SOC 2 Readiness & Compliance Remediation Project

Executive Summary

Objective

Assess the current cloud environment against SOC 2 Trust Services Criteria, highlight gaps, and deliver a prioritised remediation plan with evidence so the organisation is audit-ready and able to demonstrate strong security, availability, confidentiality, processing integrity, and privacy controls.

Why this matters

SOC 2 is a trust signal for customers and a gateway to enterprise deals. Addressing gaps early reduces audit findings, shortens audit timelines, and lowers risk across data protection, access control, and change monitoring.

What we did

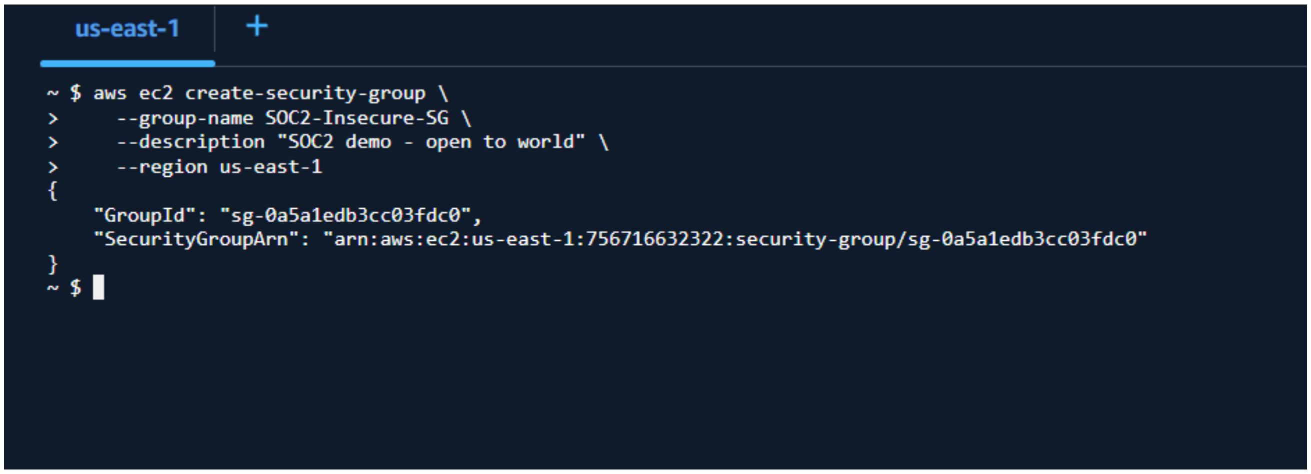
- Established a safe but realistic baseline by intentionally exposing common misconfigurations for demonstration.
- Enabled account-wide logging and configuration monitoring to create an immutable audit trail and continuous visibility.
- Deployed an automated ruleset to evaluate the environment against SOC 2-aligned controls.
- Remediated the highest-risk findings, then re-evaluated to confirm compliance.
- Collected before-and-after evidence and packaged it for clients and auditors.

Highlights and outcomes

- Measured posture against SOC 2 control objectives and removed high-impact risks.
- Centralised findings and remediation so stakeholders can act quickly and confidently.
- Delivered an evidence-backed remediation plan that maps issues to fixes and owners.
- Reduced audit readiness time and improved the likelihood of passing first time.

Step 1 – Create Insecure Security Group via AWS CLI

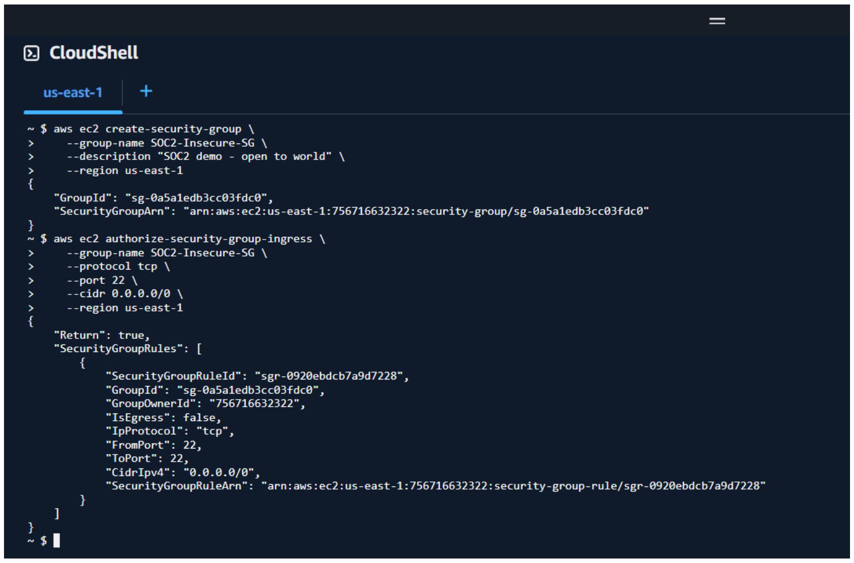
The screenshot shows the creation of a new EC2 security group named SOC2-Insecure-SG using the AWS CLI. The description explicitly states "SOC2 demo – open to world" to indicate this was intentionally created as a misconfigured resource for demonstration purposes. This step establishes a baseline insecure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



```
~ $ aws ec2 create-security-group \
>     --group-name SOC2-Insecure-SG \
>     --description "SOC2 demo - open to world" \
>     --region us-east-1
{
    "GroupId": "sg-0a5a1edb3cc03fdc0",
    "SecurityGroupArn": "arn:aws:ec2:us-east-1:756716632322:security-group/sg-0a5a1edb3cc03fdc0"
}
~ $ 
```

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

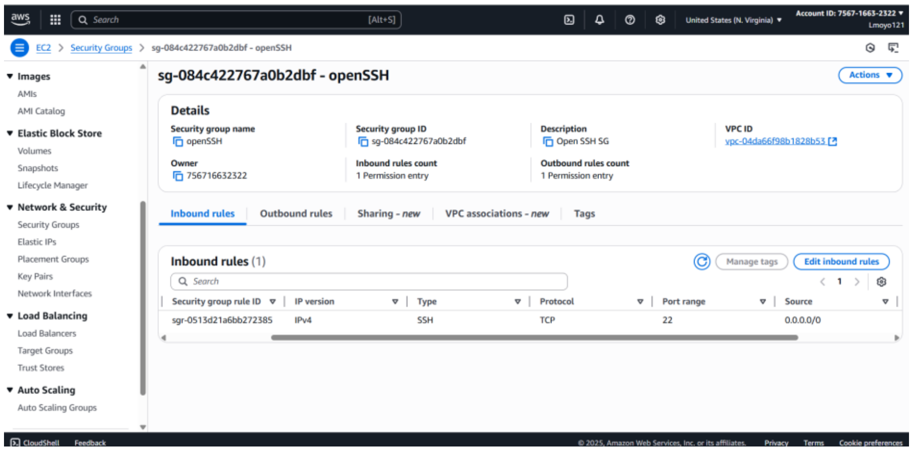
The screenshot displays the authorisation of an inbound rule on the SOC2-Insecure-SG security group, allowing TCP traffic on port 22 (SSH) from any IPv4 address (0.0.0.0/0). This configuration represents a critical security risk under SOC 2, as it allows unrestricted internet access to administrative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



```
~ $ aws ec2 create-security-group \
>     --group-name SOC2-Insecure-SG \
>     --description "SOC2 demo - open to world" \
>     --region us-east-1
{
    "GroupId": "sg-0a5a1edb3cc03fdc0",
    "SecurityGroupArn": "arn:aws:ec2:us-east-1:756716632322:security-group/sg-0a5a1edb3cc03fdc0"
}
~ $ aws ec2 authorize-security-group-ingress \
>     --group-name SOC2-Insecure-SG \
>     --protocol tcp \
>     --port 22 \
>     --cidr 0.0.0.0/0 \
>     --region us-east-1
{
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-0920ebdcb7a9d7228",
            "GroupId": "sg-0a5a1edb3cc03fdc0",
            "GroupOwnerId": "756716632322",
            "IsEgress": false,
            "IpProtocol": "tcp",
            "FromPort": 22,
            "ToPort": 22,
            "CidrIpv4": "0.0.0.0/0",
            "SecurityGroupRuleArn": "arn:aws:ec2:us-east-1:756716632322:security-group-rule/sgr-0920ebdcb7a9d7228"
        }
    ]
}
~ $
```

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.



sg-084c422767a0b2dbf - openSSH

Details

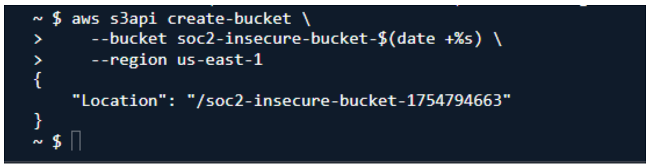
Security group name	Security group ID	Description	VPC ID
openSSH	sg-084c422767a0b2dbf	Open SSH SG	vpc-04da66f98b1828b53
Owner	**Inbound rules count**	**Outbound rules count**	
756716632322	1 Permission entry	1 Permission entry	

Inbound rules (1)

Security group rule ID	IP version	Type	Protocol	Port range	Source
sgr-0513d21a6bb272385	IPv4	SSH	TCP	22	0.0.0.0/0

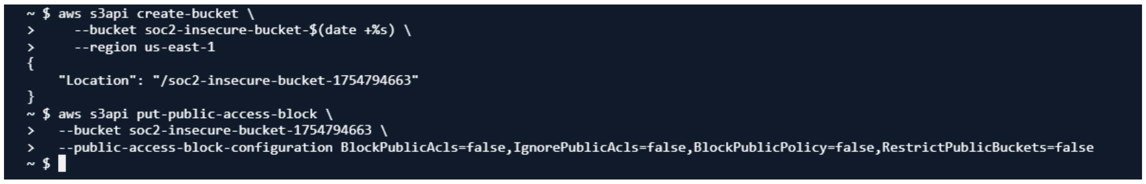
Step 4 – Create Insecure S3 Bucket via AWS CLI

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.

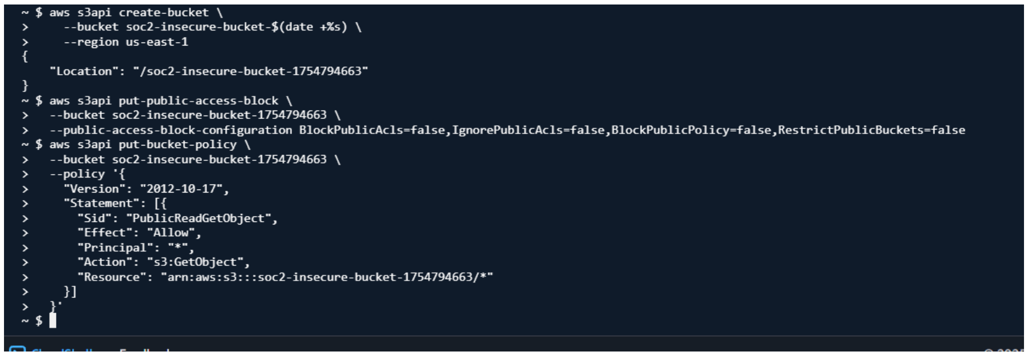


```
~ $ aws s3api create-bucket \
>     --bucket soc2-insecure-bucket-$(date +%s) \
>     --region us-east-1
{
    "Location": "/soc2-insecure-bucket-1754794663"
}
~ $
```

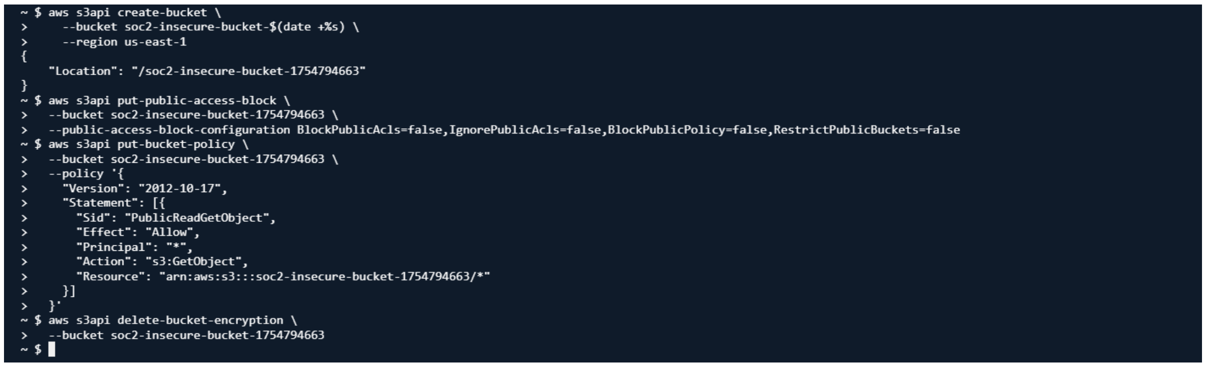
Step 5 – Disable Public Access Block Settings on S3 Bucket
The screenshot displays AWS CLI commands that explicitly disable all public access block settings (BlockPublicAcls, IgnorePublicAcls, BlockPublicPolicy, RestrictPublicBuckets) for the newly created bucket. This deliberate change allows public ACLs and bucket policies to be applied, creating a high-risk configuration in violation of SOC 2 confidentiality and privacy principles.


```
~ $ aws s3api create-bucket \
>     --bucket soc2-insecure-bucket-$(date +%s) \
>     --region us-east-1
{
    "Location": "/soc2-insecure-bucket-1754794663"
}
~ $ aws s3api put-public-access-block \
>   --bucket soc2-insecure-bucket-1754794663 \
>   --public-access-block-configuration BlockPublicAcls=false,IgnorePublicAcls=false,BlockPublicPolicy=false,RestrictPublicBuckets=false
~ $
```

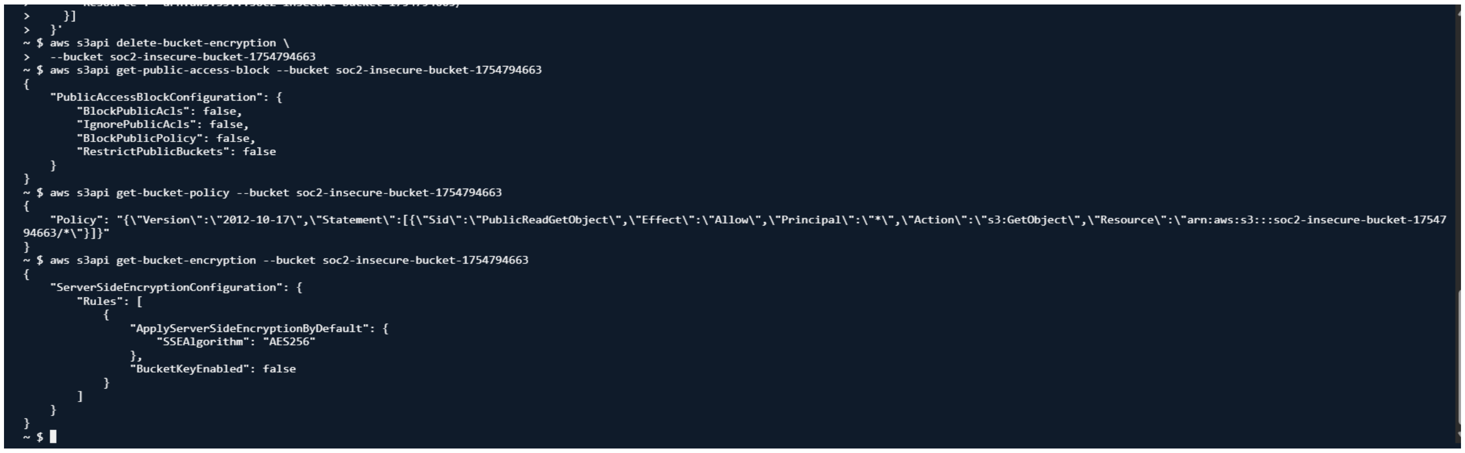
Step 6 - Apply Public Read Policy to S3 Bucket
The screenshot shows the attachment of a bucket policy granting public read access (s3:GetObject) to all principals (Principal: *) for every object in the bucket. This misconfiguration, combined with the disabled public access blocks from Step 6, results in unrestricted public access to stored data — a severe breach of SOC 2 security and confidentiality requirements, demonstrated intentionally for this exercise.


```
~ $ aws s3api create-bucket \
>     --bucket soc2-insecure-bucket-$(date +%s) \
>     --region us-east-1
{
    "Location": "/soc2-insecure-bucket-1754794663"
}
~ $ aws s3api put-public-access-block \
>   --bucket soc2-insecure-bucket-1754794663 \
>   --public-access-block-configuration BlockPublicAcls=false,IgnorePublicAcls=false,BlockPublicPolicy=false,RestrictPublicBuckets=false
~ $ aws s3api put-bucket-policy \
>   --bucket soc2-insecure-bucket-1754794663 \
>   --policy '{
>     "Version": "2012-10-17",
>     "Statement": [{
>       "Sid": "PublicReadGetObject",
>       "Effect": "Allow",
>       "Principal": "*",
>       "Action": "s3:GetObject",
>       "Resource": "arn:aws:s3:::soc2-insecure-bucket-1754794663/*"
>     }]
>   }'
~ $
```

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.

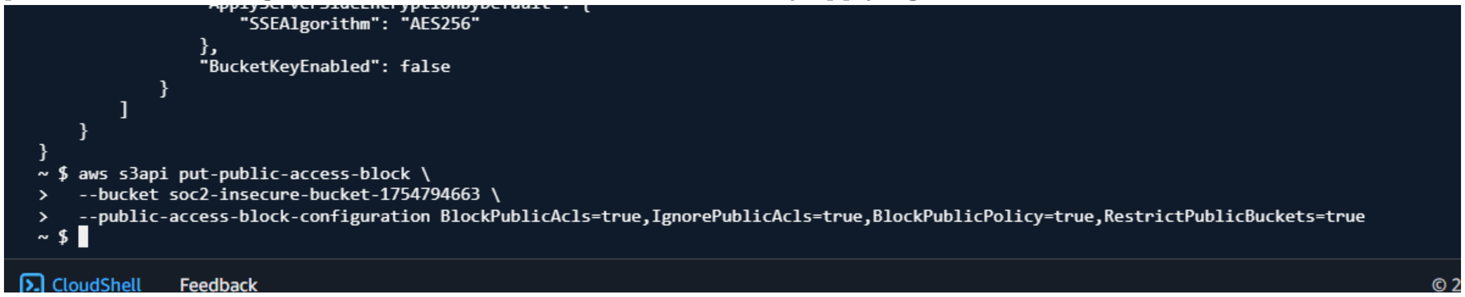

```
~ $ aws s3api create-bucket \
>     --bucket soc2-insecure-bucket-$(date +%s) \
>     --region us-east-1
{
    "Location": "/soc2-insecure-bucket-1754794663"
}
~ $ aws s3api put-public-access-block \
>   --bucket soc2-insecure-bucket-1754794663 \
>   --public-access-block-configuration BlockPublicAcls=false,IgnorePublicAcls=false,BlockPublicPolicy=false,RestrictPublicBuckets=false
~ $ aws s3api put-bucket-policy \
>   --bucket soc2-insecure-bucket-1754794663 \
>   --policy '{
>     "Version": "2012-10-17",
>     "Statement": [{
>       "Sid": "PublicReadGetObject",
>       "Effect": "Allow",
>       "Principal": "*",
>       "Action": "s3:GetObject",
>       "Resource": "arn:aws:s3:::soc2-insecure-bucket-1754794663/*"
>     }]
>   }'
~ $ aws s3api delete-bucket-encryption \
>   --bucket soc2-insecure-bucket-1754794663
~ $
```

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.



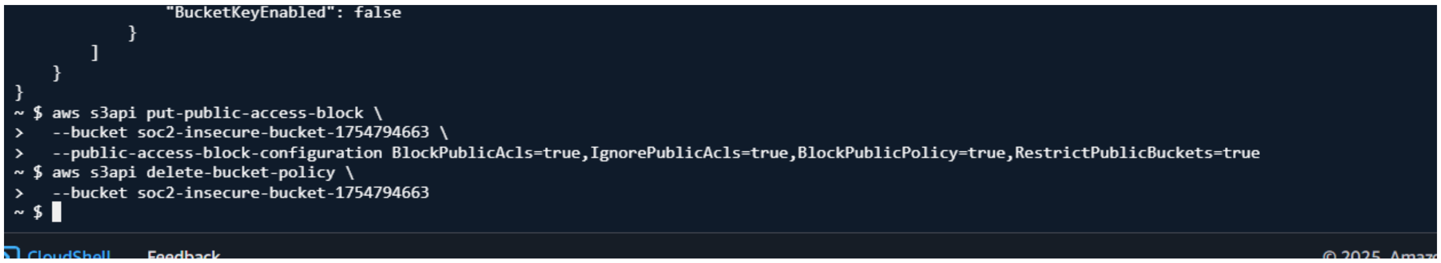
```
>     }]
>   }'
~ $ aws s3api delete-bucket-encryption \
>   --bucket soc2-insecure-bucket-1754794663
~ $ aws s3api get-public-access-block --bucket soc2-insecure-bucket-1754794663
{
    "PublicAccessBlockConfiguration": {
        "BlockPublicAcls": false,
        "IgnorePublicAcls": false,
        "BlockPublicPolicy": false,
        "RestrictPublicBuckets": false
    }
}
~ $ aws s3api get-bucket-policy --bucket soc2-insecure-bucket-1754794663
{
    "Policy": "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Sid\":\"PublicReadGetObject\",\"Effect\":\"Allow\",\"Principal\":\"*\",\"Action\":\"s3:GetObject\",\"Resource\":\"arn:aws:s3:::soc2-insecure-bucket-17547
94663/*\"}]}"
}
~ $ aws s3api get-bucket-encryption --bucket soc2-insecure-bucket-1754794663
{
    "ServerSideEncryptionConfiguration": {
        "Rules": [
            {
                "ApplyServerSideEncryptionByDefault": {
                    "SSEAlgorithm": "AES256"
                },
                "BucketKeyEnabled": false
            }
        ]
    }
}
~ $
```

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.



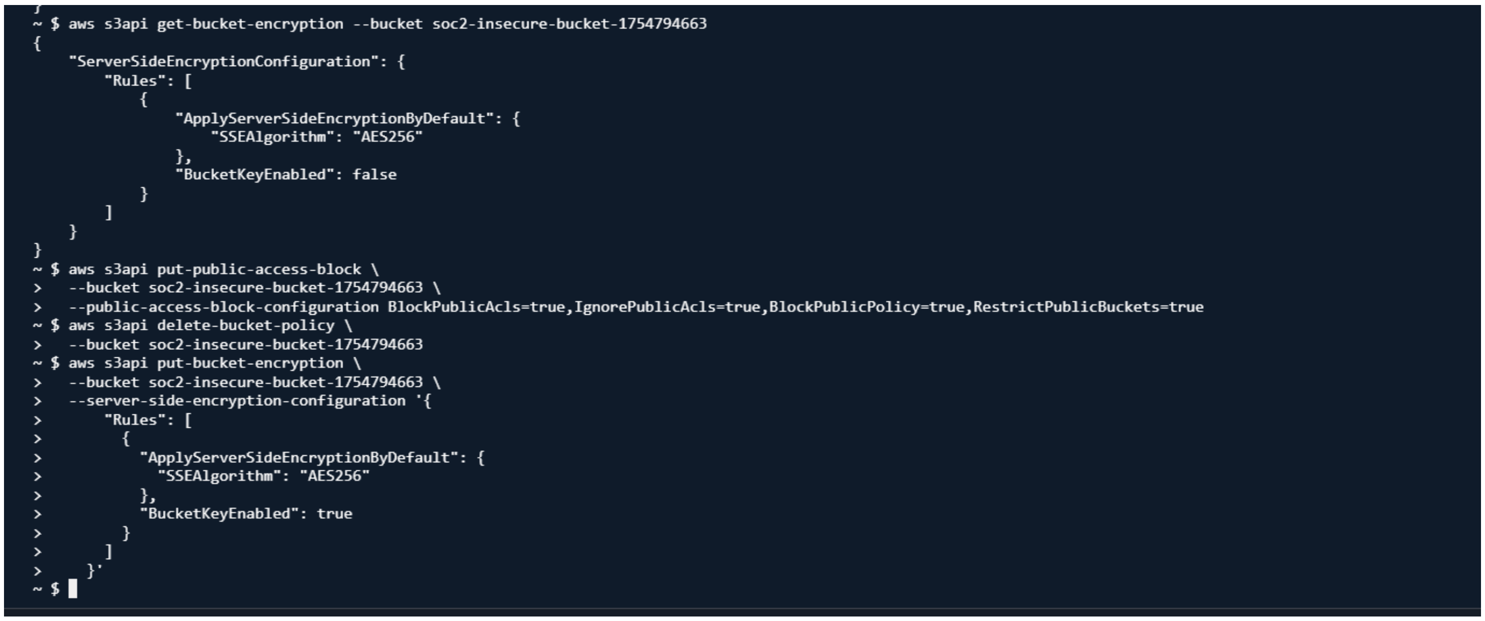
```
                    "SSEAlgorithm": "AES256"
                },
                "BucketKeyEnabled": false
            }
        ]
    }
}
~ $ aws s3api put-public-access-block \
>   --bucket soc2-insecure-bucket-1754794663 \
>   --public-access-block-configuration BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=true
~ $
```

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.



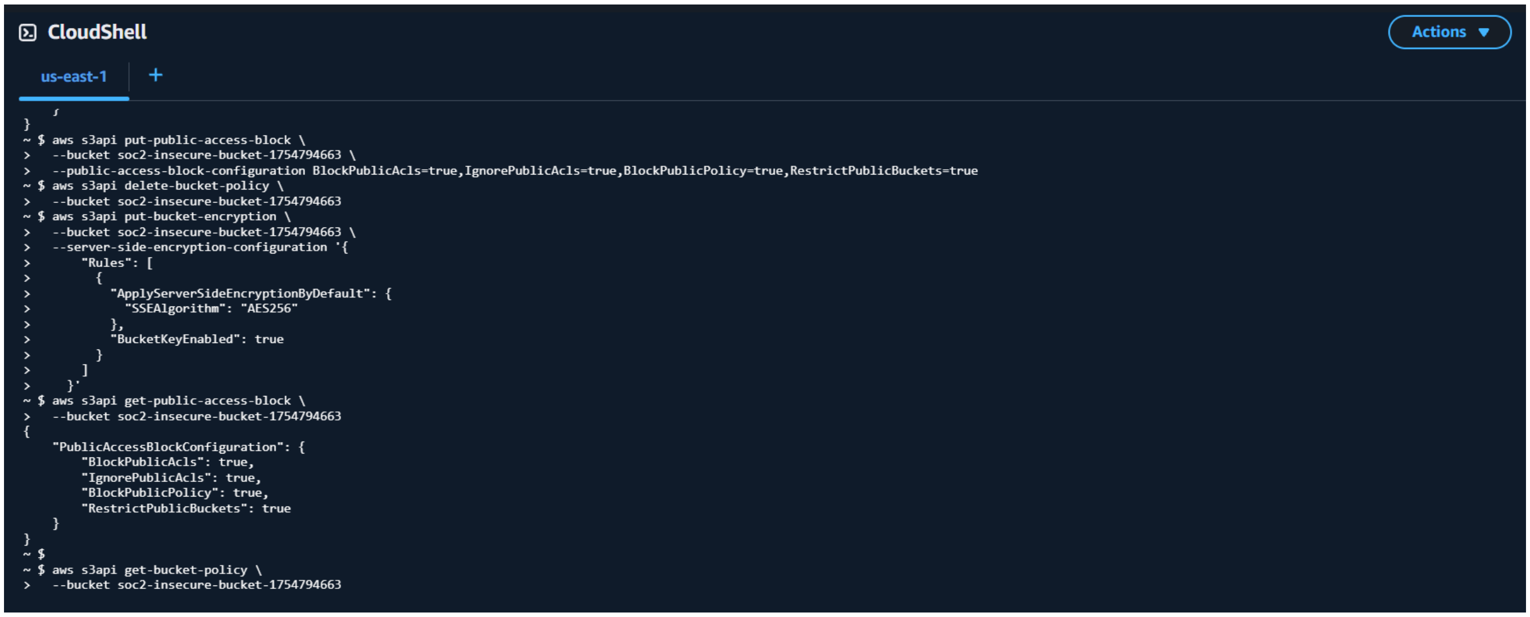
```
                "BucketKeyEnabled": false
            }
        ]
    }
}
~ $ aws s3api put-public-access-block \
>   --bucket soc2-insecure-bucket-1754794663 \
>   --public-access-block-configuration BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=true
~ $ aws s3api delete-bucket-policy \
>   --bucket soc2-insecure-bucket-1754794663
~ $
```

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.



```
~ $ aws s3api get-bucket-encryption --bucket soc2-insecure-bucket-1754794663
{
    "ServerSideEncryptionConfiguration": {
        "Rules": [
            {
                "ApplyServerSideEncryptionByDefault": {
                    "SSEAlgorithm": "AES256"
                },
                "BucketKeyEnabled": false
            }
        ]
    }
}
~ $ aws s3api put-public-access-block \
>   --bucket soc2-insecure-bucket-1754794663 \
>   --public-access-block-configuration BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=true
~ $ aws s3api delete-bucket-policy \
>   --bucket soc2-insecure-bucket-1754794663
~ $ aws s3api put-bucket-encryption \
>   --bucket soc2-insecure-bucket-1754794663 \
>   --server-side-encryption-configuration '{
>       "Rules": [
>         {
>           "ApplyServerSideEncryptionByDefault": {
>             "SSEAlgorithm": "AES256"
>           },
>           "BucketKeyEnabled": true
>         }
>       ]
>     }'
~ $
```

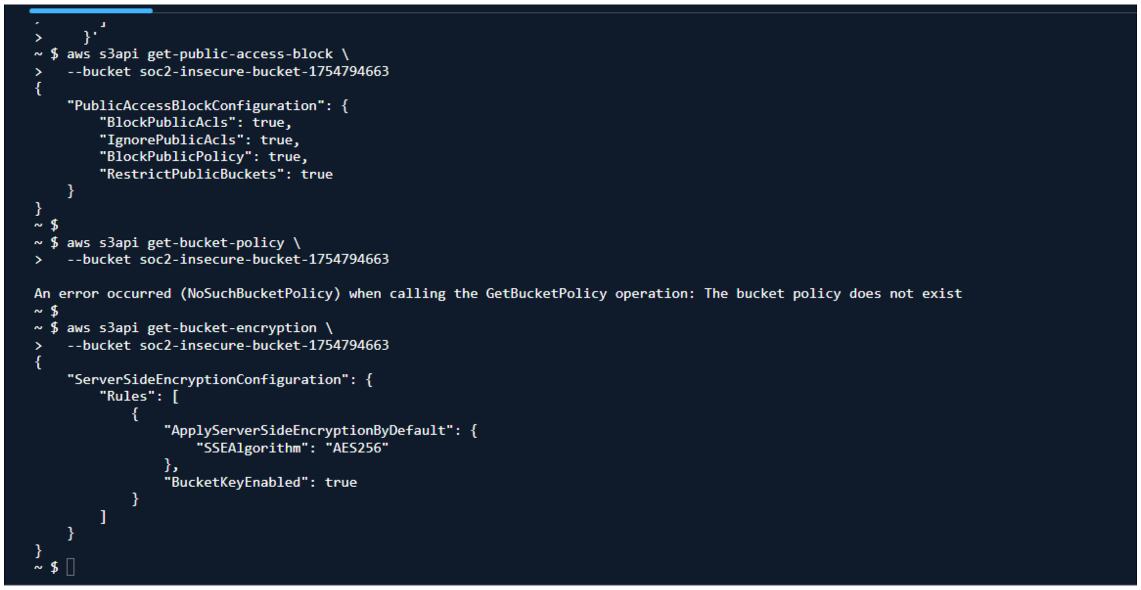
Step 12– Confirm Public Access Block and Encryption Status
Verified the public access block configuration and server-side encryption status after remediation. The output confirmed that all public access blocks were active, and encryption was enabled with a bucket key, providing evidence of restored SOC 2 compliance.



```
}
~ $ aws s3api put-public-access-block \
>   --bucket soc2-insecure-bucket-1754794663 \
>   --public-access-block-configuration BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=true
~ $ aws s3api delete-bucket-policy \
>   --bucket soc2-insecure-bucket-1754794663
~ $ aws s3api put-bucket-encryption \
>   --bucket soc2-insecure-bucket-1754794663 \
>   --server-side-encryption-configuration '{
>       "Rules": [
>         {
>           "ApplyServerSideEncryptionByDefault": {
>             "SSEAlgorithm": "AES256"
>           },
>           "BucketKeyEnabled": true
>         }
>       ]
>     }'
~ $ aws s3api get-public-access-block \
>   --bucket soc2-insecure-bucket-1754794663
{
    "PublicAccessBlockConfiguration": {
        "BlockPublicAcls": true,
        "IgnorePublicAcls": true,
        "BlockPublicPolicy": true,
        "RestrictPublicBuckets": true
    }
}
~ $
~ $ aws s3api get-bucket-policy \
>   --bucket soc2-insecure-bucket-1754794663
```

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.

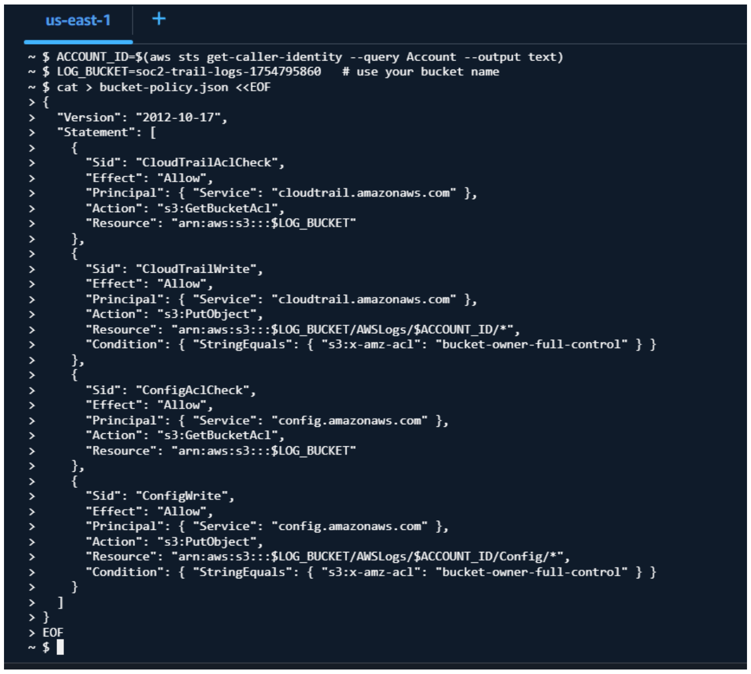


```
>     }'
~ $ aws s3api get-public-access-block \
>   --bucket soc2-insecure-bucket-1754794663
{
    "PublicAccessBlockConfiguration": {
        "BlockPublicAcls": true,
        "IgnorePublicAcls": true,
        "BlockPublicPolicy": true,
        "RestrictPublicBuckets": true
    }
}
~ $
~ $ aws s3api get-bucket-policy \
>   --bucket soc2-insecure-bucket-1754794663

An error occurred (NoSuchBucketPolicy) when calling the GetBucketPolicy operation: The bucket policy does not exist
~ $
~ $ aws s3api get-bucket-encryption \
>   --bucket soc2-insecure-bucket-1754794663
{
    "ServerSideEncryptionConfiguration": {
        "Rules": [
            {
                "ApplyServerSideEncryptionByDefault": {
                    "SSEAlgorithm": "AES256"
                },
                "BucketKeyEnabled": true
            }
        ]
    }
}
~ $
```

Step 14 – Create New CloudTrail Log Bucket

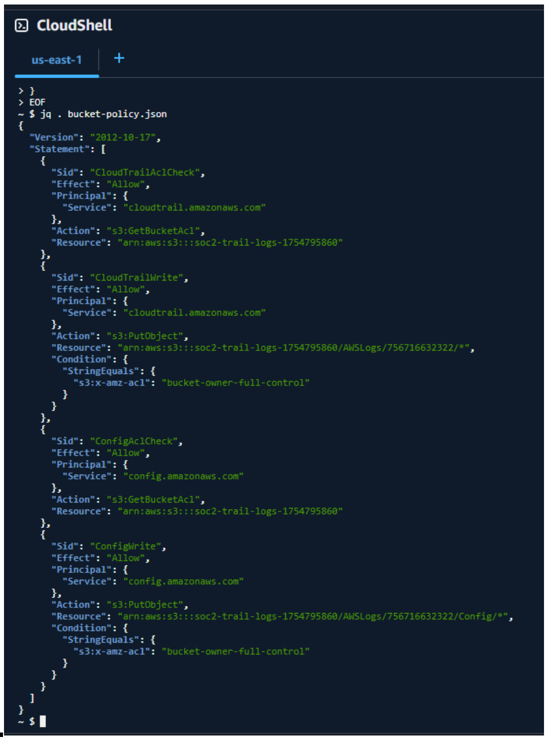
Provisioned a new S3 bucket named soc2-trail-logs-<timestamp> for CloudTrail logs using the AWS CLI. This step demonstrated the secure creation of a dedicated logging bucket, a critical element in SOC 2's requirement for immutable and tamper-proof audit logging.



```
~ $ ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
~ $ LOG_BUCKET=soc2-trail-logs-1754795860   # use your bucket name
~ $ cat > bucket-policy.json <<EOF
> {
>   "Version": "2012-10-17",
>   "Statement": [
>     {
>       "Sid": "CloudTrailAclCheck",
>       "Effect": "Allow",
>       "Principal": { "Service": "cloudtrail.amazonaws.com" },
>       "Action": "s3:GetBucketAcl",
>       "Resource": "arn:aws:s3:::$LOG_BUCKET"
>     },
>     {
>       "Sid": "CloudTrailWrite",
>       "Effect": "Allow",
>       "Principal": { "Service": "cloudtrail.amazonaws.com" },
>       "Action": "s3:PutObject",
>       "Resource": "arn:aws:s3:::$LOG_BUCKET/AWSLogs/$ACCOUNT_ID/*",
>       "Condition": { "StringEquals": { "s3:x-amz-acl": "bucket-owner-full-control" } }
>     },
>     {
>       "Sid": "ConfigAclCheck",
>       "Effect": "Allow",
>       "Principal": { "Service": "config.amazonaws.com" },
>       "Action": "s3:GetBucketAcl",
>       "Resource": "arn:aws:s3:::$LOG_BUCKET"
>     },
>     {
>       "Sid": "ConfigWrite",
>       "Effect": "Allow",
>       "Principal": { "Service": "config.amazonaws.com" },
>       "Action": "s3:PutObject",
>       "Resource": "arn:aws:s3:::$LOG_BUCKET/AWSLogs/$ACCOUNT_ID/Config/*",
>       "Condition": { "StringEquals": { "s3:x-amz-acl": "bucket-owner-full-control" } }
>     }
>   ]
> }
> EOF
~ $
```

Step 15 – Create Combined CloudTrail and AWS Config Bucket Policy

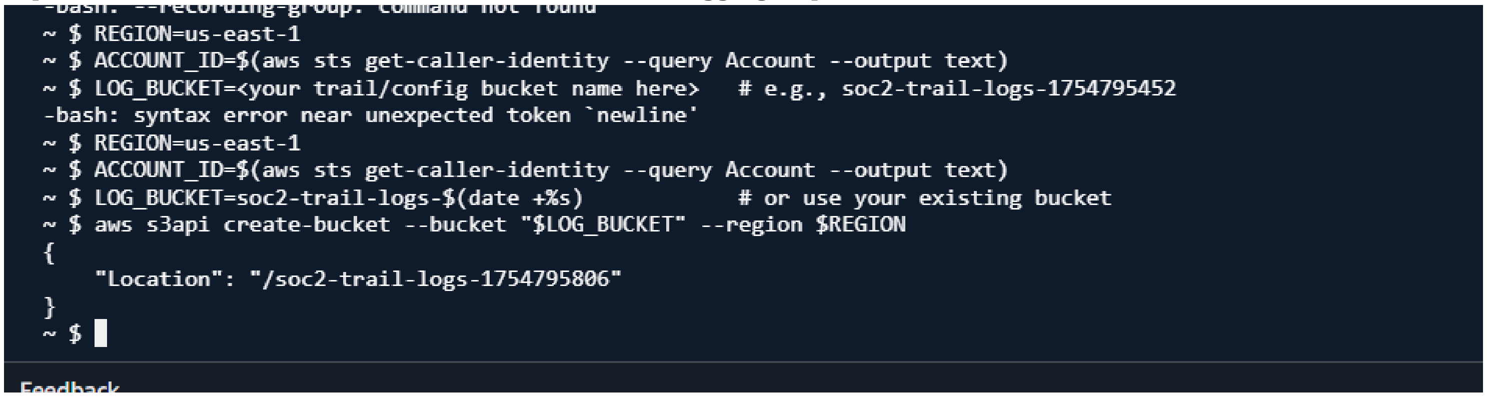
Created a bucket policy granting both cloudtrail.amazonaws.com and config.amazonaws.com services the necessary s3:GetBucketAcl and s3:PutObject permissions to store logs in the soc2-trail-logs bucket. This unified logging policy ensured that both audit trails and configuration snapshots could be stored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



```
> }
> EOF
~ $ jq . bucket-policy.json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CloudTrailAclCheck",
      "Effect": "Allow",
      "Principal": {
        "Service": "cloudtrail.amazonaws.com"
      },
      "Action": "s3:GetBucketAcl",
      "Resource": "arn:aws:s3:::soc2-trail-logs-1754795860"
    },
    {
      "Sid": "CloudTrailWrite",
      "Effect": "Allow",
      "Principal": {
        "Service": "cloudtrail.amazonaws.com"
      },
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::soc2-trail-logs-1754795860/AWSLogs/756716632322/*",
      "Condition": {
        "StringEquals": {
          "s3:x-amz-acl": "bucket-owner-full-control"
        }
      }
    },
    {
      "Sid": "ConfigAclCheck",
      "Effect": "Allow",
      "Principal": {
        "Service": "config.amazonaws.com"
      },
      "Action": "s3:GetBucketAcl",
      "Resource": "arn:aws:s3:::soc2-trail-logs-1754795860"
    },
    {
      "Sid": "ConfigWrite",
      "Effect": "Allow",
      "Principal": {
        "Service": "config.amazonaws.com"
      },
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::soc2-trail-logs-1754795860/AWSLogs/756716632322/Config/*",
      "Condition": {
        "StringEquals": {
          "s3:x-amz-acl": "bucket-owner-full-control"
        }
      }
    }
  ]
}
~ $
```

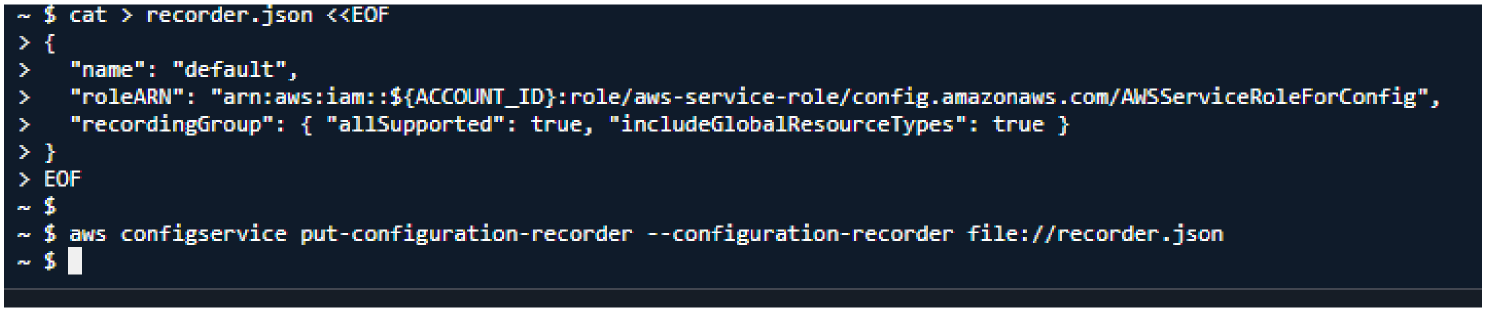
Step 16 – Create and Start CloudTrail

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.



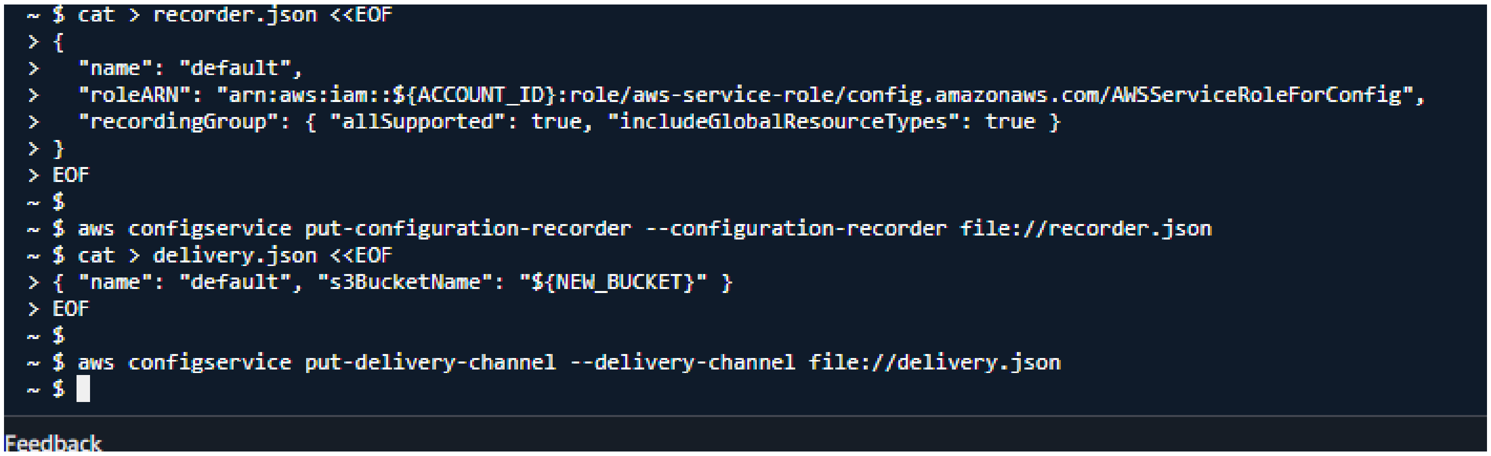
```
-bash: --recording-group: command not found
~ $ REGION=us-east-1
~ $ ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
~ $ LOG_BUCKET=<your trail/config bucket name here>   # e.g., soc2-trail-logs-1754795452
-bash: syntax error near unexpected token `newline'
~ $ REGION=us-east-1
~ $ ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
~ $ LOG_BUCKET=soc2-trail-logs-$(date +%s)            # or use your existing bucket
~ $ aws s3api create-bucket --bucket "$LOG_BUCKET" --region $REGION
{
    "Location": "/soc2-trail-logs-1754795806"
}
~ $
```

Step 17 – 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
~ $
```

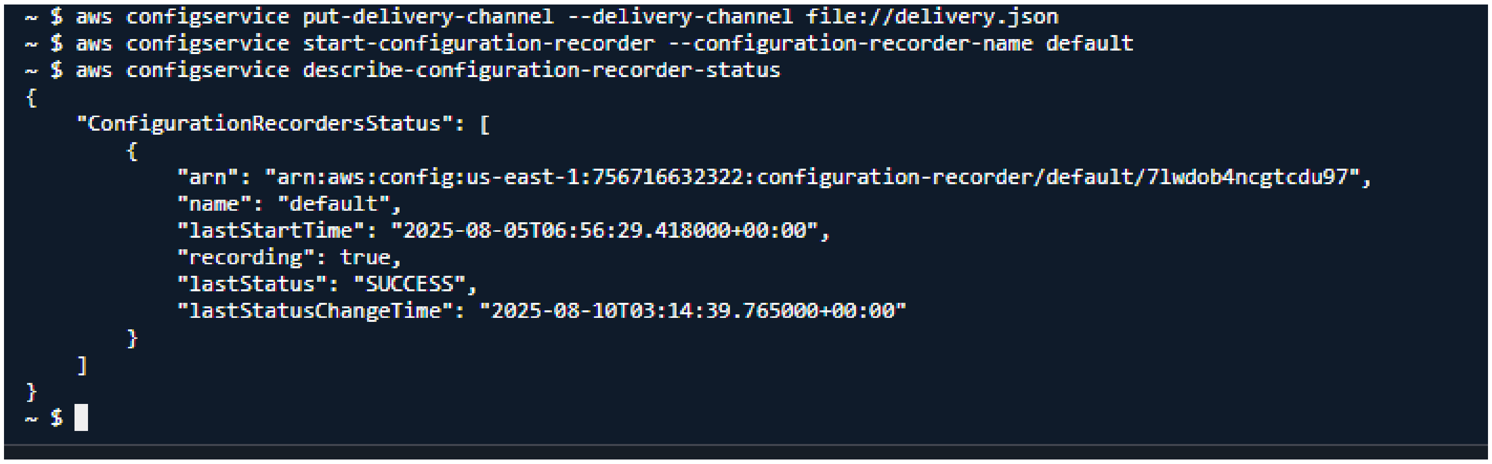
Step 18 – Configure AWS Config Delivery Channel
Created a delivery.json file specifying the new S3 bucket for AWS Config data storage and applied it as the AWS Config delivery channel. This ensured that all configuration history and compliance snapshots would be stored securely for SOC 2 evidence purposes.



```
~ $ 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

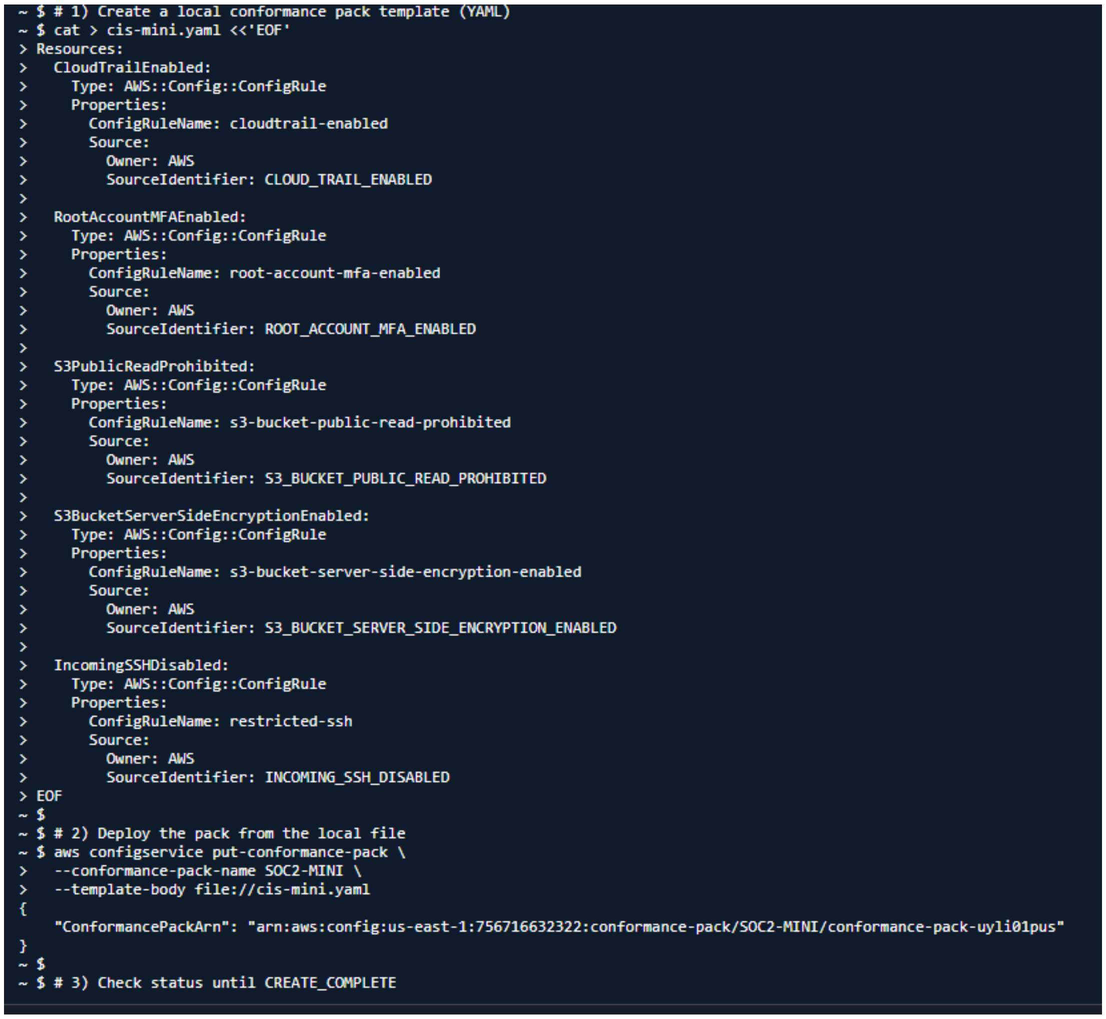
Step 19 – Start AWS Config Recording and Verify Status
Enabled the AWS Config recorder and verified its operational status via AWS CLI, confirming it was actively recording with a SUCCESS state. This step demonstrated operational evidence of SOC 2 control monitoring being in place.



```
~ $ 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 20 – Create SOC 2 Conformance Pack Template

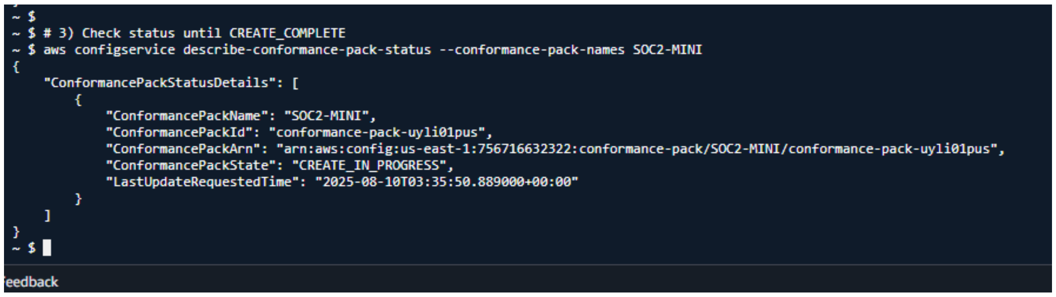
Drafted a cis-mini.yaml conformance pack template containing AWS Config rules for key SOC 2 controls: enabling CloudTrail, enforcing MFA on the root account, prohibiting public S3 buckets, requiring server-side encryption on S3, and restricting SSH access. This template served as an automated compliance 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 21 – Deploy SOC 2 Conformance Pack

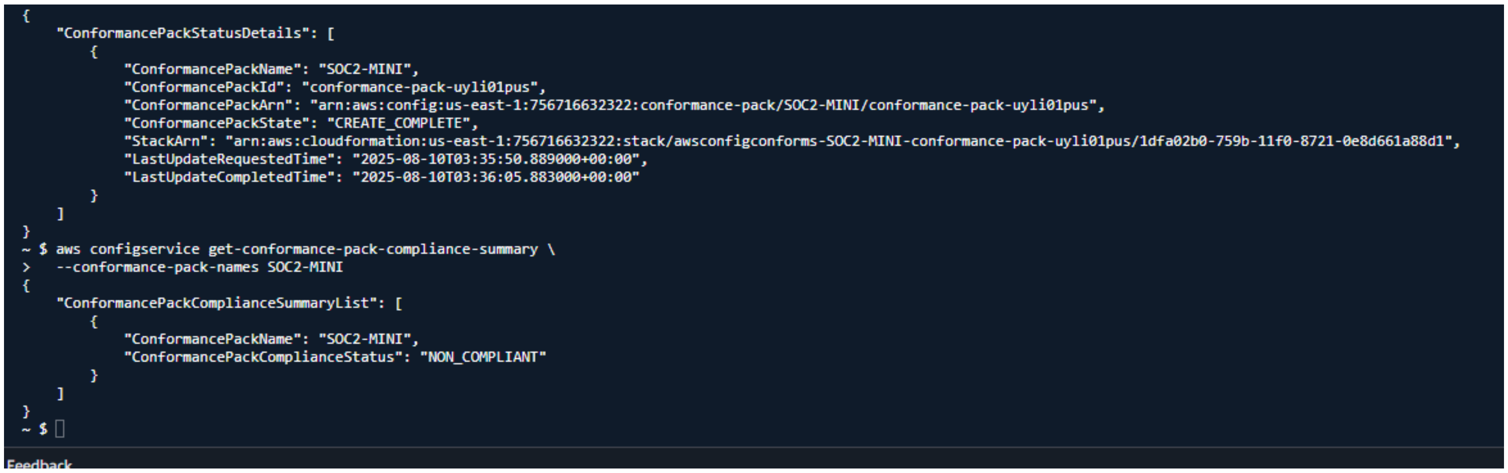
Deployed the conformance pack to the AWS account using the AWS CLI, initiating the creation of compliance checks for all the included rules. This provided a structured, automated method to continuously evaluate SOC 2-related controls.



```
~ $
~ $ # 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"
        }
    ]
}
~ $
```

Step 22 – 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.

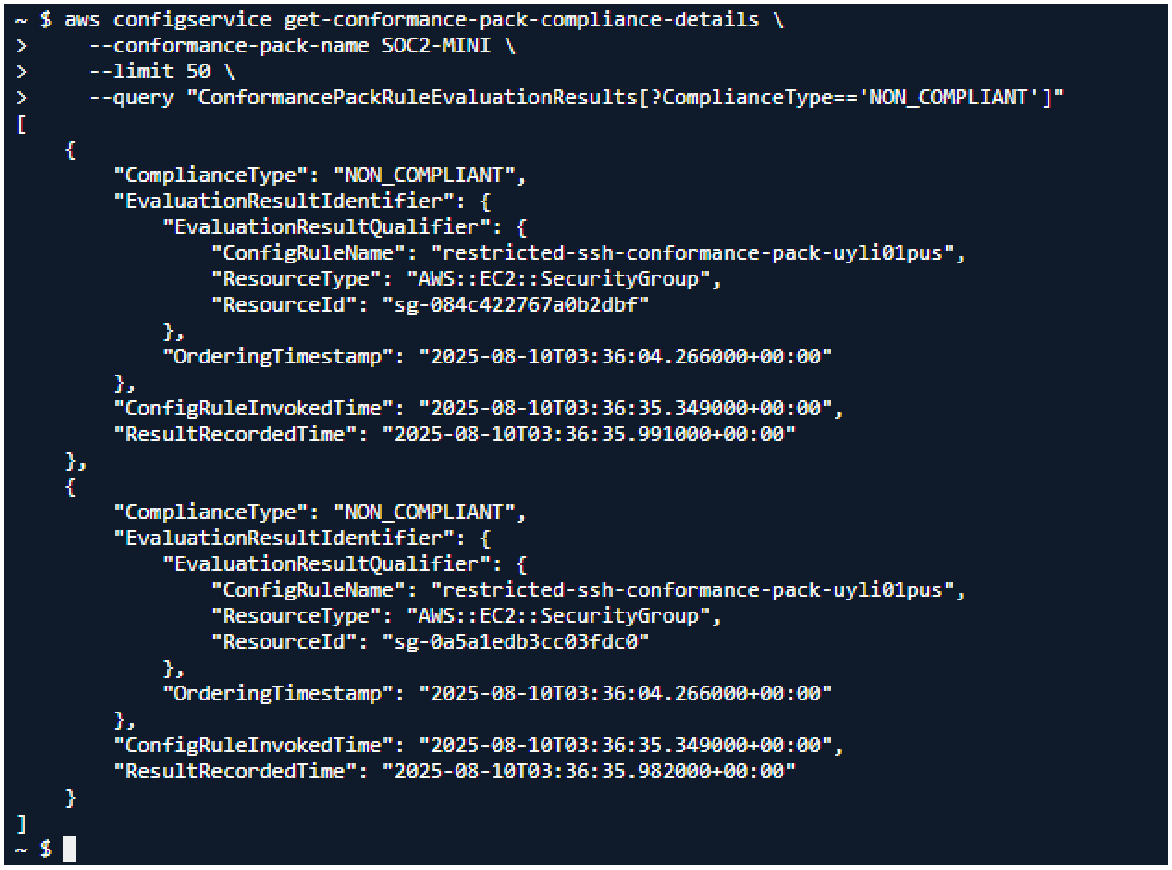


```
{
    "ConformancePackStatusDetails": [
        {
            "ConformancePackName": "SOC2-MINI",
            "ConformancePackId": "conformance-pack-uyli01pus",
            "ConformancePackArn": "arn:aws:config:us-east-1:756716632322:conformance-pack/SOC2-MINI/conformance-pack-uyli01pus",
            "ConformancePackState": "CREATE_COMPLETE",
            "StackArn": "arn:aws:cloudformation:us-east-1:756716632322:stack/awsconfigconforms-SOC2-MINI-conformance-pack-uyli01pus/1dfa02b0-759b-11f0-8721-0e8d661a88d1",
            "LastUpdateRequestedTime": "2025-08-10T03:35:50.889000+00:00",
            "LastUpdateCompletedTime": "2025-08-10T03:36:05.883000+00:00"
        }
    ]
}
~ $ aws configservice get-conformance-pack-compliance-summary \
>   --conformance-pack-names SOC2-MINI
{
    "ConformancePackComplianceSummaryList": [
        {
            "ConformancePackName": "SOC2-MINI",
            "ConformancePackComplianceStatus": "NON_COMPLIANT"
        }
    ]
}
~ $
```

Feedback

Step 23 – Identify Non-Compliant Resources

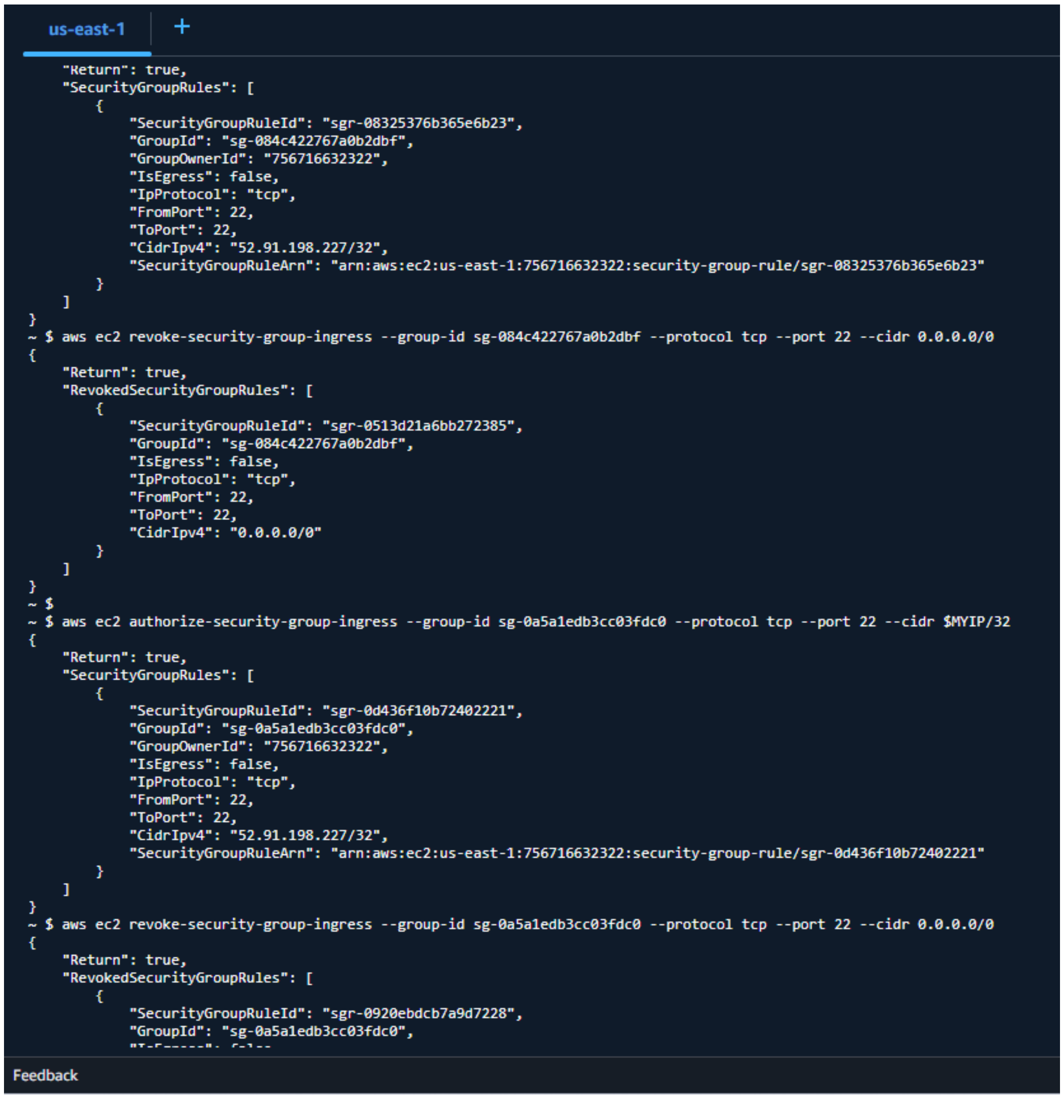
Filtered the compliance results to only display resources marked as non-compliant. Confirmed that the failing resources were two EC2 security groups with open SSH rules to all IP addresses.



```
~ $ 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 24 – Fix Security Group Issues

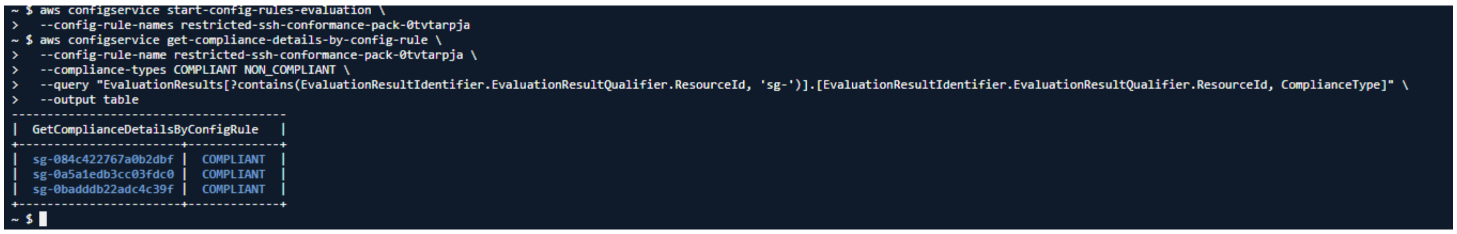
Reviewed the inbound rules for each failing security group. Removed the unrestricted SSH access and replaced it with rules allowing SSH only from a trusted IP address.



```
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-08325376b365e6b23",
            "GroupId": "sg-084c422767a0b2dbf",
            "GroupOwnerId": "756716632322",
            "IsEgress": false,
            "IpProtocol": "tcp",
            "FromPort": 22,
            "ToPort": 22,
            "CidrIpv4": "52.91.198.227/32",
            "SecurityGroupRuleArn": "arn:aws:ec2:us-east-1:756716632322:security-group-rule/sgr-08325376b365e6b23"
        }
    ]
}
~ $ aws ec2 revoke-security-group-ingress --group-id sg-084c422767a0b2dbf --protocol tcp --port 22 --cidr 0.0.0.0/0
{
    "Return": true,
    "RevokedSecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-0513d21a6bb272385",
            "GroupId": "sg-084c422767a0b2dbf",
            "IsEgress": false,
            "IpProtocol": "tcp",
            "FromPort": 22,
            "ToPort": 22,
            "CidrIpv4": "0.0.0.0/0"
        }
    ]
}
~ $
~ $ aws ec2 authorize-security-group-ingress --group-id sg-0a5a1edb3cc03fdc0 --protocol tcp --port 22 --cidr $MYIP/32
{
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-0d436f10b72402221",
            "GroupId": "sg-0a5a1edb3cc03fdc0",
            "GroupOwnerId": "756716632322",
            "IsEgress": false,
            "IpProtocol": "tcp",
            "FromPort": 22,
            "ToPort": 22,
            "CidrIpv4": "52.91.198.227/32",
            "SecurityGroupRuleArn": "arn:aws:ec2:us-east-1:756716632322:security-group-rule/sgr-0d436f10b72402221"
        }
    ]
}
~ $ aws ec2 revoke-security-group-ingress --group-id sg-0a5a1edb3cc03fdc0 --protocol tcp --port 22 --cidr 0.0.0.0/0
{
    "Return": true,
    "RevokedSecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-0920ebdcb7a9d7228",
            "GroupId": "sg-0a5a1edb3cc03fdc0",
```

Step 25 – Re-run Compliance Check

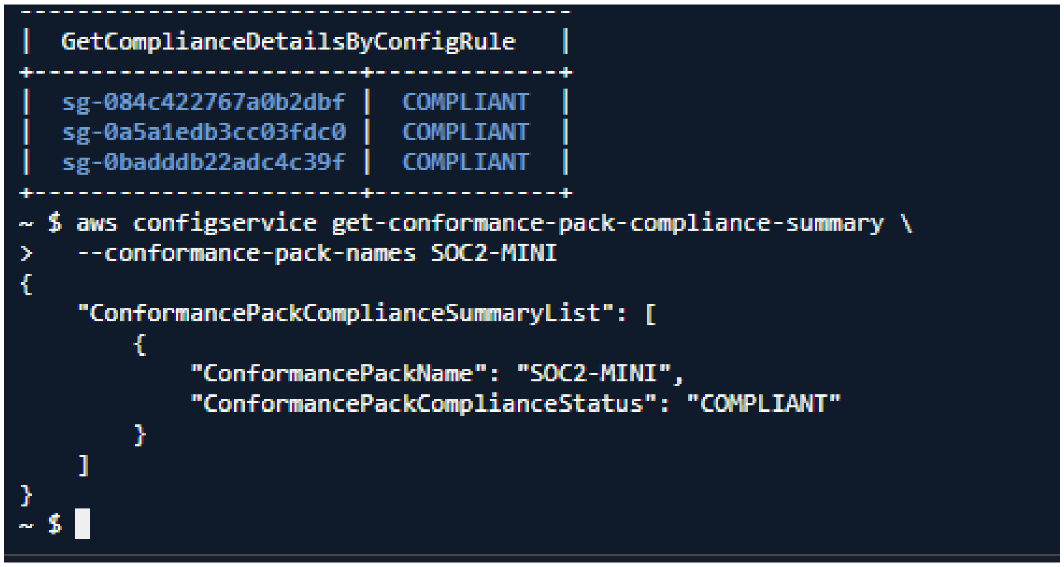
Triggered a re-evaluation of the SSH restriction rule. Verified that both previously failing security groups were now marked as compliant.



```
~ $ aws configservice start-config-rules-evaluation \
>   --config-rule-names restricted-ssh-conformance-pack-0tvtarpja
~ $ aws configservice get-compliance-details-by-config-rule \
>   --config-rule-name restricted-ssh-conformance-pack-0tvtarpja \
>   --compliance-types COMPLIANT NON_COMPLIANT \
>   --query "EvaluationResults[?contains(EvaluationResultIdentifier.EvaluationResultQualifier.ResourceId, 'sg-')].[EvaluationResultIdentifier.EvaluationResultQualifier.ResourceId, ComplianceType]" \
>   --output table
-------------------------------------
|  GetComplianceDetailsByConfigRule  |
+-----------------------+------------+
|  sg-084c422767a0b2dbf |  COMPLIANT |
|  sg-0a5a1edb3cc03fdc0 |  COMPLIANT |
|  sg-0badddb22adc4c39f |  COMPLIANT |
+-----------------------+------------+
~ $
```

Step 26 – Confirm Overall Compliance

Checked the overall status of the SOC2-MINI conformance pack. Confirmed that all rules were compliant and no issues remained



```
----------------------------------------
|  GetComplianceDetailsByConfigRule    |
+-----------------------+--------------+
|  sg-084c422767a0b2dbf |  COMPLIANT   |
|  sg-0a5a1edb3cc03fdc0 |  COMPLIANT   |
|  sg-0badddb22adc4c39f |  COMPLIANT   |
+-----------------------+--------------+
~ $ aws configservice get-conformance-pack-compliance-summary \
>   --conformance-pack-names SOC2-MINI
{
    "ConformancePackComplianceSummaryList": [
        {
            "ConformancePackName": "SOC2-MINI",
            "ConformancePackComplianceStatus": "COMPLIANT"
        }
    ]
}
~ $
```

Conclusion

This engagement demonstrates an end-to-end SOC 2 readiness path: create controlled baselines, enable logging and configuration tracking, test against automated controls, remediate, and verify. The result is a defensible narrative and evidence trail that auditors and business leaders can follow. Beyond this initial pass, we recommend establishing continuous compliance checks, periodic evidence refreshes, and ownership workflows so that posture remains strong over time rather than only at audit time.

Key benefits delivered

- Stronger security posture aligned to SOC 2.
- Lower risk of audit findings and rework.
- Clear ownership and repeatable processes for ongoing compliance.

Added deliverables

- **Created a clear remediation plan with evidence for each issue.**
  Each finding maps to the SOC 2 criterion, risk, fix, owner, due date, and the exact evidence proving remediation, with before-and-after artefacts and a validation method.
- **Produced a deliverable that can be provided directly to clients and auditors.**
  The packaged evidence includes logging and configuration status, control evaluations, pre-fix failures, remediation proof, and final compliant results, organised in an index for fast review.
- **Enabled a faster, smoother audit process.**
  With centralised evidence and verified remediation, auditors spend less time on discovery and more on confirmation, reducing cycle time and disruption to the business.