03:36:39.222000+00:00"
        },
        {
            "ComplianceType": "COMPLIANT",
            "EvaluationResultIdentifier": {
                "EvaluationResultQualifier": {
                    "ConfigRuleName": "s3-bucket-server-side-encryption-enabled-conformance-pack-uyli01pus",
                    "ResourceType": "AWS::S3::Bucket",
                    "ResourceId": "cspmdemolouism"
                },
                "OrderingTimestamp": "2025-08-05T05:03:27.900000+00:00"
```

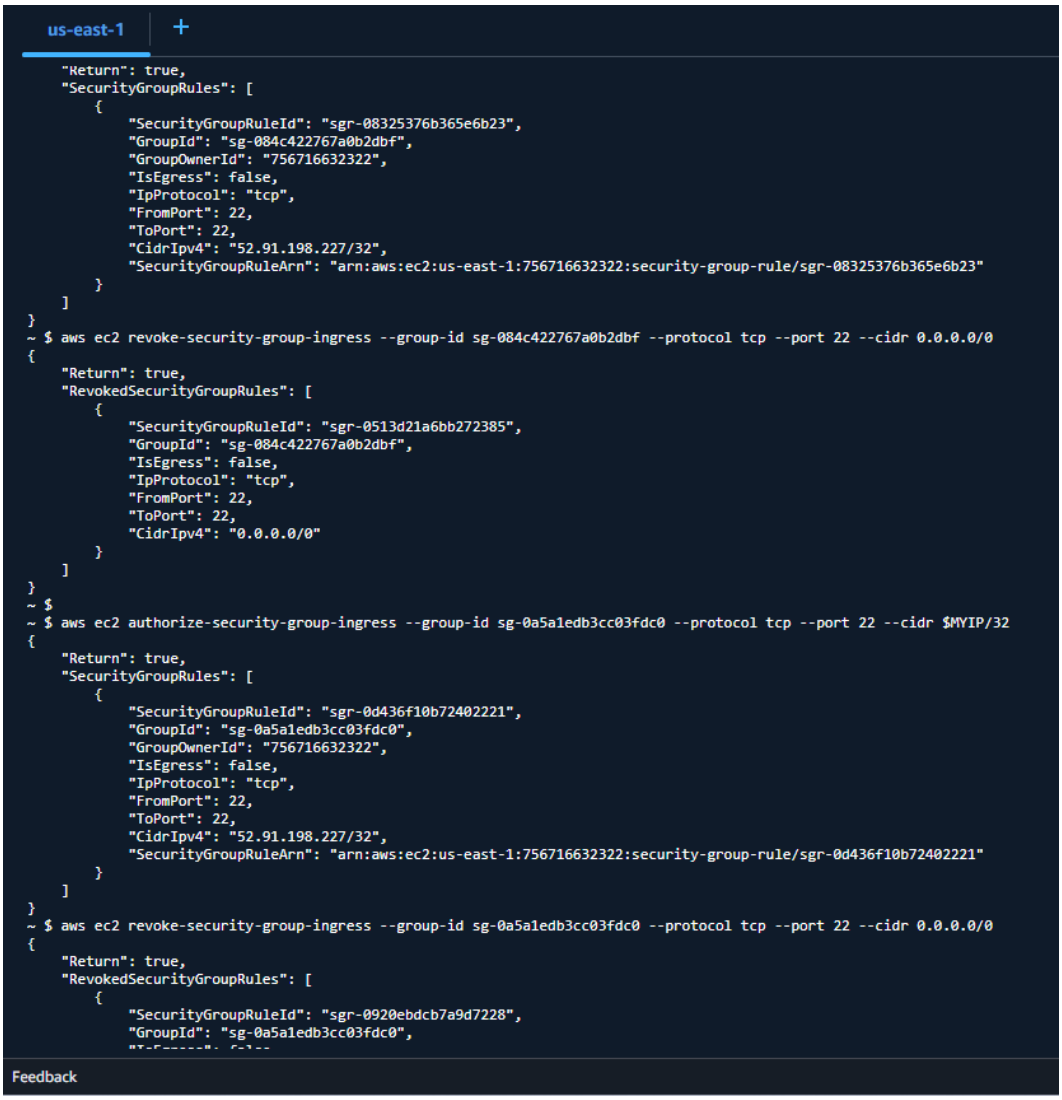
Step 29 – Identify Non-Compliant Resources

Filtered the compliance results to only display resources marked as non-compliant. Confirmed that the failing resources were two EC2 security groups with open SSH rules to all IP addresses.

```
~ $ aws configservice get-conformance-pack-compliance-details \
>     --conformance-pack-name SOC2-MINI \
>     --limit 50 \
>     --query "ConformancePackRuleEvaluationResults[?ComplianceType=='NON_COMPLIANT']"
[
    {
        "ComplianceType": "NON_COMPLIANT",
        "EvaluationResultIdentifier": {
            "EvaluationResultQualifier": {
                "ConfigRuleName": "restricted-ssh-conformance-pack-uyli01pus",
                "ResourceType": "AWS::EC2::SecurityGroup",
                "ResourceId": "sg-084c422767a0b2dbf"
            },
            "OrderingTimestamp": "2025-08-10T03:36:04.266000+00:00"
        },
        "ConfigRuleInvokedTime": "2025-08-10T03:36:35.349000+00:00",
        "ResultRecordedTime": "2025-08-10T03:36:35.991000+00:00"
    },
    {
        "ComplianceType": "NON_COMPLIANT",
        "EvaluationResultIdentifier": {
            "EvaluationResultQualifier": {
                "ConfigRuleName": "restricted-ssh-conformance-pack-uyli01pus",
                "ResourceType": "AWS::EC2::SecurityGroup",
                "ResourceId": "sg-0a5a1edb3cc03fdc0"
            },
            "OrderingTimestamp": "2025-08-10T03:36:04.266000+00:00"
        },
        "ConfigRuleInvokedTime": "2025-08-10T03:36:35.349000+00:00",
        "ResultRecordedTime": "2025-08-10T03:36:35.982000+00:00"
    }
]
~ $
```

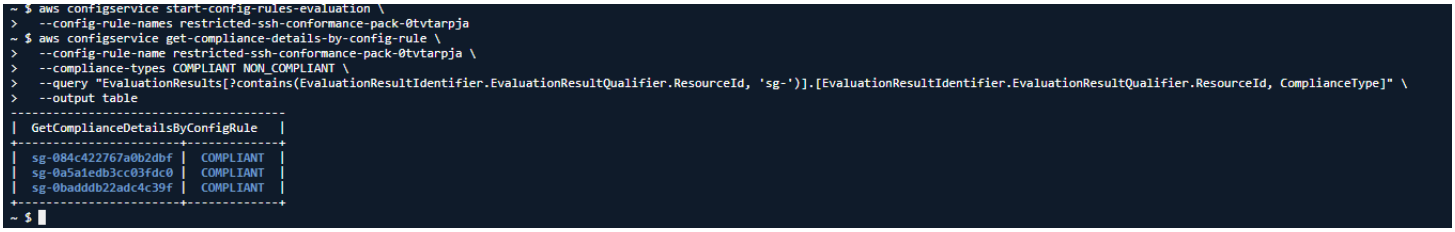
Step 30 – Fix Security Group Issues

Reviewed the inbound rules for each failing security group. Removed the unrestricted SSH access and replaced it with rules allowing SSH only from a trusted IP address.

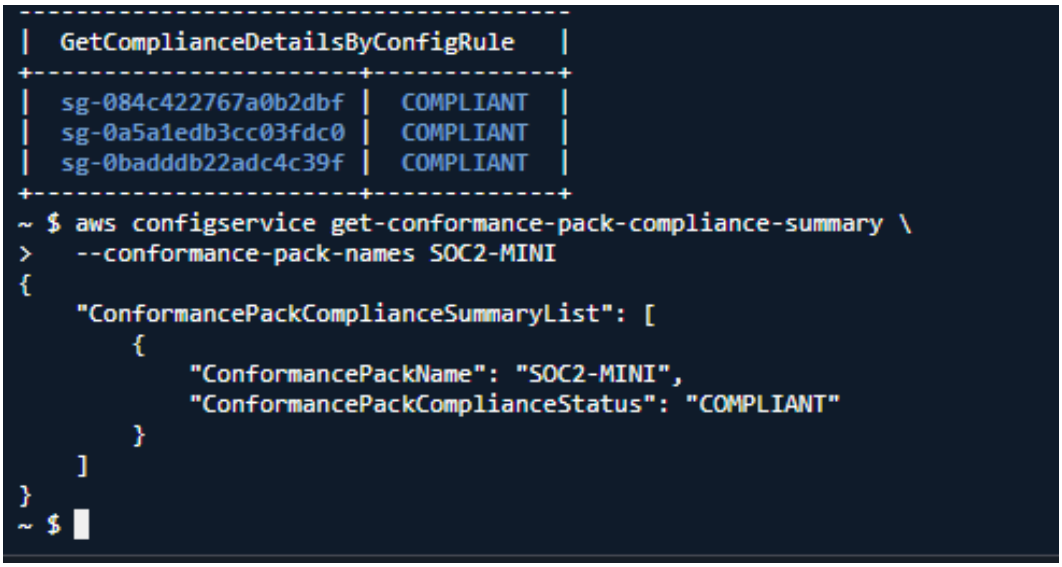


Step 31 – Re-run Compliance Check

Triggered a re-evaluation of the SSH restriction rule. Verified that both previously failing security groups were now marked as compliant.



Step 32 – Confirm Overall Compliance
Checked the overall status of the SOC2-MINI conformance pack. Confirmed that all rules were compliant and no issues remained



Conclusion
The SOC 2 Readiness and Compliance Remediation Project outlines the end-to-end process for identifying, documenting, and addressing gaps against the SOC 2 Trust Services Criteria. This reference covers practical examples of intentional misconfigurations, the detection process, and the step-by-step remediation needed to achieve compliance.

By following the methodology outlined in Steps 1–32, any organisation can:

- Reduce audit risk and avoid costly rework
- Increase trust with enterprise customers
- Strengthen its overall security and compliance program
- Shorten the SOC 2 audit timeline

Ongoing Practice
Regular compliance checks, remediation validation, and evidence collection should be built into operational routines. Implementing continuous compliance monitoring ensures sustained SOC 2 adherence, reduces the risk of control drift, and supports a stronger security posture year-round.