AWS Config

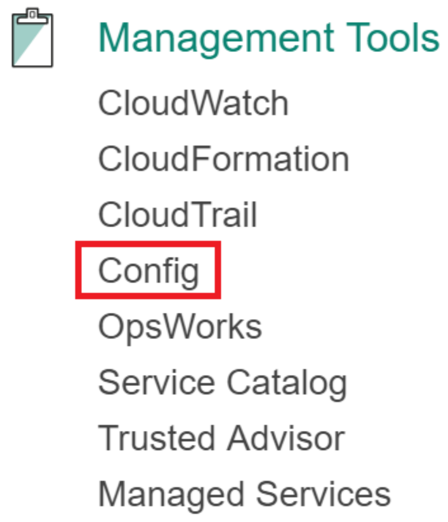
Step 1 Setting up the Configuration Recorder

Introduction

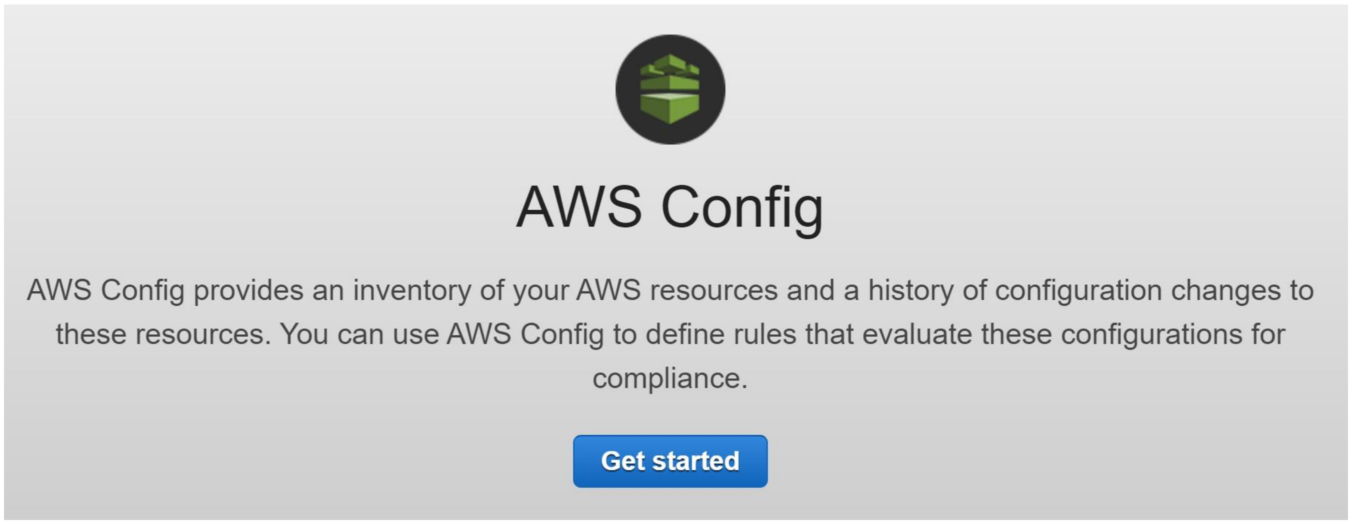
To begin using AWS Config, you must first set up the configuration recorder. You tell the configuration recorder what resource configurations to record and where to record configuration items (CIs) in AWS Simple Storage Service (S3) and through AWS Simple Notification Service (SNS). This lab won't focus on S3 or SNS, but it is a requirement to set up an S3 bucket to use AWS Config. AWS Config also uses an Identity and Access Management (IAM) role to enable the service to record configurations.

Instructions

1. In the AWS Management Console, navigate to AWS Config by clicking on **Config** under **Management Tools** in the Service drop down menu:



2. Click on **Get Started** when presented with the AWS Config welcome page:

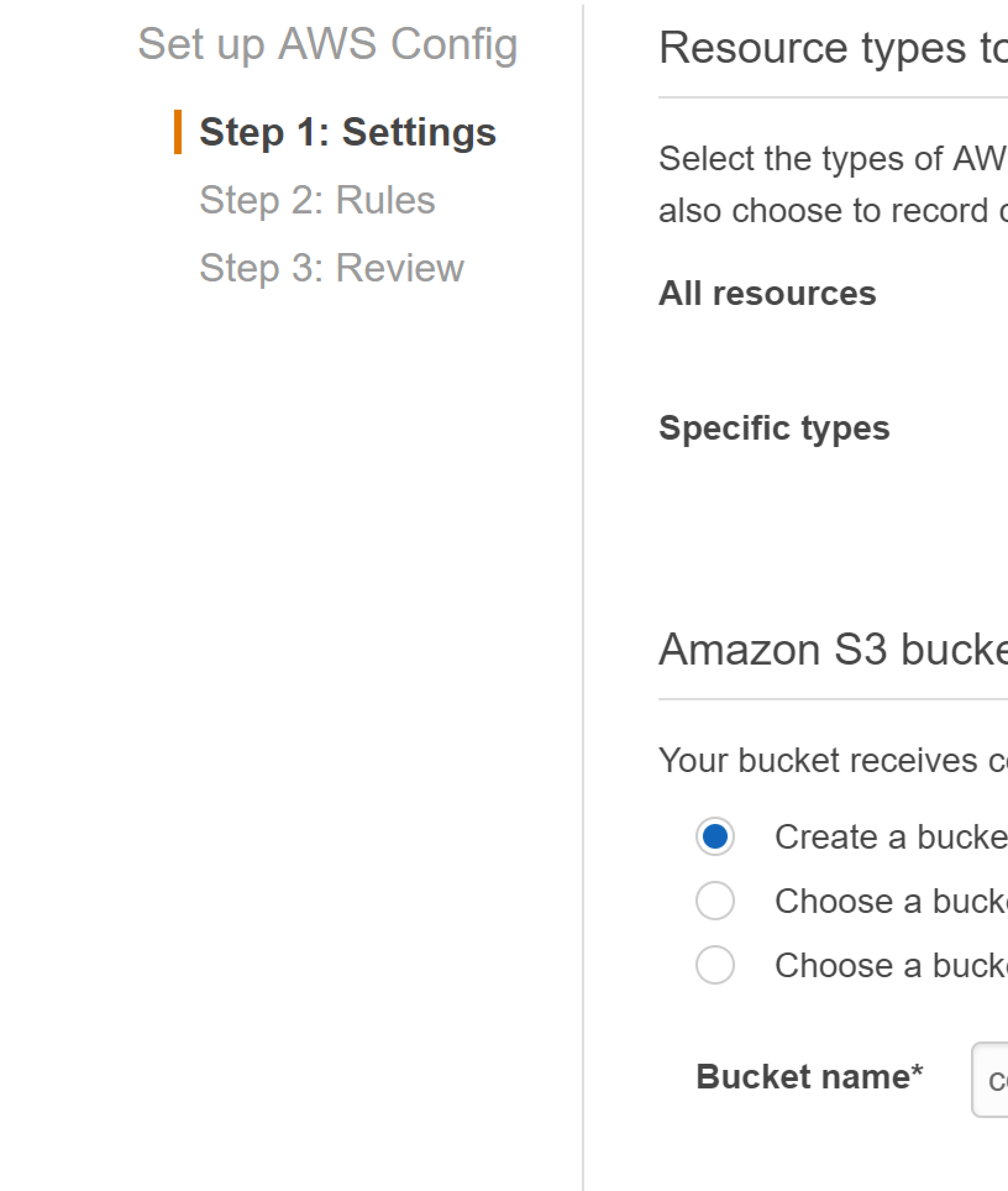

AWS Config

AWS Config provides an inventory of your AWS resources and a history of configuration changes to these resources. You can use AWS Config to define rules that evaluate these configurations for compliance.

This will directly take to you to the configuration recorder setup form.

3. Enter the following commands into the **Settings** form:

- **Resource types to record**
  - **Record all resources supported in this region**: *Unchecked*
  - **Specific Types**: *EC2: SecurityGroup* (select SecurityGroup from the drop-down menu)
- **Amazon S3 bucket**
  - **Create a bucket**: *Selected*
  - **Bucket name**: Leave the default value (it will be something like config-bucket-123456789012, changing this will mean you won't have access to the created bucket)
- **Amazon SNS**
  - **Stream configuration changes and notifications to an Amazon SNS topic**: *Unchecked*
- **AWS Config role**
  - **Choose a role from your account**: *Selected*
  - **Role name**: *config-role-us-west-2*
  - **Use the role as is**: *Checked*



Set up AWS Config

Step 1: Settings

Step 2: Rules

Step 3: Review

Resource types tc

Select the types of AW
also choose to record c

All resources

Specific types

Amazon S3 bucke

Your bucket receives c

[✓ Create a bucke]

Choose a buck

Choose a buck

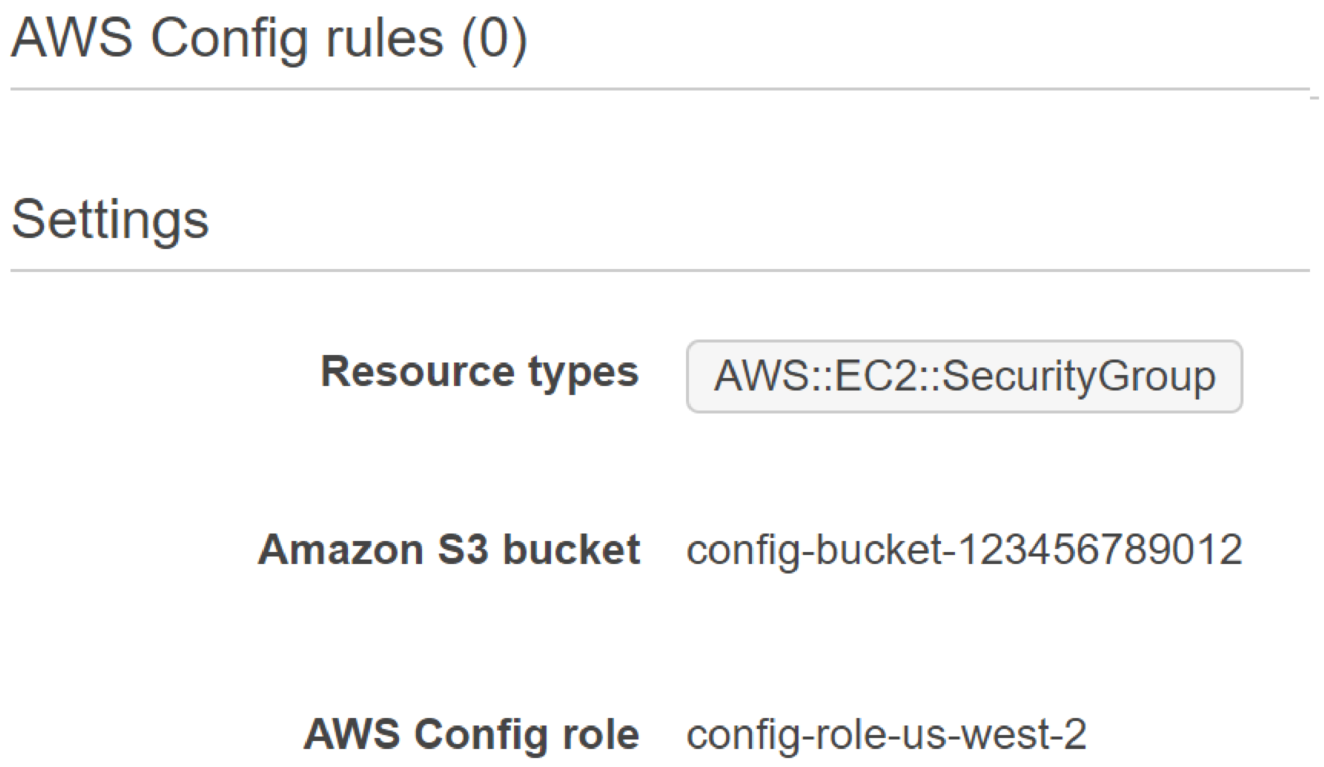
Bucket name* c

4. Once you have verified the values are correct, click **Next**.

5. In the **Rules** step, click **Skip** at the bottom right of the page.

You will go through the AWS managed rules in a future Lab Step. There are more managed rules available after you initialize the configuration recorder than there are available now, before it is initialized.

6. In the **Review** step, verify there are 0 **Config rules** and only the **AWS::EC2::SecurityGroup Resource type** appears.



AWS Config rules (0)

Settings

Resource types	AWS::EC2::SecurityGroup
Amazon S3 bucket	config-bucket-123456789012
AWS Config role	config-role-us-west-2

7. Click **Confirm** to start the configuration recorder.

Summary

In this Lab Step, you configured the configuration recorder to record EC2 security group configurations. You created an S3 bucket to store AWS Config history files and snapshots.

Step 2 Working with AWS Config Managed Rules

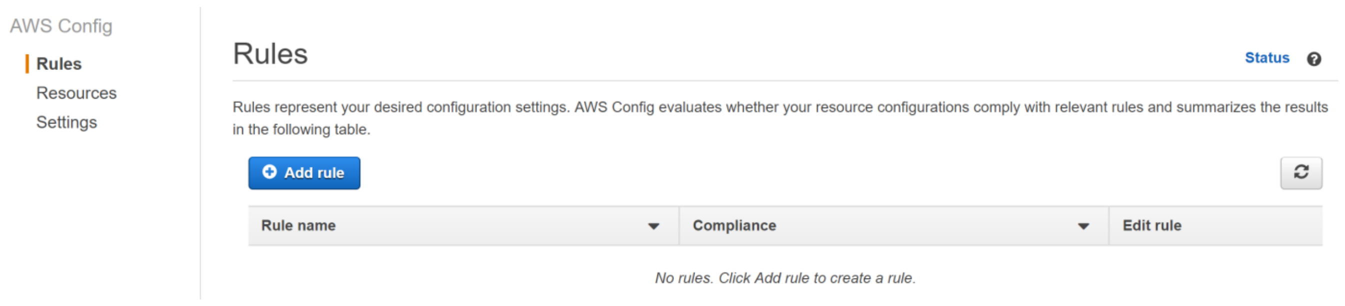
Introduction

Now that the configuration recorder has started, you can start to enable Config Rules. In this Lab Step, you will use an AWS managed rule to quickly start checking compliance. There are many managed rules provided by AWS. You should be familiar with the managed rules that are available to you since they cover many common use cases. They also don't require custom development and maintenance on your part, which can't be said for custom rules. You will work with a managed rule that checks for specific tags being set on resources.

Instructions

1. Select **Rules** in the AWS Config navigation panel on the left.

2. Click **Add rule:**

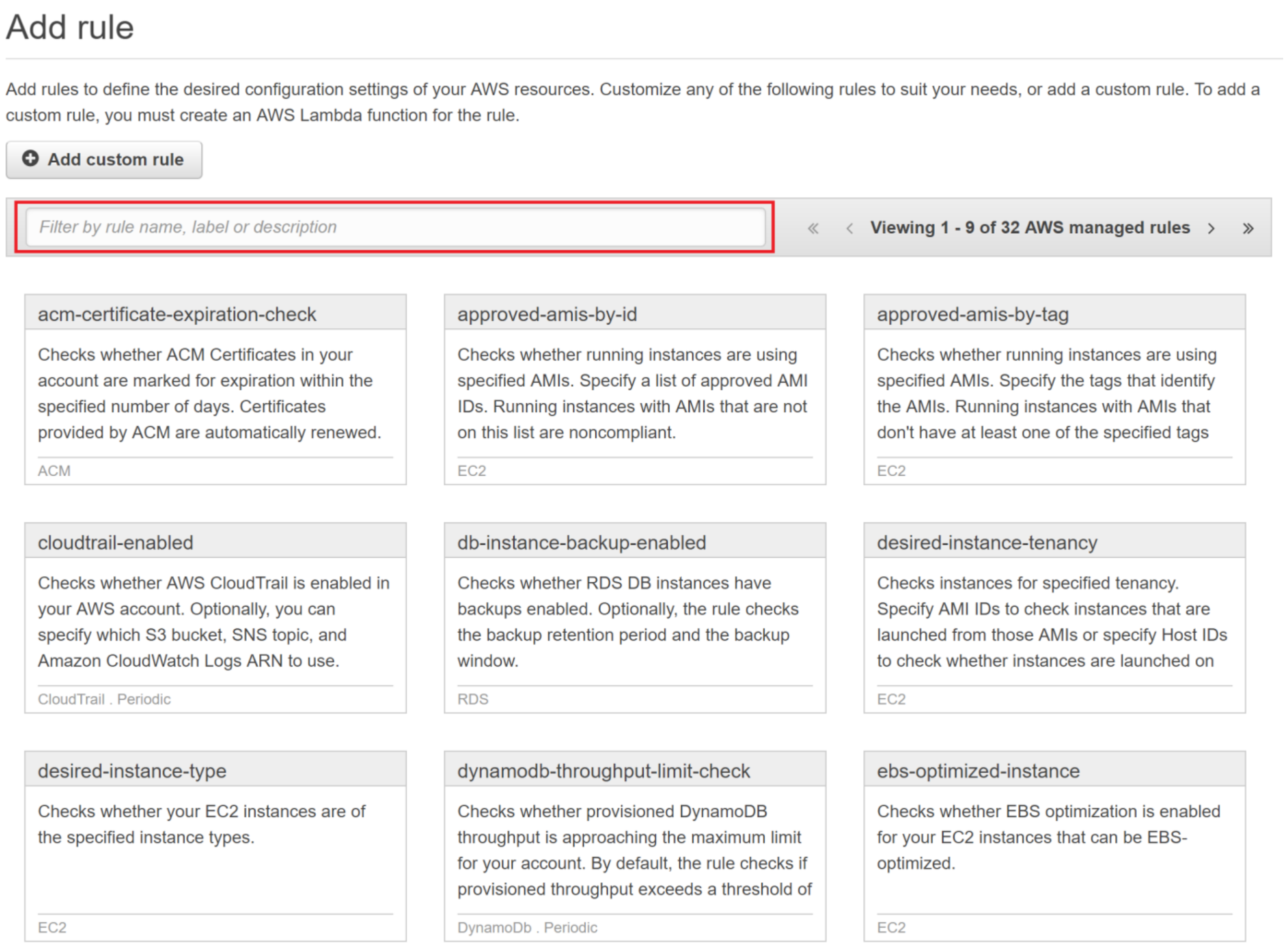


Rules represent your desired configuration settings. AWS Config evaluates whether your resource configurations comply with relevant rules and summarizes the results in the following table.

No rules. Click Add rule to create a rule.

You will be presented with a list of AWS managed rules. Take a moment to see the types of rules available in the list.

3. In the **Filter by rule name, label or description** box highlighted in the image below, enter *required-tags*



Add rule

Add rules to define the desired configuration settings of your AWS resources. Customize any of the following rules to suit your needs, or add a custom rule. To add a custom rule, you must create an AWS Lambda function for the rule.

Add custom rule
Filter by rule name, label or description
Viewing 1 - 9 of 32 AWS managed rules

acm-certificate-expiration-check
Checks whether ACM Certificates in your account are marked for expiration within the specified number of days. Certificates provided by ACM are automatically renewed.
ACM

approved-amis-by-id
Checks whether running instances are using specified AMIs. Specify a list of approved AMI IDs. Running instances with AMIs that are not on this list are noncompliant.
EC2

approved-amis-by-tag
Checks whether running instances are using specified AMIs. Specify the tags that identify the AMIs. Running instances with AMIs that don't have at least one of the specified tags
EC2

cloudtrail-enabled
Checks whether AWS CloudTrail is enabled in your AWS account. Optionally, you can specify which S3 bucket, SNS topic, and Amazon CloudWatch Logs ARN to use.
CloudTrail . Periodic

db-instance-backup-enabled
Checks whether RDS DB instances have backups enabled. Optionally, the rule checks the backup retention period and the backup window.
RDS

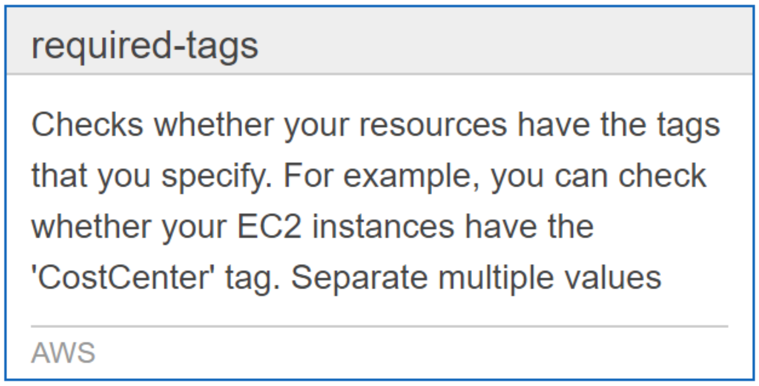
desired-instance-tenancy
Checks instances for specified tenancy. Specify AMI IDs to check instances that are launched from those AMIs or specify Host IDs to check whether instances are launched on
EC2

desired-instance-type
Checks whether your EC2 instances are of the specified instance types.
EC2

dynamodb-throughput-limit-check
Checks whether provisioned DynamoDB throughput is approaching the maximum limit for your account. By default, the rule checks if provisioned throughput exceeds a threshold of
DynamoDb . Periodic

ebs-optimized-instance
Checks whether EBS optimization is enabled for your EC2 instances that can be EBS-optimized.
EC2

5. Click on the **required-tags** rule card:



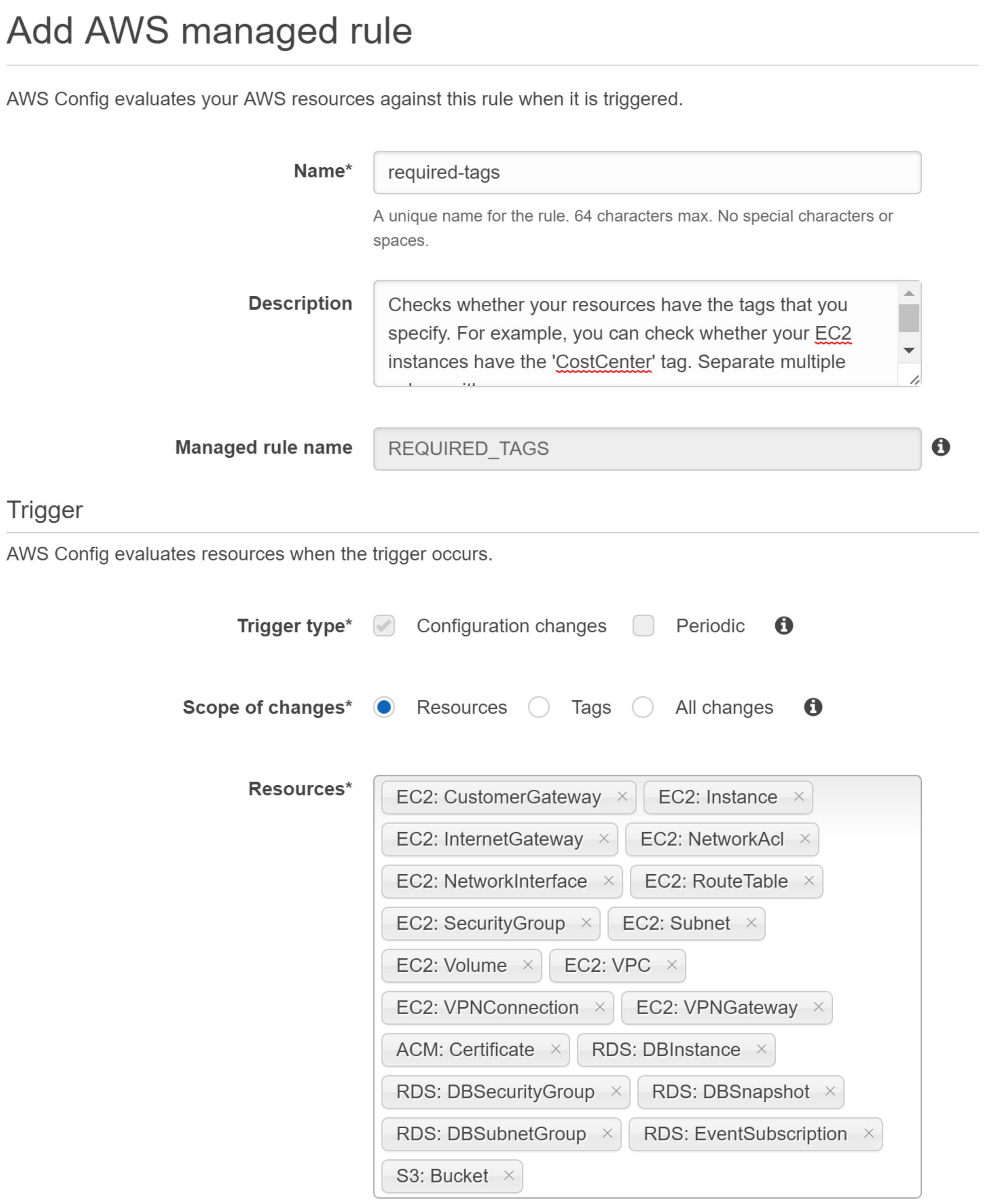
required-tags

Checks whether your resources have the tags that you specify. For example, you can check whether your EC2 instances have the 'CostCenter' tag. Separate multiple values

AWS

The **required-tags** rule can check that resources have specific tags that you require. For this lab, imagine you are building out a blue-green deployment environment. Blue-green deployments maintain two copies of the deployment environment, a blue environment and a green environment. As part of governing your environments, you require all resources to have a Deployment tag that identifies if it is part of the blue environment or the green environment.

6. In the **Add AWS managed rule** form, keep the default values for **Name**, **Description**, **Scope of changes** and the **Triggers** section:



Add AWS managed rule

AWS Config evaluates your AWS resources against this rule when it is triggered.

Name*: required-tags

A unique name for the rule. 64 characters max. No special characters or spaces.

Description: Checks whether your resources have the tags that you specify. For example, you can check whether your EC2 instances have the 'CostCenter' tag. Separate multiple

Managed rule name: REQUIRED_TAGS

Trigger

AWS Config evaluates resources when the trigger occurs.

Trigger type*: [✓ Configuration changes] Periodic

Scope of changes*: [✓ Resources] Tags All changes

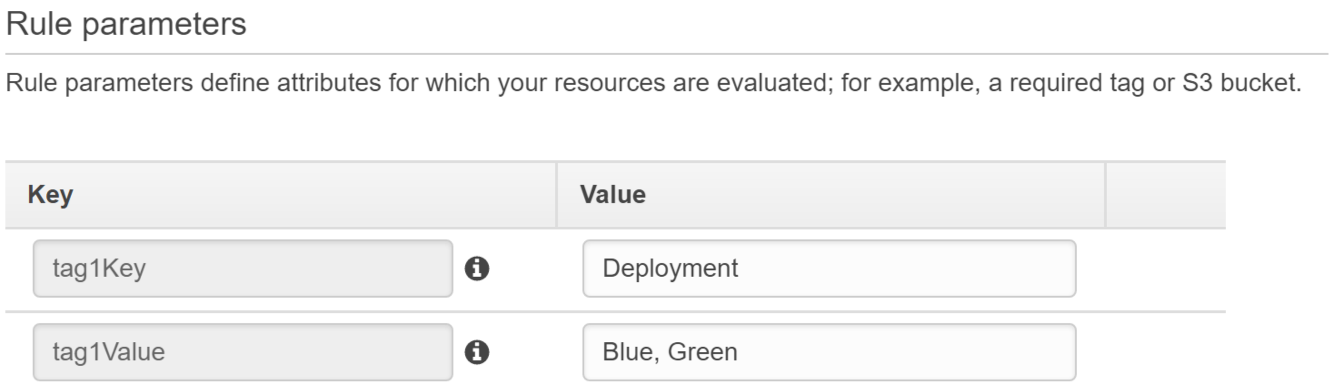
Resources*:

EC2: CustomerGateway ×
EC2: Instance ×
EC2: InternetGateway ×
EC2: NetworkAcl ×
EC2: NetworkInterface ×
EC2: RouteTable ×
EC2: SecurityGroup ×
EC2: Subnet ×
EC2: Volume ×
EC2: VPC ×
EC2: VPNConnection ×
EC2: VPNGateway ×
ACM: Certificate ×
RDS: DBInstance ×
RDS: DBSecurityGroup ×
RDS: DBSnapshot ×
RDS: DBSubnetGroup ×
RDS: EventSubscription ×
S3: Bucket ×

Notice that the **Trigger type** is set to **Configuration changes**. This means the rule will evaluate any time the configuration recorder records a new configuration. The **Resources** list contains all the types supported. You can leave all the resources in the list. It won't hurt to have all of them even though the configuration recorder is only recording security group configuration changes.

7. In the **Rule parameters** section at the bottom of the form, enter:

- **tag1Key**: *Deployment*
- **tag1Value**: *Blue, Green*



Rule parameters

Rule parameters define attributes for which your resources are evaluated; for example, a required tag or S3 bucket.

Key	Value
tag1Key	Deployment
tag1Value	Blue, Green

The key is required, while the value(s) is optional. When the value is specified, the tag must have one of the specified values. Multiple values are separated by commas. If no value is specified, the rule only checks if the tag is set with no specific criteria for the tag value.

8. Click **Save** to create the rule and have AWS Config automatically start evaluating the rule on the Resources.

It can take a minute for the evaluation to complete.

Summary

In this Lab Step, you set up an AWS Config managed rule. The rule checks resource compliance when using a specific tag and allowable values. This type of rule can be useful for many scenarios including the deployment governance scenario utilized in this lab.

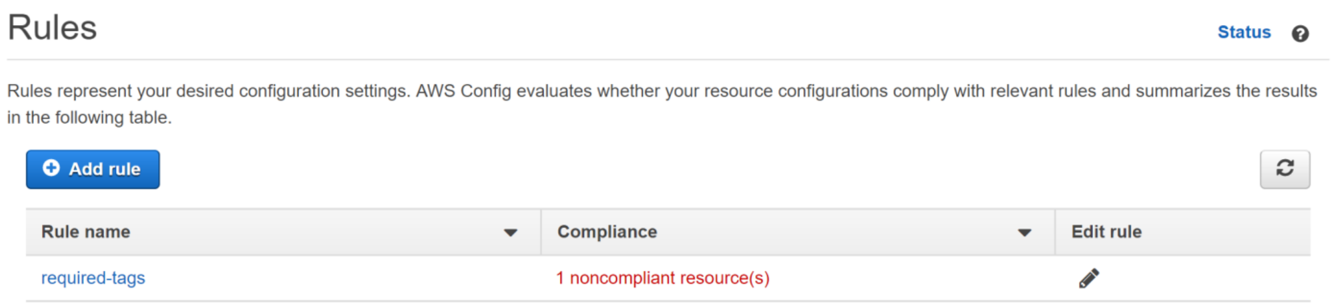
Step 3 Analyzing and Remedying a Noncompliant Resource

Introduction

Recall that the configuration recorder is recording only security group configurations. The security group is not compliant with the managed rule you previously set up. In this Lab Step, you will learn to analyze noncomplianct resources using AWS Config and use its integration with CloudTrail to review associated API calls. After analyzing, you will adjust the security group configuration and verify it is compliant.

Instructions

1. On the AWS Config **Rules** page, observe that it found **1 noncompliant resource** after rule evaluation:



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.

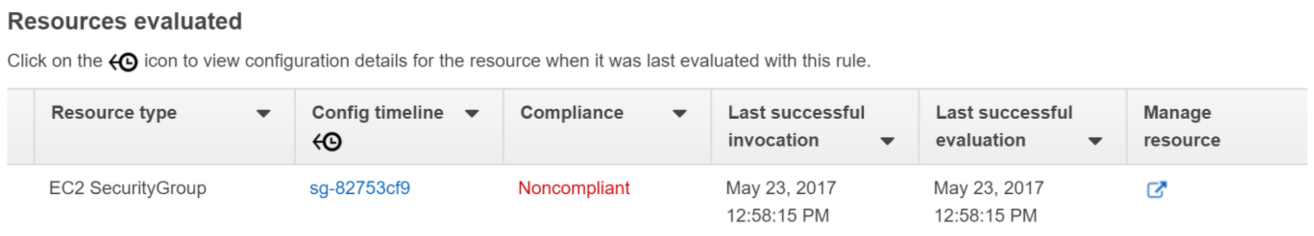
Rule name	Compliance	Edit rule
required-tags	1 noncompliant resource(s)	

Note: You can click the refresh button on the right hand side until you see the noncompliant resource. The managed rule can take about a minute to finish evaluating.

2. Click on **required-tags** in the **Rule name** column.

This will open the rule details view for the **required-tags** rule. In the top section, you will see all of the settings you configured when adding the rule.

3. Scroll down to the **Resources evaluated** section:



Resources evaluated

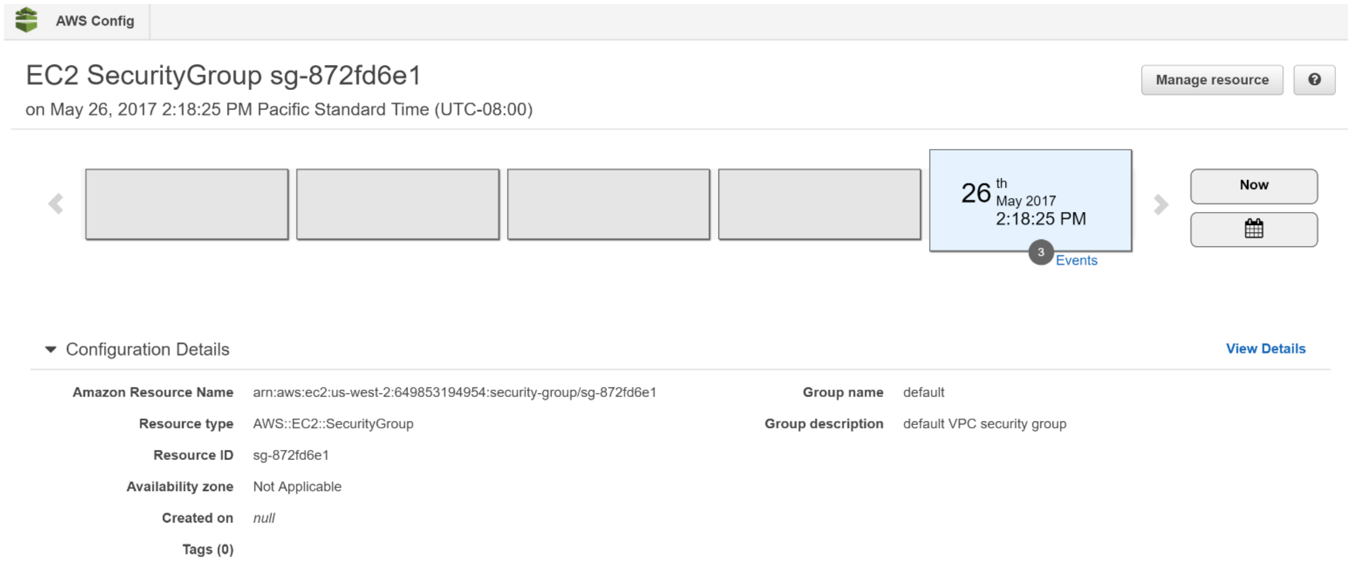
Click on the icon to view configuration details for the resource when it was last evaluated with this rule.

Resource type	Config timeline	Compliance	Last successful invocation	Last successful evaluation	Manage resource
EC2 SecurityGroup	sg-82753cf9	Noncompliant	May 23, 2017 12:58:15 PM	May 23, 2017 12:58:15 PM	

Here you will find a summary of all the resources evaluated against your rules. You will see it evaluated an **EC2 Security Group** and it is **Noncompliant** with your rules.

4. Click on the **Noncompliant** security group name under the **Config timeline** to open the Config timeline.

5. Click on the timeline box that shows some Events below the date:



EC2 SecurityGroup sg-872fd6e1

on May 26, 2017 2:18:25 PM Pacific Standard Time (UTC-08:00)

Configuration Details

Amazon Resource Name	arn:aws:ec2:us-west-2:649853194954:security-group/sg-872fd6e1	Group name	default
Resource type	AWS::EC2::SecurityGroup	Group description	default VPC security group
Resource ID	sg-872fd6e1		
Availability zone	Not Applicable		
Created on	*null*		
Tags (0)			

Note: You can periodically click on the **Now** button to refresh the timeline in case you don't see any **Events** under the configuration item in the timeline.

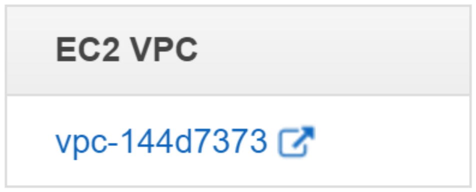
The Config timeline presents configuration changes over time. The **Configuration Details** section summarizes the configuration. You will see important configuration details for the resource configuration at the time selected in the timeline. Notice the Tags don't include a Deployment tag. The missing Deployment tag is currently causing the noncompliance.

6. Click on **View Details** on the far right of **Configuration Details**.

This will open the full configuration. There are many details here that are not in the summary. For example, scroll down and see that the **ipPermissions** that specify the security group ingress and egress permissions are visible.

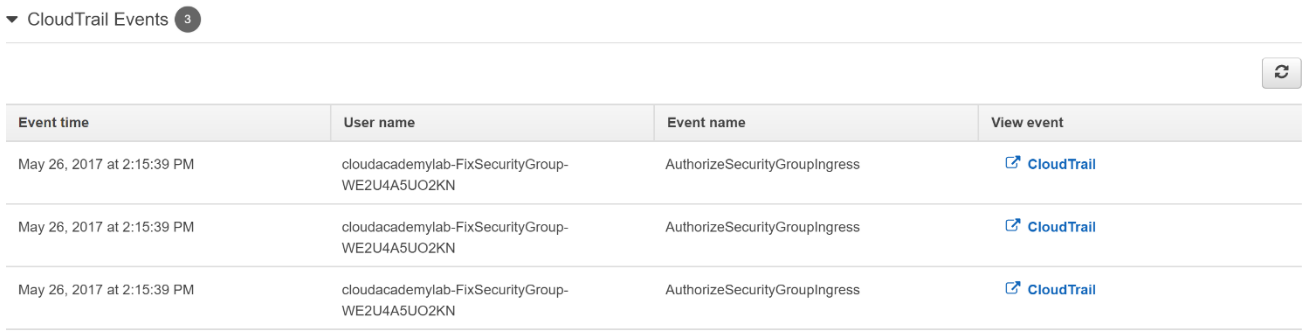
7. Click **Close** to return the the Config timeline page.

8. Scroll down and expand the **Relationships** section by clicking on the triangle to the left of it:



Here you will find all of the resources related to the security group that AWS Config found. In the case of your security group, the resource is a VPC. This summary is very useful for getting a quick overview of how a resource fits into your cloud environment.

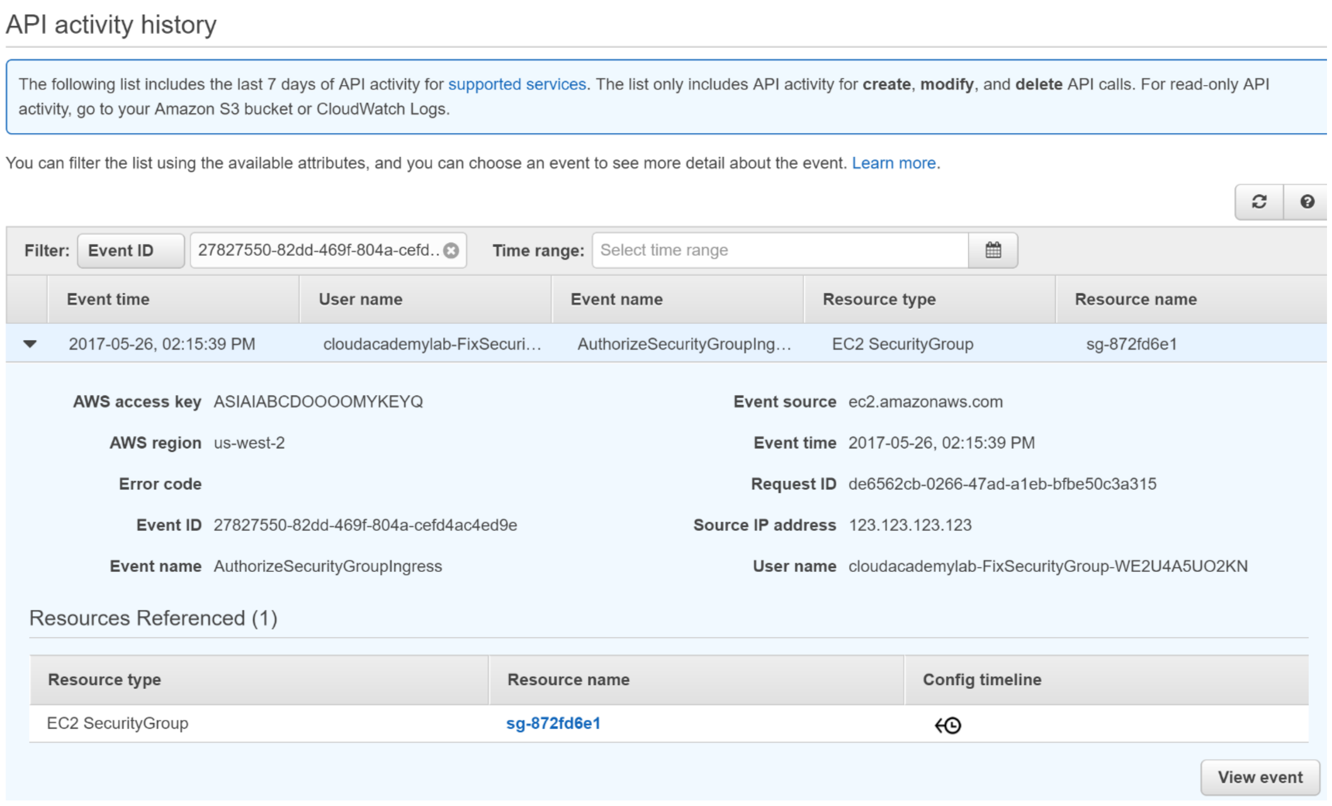
9. Expand the **CloudTrail Events** section at the bottom:



Event time	User name	Event name	View event
May 26, 2017 at 2:15:39 PM	cloudacademylab-FixSecurityGroup-WE2U4A5UO2KN	AuthorizeSecurityGroupIngress	CloudTrail
May 26, 2017 at 2:15:39 PM	cloudacademylab-FixSecurityGroup-WE2U4A5UO2KN	AuthorizeSecurityGroupIngress	CloudTrail
May 26, 2017 at 2:15:39 PM	cloudacademylab-FixSecurityGroup-WE2U4A5UO2KN	AuthorizeSecurityGroupIngress	CloudTrail

The lab environment enabled CloudTrail as part of the lab startup routine so you can view the CloudTrail events related to resource configurations within AWS Config. You can see when events happened, what user performed them, and what the name of the event is. There is also a link to view the event in CloudTrail. This can be useful for seeing the event in relation to events impacting other resources.

10. As an example, click on the **CloudTrail** link next to an **AuthorizeSecurityGroupIngress** event:



API activity history

The following list includes the last 7 days of API activity for supported services. The list only includes API activity for **create**, **modify**, and **delete** API calls. For read-only API activity, go to your Amazon S3 bucket or CloudWatch Logs.

You can filter the list using the available attributes, and you can choose an event to see more detail about the event. Learn more.

Filter: Event ID 27827550-82dd-469f-804a-cefd.. Time range: Select time range

Event time	User name	Event name	Resource type	Resource name
2017-05-26, 02:15:39 PM	cloudacademylab-FixSecuri...	AuthorizeSecurityGroupIng...	EC2 SecurityGroup	sg-872fd6e1

AWS access key	ASIAIABCDOOOOMYKEYQ	**Event source**	ec2.amazonaws.com
AWS region	us-west-2	**Event time**	2017-05-26, 02:15:39 PM
Error code		**Request ID**	de6562cb-0266-47ad-a1eb-bfbe50c3a315
Event ID	27827550-82dd-469f-804a-cefd4ac4ed9e	**Source IP address**	123.123.123.123
Event name	AuthorizeSecurityGroupIngress	**User name**	cloudacademylab-FixSecurityGroup-WE2U4A5UO2KN

Resources Referenced (1)

Resource type	Resource name	Config timeline
EC2 SecurityGroup	sg-872fd6e1	

View event

From this view inside CloudTrail, you can see more detailed information about the event. For example, the **Source IP address** of the API request. Even more details about the identity of the user are available if you click the **View event** button in the bottom right corner.

You could also filter the **API Activity history** to show events that happened before this event as a technique for analyzing the sequence of events. Since CloudTrail records all API activity, you can gain more perspective around the event in Cloudtrail. This is in contrast to the timeline view in AWS Config which only shows the events related to one resource. Lastly, note that when AWS Config is enabled, CloudTrail provides a link back to Config under the Config timeline column. This shows that AWS Config also integrates nicely into CloudTrail.

11. Return the the AWS Config browser tab, scroll to the top and click **Manage resource**.

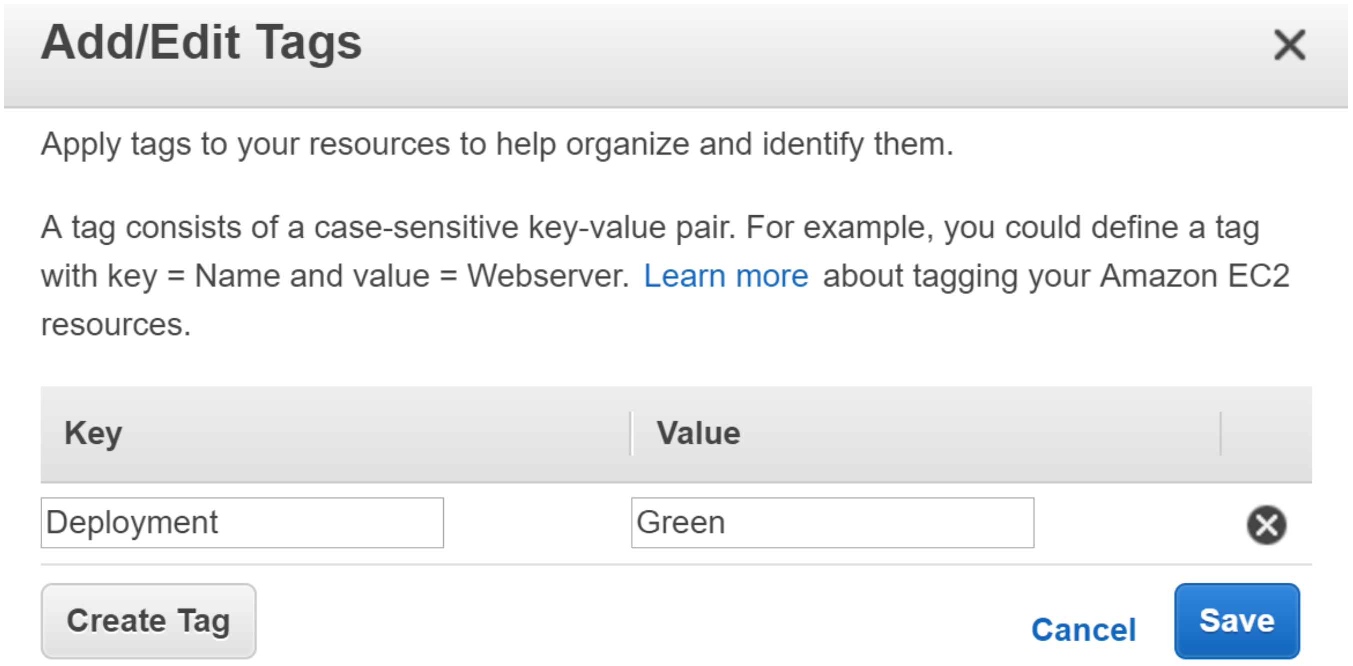
This is a quick way to go to the Management Console page for configuring the resource.

12. On the **Tags** tab, click the **Add/Edit Tags** button.

13. In the Add/Edit Tags dialog, click **Create Tag** and enter the following values:

- **Key**: *Deployment*

- **Value**: *Green*



Add/Edit Tags

Apply tags to your resources to help organize and identify them.

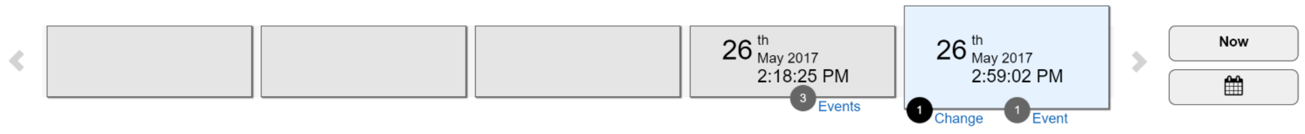
A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver. Learn more about tagging your Amazon EC2 resources.

Key	Value
Deployment	Green

This tag makes the security group compliant with the managed rule you enabled.

14. Click **Save** and return to the AWS Config timeline browser tab.

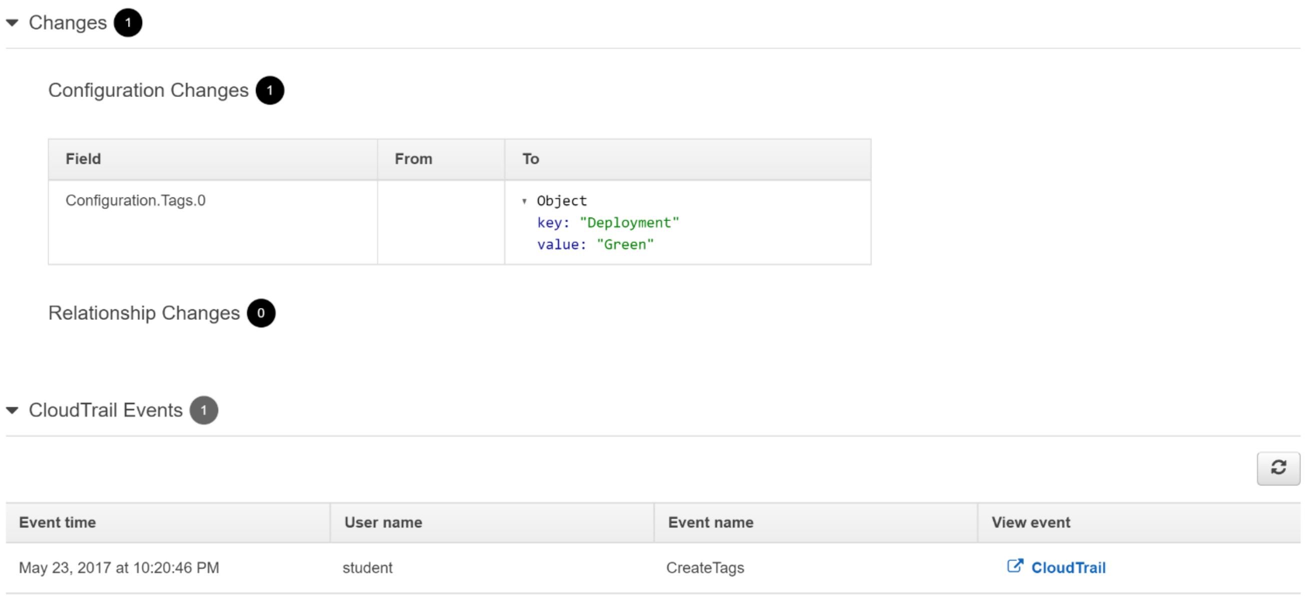
15. Click on the **Now** until you see a new configuration item in the timeline:



Note: It can take up to 10 minutes for the configuration change to be recorded. You may want to read ahead in the lab and return to this step after several minutes

Notice the new item has a **Change** and an **Event**. The **Configuration Details** also include the **Deployment:Green** tag now.

16. Expand the **Changes** and **CloudTrail Events**:


Changes 1

Configuration Changes 1

Field	From	To
Configuration.Tags.0		▾ Object key: "Deployment" value: "Green"

Relationship Changes 0

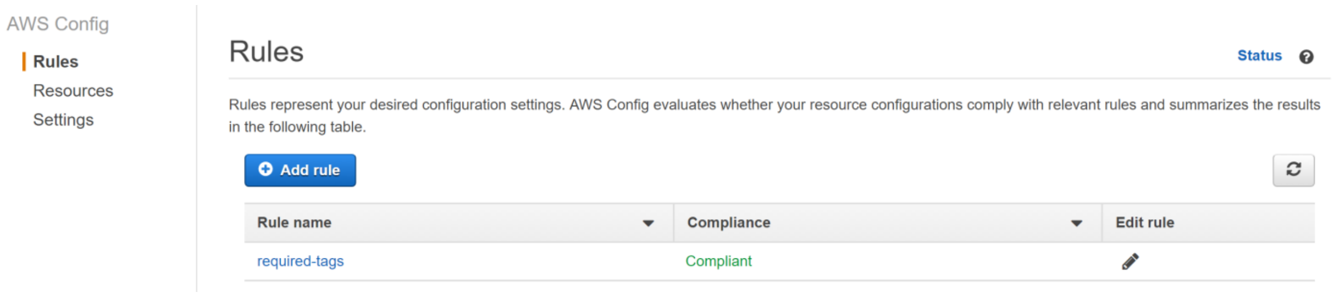
CloudTrail Events 1

Event time	User name	Event name	View event
May 23, 2017 at 10:20:46 PM	student	CreateTags	CloudTrail

In the **Changes** section, AWS Config tells you what changed from the most previously recorded configuration. In this case, it tells you the Deployment: Green tag was added in the **Configuration Changes**. There were no **Relationship Changes** this time, but AWS Config also records those changes.

In the **CloudTrail Events**, the **CreateTags** event is logged by the **student** user.

17. Return to the **Rules** page by navigating to **Services** > **Management Tools** > **Config** > **Rules**:



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.

Rule name	Compliance	Edit rule
required-tags	Compliant	

The **Compliance** column value reads **Compliant**, verifying the rule is in compliance after the security group configuration change.

Summary

In this Lab Step, you enabled an AWS Config managed rule. In analyzing a noncompliant resource, you saw how AWS Config integrates with CloudTrail to combine the benefits of both services. For practical experience, you also resolved a noncompliant configuration and observed how AWS Config reports changes in the timeline.

Step 4 Working with AWS Config Custom Rules

Introduction

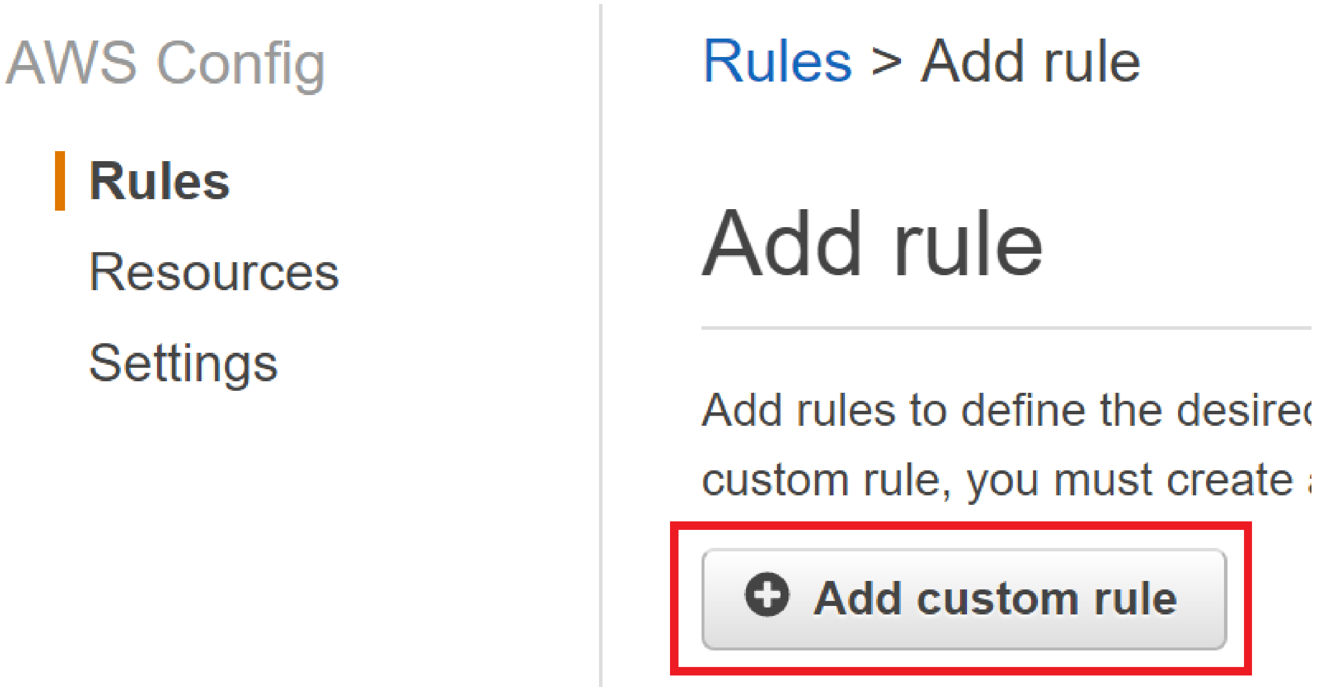
AWS Config custom rules can be used for a wide variety of scenarios when AWS managed rules don't fit your use case. Custom rules rely on an AWS Lambda function to evaluate resource configuration. You will be exposed to how to create a custom rule using pre-written code for a Lambda function. For demonstration purposes, the code simply evaluates whether an RDP tcp port is open to incoming traffic. You will check and correct noncompliance using the custom rule in this Lab Step.

Note: There is a managed rule that can achieve a similar result. The rule's name is *restricted-common-ports*. You should use that managed rule if you need to block the RDP port. The custom rule in the Lab Step is for demonstration purposes only, allowing you to get familiar with the process of working with custom rules.

Instructions

1. From the AWS Config **Rules** page, click **Add rule**.

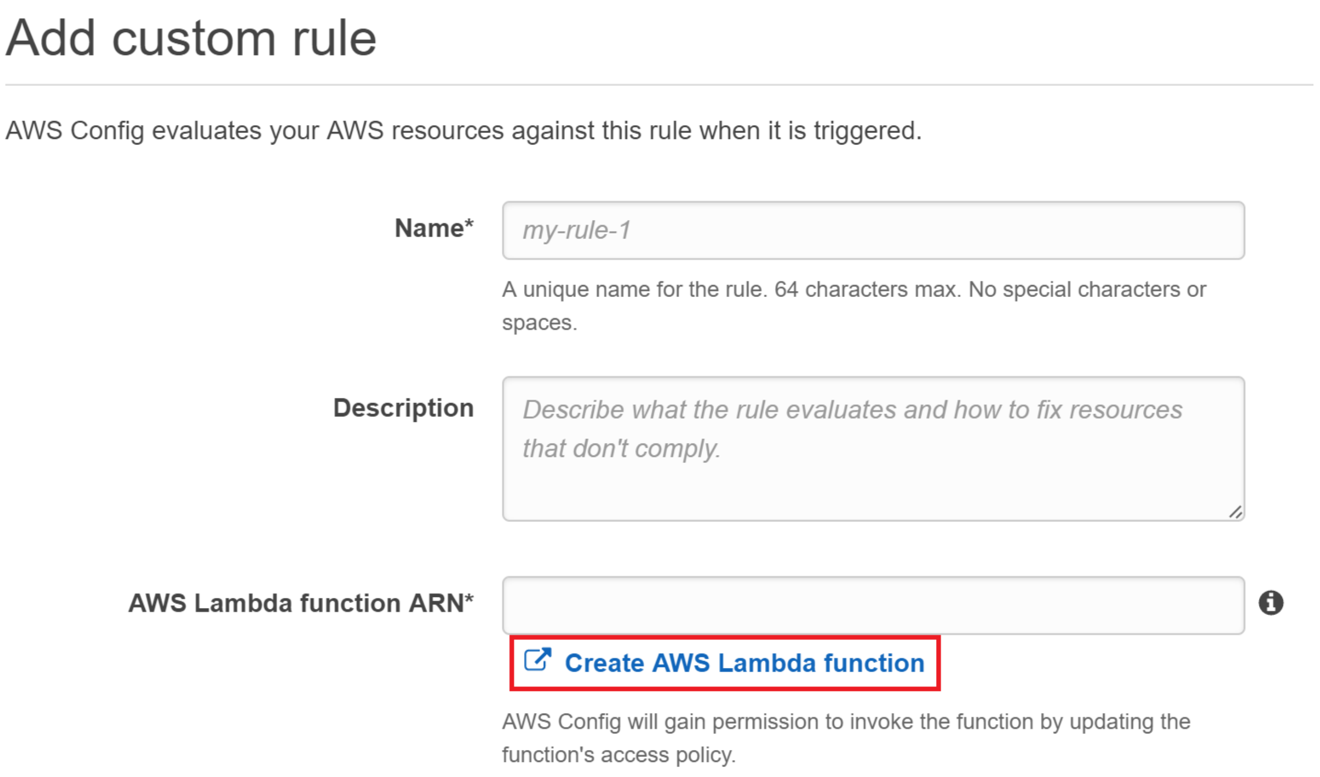
2. Click **Add custom rule** below the **Add rule** heading:



Add rule

Add rules to define the desire
custom rule, you must create

3. Click **Create AWS Lambda function**:


Add custom rule

AWS Config evaluates your AWS resources against this rule when it is triggered.

Name*	my-rule-1
	A unique name for the rule. 64 characters max. No special characters or spaces.
Description	Describe what the rule evaluates and how to fix resources that don't comply.
AWS Lambda function ARN*	
	Create AWS Lambda function
	AWS Config will gain permission to invoke the function by updating the function's access policy.

This takes you directly to the AWS Lambda new function wizard to create the function that will evaluate your custom rule.

4. In the **Basic Information** section, enter the following:

- **Name**: *evaluate-port-ingress* (Important to enter exactly as written or the Lambda function will fail to create)
- **Role**: *Choose an existing role*
- **Existing role**: *config-lambda-role*

The config-lambda-role was created for you during the lab startup. It authorizes Lambda to describe security groups (`ec2.describeSecurityGroups`) and put evaluations into AWS Config (`config.putEvaluations`).

Click **Create Function**

5. In the **Lambda function code** section, leave the **Code entry type** as **Edit code inline** and paste in the following JavaScript code block that targets Node.js:

```
//
// Ensure a security group does not allow tcp ingress on a specified port
//

var aws = require('aws-sdk');
var config = new aws.ConfigService();

// Helper function used to validate input
function checkDefined(reference, referenceName) {
  if (!reference) {
    console.log("Error: " + referenceName + " is not defined");
    throw referenceName;
  }
  return reference;
}

// Check whether the resource has been deleted. If it has, then the
evaluation is unnecessary.
function isApplicable(configurationItem, event) {
  checkDefined(configurationItem, "configurationItem");
  checkDefined(event, "event");
  var status = configurationItem.configurationItemStatus;
  var eventLeftScope = event.eventLeftScope;
  return ('OK' === status || 'ResourceDiscovered' === status) && false ===
eventLeftScope;
}

function createPutEvaluationsRequest(event, configurationItem, compliance) {
  var putEvaluationsRequest = {};

  // Put together the request that reports the evaluation status
  putEvaluationsRequest.Evaluations = [
    {
      ComplianceResourceType: configurationItem.resourceType,
      ComplianceResourceId: configurationItem.resourceId,
      ComplianceType: compliance,
      OrderingTimestamp: configurationItem.configurationItemCaptureTime
    }
  ];
  putEvaluationsRequest.ResultToken = event.resultToken;
  putEvaluationsRequest.TestMode = false;

  return putEvaluationsRequest;
}

function putEvaluations(callback, putEvaluationsRequest) {
  // Invoke the Config API to report the result of the evaluation
  config.putEvaluations(putEvaluationsRequest, function (err, data) {
    if (err) {
      callback(err);
    } else {
      callback(null, "success");
    }
  });
}

// This is the handler that's invoked by Lambda
exports.handler = function (event, context, callback) {
  event = checkDefined(event, "event");
  var invokingEvent = JSON.parse(event.invokingEvent);
  var ruleParameters = JSON.parse(event.ruleParameters);
  var configurationItem = checkDefined(invokingEvent.configurationItem,
"invokingEvent.configurationItem");
  var compliance = 'NOT_APPLICABLE';
  var putEvaluationsRequest = {};

  if (isApplicable(configurationItem, event)) {
    checkDefined(configurationItem, "configurationItem");
    checkDefined(configurationItem.configuration,
"configurationItem.configuration");
    checkDefined(ruleParameters, "ruleParameters");

    // This is where it is determined whether the resource is compliant or
not.
    // In this example, we look at the IP permissions of the EC2 security
group and determine
    // if it allows ingress traffic on a specified TCP port. If the port is
open, the
    // security group is marked non-compliant. Otherwise, it is marked
compliant.
    if ('AWS::EC2::SecurityGroup' !== configurationItem.resourceType) {
      putEvaluationsRequest = createPutEvaluationsRequest(event,
configurationItem, compliance);
      putEvaluations(callback, putEvaluationsRequest);
    } else {
      groupId = configurationItem.configuration.groupId;

      var ec2 = new aws.EC2({ apiVersion: '2016-11-15' });
      var params = {
        DryRun: false,
        GroupIds: [
          groupId
        ]
      };
      ec2.describeSecurityGroups(params, function (err, data) {
        var compliance = 'COMPLIANT';
        if (err) {
          compliance = 'NON_COMPLIANT';
        } else {
          var ipPermissions = data.SecurityGroups[0].IpPermissions;
          for (var i = 0; i < ipPermissions.length; i++) {
            var ipPermission = ipPermissions[i];
            // The actual test condition (allows default allow all rule
(IpProtocol === '-1') to be compliant for demonstration purposes)
            if (ipPermission.IpProtocol === 'tcp'
                && ipPermission.FromPort >= ruleParameters.port
                && ipPermission.ToPort <= ruleParameters.port) {
              compliance = 'NON_COMPLIANT';
              break;
            }
          }
        }
        putEvaluationsRequest = createPutEvaluationsRequest(event,
configurationItem, compliance);
        putEvaluations(callback, putEvaluationsRequest);
      });
    }
  } else {
    putEvaluationsRequest = createPutEvaluationsRequest(event,
configurationItem, compliance);
    putEvaluations(callback, putEvaluationsRequest);
  }
};
```

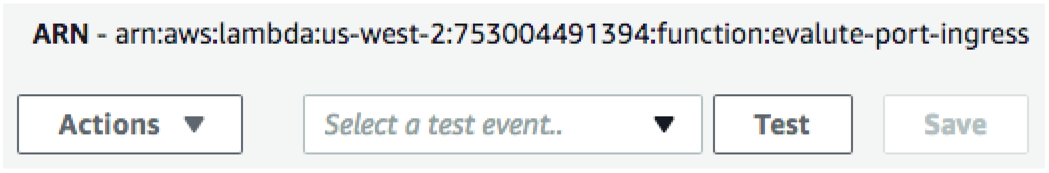
Note: It is not important to understand the code. All you need to know is that the code checks whether a security group allows inbound traffic on a specified TCP port. It makes use of the AWS SDK to access AWS Config and EC2. It is possible to modify the Lambda function so that it automatically enforces compliance by modifying the security group. This is left as an exercise to the student.

6. In the **Basic Settings** frame, set the following values:

- **Memory (MB)**: *128* (default value)
- **Timeout**: *0* min *30* sec
- Default values for the rest

This allows the Lambda function 30 seconds and 128MB of peak memory to complete evaluation. This is more than enough for the simple check performed by the code.

7. Copy the Lambda function's **ARN** in the upper right corner:


ARN - arn:aws:lambda:us-west-2:753004491394:function:evalute-port-ingress

8. Return to the AWS Config **Add custom rule** browser tab and enter the following (paste the **AWS Lambda function ARN** first while it's in your clipboard):

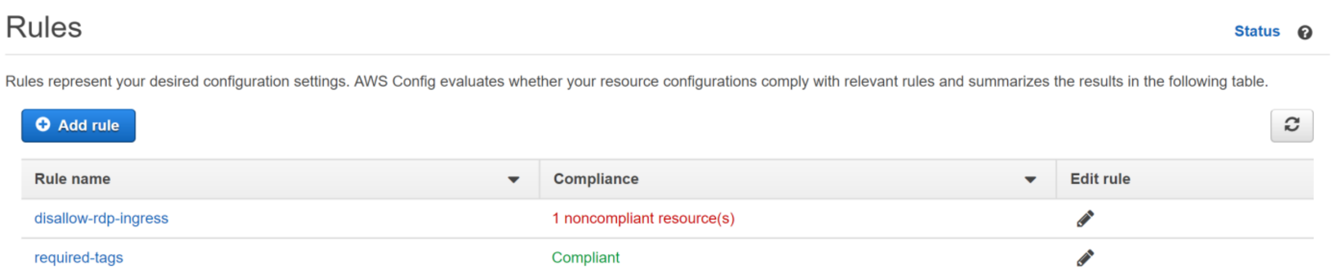
- **Name**: *disallow-rdp-ingress*
- **Description**: *Check security groups for allowing incoming RDP TCP traffic. Disallow the traffic to become compliant.*
- **AWS Lambda function ARN**: Paste the ARN copied from the AWS Lambda function page
- **Trigger**
    - **Trigger type**: *Configuration changes*
    - **Scope of changes**: *Resources*
    - **Resources**: *EC2: SecurityGroup*
- **Rule parameters**
    - **Key**: *port* (lower-case)
    - **Value**: *3389*

RDP traffic is over tcp port 3389 by default. There are ways to get RDP traffic around this rule, but it is only for demonstration purposes. **Rule parameters** are a way to make your custom rules generalizable. The same rule could be used to block any desired port using different settings of the **Rule parameters**. If you are curious about the code, the values set for **Rule parameters** correspond to `ruleParameters` in the code above.

9. Click **Save**.

Your custom rule will automatically begin evaluating.

10. After a minute, click the refresh button on the right side of the **Rules** section:



Rules

Status

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

Rule name	Compliance	Edit rule
disallow-rdp-ingress	1 noncompliant resource(s)	
required-tags	Compliant	

You will see the rule reports that there is **1 noncompliant resource**. You will now investigate and enforce compliance.

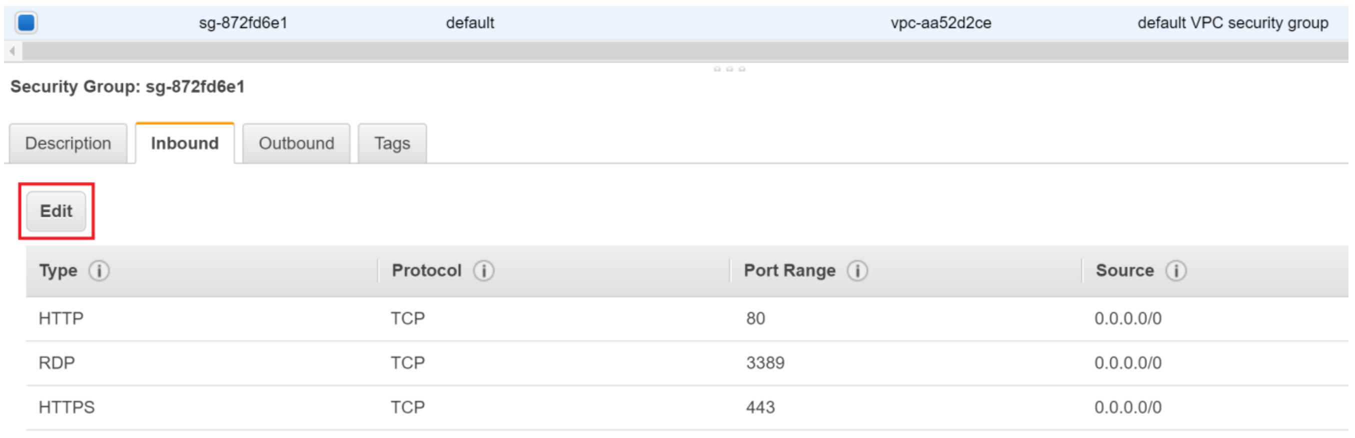
11. Click on **disallow-rdp-ingress** in the **Rules** table.

12. Open the **Manage resource** link for the for the **Noncompliant** security group.

13. Select the **Inbound** tab.

Notice the security group is allowing incoming RDP traffic. That is causing the noncompliance.

14. Click the **Edit** button:



	sg-872fd6e1	default	vpc-aa52d2ce	default VPC security group

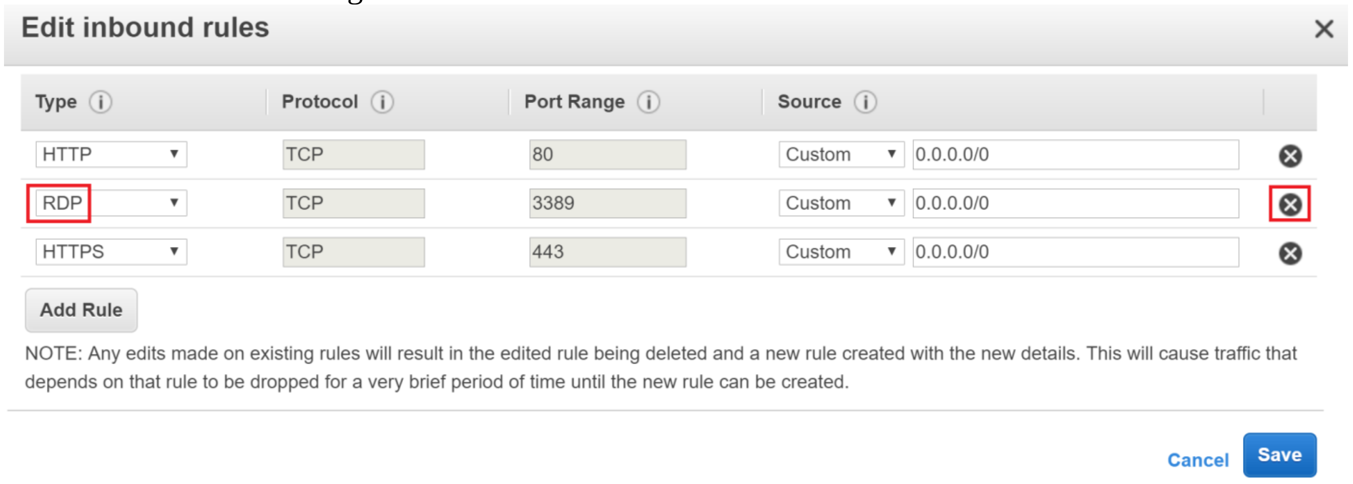
Security Group: sg-872fd6e1

Description | Inbound | Outbound | Tags

Edit

Type	Protocol	Port Range	Source
HTTP	TCP	80	0.0.0.0/0
RDP	TCP	3389	0.0.0.0/0
HTTPS	TCP	443	0.0.0.0/0

15. Click on the **x** on the right side of the **RDP** row:


Edit inbound rules

Type	Protocol	Port Range	Source		
HTTP	TCP	80	Custom	0.0.0.0/0	
RDP	TCP	3389	Custom	0.0.0.0/0	
HTTPS	TCP	443	Custom	0.0.0.0/0	

Add Rule

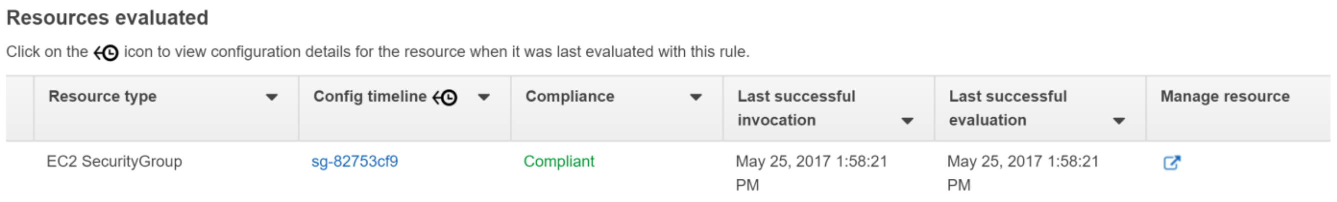
NOTE: Any edits made on existing rules will result in the edited rule being deleted and a new rule created with the new details. This will cause traffic that depends on that rule to be dropped for a very brief period of time until the new rule can be created.

Cancel Save

16. Click **Save**.

17. Navigate back to **Service** > **Management Tools** > **Config** > **Rules** and click on **disallow-rdp-ingress**.

The rule is configured to evaluate on configuration changes. After a few minutes the configuration change will be recorded and by refreshing the page, you will see the security group has changed to the **Compliant** state:

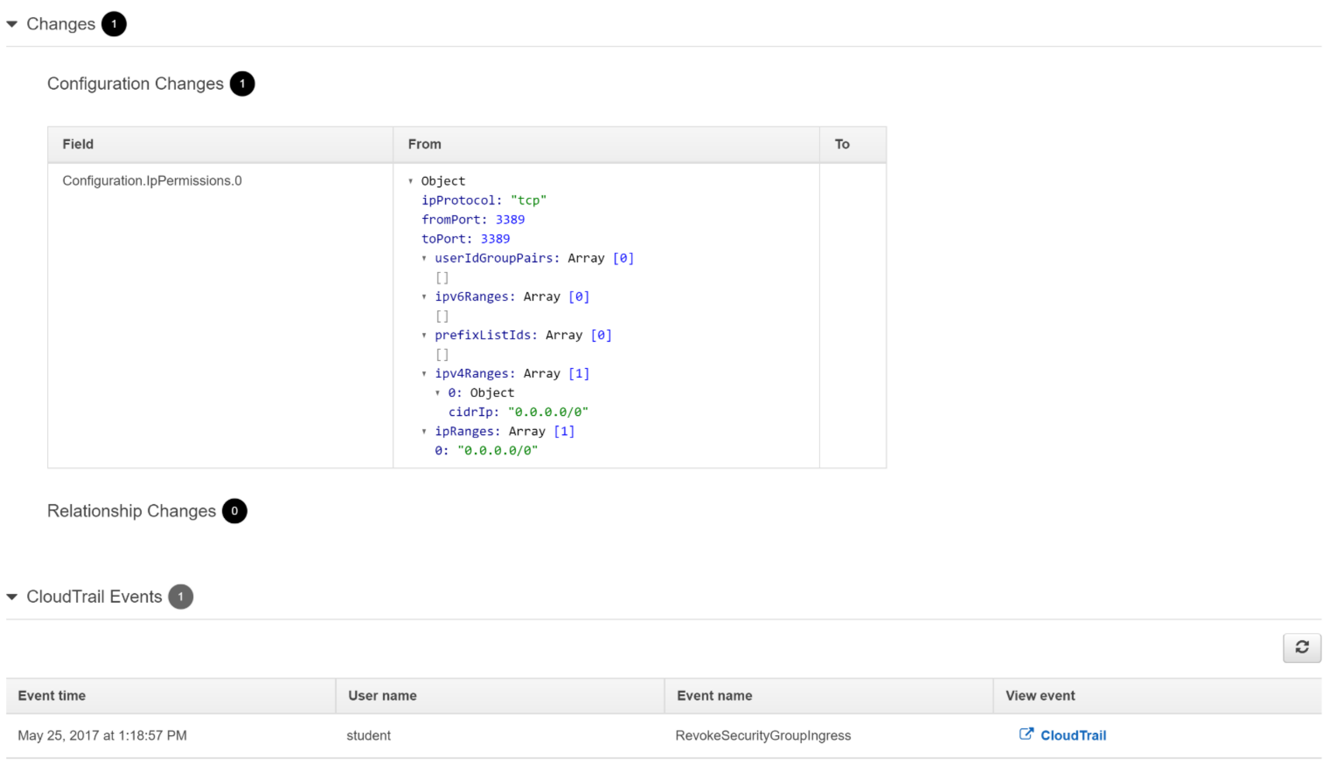

Resources evaluated

Click on the icon to view configuration details for the resource when it was last evaluated with this rule.

Resource type	Config timeline	Compliance	Last successful invocation	Last successful evaluation	Manage resource
EC2 SecurityGroup	sg-82753cf9	Compliant	May 25, 2017 1:58:21 PM	May 25, 2017 1:58:21 PM	

Note: You can click on the **Re-evaluate** button further down the page to check the rule immediately. This will allow the **Compliance** state to be updated, but it won't speed up the time it takes for configuration items to be recorded in the next instruction. This is because the code above directly queries the security group instead of depending on a configuration item.

18. Click on the security group name under the **Config timeline**:


Changes 1

Configuration Changes 1

Field	From	To
Configuration.IpPermissions.0	▾ Object ipProtocol: "tcp" fromPort: 3389 toPort: 3389 ▾ userIdGroupPairs: Array [0] [] ▾ ipv6Ranges: Array [0] [] ▾ prefixListIds: Array [0] [] ▾ ipv4Ranges: Array [1] ▾ 0: Object cidrIp: "0.0.0.0/0" ▾ ipRanges: Array [1] 0: "0.0.0.0/0"	

Relationship Changes 0

CloudTrail Events 1

Event time	User name	Event name	View event
May 25, 2017 at 1:18:57 PM	student	RevokeSecurityGroupIngress	CloudTrail

You can see the **Configuration Changes** section recorded the removal of the RDP IpPermission (ingress rule). Further down under **CloudTrail Events**, the **RevokeSecurityGroupIngress** event that triggered the change has been recorded in CloudTrail and linked to by Config.

Summary

In this Lab Step, you created a custom rule for AWS Config. The rule used a lambda function that detects if a security group is allowing incoming tcp traffic over the default RDP port. You also got more practice investigating noncompliant resources and modifying configurations to bring about compliance.