AWS Cloud Security Posture Assessment - Demonstration Project

Enhancing Cyber Resilience Through Visibility, Compliance, and Risk Reduction

Executive Summary

Objective:

Demonstrate how to identify, prioritise, and remediate common AWS misconfigurations using native services (CloudTrail, AWS Config, Security Hub, GuardDuty), and leave the environment with continuous logging, compliance evaluation, and a clear path to sustained risk reduction.

What we did (this demo).

Enabled CloudTrail and AWS Config (recording SUCCESS), turned on GuardDuty and Security Hub, locked down public S3 access (account + bucket level), removed risky default security group rules, disabled SSM public document sharing, and enabled ECR enhanced scanning. Result: Security Hub active findings trending to 0/0/0 (Crit/High/Med) with evidence suitable for audit.

Why This Matters:

Misconfiguration is still the number 1 cloud breach vector. This approach reduces attack surface, provides board-level visibility, and aligns to CIS AWS Foundations, supporting SOC 2/NIST readiness.

Business Value Delivered:

- Reduced Risk: Early detection of high-impact vulnerabilities (e.g., open SSH ports, lack of MFA).

- Audit Readiness: Measured against CIS AWS Foundations with clear evidence and remediation steps.

- Operational Efficiency: Centralised findings via AWS Security Hub for streamlined remediation.

- Board-Level Reporting: Executive compliance summary for leadership and auditors.

Approach:

1. Baseline Posture Review – Establish current security state via S3, IAM, Security Group, and CloudTrail assessments.

2. Visibility Enablement – Deploy AWS Config and CloudTrail to capture all configuration and API activity.

3. Compliance Benchmarking – Apply the CIS AWS Foundations Benchmark to measure compliance levels.

4. Consolidated Reporting – Aggregate all findings in AWS Security Hub for a single source of truth.

5. Risk Remediation Roadmap – Prioritise remediation efforts based on risk severity and compliance impact.

Key Outcomes (from this Demo):

- Identified publicly exposed S3 buckets and insecure IAM admin credentials.

- Closed open SSH access to the internet (0.0.0.0/0).

- Enabled comprehensive activity logging via CloudTrail.

- Deployed CIS benchmark conformance packs to track and report compliance.

- Produced pack-level and rule-level evidence of current posture:
  (COMPLIANT / NON_COMPLIANT / INSUFFICIENT_DATA) and a remediation path to reach compliance.

- Integrated Security Hub and GuardDuty for ongoing threat detection.

- Produced a compliance summary report suitable for executive and audit teams.

Strategic Alignment:

This assessment directly supports enterprise Cyber Transformation goals by:

- Improving Governance: Aligning configurations to recognised standards (CIS, SOC 2 readiness).

- Reducing Threat Surface: Eliminating high-risk misconfigurations proactively.

- Enhancing Resilience: Establishing ongoing monitoring and alerting capabilities.

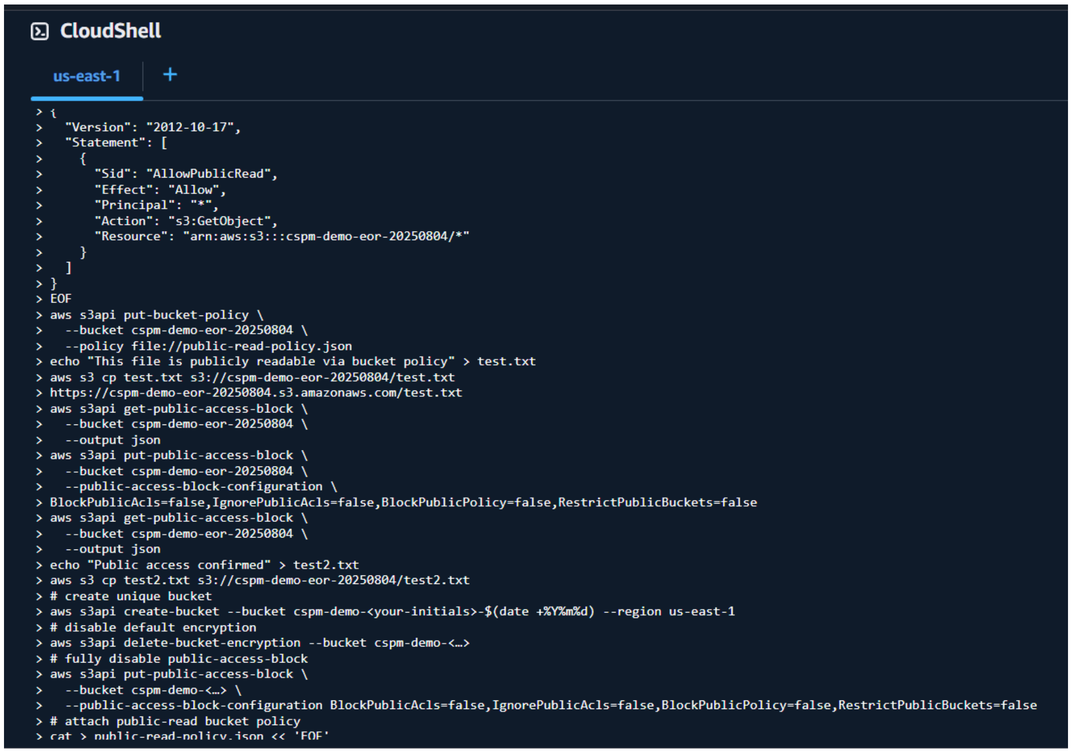
- Supporting Business Objectives: Enabling secure growth in the cloud without operational slowdowns.

<u>Cloud Security Posture Demo (AWS CSPM Walkthrough)</u>

This demo simulates an insecure AWS environment and walks through detection and remediation using best practices - ideal for clients looking to improve visibility and control of their AWS accounts.

<u>Baseline Posture Review</u>

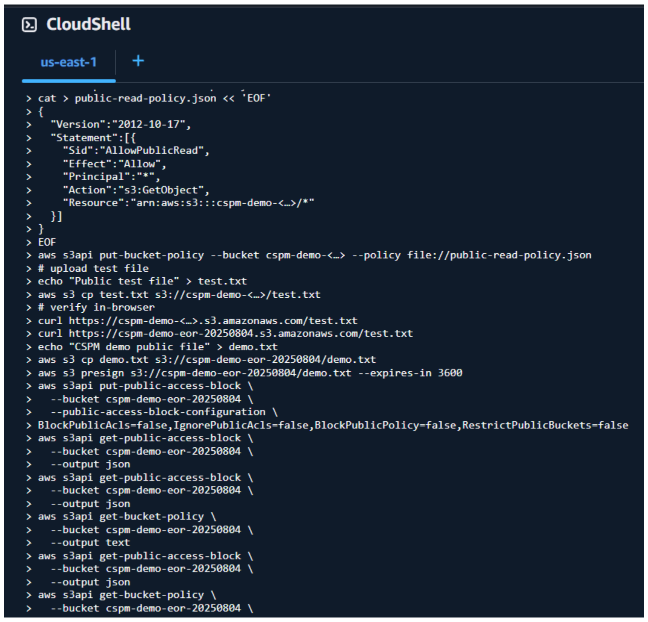
1. Public S3 Bucket Policy Configured



```
> {
>   "Version": "2012-10-17",
>   "Statement": [
>     {
>       "Sid": "AllowPublicRead",
>       "Effect": "Allow",
>       "Principal": "*",
>       "Action": "s3:GetObject",
>       "Resource": "arn:aws:s3:::cspm-demo-eor-20250804/*"
>     }
>   ]
> }
> EOF
> aws s3api put-bucket-policy \
>   --bucket cspm-demo-eor-20250804 \
>   --policy file://public-read-policy.json
> echo "This file is publicly readable via bucket policy" > test.txt
> aws s3 cp test.txt s3://cspm-demo-eor-20250804/test.txt
> https://cspm-demo-eor-20250804.s3.amazonaws.com/test.txt
> aws s3api get-public-access-block \
>   --bucket cspm-demo-eor-20250804 \
>   --output json
> aws s3api put-public-access-block \
>   --bucket cspm-demo-eor-20250804 \
>   --public-access-block-configuration \
> BlockPublicAcls=false,IgnorePublicAcls=false,BlockPublicPolicy=false,RestrictPublicBuckets=false
> aws s3api get-public-access-block \
>   --bucket cspm-demo-eor-20250804 \
>   --output json
> echo "Public access confirmed" > test2.txt
> aws s3 cp test2.txt s3://cspm-demo-eor-20250804/test2.txt
> # create unique bucket
> aws s3api create-bucket --bucket cspm-demo-<your-initials>-$(date +%Y%m%d) --region us-east-1
> # disable default encryption
> aws s3api delete-bucket-encryption --bucket cspm-demo-<…>
> # fully disable public-access-block
> aws s3api put-public-access-block \
>   --bucket cspm-demo-<…> \
>   --public-access-block-configuration BlockPublicAcls=false,IgnorePublicAcls=false,BlockPublicPolicy=false,RestrictPublicBuckets=false
> # attach public-read bucket policy
> cat > public-read-policy.json << 'EOF'
```

I created a test S3 bucket and attached a policy that makes all files inside publicly readable. This is a common misconfiguration that exposes sensitive files to the internet without restriction.

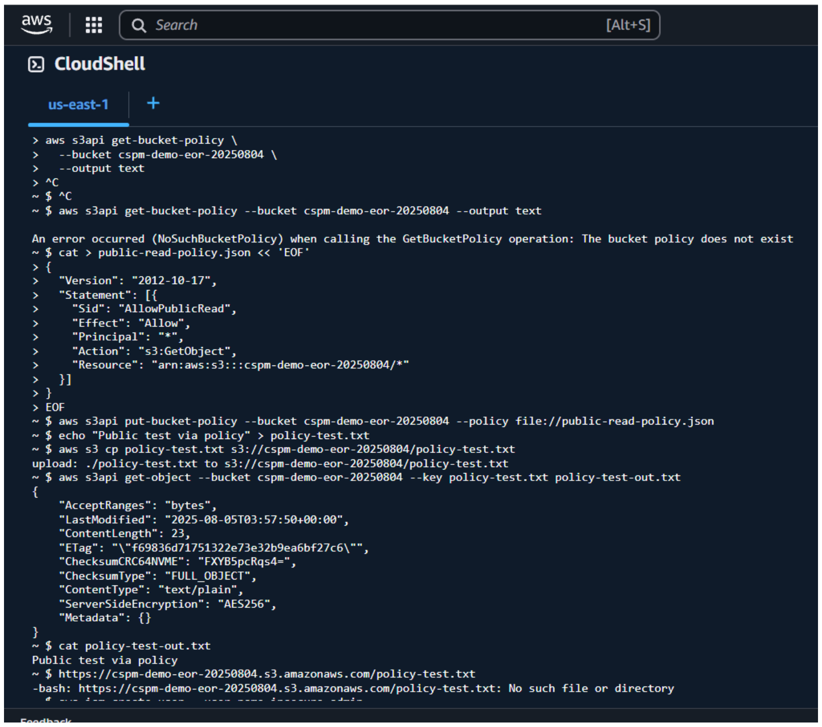
2. Public Access Confirmed via Policy and Uploads



```
> cat > public-read-policy.json << 'EOF'
> {
>   "Version":"2012-10-17",
>   "Statement":[{
>     "Sid":"AllowPublicRead",
>     "Effect":"Allow",
>     "Principal":"*",
>     "Action":"s3:GetObject",
>     "Resource":"arn:aws:s3:::cspm-demo-<…>/*"
>   }]
> }
> EOF
> aws s3api put-bucket-policy --bucket cspm-demo-<…> --policy file://public-read-policy.json
> # upload test file
> echo "Public test file" > test.txt
> aws s3 cp test.txt s3://cspm-demo-<…>/test.txt
> # verify in-browser
> curl https://cspm-demo-<…>.s3.amazonaws.com/test.txt
> curl https://cspm-demo-eor-20250804.s3.amazonaws.com/test.txt
> echo "CSPM demo public file" > demo.txt
> aws s3 cp demo.txt s3://cspm-demo-eor-20250804/demo.txt
> aws s3 presign s3://cspm-demo-eor-20250804/demo.txt --expires-in 3600
> aws s3api put-public-access-block \
>   --bucket cspm-demo-eor-20250804 \
>   --public-access-block-configuration \
> BlockPublicAcls=false,IgnorePublicAcls=false,BlockPublicPolicy=false,RestrictPublicBuckets=false
> aws s3api get-public-access-block \
>   --bucket cspm-demo-eor-20250804 \
>   --output json
> aws s3api get-public-access-block \
>   --bucket cspm-demo-eor-20250804 \
>   --output json
> aws s3api get-bucket-policy \
>   --bucket cspm-demo-eor-20250804 \
>   --output text
> aws s3api get-public-access-block \
>   --bucket cspm-demo-eor-20250804 \
>   --output json
> aws s3api get-bucket-policy \
>   --bucket cspm-demo-eor-20250804 \
```

A simple `.txt` file is uploaded and successfully accessed from a public URL, verifying that the misconfigured policy works - highlighting a high-risk exposure.

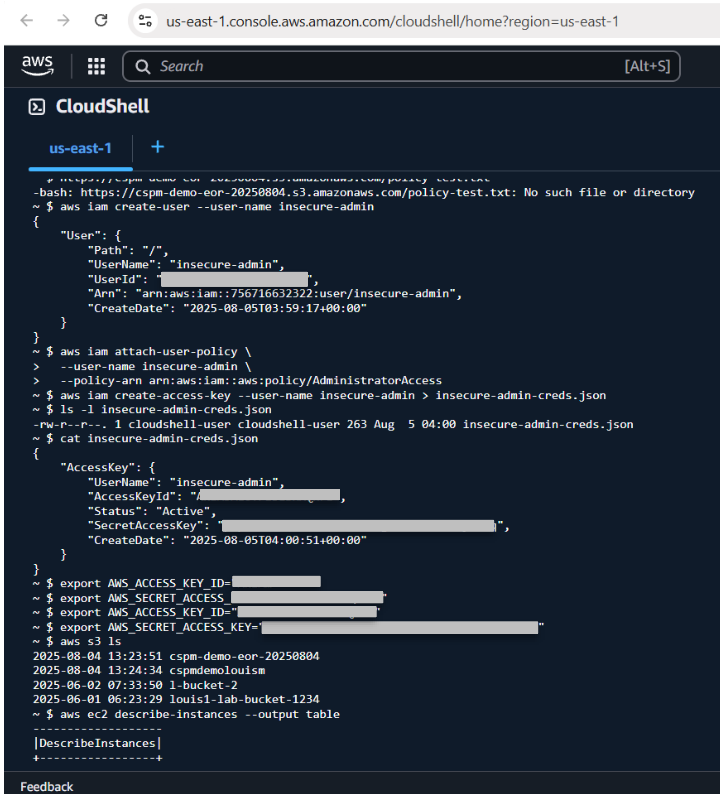
3. Missing Bucket Policy Detected



```
> aws s3api get-bucket-policy \
>   --bucket cspm-demo-eor-20250804 \
>   --output text
> ^C
~ $ ^C
~ $ aws s3api get-bucket-policy --bucket cspm-demo-eor-20250804 --output text

An error occurred (NoSuchBucketPolicy) when calling the GetBucketPolicy operation: The bucket policy does not exist
~ $ cat > public-read-policy.json << 'EOF'
> {
>   "Version": "2012-10-17",
>   "Statement": [{
>     "Sid": "AllowPublicRead",
>     "Effect": "Allow",
>     "Principal": "*",
>     "Action": "s3:GetObject",
>     "Resource": "arn:aws:s3:::cspm-demo-eor-20250804/*"
>   }]
> }
> EOF
~ $ aws s3api put-bucket-policy --bucket cspm-demo-eor-20250804 --policy file://public-read-policy.json
~ $ echo "Public test via policy" > policy-test.txt
~ $ aws s3 cp policy-test.txt s3://cspm-demo-eor-20250804/policy-test.txt
upload: ./policy-test.txt to s3://cspm-demo-eor-20250804/policy-test.txt
~ $ aws s3api get-object --bucket cspm-demo-eor-20250804 --key policy-test.txt policy-test-out.txt
{
    "AcceptRanges": "bytes",
    "LastModified": "2025-08-05T03:57:50+00:00",
    "ContentLength": 23,
    "ETag": "\"f69836d71751322e73e32b9ea6bf27c6\"",
    "ChecksumCRC64NVME": "FXYB5pcRqs4=",
    "ChecksumType": "FULL_OBJECT",
    "ContentType": "text/plain",
    "ServerSideEncryption": "AES256",
    "Metadata": {}
}
~ $ cat policy-test-out.txt
Public test via policy
~ $ https://cspm-demo-eor-20250804.s3.amazonaws.com/policy-test.txt
-bash: https://cspm-demo-eor-20250804.s3.amazonaws.com/policy-test.txt: No such file or directory
```

I attempt to retrieve a bucket policy and receive an error - this simulates a common issue where S3 buckets are left without any enforced security policies. I then attach the insecure public-read policy.

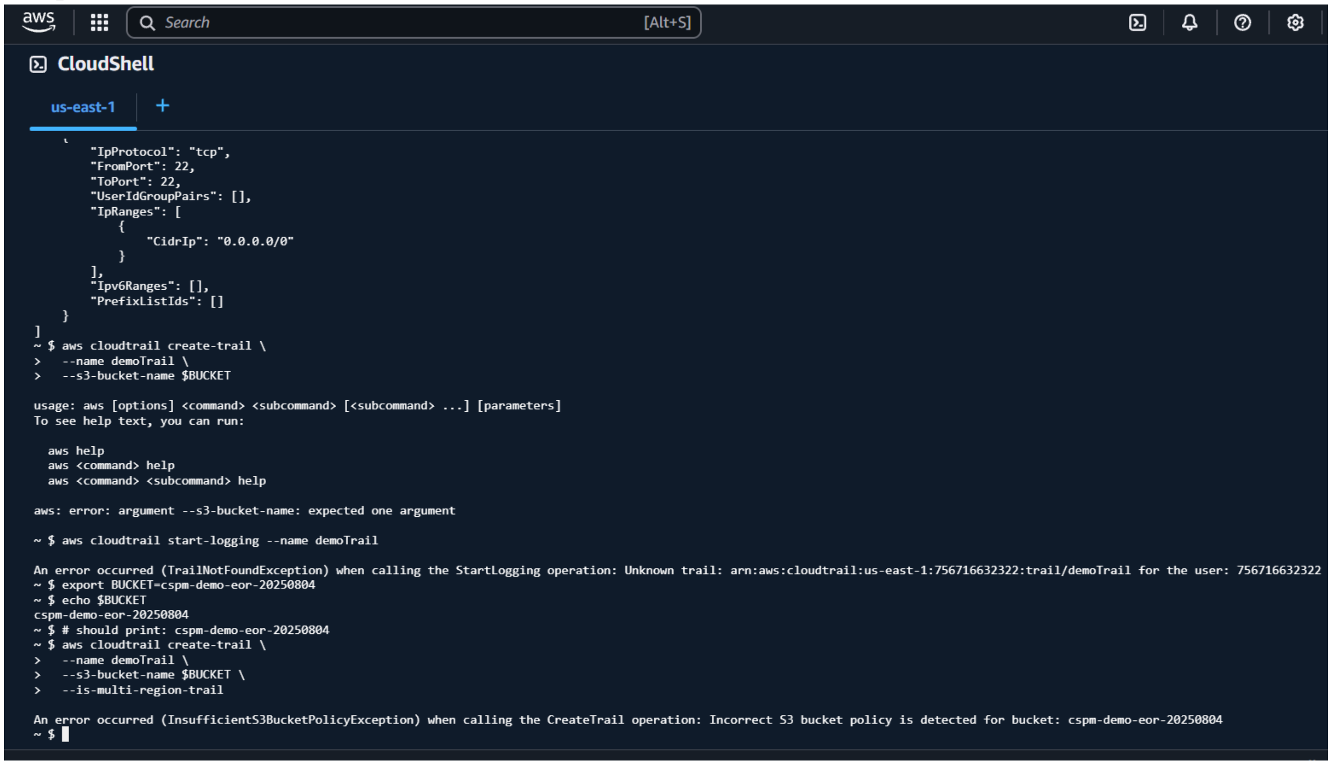
4. Insecure Admin IAM User Created


us-east-1.console.aws.amazon.com/cloudshell/home?region=us-east-1
Search [Alt+S]
CloudShell
us-east-1
Feedback

```
-bash: https://cspm-demo-eor-20250804.s3.amazonaws.com/policy-test.txt: No such file or directory
~ $ aws iam create-user --user-name insecure-admin
{
    "User": {
        "Path": "/",
        "UserName": "insecure-admin",
        "UserId": "[redacted]",
        "Arn": "arn:aws:iam::756716632322:user/insecure-admin",
        "CreateDate": "2025-08-05T03:59:17+00:00"
    }
}
~ $ aws iam attach-user-policy \
>   --user-name insecure-admin \
>   --policy-arn arn:aws:iam::aws:policy/AdministratorAccess
~ $ aws iam create-access-key --user-name insecure-admin > insecure-admin-creds.json
~ $ ls -l insecure-admin-creds.json
-rw-r--r--. 1 cloudshell-user cloudshell-user 263 Aug  5 04:00 insecure-admin-creds.json
~ $ cat insecure-admin-creds.json
{
    "AccessKey": {
        "UserName": "insecure-admin",
        "AccessKeyId": "A[redacted],
        "Status": "Active",
        "SecretAccessKey": "[redacted]",
        "CreateDate": "2025-08-05T04:00:51+00:00"
    }
}
~ $ export AWS_ACCESS_KEY_ID=[redacted]
~ $ export AWS_SECRET_ACCESS_[redacted]"
~ $ export AWS_ACCESS_KEY_ID="[redacted]"
~ $ export AWS_SECRET_ACCESS_KEY="[redacted]"
~ $ aws s3 ls
2025-08-04 13:23:51 cspm-demo-eor-20250804
2025-08-04 13:24:34 cspmdemolouism
2025-06-02 07:33:50 l-bucket-2
2025-06-01 06:23:29 louis1-lab-bucket-1234
~ $ aws ec2 describe-instances --output table
-------------------
|DescribeInstances|
+-----------------+
```

A new IAM user with full admin rights have been created and given hardcoded access keys. This insecure practice is often exploited in breaches and is flagged in cloud security audits.

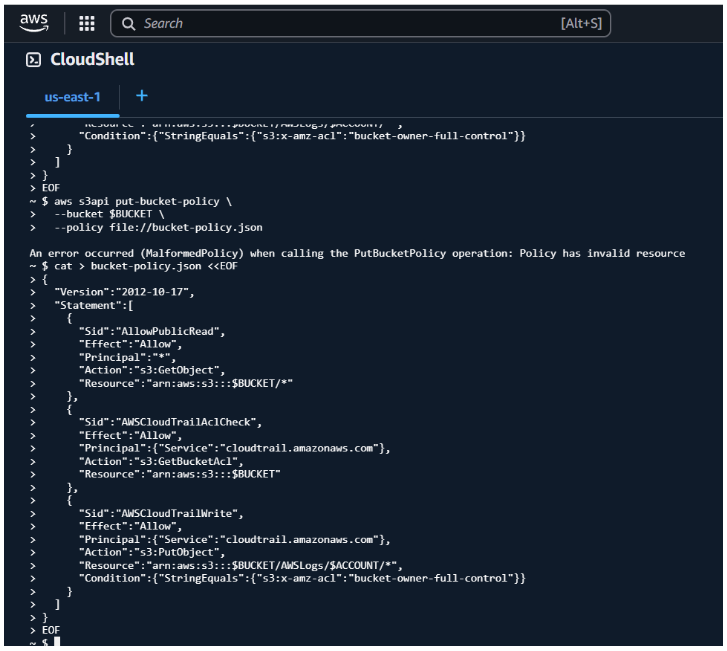
5. Open SSH Port to the World



```
        "IpProtocol": "tcp",
        "FromPort": 22,
        "ToPort": 22,
        "UserIdGroupPairs": [],
        "IpRanges": [
            {
                "CidrIp": "0.0.0.0/0"
            }
        ],
        "Ipv6Ranges": [],
        "PrefixListIds": []
    }
]
~ $ aws cloudtrail create-trail \
>   --name demoTrail \
>   --s3-bucket-name $BUCKET

usage: aws [options] <command> <subcommand> [<subcommand> ...] [parameters]
To see help text, you can run:

  aws help
  aws <command> help
  aws <command> <subcommand> help

aws: error: argument --s3-bucket-name: expected one argument

~ $ aws cloudtrail start-logging --name demoTrail

An error occurred (TrailNotFoundException) when calling the StartLogging operation: Unknown trail: arn:aws:cloudtrail:us-east-1:756716632322:trail/demoTrail for the user: 756716632322
~ $ export BUCKET=cspm-demo-eor-20250804
~ $ echo $BUCKET
cspm-demo-eor-20250804
~ $ # should print: cspm-demo-eor-20250804
~ $ aws cloudtrail create-trail \
>   --name demoTrail \
>   --s3-bucket-name $BUCKET \
>   --is-multi-region-trail

An error occurred (InsufficientS3BucketPolicyException) when calling the CreateTrail operation: Incorrect S3 bucket policy is detected for bucket: cspm-demo-eor-20250804
~ $ 
```

A Security Group is shown allowing port 22 (SSH) access from any IP address (`0.0.0.0/0`), another classic misconfiguration that exposes resources to brute force or unauthorised remote access.

6. CloudTrail Setup Fails: Bucket Policy Invalid

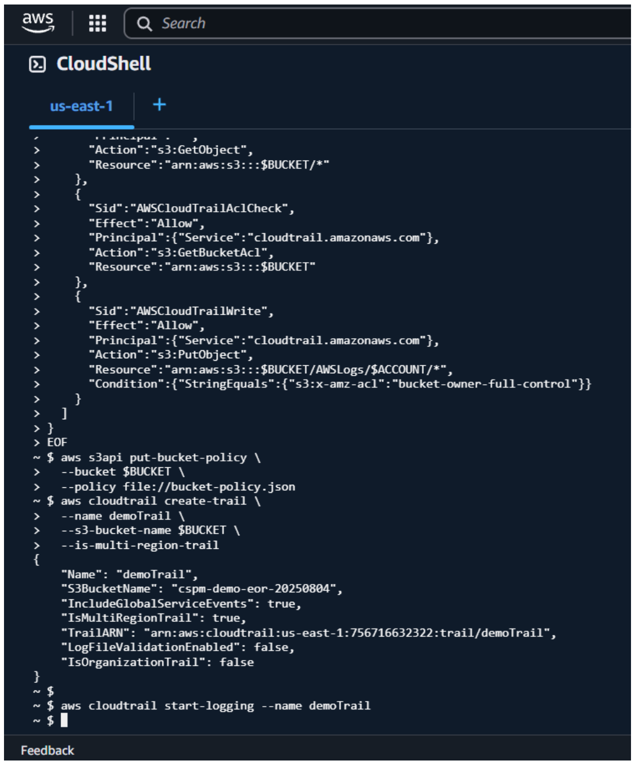


```
>       "Condition":{"StringEquals":{"s3:x-amz-acl":"bucket-owner-full-control"}}
>     }
>   ]
> }
> EOF
~ $ aws s3api put-bucket-policy \
>   --bucket $BUCKET \
>   --policy file://bucket-policy.json

An error occurred (MalformedPolicy) when calling the PutBucketPolicy operation: Policy has invalid resource
~ $ cat > bucket-policy.json <<EOF
> {
>   "Version":"2012-10-17",
>   "Statement":[
>     {
>       "Sid":"AllowPublicRead",
>       "Effect":"Allow",
>       "Principal":"*",
>       "Action":"s3:GetObject",
>       "Resource":"arn:aws:s3:::$BUCKET/*"
>     },
>     {
>       "Sid":"AWSCloudTrailAclCheck",
>       "Effect":"Allow",
>       "Principal":{"Service":"cloudtrail.amazonaws.com"},
>       "Action":"s3:GetBucketAcl",
>       "Resource":"arn:aws:s3:::$BUCKET"
>     },
>     {
>       "Sid":"AWSCloudTrailWrite",
>       "Effect":"Allow",
>       "Principal":{"Service":"cloudtrail.amazonaws.com"},
>       "Action":"s3:PutObject",
>       "Resource":"arn:aws:s3:::$BUCKET/AWSLogs/$ACCOUNT/*",
>       "Condition":{"StringEquals":{"s3:x-amz-acl":"bucket-owner-full-control"}}
>     }
>   ]
> }
> EOF
~ $
```

Attempting to create a CloudTrail fails due to an incorrect S3 bucket policy - this shows the importance of policy validation when setting up security logging infrastructure.

Visibility Enablement

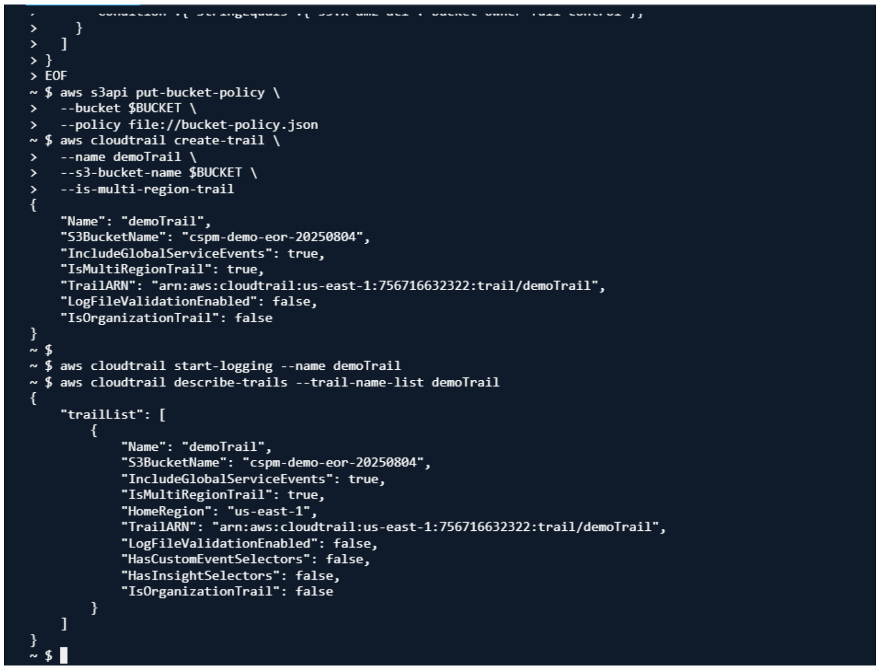
7. Fixing Bucket Policy for CloudTrail Integration



```
>       "Action":"s3:GetObject",
>       "Resource":"arn:aws:s3:::$BUCKET/*"
>     },
>     {
>       "Sid":"AWSCloudTrailAclCheck",
>       "Effect":"Allow",
>       "Principal":{"Service":"cloudtrail.amazonaws.com"},
>       "Action":"s3:GetBucketAcl",
>       "Resource":"arn:aws:s3:::$BUCKET"
>     },
>     {
>       "Sid":"AWSCloudTrailWrite",
>       "Effect":"Allow",
>       "Principal":{"Service":"cloudtrail.amazonaws.com"},
>       "Action":"s3:PutObject",
>       "Resource":"arn:aws:s3:::$BUCKET/AWSLogs/$ACCOUNT/*",
>       "Condition":{"StringEquals":{"s3:x-amz-acl":"bucket-owner-full-control"}}
>     }
>   ]
> }
> EOF
~ $ aws s3api put-bucket-policy \
>   --bucket $BUCKET \
>   --policy file://bucket-policy.json
~ $ aws cloudtrail create-trail \
>   --name demoTrail \
>   --s3-bucket-name $BUCKET \
>   --is-multi-region-trail
{
    "Name": "demoTrail",
    "S3BucketName": "cspm-demo-eor-20250804",
    "IncludeGlobalServiceEvents": true,
    "IsMultiRegionTrail": true,
    "TrailARN": "arn:aws:cloudtrail:us-east-1:756716632322:trail/demoTrail",
    "LogFileValidationEnabled": false,
    "IsOrganizationTrail": false
}
~ $
~ $ aws cloudtrail start-logging --name demoTrail
~ $
```

We correct the bucket policy by adding permissions required by AWS CloudTrail to write logs to the S3 bucket - resolving the error seen in the previous step.

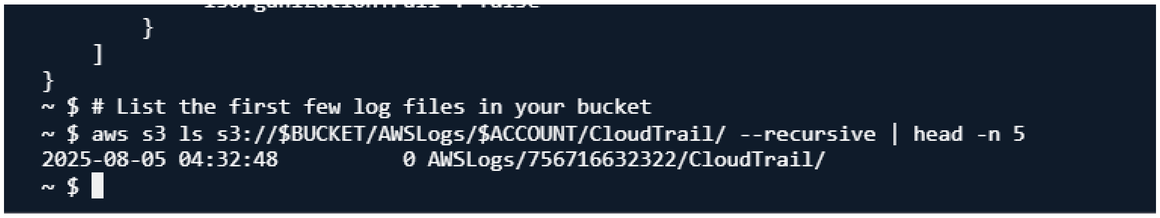
8. CloudTrail Successfully Created



```
>     }
>   ]
> }
> EOF
~ $ aws s3api put-bucket-policy \
>   --bucket $BUCKET \
>   --policy file://bucket-policy.json
~ $ aws cloudtrail create-trail \
>   --name demoTrail \
>   --s3-bucket-name $BUCKET \
>   --is-multi-region-trail
{
    "Name": "demoTrail",
    "S3BucketName": "cspm-demo-eor-20250804",
    "IncludeGlobalServiceEvents": true,
    "IsMultiRegionTrail": true,
    "TrailARN": "arn:aws:cloudtrail:us-east-1:756716632322:trail/demoTrail",
    "LogFileValidationEnabled": false,
    "IsOrganizationTrail": false
}
~ $
~ $ aws cloudtrail start-logging --name demoTrail
~ $ aws cloudtrail describe-trails --trail-name-list demoTrail
{
    "trailList": [
        {
            "Name": "demoTrail",
            "S3BucketName": "cspm-demo-eor-20250804",
            "IncludeGlobalServiceEvents": true,
            "IsMultiRegionTrail": true,
            "HomeRegion": "us-east-1",
            "TrailARN": "arn:aws:cloudtrail:us-east-1:756716632322:trail/demoTrail",
            "LogFileValidationEnabled": false,
            "HasCustomEventSelectors": false,
            "HasInsightSelectors": false,
            "IsOrganizationTrail": false
        }
    ]
}
~ $
```

CloudTrail is successfully configured to track API activity across the account and linked to our demo S3 bucket. This is critical for audit logging and forensic visibility.

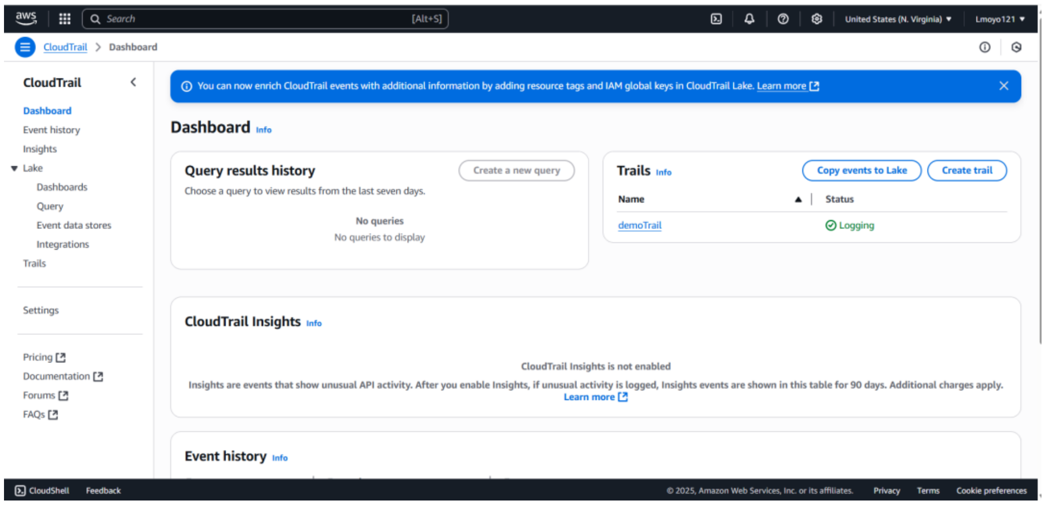
9. CloudTrail Logging Started and Verified



```
        }
    ]
}
~ $ # List the first few log files in your bucket
~ $ aws s3 ls s3://$BUCKET/AWSLogs/$ACCOUNT/CloudTrail/ --recursive | head -n 5
2025-08-05 04:32:48          0 AWSLogs/756716632322/CloudTrail/
~ $
```

Logging is activated and validated. This confirms that activity across AWS services will now be captured and stored securely in our designated S3 bucket.

10. CloudTrail Status Visible in Console



You can now enrich CloudTrail events with additional information by adding resource tags and IAM global keys in CloudTrail Lake. Learn more

Dashboard Info

Query results history
Choose a query to view results from the last seven days.
No queries
No queries to display

Trails Info

Name	Status
demoTrail	✓ Logging

CloudTrail Insights Info

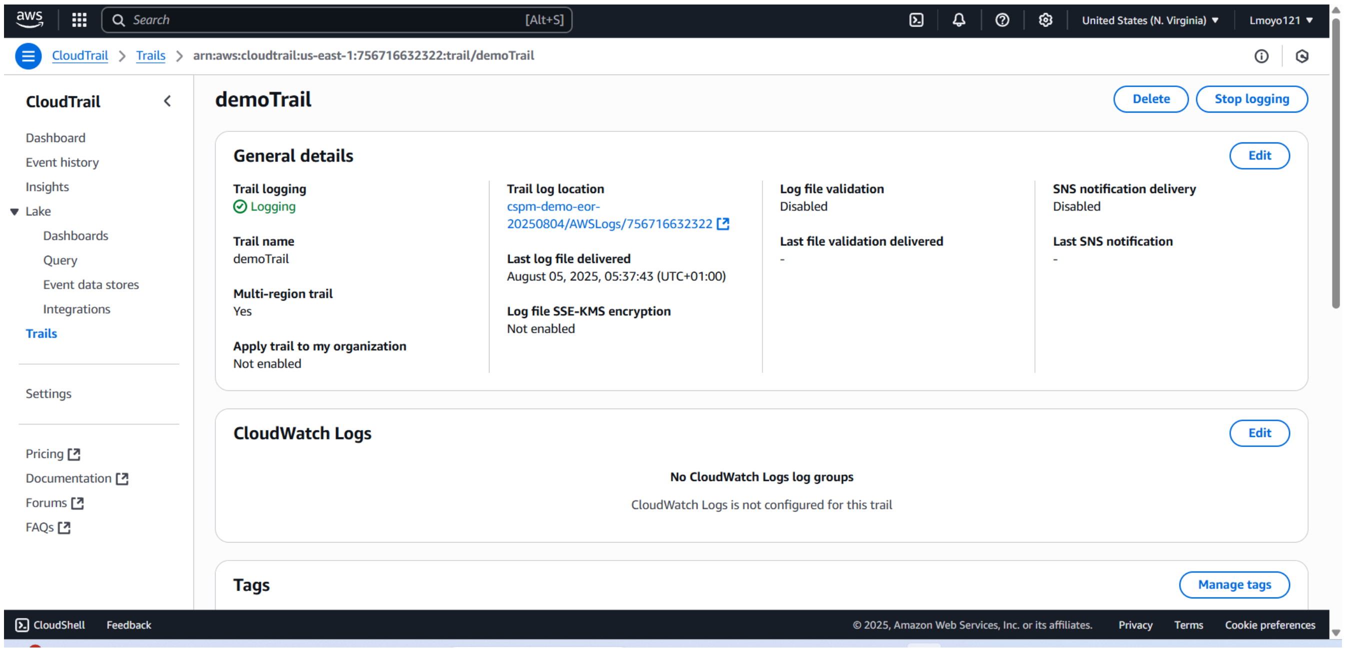
CloudTrail Insights is not enabled
Insights are events that show unusual API activity. After you enable Insights, if unusual activity is logged, Insights events are shown in this table for 90 days. Additional charges apply. Learn more

Event history Info

© 2025, Amazon Web Services, Inc. or its affiliates.

Here we see that the CloudTrail trail demoTrail is visible in the console and shows as Logging. This indicates that the service is actively capturing events.

11. S3 Bucket Destination and Logging Status Displayed



demoTrail

General details

Trail logging
Logging

Trail name
demoTrail

Multi-region trail
Yes

Apply trail to my organization
Not enabled

Trail log location
cspm-demo-eor-20250804/AWSLogs/756716632322

Last log file delivered
August 05, 2025, 05:37:43 (UTC+01:00)

Log file SSE-KMS encryption
Not enabled

Log file validation
Disabled

Last file validation delivered
-

SNS notification delivery
Disabled

Last SNS notification
-

CloudWatch Logs

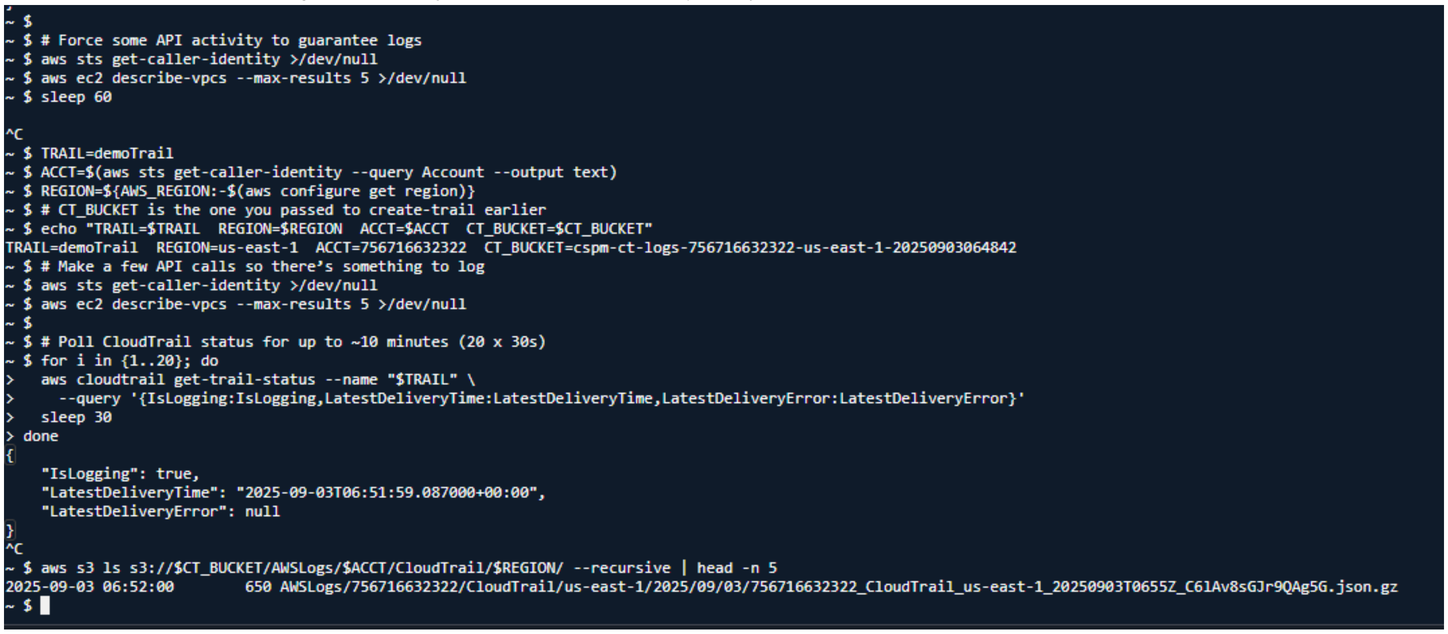
No CloudWatch Logs log groups

CloudWatch Logs is not configured for this trail

Tags

The trail detail's view shows the S3 bucket path where CloudTrail will deliver logs and that logging is currently enabled. However, the field Last log file delivered is empty, meaning no log delivery confirmation is shown here. This step verifies that CloudTrail is configured correctly and points to an S3 bucket, but the actual presence of logs still needs to be checked directly in the S3 console.

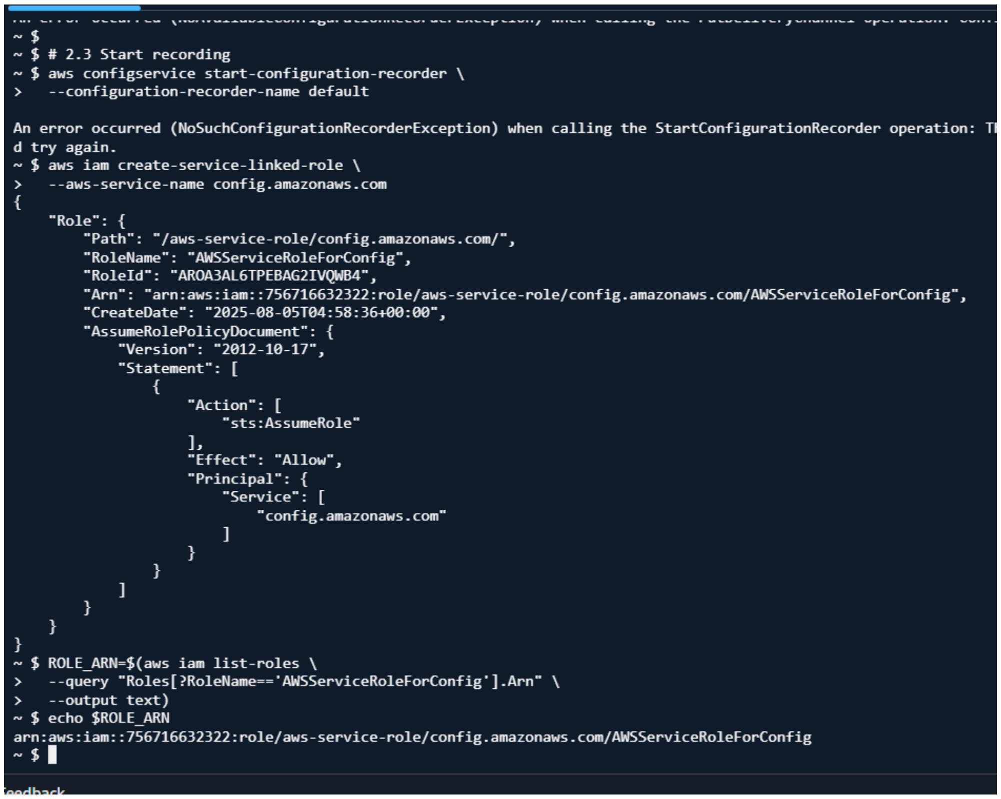
12. CloudTrail Delivery Proven (CLI: Status + S3 Objects)



```
~ $
~ $ # Force some API activity to guarantee logs
~ $ aws sts get-caller-identity >/dev/null
~ $ aws ec2 describe-vpcs --max-results 5 >/dev/null
~ $ sleep 60

^C
~ $ TRAIL=demoTrail
~ $ ACCT=$(aws sts get-caller-identity --query Account --output text)
~ $ REGION=${AWS_REGION:-$(aws configure get region)}
~ $ # CT_BUCKET is the one you passed to create-trail earlier
~ $ echo "TRAIL=$TRAIL  REGION=$REGION  ACCT=$ACCT  CT_BUCKET=$CT_BUCKET"
TRAIL=demoTrail  REGION=us-east-1  ACCT=756716632322  CT_BUCKET=cspm-ct-logs-756716632322-us-east-1-20250903064842
~ $ # Make a few API calls so there's something to log
~ $ aws sts get-caller-identity >/dev/null
~ $ aws ec2 describe-vpcs --max-results 5 >/dev/null
~ $
~ $ # Poll CloudTrail status for up to ~10 minutes (20 x 30s)
~ $ for i in {1..20}; do
>   aws cloudtrail get-trail-status --name "$TRAIL" \
>     --query '{IsLogging:IsLogging,LatestDeliveryTime:LatestDeliveryTime,LatestDeliveryError:LatestDeliveryError}'
>   sleep 30
> done
{
    "IsLogging": true,
    "LatestDeliveryTime": "2025-09-03T06:51:59.087000+00:00",
    "LatestDeliveryError": null
}
^C
~ $ aws s3 ls s3://$CT_BUCKET/AWSLogs/$ACCT/CloudTrail/$REGION/ --recursive | head -n 5
2025-09-03 06:52:00        650 AWSLogs/756716632322/CloudTrail/us-east-1/2025/09/03/756716632322_CloudTrail_us-east-1_20250903T0655Z_C61Av8sGJr9QAg5G.json.gz
~ $
```

Using CloudShell, we confirm that the trail demoTrail is actively logging and delivering to S3. The aws cloudtrail get-trail-status --name demoTrail output shows IsLogging: true with a populated LatestDeliveryTime, and aws s3 ls s3://<bucket>/AWSLogs/<account>/CloudTrail/<region>/ --recursive lists at least one .json.gz log object. Together, these prove CloudTrail is writing logs to the designated S3 bucket.

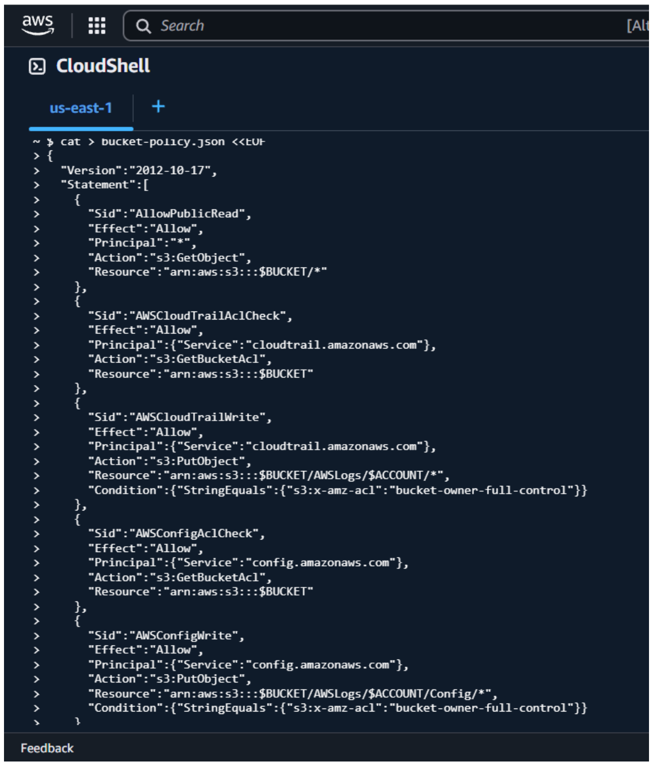
13. AWS Config Setup Failed: Missing Role



```
~ $
~ $ # 2.3 Start recording
~ $ aws configservice start-configuration-recorder \
>   --configuration-recorder-name default

An error occurred (NoSuchConfigurationRecorderException) when calling the StartConfigurationRecorder operation: T
d try again.
~ $ aws iam create-service-linked-role \
>   --aws-service-name config.amazonaws.com
{
    "Role": {
        "Path": "/aws-service-role/config.amazonaws.com/",
        "RoleName": "AWSServiceRoleForConfig",
        "RoleId": "AROA3AL6TPEBAG2IVQWB4",
        "Arn": "arn:aws:iam::756716632322:role/aws-service-role/config.amazonaws.com/AWSServiceRoleForConfig",
        "CreateDate": "2025-08-05T04:58:36+00:00",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Action": [
                        "sts:AssumeRole"
                    ],
                    "Effect": "Allow",
                    "Principal": {
                        "Service": [
                            "config.amazonaws.com"
                        ]
                    }
                }
            ]
        }
    }
}
~ $ ROLE_ARN=$(aws iam list-roles \
>   --query "Roles[?RoleName=='AWSServiceRoleForConfig'].Arn" \
>   --output text)
~ $ echo $ROLE_ARN
arn:aws:iam::756716632322:role/aws-service-role/config.amazonaws.com/AWSServiceRoleForConfig
~ $
```

An attempt to start AWS Config fails due to a missing service-linked role. This illustrates a typical AWS setup hurdle that must be remediated to ensure visibility into resource compliance.

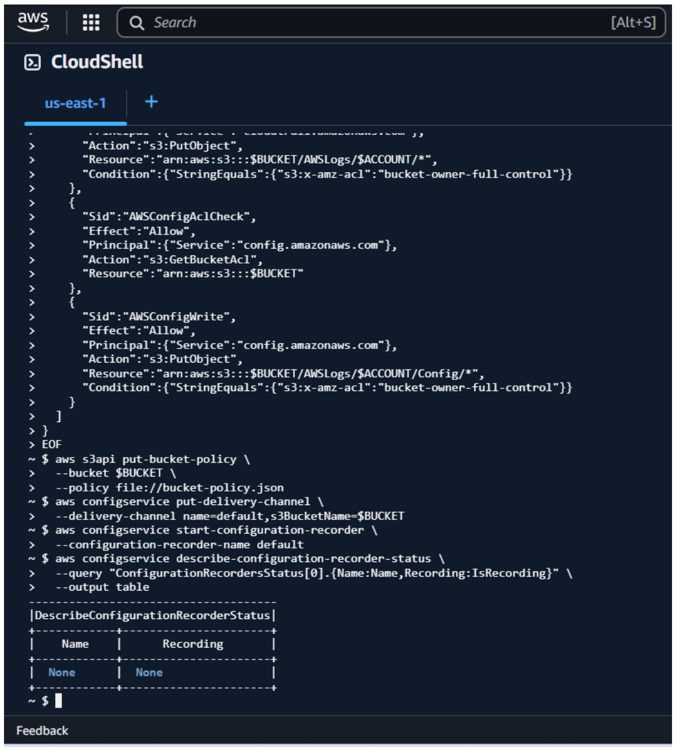
14. Creating and Verifying Config Service Role



```
~ $ cat > bucket-policy.json <<EOF
> {
>   "Version":"2012-10-17",
>   "Statement":[
>     {
>       "Sid":"AllowPublicRead",
>       "Effect":"Allow",
>       "Principal":"*",
>       "Action":"s3:GetObject",
>       "Resource":"arn:aws:s3:::$BUCKET/*"
>     },
>     {
>       "Sid":"AWSCloudTrailAclCheck",
>       "Effect":"Allow",
>       "Principal":{"Service":"cloudtrail.amazonaws.com"},
>       "Action":"s3:GetBucketAcl",
>       "Resource":"arn:aws:s3:::$BUCKET"
>     },
>     {
>       "Sid":"AWSCloudTrailWrite",
>       "Effect":"Allow",
>       "Principal":{"Service":"cloudtrail.amazonaws.com"},
>       "Action":"s3:PutObject",
>       "Resource":"arn:aws:s3:::$BUCKET/AWSLogs/$ACCOUNT/*",
>       "Condition":{"StringEquals":{"s3:x-amz-acl":"bucket-owner-full-control"}}
>     },
>     {
>       "Sid":"AWSConfigAclCheck",
>       "Effect":"Allow",
>       "Principal":{"Service":"config.amazonaws.com"},
>       "Action":"s3:GetBucketAcl",
>       "Resource":"arn:aws:s3:::$BUCKET"
>     },
>     {
>       "Sid":"AWSConfigWrite",
>       "Effect":"Allow",
>       "Principal":{"Service":"config.amazonaws.com"},
>       "Action":"s3:PutObject",
>       "Resource":"arn:aws:s3:::$BUCKET/AWSLogs/$ACCOUNT/Config/*",
>       "Condition":{"StringEquals":{"s3:x-amz-acl":"bucket-owner-full-control"}}
>     }
```

We create the necessary service-linked role for AWS Config and verify its ARN, preparing the account to enable configuration recording and compliance tracking.

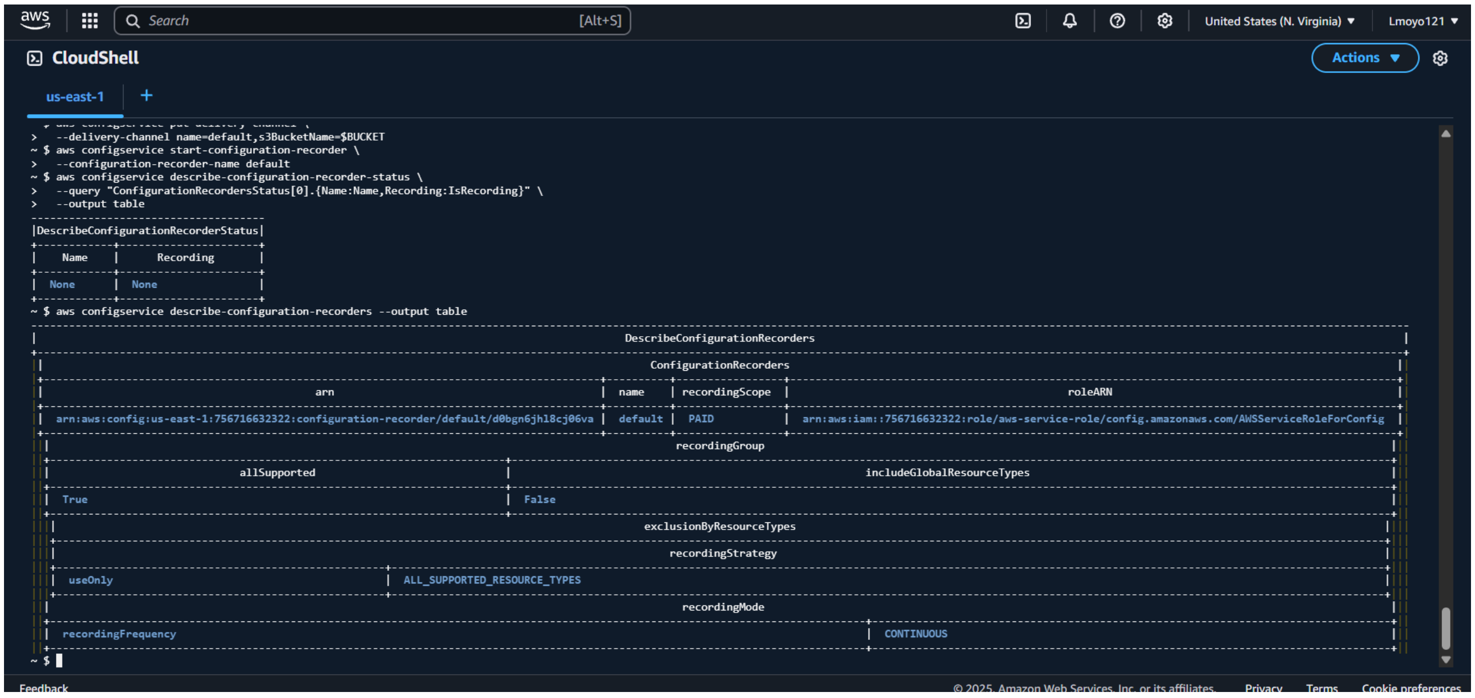
15. Bucket Policy Attached - Allows AWS Config Delivery



```
>       "Action":"s3:PutObject",
>       "Resource":"arn:aws:s3:::$BUCKET/AWSLogs/$ACCOUNT/*",
>       "Condition":{"StringEquals":{"s3:x-amz-acl":"bucket-owner-full-control"}}
>     },
>     {
>       "Sid":"AWSConfigAclCheck",
>       "Effect":"Allow",
>       "Principal":{"Service":"config.amazonaws.com"},
>       "Action":"s3:GetBucketAcl",
>       "Resource":"arn:aws:s3:::$BUCKET"
>     },
>     {
>       "Sid":"AWSConfigWrite",
>       "Effect":"Allow",
>       "Principal":{"Service":"config.amazonaws.com"},
>       "Action":"s3:PutObject",
>       "Resource":"arn:aws:s3:::$BUCKET/AWSLogs/$ACCOUNT/Config/*",
>       "Condition":{"StringEquals":{"s3:x-amz-acl":"bucket-owner-full-control"}}
>     }
>   ]
> }
> EOF
~ $ aws s3api put-bucket-policy \
>   --bucket $BUCKET \
>   --policy file://bucket-policy.json
~ $ aws configservice put-delivery-channel \
>   --delivery-channel name=default,s3BucketName=$BUCKET
~ $ aws configservice start-configuration-recorder \
>   --configuration-recorder-name default
~ $ aws configservice describe-configuration-recorder-status \
>   --query "ConfigurationRecordersStatus[0].{Name:Name,Recording:IsRecording}" \
>   --output table
-------------------------------------
|DescribeConfigurationRecorderStatus|
+------------+----------------------+
|    Name    |      Recording       |
+------------+----------------------+
|  None      |  None                |
+------------+----------------------+
~ $
```

Purpose: allow AWS Config to write snapshots and configuration items to S3. What changed: policy grants config.amazonaws.com s3:GetBucketAcl and s3:PutObject with bucket-owner-full-control on the correct key prefix. Outcome: S3 delivery prerequisites satisfied.

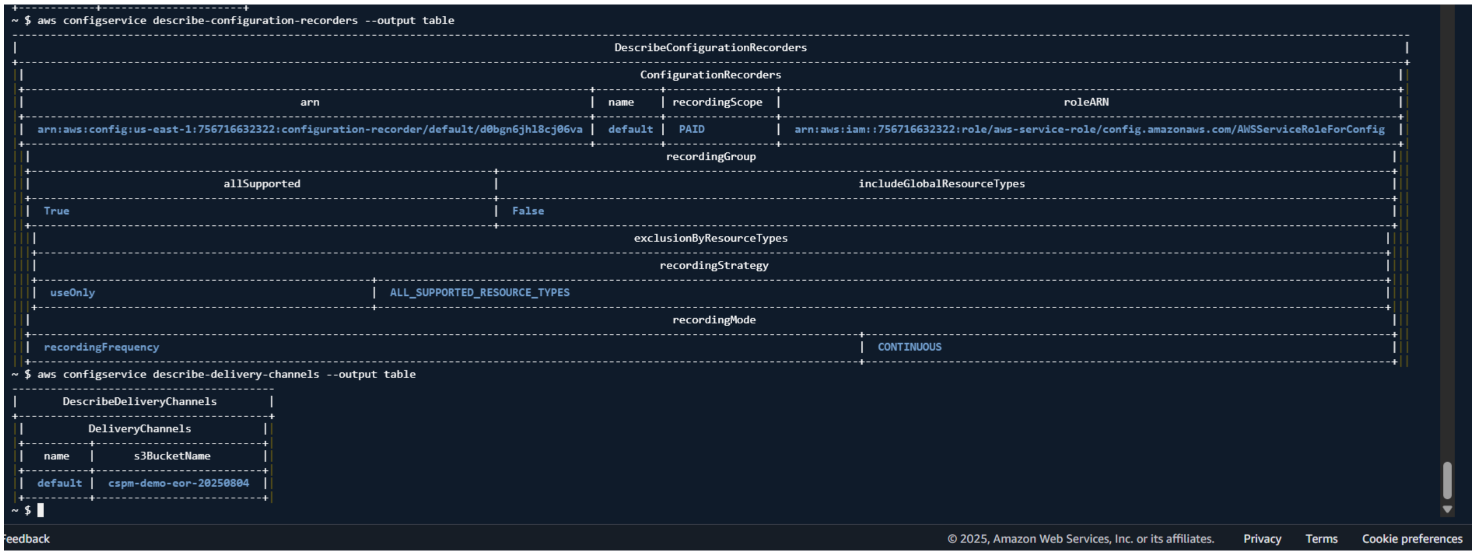
16. Create Delivery Channel & Configuration Recorder



```
>   --delivery-channel name=default,s3BucketName=$BUCKET
~ $ aws configservice start-configuration-recorder \
>   --configuration-recorder-name default
~ $ aws configservice describe-configuration-recorder-status \
>   --query "ConfigurationRecordersStatus[0].{Name:Name,Recording:IsRecording}" \
>   --output table
-------------------------------------
|DescribeConfigurationRecorderStatus|
+------------+----------------------+
|    Name    |      Recording       |
+------------+----------------------+
|  None      |  None                |
+------------+----------------------+
~ $ aws configservice describe-configuration-recorders --output table
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
|                                                                      DescribeConfigurationRecorders                                                                                      |
+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
||                                                                      ConfigurationRecorders                                                                                           ||
|+-----------------------------------------------------------------------------------+-----------+-----------------+--------------------------------------------------------------------------+|
||                                         arn                                       |   name    | recordingScope  |                                 roleARN                                  ||
|+-----------------------------------------------------------------------------------+-----------+-----------------+--------------------------------------------------------------------------+|
||  arn:aws:config:us-east-1:756716632322:configuration-recorder/default/d0bgn6jhl8cj06va |  default  |  PAID           |  arn:aws:iam::756716632322:role/aws-service-role/config.amazonaws.com/AWSServiceRoleForConfig  ||
|+-----------------------------------------------------------------------------------+-----------+-----------------+--------------------------------------------------------------------------+|
|||                                                                      recordingGroup                                                                                                  |||
||+----------------------------------------------------------------+-------------------------------------------------------------------------------------------------------------------+||
|||                          allSupported                          |                                      includeGlobalResourceTypes                                                   |||
||+----------------------------------------------------------------+-------------------------------------------------------------------------------------------------------------------+||
|||  True                                                          |  False                                                                                                            |||
||+----------------------------------------------------------------+-------------------------------------------------------------------------------------------------------------------+||
||||                                                                      exclusionByResourceTypes                                                                                      ||||
|||+----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+|||
||||                                                                      recordingStrategy                                                                                             ||||
|||+-----------------------------------------------------+----------------------------------------------------------------------------------------------------------------------------+|||
||||  useOnly                                            |  ALL_SUPPORTED_RESOURCE_TYPES                                                                                              ||||
|||+-----------------------------------------------------+----------------------------------------------------------------------------------------------------------------------------+|||
|||                                                                      recordingMode                                                                                                   |||
||+-------------------------------------------------------------------------------------------------------------+----------------------------------------------------------------------------+||
|||  recordingFrequency                                                                                         |  CONTINUOUS                                                                |||
||+-------------------------------------------------------------------------------------------------------------+----------------------------------------------------------------------------+||
~ $
```

Action: created the delivery channel targeting the S3 bucket and a configuration recorder for all supported resource types. Mode: continuous recording enabled. Outcome: AWS Config is ready to start recording changes.

17. Configuration Recorder Successfully Created

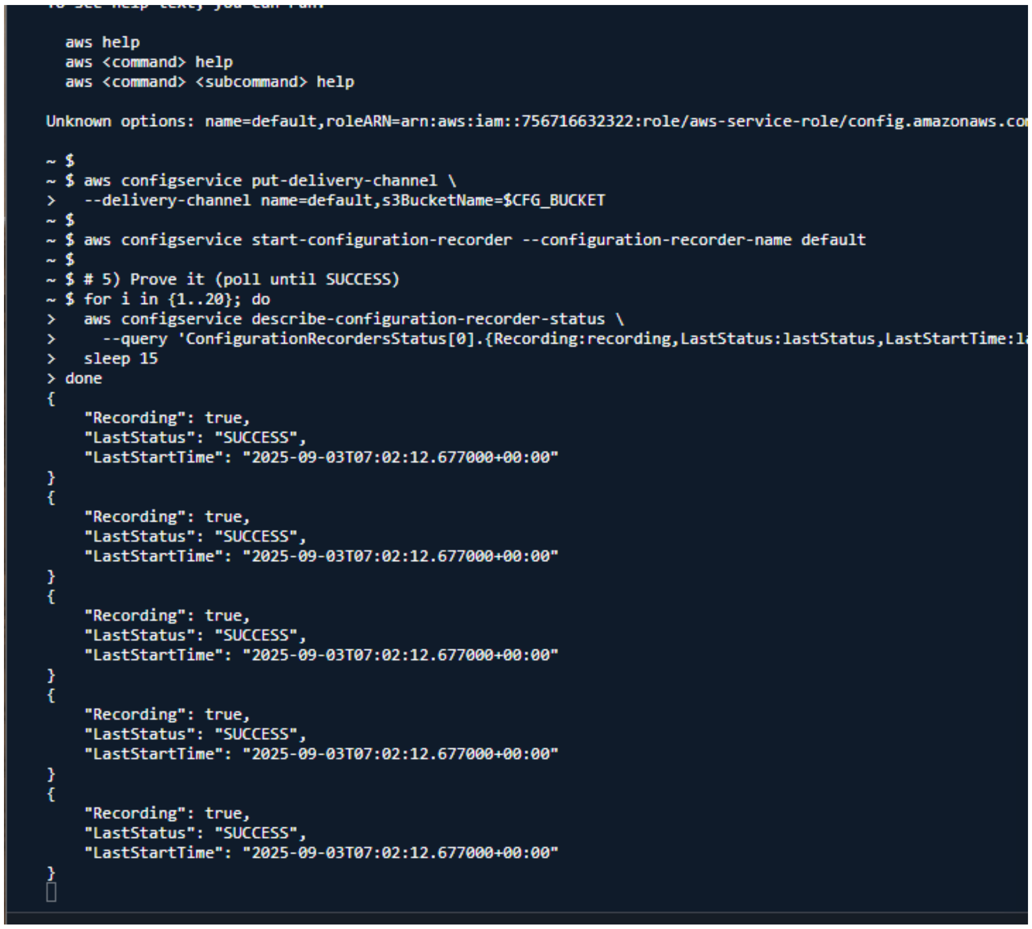


```
~ $ aws configservice describe-configuration-recorders --output table
-----------------------------------------------------------------------------------------------------------------------------------------------------
|                                                    DescribeConfigurationRecorders                                                                 |
+---------------------------------------------------------------------------------------------------------------------------------------------------+
||                                                      ConfigurationRecorders                                                                     ||
|+-----------------------------------------------------------------------------------+----------+-----------------+--------------------------------+|
||                                       arn                                         |  name    | recordingScope  |            roleARN             ||
|+-----------------------------------------------------------------------------------+----------+-----------------+--------------------------------+|
||  arn:aws:config:us-east-1:756716632322:configuration-recorder/default/d0bgn6jhl8cj06va |  default |  PAID           |  arn:aws:iam::756716632322:role/aws-service-role/config.amazonaws.com/AWSServiceRoleForConfig  ||
|+-----------------------------------------------------------------------------------+----------+-----------------+--------------------------------+|
|||                                                       recordingGroup                                                                          |||
||+-----------------------------------------------------------+-----------------------------------------------------------------------------------+||
|||                      allSupported                         |                         includeGlobalResourceTypes                                |||
||+-----------------------------------------------------------+-----------------------------------------------------------------------------------+||
|||  True                                                     |  False                                                                            |||
||+-----------------------------------------------------------+-----------------------------------------------------------------------------------+||
||||                                                   exclusionByResourceTypes                                                                   ||||
|||+-------------------------------------------------------------------------------------------------------------------------------------------+|||
||||                                                      recordingStrategy                                                                       ||||
|||+----------------------------------------------+--------------------------------------------------------------------------------------------+|||
||||  useOnly                                     |  ALL_SUPPORTED_RESOURCE_TYPES                                                                  ||||
|||+----------------------------------------------+--------------------------------------------------------------------------------------------+|||
|||                                                       recordingMode                                                                           |||
||+-------------------------------------------------------------------------------------------------------+---------------------------------------+||
|||  recordingFrequency                                                                                   |  CONTINUOUS                           |||
||+-------------------------------------------------------------------------------------------------------+---------------------------------------+||
~ $ aws configservice describe-delivery-channels --output table
-----------------------------------
|     DescribeDeliveryChannels    |
+---------------------------------+
||        DeliveryChannels       ||
|+---------+---------------------+|
||  name   |    s3BucketName     ||
|+---------+---------------------+|
||  default|  cspm-demo-eor-20250804 ||
|+---------+---------------------+|
~ $
```

Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Evidence: CLI output confirms successful creation of the delivery channel and configuration recorder.Next step: start the recorder and verify status.

18. Configuration Recorder Running (S3 Delivery Channel Verified)

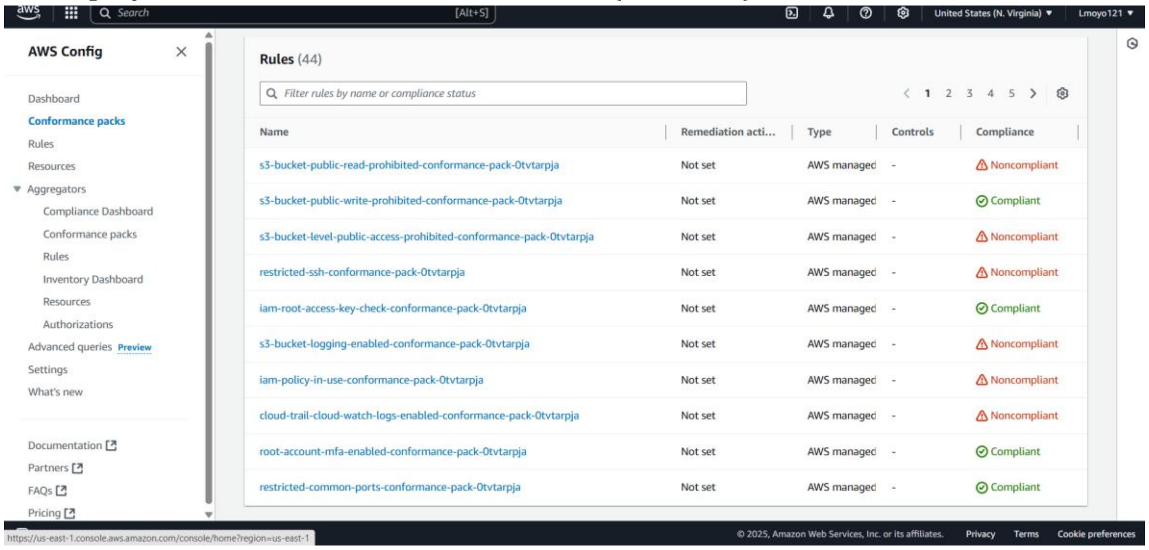


```
To see help text, you can run:

  aws help
  aws <command> help
  aws <command> <subcommand> help

Unknown options: name=default,roleARN=arn:aws:iam::756716632322:role/aws-service-role/config.amazonaws.co

~ $
~ $ aws configservice put-delivery-channel \
>   --delivery-channel name=default,s3BucketName=$CFG_BUCKET
~ $
~ $ aws configservice start-configuration-recorder --configuration-recorder-name default
~ $
~ $ # 5) Prove it (poll until SUCCESS)
~ $ for i in {1..20}; do
>   aws configservice describe-configuration-recorder-status \
>     --query 'ConfigurationRecordersStatus[0].{Recording:recording,LastStatus:lastStatus,LastStartTime:l
>   sleep 15
> done
{
    "Recording": true,
    "LastStatus": "SUCCESS",
    "LastStartTime": "2025-09-03T07:02:12.677000+00:00"
}
{
    "Recording": true,
    "LastStatus": "SUCCESS",
    "LastStartTime": "2025-09-03T07:02:12.677000+00:00"
}
{
    "Recording": true,
    "LastStatus": "SUCCESS",
    "LastStartTime": "2025-09-03T07:02:12.677000+00:00"
}
{
    "Recording": true,
    "LastStatus": "SUCCESS",
    "LastStartTime": "2025-09-03T07:02:12.677000+00:00"
}
{
    "Recording": true,
    "LastStatus": "SUCCESS",
    "LastStartTime": "2025-09-03T07:02:12.677000+00:00"
}
```

CloudShell output shows Recording: true and LastStatus: "SUCCESS" with a valid LastStartTime. That's the definitive signal that AWS Config is actively recording and can deliver to the S3 bucket you configured.

Compliance Benchmarking

19. Deploy CIS AWS Foundations Benchmark (v1.2.0 L1)

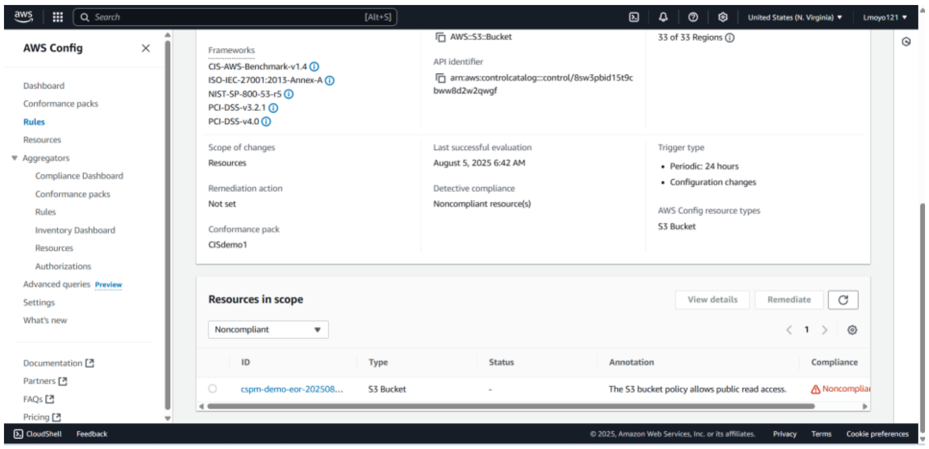


Rules (44)

Name	Remediation acti...	Type	Controls	Compliance
s3-bucket-public-read-prohibited-conformance-pack-0tvtarpja	Not set	AWS managed	-	Noncompliant
s3-bucket-public-write-prohibited-conformance-pack-0tvtarpja	Not set	AWS managed	-	Compliant
s3-bucket-level-public-access-prohibited-conformance-pack-0tvtarpja	Not set	AWS managed	-	Noncompliant
restricted-ssh-conformance-pack-0tvtarpja	Not set	AWS managed	-	Noncompliant
iam-root-access-key-check-conformance-pack-0tvtarpja	Not set	AWS managed	-	Compliant
s3-bucket-logging-enabled-conformance-pack-0tvtarpja	Not set	AWS managed	-	Noncompliant
iam-policy-in-use-conformance-pack-0tvtarpja	Not set	AWS managed	-	Noncompliant
cloud-trail-cloud-watch-logs-enabled-conformance-pack-0tvtarpja	Not set	AWS managed	-	Noncompliant
root-account-mfa-enabled-conformance-pack-0tvtarpja	Not set	AWS managed	-	Compliant
restricted-common-ports-conformance-pack-0tvtarpja	Not set	AWS managed	-	Compliant

We deploy the CIS AWS Foundations Benchmark v1.2.0 Level 1 via an AWS Config Conformance Pack. This seeds the environment with managed rules that continuously evaluate core controls (e.g., CloudTrail enabled, root account MFA, S3 public access). This is the kickoff for automated "compliance-as-code."

20. CIS Conformance Pack Deployment in Progress



Frameworks: CIS-AWS-Benchmark-v1.4, ISO-IEC-27001:2013-Annex-A, NIST-SP-800-53-r5, PCI-DSS-v3.2.1, PCI-DSS-v4.0

AWS::S3::Bucket

API identifier: arn:aws:controlcatalog:::control/8sw3pbid15t9cbww8d2w2qwgf

33 of 33 Regions

Scope of changes: Resources

Remediation action: Not set

Conformance pack: CISdemo1

Last successful evaluation: August 5, 2025 6:42 AM

Detective compliance: Noncompliant resource(s)

Trigger type:
- Periodic: 24 hours
- Configuration changes

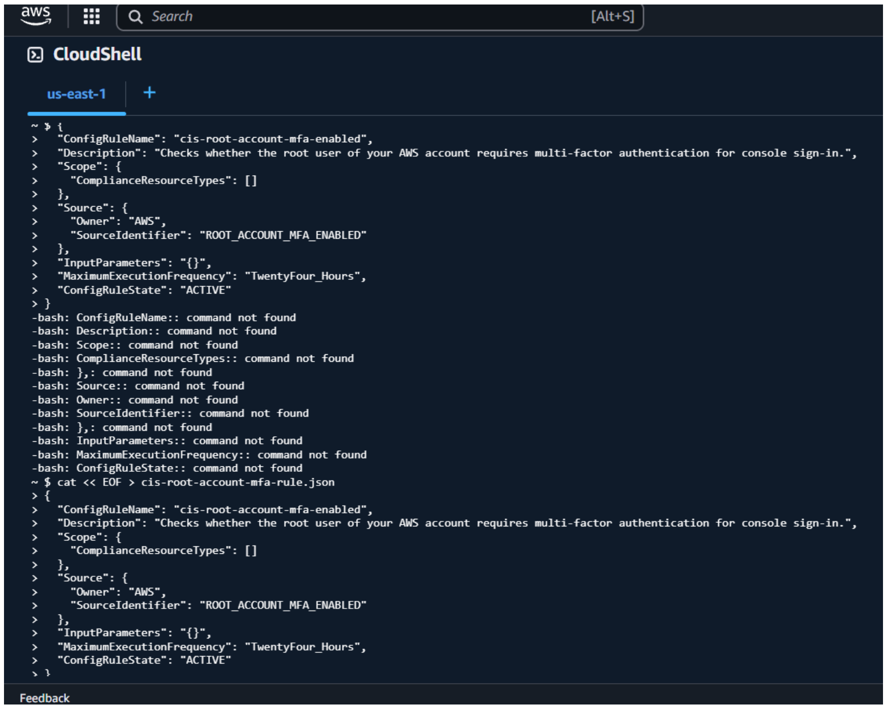
AWS Config resource types: S3 Bucket

Resources in scope — Noncompliant

ID	Type	Status	Annotation	Compliance
cspm-demo-eor-202508...	S3 Bucket	-	The S3 bucket policy allows public read access.	Noncomplia

The deployment of the CIS AWS Foundations Benchmark Conformance Pack has been initiated. This step kicks off the evaluation of our AWS environment against standardised security controls.

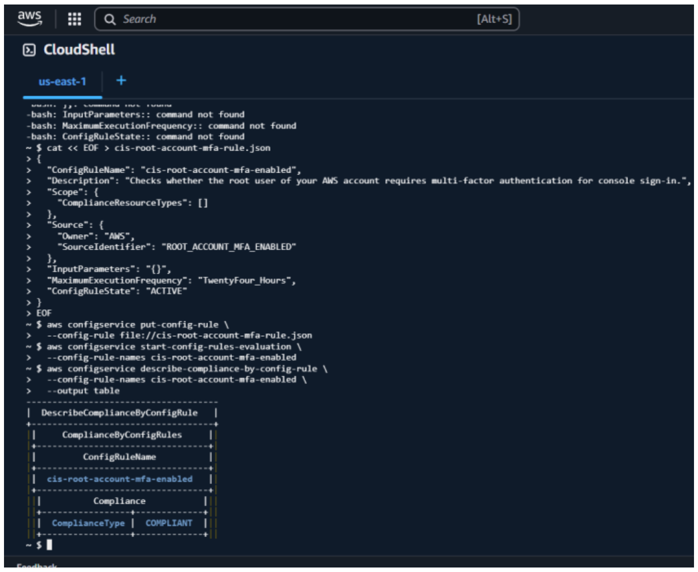
21. CIS Config Rule Definition for MFA



```
~ $ {
>   "ConfigRuleName": "cis-root-account-mfa-enabled",
>   "Description": "Checks whether the root user of your AWS account requires multi-factor authentication for console sign-in.",
>   "Scope": {
>     "ComplianceResourceTypes": []
>   },
>   "Source": {
>     "Owner": "AWS",
>     "SourceIdentifier": "ROOT_ACCOUNT_MFA_ENABLED"
>   },
>   "InputParameters": "{}",
>   "MaximumExecutionFrequency": "TwentyFour_Hours",
>   "ConfigRuleState": "ACTIVE"
> }
-bash: ConfigRuleName:: command not found
-bash: Description:: command not found
-bash: Scope:: command not found
-bash: ComplianceResourceTypes:: command not found
-bash: },: command not found
-bash: Source:: command not found
-bash: Owner:: command not found
-bash: SourceIdentifier:: command not found
-bash: },: command not found
-bash: InputParameters:: command not found
-bash: MaximumExecutionFrequency:: command not found
-bash: ConfigRuleState:: command not found
~ $ cat << EOF > cis-root-account-mfa-rule.json
> {
>   "ConfigRuleName": "cis-root-account-mfa-enabled",
>   "Description": "Checks whether the root user of your AWS account requires multi-factor authentication for console sign-in.",
>   "Scope": {
>     "ComplianceResourceTypes": []
>   },
>   "Source": {
>     "Owner": "AWS",
>     "SourceIdentifier": "ROOT_ACCOUNT_MFA_ENABLED"
>   },
>   "InputParameters": "{}",
>   "MaximumExecutionFrequency": "TwentyFour_Hours",
>   "ConfigRuleState": "ACTIVE"
> }
```

Here we review the AWS Config rule definition for enforcing root account MFA (cis-root-account-mfa-enabled). The rule metadata specifies its purpose, source identifier, and execution frequency. By deploying this rule, Config can continuously check whether the root account has MFA enabled and flag noncompliance for remediation.

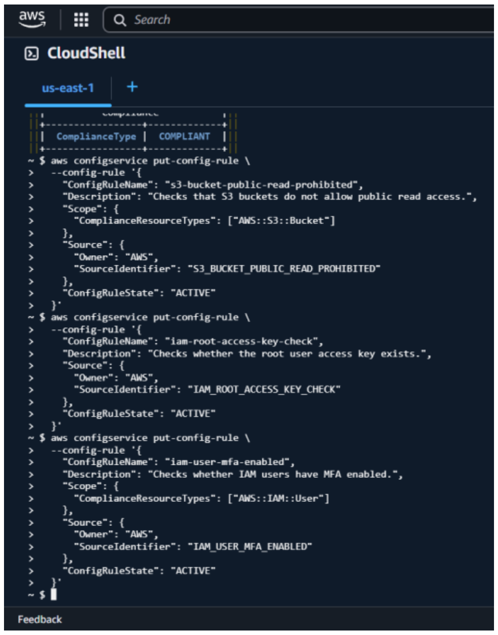
22. AWS Config Compliance Evaluation Results



```
-bash: InputParameters:: command not found
-bash: MaximumExecutionFrequency:: command not found
-bash: ConfigRuleState:: command not found
~ $ cat << EOF > cis-root-account-mfa-rule.json
> {
>   "ConfigRuleName": "cis-root-account-mfa-enabled",
>   "Description": "Checks whether the root user of your AWS account requires multi-factor authentication for console sign-in.",
>   "Scope": {
>     "ComplianceResourceTypes": []
>   },
>   "Source": {
>     "Owner": "AWS",
>     "SourceIdentifier": "ROOT_ACCOUNT_MFA_ENABLED"
>   },
>   "InputParameters": "{}",
>   "MaximumExecutionFrequency": "TwentyFour_Hours",
>   "ConfigRuleState": "ACTIVE"
> }
> EOF
~ $ aws configservice put-config-rule \
>   --config-rule file://cis-root-account-mfa-rule.json
~ $ aws configservice start-config-rules-evaluation \
>   --config-rule-names cis-root-account-mfa-enabled
~ $ aws configservice describe-compliance-by-config-rule \
>   --config-rule-names cis-root-account-mfa-enabled \
>   --output table
-----------------------------------
|  DescribeComplianceByConfigRule  |
+---------------------------------+
||     ComplianceByConfigRules    ||
|+-------------------------------+|
||         ConfigRuleName         ||
|+-------------------------------+|
||  cis-root-account-mfa-enabled  ||
|+-------------------------------+|
|||          Compliance          |||
||+---------------+-------------+||
|||  ComplianceType |  COMPLIANT  |||
||+---------------+-------------+||
~ $
```

Using the AWS CLI, we evaluate the CIS root account MFA rule in AWS Config and confirm the compliance status. The rule returns as COMPLIANT, showing that MFA is correctly enforced on the root account. This step demonstrates how Config can provide rule-by-rule compliance results for audit and remediation purposes.

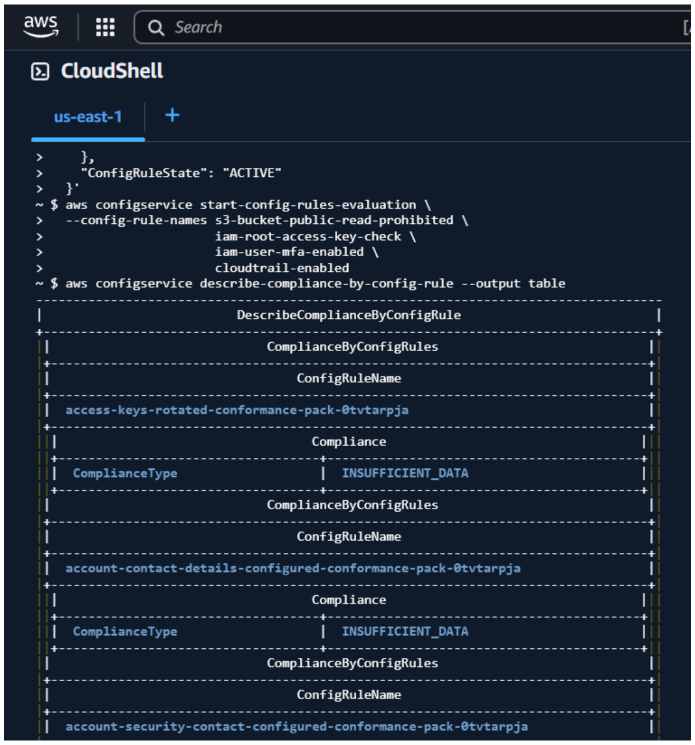
23. Additional AWS Config Rules Deployed



```
||+-----------------+------------+||
|||  ComplianceType |  COMPLIANT |||
||+-----------------+------------+||
~ $ aws configservice put-config-rule \
>   --config-rule '{
>     "ConfigRuleName": "s3-bucket-public-read-prohibited",
>     "Description": "Checks that S3 buckets do not allow public read access.",
>     "Scope": {
>       "ComplianceResourceTypes": ["AWS::S3::Bucket"]
>     },
>     "Source": {
>       "Owner": "AWS",
>       "SourceIdentifier": "S3_BUCKET_PUBLIC_READ_PROHIBITED"
>     },
>     "ConfigRuleState": "ACTIVE"
>   }'
~ $ aws configservice put-config-rule \
>   --config-rule '{
>     "ConfigRuleName": "iam-root-access-key-check",
>     "Description": "Checks whether the root user access key exists.",
>     "Source": {
>       "Owner": "AWS",
>       "SourceIdentifier": "IAM_ROOT_ACCESS_KEY_CHECK"
>     },
>     "ConfigRuleState": "ACTIVE"
>   }'
~ $ aws configservice put-config-rule \
>   --config-rule '{
>     "ConfigRuleName": "iam-user-mfa-enabled",
>     "Description": "Checks whether IAM users have MFA enabled.",
>     "Scope": {
>       "ComplianceResourceTypes": ["AWS::IAM::User"]
>     },
>     "Source": {
>       "Owner": "AWS",
>       "SourceIdentifier": "IAM_USER_MFA_ENABLED"
>     },
>     "ConfigRuleState": "ACTIVE"
>   }'
~ $
```

Here we deploy additional AWS Config rules to enforce critical security baselines. These include checks to prevent public S3 access, prohibit root account access keys, and ensure MFA is enabled for IAM users. By activating these rules, AWS Config continuously evaluates resources against best practices, surfacing misconfigurations for remediation.

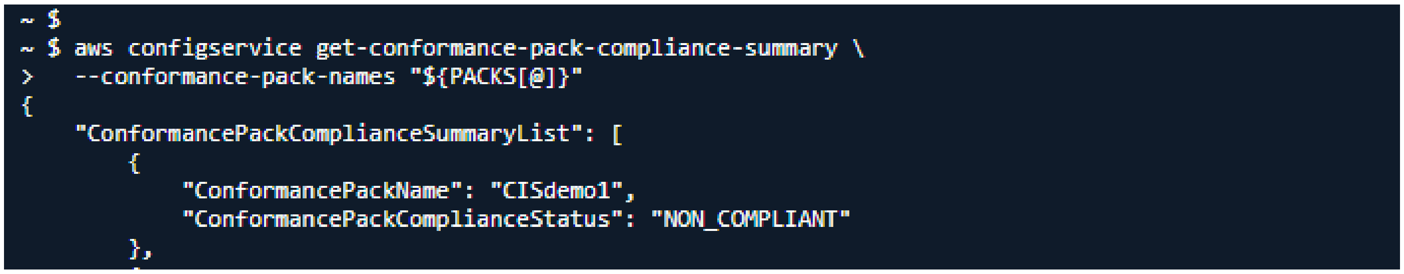
24. AWS Config Compliance Results Across Multiple Rules



```
>     },
>     "ConfigRuleState": "ACTIVE"
>   }'
~ $ aws configservice start-config-rules-evaluation \
>   --config-rule-names s3-bucket-public-read-prohibited \
>                       iam-root-access-key-check \
>                       iam-user-mfa-enabled \
>                       cloudtrail-enabled
~ $ aws configservice describe-compliance-by-config-rule --output table
--------------------------------------------------------------------------
|                      DescribeComplianceByConfigRule                    |
+------------------------------------------------------------------------+
||                        ComplianceByConfigRules                       ||
|+----------------------------------------------------------------------+|
||                            ConfigRuleName                            ||
|+----------------------------------------------------------------------+|
||  access-keys-rotated-conformance-pack-0tvtarpja                      ||
|+----------------------------------------------------------------------+|
|||                            Compliance                              |||
||+-------------------------------------+------------------------------+||
|||  ComplianceType                     |  INSUFFICIENT_DATA           |||
||+-------------------------------------+------------------------------+||
||                        ComplianceByConfigRules                       ||
|+----------------------------------------------------------------------+|
||                            ConfigRuleName                            ||
|+----------------------------------------------------------------------+|
||  account-contact-details-configured-conformance-pack-0tvtarpja       ||
|+----------------------------------------------------------------------+|
|||                            Compliance                              |||
||+-------------------------------------+------------------------------+||
|||  ComplianceType                     |  INSUFFICIENT_DATA           |||
||+-------------------------------------+------------------------------+||
||                        ComplianceByConfigRules                       ||
|+----------------------------------------------------------------------+|
||                            ConfigRuleName                            ||
|+----------------------------------------------------------------------+|
||  account-security-contact-configured-conformance-pack-0tvtarpja     ||
```

Here we initiate an on-demand compliance evaluation for multiple AWS Config rules, including checks for S3 public access, root account access keys, MFA enforcement, and CloudTrail enablement. The results show compliance states (e.g., INSUFFICIENT_DATA, COMPLIANT, or NON_COMPLIANT) for each rule. This provides direct visibility into which controls are properly enforced and where further investigation is needed.

25. Conformance Pack Compliance Summary (Pack-Level Status)

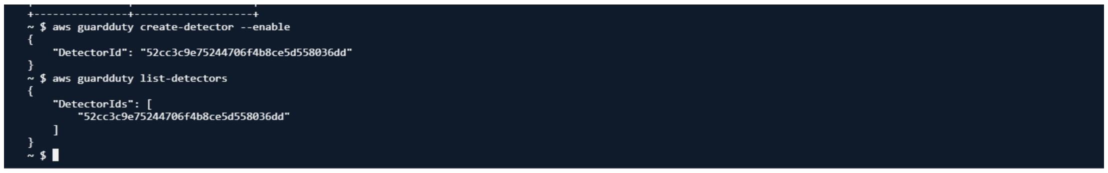


```
~ $
~ $ aws configservice get-conformance-pack-compliance-summary \
>   --conformance-pack-names "${PACKS[@]}"
{
    "ConformancePackComplianceSummaryList": [
        {
            "ConformancePackName": "CISdemo1",
            "ConformancePackComplianceStatus": "NON_COMPLIANT"
        },
```

We query AWS Config for the pack-level roll-up. Here, **CISdemo1** is **NON_COMPLIANT** (expected while individual rules are still failing or evaluating). Use the per-rule view to see which rules/resources are driving the status, then remediate and re-evaluate.

Consolidated Reporting

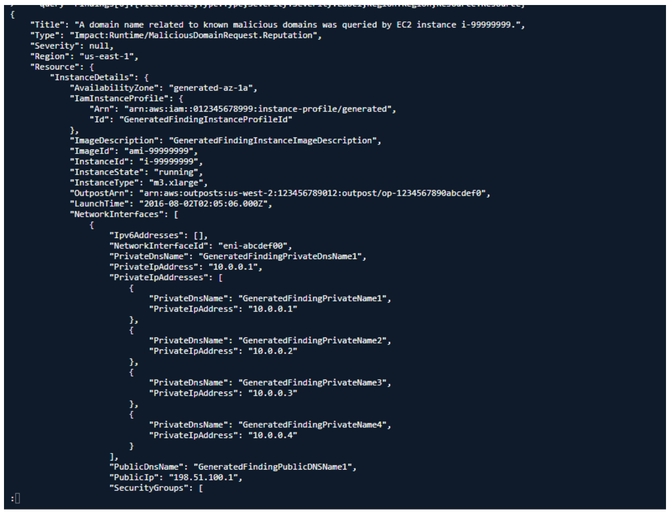
26. GuardDuty Detector Successfully Enabled



```
+--------------+------------------+
~ $ aws guardduty create-detector --enable
{
    "DetectorId": "52cc3c9e75244706f4b8ce5d558036dd"
}
~ $ aws guardduty list-detectors
{
    "DetectorIds": [
        "52cc3c9e75244706f4b8ce5d558036dd"
    ]
}
~ $ 
```

Using the AWS CLI, we enable Amazon GuardDuty by creating a new detector and confirming its presence with list-detectors. This validates that GuardDuty is now actively monitoring threats across the account. At this stage, no specific findings (such as MFA control failures) are being displayed - this step only confirms that the service has been successfully initialised.

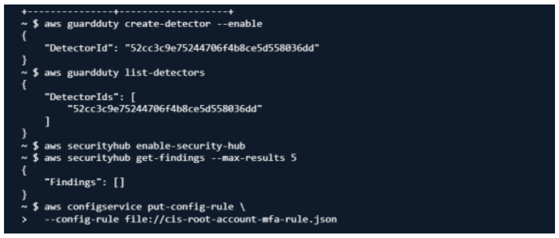
27. GuardDuty Finding Example (CLI)



```
{
    "Title": "A domain name related to known malicious domains was queried by EC2 instance i-99999999.",
    "Type": "Impact:Runtime/MaliciousDomainRequest.Reputation",
    "Severity": null,
    "Region": "us-east-1",
    "Resource": {
        "InstanceDetails": {
            "AvailabilityZone": "generated-az-1a",
            "IamInstanceProfile": {
                "Arn": "arn:aws:iam::012345678999:instance-profile/generated",
                "Id": "GeneratedFindingInstanceProfileId"
            },
            "ImageDescription": "GeneratedFindingInstanceImageDescription",
            "ImageId": "ami-99999999",
            "InstanceId": "i-99999999",
            "InstanceState": "running",
            "InstanceType": "m3.xlarge",
            "OutpostArn": "arn:aws:outposts:us-west-2:123456789012:outpost/op-1234567890abcdef0",
            "LaunchTime": "2016-08-02T02:05:06.000Z",
            "NetworkInterfaces": [
                {
                    "Ipv6Addresses": [],
                    "NetworkInterfaceId": "eni-abcdef00",
                    "PrivateDnsName": "GeneratedFindingPrivateDnsName1",
                    "PrivateIpAddress": "10.0.0.1",
                    "PrivateIpAddresses": [
                        {
                            "PrivateDnsName": "GeneratedFindingPrivateName1",
                            "PrivateIpAddress": "10.0.0.1"
                        },
                        {
                            "PrivateDnsName": "GeneratedFindingPrivateName2",
                            "PrivateIpAddress": "10.0.0.2"
                        },
                        {
                            "PrivateDnsName": "GeneratedFindingPrivateName3",
                            "PrivateIpAddress": "10.0.0.3"
                        },
                        {
                            "PrivateDnsName": "GeneratedFindingPrivateName4",
                            "PrivateIpAddress": "10.0.0.4"
                        }
                    ],
                    "PublicDnsName": "GeneratedFindingPublicDNSName1",
                    "PublicIp": "198.51.100.1",
                    "SecurityGroups": [
:
```

In CloudShell we generate GuardDuty sample findings and display one with key metadata (Title, Type, numeric Severity, Region, InstanceId, PublicIp). This confirms GuardDuty is enabled and producing actionable findings for triage.

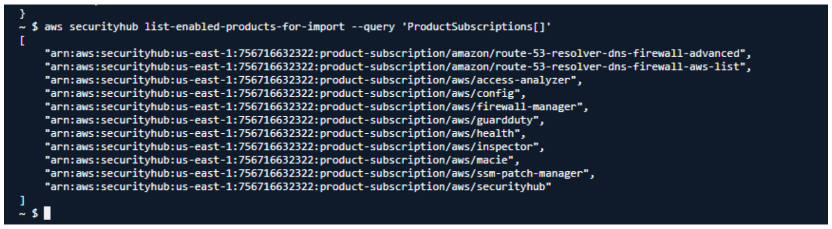
28. Security Hub Enabled (GuardDuty Integrated)



```
+---------------+------------------+
~ $ aws guardduty create-detector --enable
{
    "DetectorId": "52cc3c9e75244706f4b8ce5d558036dd"
}
~ $ aws guardduty list-detectors
{
    "DetectorIds": [
        "52cc3c9e75244706f4b8ce5d558036dd"
    ]
}
~ $ aws securityhub enable-security-hub
~ $ aws securityhub get-findings --max-results 5
{
    "Findings": []
}
~ $ aws configservice put-config-rule \
>   --config-rule file://cis-root-account-mfa-rule.json
```

We enable AWS Security Hub and confirm integration with GuardDuty so threat findings flow centrally. This establishes Security Hub as the single pane of glass for posture and detections. (Standards are enabled in the next step; this slide proves the hub is on and ingesting.)

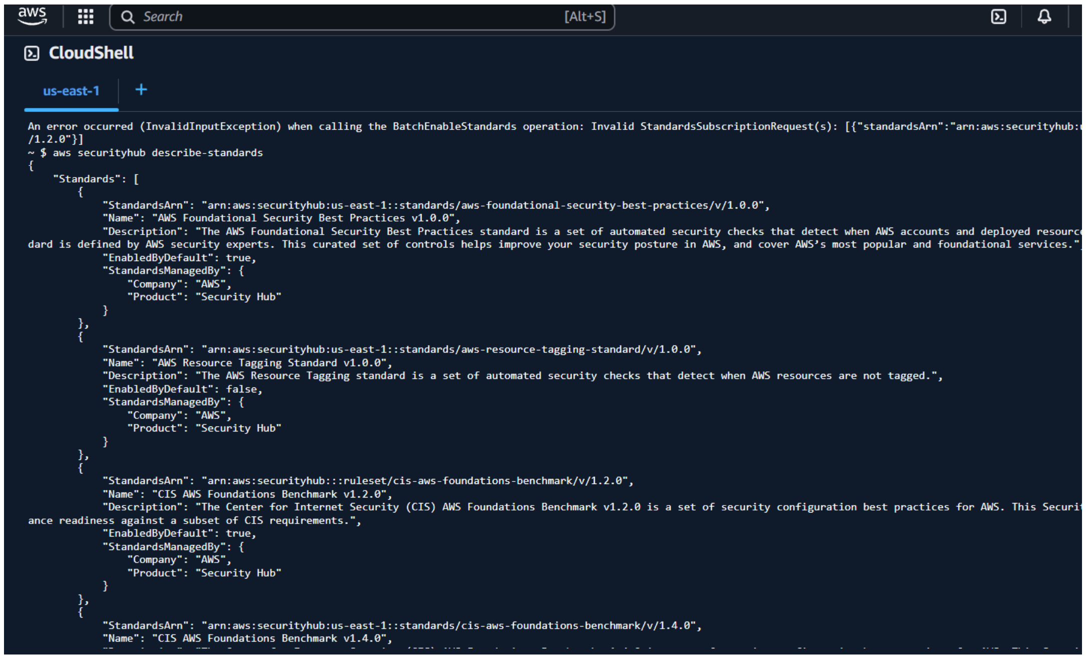
29. Security Hub Integration with GuardDuty Verified (Product Subscriptions)



```
}
~ $ aws securityhub list-enabled-products-for-import --query 'ProductSubscriptions[]'
[
    "arn:aws:securityhub:us-east-1:756716632322:product-subscription/amazon/route-53-resolver-dns-firewall-advanced",
    "arn:aws:securityhub:us-east-1:756716632322:product-subscription/amazon/route-53-resolver-dns-firewall-aws-list",
    "arn:aws:securityhub:us-east-1:756716632322:product-subscription/aws/access-analyzer",
    "arn:aws:securityhub:us-east-1:756716632322:product-subscription/aws/config",
    "arn:aws:securityhub:us-east-1:756716632322:product-subscription/aws/firewall-manager",
    "arn:aws:securityhub:us-east-1:756716632322:product-subscription/aws/guardduty",
    "arn:aws:securityhub:us-east-1:756716632322:product-subscription/aws/health",
    "arn:aws:securityhub:us-east-1:756716632322:product-subscription/aws/inspector",
    "arn:aws:securityhub:us-east-1:756716632322:product-subscription/aws/macie",
    "arn:aws:securityhub:us-east-1:756716632322:product-subscription/aws/ssm-patch-manager",
    "arn:aws:securityhub:us-east-1:756716632322:product-subscription/aws/securityhub"
]
~ $
```

Using CloudShell, we confirm that Security Hub is subscribed to the GuardDuty product. The output of
aws securityhub list-enabled-products-for-import --query 'ProductSubscriptions[]'
includes an ARN containing **product/aws/guardduty**, proving GuardDuty findings can flow into Security Hub for centralised visibility.

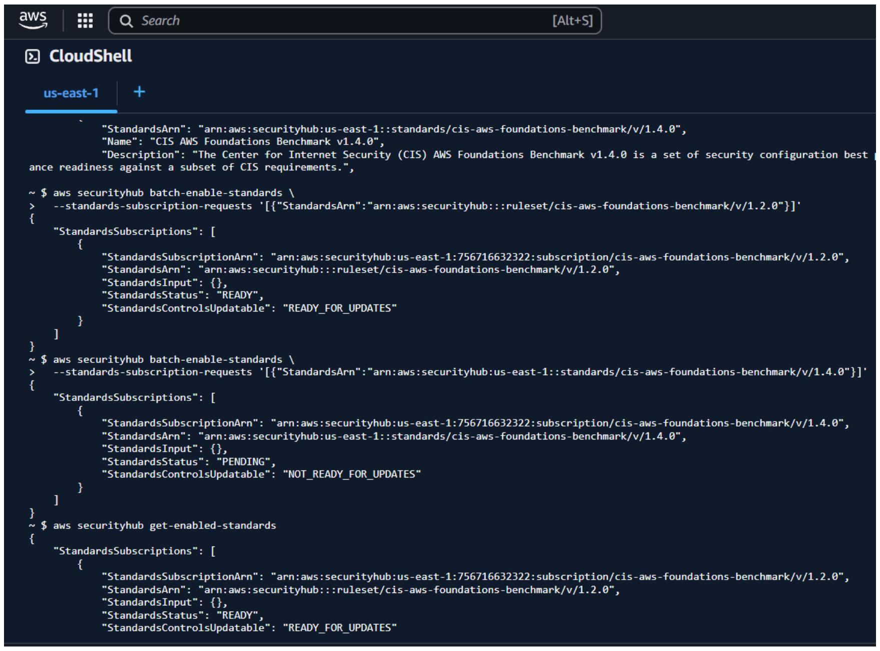
30. Available Security Hub Standards Listed



```
An error occurred (InvalidInputException) when calling the BatchEnableStandards operation: Invalid StandardsSubscriptionRequest(s): [{"standardsArn":"arn:aws:securityhub:
/1.2.0"}]
~ $ aws securityhub describe-standards
{
    "Standards": [
        {
            "StandardsArn": "arn:aws:securityhub:us-east-1::standards/aws-foundational-security-best-practices/v/1.0.0",
            "Name": "AWS Foundational Security Best Practices v1.0.0",
            "Description": "The AWS Foundational Security Best Practices standard is a set of automated security checks that detect when AWS accounts and deployed resourc
dard is defined by AWS security experts. This curated set of controls helps improve your security posture in AWS, and cover AWS's most popular and foundational services."
            "EnabledByDefault": true,
            "StandardsManagedBy": {
                "Company": "AWS",
                "Product": "Security Hub"
            }
        },
        {
            "StandardsArn": "arn:aws:securityhub:us-east-1::standards/aws-resource-tagging-standard/v/1.0.0",
            "Name": "AWS Resource Tagging Standard v1.0.0",
            "Description": "The AWS Resource Tagging standard is a set of automated security checks that detect when AWS resources are not tagged.",
            "EnabledByDefault": false,
            "StandardsManagedBy": {
                "Company": "AWS",
                "Product": "Security Hub"
            }
        },
        {
            "StandardsArn": "arn:aws:securityhub:::ruleset/cis-aws-foundations-benchmark/v/1.2.0",
            "Name": "CIS AWS Foundations Benchmark v1.2.0",
            "Description": "The Center for Internet Security (CIS) AWS Foundations Benchmark v1.2.0 is a set of security configuration best practices for AWS. This Securi
ance readiness against a subset of CIS requirements.",
            "EnabledByDefault": true,
            "StandardsManagedBy": {
                "Company": "AWS",
                "Product": "Security Hub"
            }
        },
        {
            "StandardsArn": "arn:aws:securityhub:us-east-1::standards/cis-aws-foundations-benchmark/v/1.4.0",
            "Name": "CIS AWS Foundations Benchmark v1.4.0",
```

Using the AWS CLI, we query Security Hub to list all available security standards. These include AWS Foundational Security Best Practices and CIS AWS Foundations Benchmarks. Each standard represents a collection of automated security checks that can be enabled to continuously evaluate the environment against industry benchmarks. This step establishes the baseline options for compliance monitoring, prior to enabling or drilling into specific controls.

31. Security Hub Standards Enabled & Initialising

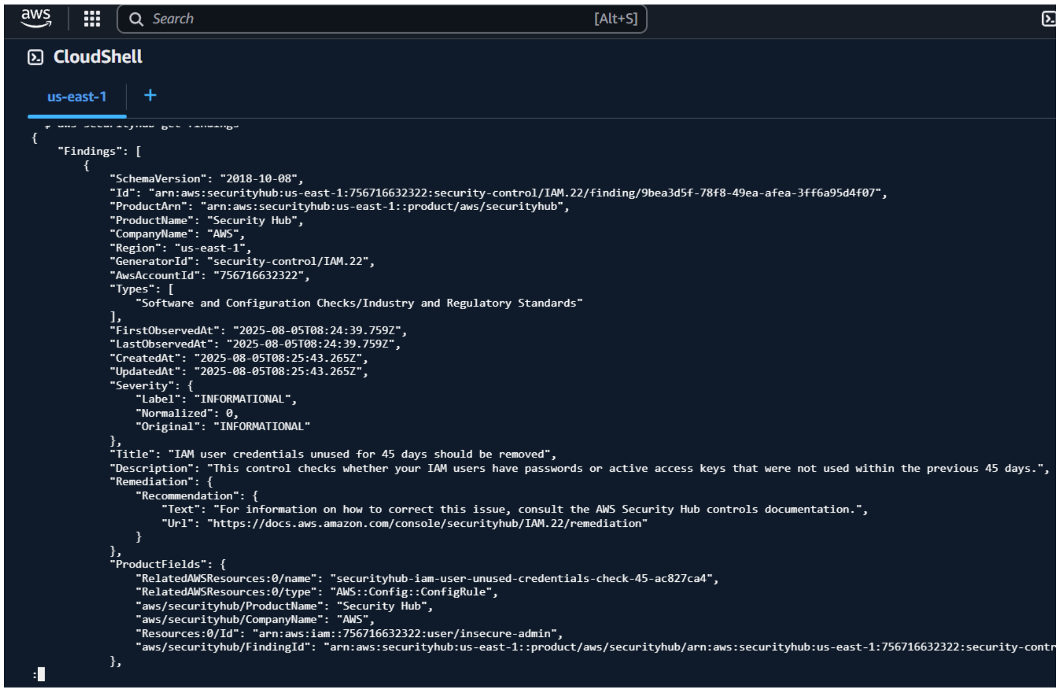


```
        "StandardsArn": "arn:aws:securityhub:us-east-1::standards/cis-aws-foundations-benchmark/v/1.4.0",
        "Name": "CIS AWS Foundations Benchmark v1.4.0",
        "Description": "The Center for Internet Security (CIS) AWS Foundations Benchmark v1.4.0 is a set of security configuration best
ance readiness against a subset of CIS requirements.",

~ $ aws securityhub batch-enable-standards \
>   --standards-subscription-requests '[{"StandardsArn":"arn:aws:securityhub:::ruleset/cis-aws-foundations-benchmark/v/1.2.0"}]'
{
    "StandardsSubscriptions": [
        {
            "StandardsSubscriptionArn": "arn:aws:securityhub:us-east-1:756716632322:subscription/cis-aws-foundations-benchmark/v/1.2.0",
            "StandardsArn": "arn:aws:securityhub:::ruleset/cis-aws-foundations-benchmark/v/1.2.0",
            "StandardsInput": {},
            "StandardsStatus": "READY",
            "StandardsControlsUpdatable": "READY_FOR_UPDATES"
        }
    ]
}
~ $ aws securityhub batch-enable-standards \
>   --standards-subscription-requests '[{"StandardsArn":"arn:aws:securityhub:us-east-1::standards/cis-aws-foundations-benchmark/v/1.4.0"}]'
{
    "StandardsSubscriptions": [
        {
            "StandardsSubscriptionArn": "arn:aws:securityhub:us-east-1:756716632322:subscription/cis-aws-foundations-benchmark/v/1.4.0",
            "StandardsArn": "arn:aws:securityhub:us-east-1::standards/cis-aws-foundations-benchmark/v/1.4.0",
            "StandardsInput": {},
            "StandardsStatus": "PENDING",
            "StandardsControlsUpdatable": "NOT_READY_FOR_UPDATES"
        }
    ]
}
~ $ aws securityhub get-enabled-standards
{
    "StandardsSubscriptions": [
        {
            "StandardsSubscriptionArn": "arn:aws:securityhub:us-east-1:756716632322:subscription/cis-aws-foundations-benchmark/v/1.2.0",
            "StandardsArn": "arn:aws:securityhub:::ruleset/cis-aws-foundations-benchmark/v/1.2.0",
            "StandardsInput": {},
            "StandardsStatus": "READY",
            "StandardsControlsUpdatable": "READY_FOR_UPDATES"
```

We enable two standards in Security Hub:

- **CIS AWS Foundations v1.2.0**
- **AWS Foundational Security Best Practices (FSBP)**
  Status may show **READY** or **INCOMPLETE** briefly while controls initialise and begin evaluating with AWS Config data. This turns on continuous benchmark scoring across the account.

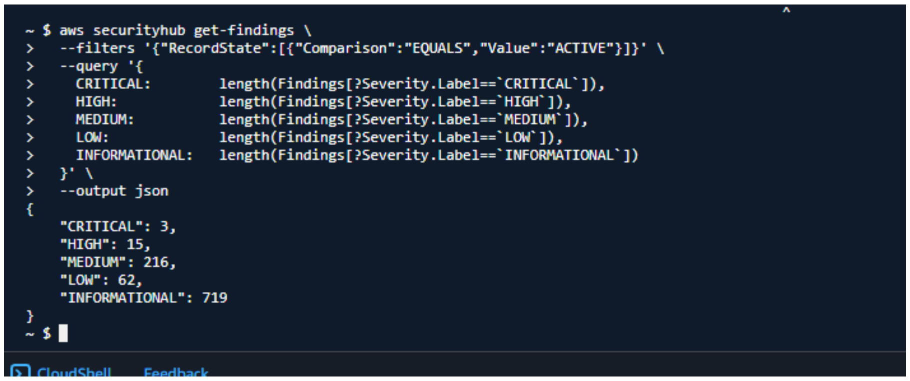
32. Security Hub Findings with Severity Metadata



```
{
    "Findings": [
        {
            "SchemaVersion": "2018-10-08",
            "Id": "arn:aws:securityhub:us-east-1:756716632322:security-control/IAM.22/finding/9bea3d5f-78f8-49ea-afea-3ff6a95d4f07",
            "ProductArn": "arn:aws:securityhub:us-east-1::product/aws/securityhub",
            "ProductName": "Security Hub",
            "CompanyName": "AWS",
            "Region": "us-east-1",
            "GeneratorId": "security-control/IAM.22",
            "AwsAccountId": "756716632322",
            "Types": [
                "Software and Configuration Checks/Industry and Regulatory Standards"
            ],
            "FirstObservedAt": "2025-08-05T08:24:39.759Z",
            "LastObservedAt": "2025-08-05T08:24:39.759Z",
            "CreatedAt": "2025-08-05T08:25:43.265Z",
            "UpdatedAt": "2025-08-05T08:25:43.265Z",
            "Severity": {
                "Label": "INFORMATIONAL",
                "Normalized": 0,
                "Original": "INFORMATIONAL"
            },
            "Title": "IAM user credentials unused for 45 days should be removed",
            "Description": "This control checks whether your IAM users have passwords or active access keys that were not used within the previous 45 days.",
            "Remediation": {
                "Recommendation": {
                    "Text": "For information on how to correct this issue, consult the AWS Security Hub controls documentation.",
                    "Url": "https://docs.aws.amazon.com/console/securityhub/IAM.22/remediation"
                }
            },
            "ProductFields": {
                "RelatedAWSResources:0/name": "securityhub-iam-user-unused-credentials-check-45-ac827ca4",
                "RelatedAWSResources:0/type": "AWS::Config::ConfigRule",
                "aws/securityhub/ProductName": "Security Hub",
                "aws/securityhub/CompanyName": "AWS",
                "Resources:0/Id": "arn:aws:iam::756716632322:user/insecure-admin",
                "aws/securityhub/FindingId": "arn:aws:securityhub:us-east-1::product/aws/securityhub/arn:aws:securityhub:us-east-1:756716632322:security-contr
            },
:
```

Using the AWS CLI, we retrieve Security Hub findings that include detailed metadata such as title, description, severity level, and remediation guidance. This allows security teams to prioritise issues (e.g., informational, low, medium, high, or critical) and take corrective actions quickly. In this example, Security Hub flagged unused IAM user credentials, with guidance on remediation.

33. Security Hub Severity Summary (Active Findings)

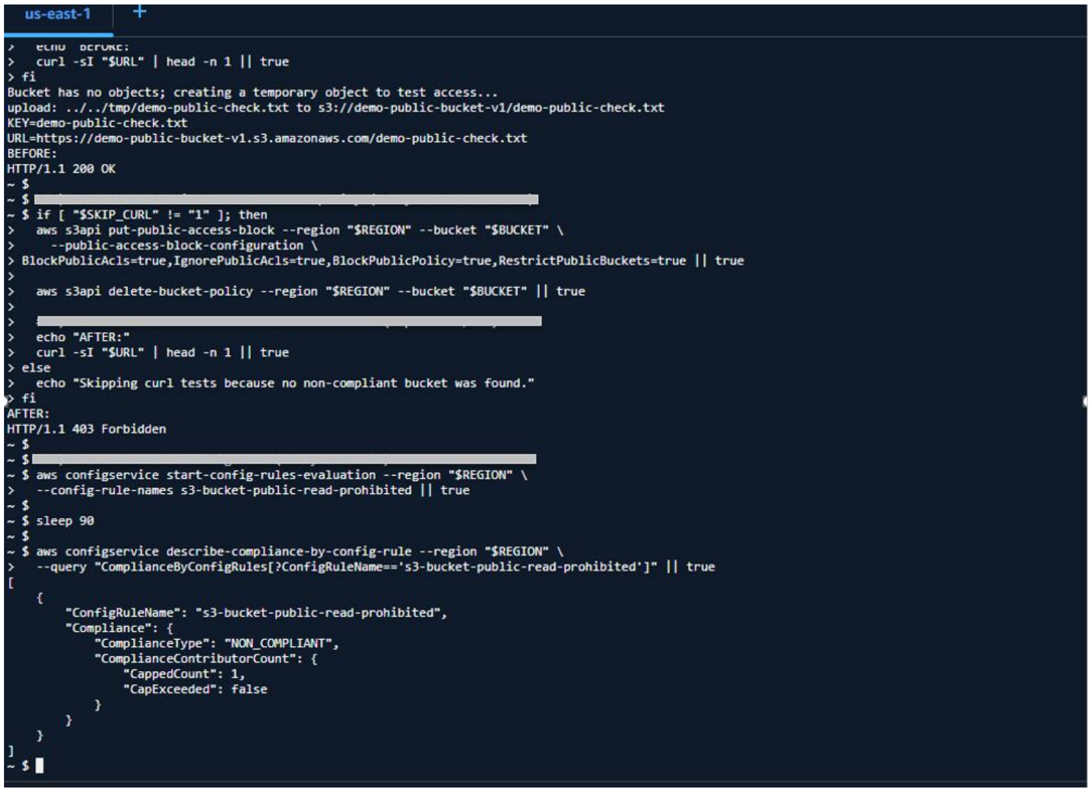


```
~ $ aws securityhub get-findings \
>   --filters '{"RecordState":[{"Comparison":"EQUALS","Value":"ACTIVE"}]}' \
>   --query '{
>     CRITICAL:        length(Findings[?Severity.Label==`CRITICAL`]),
>     HIGH:            length(Findings[?Severity.Label==`HIGH`]),
>     MEDIUM:          length(Findings[?Severity.Label==`MEDIUM`]),
>     LOW:             length(Findings[?Severity.Label==`LOW`]),
>     INFORMATIONAL:   length(Findings[?Severity.Label==`INFORMATIONAL`])
>   }' \
>   --output json
{
    "CRITICAL": 3,
    "HIGH": 15,
    "MEDIUM": 216,
    "LOW": 62,
    "INFORMATIONAL": 719
}
~ $
```

We query Security Hub for **ACTIVE** findings and summarise counts by severity (CRITICAL/HIGH/MEDIUM/LOW/INFORMATIONAL). This gives a fast, board-level view of current risk.

Risk Remediation

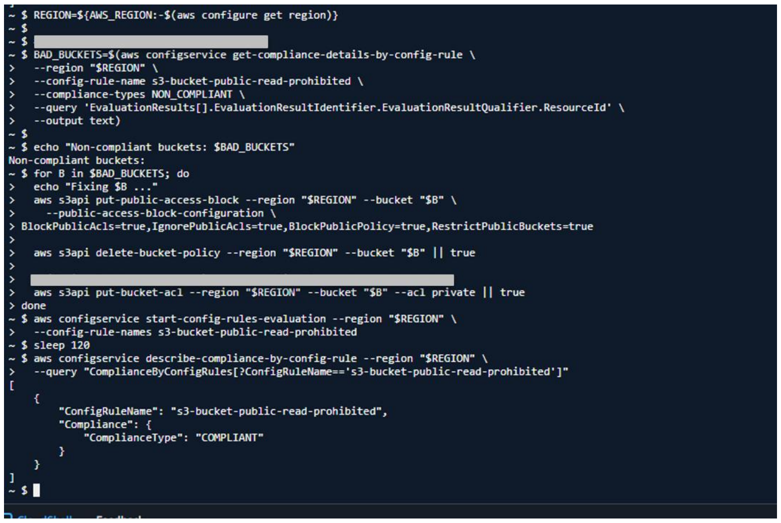
34. S3 Public Read Remediation (Before/After + Re-check)



```
>   echo "BEFORE:"
>   curl -sI "$URL" | head -n 1 || true
> fi
Bucket has no objects; creating a temporary object to test access...
upload: ../../tmp/demo-public-check.txt to s3://demo-public-bucket-v1/demo-public-check.txt
KEY=demo-public-check.txt
URL=https://demo-public-bucket-v1.s3.amazonaws.com/demo-public-check.txt
BEFORE:
HTTP/1.1 200 OK
~ $
~ $ [redacted]
~ $ if [ "$SKIP_CURL" != "1" ]; then
>   aws s3api put-public-access-block --region "$REGION" --bucket "$BUCKET" \
>     --public-access-block-configuration \
> BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=true || true
>
>   aws s3api delete-bucket-policy --region "$REGION" --bucket "$BUCKET" || true
>
>   [redacted]
>   echo "AFTER:"
>   curl -sI "$URL" | head -n 1 || true
> else
>   echo "Skipping curl tests because no non-compliant bucket was found."
> fi
AFTER:
HTTP/1.1 403 Forbidden
~ $
~ $ [redacted]
~ $ aws configservice start-config-rules-evaluation --region "$REGION" \
>   --config-rule-names s3-bucket-public-read-prohibited || true
~ $
~ $ sleep 90
~ $
~ $ aws configservice describe-compliance-by-config-rule --region "$REGION" \
>   --query "ComplianceByConfigRules[?ConfigRuleName=='s3-bucket-public-read-prohibited']" || true
[
    {
        "ConfigRuleName": "s3-bucket-public-read-prohibited",
        "Compliance": {
            "ComplianceType": "NON_COMPLIANT",
            "ComplianceContributorCount": {
                "CappedCount": 1,
                "CapExceeded": false
            }
        }
    }
]
~ $
```

Verified public access with curl -I (200 OK), then blocked exposure by enabling S3 Block Public Access (account + bucket), removing bucket policy/ACLs, and setting object ACLs to private. Re-test returns 403 Forbidden. We triggered a Config re-evaluation; the rule may briefly show NON_COMPLIANT while old evaluations are cached. Final COMPLIANT status is confirmed in the next step (bulk remediation re-check).

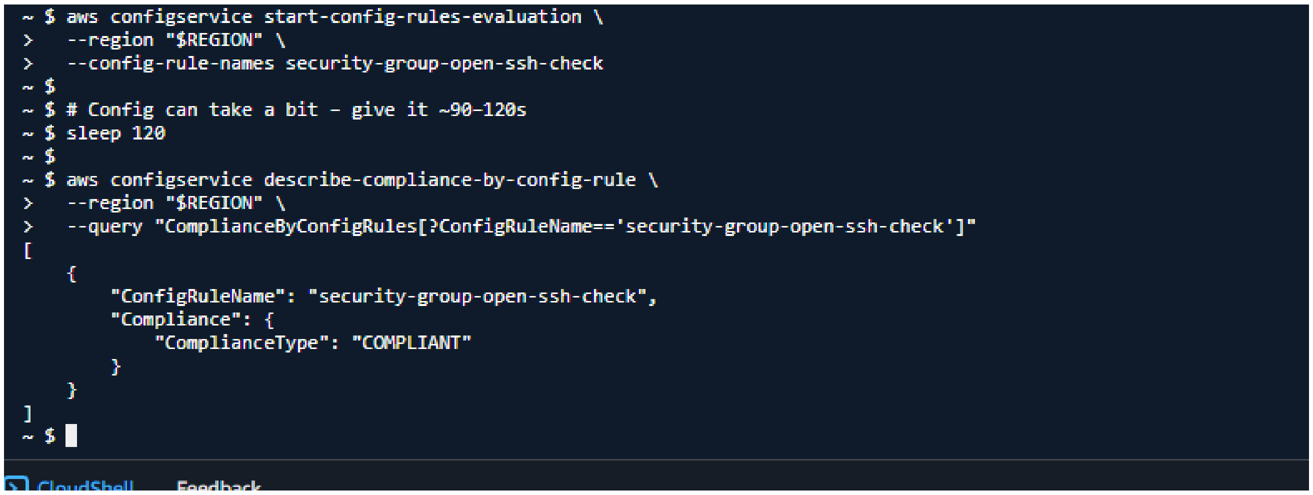
35. Bulk Remediate Public S3 Buckets and Confirm Compliance



```
~ $ REGION=${AWS_REGION:-$(aws configure get region)}
~ $
~ $ [redacted]
~ $ BAD_BUCKETS=$(aws configservice get-compliance-details-by-config-rule \
>   --region "$REGION" \
>   --config-rule-name s3-bucket-public-read-prohibited \
>   --compliance-types NON_COMPLIANT \
>   --query 'EvaluationResults[].EvaluationResultIdentifier.EvaluationResultQualifier.ResourceId' \
>   --output text)
~ $
~ $ echo "Non-compliant buckets: $BAD_BUCKETS"
Non-compliant buckets:
~ $ for B in $BAD_BUCKETS; do
>   echo "Fixing $B ..."
>   aws s3api put-public-access-block --region "$REGION" --bucket "$B" \
>     --public-access-block-configuration \
> BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=true
>
>   aws s3api delete-bucket-policy --region "$REGION" --bucket "$B" || true
>
>  [redacted]
>   aws s3api put-bucket-acl --region "$REGION" --bucket "$B" --acl private || true
> done
~ $ aws configservice start-config-rules-evaluation --region "$REGION" \
>   --config-rule-names s3-bucket-public-read-prohibited
~ $ sleep 120
~ $ aws configservice describe-compliance-by-config-rule --region "$REGION" \
>   --query "ComplianceByConfigRules[?ConfigRuleName=='s3-bucket-public-read-prohibited']"
[
    {
        "ConfigRuleName": "s3-bucket-public-read-prohibited",
        "Compliance": {
            "ComplianceType": "COMPLIANT"
        }
    }
]
~ $
```

Queried AWS Config for all non-compliant buckets and bulk-remediated: applied Public Access Block (all four settings), removed bucket policies, and set ACLs to private. Re-evaluated the rule set and confirmed COMPLIANT. Idempotent commands—safe to rerun.

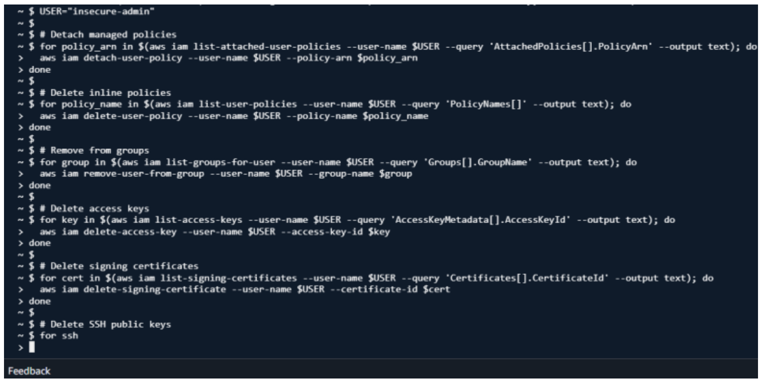
36. Config Rule Status: COMPLIANT After Remediation



```
~ $ aws configservice start-config-rules-evaluation \
>   --region "$REGION" \
>   --config-rule-names security-group-open-ssh-check
~ $
~ $ # Config can take a bit - give it ~90-120s
~ $ sleep 120
~ $
~ $ aws configservice describe-compliance-by-config-rule \
>   --region "$REGION" \
>   --query "ComplianceByConfigRules[?ConfigRuleName=='security-group-open-ssh-check']"
[
    {
        "ConfigRuleName": "security-group-open-ssh-check",
        "Compliance": {
            "ComplianceType": "COMPLIANT"
        }
    }
]
~ $
```

Description: Internet-exposed SSH was removed (IPv4 0.0.0.0/0, IPv6 ::/0) and default security group rules were cleared. An on-demand evaluation of the AWS Config managed rule security-group-open-ssh-check returned ComplianceType: COMPLIANT. This maps to Security Hub control EC2.2 (“Security groups should not allow unrestricted SSH”).

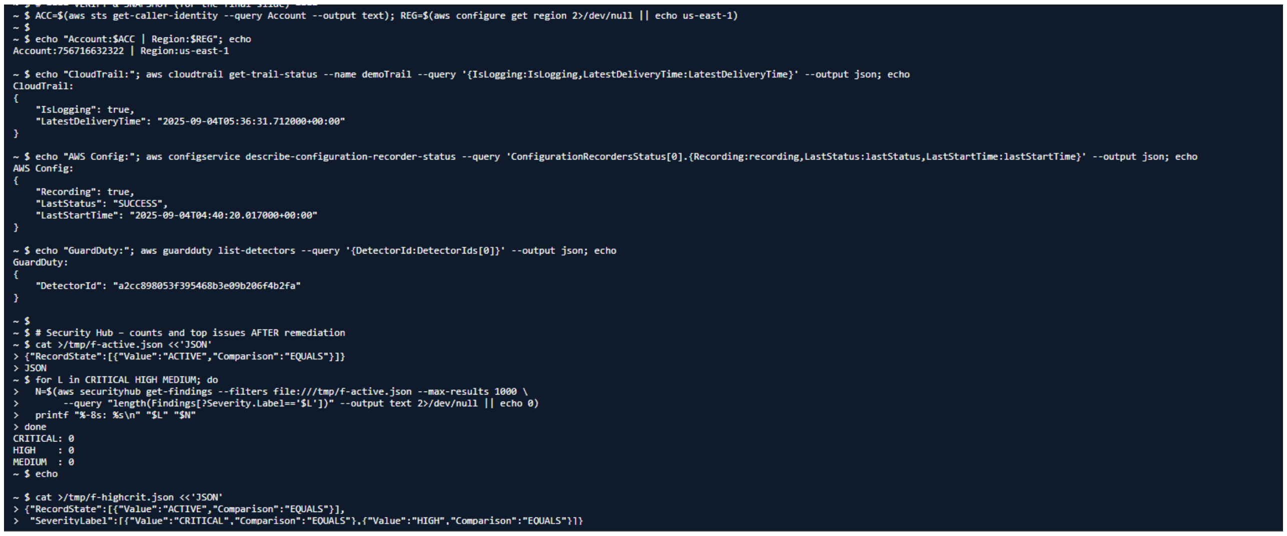
37. Access-risk wrap-up



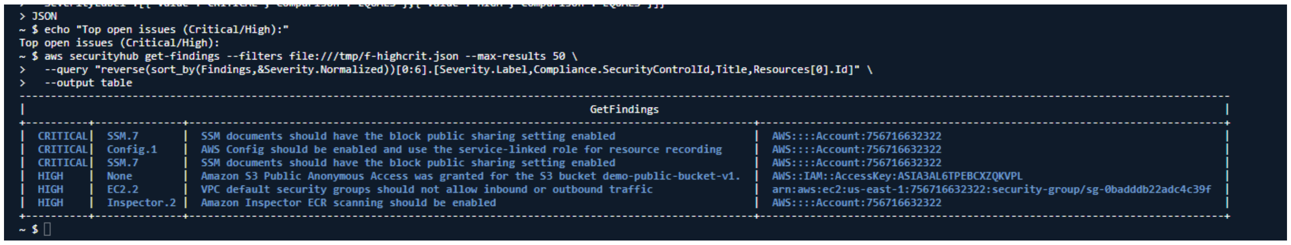
```
~ $ USER="insecure-admin"
~ $
~ $ # Detach managed policies
~ $ for policy_arn in $(aws iam list-attached-user-policies --user-name $USER --query 'AttachedPolicies[].PolicyArn' --output text); do
>   aws iam detach-user-policy --user-name $USER --policy-arn $policy_arn
> done
~ $
~ $ # Delete inline policies
~ $ for policy_name in $(aws iam list-user-policies --user-name $USER --query 'PolicyNames[]' --output text); do
>   aws iam delete-user-policy --user-name $USER --policy-name $policy_name
> done
~ $
~ $ # Remove from groups
~ $ for group in $(aws iam list-groups-for-user --user-name $USER --query 'Groups[].GroupName' --output text); do
>   aws iam remove-user-from-group --user-name $USER --group-name $group
> done
~ $
~ $ # Delete access keys
~ $ for key in $(aws iam list-access-keys --user-name $USER --query 'AccessKeyMetadata[].AccessKeyId' --output text); do
>   aws iam delete-access-key --user-name $USER --access-key-id $key
> done
~ $
~ $ # Delete signing certificates
~ $ for cert in $(aws iam list-signing-certificates --user-name $USER --query 'Certificates[].CertificateId' --output text); do
>   aws iam delete-signing-certificate --user-name $USER --certificate-id $cert
> done
~ $
~ $ # Delete SSH public keys
~ $ for ssh
> 
Feedback
```

Here we remediate an insecure IAM user account (insecure-admin) by detaching policies, removing group memberships, and deleting access keys, certificates, and SSH keys. This ensures that dormant or overly permissive accounts are cleaned up, reducing attack surface and strengthening account security.

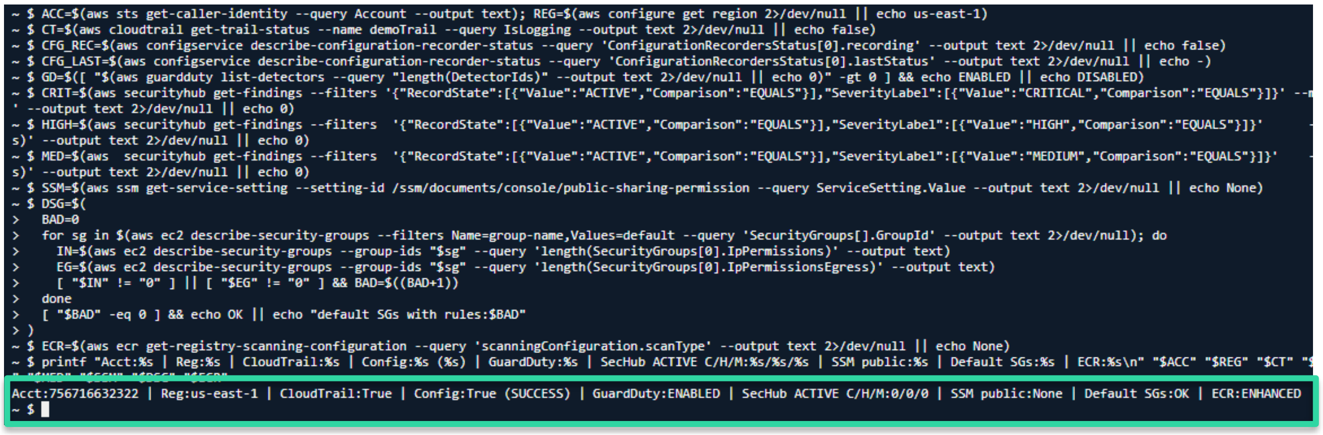
38. Final CSPM Posture & Remediation Status



```
~ $ ACC=$(aws sts get-caller-identity --query Account --output text); REG=$(aws configure get region 2>/dev/null || echo us-east-1)
~ $
~ $ echo "Account:$ACC | Region:$REG"; echo
Account:756716632322 | Region:us-east-1

~ $ echo "CloudTrail:"; aws cloudtrail get-trail-status --name demoTrail --query '{IsLogging:IsLogging,LatestDeliveryTime:LatestDeliveryTime}' --output json; echo
CloudTrail:
{
    "IsLogging": true,
    "LatestDeliveryTime": "2025-09-04T05:36:31.712000+00:00"
}

~ $ echo "AWS Config:"; aws configservice describe-configuration-recorder-status --query 'ConfigurationRecordersStatus[0].{Recording:recording,LastStatus:lastStatus,LastStartTime:lastStartTime}' --output json; echo
AWS Config:
{
    "Recording": true,
    "LastStatus": "SUCCESS",
    "LastStartTime": "2025-09-04T04:40:20.017000+00:00"
}

~ $ echo "GuardDuty:"; aws guardduty list-detectors --query '{DetectorId:DetectorIds[0]}' --output json; echo
GuardDuty:
{
    "DetectorId": "a2cc898053f395468b3e09b206f4b2fa"
}

~ $
~ $ # Security Hub – counts and top issues AFTER remediation
~ $ cat >/tmp/f-active.json <<'JSON'
> {"RecordState":[{"Value":"ACTIVE","Comparison":"EQUALS"}]}
> JSON
~ $ for L in CRITICAL HIGH MEDIUM; do
>   N=$(aws securityhub get-findings --filters file:///tmp/f-active.json --max-results 1000 \
>        --query "length(Findings[?Severity.Label=='$L'])" --output text 2>/dev/null || echo 0)
>   printf "%-8s: %s\n" "$L" "$N"
> done
CRITICAL: 0
HIGH    : 0
MEDIUM  : 0
~ $ echo

~ $ cat >/tmp/f-highcrit.json <<'JSON'
> {"RecordState":[{"Value":"ACTIVE","Comparison":"EQUALS"}],
>  "SeverityLabel":[{"Value":"CRITICAL","Comparison":"EQUALS"},{"Value":"HIGH","Comparison":"EQUALS"}]}
```



```
> JSON
~ $ echo "Top open issues (Critical/High):"
Top open issues (Critical/High):
~ $ aws securityhub get-findings --filters file:///tmp/f-highcrit.json --max-results 50 \
>   --query "reverse(sort_by(Findings,&Severity.Normalized))[0:6].[Severity.Label,Compliance.SecurityControlId,Title,Resources[0].Id]" \
>   --output table
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
|                                                                 GetFindings                                                                                              |
+----------+--------------+-------------------------------------------------------------------------------------+--------------------------------------------------------------+
|  CRITICAL|  SSM.7       |  SSM documents should have the block public sharing setting enabled                 |  AWS::::Account:756716632322                                 |
|  CRITICAL|  Config.1    |  AWS Config should be enabled and use the service-linked role for resource recording |  AWS::::Account:756716632322                                 |
|  CRITICAL|  SSM.7       |  SSM documents should have the block public sharing setting enabled                 |  AWS::::Account:756716632322                                 |
|  HIGH    |  None        |  Amazon S3 Public Anonymous Access was granted for the S3 bucket demo-public-bucket-v1. |  AWS::IAM::AccessKey:ASIA3AL6TPEBCXZQKVPL                |
|  HIGH    |  EC2.2       |  VPC default security groups should not allow inbound or outbound traffic           |  arn:aws:ec2:us-east-1:756716632322:security-group/sg-0badddb22adc4c39f |
|  HIGH    |  Inspector.2 |  Amazon Inspector ECR scanning should be enabled                                    |  AWS::::Account:756716632322                                 |
+----------+--------------+-------------------------------------------------------------------------------------+--------------------------------------------------------------+
~ $ 
```



```
~ $ ACC=$(aws sts get-caller-identity --query Account --output text); REG=$(aws configure get region 2>/dev/null || echo us-east-1)
~ $ CT=$(aws cloudtrail get-trail-status --name demoTrail --query IsLogging --output text 2>/dev/null || echo false)
~ $ CFG_REC=$(aws configservice describe-configuration-recorder-status --query 'ConfigurationRecordersStatus[0].recording' --output text 2>/dev/null || echo false)
~ $ CFG_LAST=$(aws configservice describe-configuration-recorder-status --query 'ConfigurationRecordersStatus[0].lastStatus' --output text 2>/dev/null || echo -)
~ $ GD=$([ "$(aws guardduty list-detectors --query "length(DetectorIds)" --output text 2>/dev/null || echo 0)" -gt 0 ] && echo ENABLED || echo DISABLED)
~ $ CRIT=$(aws securityhub get-findings --filters '{"RecordState":[{"Value":"ACTIVE","Comparison":"EQUALS"}],"SeverityLabel":[{"Value":"CRITICAL","Comparison":"EQUALS"}]}' --m
' --output text 2>/dev/null || echo 0)
~ $ HIGH=$(aws securityhub get-findings --filters  '{"RecordState":[{"Value":"ACTIVE","Comparison":"EQUALS"}],"SeverityLabel":[{"Value":"HIGH","Comparison":"EQUALS"}]}'     -
s)' --output text 2>/dev/null || echo 0)
~ $ MED=$(aws  securityhub get-findings --filters  '{"RecordState":[{"Value":"ACTIVE","Comparison":"EQUALS"}],"SeverityLabel":[{"Value":"MEDIUM","Comparison":"EQUALS"}]}'    -
s)' --output text 2>/dev/null || echo 0)
~ $ SSM=$(aws ssm get-service-setting --setting-id /ssm/documents/console/public-sharing-permission --query ServiceSetting.Value --output text 2>/dev/null || echo None)
~ $ DSG=$(
>   BAD=0
>   for sg in $(aws ec2 describe-security-groups --filters Name=group-name,Values=default --query 'SecurityGroups[].GroupId' --output text 2>/dev/null); do
>     IN=$(aws ec2 describe-security-groups --group-ids "$sg" --query 'length(SecurityGroups[0].IpPermissions)' --output text)
>     EG=$(aws ec2 describe-security-groups --group-ids "$sg" --query 'length(SecurityGroups[0].IpPermissionsEgress)' --output text)
>     [ "$IN" != "0" ] || [ "$EG" != "0" ] && BAD=$((BAD+1))
>   done
>   [ "$BAD" -eq 0 ] && echo OK || echo "default SGs with rules:$BAD"
> )
~ $ ECR=$(aws ecr get-registry-scanning-configuration --query 'scanningConfiguration.scanType' --output text 2>/dev/null || echo None)
~ $ printf "Acct:%s | Reg:%s | CloudTrail:%s | Config:%s (%s) | GuardDuty:%s | SecHub ACTIVE C/H/M:%s/%s/%s | SSM public:%s | Default SGs:%s | ECR:%s\n" "$ACC" "$REG" "$CT" "$
Acct:756716632322 | Reg:us-east-1 | CloudTrail:True | Config:True (SUCCESS) | GuardDuty:ENABLED | SecHub ACTIVE C/H/M:0/0/0 | SSM public:None | Default SGs:OK | ECR:ENHANCED
~ $ 
```

- Controls live: CloudTrail ON, Config SUCCESS, GuardDuty ENABLED, SecHub ACTIVE 0/0/0.
- Key remediations**:** S3 public access blocked (acct & bucket), default SGs not used for workloads; internet-exposed SSH removed, SSM public sharing disabled, ECR scanning ENHANCED.
- Outcome: Baseline risks removed; ongoing monitoring and weekly SecHub triage recommended.

This AWS Cloud Security Posture Assessment (CSPM) demonstration provided a full walkthrough of identifying, validating, and remediating common misconfigurations in an AWS environment. It showcased how to leverage native AWS security tools - including AWS Config, CloudTrail, Security Hub and GuardDuty, to gain visibility, enforce compliance frameworks like CIS, and reduce security risk.

By following this methodology, organisations can:

- Establish continuous monitoring and compliance tracking.
- Detect high-risk misconfigurations early and remediate them efficiently.
- Improve readiness for audits against standards such as CIS, NIST, and SOC 2.
- Maintain a strong, resilient security posture in the cloud.

Remediation Roadmap

Days 0–7 (Stabilise)

- Turn on everywhere: CloudTrail (multi-Region), AWS Config, GuardDuty, Security Hub (auto-enable).
- Block data exposure: S3 Public Access Block (account + buckets); default encryption on log buckets.
- Network hygiene: remove rules on **default** security groups; block 0.0.0.0/0 for SSH/RDP.
- Identity/SSM: root MFA verified; disable SSM public document sharing.
- Inspector2: enable EC2/ECR/Lambda enhanced scanning.
  Outcome: controls active; major misconfigs removed.

By Day 30 (Prevent Drift)

- Deploy CIS + AWS FSBP **Conformance Packs** and key **Config Rules** (S3 public, trail enabled, MFA, key age).
- Org-level enablement (if using AWS Organizations): Security Hub auto-enable, GuardDuty org admin, Config Aggregator.
- Wire alerts: Security Hub **Critical/High** → SNS/Slack/Jira with ownership.

Days 60–90 (Operate)

- Automate fixes (SSM Automation/Lambda) for public S3, open SGs, missing encryption.
- Exception process with owner + expiry.
- Monthly evidence export (Config evaluations & SecHub summaries) to the logs bucket.
  KPIs: SecHub ACTIVE Crit/High/Med trend ↓; 100% accounts with controls on; MTTR for Crit/High; # changes blocked by guardrails.

Ongoing Practice

Regularly reviewing configuration baselines, validating access controls, and integrating findings into security operations ensures sustained cloud compliance and resilience. Combining AWS native security services with a structured remediation process creates an environment where security and compliance are continuously maintained, not just assessed periodically.