

AWS Cloud Security Posture Assessment - Demonstration Project

Enhancing Cyber Resilience Through Visibility, Compliance, and Risk Reduction

Executive Summary

Objective:

This project simulates a typical insecure AWS environment, then demonstrates how to detect and remediate misconfigurations using AWS native security services and industry best practices. The deliverable is designed for business leaders who need actionable insights into cloud security risks and a clear roadmap for remediation.

Why This Matters:

Cloud misconfigurations remain one of the top causes of data breaches. Organisations lacking visibility into their AWS environment risk:

- Data exposure via public S3 buckets
- Unauthorised account access via overly permissive IAM roles
- Unmonitored API activity and configuration changes
- Non-compliance with frameworks like CIS, NIST, or SOC 2

Business Value Delivered:

- Reduced Risk: Early detection of high-impact vulnerabilities (e.g., open SSH ports, lack of MFA).
- Audit Readiness: Audit Readiness: Measured against CIS AWS Foundations with clear evidence and remediation steps.
- Operational Efficiency: Centralized 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 – Prioritize 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 recognized 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.

Next Steps for a Client Engagement:

1. Deploy this assessment in the client's AWS environment.
2. Present findings in a board-ready report with risk heatmaps.
3. Facilitate remediation workshops with DevOps/Cloud teams.
4. Establish an ongoing compliance monitoring dashboard for leadership.

Cloud Security Posture Demo (AWS CSPM Walkthrough)

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.

1. Public S3 Bucket Policy Configured

```
CloudShell

us-east-1 +

> {
>   "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-<u>
> # fully disable public-access-block
> aws s3api put-public-access-block \
>   --bucket cspm-demo-<u> \
>   --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 attach 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

```
CloudShell

us-east-1 +

> 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-<u>/*"
>   }]
> }
> EOF
> aws s3api put-bucket-policy --bucket cspm-demo-<u> --policy file://public-read-policy.json
> # upload test file
> echo "Public test file" > test.txt
> aws s3 cp test.txt s3://cspm-demo-<u>/test.txt
> # verify in-browser
> curl https://cspm-demo-<u>.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 text
> 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-bucket-policy \
>   --bucket cspm-demo-eor-20250804 \
>   --output json
```

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 | Search [Alt+S]

CloudShell

us-east-1 +

> 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'
~$ cat > public-read-policy.json << '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\"",
  "ChecksumCRC64NMVE": "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

aws | Search [Alt+S]

CloudShell

us-east-1 +

~$ 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
~$ aws iam create-user --user-name insecure-admin
{
  "User": {
    "Path": "/",
    "UserName": "insecure-admin",
    "UserId": "AIDAJ54TXXXXXX",
    "Arn": "arn:aws:iam::75616632322: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": "AKIAI44QH8DHBEXAMPLE",
    "Status": "Active",
    "SecretAccessKey": "wJalrXU3WhdcZBKs1bzP0t4qR3zUq8S32U",
    "CreateDate": "2025-08-05T04:00:51+00:00"
  }
}
~$ export AWS_ACCESS_KEY_ID=AKIAI44QH8DHBEXAMPLE
~$ export AWS_SECRET_ACCESS_KEY=wJalrXU3WhdcZBKs1bzP0t4qR3zUq8S32U
~$ export AWS_ACCESS_KEY_ID="AKIAI44QH8DHBEXAMPLE"
~$ export AWS_SECRET_ACCESS_KEY="wJalrXU3WhdcZBKs1bzP0t4qR3zUq8S32U"
~$ aws s3 ls
2025-08-04 13:23:51 cspm-demo-eor-20250804
2025-08-04 13:24:34 cspmdemolouis1
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 is 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

```
aws | Search [Alt+S]

CloudShell

us-east-1 +

{
  "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

```
aws | Search [Alt+S]

CloudShell

us-east-1 +

~ $ 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
~ $ 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.

7. Fixing Bucket Policy for CloudTrail Integration



```
aws | Search

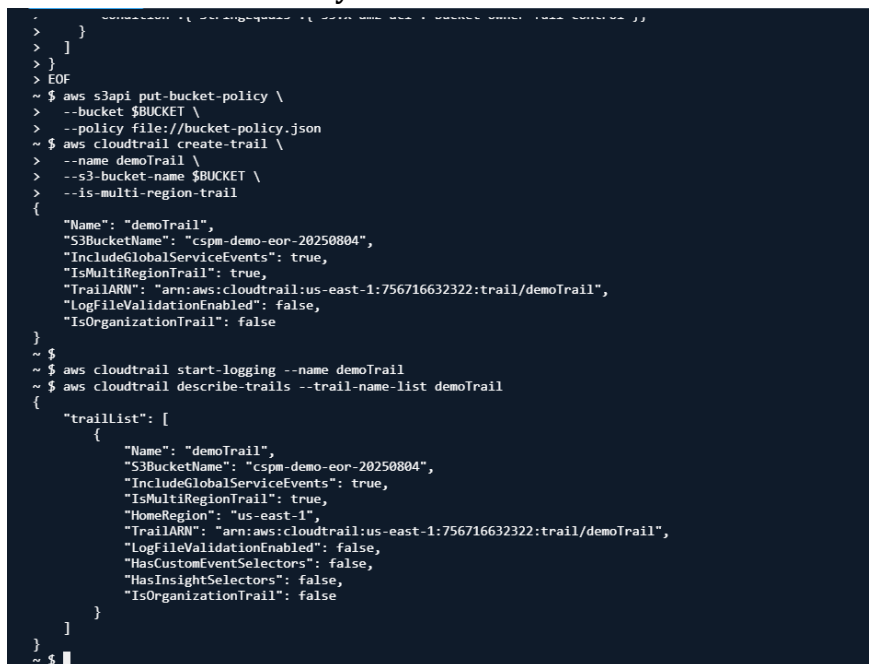
CloudShell

us-east-1 +

> {
>   "Action": "s3:GetObject",
>   "Resource": "arn:aws:s3:::$BUCKET/*"
> },
> {
>   "Sid": "AWSCloudTrailAcICheck",
>   "Effect": "Allow",
>   "Principal": { "Service": "cloudtrail.amazonaws.com" },
>   "Action": "s3:GetBucketAcl",
>   "Resource": "arn:aws:s3:::$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



```
>   "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
~ $ 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
    }
  ]
}
~ $
```

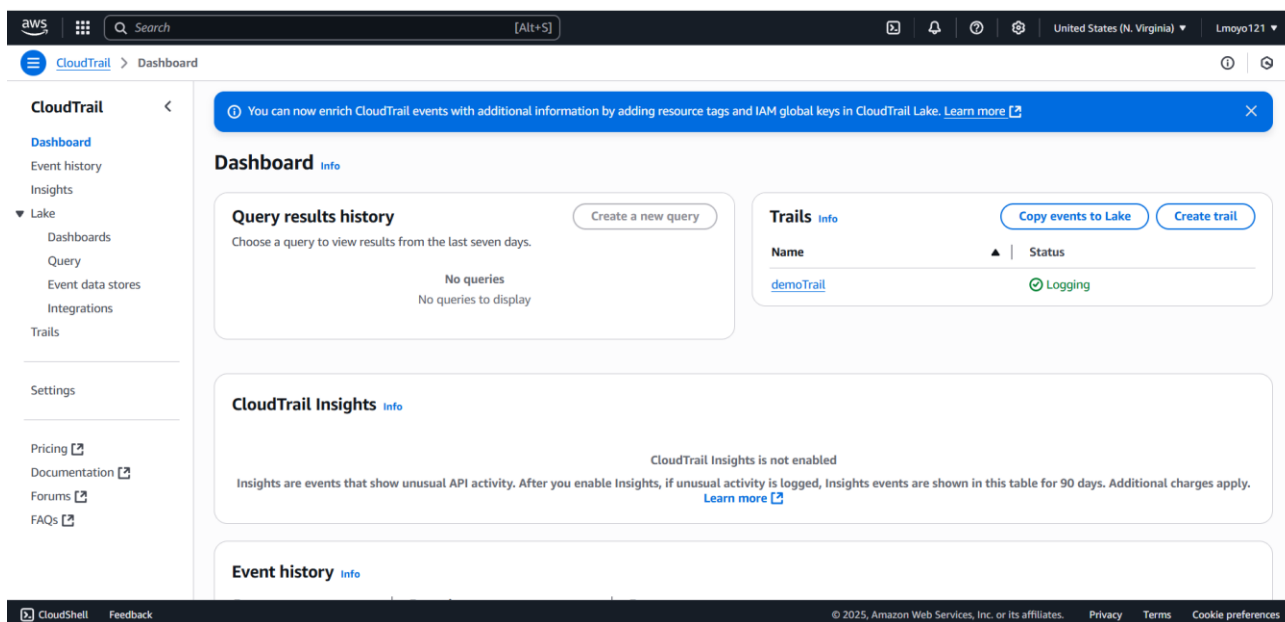
CloudTrail is successfully configured to track API activity across the account and linked to our demo S3 bucket. This is critical for audit logging and forensic visibility.

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



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. At this stage, we know CloudTrail is enabled, but this does not yet confirm that log files have been successfully delivered to the S3 bucket.

11. S3 Bucket Destination and Logging Status Displayed

The screenshot shows the AWS CloudTrail console interface. The left sidebar contains navigation links: CloudTrail, Trails, Dashboard, Event history, Insights, Lake, Dashboards, Query, Event data stores, Integrations, Trails (selected), Settings, Pricing, Documentation, Forums, and FAQs. The main content area displays the details for a trail named 'demoTrail'. The 'General details' section includes: Trail logging (Logging), Trail log location (cspm-demo-eor-20250804/AWSLogs/756716632322), Trail name (demoTrail), Multi-region trail (Yes), Apply trail to my organization (Not enabled), Log file validation (Disabled), Last file validation delivered (-), SNS notification delivery (Disabled), and Last SNS notification (-). The 'CloudWatch Logs' section shows 'No CloudWatch Logs log groups' with a note that CloudWatch Logs is not configured for this trail. The 'Tags' section is empty. At the bottom, there is a 'CloudShell' button and a 'Feedback' link. The footer contains copyright information for Amazon Web Services, Inc. and links to Privacy, Terms, and Cookie preferences.

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 sts get-caller-identity --query Region --output text)
~ $ # 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_202509030655Z_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

```
An error occurred (NoSuchConfigurationRecorderException) when calling the StartConfigurationRecorder operation: The configuration recorder does not exist. You must create the configuration recorder before you can start it.

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

An error occurred (NoSuchConfigurationRecorderException) when calling the StartConfigurationRecorder operation: The configuration recorder does not exist. You must create the configuration recorder before you can start it.
~ $ aws iam create-service-linked-role \
> --aws-service-name config.amazonaws.com
{
  "Role": {
    "Path": "/aws-service-role/config.amazonaws.com/",
    "RoleName": "AWSServiceRoleForConfig",
    "RoleId": "AROA3AL6TPEBAG2IVQNB4",
    "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



```
aws | Search [Alt]
CloudShell
us-east-1 +
~ $ 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. Expanded Bucket Policy for Config Support

   Q Search

CloudShell

us-east-1

+

```
> cat s3api put-bucket-policy.json
{
  "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 |
+-----+-----+
~ $
```

Feedback

We are updating the S3 bucket policy to allow AWS Config to write data to the correct location. This step ensures secure and successful delivery of configuration snapshots.

16. AWS Configuration Recorder Setup

```
aws CloudShell
us-east-1 +
$ aws configservice put-recorder-configuration \
  --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
+-----+-----+-----+-----+-----+-----+
| arn | name | recordingScope | roleARN |
+-----+-----+-----+-----+-----+-----+
| arn:aws:config:us-east-1:756716632322:configuration-recorder/default:d0bgn6jh18cj06va | 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 |
+-----+-----+-----+-----+-----+-----+
```

We set up and check the configuration recorder and delivery channel - key components that enable AWS Config to monitor and record resource changes.

17. Configuration Recorder Successfully Created

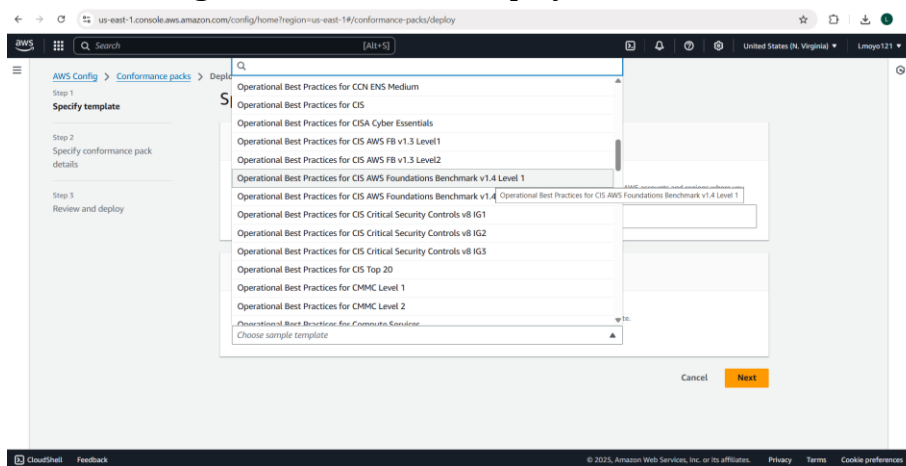
```
~ $ aws configservice describe-configuration-records --output table
DescribeConfigurationRecords
ConfigurationRecords
+-----+-----+-----+-----+-----+
| arn                                     | name | recordingScope | roleARN |
+-----+-----+-----+-----+-----+
| arn:aws:config:us-east-1:756716632322:configuration-recorder/default:d0bgn6jh18cj06va | 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 |
+-----+-----+-----+-----+
| recordingFrequency | recordingMode |
+-----+-----+-----+-----+
| | CONTINUOUS |
+-----+-----+-----+-----+

~ $ aws configservice describe-delivery-channels --output table
DescribeDeliveryChannels
DeliveryChannels
+-----+-----+
| name | s3BucketName |
+-----+-----+
| default | cspm-demo-eor-20250804 |
+-----+-----+

~ $
```

The configuration recorder is now live and configured with continuous recording mode - meaning all supported AWS resources will now be monitored in real time.

18. AWS Config Conformance Pack Deployment Initiated



Here we see the start of deploying a Conformance Pack in AWS Config. Conformance Packs are collections of AWS Config rules based on industry standards such as CIS, NIST, PCI DSS, and CMMC. By selecting a template, we can enforce security best practices and continuously monitor resources for compliance against a chosen benchmark. This step represents the initiation of compliance-as-code within the environment, rather than a dashboard summary.

19. Configuration Recorder Setup Failure

```
}
~ $ aws configservice put-configuration-recorder \
> --configuration-recorder name=default,roleARN=arn:aws:iam:$(aws sts get-caller-identity --query Account --output text):role/aws-service-role/config.amazonaws.com/AWSServiceRoleForConfig
~ $
~ $ aws configservice put-delivery-channel \
> --delivery-channel name=default,s3BucketName=<your-config-bucket-name>
-bash: syntax error near unexpected token `newline'
~ $
~ $ aws configservice start-configuration-recorder --configuration-recorder-name default
~ $
```

This attempt to set up the configuration recorder and delivery channel failed due to a syntax error and a missing bucket policy. Notice the syntax error message near the newline character and the lack of permission for the delivery channel to access the S3 bucket.

20. Fixing S3 Bucket Policy for AWS Config Delivery

```
us-east-1 +
An error occurred (ResourceConflictException) when calling the EnableSecurityHub operation: Account is already subscribed to Security Hub
~ $ aws s3 mb s3://my-config-bucket-logs
make_bucket: my-config-bucket-logs
~ $ aws configservice put-delivery-channel \
> --delivery-channel name=default,s3BucketName=my-config-bucket-logs

An error occurred (InsufficientDeliveryPolicyException) when calling the PutDeliveryChannel operation: Insufficient delivery policy to s3 bucket: my-config-bucket-logs, unable to write to bucket, provided s3 key prefix is
'null', provided kms key is 'null'.
~ $ aws s3api put-bucket-policy --bucket my-config-bucket-logs --policy '{
> "Version": "2012-10-17",
> "Statement": [
> {
> "Sid": "AWSConfigBucketPermissionsCheck",
> "Effect": "Allow",
> "Principal": {
> "Service": "config.amazonaws.com"
> },
> "Action": "s3:GetBucketAcl",
> "Resource": "arn:aws:s3::my-config-bucket-logs"
> },
> {
> "Sid": "AWSConfigBucketDelivery",
> "Effect": "Allow",
> "Principal": {
> "Service": "config.amazonaws.com"
> },
> "Action": "s3:PutObject",
> "Resource": "arn:aws:s3::my-config-bucket-logs/AWSLogs/*",
> "Condition": {
> "StringEquals": {
> "s3:x-amz-acl": "bucket-owner-full-control"
> }
> }
> }
> ]
> }'
~ $ aws configservice put-delivery-channel \
> --delivery-channel name=default,s3BucketName=my-config-bucket-logs
~ $ aws configservice start-configuration-recorder --configuration-recorder-name default
~ $ |
```

Here, a proper bucket policy is added to the `my-config-bucket-logs` S3 bucket to allow AWS Config to deliver logs. This includes granting permissions to `config.amazonaws.com` for both `s3:GetBucketAcl` and `s3:PutObject` actions, with conditions ensuring full control by the bucket owner. The delivery channel command is then re-run successfully.

21. Configuration Recorder Running

```
> {
> "Effect": "Allow",
> "Principal": {
> "Service": "config.amazonaws.com"
> },
> "Action": "s3:GetBucketAcl",
> "Resource": "arn:aws:s3::my-config-bucket-logs"
> },
> {
> "Sid": "AWSConfigBucketDelivery",
> "Effect": "Allow",
> "Principal": {
> "Service": "config.amazonaws.com"
> },
> "Action": "s3:PutObject",
> "Resource": "arn:aws:s3::my-config-bucket-logs/AWSLogs/*",
> "Condition": {
> "StringEquals": {
> "s3:x-amz-acl": "bucket-owner-full-control"
> }
> }
> }
> }
~ $ aws configservice put-delivery-channel \
~ $ --delivery-channel name=default,s3BucketName=my-config-bucket-logs
~ $ 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/7lwdbn4ngtcd97",
      "name": "default",
      "lastStartTime": "2025-08-05T06:56:29.418000+00:00",
      "recording": true,
      "lastStatus": "SUCCESS",
      "lastStatusChangeTime": "2025-08-05T08:41:06.863000+00:00"
    }
  ]
}
```

Final verification of the configuration recorder shows it is successfully recording. The status shows `recording: true` and `lastStatus: SUCCESS`, confirming that AWS Config is now operational.

22. Configuration Recorder Running (S3 Delivery Channel Verified)

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

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

~ $
~ $ 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:lastStartTime}'
>   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.

AWS Config

Dashboard

Compliance packs

Rules

Resources

Aggregators

Compliance Dashboard

Conformance packs

Rules

Inventory Dashboard

Resources

Authorizations

Advanced queries

Settings

What's new

Documentation

Partners

FAQs

Pricing

Rules (44)

<

1

2

3

4

5

>

⚙️

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

https://us-east-1.console.aws.amazon.com/console/home?region=us-east-1

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

AWS
[Alt+]5
United States (N. Virginia)
Lmoyo121

AWS Config

- [Dashboard](#)
- [Conformance packs](#)
- [Rules](#)
- [Resources](#)
- ▼ Aggregators
 - [Compliance Dashboard](#)
 - [Conformance packs](#)
 - [Rules](#)
 - [Inventory Dashboard](#)
 - [Resources](#)
 - [Authorizations](#)
- [Advanced queries](#) [Preview](#)
- [Settings](#)
- [What's new](#)

- [Documentation](#)
- [Partners](#)
- [FAQs](#)
- [Pricing](#)

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

arm:aws:controlcatalog::control/8sw3pbid15t9c
bwv8d2w2qwgf

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

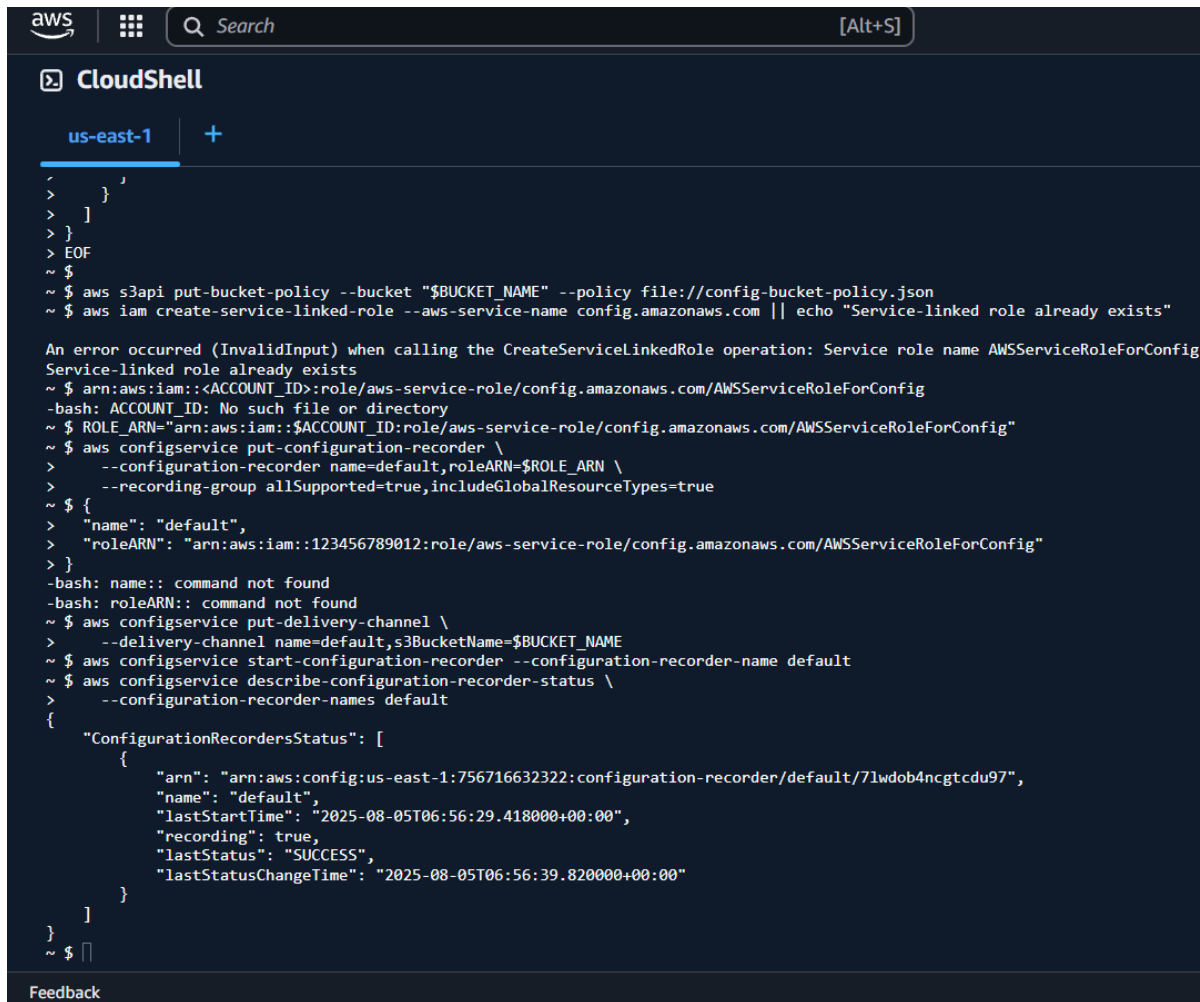
[View details](#)
[Remediate](#)

Noncompliant ▼
< 1 >

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

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

25. AWS Config Recorder and Delivery Channel Validated

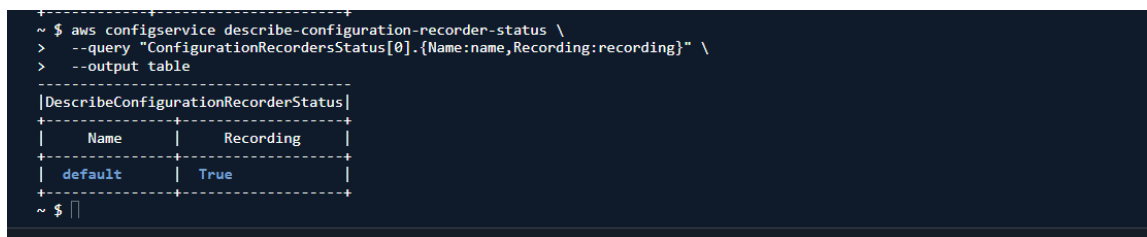


```
aws [Search] [Alt+S]
CloudShell
us-east-1 +
~$ }
~$ ]
~$ }
~$ EOF
~$
~$ aws s3api put-bucket-policy --bucket "$BUCKET_NAME" --policy file://config-bucket-policy.json
~$ aws iam create-service-linked-role --aws-service-name config.amazonaws.com || echo "Service-linked role already exists"

An error occurred (InvalidInput) when calling the CreateServiceLinkedRole operation: Service role name AWSServiceRoleForConfig
Service-linked role already exists
~$ arn:aws:iam::<ACCOUNT_ID>:role/aws-service-role/config.amazonaws.com/AWSServiceRoleForConfig
-bash: ACCOUNT_ID: No such file or directory
~$ ROLE_ARN="arn:aws:iam::${ACCOUNT_ID}:role/aws-service-role/config.amazonaws.com/AWSServiceRoleForConfig"
~$ aws configservice put-configuration-recorder \
> --configuration-recorder name=default,roleARN=$ROLE_ARN \
> --recording-group allSupported=true,includeGlobalResourceTypes=true
~$ {
> "name": "default",
> "roleARN": "arn:aws:iam::123456789012:role/aws-service-role/config.amazonaws.com/AWSServiceRoleForConfig"
> }
-bash: name:: command not found
-bash: roleARN:: command not found
~$ aws configservice put-delivery-channel \
> --delivery-channel name=default,s3BucketName=$BUCKET_NAME
~$ aws configservice start-configuration-recorder --configuration-recorder-name default
~$ aws configservice describe-configuration-recorder-status \
> --configuration-recorder-names default
{
  "ConfigurationRecordersStatus": [
    {
      "arn": "arn:aws:config:us-east-1:756716632322:configuration-recorder/default/71wdob4ncgtcd97",
      "name": "default",
      "lastStartTime": "2025-08-05T06:56:29.418000+00:00",
      "recording": true,
      "lastStatus": "SUCCESS",
      "lastStatusChangeTime": "2025-08-05T06:56:39.820000+00:00"
    }
  ]
}
~$
```

Using the AWS CLI, we configure and validate the AWS Config service by setting up the configuration recorder and delivery channel. The final status output confirms that the recorder is active and successfully delivering configuration data. This ensures that changes to supported AWS resources are being continuously tracked and stored in the S3 bucket for audit and compliance purposes.

26. AWS Config Recorder Status Confirmed



```
~$ aws configservice describe-configuration-recorder-status \
> --query "ConfigurationRecordersStatus[0].{Name:name,Recording:recording}" \
> --output table

DescribeConfigurationRecorderStatus
+-----+-----+
| Name | Recording |
+-----+-----+
| default | True |
+-----+-----+
~$
```

Using the AWS CLI, we confirm that the AWS Config recorder named default is actively recording configuration changes across supported resources. This ensures that configuration history is being captured and delivered to the S3 bucket. To view specific compliance rule results, additional commands such as describe-compliance-by-config-rule would be required.

27. 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.

28. GuardDuty Finding Example (CLI)

```
query -- Findings[0] {title,title,type,type,severity,severity,region,region,resource,resource}
{
  "Title": "A domain name related to known malicious domains was queried by EC2 instance i-999999999.",
  "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-999999999",
      "InstanceId": "i-999999999",
      "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.

29. Security Hub Enabled and CIS Rule Deployed

```
+-----+-----+
~ $ 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
```

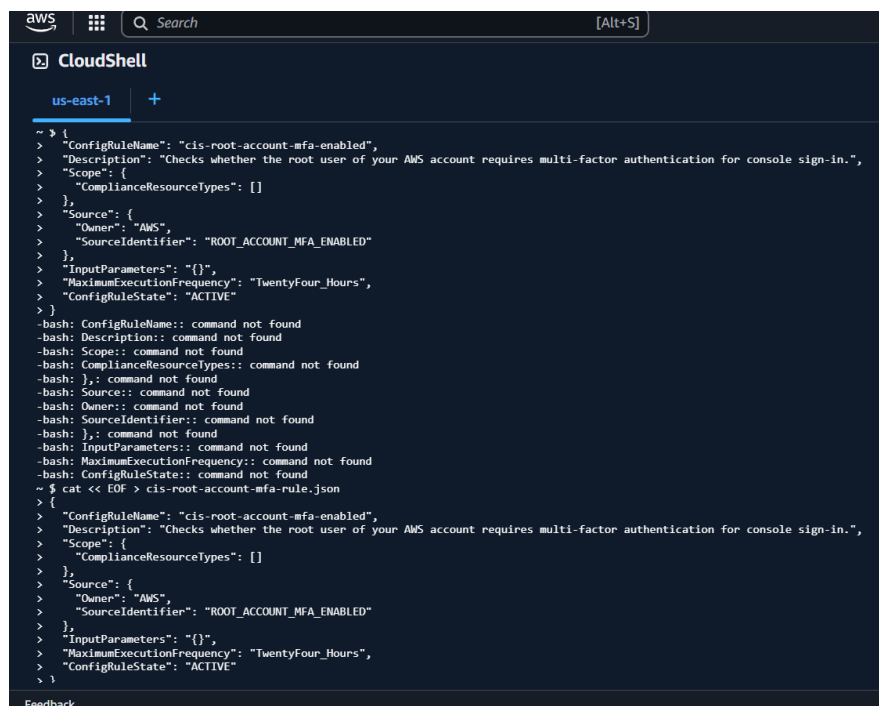
Here, we integrate GuardDuty with Security Hub and begin deploying CIS compliance rules in AWS Config. GuardDuty is confirmed as active, Security Hub is enabled for centralised findings, and a CIS rule for root account MFA is created. While no findings are yet returned, this step establishes the foundation for automated compliance monitoring and remediation guidance.

30. 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/maciek",
  "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 centralized visibility.

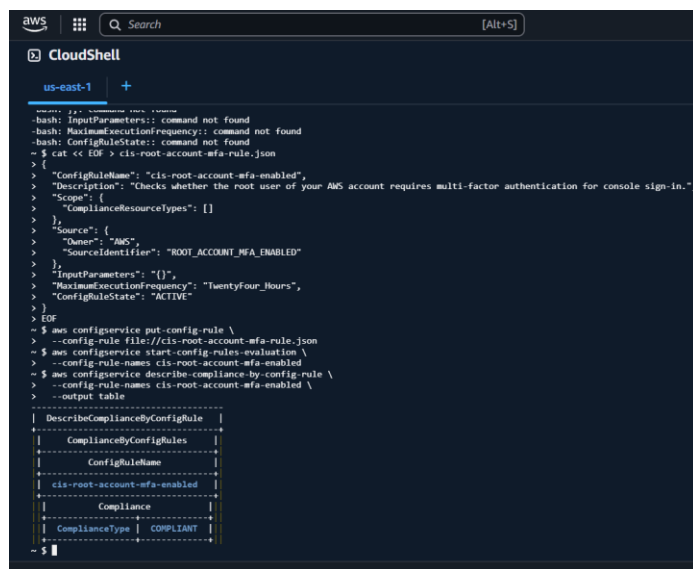
31. CIS Config Rule Definition for MFA



```
aws CloudShell [Alt+S]
us-east-1 +
~$ {
> "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.

32. AWS Config Compliance Evaluation Results



```
aws CloudShell [Alt+S]
us-east-1 +
~$ {
> "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"
> }
> }
~$ cat << EOF > cis-root-account-mfa-rule.json
~$ 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.

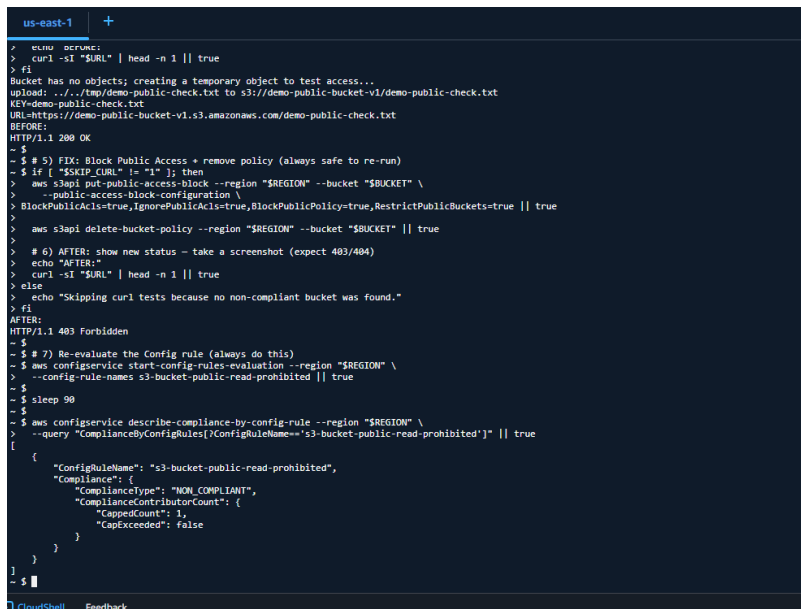
33. Additional AWS Config Rules Deployed



```
aws
CloudShell
us-east-1 +
+-----+
| 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.

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



```
us-east-1 +
> echo occur:
> curl -sI "$URL" | head -n 1 || true
> #1
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
$
# 5) FIX: Block Public Access + remove policy (always safe to re-run)
$ 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
$
# 6) AFTER: show new status - take a screenshot (expect 403/404)
$ echo "AFTER:"
> curl -sI "$URL" | head -n 1 || true
> else
> echo "Skipping curl tests because no non-compliant bucket was found."
> #1
AFTER:
HTTP/1.1 403 Forbidden
$
# 7) Re-evaluate the Config rule (always do this)
$ 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
    }
  }
}
]
$
```

Validated the bucket was publicly readable by issuing curl -I (returned **200 OK**). Remediated by enabling **S3 Block Public Access** (BlockPublicAcls, IgnorePublicAcls, BlockPublicPolicy, RestrictPublicBuckets) and removing the bucket policy. A re-test returned **403 Forbidden**, confirming public read is blocked. Finally, re-evaluated the AWS Config rule **s3-bucket-public-read-prohibited**; it should show **COMPLIANT** after propagation, or once any remaining public buckets are similarly fixed.

35. Bulk Remediate Public S3 Buckets and Confirm Compliance

```
~ $ REGION=$(aws configure get region)
~ $
~ $ # List all non-compliant bucket names
~ $ 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 "$B" || true
>
> # (Optional but recommended) remove public ACL grants on the bucket
> aws s3api put-bucket-acl --region "$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"
    }
  }
]
```

CloudShell Feedback

36. Expanded Set of AWS Config Rules Deployed

```
aws
CloudShell
us-east-1 +
~ $ aws configservice put-config-rule \
> --config-rule '{
>   "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"
> }'
~ $ aws configservice put-config-rule \
> --config-rule '{
>   "ConfigRuleName": "cloudtrail-enabled",
>   "Description": "Checks whether AWS CloudTrail is enabled in your account.",
>   "Source": {
>     "Owner": "AWS",
>     "SourceIdentifier": "CLOUD_TRAIL_ENABLED"
>   },
>   "ConfigRuleState": "ACTIVE"
> }'
~ $
```

Here we expand the compliance scope by deploying multiple AWS Config rules to enforce best practices. These rules cover critical security areas including S3 bucket public access, root access key restrictions, IAM user MFA enforcement, and CloudTrail enablement. Once activated, these rules continuously evaluate account resources and surface violations for remediation.

37. 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"  
    }  
  }  
]  
~ $
```

After removing world-open SSH (0.0.0.0/0 and ::/0) from all offending security groups and triggering a rule evaluation, the managed rule **security-group-open-ssh-check** reports **COMPLIANT**. This confirms the control is enforced and the environment is back within policy.

38. AWS Config Compliance Results Across Multiple Rules



```
aws | Search  
CloudShell  
us-east-1 +  
> },  
> "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-0tv tarpja |  
+-----+-----+-----+-----+  
| Compliance |  
+-----+-----+-----+-----+  
| ComplianceType | INSUFFICIENT_DATA |  
+-----+-----+-----+-----+  
| ComplianceByConfigRules |  
+-----+-----+-----+-----+  
| ConfigRuleName |  
+-----+-----+-----+-----+  
| account-contact-details-configured-conformance-pack-0tv tarpja |  
+-----+-----+-----+-----+  
| Compliance |  
+-----+-----+-----+-----+  
| ComplianceType | INSUFFICIENT_DATA |  
+-----+-----+-----+-----+  
| ComplianceByConfigRules |  
+-----+-----+-----+-----+  
| ConfigRuleName |  
+-----+-----+-----+-----+  
| account-security-contact-configured-conformance-pack-0tv tarpja |
```

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.

39. Compliance Status for Conformance Pack Rules

Compliance	
ComplianceType	INSUFFICIENT_DATA
ComplianceByConfigRules	
ConfigRuleName	
account-security-questions-configured-conformance-pack-0tv tarpja	

Feedback

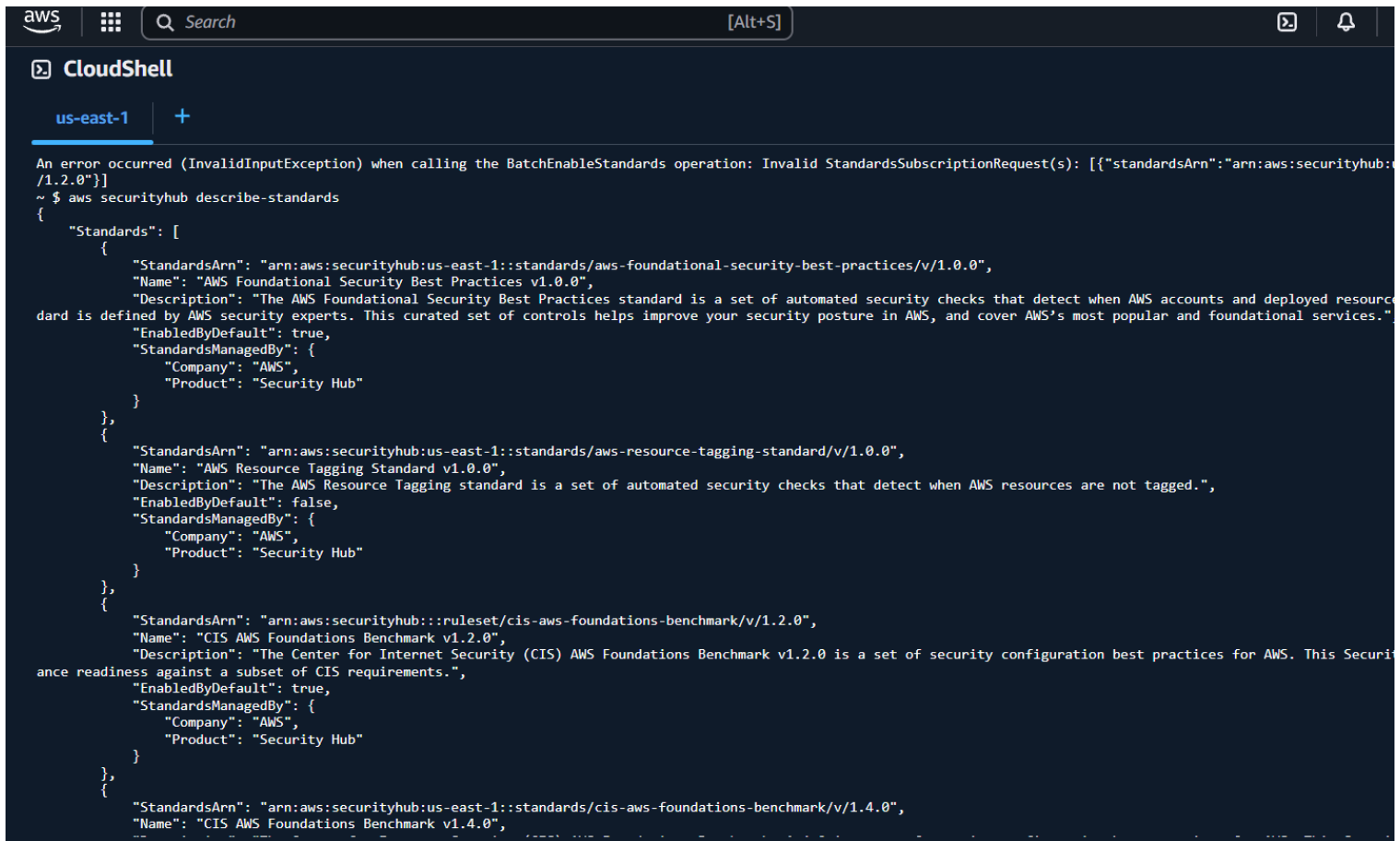
Here we view compliance results for rules deployed as part of a conformance pack. Each rule is evaluated and marked as COMPLIANT, NON_COMPLIANT, or INSUFFICIENT_DATA. This allows auditors to quickly determine which areas of the environment meet the requirements of the selected benchmark and which need remediation.

40. 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.

41. Available Security Hub Standards Listed



```
aws | [Grid Icon] | Q Search | [Alt+S] | [Close Icon] | [Bell Icon]
CloudShell
us-east-1 +
An error occurred (InvalidInputException) when calling the BatchEnableStandards operation: Invalid StandardsSubscriptionRequest(s): [{"standardsArn":"arn:aws:securityhub:us-east-1::standards/aws-foundational-security-best-practices/v/1.0.0"}]
~ $ 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 resources are not configured in a secure manner. This standard 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 Security Hub standard provides a baseline for AWS security posture and 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",
      "Description": "The Center for Internet Security (CIS) AWS Foundations Benchmark v1.4.0 is a set of security configuration best practices for AWS. This Security Hub standard provides a baseline for AWS security posture and readiness against a subset of CIS requirements.",
      "EnabledByDefault": true,
      "StandardsManagedBy": {
        "Company": "AWS",
        "Product": "Security Hub"
      }
    }
  ]
}
```

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.

42. Security Hub Standards Enabled and Status Verified

```
aws | [Search] [Alt+S]
CloudShell
us-east-1 +
~ $ 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 CIS AWS Foundations Benchmark standards in Security Hub and verify their subscription status. The CLI output confirms which benchmarks are active (READY) and which are still being provisioned (PENDING). This ensures that Security Hub is configured to continuously evaluate resources against recognised security baselines.

43. Security Hub Standards Status Confirmed

```
aws | [Search] [Alt+S]
CloudShell
us-east-1 +
~ $ 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"
    },
    {
      "StandardsSubscriptionArn": "arn:aws:securityhub:us-east-1:756716632322:subscription/aws-foundational-security-best-practices/v/1.0.0",
      "StandardsArn": "arn:aws:securityhub:us-east-1::standards/aws-foundational-security-best-practices/v/1.0.0",
      "StandardsInput": {},
      "StandardsStatus": "READY",
      "StandardsControlsUpdatable": "READY_FOR_UPDATES"
    },
    {
      "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": "READY_FOR_UPDATES"
    }
  ]
}
```

Using the AWS CLI, we verify the subscription status of multiple Security Hub standards. The output confirms that the CIS AWS Foundations Benchmarks (v1.2.0 and v1.4.0) and AWS Foundational Security Best Practices are enabled, with some in READY state and others still PENDING. This ensures Security Hub is actively aligned with recognised security frameworks for continuous compliance monitoring.

44. Security Hub Findings with Severity Metadata

```
aws securityhub get-findings
{
  "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": {
        "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-control/IAM.22/finding/9bea3d5f-78f8-49ea-afea-3ff6a95d4f07"
      }
    }
  ],
}
```

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.

45. 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 summarize counts by severity (CRITICAL/HIGH/MEDIUM/LOW/INFORMATIONAL). This gives a fast, board-level view of current risk. If the summary is empty in a fresh account, enable GuardDuty and create sample findings, then re-run.

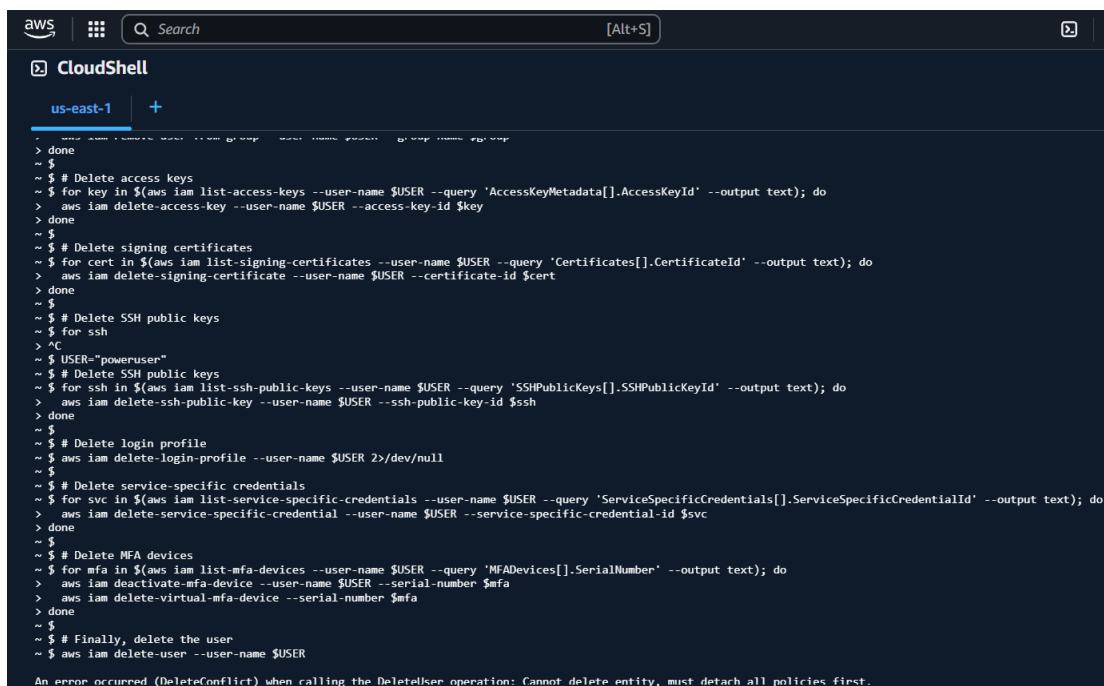
46. IAM User Cleanup and Remediation

```
~ $ 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.

47. IAM User Deprovisioning and Account Hardening



```
aws CloudShell
us-east-1 +
~ $
~ $ # 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
> ^C
~ $ USER="poweruser"
~ $ # Delete SSH public keys
~ $ for ssh in $(aws iam list-ssh-public-keys --user-name $USER --query 'SSHPublicKeys[].SSHPublicKeyId' --output text); do
>   aws iam delete-ssh-public-key --user-name $USER --ssh-public-key-id $ssh
> done
~ $
~ $ # Delete login profile
~ $ aws iam delete-login-profile --user-name $USER 2>/dev/null
~ $
~ $ # Delete service-specific credentials
~ $ for svc in $(aws iam list-service-specific-credentials --user-name $USER --query 'ServiceSpecificCredentials[].ServiceSpecificCredentialId' --output text); do
>   aws iam delete-service-specific-credential --user-name $USER --service-specific-credential-id $svc
> done
~ $
~ $ # Delete MFA devices
~ $ for mfa in $(aws iam list-mfa-devices --user-name $USER --query 'MFADevices[].SerialNumber' --output text); do
>   aws iam deactivate-mfa-device --user-name $USER --serial-number $mfa
>   aws iam delete-virtual-mfa-device --serial-number $mfa
> done
~ $
~ $ # Finally, delete the user
~ $ aws iam delete-user --user-name $USER

An error occurred (DeleteConflict) when calling the DeleteUser operation: Cannot delete entity, must detach all policies first.
```

Here we complete the remediation of a privileged IAM user (poweruser) by removing SSH keys, login profiles, service-specific credentials, MFA devices, and finally attempting to delete the user. These steps demonstrate how to deprovision insecure or unnecessary accounts, ensuring the environment adheres to least-privilege principles and reducing potential attack vectors.

48. IAM Cleanup Completed and Remaining Users Verified

```
~$ # 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 $policy_name
> done
~$ aws iam delete-user --user-name poweruser
~$ aws iam list-users --query 'Users[].UserName'
[
  "insecure-admin"
]
~$
```

49. Final Posture Check (One-Liner Health Snapshot)

```
}
~$ echo "CloudTrail:" && aws cloudtrail get-trail-status --name demoTrail --query '{IsLogging:IsLogging,LastDeliveryTime:LatestDeliveryTime}'
CloudTrail:
{
  "IsLogging": true,
  "LastDeliveryTime": "2025-09-03T08:31:02.255000+00:00"
}
~$ echo "Config:" && aws configservice describe-configuration-recorder-status --query 'ConfigurationRecordersStatus[0].{Recording:recording,LastStatus:lastStatus}'
Config:
{
  "Recording": true,
  "LastStatus": "SUCCESS"
}
~$ echo "SecurityHub Standards:" && aws securityhub get-enabled-standards --query 'StandardsSubscriptions[0].{Name:StandardsArn,Status:StandardsStatus}'
SecurityHub Standards:
[
  {
    "Name": "arn:aws:securityhub::ruleset/cis-aws-foundations-benchmark/v/1.2.0",
    "Status": "READY"
  },
  {
    "Name": "arn:aws:securityhub:us-east-1::standards/aws-foundational-security-best-practices/v/1.0.0",
    "Status": "READY"
  }
]
~$ echo "GuardDuty Detector:" && aws guardduty list-detectors --query DetectorIds[0]
GuardDuty Detector:
"32cc873474ffe40128086fac3be828f0"
~$
```

In the final step, we completed the deprovisioning of the poweruser account by removing all attached policies and deleting the user. A subsequent query of IAM users confirms that only insecure-admin remains. This validates the successful cleanup of unnecessary privileged accounts, a critical part of CSPM to reduce risk exposure.

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.

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.