Lab 4 – Governance – Automating Security

As you’ve seen thus far, deploying a secure environment via CloudFormation is a fairly simple process and, since you’re executing the same exact code, will deploy the same secure environment every time the template is run. However, once an environment is deployed, businesses need to ensure that future changes don’t negatively impact their security posture. With the addition of automation, businesses can reduce common pitfalls that could put their data and infrastructure at risk.

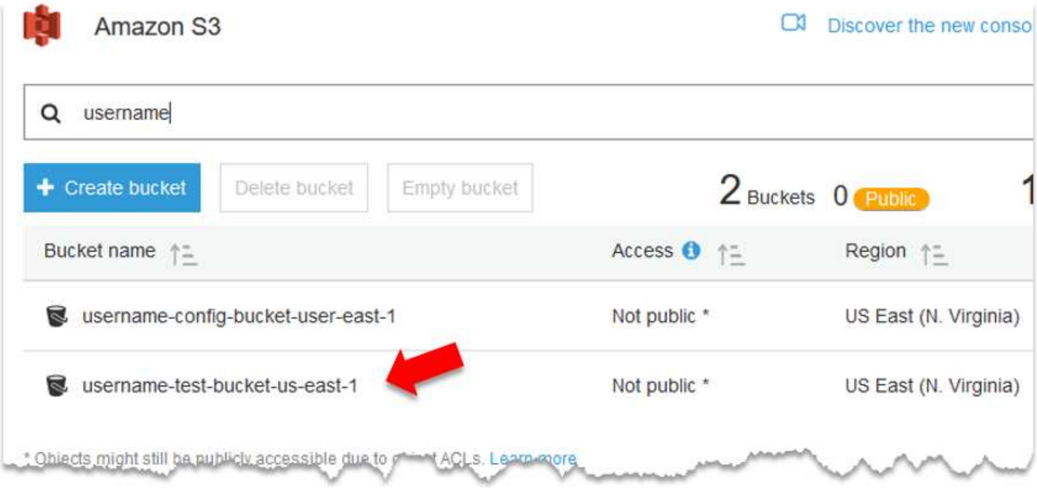
Goal – This lab will demonstrate how to use AWS native tools to help ensure that the ACL for an S3 bucket doesn’t allow public access to the data held within it. Lab participants will purposely enable public access to their S3 bucket and see the automatic remediation driven by AWS Config, AWS Lambda, and Amazon SNS.

Prerequisites - Access to the AWS VPC console with an IAM user with appropriate permissions to add S3 bucket tags and configure Amazon SNS.

1. Navigate to the **S3** console.

2. Locate the test S3 bucket that was created during the initial deployment. The bucket name should look something like: ***username.*****test-bucket.us-east-1**.

   Notice in the console that the access shows as **Not public***.



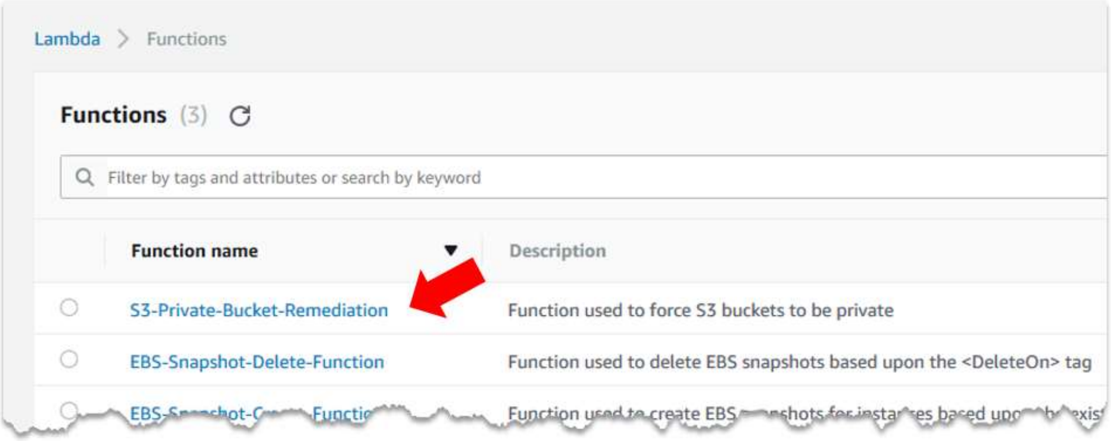
Bucket name	Access	Region
username-config-bucket-user-east-1	Not public *	US East (N. Virginia)
username-test-bucket-us-east-1	Not public *	US East (N. Virginia)

* Objects might still be publicly accessible due to object ACLs. Learn more

3. There will also be a second bucket named ***username.*****config-bucket-us-east-1**. This bucket will be where AWS Config logs will be stored. If you’d like, you can click on the bucket to view the folder structure and log files.

4. Navigate to the **Lambda** console, located in the Compute section.

5. In the console, notice that you've deployed several Lambda functions already. The Lambda function should be called **S3-Private-Bucket-Remediation** and have a description of **Function used to force S3 buckets to be private**. Also notice that the runtime environment is Python 3.6, which means the function is written using Python. Feel free to open the function and review the code.



Lambda > Functions

Functions (3)

Filter by tags and attributes or search by keyword

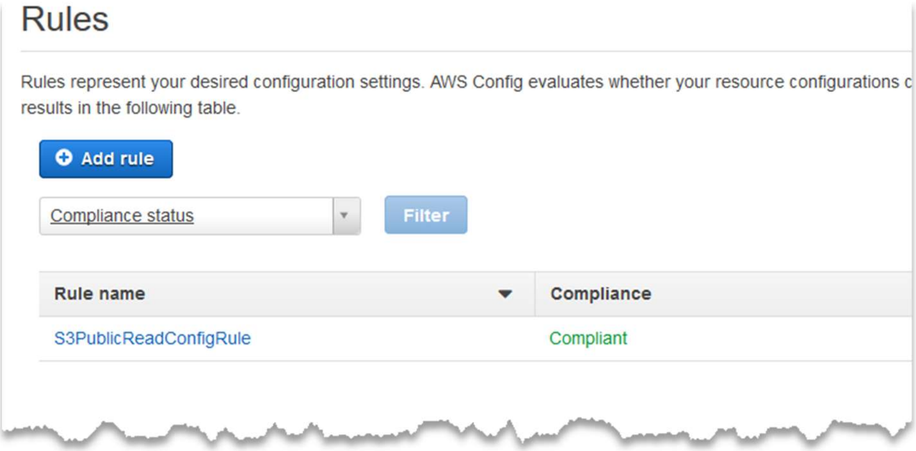
	Function name	Description
○	S3-Private-Bucket-Remediation	Function used to force S3 buckets to be private
○	EBS-Snapshot-Delete-Function	Function used to delete EBS snapshots based upon the <DeleteOn> tag
○	EBS-Snapshot-Create-Function	Function used to create EBS snapshots for instances based upon the exis

6. From the Services menu, navigate to the **Config** console. Config is located under the Management Tools section.

7. CloudFormation has already set up resources and rules in Config, but let's take a look at what already exists. On the dashboard, notice that it has Config rule compliance widgets on the right, indicating that the one Config rule that is configured is currently Compliant. Let's take a look at the Config rule itself.

8. Click on **Rules** from the navigation pane on the left. Notice that you have one rule configured, named **S3PublicReadConfigRule**, and currently reporting as Compliant.

__Note__: This rule is configured to scan S3 bucket permission to ensure they do not allow public access. If they do not, the rule reports as Compliant. If they allow public permissions, the rule is Not Complaint.



Rules

Rules represent your desired configuration settings. AWS Config evaluates whether your resource configurations c results in the following table.

Rule name	Compliance
S3PublicReadConfigRule	Compliant

9. Click on **Settings** within the Config console. The settings page is where all the core AWS Config settings will be configured. Notice that CloudFormation has already configured Config to store the configuration history and files in the ***username*-config-bucket-us-east-1** bucket as previously indicated.

10. **Note the name** of the SNS topic used to stream Config logs and send out notifications (email, SMS, etc.)

    ***Note**: If AWS Config was already configured in your account, and you indicated that on the initial template deploy, you may not have an SNS Topic configured. If not, click the checkbox to **Stream configuration changes and notifications to an Amazon SNS topic** and select **Choose a Topic from your account**. Select the topic named **Innovation-Day-Config-Topic**.*

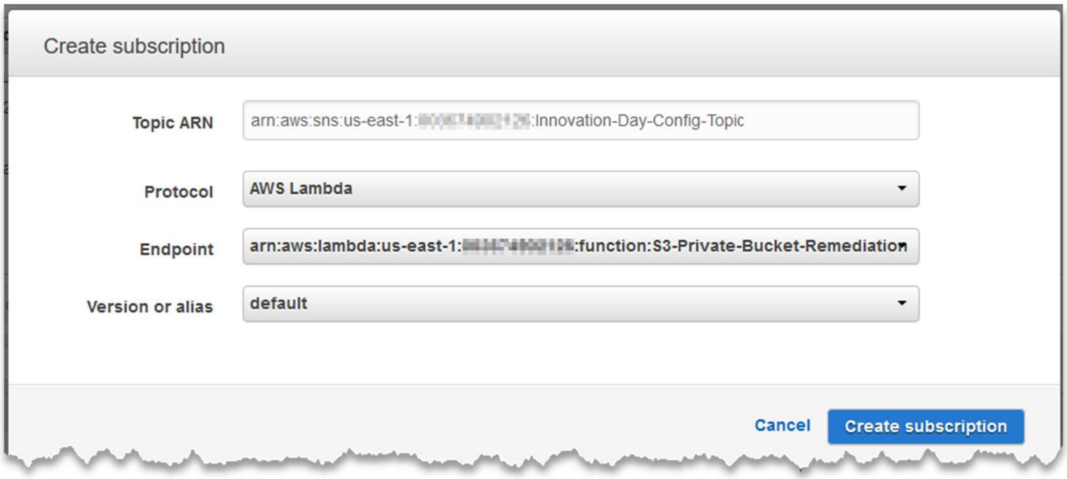
11. Navigate to the **Simple Notification Service (SNS)** console. It can be found under the Application Integration section.

12. On the left, click **Topics**.

13. Select the hyperlink on the ARN next to the **Innovation-Day-Config-Topic** found earlier in the lab.

14. Under the Subscriptions section, click **Create subscription**.

15. In the pop-up box, select **AWS Lambda** in the Protocol drop-down. Select the AWS Lambda function in the Endpoint drop-down named **S3-Private-Bucket-Remediation** as part of the endpoint name.



Create subscription

Topic ARN: arn:aws:sns:us-east-1:[redacted]:Innovation-Day-Config-Topic

Protocol: AWS Lambda

Endpoint: arn:aws:lambda:us-east-1:[redacted]:function:S3-Private-Bucket-Remediation

Version or alias: default

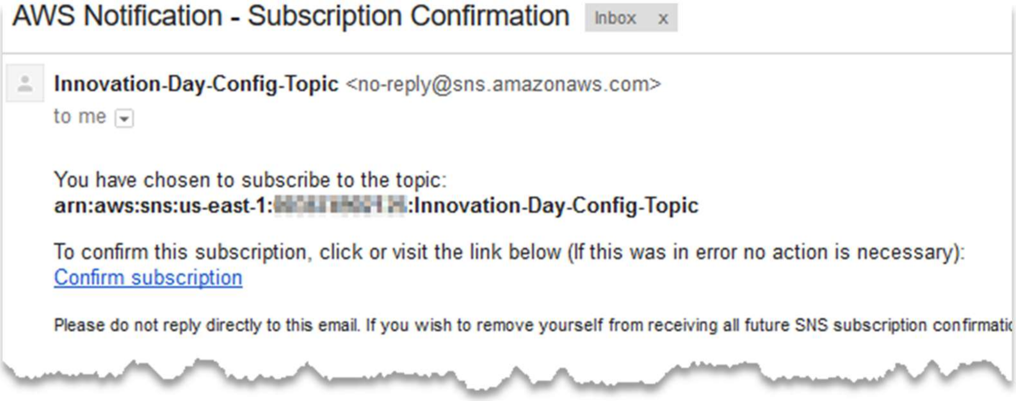
16. Click the **Create subscription** button.

17. Click **Create Subscription** again to add a second subscription.

18. Under Protocol, select **Email**. **Enter an email address** that you have access to during the lab.

19. Click **Create Subscription**.

20. Login to your email and **confirm the subscription** by clicking the link in the email from the SNS topic.



AWS Notification - Subscription Confirmation

Innovation-Day-Config-Topic <no-reply@sns.amazonaws.com>

to me

You have chosen to subscribe to the topic:

arn:aws:sns:us-east-1:[redacted]:Innovation-Day-Config-Topic

To confirm this subscription, click or visit the link below (If this was in error no action is necessary):

Confirm subscription

Please do not reply directly to this email. If you wish to remove yourself from receiving all future SNS subscription confirmatic

21. Navigate to the **S3** console.

22. In the console, click on the **test bucket** - it should be named: ***username*.test-bucket-us-east-1**.

    Note: *There are two buckets similarly named for lab purposes. Make sure to select the bucket with 'test-bucket' as part of the name.*

23. Select the **Properties** tab and click on the **Tags** box. Notice the Tags associated with this S3 bucket. For this lab, you're going to focus on the tag with the key of **Data_Classification**. The value of this tag is **Private**, meaning that any data in this bucket should not be publically accessible.

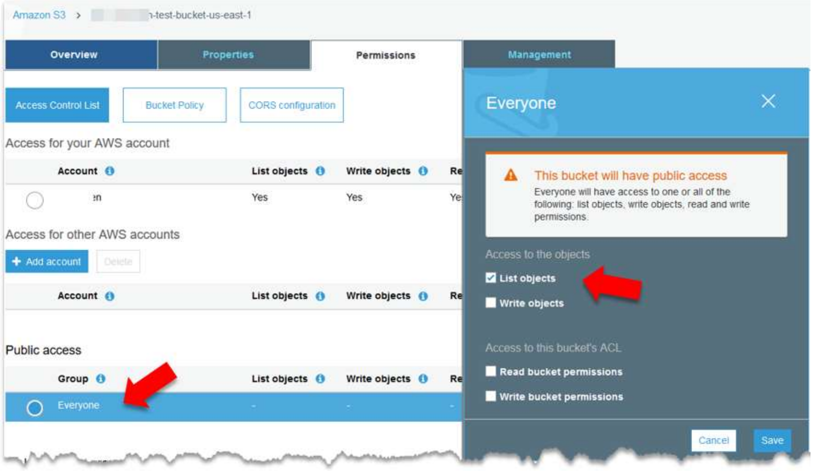
    Note*: This tag's key-value pair is a simple example for the lab. In a production environment, you may have different tags and a predefined list of values to identify data classification for the data within an S3 bucket.*

24. Click on the **Permission** tab.

25. Under the Public access section, click the *large* radio button next to **Everyone**.

26. Check the box next to **List Objects**. Click **Save**.

    Note: *As indicated, this bucket is now publically accessible to* ***ANYBODY*** *on the Internet. Please do not put any data in this bucket that shouldn't be shared with the world.*



Amazon S3 > [redacted]n-test-bucket-us-east-1

Access for your AWS account

	Account	List objects	Write objects	Re
○	[redacted]n	Yes	Yes	Ye

Access for other AWS accounts

Account	List objects	Write objects	Re

Public access

	Group	List objects	Write objects	Re
○	Everyone	-	-	-

Everyone

This bucket will have public access
Everyone will have access to one or all of the following: list objects, write objects, read and write permissions.

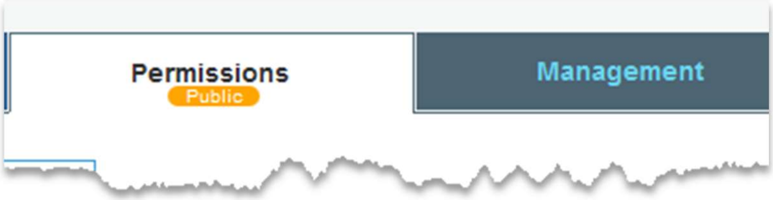
Access to the objects

☑ List objects
☐ Write objects

Access to this bucket's ACL

☐ Read bucket permissions
☐ Write bucket permissions

27. Notice the Permissions tab now has an orange icon that says **Public** on it. The icon is a visual aid to let you know that objects in this bucket are publically available.



28. Navigate back to the **AWS Config** console.

    On the dashboard, you may notice that the rule still shows in compliance. As AWS Config doesn't report in real-time, it takes a while to evaluate changes in the environment. To speed up the lab, let's force a new evaluation.

29. Click **Rules** in the navigation pane to the left.

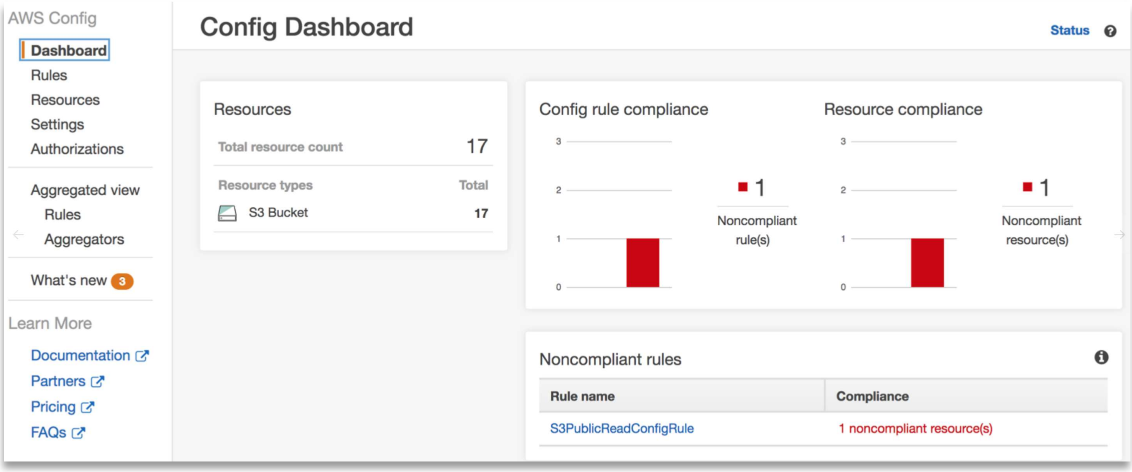
30. Select the **S3PublicReadConfigRule**.

31. Select the **Re-evaluate** button to force Config to check the status of the resources.

    If you configured an email address within the SNS topic, you should receive an email with information about the evaluation and any compliance issues.

32. Shortly, you should be able to **refresh** the page, and the Config rule will now show that it is Noncompliant.

    Note: *It may take up to 5 minutes for the rule to evaluate properly – but don't wait too long or the Lambda function will modify the permissions, and the rule will be compliant again.*



Resources

Total resource count	17
Resource types	Total
S3 Bucket	17

Noncompliant rules

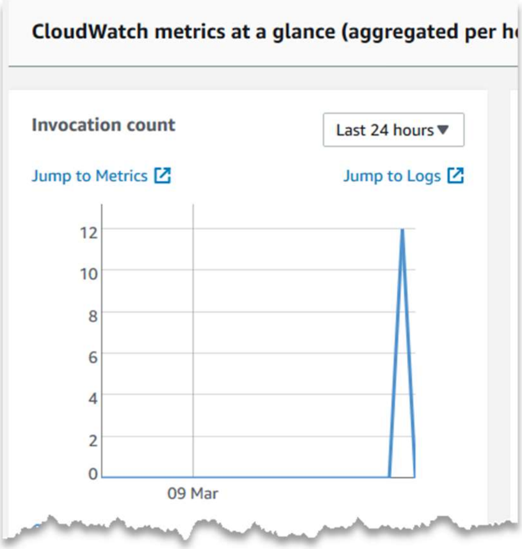
Rule name	Compliance
S3PublicReadConfigRule	1 noncompliant resource(s)

33. Navigate back to the **Lambda** console.

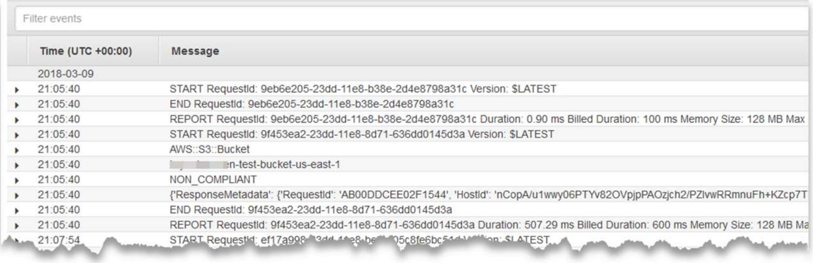
34. Under the Functions menu, locate the **S3 function** you evaluated earlier and **click the name** to open the function properties.

35. Click on the **Monitoring** tab.

36. Notice that the Invocation count is greater than **0**, meaning that the Lambda function has executed at least once.



37. Click **Jump to Logs** in the Invocation count metric window. You'll be redirected to CloudWatch Logs where you can view information about what the Lambda function did, but more importantly – it was executed.

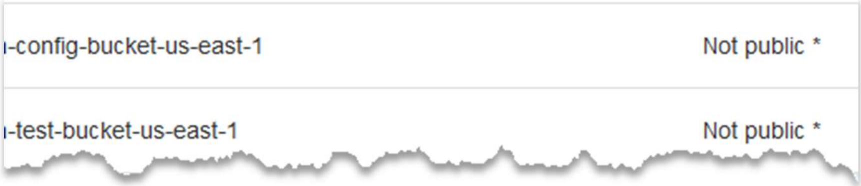


```
Filter events

     Time (UTC +00:00)   Message
     2018-03-09
▸    21:05:40            START RequestId: 9eb6e205-23dd-11e8-b38e-2d4e8798a31c Version: $LATEST
▸    21:05:40            END RequestId: 9eb6e205-23dd-11e8-b38e-2d4e8798a31c
▸    21:05:40            REPORT RequestId: 9eb6e205-23dd-11e8-b38e-2d4e8798a31c Duration: 0.90 ms Billed Duration: 100 ms Memory Size: 128 MB Max
▸    21:05:40            START RequestId: 9f453ea2-23dd-11e8-8d71-636dd0145d3a Version: $LATEST
▸    21:05:40            AWS::S3::Bucket
▸    21:05:40            [redacted]n-test-bucket-us-east-1
▸    21:05:40            NON_COMPLIANT
▸    21:05:40            {'ResponseMetadata': {'RequestId': 'AB00DDCEE02F1544', 'HostId': 'nCopA/u1wwy06PTYv82OVpjpPAOzjch2/PZlvwRRmnuFh+KZcp7T
▸    21:05:40            END RequestId: 9f453ea2-23dd-11e8-8d71-636dd0145d3a
▸    21:05:40            REPORT RequestId: 9f453ea2-23dd-11e8-8d71-636dd0145d3a Duration: 507.29 ms Billed Duration: 600 ms Memory Size: 128 MB Ma
▸    21:07:54            START RequestId: ef17a998 [illegible] 5c8fe6bc [illegible] $LATEST
```

38. Navigate back to the **S3** console

39. Does the S3 bucket still have public access?



ı-config-bucket-us-east-1	Not public *
ı-test-bucket-us-east-1	Not public *

You Did It. You're done with this Lab.

Conclusion: In this lab, participants used AWS Config, AWS Lambda, and Amazon SNS to enable automatic remediation of an S3 bucket that was purposely configured all allow public access. Based on the results of AWS Config, the Lambda function modified the ACL of the bucket to remain private.