

Summer Internship Report

Name: Dharvi Patel(17IT067)

AWS Continuous Compliance

IT compliance they think of spreadsheets, documents, audits, and generally, how “compliance” slows down the velocity of delivering value to end users. With AWS, however, everything is accessible via an API, and as a result, compliance can be treated as a code asset just like any other part of the software system. You can version, test, codify, monitor, and run compliance continuously. By doing this, you can ensure that all of your AWS infrastructure is always compliant with the control directives that ensure adherence to compliance regimes and good engineering practices.

learn how to use AWS services that provide the ability to define compliance as code, including AWS Config Rules, AWS Lambda, Amazon CloudWatch, AWS cloudTrail, Security Hub, etc.

AWS services :

AWS Config:



AWS Config is a service that enables you to assess, audit, and evaluate the configurations of your AWS resources. Config continuously monitors and records your AWS resource configurations and allows you to automate the evaluation of recorded configurations against desired configurations. With Config, you can review changes in configurations and relationships between AWS resources, dive into detailed resource configuration histories, and determine your overall compliance against the configurations specified in your internal guidelines. This enables you to simplify

compliance auditing, security analysis, change management, and operational troubleshooting.

CloudTrail:

AWS CloudTrail is a service that enables governance, compliance, operational auditing, and risk auditing of your AWS account. With CloudTrail, you can log, continuously monitor, and retain



troubleshooting.

account activity related to actions across your AWS infrastructure. CloudTrail provides event history of your AWS account activity, including actions taken through the AWS Management Console, AWS SDKs, command line tools, and other AWS services. This event history simplifies security analysis, resource change tracking, and troubleshooting. In addition, you can use CloudTrail to detect unusual activity in your AWS accounts. These capabilities help simplify operational analysis and

Security Hub:



AWS Security Hub gives you a comprehensive view of your high-priority security alerts and security posture across your AWS accounts. There are a range of powerful security tools at your disposal, from firewalls and endpoint protection to vulnerability and compliance scanners. But oftentimes this leaves your team switching back-and-forth between these tools to deal with hundreds, and sometimes thousands, of security alerts every day. With Security Hub, you now have a single place that aggregates, organizes, and prioritizes your security alerts, or findings, from multiple AWS services, such as Amazon GuardDuty, Amazon Inspector, Amazon Macie, AWS Identity and Access Management (IAM) Access Analyzer, and AWS Firewall Manager, as well as from AWS Partner solutions. AWS Security Hub continuously monitors your environment using automated security checks based on the AWS best practices and industry standards that your organization follows. You can also take action on these security findings by investigating them in Amazon Detective or by using Amazon CloudWatch Event rules to send the findings to ticketing, chat, Security Information and Event Management (SIEM), Security Orchestration Automation and Response (SOAR), and incident management tools or to custom remediation playbooks. AWS Security Hub in just a few clicks in the Management Console and once enabled, Security Hub will begin aggregating and prioritizing findings and conducting security checks.

AWS Lambda:

AWS Lambda lets you run code without provisioning or managing servers. You pay only for the compute time you consume.



With Lambda, you can run code for virtually any type of application or backend service - all with zero administration. Just upload your code and Lambda takes care of everything required to run and scale your code with high availability. You can set up your code to automatically trigger from other AWS services or call it directly from any web or mobile app.

Study RDK and check AWS Config

First of all install the rdk using pip command. Then you enter through command access key and secret access key and region to access the create rule from AWS.

```
C:\Users\hp>rdk -k AKIAICHWARSG3KWVE4IA -s mpIGA/SmX3v250WjxC0/nfzRQTTH0shQQqbntd6p -r us-east-1 init
Running init!
Found Bucket: config-bucket-520477949117
Creating IAM role config-role
Waiting for IAM role to propagate
Creating delivery channel to bucket config-bucket-520477949117
Config Service is ON
Config setup complete.
Creating Code bucket config-rule-code-bucket-520477949117-us-east-1
```

In this step you create the rule locally this rule is store in your directory.

```
C:\Users\hp>rdk create MyRule --runtime python3.6 --resource-types AWS::EC2::Instance
Running create!
Local Rule files created.
```

In this step test the rule locally and debug the rule and get the how many numbers of error in the rule.

```
C:\Users\hp>rdk test-local MyRule
Running local test!
Testing MyRule
Looking for tests in C:\Users\hp\MyRule
MyRule_test.py
Debug!
<unittest.suite.TestSuite tests=[<unittest.suite.TestSuite tests=[<MyRule_test.ComplianceTest testMethod=test
<unittest.suite.TestSuite tests=[<MyRule_test.TestStsErrors testMethod=test_sts_access_denied>, <MyRule_test
ors testMethod=test_sts_unknown_error>]]>]
test_sample (MyRule_test.ComplianceTest) ... ok
test_sts_access_denied (MyRule_test.TestStsErrors) ... ok
test_sts_unknown_error (MyRule_test.TestStsErrors) ... ok
```

In this step modify the rule locally and then you can deploy the rule into AWS.

```
C:\Users\hp>rdk modify MyRule --runtime python2.7 --maximum-frequency TwentyFour_Hours
Running modify!
Modified Rule 'MyRule'. Use the `deploy` command to push your changes to AWS.

C:\Users\hp>
```

In this step you can deploy the rule use bellow command.

```
C:\Users\hp>rdk deploy MyRule
Running deploy!
Found Custom Rule.
Zipping MyRule
Uploading MyRule
Upload complete.
Creating CloudFormation Stack for MyRule
Waiting for CloudFormation stack operation to complete...
Waiting for CloudFormation stack operation to complete...
Waiting for CloudFormation stack operation to complete...
Waiting for CloudFormation stack operation to complete...
Waiting for CloudFormation stack operation to complete...
CloudFormation stack operation complete.
Config deploy complete.
```

After deploy the rule you can check the log files.

```
C:\Users\hp>rdk logs MyRule -n 5
'stty' is not recognized as an internal or external command,
operable program or batch file.
Using default terminal rows and columns.
2020-07-01 22:48:45 - END RequestId: b5b300ab-0316-4069-9275-b140271c91f8
'stty' is not recognized as an internal or external command,
operable program or batch file.
Using default terminal rows and columns.
2020-07-01 22:48:45 - REPORT RequestId: b5b300ab-0316-4069-9275-b140271c91f8
                        Duration: 1165.38 ms    Billed Duration: 1200 ms    Memory
                        Size: 256 MB    Max Memory Used: 68 MB    Init Duration:
                        178.79 ms
'stty' is not recognized as an internal or external command,
operable program or batch file.
Using default terminal rows and columns.
```

See below, This is the AWS console which is rule which I have create the above and deploy the AWS. This rule is deploy in us-east-1 so make sure you open the console in us-east-1. This console is config console.


```

C:\Users\hp>rdk sample-ci AWS::EC2::Instance
{
  "version": "1.2",
  "accountId": "681361479661",
  "configurationItemCaptureTime": "2017-04-05T22:23:11.677Z",
  "configurationItemStatus": "ResourceDiscovered",
  "configurationStateId": "1491430991677",
  "configurationItemMD5Hash": "7d83283adb8b966945d43cee39c7419c",
  "arn": "arn:aws:ec2:us-east-1:681361479661:instance/i-03402838daac1d611",
  "resourceType": "AWS::EC2::Instance",
  "resourceId": "i-03402838daac1d611",
  "awsRegion": "us-east-1",
  "availabilityZone": "us-east-1b",
  "resourceCreationTime": "2017-04-05T22:15:53.000Z",
  "tags": {},
  "relatedEvents": [
    "d3d87c29-bde3-4380-a7de-810f379246cc"
  ],
  "relationships": [
    {
      "resourceType": "AWS::EC2::NetworkInterface",
      "resourceId": "eni-d055cfc4",
      "relationshipName": "Contains NetworkInterface"
    },
    {
      "resourceType": "AWS::EC2::SecurityGroup",
      "resourceId": "sg-fd215482",
      "relationshipName": "Is associated with SecurityGroup"
    },
    {
      "resourceType": "AWS::EC2::Subnet",
      "resourceId": "subnet-1aacc7f",
      "relationshipName": "Is contained in Subnet"
    },
    {
      "resourceType": "AWS::EC2::Volume",
      "resourceId": "vol-0c24aa343c564eda8",
      "relationshipName": "Is attached to Volume"
    },
    {
      "resourceType": "AWS::EC2::VPC",
      "resourceId": "vpc-79b3ea1e",
      "relationshipName": "Is contained in Vpc"
    }
  ],
  "configuration": {
    "instanceId": "i-03402838daac1d611",
    "imageId": "ami-22ce4934",
    "state": {
      "code": 16,
      "name": "running"
    },
    "privateDnsName": "ip-172-31-74-239.ec2.internal",
    "publicDnsName": "ec2-34-205-29-138.compute-1.amazonaws.com",
    "stateTransitionReason": "",
    "keyName": "ssm-key",
    "amiLaunchIndex": 0,
  }
}

```

Manually create rule and check compliance

Here below settings is rule setting. This rule is related to s3 bucket which is for s3 bucket is not public if public so it is non complaint.

Settings

Recording is on

Turn off

Resource types to record

Select the types of resources for which you want AWS Config to record configuration changes. By default, AWS Config records configuration changes for all supported resources. You can also choose to record configuration changes for supported global resources in this region.

All resources ☒ Record all resources supported in this region ⓘ

Include global resources (e.g., AWS IAM resources) ⓘ ☒

Specific types

Data retention period Default period is 7 years

☐ Set a custom retention period for configuration items recorded by AWS Config

Amazon S3 bucket*

Your bucket receives configuration history and configuration snapshot files, which contain details for the resources that AWS Config records.

☐ Create a bucket

☒ Choose a bucket from your account

☐ Choose a bucket from another account ⓘ

Bucket name* config-bucket-520477949117 / Prefix (optional) / AWSLogs520477949117/Config/us-east-2

Amazon SNS topic

☒ Stream configuration changes and notifications to an Amazon SNS topic.

⚠ If you choose email as the notification endpoint for your SNS topic, this can cause a high volume of email. [Learn more.](#)

☐ Create a topic

☒ Choose a topic from your account

☐ Choose a topic from another account ⓘ

This is s3 bucket list which is not a public.

Welcome to Amazon S3. Create new buckets or select an existing bucket to view and configure properties.

We've temporarily re-enabled the previous version of the S3 console while we continue to improve the new S3 console experience. [Switch to the new console.](#)

Discover the console

Search for buckets All access types

+ Create bucket Edit public access settings Empty Delete

3 Buckets 2 Regions

<input type="checkbox"/> Bucket name	Access ⓘ	Region	Date created
<input type="checkbox"/> cloudtraillogs-0501	Objects can be public	US East (Ohio)	Jun 30, 2020 4:58:31 PM GMT+0530
<input type="checkbox"/> config-bucket-520477949117	Objects can be public	US East (Ohio)	Jun 30, 2020 2:39:22 PM GMT+0530
<input type="checkbox"/> config-rule-code-bucket-520477949117-us-east-1	Objects can be public	US East (N. Virginia)	Jul 1, 2020 5:43:08 PM GMT+0530

This is rule which is we created and this is complaint because all buckets are in non public mode.

The redesigned AWS Config console is now available for use. We've completely redesigned the console to improve the overall look and feel. Try it out now.

Rules

Rules represent your desired configuration settings. AWS Config evaluates whether your resource configurations comply with relevant rules and summarizes the results in the following table.

Add rule Manage remediation View details Edit

Compliance status Filter

Rule name	Compliance	Remediation action
<input type="radio"/> s3-bucket-public-write-prohibited	Compliant	AWS-PublishSNSNotification

Below is the recover the our non compliant resources using cloudWatch and Lambda function. cloudWatch is used for trigger the log file where is error and which time this error will occur. You can use cloud Trail to use the recording the event which is performed in AWS account. Lambda function is connected to the cloud Watch and is triggered the non compliant resource to compliant.

The screenshot displays the AWS Management Console interface for configuring an Event Rule. The left sidebar shows navigation options like CloudWatch, Alarms, and Rules. The main content area is titled 'Rules > S3ComplianceRule' and includes a 'Summary' section. The 'Event pattern' is defined with a JSON structure that triggers on S3 bucket events where the compliance type is 'non_compliant'. The rule's status is 'Enabled'. Below the summary, the 'Targets' section shows a single target: a Lambda function named 'awsCompliant' with the role 'Matched event'.

```
Event pattern: {
  "source": [
    "aws.config"
  ],
  "detail": {
    "requestParameters": {
      "evaluations": {
        "complianceType": [
          "non_compliant"
        ]
      }
    },
    "additionalEventData": {
      "s3_bucket_public_write_protection": [
        "s3_bucket_public_write_protection"
      ]
    }
  }
}
```

Type	Name	Input	Role	Additional parameters
Lambda function	awsCompliant	Matched event		

This screenshot shows the 'Function code' editor for the 'awsCompliant' Lambda function. The code is written in JavaScript and uses the AWS SDK to interact with S3. It iterates through the 'evaluations' array in the event data, identifies non-compliant resources, and attempts to delete the bucket policy for each. The code includes error handling and logging.

```
1 var AWS = require('aws-sdk');
2 exports.handler = async (event) => {
3   console.log("request", JSON.stringify(event, undefined, 2));
4
5   var s3 = new AWS.S3({apiVersion: '2006-03-01'});
6   var resource = event['detail']['requestParameters']['evaluations'];
7   console.log("evaluations:", JSON.stringify(resource, null, 2));
8
9   for(var i=0, len = resource.length; i<len; i++){
10     if(resource[i]['complianceType'] === "NON_COMPLIANT"){
11       console.log(resource[i]['complianceResourceId']);
12       var params = {
13         Bucket: resource[i]['complianceResourceId']
14       };
15       s3.deleteBucketPolicy(params, function(err, data){
16         if (err) console.log(err, err.stack);
17         else console.log(data);
18       });
19     }
20   }
21 };
```

Environment variables (0)