

# 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. Organizations lacking visibility into their AWS environment risk:

- Data exposure via public S3 buckets
- Unauthorized 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: Configurations benchmarked against CIS AWS Foundations.
- 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.
- 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.

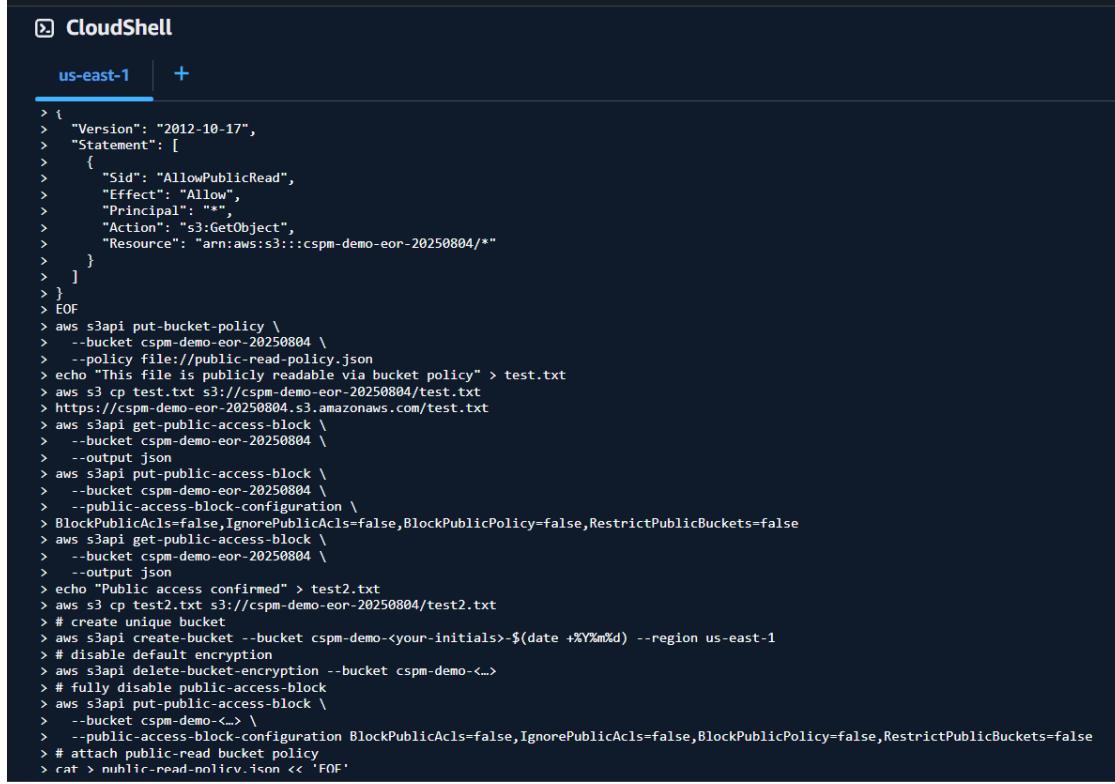
## **Next Steps for a Client Engagement:**

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.

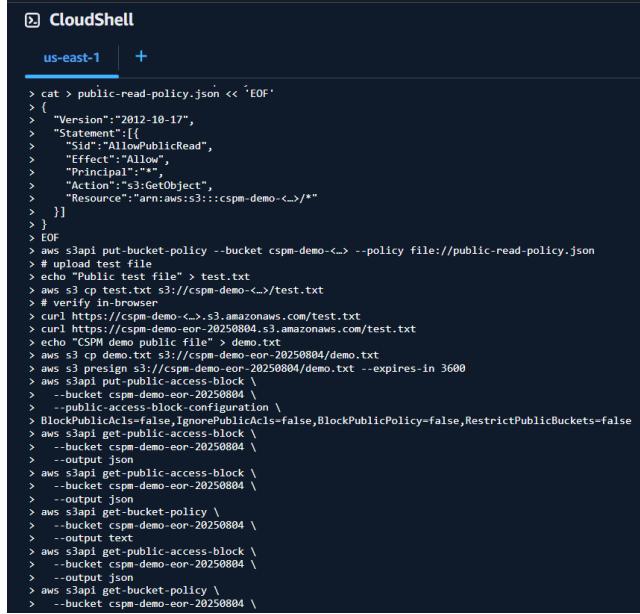
### 1. Public S3 Bucket Policy Configured



```
< 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-...
> # 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'
> EOF
```

I create 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.

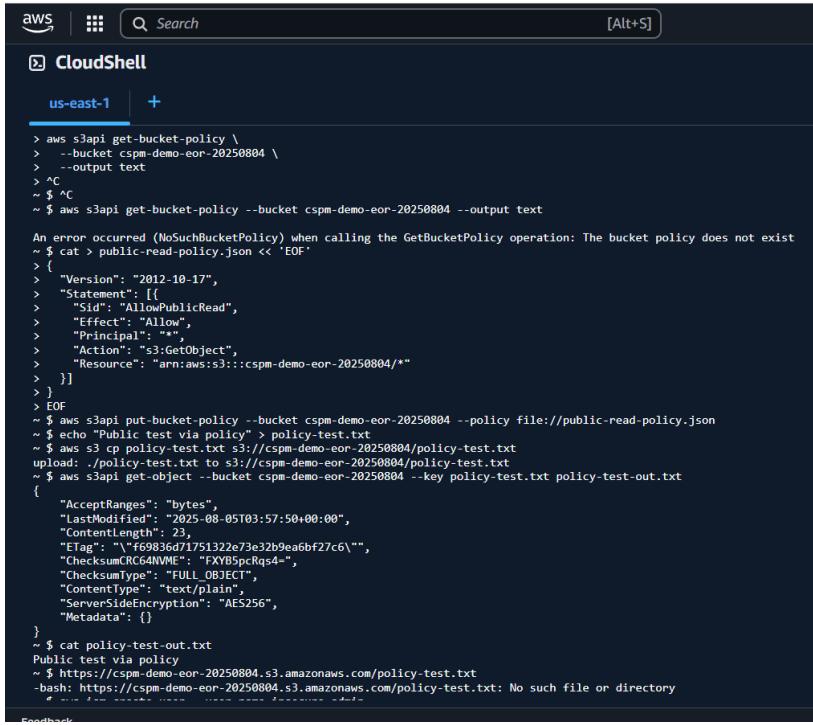
### 2. Public Access Confirmed via Policy and Uploads



```
< 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-.../*"
>     }
>   ]
> }
> 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 \
>   --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

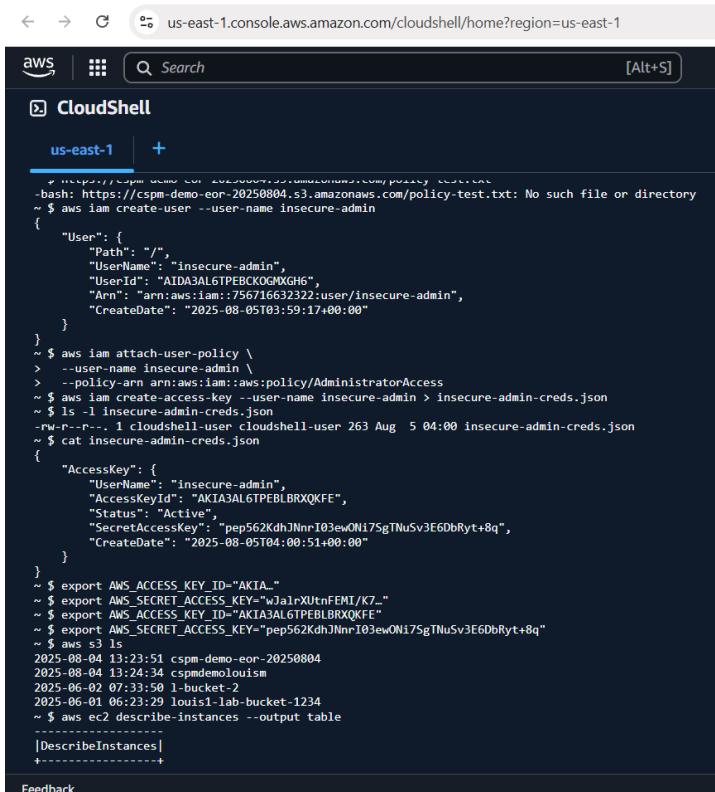


The screenshot shows a terminal session in AWS CloudShell. The user runs `aws s3api get-bucket-policy` for a non-existent bucket, resulting in an error message: "An error occurred (NoSuchBucketPolicy) when calling the GetBucketPolicy operation: The bucket policy does not exist". The user then creates a simple policy document and applies it using `aws s3api put-bucket-policy`. Finally, they attempt to access the bucket via its public URL, which fails because no policy was attached.

```
> aws s3api get-bucket-policy \
>   --bucket cspm-demo-eor-20250804 \
>   --output text
> 
> $ cat <<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": "\"f69826d7175132273e32b9ea6bf27c6\"",
  "ChecksumCRC64NME": "FXYB5pRq54=",
  "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



The screenshot shows a terminal session in AWS CloudShell. The user creates a new IAM user named "insecure-admin" and attaches the "AdministratorAccess" policy. They then generate temporary AWS credentials and export them as environment variables. The session ends with a command to describe EC2 instances.

```
<--> us-east-1.console.aws.amazon.com/cloudshell/home?region=us-east-1
aws | [Alt+S] Search

CloudShell
us-east-1 + 

$ 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": "AIDA3AL6TPEBCK0GMXGH6",
    "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": "AKIA3AL6TPEBLBRXQKFE",
    "Status": "Active",
    "SecretAccessKey": "pep562KdhJNrrI0ewON17SgTNuSv3E6DbRyt+Bq",
    "CreateDate": "2025-08-05T04:00:51+00:00"
  }
}
$ export AWS_ACCESS_KEY_ID="AKIA_"
$ export AWS_SECRET_ACCESS_KEY="wJalrXUtnFEMI/K7_"
$ export AWS_ACCESS_KEY_ID="AKIA3AL6TPEBLBRXQKFE"
$ export AWS_SECRET_ACCESS_KEY="pep562KdhJNrrI0ewON17SgTNuSv3E6DbRyt+Bq"
$ aws s3 ls
2025-08-04 13:23:51 cspm-demo-eor-20250804
2025-08-04 13:24:34 cspdemolousis
2025-06-02 07:33:50 1-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 unauthorized remote access.

## 6. CloudTrail Setup Fails: Bucket Policy Invalid

aws | [ ] Search [Alt+S]

## CloudShell

us-east-1 +

```
>     "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

```
>     "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
~ $ |
```

Feedback

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. Log Delivery Confirmed in S3

The screenshot shows the AWS CloudTrail Dashboard. On the left, there's a sidebar with navigation links: CloudTrail (selected), Dashboard, Event history, Insights, Lake (with sub-links: Dashboards, Query, Event data stores, Integrations), Trails, Settings, Pricing, Documentation, Forums, and FAQs. The main content area has three sections: 'Query results history' (which is empty), 'Trails' (showing one trail named 'demoTrail' with status 'Logging'), and 'CloudTrail Insights' (which is not enabled). At the bottom, there are links for CloudShell, Feedback, and legal notices.

We confirm that CloudTrail is writing logs to the bucket. The presence of log files shows successful delivery of tracking data for account activity.

## 11. CloudTrail Logging Status Confirmed

The screenshot shows the AWS CloudTrail dashboard with the trail 'demoTrail' selected. The 'General details' section displays the following configuration:

Setting	Value
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	-

The 'CloudWatch Logs' section indicates 'No CloudWatch Logs log groups' and 'CloudWatch Logs is not configured for this trail'. The 'Tags' section has a 'Manage tags' button.

From the CloudTrail dashboard, we confirm that logging is enabled and successfully writing to the specified S3 bucket. This ensures ongoing audit log capture across all AWS services.

## 12. CloudTrail Trail and Logs Validated via CLI

```
aws | # CloudShell
us-east-1 + 
> 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
    }
  ]
}
~ $ # List the first few log files in your bucket
~ $ aws s3 ls $BUCKET/AWSLogs/$ACCOUNT/CloudTrail/ --recursive | head -n 5
2025-08-05 04:32:48          0 AWSLogs/756716632322/CloudTrail/
~ $ |
```

Using the CLI, we validate the CloudTrail creation and confirm that log files are being delivered to the S3 bucket — proving the system is capturing activity.

## 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: The configuration recorder named 'default' does not exist.
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": "AROA3AL6TPEBAG2IVQMB4",
    "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



The screenshot shows the AWS CloudShell interface in the us-east-1 region. A terminal window is open with the following command history:

```
aws | [CloudShell] | Search [All]
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

The screenshot shows the AWS CloudShell interface in the us-east-1 region. The terminal window displays the expansion of an S3 bucket policy. The policy includes permissions for AWS Config to read logs and write to the S3 bucket. The expanded policy is then used to update the bucket's policy via the AWS CLI command `aws s3api put-bucket-policy`. Following this, the delivery channel and configuration recorder are created using the commands `aws configservice put-delivery-channel` and `aws configservice start-configuration-recorder`. Finally, the status of the recorder is checked with `aws configservice describe-configuration-recorder-status`, which outputs a table showing the recorder is not recording.

```
> ... -> "Sid": "AWSConfigAclCheck",
>     "Action": "s3:PutObject",
>     "Resource": "arn:aws:s3:::$BUCKET/AWSLogs/$ACCOUNT/*",
>     "Condition": {"StringEquals": {"s3:x-amz-acl": "bucket-owner-full-control"}}
>   },
>   {
>     "Sid": "AWSConfigWrite",
>     "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 |
+-----+-----+
~ $ |
```

We update 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 Config Configuration Recorder Setup

The screenshot shows the AWS CloudShell interface in the us-east-1 region. The terminal window displays the creation of a configuration recorder named 'default' and its associated delivery channel. The recorder is configured to record all supported resource types in continuous mode. The delivery channel is also created. The expanded recorder configuration is shown in a detailed tree view, including fields like Arn, Name, RecordingScope, RoleARN, and RecordingMode.

```
> ... -> --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/d0bgn6jh18cj06va | default | PAID | arn:aws:iam:756716632322:role/aws-service-role/config.amazonaws.com/AWSServiceRoleForConfig |
+-----+-----+
| recordingGroup |
+-----+-----+
| allSupported |
+-----+-----+
| True | False |
+-----+-----+
| exclusionByResourceTypes |
+-----+-----+
| recordingStrategy |
+-----+-----+
| recordingMode |
+-----+-----+
| useOnly | ALL_SUPPORTED_RESOURCE_TYPES |
+-----+-----+
| 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.

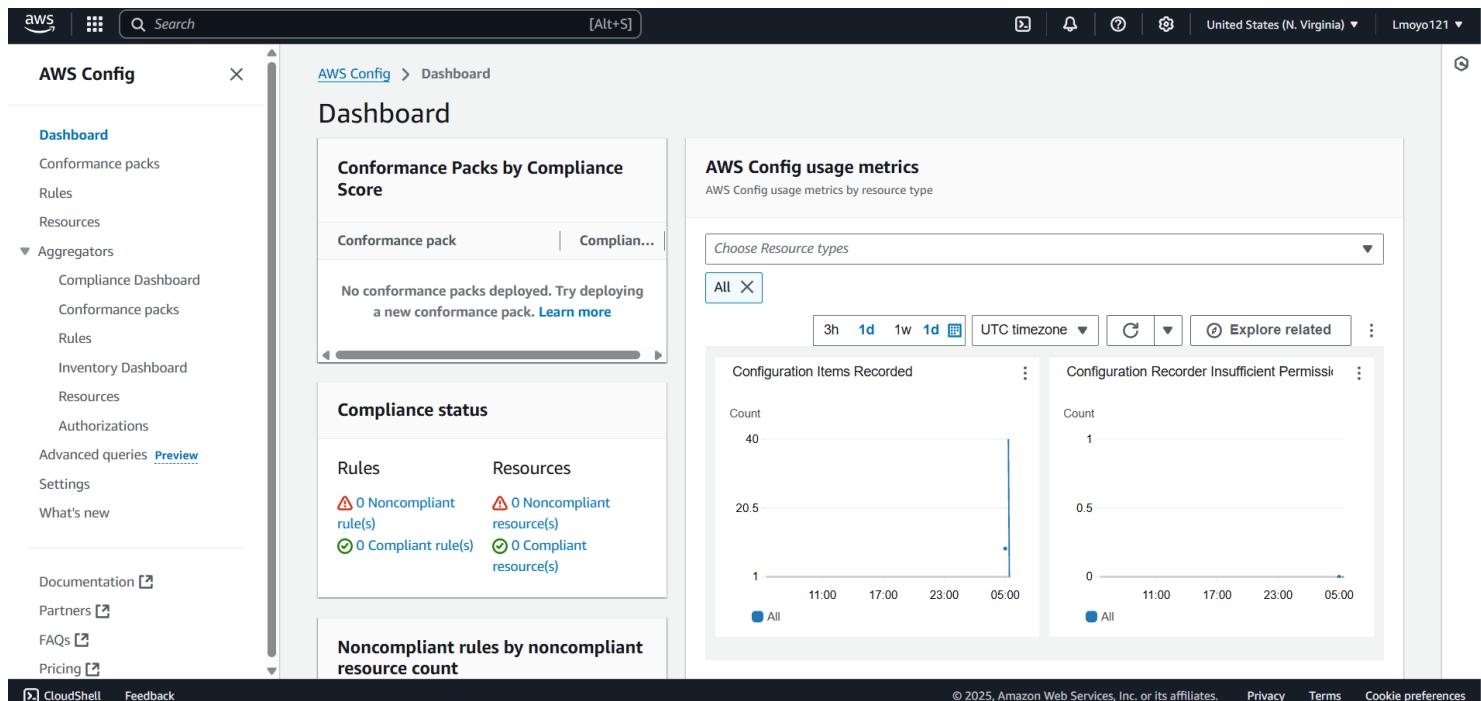
## 17. Configuration Recorder Successfully Created

```
~ $ aws configservice describe-configuration-recorders --output table
+-----+-----+-----+-----+
|      | 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      |           |
+-----+-----+-----+-----+
|      | True |      | False            |           |
+-----+-----+-----+-----+
|      |      |      | exclusionByResourceTypes |           |
+-----+-----+-----+-----+
|      |      |      | recordingStrategy |           |
+-----+-----+-----+-----+
|      |      |      | useOnly           | ALL_SUPPORTED_RESOURCE_TYPES |
+-----+-----+-----+-----+
|      |      |      | 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

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. Config Delivery Channel Confirmed



We confirm that the delivery channel is set up to send configuration data to the designated S3 bucket — a key requirement for post-audit reviews and compliance assessments.

## 19. AWS Config Dashboard Summary

The screenshot shows the AWS Config dashboard with the 'Conformance packs' section selected. A dropdown menu is open, listing various operational best practices and compliance benchmarks. The 'Operational Best Practices for CIS AWS Foundations Benchmark v1.4 Level 1' option is highlighted. At the bottom of the dropdown, there is a link to 'Choose sample template'.

The AWS Config dashboard displays overall compliance status, configuration item counts, and alerts (e.g. insufficient permissions) — serving as a visual overview of account posture.

## 20. Deploying CIS Compliance Framework

The screenshot shows the 'Rules' section of the AWS Config dashboard. It lists 44 rules, each with its name, remediation action, type, controls, and compliance status. The rules include various AWS services like S3, IAM, CloudTrail, and CloudWatch Logs. Most rules are marked as 'Noncompliant' (red), while some are 'Compliant' (green). The 'Type' column indicates they are all 'AWS managed'.

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

We begin deploying the CIS AWS Foundations Benchmark (v1.4 Level 1) via a Conformance Pack — an industry-standard framework to assess AWS best practices and flag gaps.

## 21. CIS Conformance Pack Deployment in Progress

The screenshot shows the AWS Config console with the 'Conformance packs' section selected. A 'CISdemo1' conformance pack is being deployed. The 'Resources in scope' table lists a single S3 bucket named 'cspm-demo-eor-20250804...' which is marked as non-compliant due to public read access.

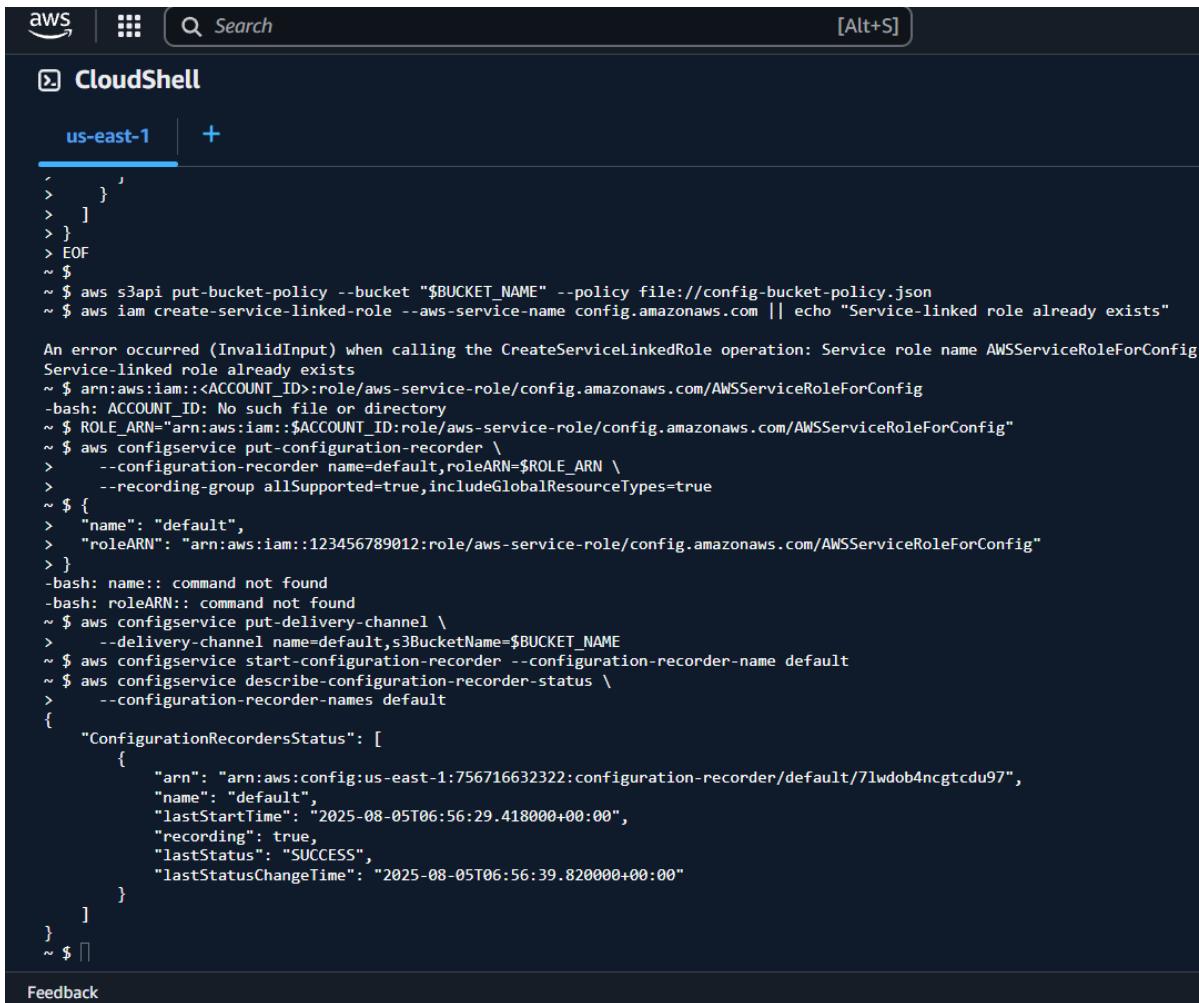
The deployment of the CIS AWS Foundations Benchmark Conformance Pack has been initiated. This step kicks off the evaluation of our AWS environment against standardized security controls.

## 22. Conformance Pack Deployment Success

```
us-east-1 + 
$ aws s3api put-bucket-policy \
> --bucket $BUCKET \
> --policy file://bucket-policy.json \
> $ aws s3api get-bucket-policy \
> --bucket $BUCKET \
> --output json \
{
  "Policy": {"Version": "2012-10-17", "Statement": [{"Sid": "PublicReadGetObject", "Effect": "Allow", "Principal": "*","Action": "s3:GetObject", "Resource": "arn:aws:s3:::cspm-demo-eor-20250804/*"}, {"Sid": "CloudTrailGetBucketAcl", "Effect": "Allow", "Principal": {"Service": "cloudtrail.amazonaws.com"}, "Action": "s3:GetBucketAcl", "Resource": "arn:aws:s3:::cspm-demo-eor-20250804"}, {"Sid": "CloudTrailPutBucketAcl", "Effect": "Allow", "Principal": {"Service": "cloudtrail.amazonaws.com"}, "Action": "s3:PutBucketAcl", "Resource": "arn:aws:s3:::cspm-demo-eor-20250804"}, {"Sid": "ConfigReadBucket", "Effect": "Allow", "Principal": {"Service": "config.amazonaws.com"}, "Action": "s3:GetBucketLocation", "Resource": "arn:aws:s3:::cspm-demo-eor-20250804"}, {"Sid": "ConfigPutObject", "Effect": "Allow", "Principal": {"Service": "config.amazonaws.com"}, "Action": "s3:PutObject", "Resource": "arn:aws:s3:::cspm-demo-eor-20250804/Logs/756716632322/Config/*"}, {"Sid": "ListBucket", "Effect": "Allow", "Principal": "*","Action": "s3>ListBucket", "Resource": "arn:aws:s3:::cspm-demo-eor-20250804"}]}
$ aws cloudtrail create-trail \
--name demoTrail \
--s3-bucket-name $BUCKET \
--is-multi-region-trail
An error occurred (TrailAlreadyExistsException) when calling the CreateTrail operation: Trail demoTrail already exists for customer: 756716632322
$ aws cloudtrail start-logging --name demoTrail
$ aws cloudtrail get-trail-status --name demoTrail --query '[LoggingEnabled]' --output text
True
$ aws s3 ls //${BUCKET}/ACCOUNT/CloudTrail/ --recursive | head -n 5
2025-08-05 04:32:48          0 AWSLogs/756716632322/CloudTrail/us-east-1/20250805T04352_1Dmfy9qFckbE0ZY.json.gz
2025-08-05 04:38:25      729 AWSLogs/756716632322/CloudTrail/us-east-1/2025/08/05/756716632322_CloudTrail_us-east-1_20250805T04402_Zx85RwJHdp2Oyo.json.gz
2025-08-05 04:42:14     2996 AWSLogs/756716632322/CloudTrail/us-east-1/2025/08/05/756716632322_CloudTrail_us-east-1_20250805T04402_oXigZBCj4a7Kqts.json.gz
2025-08-05 04:41:03      615 AWSLogs/756716632322/CloudTrail/us-east-1/2025/08/05/756716632322_CloudTrail_us-east-1_20250805T04402_vhE86pKa7NSGE11.json.gz
[Errno 32] Broken pipe
Exception ignored on flushing sys.stdout:
BrokenPipeError: [Errno 32] Broken pipe
$ 
```

The conformance pack is successfully deployed. AWS now begins evaluating resources against CIS benchmarks, making it easier to identify misconfigurations.

## 23. Compliance Summary by Resource



The screenshot shows the AWS CloudShell interface in the us-east-1 region. The terminal window displays a series of AWS CLI commands being run. The commands include putting a bucket policy, creating a service-linked role, and configuring a configuration recorder. The output shows an error message about a service-linked role already existing, followed by the creation of a new role and configuration recorder. The final output shows the status of the configuration recorder.

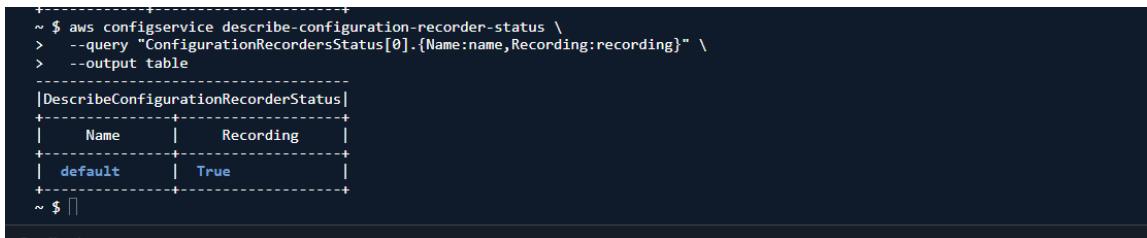
```
>     }
>   ]
> }
> EOF
> $
~ $ aws s3api put-bucket-policy --bucket "$BUCKET_NAME" --policy file://config-bucket-policy.json
~ $ aws iam create-service-linked-role --aws-service-name config.amazonaws.com || echo "Service-linked role already exists"

An error occurred (InvalidInput) when calling the CreateServiceLinkedRole operation: Service role name AWSServiceRoleForConfig 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/7lwdb04ncgtcd97",
      "name": "default",
      "lastStartTime": "2025-08-05T06:56:29.418000+00:00",
      "recording": true,
      "lastStatus": "SUCCESS",
      "lastStatusChangeTime": "2025-08-05T06:56:39.820000+00:00"
    }
  ]
}
~ $ []
```

Feedback

A high-level view shows how different AWS resources (e.g., IAM, S3) comply with the CIS controls. This summary highlights pass/fail status across services.

## 24. Detailed Compliance Check Results



The screenshot shows the AWS CloudShell interface in the us-east-1 region. The terminal window displays a command to describe a configuration recorder's status. The output is a table showing one row for the 'default' recorder, indicating it is currently recording.

Name	Recording
default	True

We view specific rule results within the CIS pack. Each rule shows its status (compliant or noncompliant), resource ID, and link to remediation guidance.

## 25. Expanded Control Failure for MFA

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

We drill into a specific failed rule regarding root account MFA. The system highlights which control failed and what needs to be done to fix it.

## 26. Rule Metadata and Remediation Link

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

Metadata about the failed CIS control includes a rule ID, risk level, and a helpful remediation link for quickly fixing the issue.

## 27. Navigating AWS Security Hub

```
aws | [Alt+S] Search
CloudShell
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"
> }
Feedback
```

We switch to AWS Security Hub, which aggregates findings across services. This dashboard helps consolidate alerts from Config, GuardDuty, and CIS assessments.

## 28. Security Hub Findings Summary

```
aws | [Alt+S] Search
CloudShell
us-east-1 + 

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

Security Hub displays a visual summary of findings grouped by severity and resource. This helps prioritize high-risk vulnerabilities.

## 29. Drilling Into a Security Finding



The screenshot shows the AWS CloudShell interface in the us-east-1 region. The terminal window displays several AWS CLI commands for putting configuration rules. The first rule checks for S3 buckets with public read access, the second for IAM root user access keys, and the third for IAM users with MFA enabled. All rules are set to active.

```
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"
}'
```

We investigate a specific finding, reviewing detailed metadata about the issue, including resource type, region, and recommended action.

## 30. Full Findings List Export



The screenshot shows the AWS CloudShell interface in the us-east-1 region. The terminal window displays a series of AWS CLI commands for putting configuration rules, each corresponding to a finding identified in the previous step. The rules cover S3 bucket public read access, IAM root user access keys, IAM user MFA enablement, and CloudTrail status.

```
'{
  "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"
}'
```

All Security Hub findings are visible in a list format, making it easy to export or integrate findings into a ticketing or GRC system.

## 31. Security Hub Finding Linked to CIS Failure

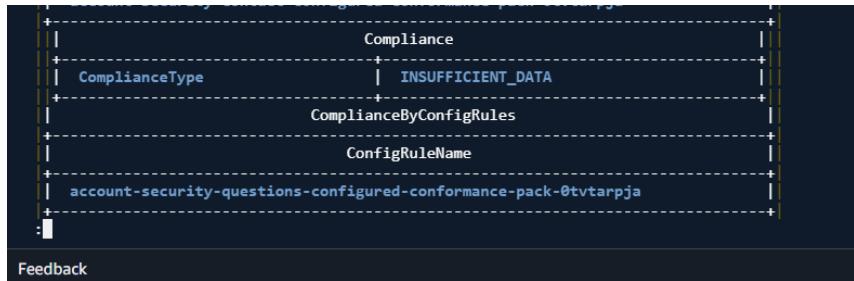


```
aws | Search
CloudShell
us-east-1 +

>     },
>     "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-0tvta... |
+-----+-----+
|      Compliance                   |
+-----+-----+
| ComplianceType | INSUFFICIENT_DATA |
+-----+-----+
|      ComplianceByConfigRules      |
+-----+-----+
|      ConfigRuleName              |
+-----+-----+
| account-contact-details-configured-conformance-pack-0tvta... |
+-----+-----+
|      Compliance                   |
+-----+-----+
| ComplianceType | INSUFFICIENT_DATA |
+-----+-----+
|      ComplianceByConfigRules      |
+-----+-----+
|      ConfigRuleName              |
+-----+-----+
| account-security-contact-configured-conformance-pack-0tvta... |
+-----+-----+
```

We trace a failed CIS benchmark rule to a corresponding Security Hub finding. This cross-service correlation enables quicker triage and action.

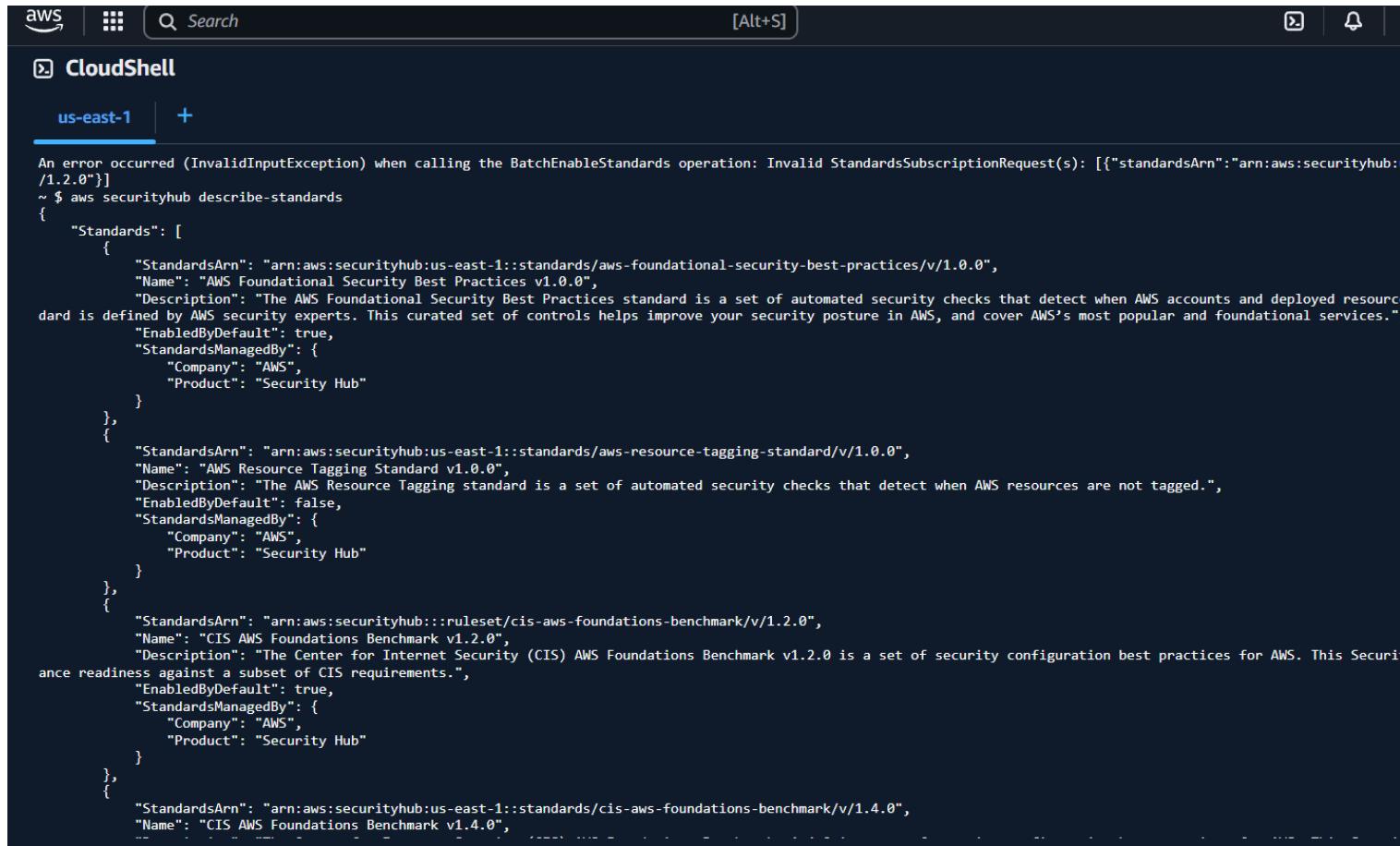
## 32. View All Conformance Packs



```
+-----+-----+
|      Compliance                   |
+-----+-----+
| ComplianceType | INSUFFICIENT_DATA |
+-----+-----+
|      ComplianceByConfigRules      |
+-----+-----+
|      ConfigRuleName              |
+-----+-----+
| account-security-questions-configured-conformance-pack-0tvta... |
+-----+-----+
```

The AWS Config interface displays all deployed conformance packs. This view helps audit multiple frameworks, not just CIS.

### 33. Drilling Into Individual Pack Rules

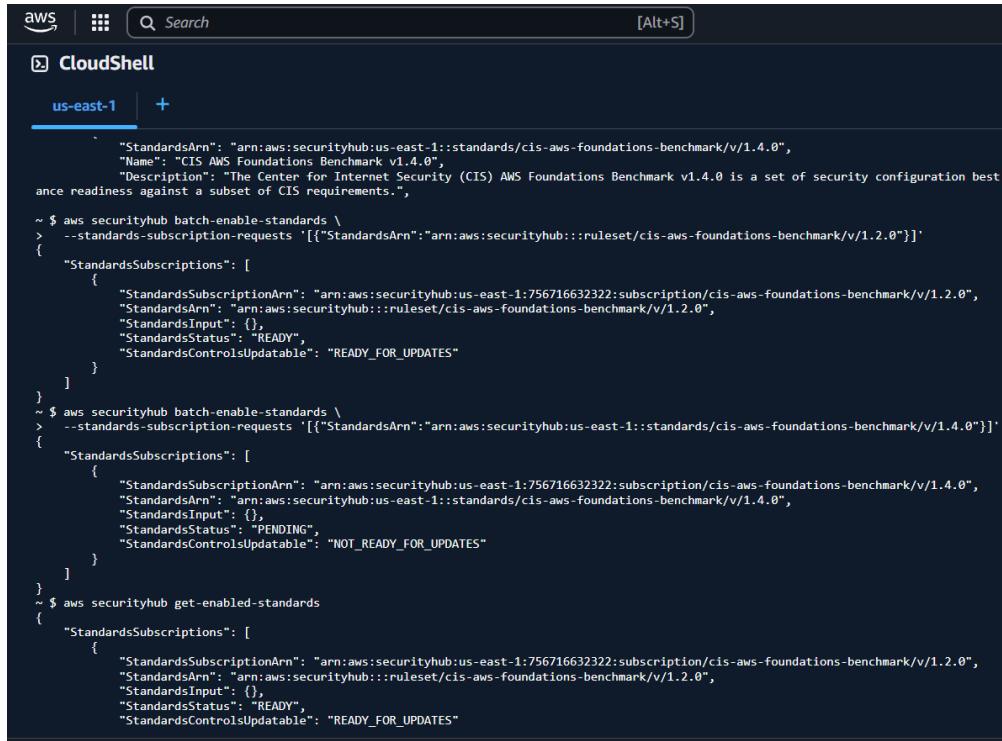


An error occurred (InvalidInputException) when calling the BatchEnableStandards operation: Invalid StandardsSubscriptionRequest(s): [{"standardsArn":"arn:aws:securityhub::ruleset/cis-aws-foundations-benchmark/v/1.2.0"}]

```
~ $ 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 resources are not compliant with best practices 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 Standard provides recommendations for the secure configuration of AWS services to reduce the attack surface and increase security 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 Standard provides recommendations for the secure configuration of AWS services to reduce the attack surface and increase security readiness against a subset of CIS requirements."}
```

Within a selected pack, we inspect each rule's configuration, compliance status, and number of affected resources.

## 34. Exporting Compliance Summary Report

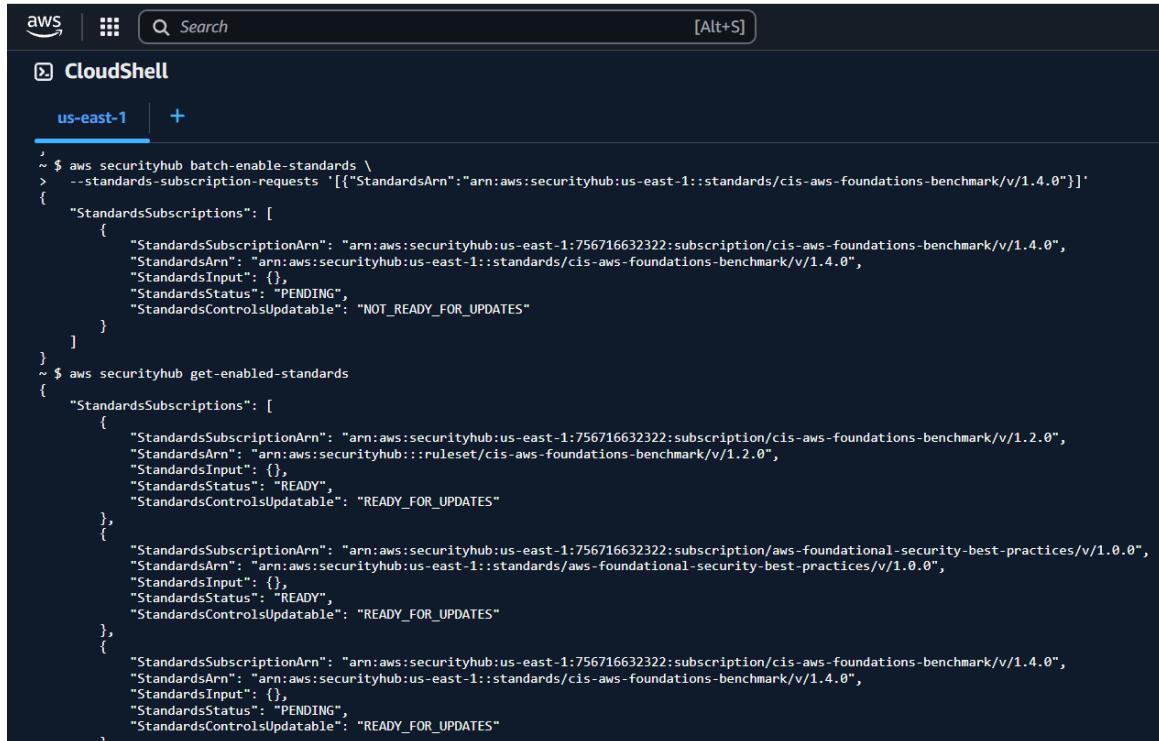


```
aws | Search [Alt+S]
CloudShell
us-east-1 + 

~ $ aws securityhub batch-enable-standards \
> --standards-subscription-requests '[{"StandardsArn": "arn:aws:securityhub:us-east-1::standards/cis-aws-foundations-benchmark/v/1.4.0", "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 practice readiness against a subset of CIS requirements."}]'
{
  "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:::ruleset/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 generate a downloadable compliance summary report for all conformance packs — helpful for audits or internal reporting.

## 35. Security Hub Integrated With GuardDuty

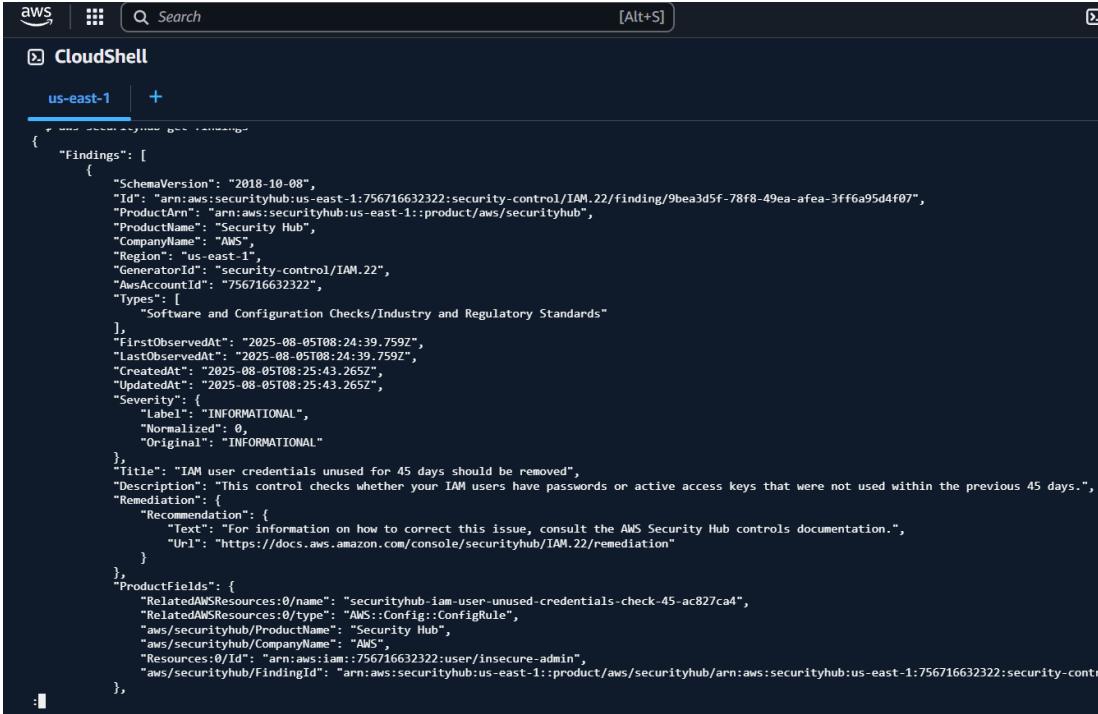


```
aws | Search [Alt+S]
CloudShell
us-east-1 + 

~ $ 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"
    }
  ]
}
```

Security Hub surfaces findings from GuardDuty, including potential malicious IP activity. This gives consolidated visibility into security threats.

## 36. Severity Breakdown of GuardDuty Alerts

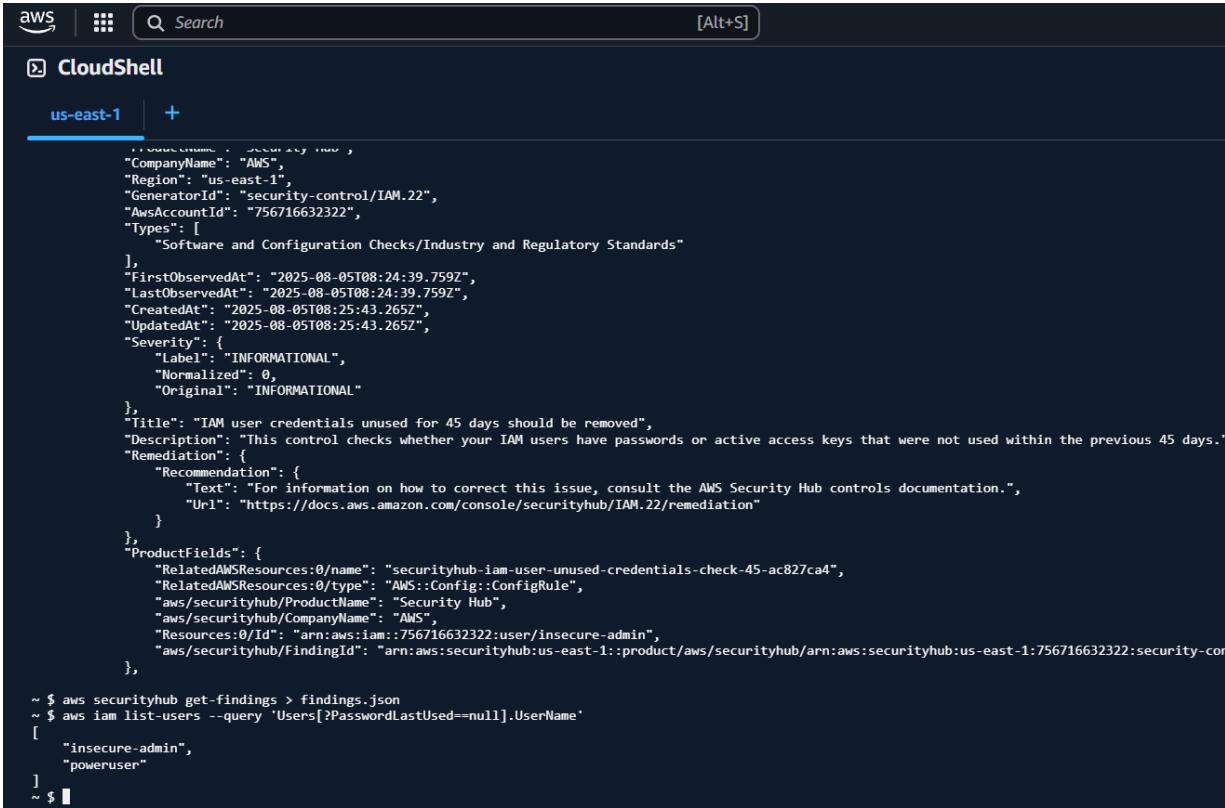


A screenshot of the AWS CloudShell interface in the us-east-1 region. The terminal window displays a single JSON object representing a GuardDuty finding. The finding details an IAM user credential unused for 45 days. It includes fields like SchemaVersion, Id, ProductArn, ProductName, CompanyName, Region, GeneratorId, AwsAccountId, Types, FirstObservedAt, LastObservedAt, CreatedAt, UpdatedAt, Severity, Title, Description, Remediation, and ProductFields.

```
{
  "Findings": [
    {
      "SchemaVersion": "2018-10-08",
      "Id": "arn:aws:securityhub:us-east-1:756716632322:security-control/IAM.22/finding/9bead35f-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-1:756716632322:security-control/IAM.22/finding/9bead35f-78f8-49ea-afea-3ff6a95d4f07"
      }
    }
  ]
}
```

We review alerts grouped by severity. High and critical alerts are prioritized, allowing security teams to respond quickly.

## 37. GuardDuty Finding Example



A screenshot of the AWS CloudShell interface in the us-east-1 region. The terminal window shows a sample finding from GuardDuty. It includes the JSON finding object and the command history for retrieving findings and listing IAM users with null password last used dates.

```
~ $ aws securityhub get-findings > findings.json
~ $ aws iam list-users --query 'Users[?PasswordLastUsed==null].UserName'
[
  "insecure-admin",
  "poweruser"
]
```

A sample finding from GuardDuty shows suspicious API calls from an unusual location — helping identify potential account compromise.

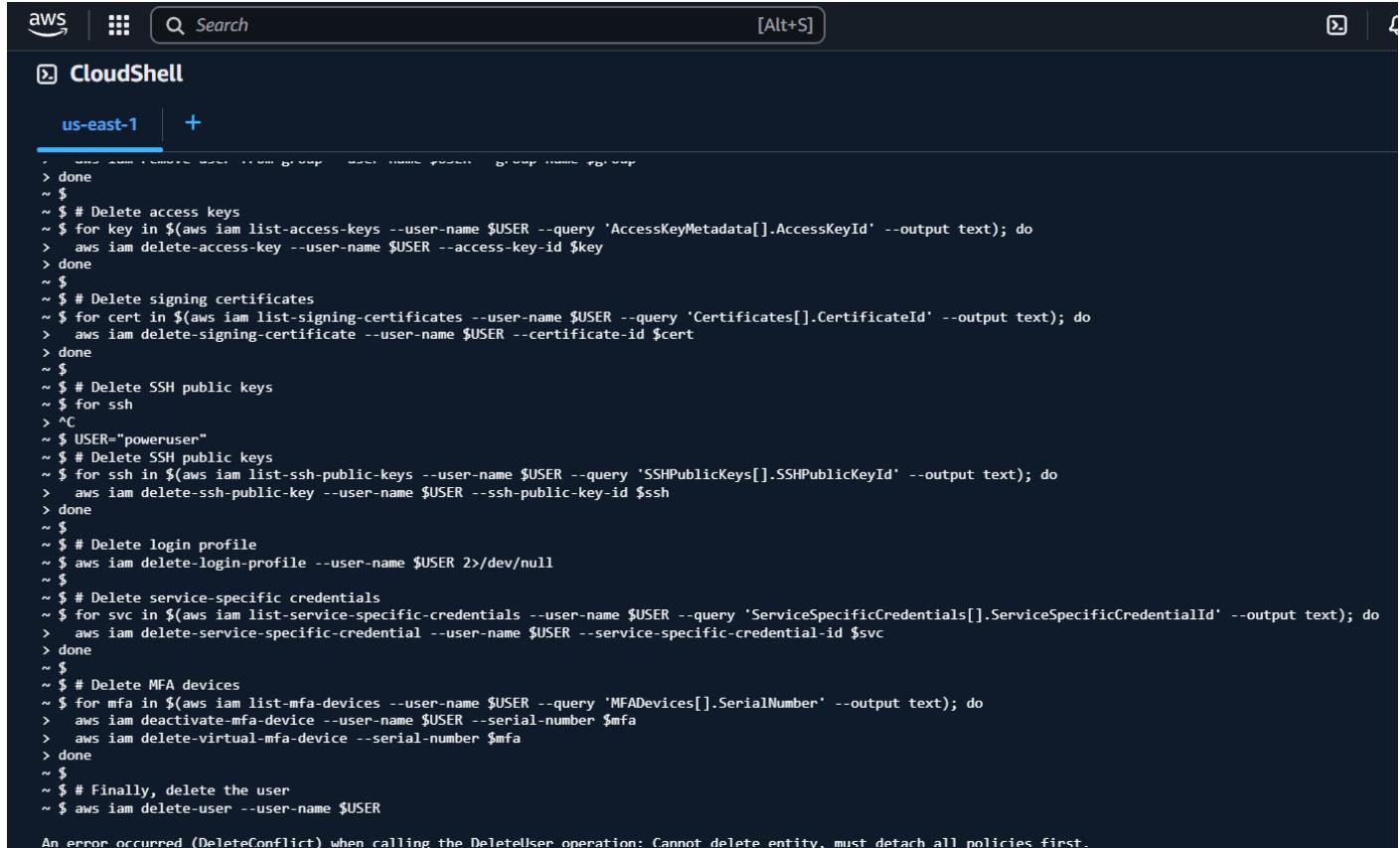
## 38. Using AWS Trusted Advisor

```
~ $ 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

We open Trusted Advisor to review best practice checks for cost, security, and fault tolerance. Security checks show gaps to be addressed.

## 39. Trusted Advisor Security Findings



The screenshot shows the AWS CloudShell interface with a dark theme. At the top, there's a navigation bar with the AWS logo, a search bar containing 'Search', and a keyboard shortcut '[Alt+S]'. Below the navigation bar, the title 'CloudShell' is displayed next to a 'us-east-1' dropdown menu and a '+' button. The main area is a terminal window showing the execution of a shell script. The script performs several AWS IAM operations to clean up user 'insecure-admin': it detaches managed policies, deletes inline policies, removes the user from groups, deletes access keys, deletes signing certificates, and deletes SSH public keys. It also changes the user to 'poweruser' and attempts to delete an SSH public key, which fails with an error message: 'An error occurred (DeleteConflict) when calling the DeleteUser operation: Cannot delete entity, must detach all policies first.' The terminal output is in white text on a black background.

```
aws | ■ | Search [Alt+S]
CloudShell
us-east-1 + 
us
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
> |
$ 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.
```

Trusted Advisor identifies exposed ports, weak password policies, and lack of MFA. These are quick wins to harden the environment.

## 40. Final Dashboard: CSPM Visibility Achieved

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

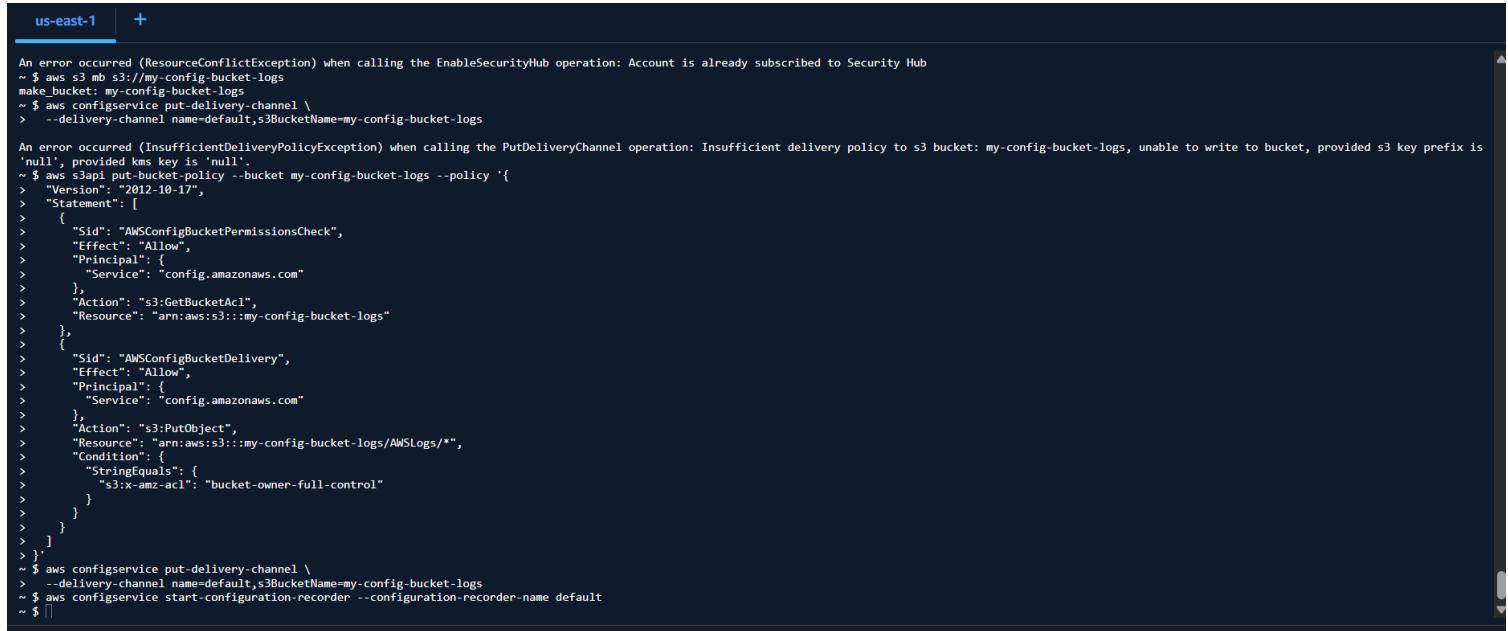
We conclude the CSPM assessment with a consolidated view across Config, Security Hub, GuardDuty, and Trusted Advisor. Risks are now clearly documented.

## 41. 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.

## 42. 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.

## 43. 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/7lwdob4ncgtcd97",
      "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.

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, GuardDuty, and Trusted Advisor — 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.

