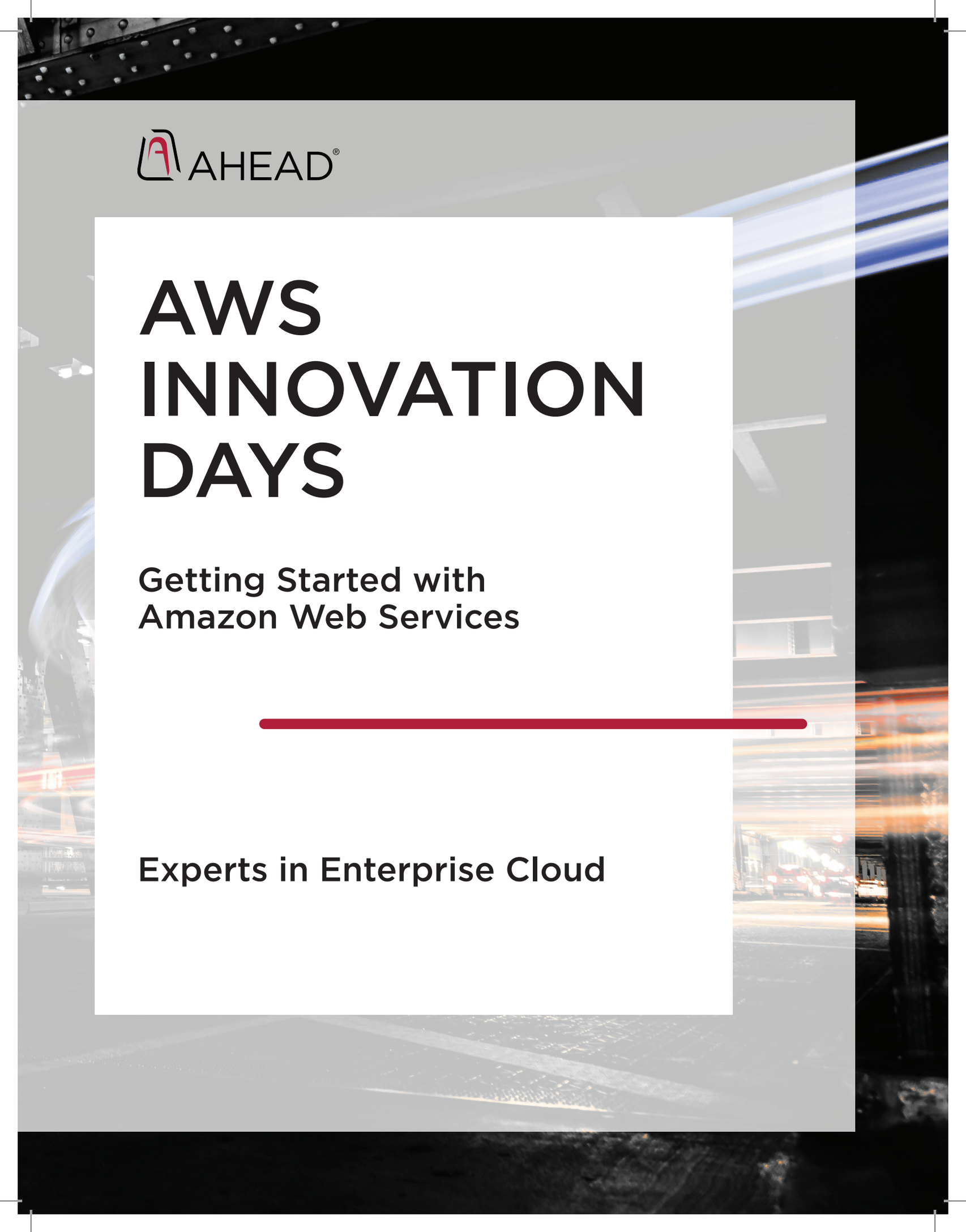


AHEAD®

AWS INNOVATION DAYS

Getting Started with Amazon Web Services

Experts in Enterprise Cloud

AHEAD Innovation Days

AWS – Automation, Security, and Governance

LOCATION

AWS Offices
150 W. Jefferson Avenue,
10th Floor
Coney Island Room
Detroit, MI 48226

TIME

April 24, 2018
1:00pm to 4:30pm

AHEAD ATTENDEES

Bryan Krausen,
Sr. Solutions Architect

Eric Shanks
Sr. Solutions Architect

Greg Thursam,
Solutions Principal
AWS and DevOps

Adam Youngblood,
Technical Architect

Workshop Purpose

With the acceleration of cloud adoption in the enterprise, IT professionals require a different skill set to manage and operate workloads in the cloud. As a follow-up our **Getting Started with AWS** workshop, participants will learn key skills to ensure that critical workloads operate efficiently and company data remains secure on the AWS platform. The use of the native AWS tools, along with the automation and self-healing concepts used in the labs, will help prepare participants to manage AWS infrastructure for the foreseeable future.

Agenda

- **Lab 1 – AWS CloudFormation Overview**
  - In this lab, participants will use CloudFormation to create the base infrastructure and additional AWS resources to use for subsequent labs. The lab will make use of nested CloudFormation templates to deploy the resources.
- **Lab 2 – Infrastructure as Code - Operations**
  - In this lab, participants will learn different techniques and use cases on how to manage changes to your automated AWS environment. Users will learn how to update existing stacks and add operational controls to the environment, such as data protection.
- **Lab 3 – Image Operations**
  - In this session, participants will learn how to use AWS native services for image patch management of EC2 instances. The lab will cover the integration of multiple services and how they work together to build a comprehensive operational strategy.
- **Lab 4 – Governance – Automating Security**
  - In this lab, participants will learn how to automate the security and self-remediation of their AWS environments using AWS native tools. Using the combination of Config, SNS, and Lambda, users will ensure that their sensitive data stored in S3 remains private and out of the hands of unauthorized users.
- **Lab 5 – Auto Scaling Workloads**
  - In this session, users will learn how to deploy an application in a highly scalable fashion to support the scaling of resources on-demand. Participants will configure autoscaling for a web application and experience how changes in the environment can trigger the addition, or removal, of resources as required by the application.

Table of Contents

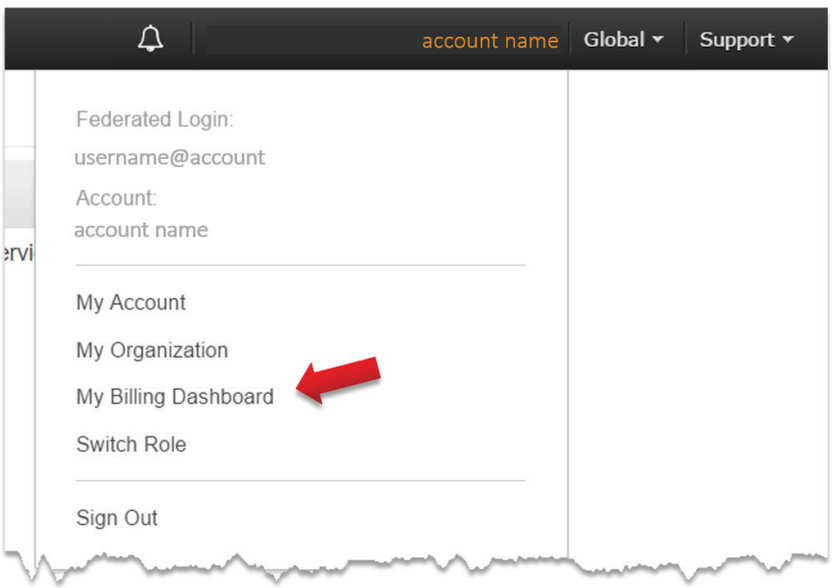
Workshop Prerequisites

To complete the labs, participants should provide:

- A laptop to access Amazon Web Services
- A pre-existing AWS account with administrative privileges

Adding a Promotional Credit to Your Account

1. After logging into your account, click your username name in the top right corner and select **My Billing Dashboard**.



2. On the left, open the **Credits** page of the Billing and Cost Management console.
3. In the **Promo Code** box, type the promotional code provided by lab proctor.
4. In the box labeled **Please type the characters as shown above**, type the code.
5. Choose **Redeem**.

   If the promotional code was typed correctly, is a valid code, hasn't expired, and hasn't been previously redeemed, it is added automatically to your AWS account.

Lab 1- AWS CloudFormation Overview

AWS CloudFormation – CloudFormation provides AWS users with a simple way to describe and provision infrastructure and resources within the AWS environment. By writing CloudFormation templates using YAML or JSON, users can create, modify, or destroy resources in an automated and secure manner that are required by your applications. Often referred to as Infrastructure as Code, CloudFormation can be used to codify your entire cloud environment without having to perform manual actions or write custom scripts for deploying applications. This also allows the user to standardize infrastructure components across accounts, regions, or business units.

Goal - Create the baseline infrastructure for subsequent labs using AWS CloudFormation.

Prerequisites - An active AWS account with a valid payment method.

---

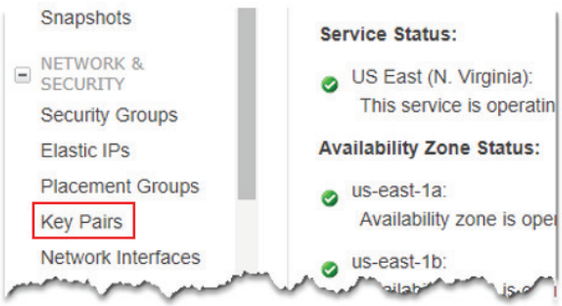
1. Log in to the **AWS console** using the URL:

   https://console.aws.amazon.com/console/home

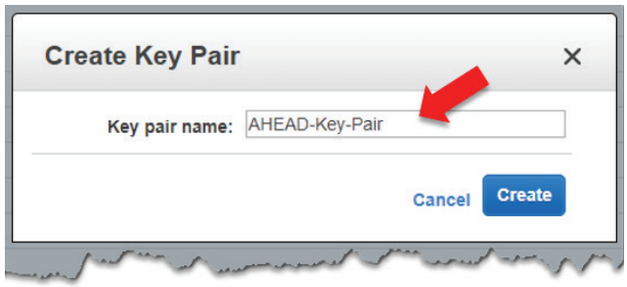
   Note: *While the preferred method of managing day-to-day operations of an AWS account is the use of Federated roles and policies, meaning the use of an external Identify Provider to log onto the AWS platform, feel free to use an IAM account of your choice as long as the account to be used has been assigned the appropriate permissions.*

2. Once logged in, click on Services and select EC2 under the Compute section:

3. On the left pane, select Key Pairs under the NETWORK & SECURITY section:



4. Click **Create Key Pair** at the top of the console and provide a **Key pair name** in the pop-up box that appears (example: AHEAD-Key-Pair). This key will be used to encrypt and decrypt the local administrator/root password for the EC2 instances that will be using throughout the labs and will be needed to log in to the instances:

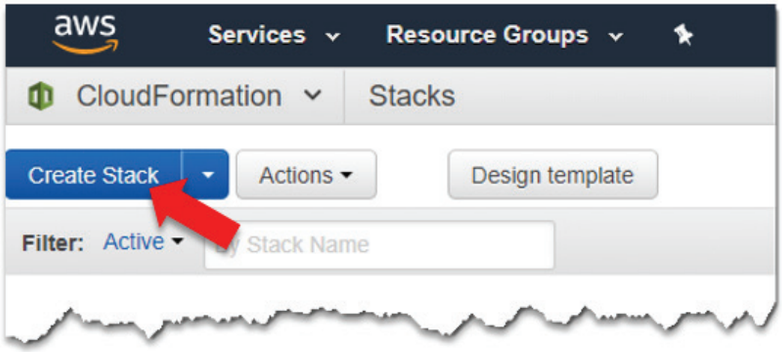


5. Click **Create**.

6. The **.pem file** will be automatically download – stored it somewhere familiar, yet safe, as you'll need it for future labs.

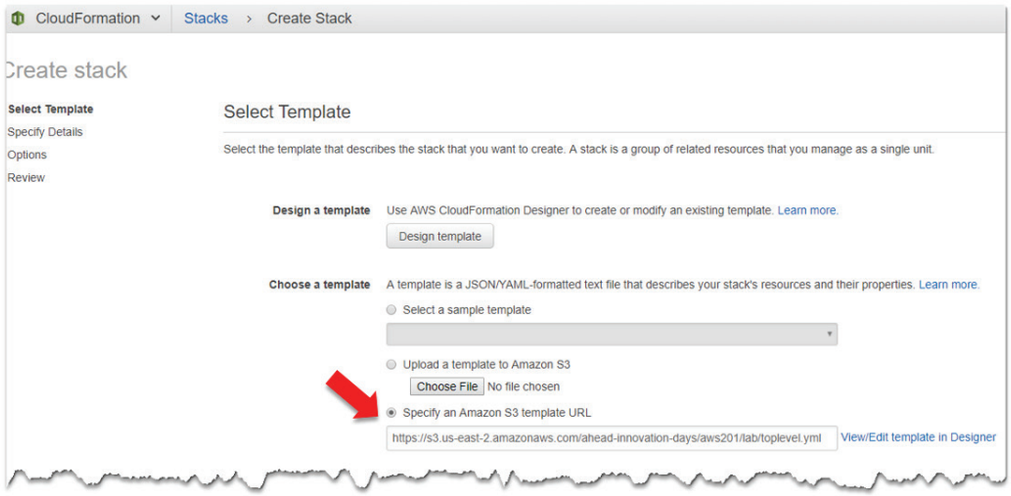
7. Now that you've created a new Key Pair, let's get started on deploying the initial CloudFormation template. Click on the Services menu at the top and select **CloudFormation** under the Management Tools section:

8. Let's create a new stack using the supplied templates provided by AHEAD. The supplied template is written using nested stacks, in which CloudFormation will automatically create child stacks for each component when deploying the template. In the CloudFormation console, click **Create Stack**:



9. Under Choose a template, select the option Specify an Amazon S3 template URL and enter the following URL in the text box:

   https://s3.us-east-2.amazonaws.com/ahead-innovation-days/aws201/lab1/toplevel.yml

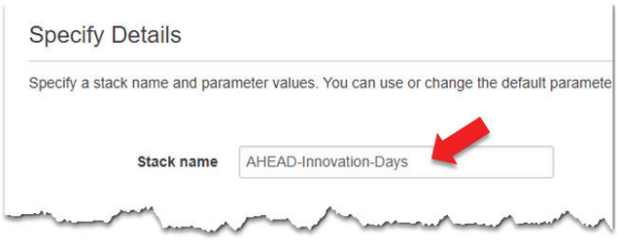


10. Click **Next** to proceed.

    ***Note**: CloudFormation will perform template validation before allowing you to continue. For example, if your template is formatted incorrectly, it will alert you to the error in which you must address to continue.*

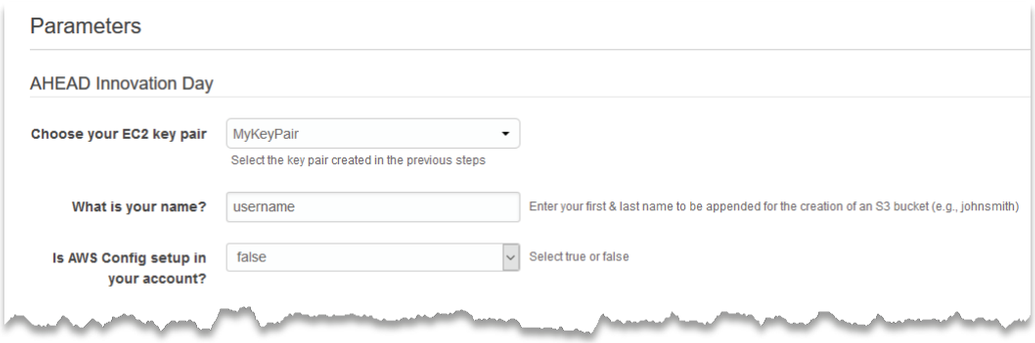
11. Give this stack a name - enter **AHEAD-Innovation-Days** as the Stack name.

    ***Note**: CloudFormation Stack names may only contain letters, numbers, and dashes. They cannot include a <space> and must start with an alpha character.*



12. For this template, you will be using input Parameters to pass values to the template to properly create the desired resources. In this case, you're going to be entering several parameters here, including selecting the EC2 Key Pair created in previous steps. Select your EC2 Key Pair from the drop-down list, type in your name, and select true or false if AWS Config is already configured in your account:

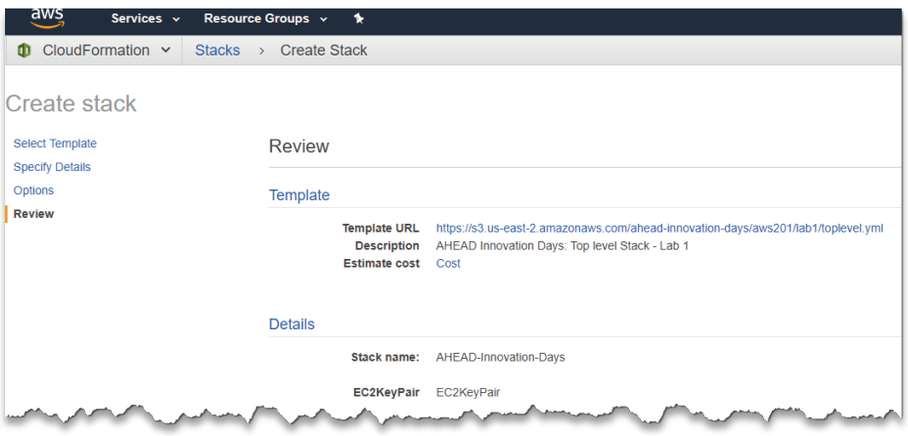
    Note: *The selection of True/False for Config will dictate how CloudFormation will configure AWS Config. If Config is already setup, selecting True will not configure the AWS Config recorder but will still create new Config rules for this lab.*



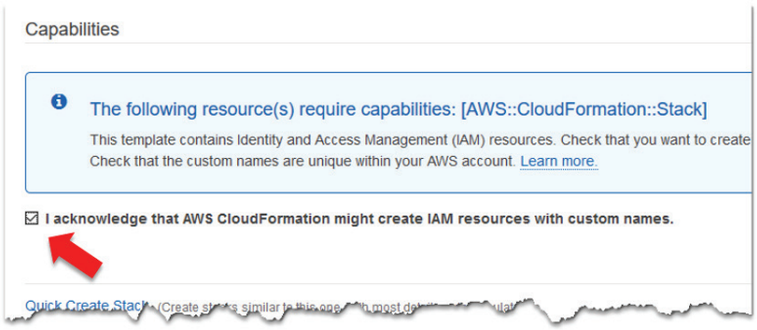
13. The **Options** page allows us to add additional information to the stacks, such as Tags, Permissions, and other Advanced Options at the bottom. For this lab, you'll leave all the defaults as they are and **click Next**.

    Note*: While you are not making changes on the Options Page, feel free to review some of the options available. The template will create a few tags for you but additional tags may be added here, if desired. Some of the other important options include "Termination Protection" and "Rollback on failure" options.*

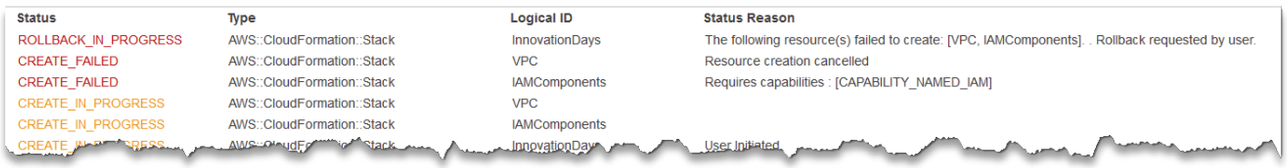
14. Review the selections on the **Review** page and make sure everything is correct. If you need to make changes, click **Previous** to go back and make the desired changes:



15. **Click the checkbox** below the blue notification which acknowledges that this template will create IAM resources in your account.



***Important**: If the above checkbox is not acknowledged, the template will proceed to execute, but you'll get an error during the deployment – if this happens, delete the stack and recreate it:*

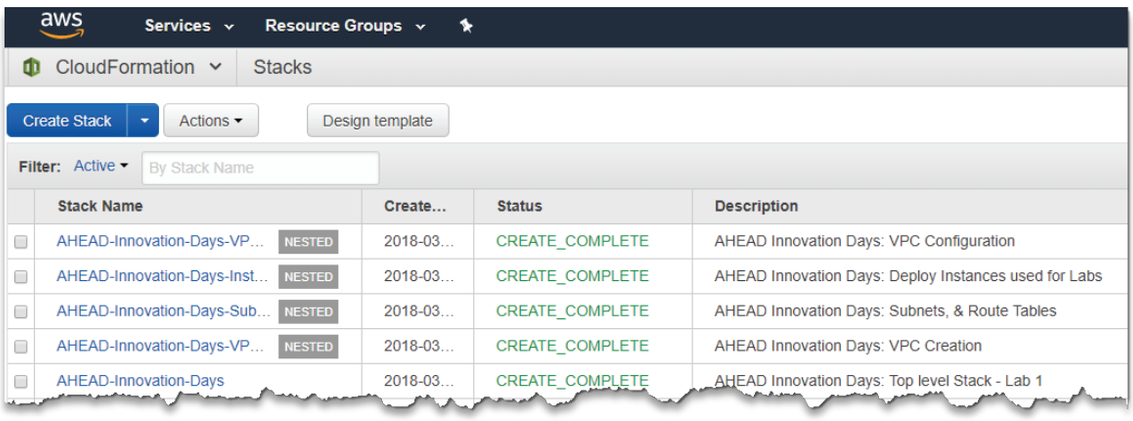


Status	Type	Logical ID	Status Reason
ROLLBACK_IN_PROGRESS	AWS::CloudFormation::Stack	InnovationDays	The following resource(s) failed to create: [VPC, IAMComponents]. . Rollback requested by user.
CREATE_FAILED	AWS::CloudFormation::Stack	VPC	Resource creation cancelled
CREATE_FAILED	AWS::CloudFormation::Stack	IAMComponents	Requires capabilities : [CAPABILITY_NAMED_IAM]
CREATE_IN_PROGRESS	AWS::CloudFormation::Stack	VPC	
CREATE_IN_PROGRESS	AWS::CloudFormation::Stack	IAMComponents	
CREATE_IN_PROGRESS	AWS::CloudFormation::Stack	InnovationDays	User Initiated

16. If everything looks correct, let's create the stack. **Click Create**.

***Important**: As mentioned above, this template is using nested stacks which break different AWS components into their own "child stack," such as networking, security components, and compute resources. This strategy allows the use of CloudFormation while still allowing different IT teams to own and manage their respective areas of expertise within the AWS deployment.*

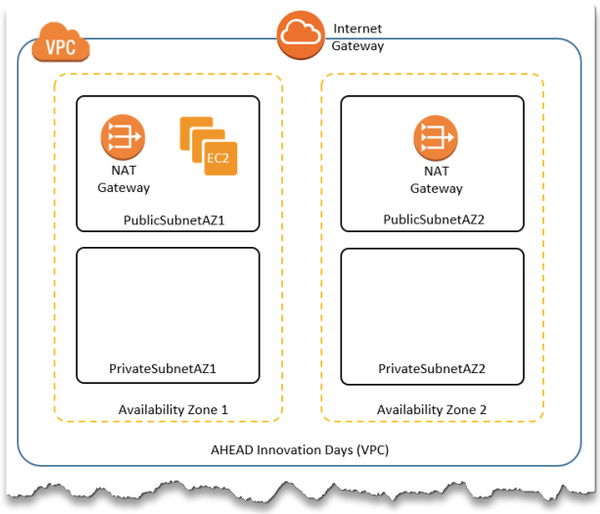
17. Once back in the CloudFormation console, you may see many different stacks deployed, each with a **NESTED** label, notating that they are a child stack beneath a top-level stack. If the deployment was successful, the console should look similar to the following screenshot:



18. Now that the stack has successful deployed let's take a look at what was deployed. Feel free to click in the console to view the different resources that were created. These include:

- VPC Creation
- Four Subnets, an Internet Gateway, two NAT Gateways
- Routing Tables and default routes
- EC2 Instances for use in subsequent labs
- Lambda functions
- SNS topics
- AWS Config Rules

For reference, the drawing below represents the VPC that was deployed using the CloudFormation template. The deployment is a standard VPC configuration using two (2) public subnets, two (2) private subnets, both of which span two Availability Zones to provide physical redundancy for applications. In addition to the VPC and subnets, two (2) NAT Gateways have been implemented to provide the Internet access for any workloads residing in the private subnets.



That's it. You've completed this lab.

Conclusion: In this lab, participants deployed a nested CloudFormation stack, which created and configured some AWS resources for the proceeding labs.

Note: Don't delete the stack. All these resources will be used in the next few labs.

Lab 2- Infrastructure as Code - Operations

Day 2 Operations - While Infrastructure as Code provides us the ability to quickly create resources across infrastructure, it's important that modifications to the existing infrastructure are also made through code as well. Rather than a manual process, implementing changes through code ensures consistency while providing a path to continuous integration and automated testing. The initial templates, and modifications to any templates, generally follow processes such as code reviews and the use of version control tools, such as GitHub.

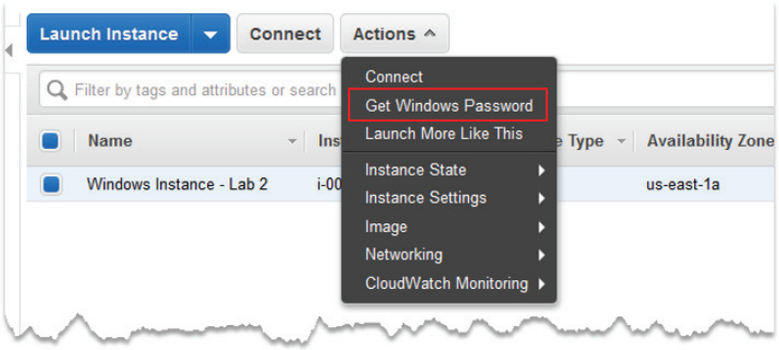
Goal – This lab is broken up into two (2) parts. The first section of this lab will teach users how to create and apply change sets to modify existing CloudFormation stacks. The second part of the lab will walk users on how to deploy additional components, using a CloudFormation template, to assist with data protection in the AWS environment.

Prerequisites – Completion of Lab 1. A Remote Desktop application to connect to a Windows Server.

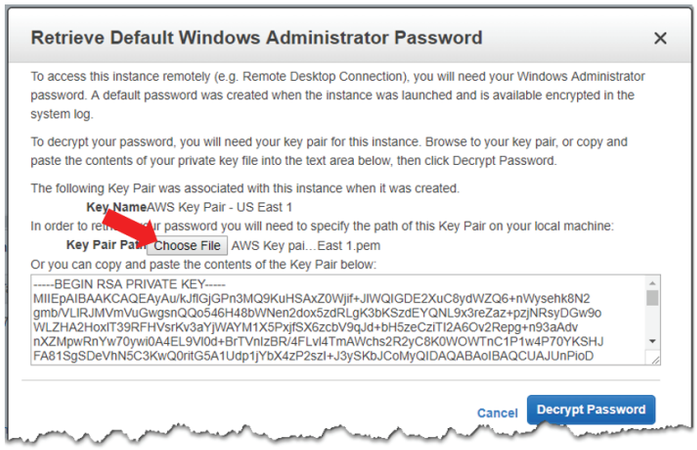
---

1. Now that you've deployed AWS resources let's try to make use of them. Navigate to the EC2 console by **Services** and selecting **EC2**. Click **Instances** on the left.

2. Prior to connecting to the instance, you need to retrieve the local administrator password. To do this, you need to decrypt it using the EC2 Key Pair that was created earlier. Select **Windows Instance – Lab 2** instance in the console, click **Actions**, and select **Get Windows Password**.

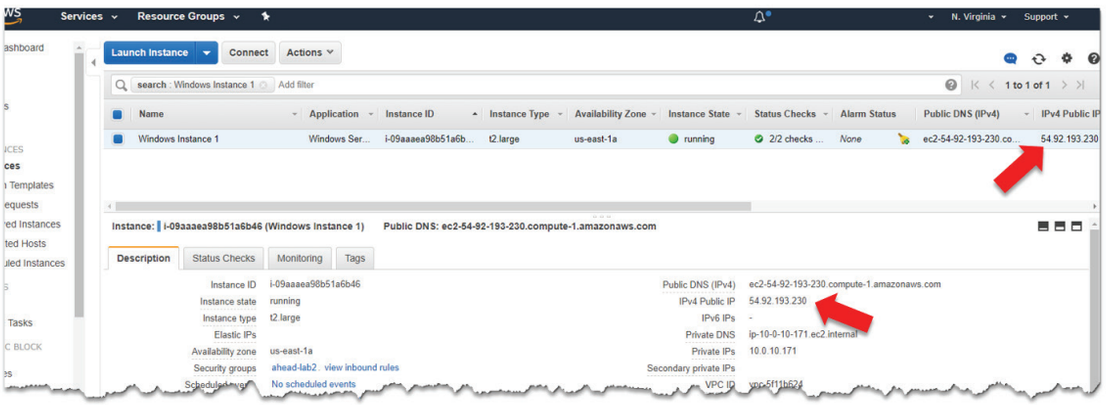


3. Select the **Choose File** button and **select the Private Key** that was downloaded in Lab 1. Alternatively, you can paste the private key into the text box. **Click Decrypt Password.**

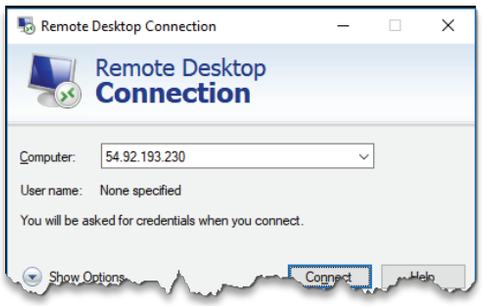


4. **Copy the password** from the box and **click Close**.

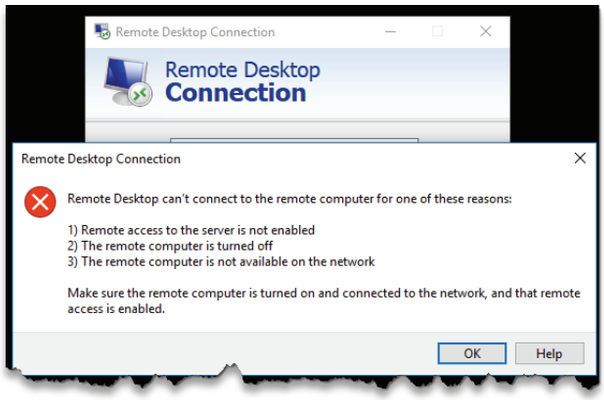
5. In the EC2 console, select the server named **Windows Instance – Lab 2** and note the public IP address.



6. As this is a Microsoft Windows 2016 Server instance, let's try to connect to it using Remote Desktop Connection (RDP). **Enter your public IP address** in the Computer field and **click the Connect button**:

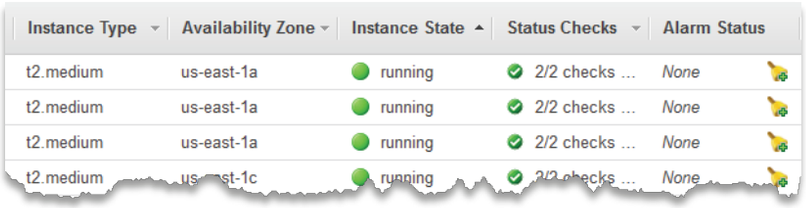


7. Can't connect? Perhaps the instance isn't deployed correctly, or there is a mistake in the network configuration. This is by design for the lab, so let's take a look at the resources deployed so you can fix this problem and manage the new instance.



8. Back in the EC2 console, **look at the information** on the EC2 console and see what's wrong:

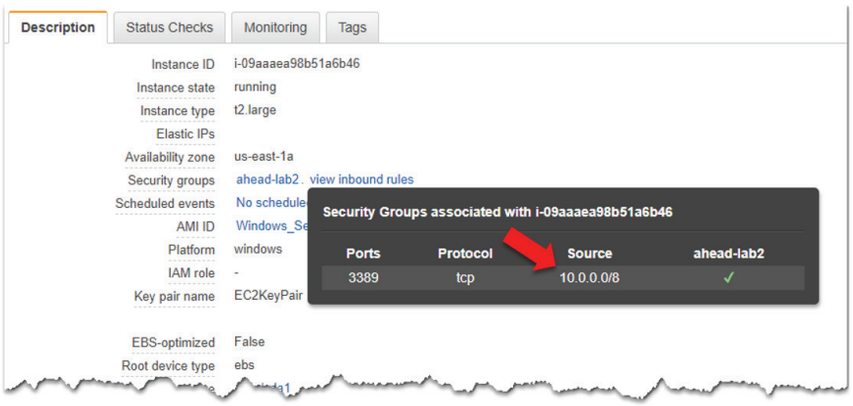
- Is the instance running as indicated by the **Instance State**? Yes
- Does the console indicate that both Status Checks have passed? Yes



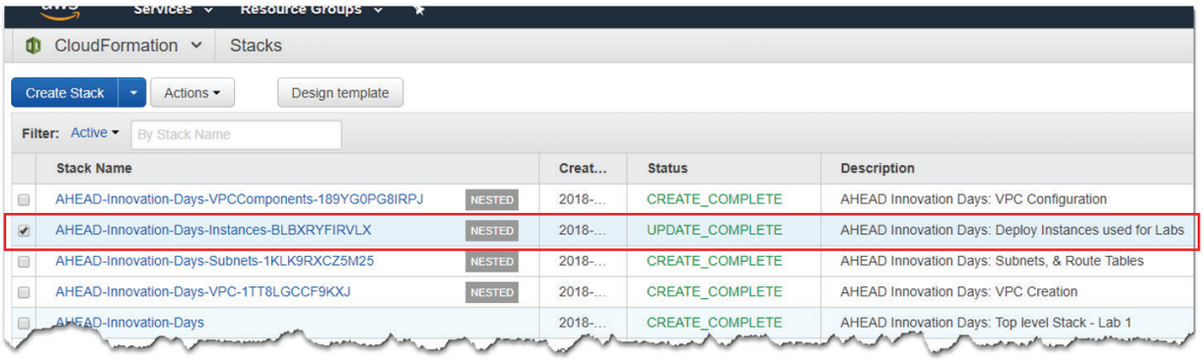
Hmm…the problem might not be the instance configuration then, so let's look at the network:

- Did you deploy an **Internet Gateway** for the VPC? Yes
- Is **routing** probably configured for internet access? Yes
- Are you allowing **tcp/3389** from the public IP address? Nope

Click on **view inbound rules** to see the current rules being applied. It looks like the **instances.yml** template successfully created the security group and the proper inbound rule, but it's only allowing RDP from private IP addresses in the **10.0.0.0/16** IP space. As you're trying to connect over the public Internet, you need to allow traffic from your **Public IP address**. While you could quickly fix this in the console, you're going to continue using CloudFormation to manage the existing lab environment to ensure consistency within the environment.

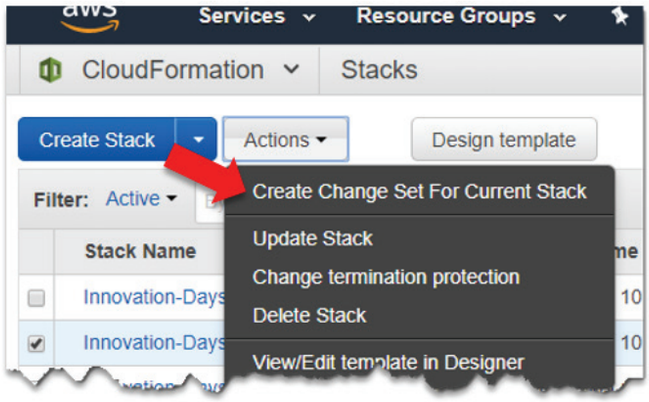


9. Click on the Services menu and navigate back to the CloudFormation console. Find the Nested Stack that deployed the Security Group and instances. You can find it by the name **AHEAD-Innovation-Days-Instances-xxxxxx** or by looking at the Description in the list:

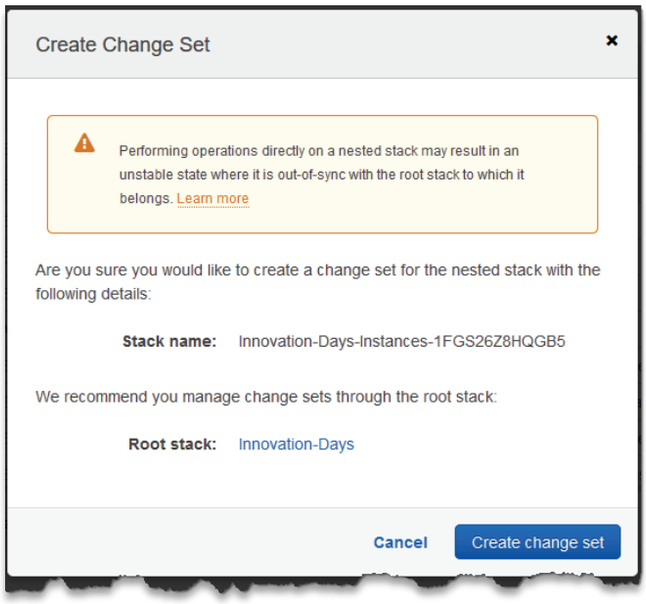


***Note**: Feel free to view the actual template to see the underlying code. You can view the code by clicking the stack name and expanding the "Template" subsection. Look for the "Resources" section and view the configuration for the Security Group – see where it defined the 10.0.0.0/16 network for the security group.*

10. You're going to create a Change Set to modify this nested stack so you can ensure that the security group is configured correctly. With the **Instances** nested stack selected, click **Create Change Set For Current Stack** from the Actions menu:



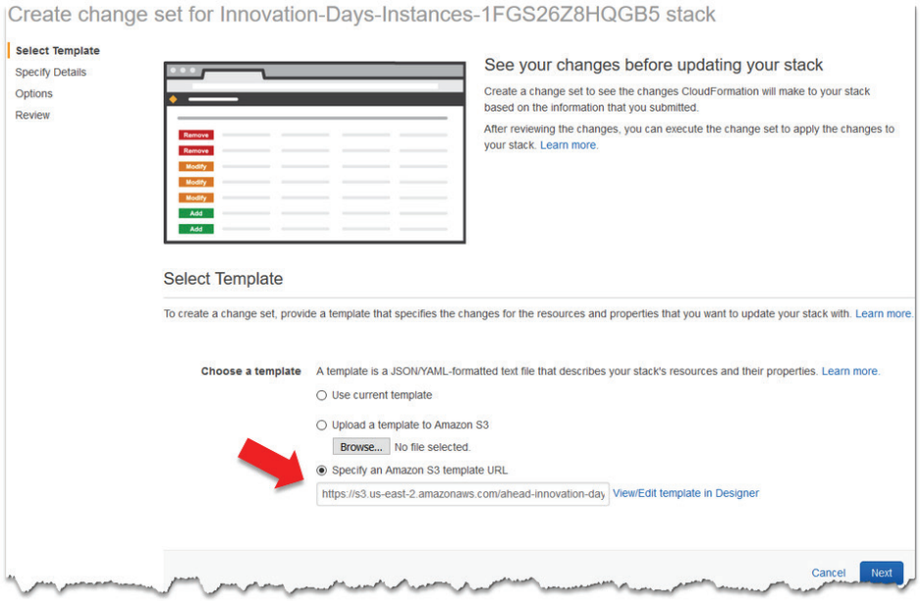
11. Read the warning that is presented and select **Create change set**:



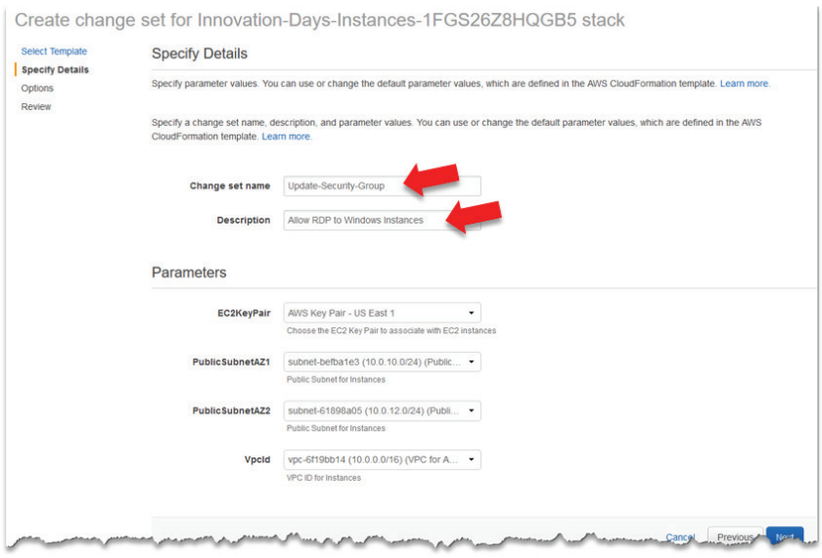
***Note**: For the sake of simplicity, you are creating a change set on a child stack in the lab. In a production environment, you'll want to create the change set on the parent stack vs. a child stack to ensure the root stack is "in-sync" with all the child stacks.*

12. Similar to deploying the original stack earlier, you need to provide the URL where the updated template is stored. Use the following for the **Specify an Amazon S3 template URL** and click **Next**:

    https://s3.us-east-2.amazonaws.com/ahead-innovation-days/aws201/lab2/instances_v2.yml



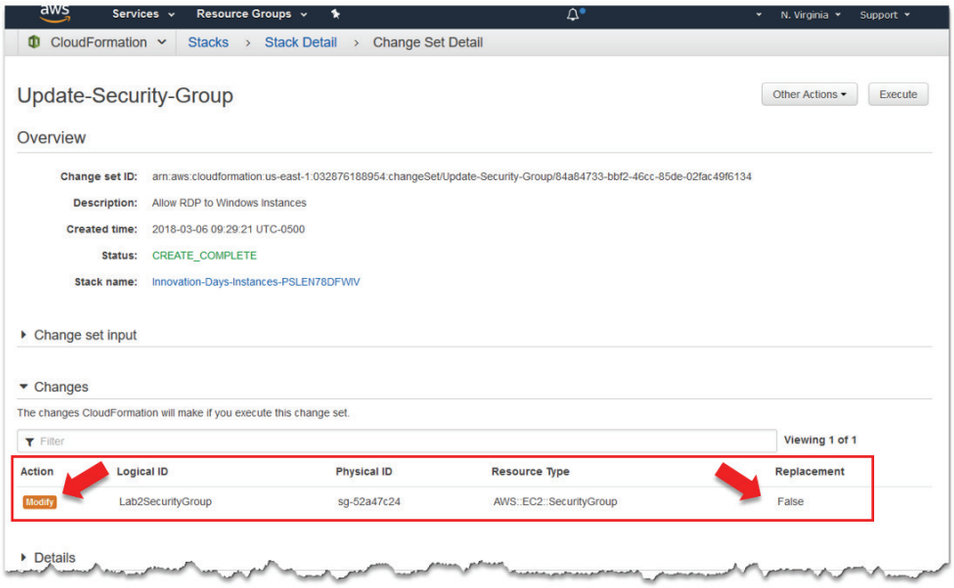
13. Enter a change set name **Update-Security-Group** and a Description **Allow RDP to Windows Instances**. Do not change any of the Parameter fields. Click **Next**.



14. On the Options page, leave all the defaults and click **Next**.

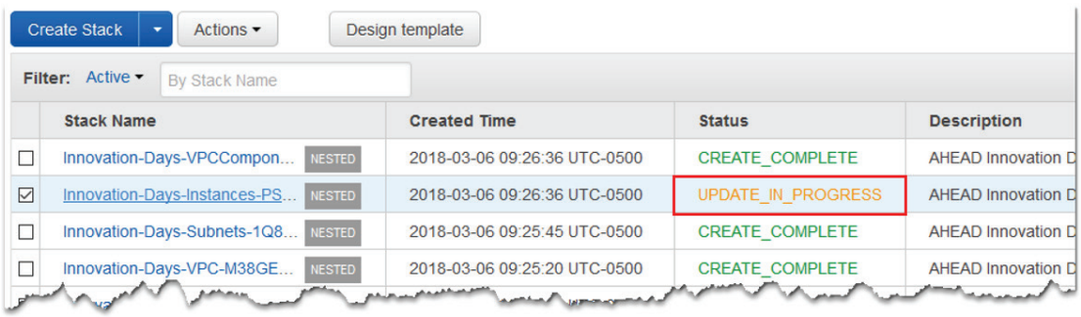
15. Review the changes to ensure accuracy. If everything looks correct, click the **acknowledgement box** for the creation of IAM roles. Click **Create change set**.

16. Once complete, you'll be on the **Change Set Detail** page which displays information about the change set you just created. The important subsection (highlighted below) is the **Changes** section, which will list the changes that will be applied to your environment. In this case, you're modifying the Security Group to allow RDP from all IP addresses (vs. just 10.0.0.0/8). Note the action for the resources is **Modify** and that it does NOT require a resource replacement as indicated.



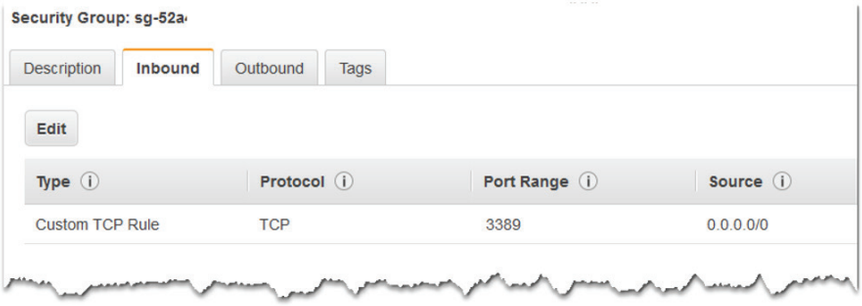
17. Now that you're aware of the impending changes to your environment let's execute the change set. Click the **Execute** button to make the desired changes. Verify you want to execute the changes by clicking **Execute** again.

    You'll be redirected back to the CloudFormation console. Take a look at the **Instances** stack and notice the status is now UPDATE_IN_PROGRESS.



18. Once the status changes to UPDATE_COMPLETE, you can check to see if you can connect to your EC2 instance via RDP. If you can connect, you've successfully executed a change set to modify your existing infrastructure to manage your EC2 instances.

    Note: *Feel free to navigate to the VPC or EC2 console to view the Security Group and verify the Inbound rules now allow port 3389 from all IP addresses rather than just 10.0.0.0/8.*



You've completed this part of the lab.

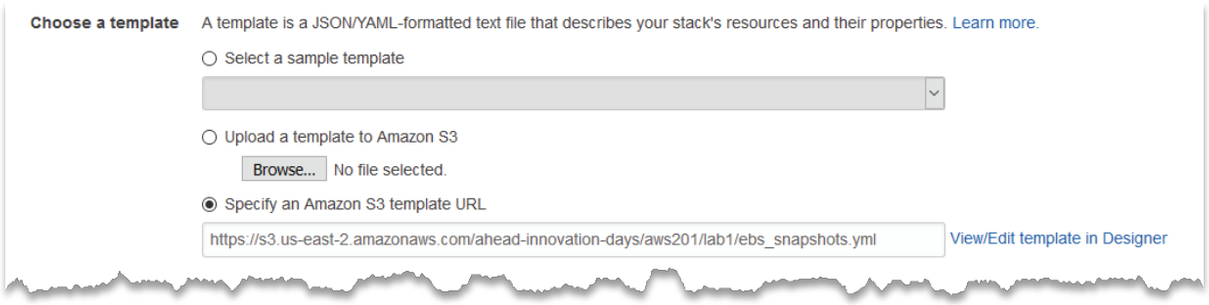
Feel free to continue to part 2 on the next page.

19. Navigate back to the **CloudFormation** console.

20. Click on the **Create Stack** button.

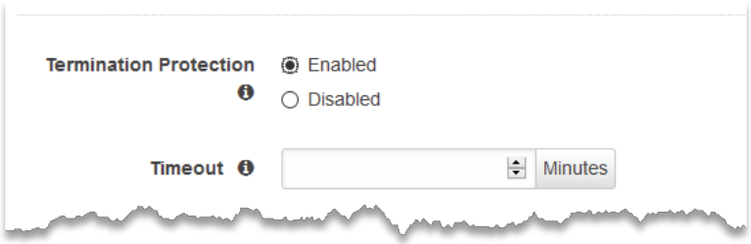
21. Select the radio button next to **Specify an Amazon S3 template URL** and enter the following URL:

    https://s3.us-east-2.amazonaws.com/ahead-innovation-days/aws201/lab2/ebs_snapshots.yml



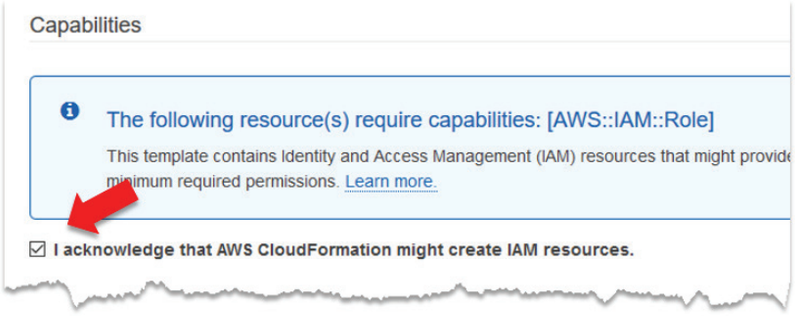
22. Enter **EBS-Snapshots** for the stack name and click **Next**

23. On the Options page, expand the **Advanced** section and select **Enabled** for Termination Protection. Click the **Next** button.



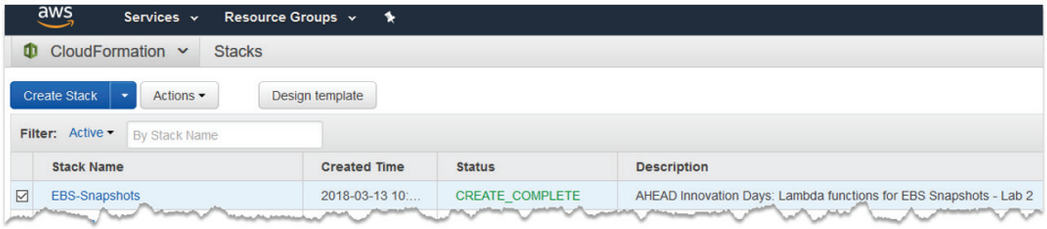
Note: *Termination protection will prevent a stack from accidentally being deleted. This setting helps ensure that critical/important stacks aren't deleted without additional intervention. It's a good idea to enable this on stacks that are creating your base infrastructure or provision any other critical resources throughout your AWS deployment.*

24. **Check the box** to acknowledge that AWS CloudFormation might create IAM resources. As this template will create an IAM role, the stack will not deploy resources correctly without this acknowledgment.



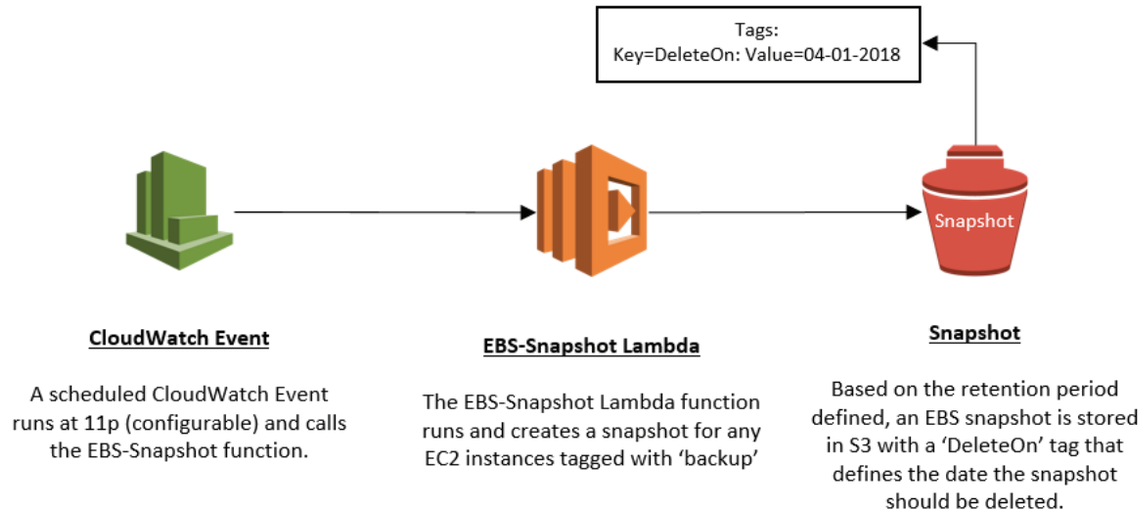
25. Click Create.

26. The stack will create an **IAM role** and **two (2) Lambda functions** that will be used to assist with creating a simple data protection strategy for the instances.



Prior to making use of the Lambda functions that were deployed, let's take a look at what they do:

EBS-Snapshot-Create-Function: This function was developed to take scheduled snapshots of EBS volumes attached to EC2 instances. When executed, the function scans EC2 instances and looks for a Tag of "Backup/backup." When it finds an instance with that Tag, it grabs the value of the "Retention" tag, if it exists. With that data, the function initiates a snapshot of the EBS volume(s) attached to the instance. The Lambda function tags the snapshot with a "DeleteOn" tag key and calculates the date for deletion based on the Retention value. The function uses seven (7) days as the default value if no Retention value is configured.



EBS-Snapshot-Delete-Function: This function was developed to delete expired snapshots to eliminate the cost associated with unneeded snapshots. When executed, the function scans all the snapshots in your account and looks for a Tag of 'DeleteOn.' If the 'DeleteOn' date matches the date the Lambda function is executed on, the function deletes the snapshot.

CloudWatch Event

A scheduled CloudWatch Event runs at 10p (configurable) and calls the Delete-Snapshot function.

EBS-Delete-Snapshot Lambda

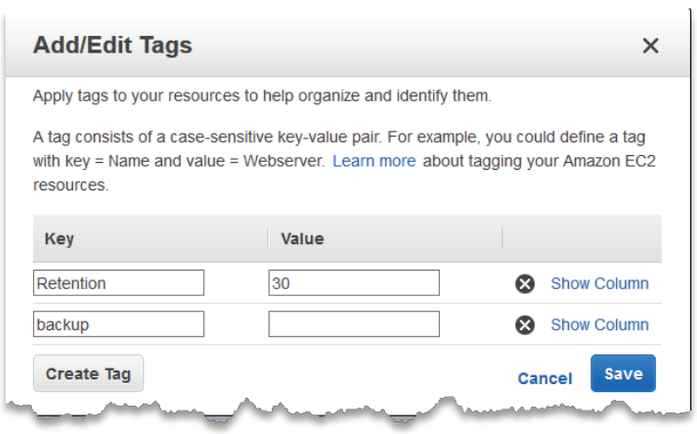
The delete Snapshot Lambda looks for any EBS snapshots that have a 'DeleteOn' tag equal to the current day and deletes the snapshot.

27. Browse to the **EC2** console and select the **Windows Instance – Lab 2** instance. Click on the Actions menu and select **Instance Settings → Add/Edit Tags**.

28. **Create two new tags** as follows:

    - **Key:** backup
      Value: <leave blank>
    - **Key:** Retention
      Value: 30

29. **Click Save.**

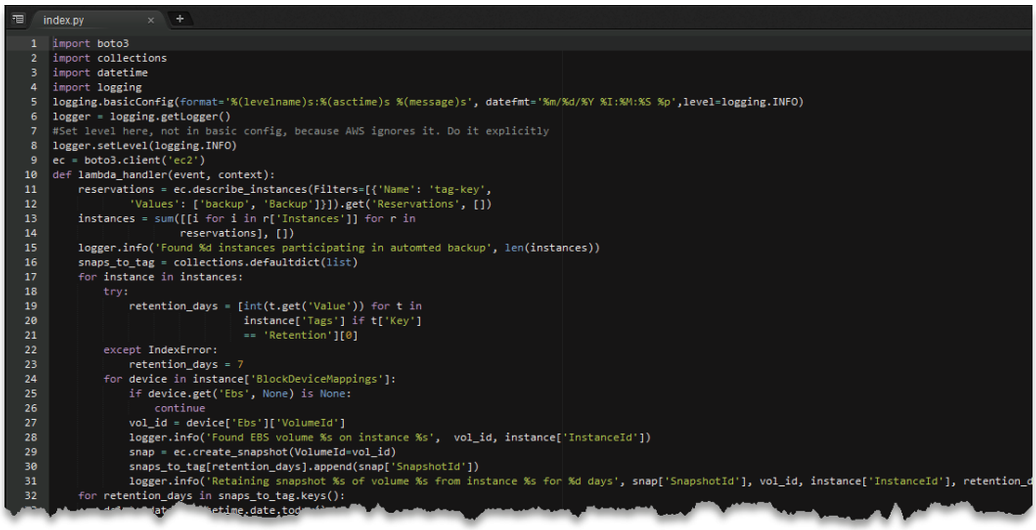


30. While still in the EC2 console, click on **Snapshots** in the navigation pane and notice there are currently no snapshots for our lab instances.

31. Navigate to the **Lambda** console under the Compute section.

32. Click the function with the name of **EBS-Snapshot-Create-Function** to open the properties of this function.

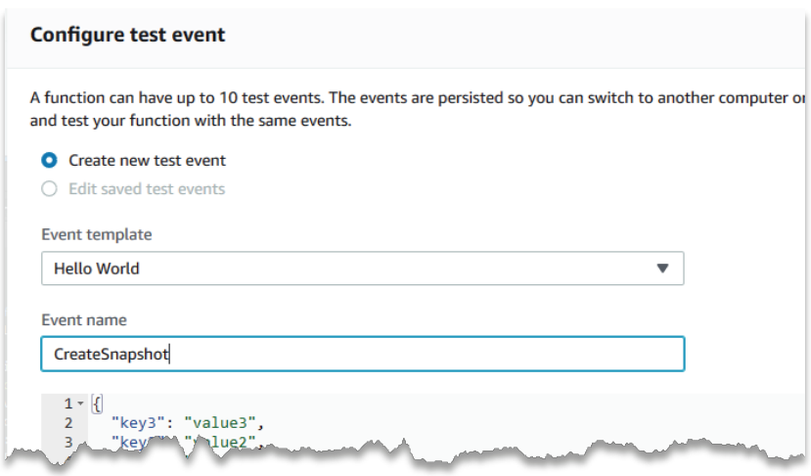
33. Scroll down to **view the code** of the Lambda function - don't worry, you don't need to be able to read it. This Lambda function is written in Python 2.7.



34. At the top of the page, click the **Test** button.

Note: *The test button will allow us to manually execute the Lambda function without requiring additional AWS components, such as Config or CloudWatch.*

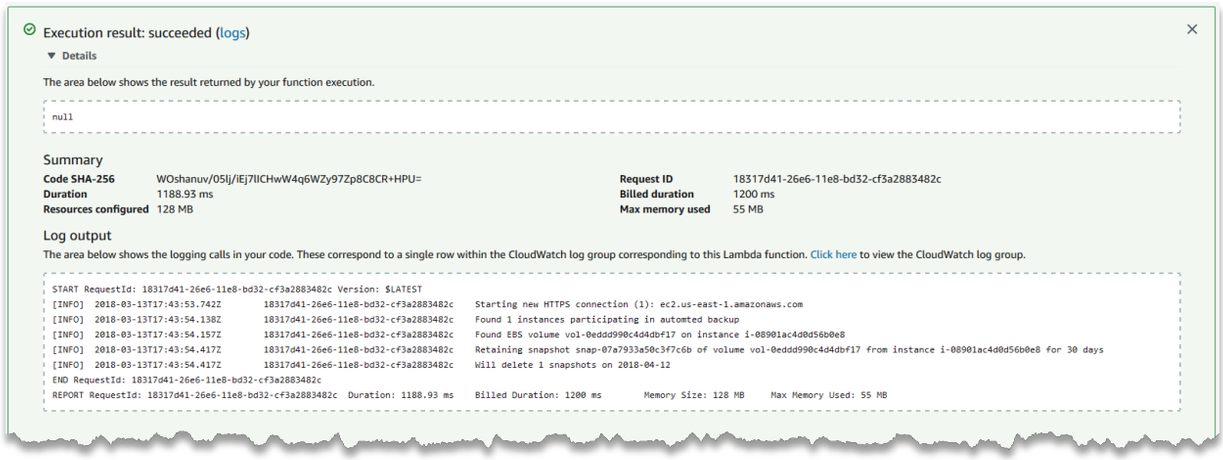
35. Enter **CreateSnapshot** as the Event name and leave all the defaults. **Click Create**.



Note: *For this particular Lambda function, you don't need to pass it any information to work correctly (because it's going out to look for Tags) therefore the JSON code is irrelevant for this test event.*

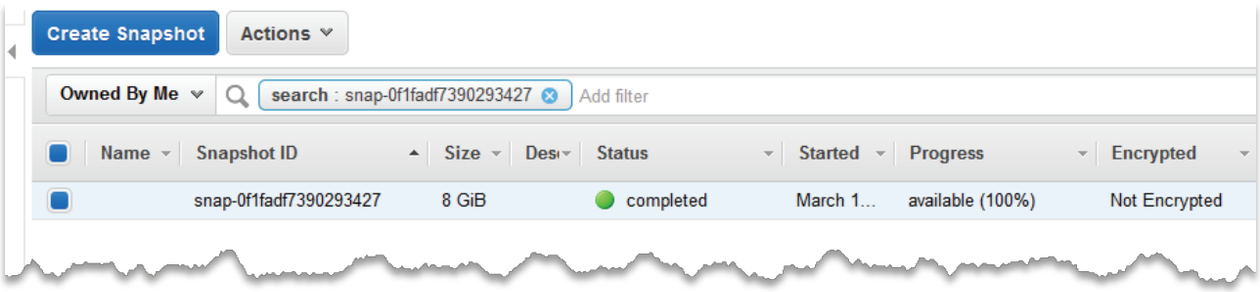
36. You're ready to execute this Lambda function. Ensure the **CreateSnapshot** test event is selected on the drop-down and **click the Test button**.

37. Did it run successfully? Take a look at the logs – **expand the Details** for the Execution result.

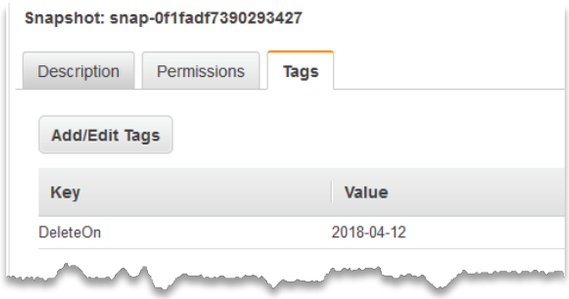


Notice that the Lambda found one (1) instance participating in the automated backup. Secondly, it found the EBS volume attached to the instance and created a snapshot of the volume. Finally, the log output states that the snapshot will be deleted thirty (30) days from now (the value for the Retention tag).

38. Browse to the EC2 console and click on **Snapshots**. You should now have a snapshot listed that was taken of the Windows instance. If you already have snapshots in your account, sort the list by the **Started** column to quickly find the latest snapshot taken.



39. Click on the **Tags** tab in the bottom pane. Notice the **DeleteOn** key with a value that equals 30 days from today's date. That tag is what the EBS-Snapshot-Delete-Function will be looking for when it is executed.



40. Navigate back to the **Lambda** console. **Create a test event** for the **EBS-Snapshot-Delete-Function** Lambda function just like you did for the EBS-Snapshot-Delete-Function in **Step 35.**

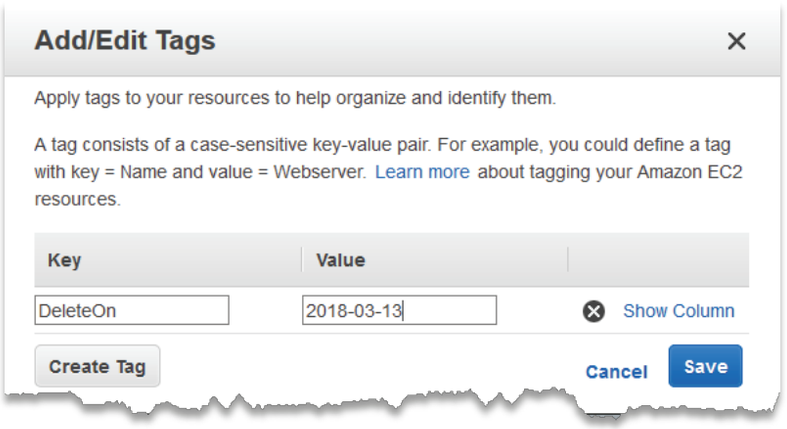
41. Execute it by clicking the **Test** button.

42. Navigate back to the **EC2** console and check to see if it deleted the snapshot.

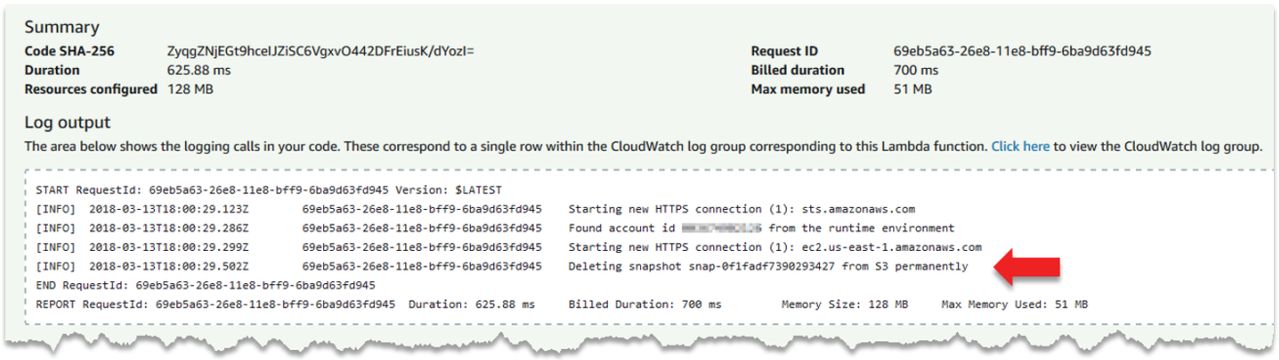
43. **It didn't delete it**. While the function executed correctly, it didn't find any snapshots to delete. If you recall, the 'EBS-Snapshot-Delete-Function' function looks for the value of the date in the **DeleteOn** tag, and if it matches today's date, it will delete it.

44. While still in the EC2 console, click the snapshot, and select **Add/Edit Tags** from the Action menu.

45. **Modify the Value** of the tag to match today's date. Click **Save**.

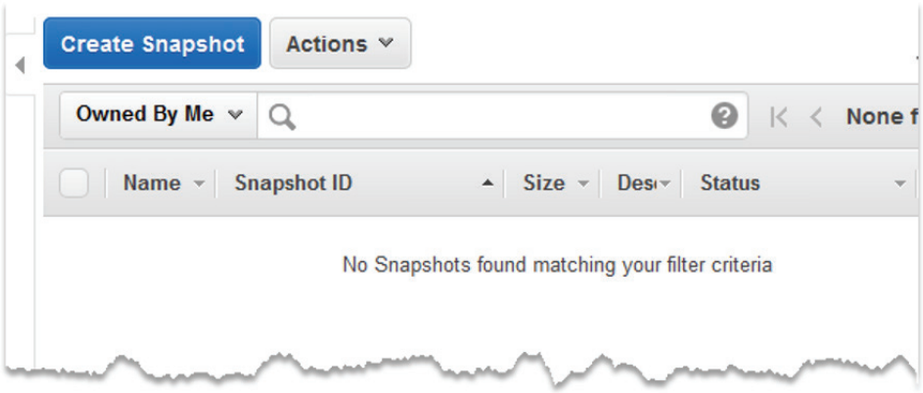


46. Navigate back to the Lambda console, open the properties of the **EBS-Snapshot-Delete-Function** function and **execute** it again. When successful, **expand the Details** and look at the logs.



This time it found a snapshot to delete, as the value of the tag matched today's date.

47. Navigate back to the **EC2** console and verify that the snapshot has been deleted.



That's it. You've completed this lab.

Conclusion: In this lab, participants deployed a CloudFormation change set to modify the existing AWS infrastructure using code. Then you deployed two Lambda functions to help form a data protection strategy for the EC2 instances.

Note: You can delete the stack if you wish, although nothing in the lab will incur costs if you've deleted the snapshot.

Lab 3 – Image Operations

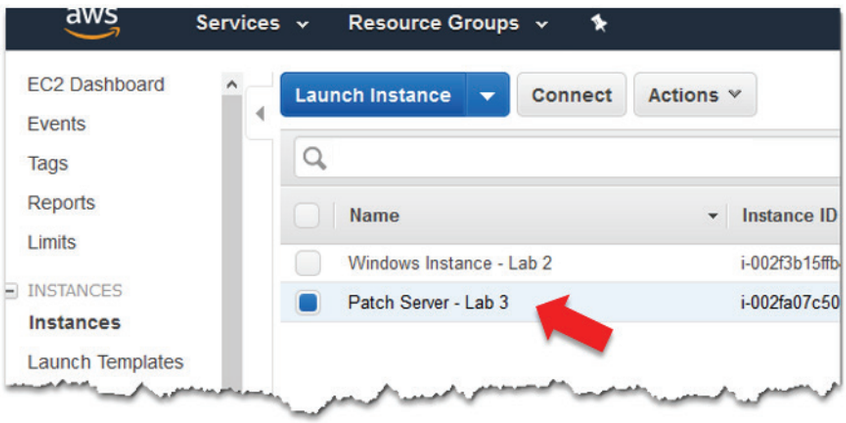
AWS Systems Manager – After resources are provisioned, keeping them secure and compliant can be a seemingly large task for administrators and often takes many hours to scan, stage, and remediate patches for persistent instances in an environment. In addition, third-party tools are often expensive and complicated to manage to help ensure workloads are being patched as required by an established security policy. As a free tool provided by AWS, Systems Manager provides a centralized interface to accomplish a variety of tasks, including automation of tasks, viewing infrastructure performance and configuration and even application management. AWS Systems Manager can also be used to patch instances throughout your AWS infrastructure with a single interface.

Goal – Using an EC2 instance deployed in the initial lab, use AWS Systems Manager to create a maintenance window, assign the instance a baseline for patches, and scan the instance for compliance against the patch baseline.

Prerequisites - Access to the AWS VPC console with an IAM user with appropriate permissions for EC2 and AWS Systems Manager.

---

1. Navigate the **EC2** console and select Instances from the left menu pane. Select the **Patch Server – Lab 3** instance:

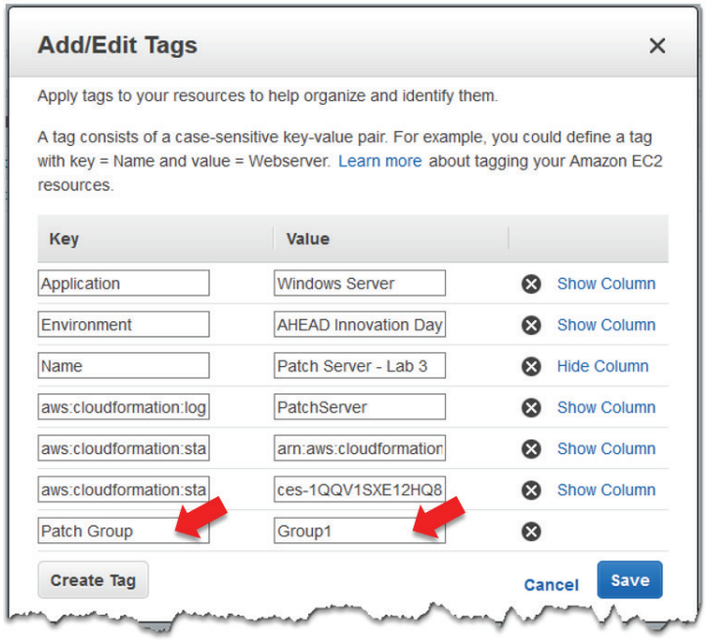


2. From the **Actions** menu, choose **Instance Settings → Add/Edit Tags**.

   Note: *You will see several tags attached to the instance already, some that were created in the CloudFormation template and some that CloudFormation will automatically append to any resource it creates.*

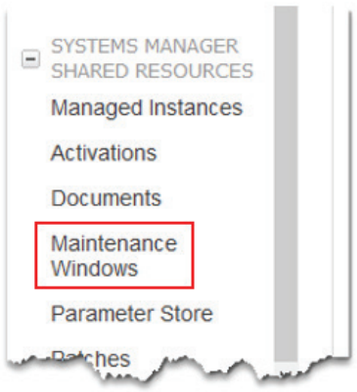
3. Click the **Create Tag** button and add the following tag:

   - **Key:** Patch Group (must include a space)
     Value: Group1

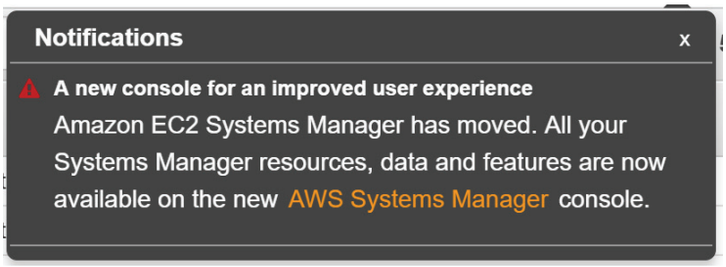


4. Click the **Save** button to apply the new tag.

5. Navigate to the EC2 console and select **Maintenance Windows** under the Systems Manager Shared Resources section (located at the bottom left of the navigation pane).



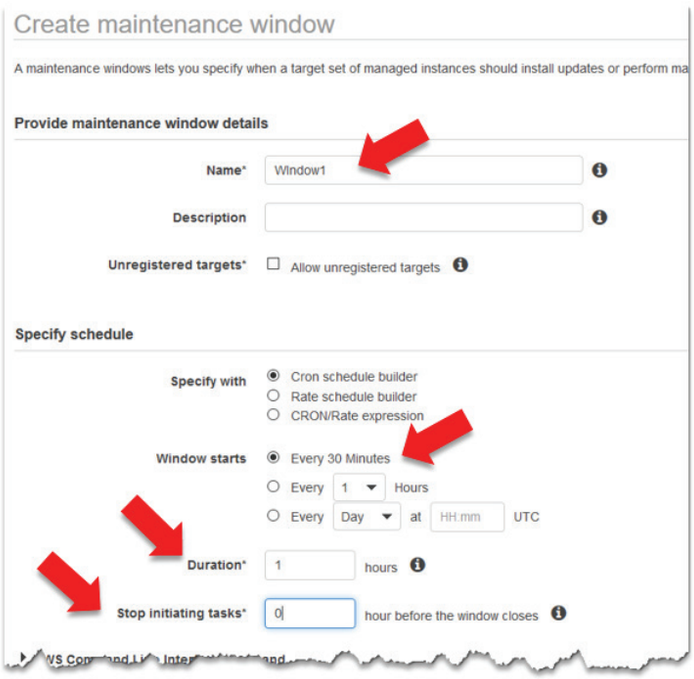
***Note**: If you receive a message regarding the new console for AWS Systems Manager, you can ignore it for now. Just click the X in the top right corner of the message.*



6. Choose **Create a Maintenance Window**.

7. In the Name field, type **Window1**.

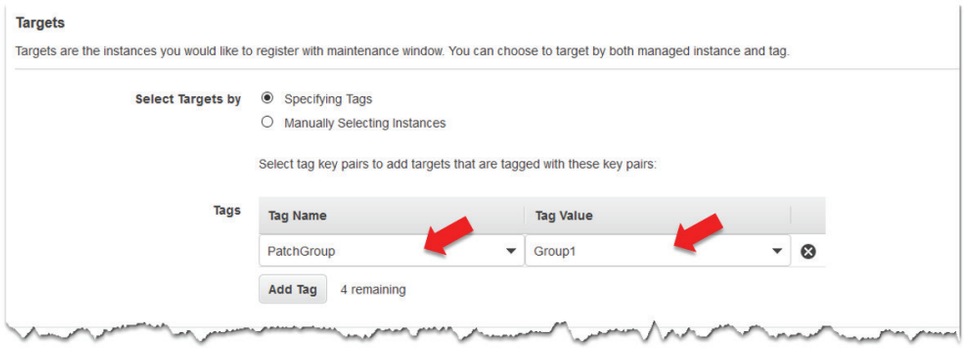
8. In the schedule section, make sure that the Window starts **every 30 minutes**. Enter **1** in the **Duration** field and 0 into the **Stop initiating tasks** field.



9. Click the **Create Maintenance Window** button to finish.

10. Select the Maintenance Window that was just created (**Window1**). From the Actions menu, select **Register Targets**.

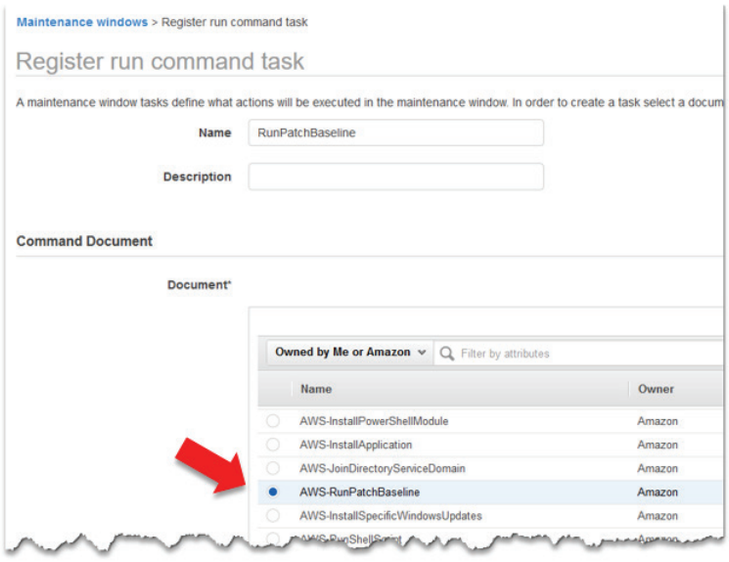
11. Under the Targets section, use the drop-down and select the tag **Patch Group** that was created earlier. For the Tag value, select **Group1**.



12. Click **Register Targets**. Click **Close** on the acknowledgment.

13. Select the Maintenance Window and choose **Register run command task** from the Actions menu.

14. Type **RunPatchBaseline** under the Name field. Select **AWS-RunPatchBaseline** from the list of command document section.



15. Under **Targets**, ensure that a target is selected.

    Note: *If you have multiple Maintenance Windows, you might need to identify the correct one by navigating back to the Maintenance Window console and making note of the Window Target ID.*

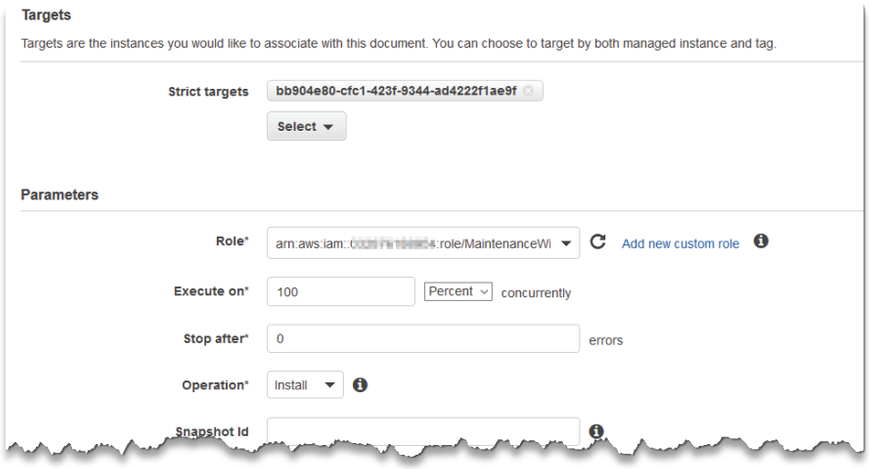
16. Under **Role***, select the **MaintenanceWindowRole** that was created as part of the initial CloudFormation template.

17. In the **Execute on*** field, enter **100** and ensure **Percent** is selected from the drop-down.

18. In the **Stop after*** field, type **0**.

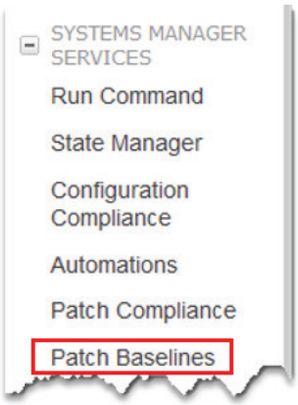
19. Modify the Operation field and change to **Install**.

    Your selections should like the screenshot below:



20. Click **Register Task**. Click **Close** on the acknowledgment.

21. In the EC2 console, select **Patch Baselines** under the Systems Manager Services group:



22. Select the **AWS-DefaultPatchBaseline** from the list of default baselines. It should have **Windows** listed as the Operating System.

23. From the Actions menu, select **Modify Patch Groups**.

24. In the Patch Groups field, type **Group1**, be sure to **click the check mark** to the right of the text box.

25. Click **Close**.

26. In the EC2 console, click on **Run Command** on the navigation pane under the System Manager Services section.

    Note: *For lab purposes, you will be manually executing a Scan operation. In a production environment, this would be configured to run on a recurring schedule.*

27. Select the **Run a command** button.

28. In the command document section, select the radio button next to **AWS-RunPatchBaseline**.

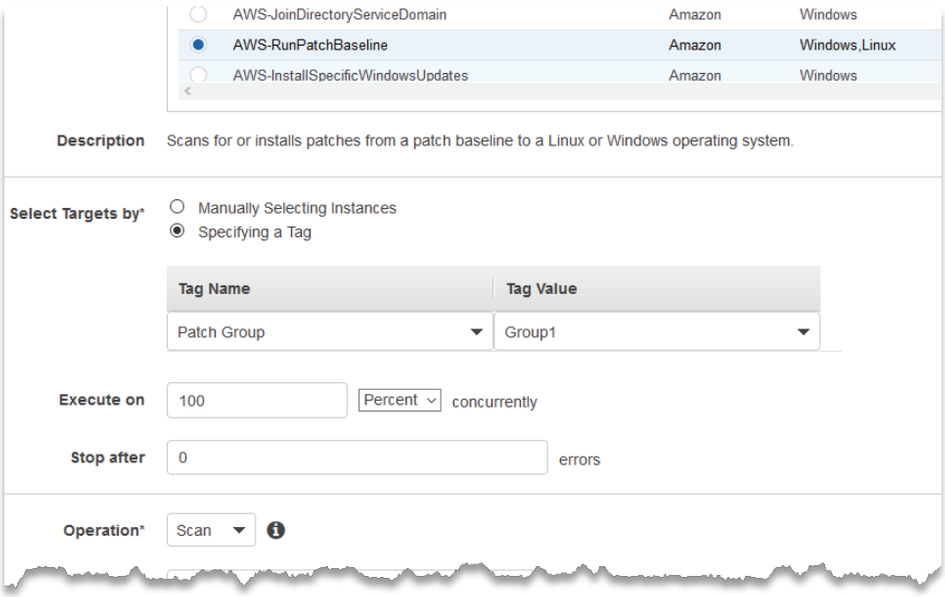
29. Under **Select Targets by***, click the radio button next to **Specifying a Tag**.

30. Under Tag Name, select **Patch Group**. Select the value of **Group1** under the Tag Value.

31. In the Execute On text box, enter **100** and modify the concurrency box to **Percent**.

32. Enter **0** in the Stop after text box.

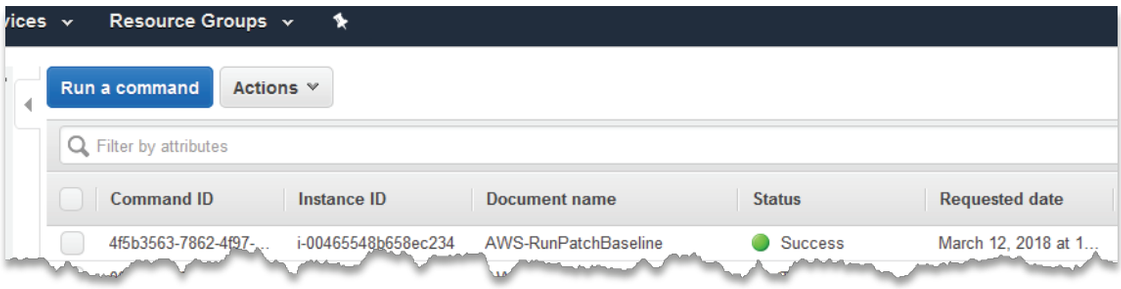
33. In the Operation* box, select “Scan.”



34. Click **Run**.

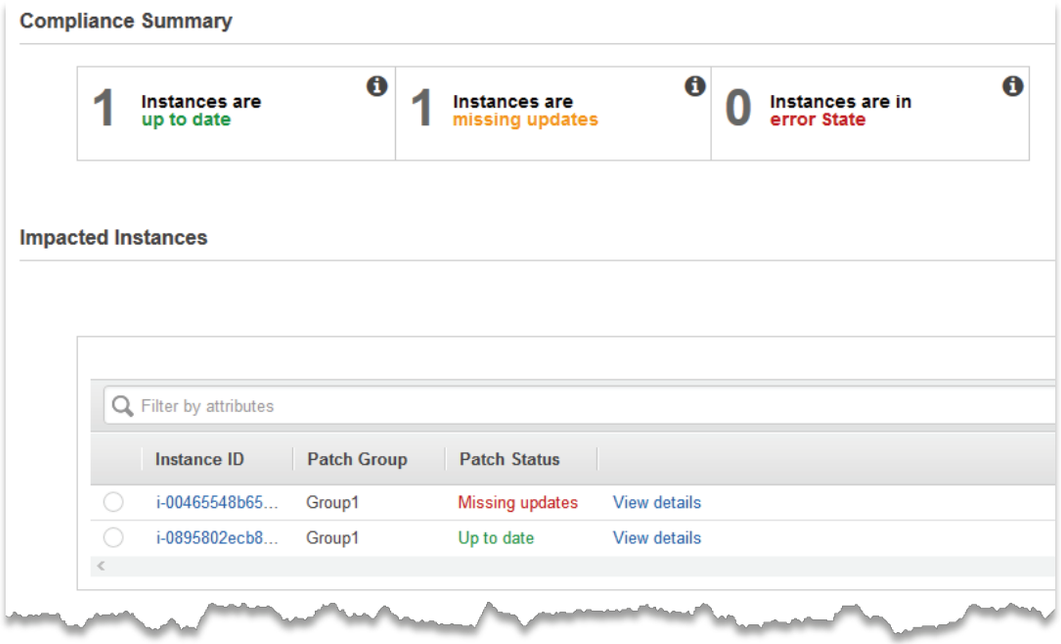
35. You should receive a success message. Click **View Result**.

36. Wait until the Run command status changes to **Success** (this takes about 10 minutes on average).



37. Navigate to the **Patch Compliance** console found in the navigation pane in the EC2 console under the Systems Manager Services section.

38. Is it reporting that your instances up to date?



***Note**: The screenshot above is an example. It's very likely that your instances are indeed up to date, as Amazon keeps its AMIs up to date with the latest Microsoft patches. However, workloads that have persisted longer than a month or so would likely require security patches.*

39. Select the instance and click **View Details**.

40. Are there patches listed for the instance?

That's it. You've completed this lab.

Conclusion: In this lab, participants used AWS Systems Manager to enable patching compliance for instances running in EC2. You created a Maintenance Window to install patches and scanned the EC2 instances to ensure compliance against a patch baseline.

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.

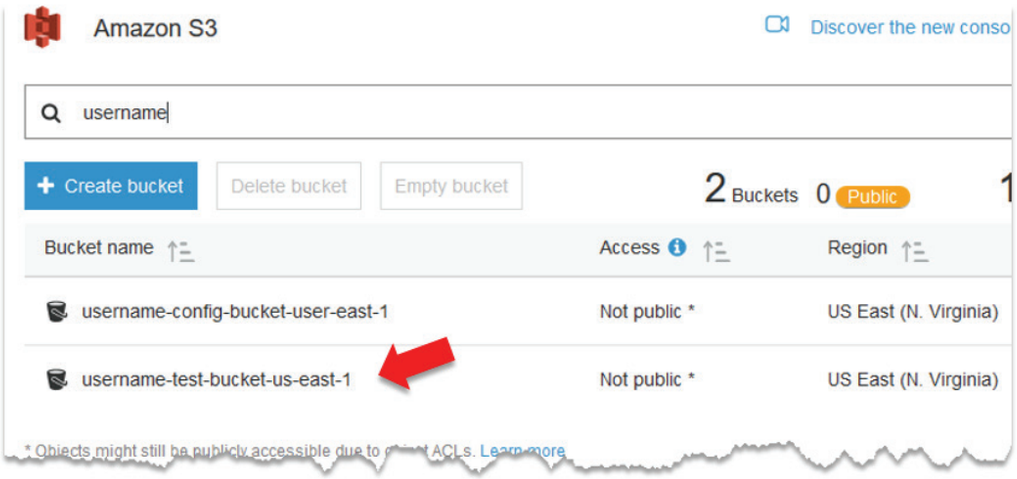
Prerequisites - Access to the AWS VPC console with an IAM user with appropriate permissions to add S3 bucket tags and configure Amazon SNS.

---

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**.

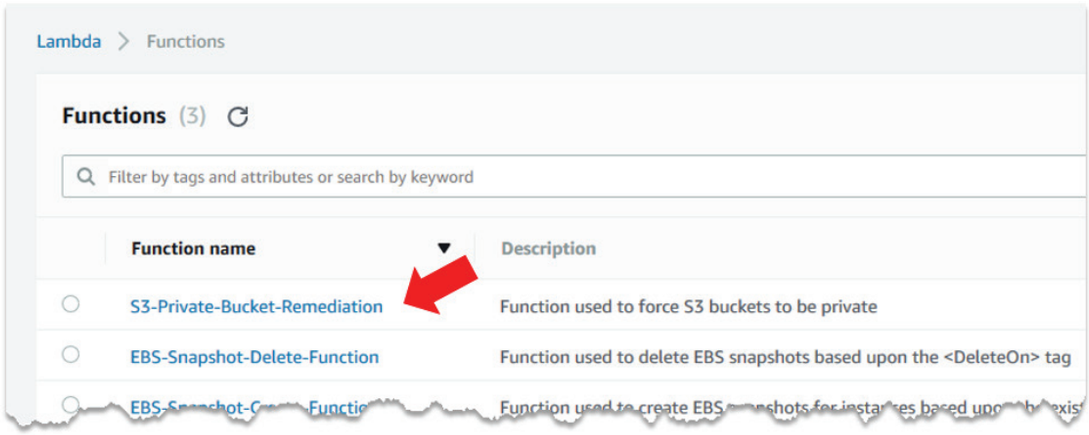
   Notice in the console that the access shows as **Not public***.



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.

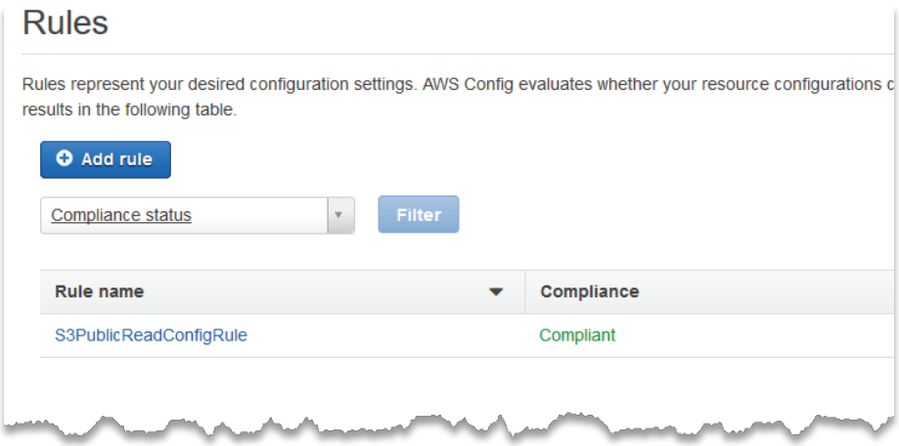


6. From the Services menu, navigate to the **Config** console. Config is located under the Management Tools section.

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.*



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**.*

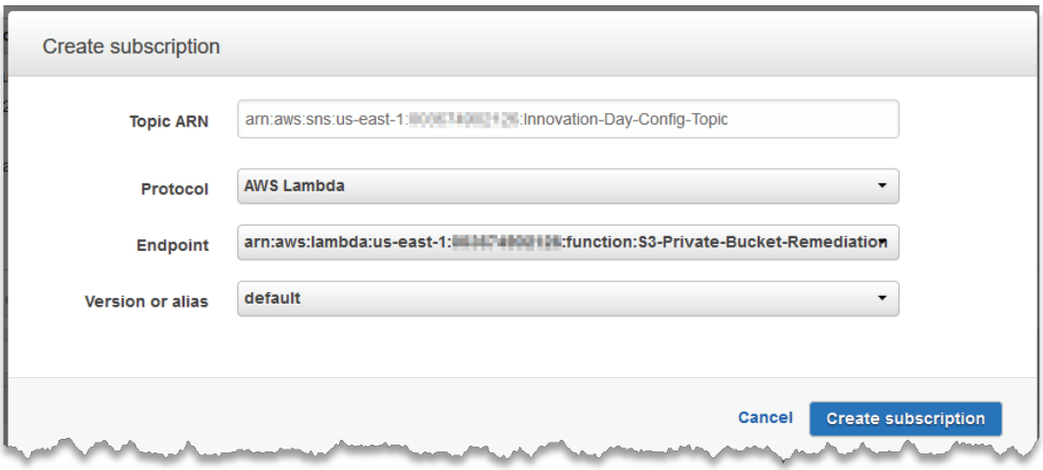
11. Navigate to the **Simple Notification Service (SNS)** console. It can be found under the Application Integration section.

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.



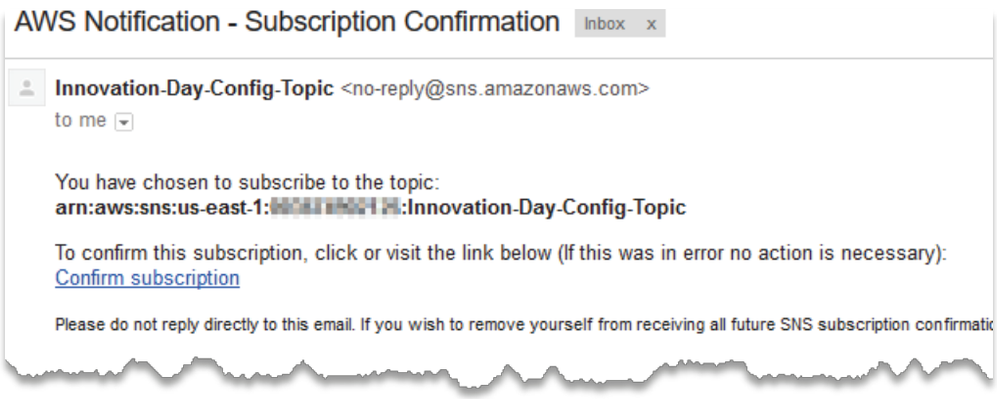
16. Click the **Create subscription** button.

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.



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.

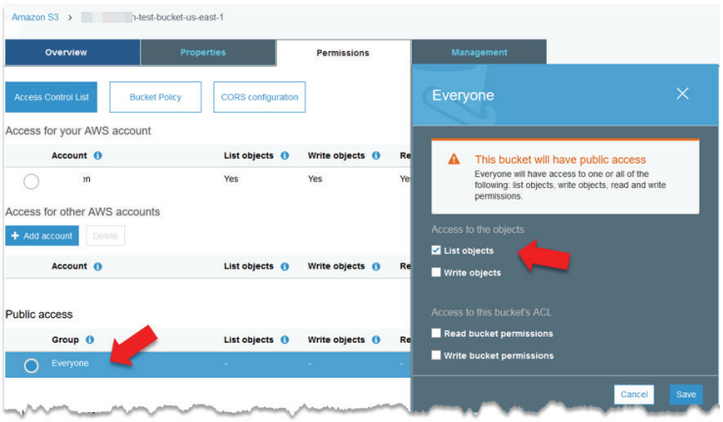
    Note*: This tag's key-value pair is a simple example for the lab. In a production environment, you may have different tags and a predefined list of values to identify data classification for the data within an S3 bucket.*

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.*



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.

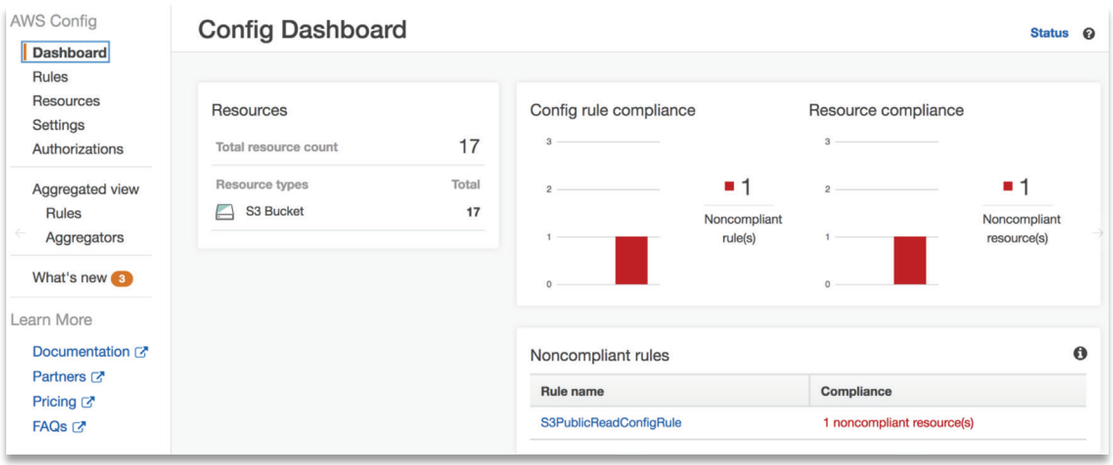
30. Select the **S3PublicReadConfigRule**.

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.*

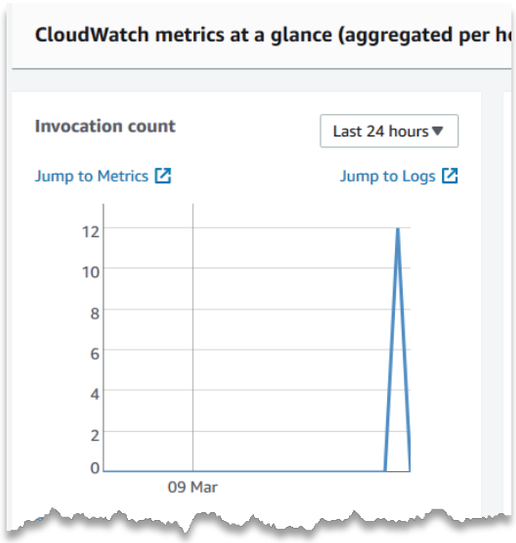


33. Navigate back to the **Lambda** console.

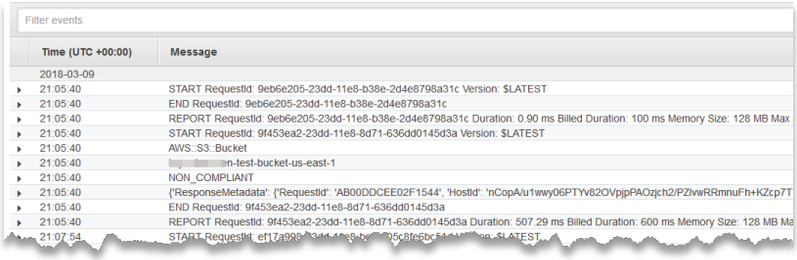
34. Under the Functions menu, locate the **S3 function** you evaluated earlier and **click the name** to open the function properties.

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.



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

39. Does the S3 bucket still have public access?

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.

Lab 5 – AutoScaling

EC2 Autoscaling – In EC2, Auto Scaling is the ability to scale resources based upon the demand for your application. As the workload increases, Auto Scaling can provision additional resources to absorb requests or other workloads to ensure the performance and availability of your application meets pre-defined minimum standards. As that workload decreases, Auto Scaling can decrease capacity to reduce costs while maintaining a minimal viable configuration. Auto Scaling is also used to help ensure the health and availability of your fleet and ensuring that applications are still accessible in the event of an AWS outage.

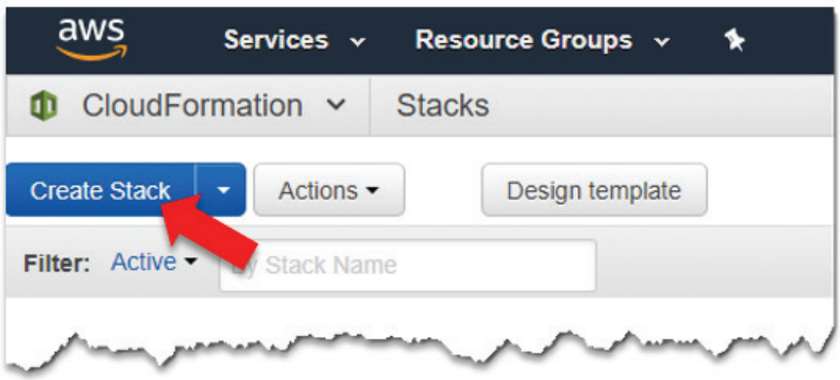
Goal - In this session, users will learn how to deploy an application in a highly scalable fashion to support the scaling of resources on-demand. Participants will configure autoscaling for a web application and experience how changes in the environment can trigger the addition, or removal, of resources as required by the application.

Prerequisites – A VPC with public subnets that allow internet access for resources with public IP addresses. If you've completed Lab 1, you're all set to go.

---

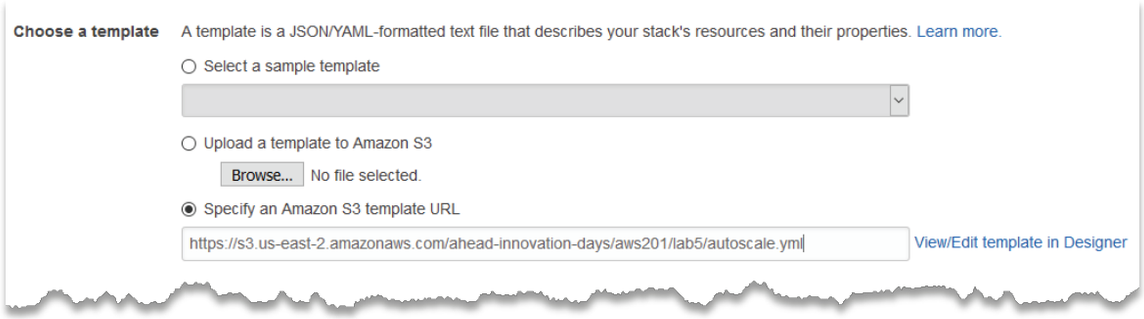
1. Navigate to the **CloudFormation** console.

2. Create a new stack by clicking the **Create Stack** button at the top.



3. Click the radio button next to **Specify an Amazon S3 template URL**. Enter the following URL in the text box.

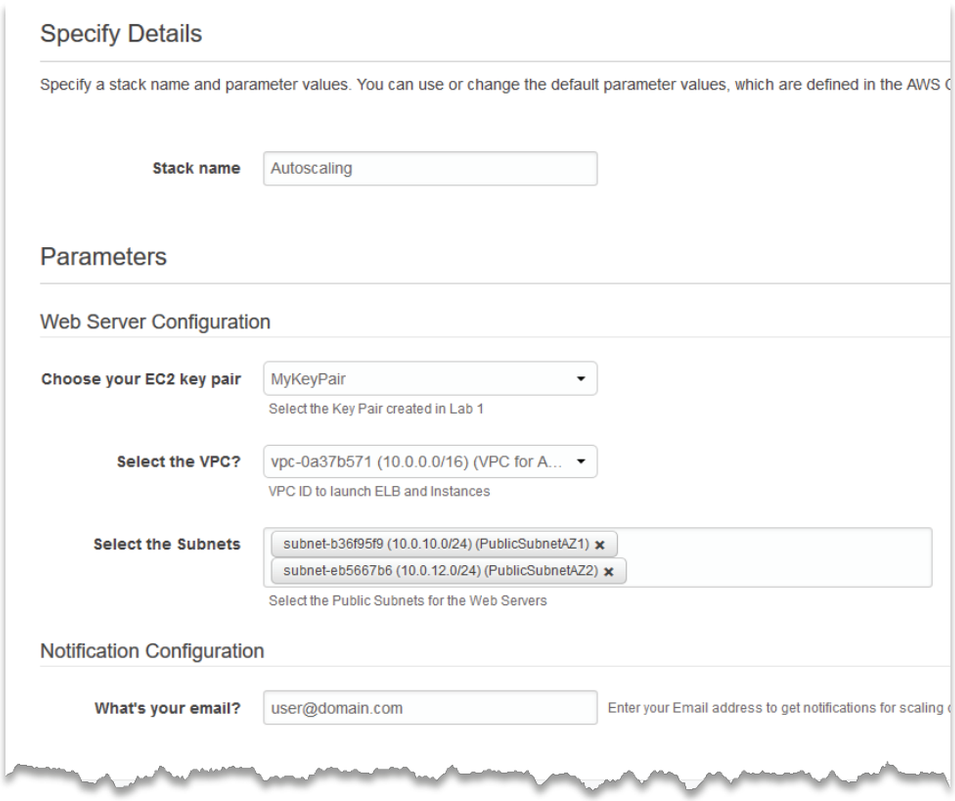
   https://s3.us-east-2.amazonaws.com/ahead-innovation-days/aws201/lab5/autoscale.yml



4. In the Stack name text box, type **Autoscaling**.

5. In the Parameters section, provide the values as listed below:

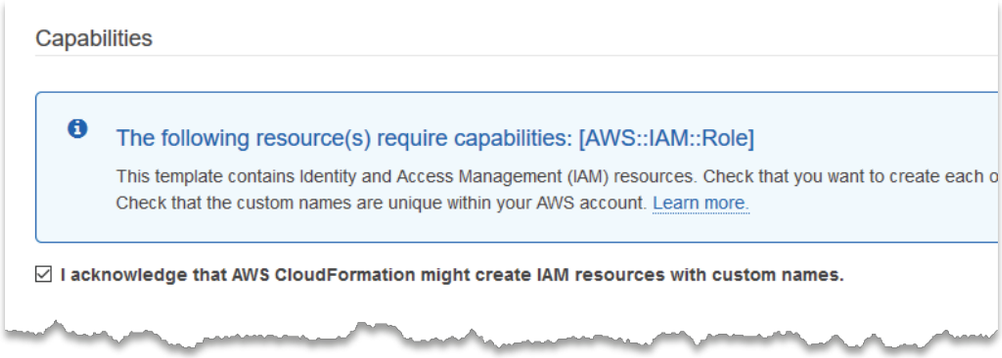
- **Web Server Configuration**
  - **EC2 Key Pair** – Select the Key Pair created at the beginning of Lab 1
  - **VPC**: Select 'VPC for AHEAD Innovation Days' from the list
  - **Instance Subnets**: Select 'PublicSubnetAZ1' and 'PublicSubnetAZ2' from the list
- **Notification Configuration**
  - **Email Address** – Enter your email address to receive alerts for autoscaling operations.



6. Click **Next**.

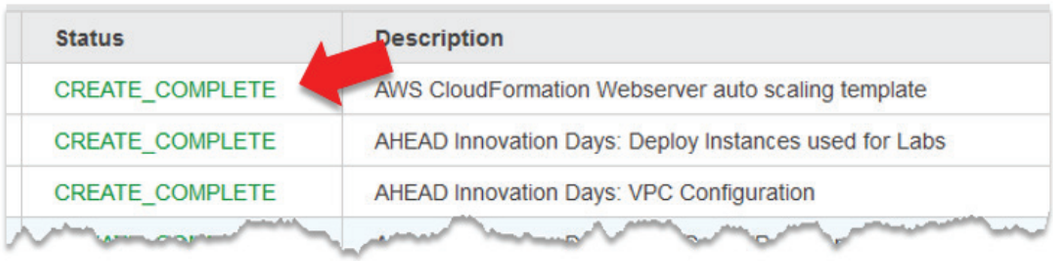
7. On the Options page, leave the defaults and select **Next**.

8. On the Review page, **click the checkbox** to acknowledge that the template will create an IAM role.

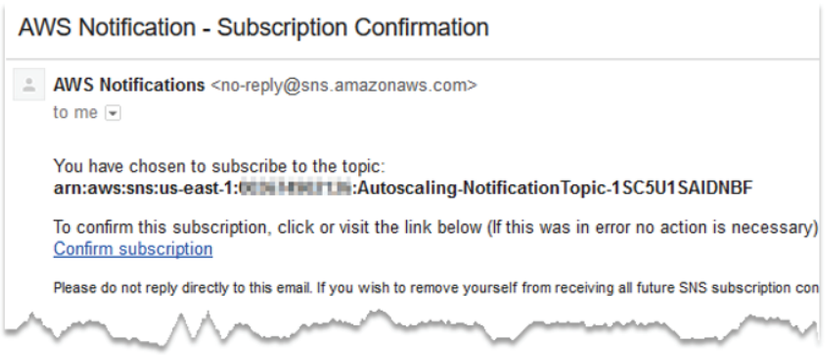


9. Click **Create** to launch the stack.

10. In the CloudFormation console, **refresh** the window until the status for the Autoscaling stack changes to CREATE_COMPLETE.

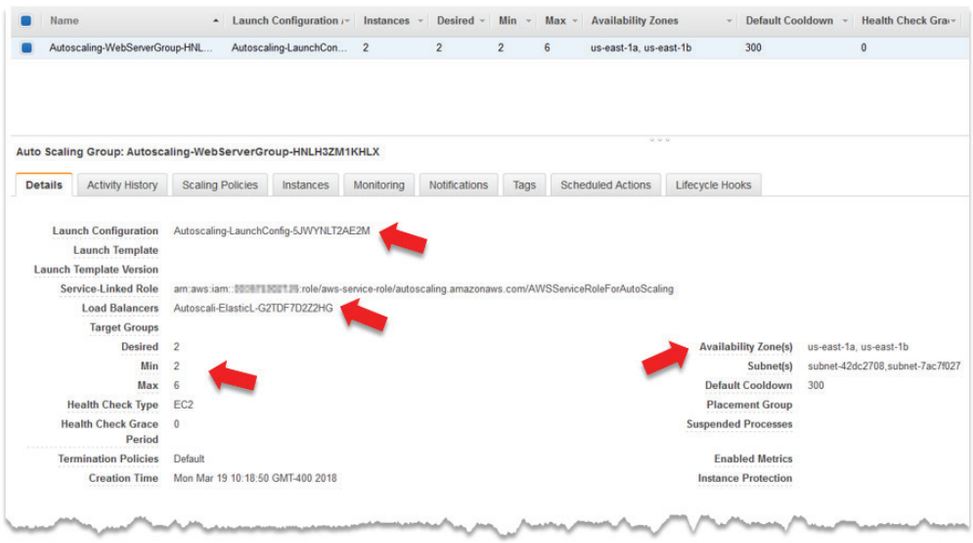


11. Once the template successfully deploys, make sure to **confirm the SNS topic subscription** created by the template using the email you received during the deployment.



12. Before you make any changes, let's take a glance at what the CloudFormation template created. Browse to the **EC2** console and click on **Auto Scaling Groups** in the AUTO SCALING section. Take a look at the configuration for this Auto Scaling Group in the lower pane, starting with the Details tab:

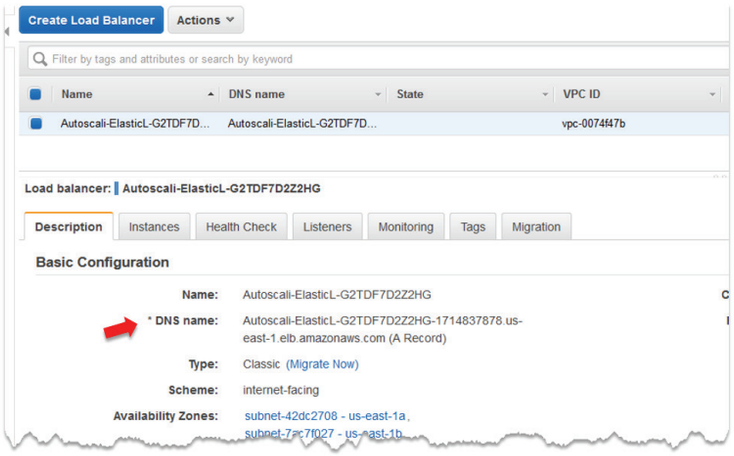
    - **Details Tab:** Notice that this Auto Scaling Group is using the Launch Configuration that was created from the CloudFormation template. It also uses an Elastic Load Balancer for the instances to provide load balancing and redundancy for incoming traffic. The Auto Scaling Group will provision instances across two Availability Zones and will start with a minimum of two (2) instances, scaling up to a maximum of six (6).
    - **Activity History Tab:** Notice that the ASG has already created the two (2) instances as the minimum deployment.
    - **Scaling Policies Tab:** There are two (2) simple auto scaling policies for this lab, one that scales up if any of the existing instance's CPU Utilization breaches 40% for over 60 seconds. For this example, assume that this means a spike in traffic and more resources are needed to properly serve the application. The second policy is for scaling down – if the instance's CPU Utilization is below 20% for 120 seconds, assume demand has decreased, and Auto Scale can reduce the number of instances needed to serve the application.
    - **Instances Tab:** Information on the two initial instances currently running. Notice that they are in separate Availability Zones to provide redundancy.



Now that you understand what was deployed, let's put it to work to see how Auto Scaling can dynamically respond to events in the environment.

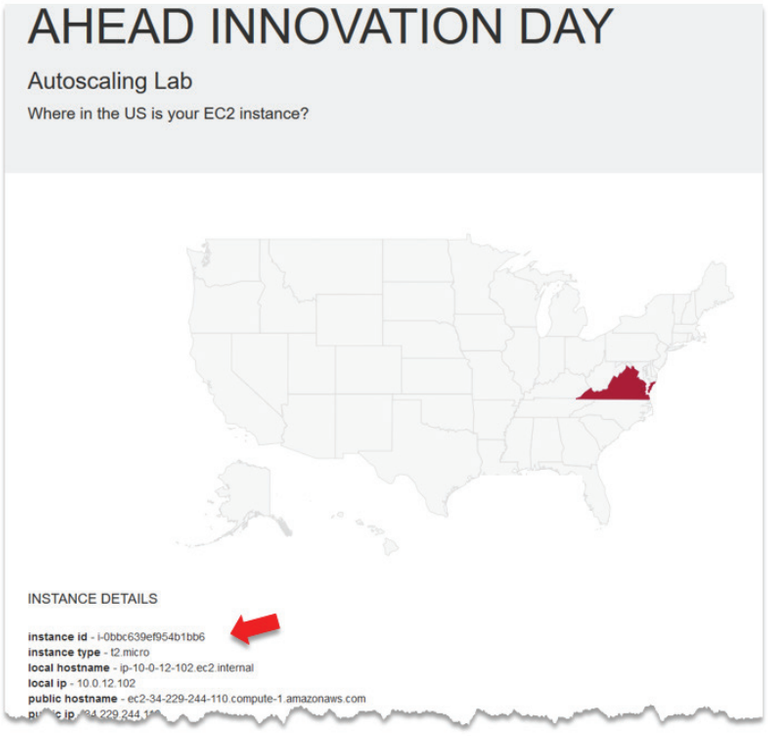
13. In the EC2 console, select **Load Balancers** on the left navigation pane.

14. **Select the box** next to the load balancer, if not already selected, and view the **Description** tab in the lower pane. Copy the **URL** listed for the **DNS Name**.



15. **Paste the URL into your web browser** and view the website that displays. The website is the test application that you want to ensure uptime and availability for using the Auto Scaling Group.

    ***Note**: As stated, there are two (2) instances serving this application. Feel free to refresh your browser and see how the Instance Details at the bottom change as the request gets directed back and forth between the supporting instances.*



Now that you've seen that the deployed infrastructure works as deployed, you're going to see EC2 Auto Scale in action. To do this, you're going to use a stress tool that was installed on the instances during

the provisioning process. You're going to use the tool to create artificial workloads for the CPU and see how Auto Scale responds.

16. In the EC2 console, click **Instances** on the left. Again, notice there are only two (2) instances running the application right now – both named **WebServer**.

17. In the navigation pane, click **Run Command** under the SYSTEMS MANAGER SERVICES section.

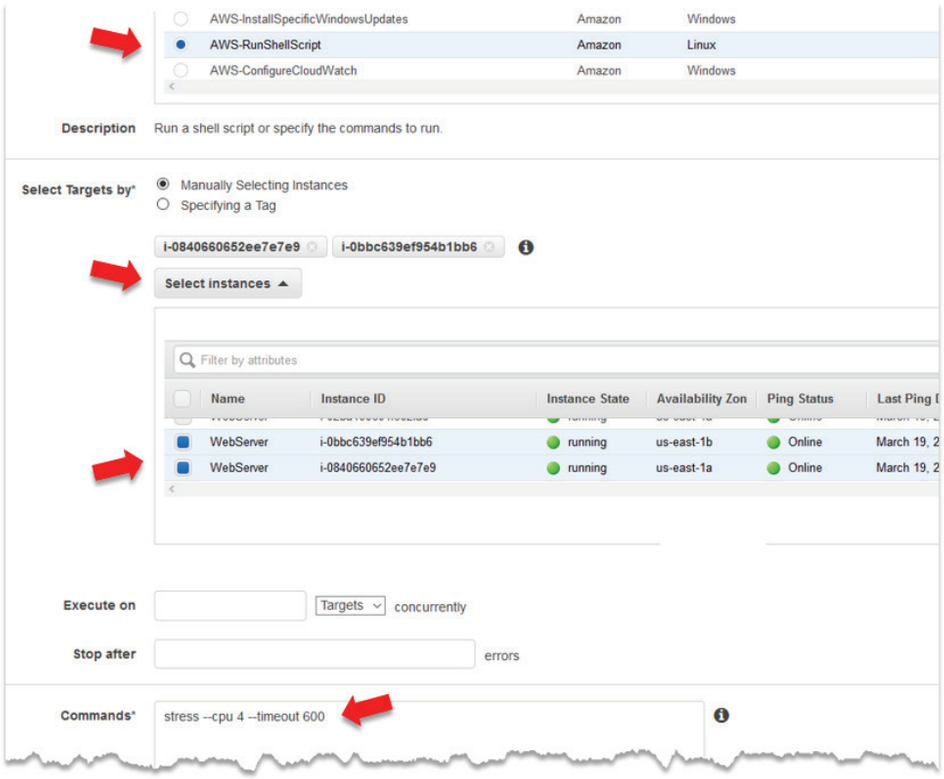
18. Click the **Run a command** button.

19. In the command document section, select the radio button next to **AWS-RunShellScript** from the list.

20. For the **Select Targets by*** section, select **Manually Selecting Instances** and click the **Select Instances** button. Select the two instances named **WebServer**.

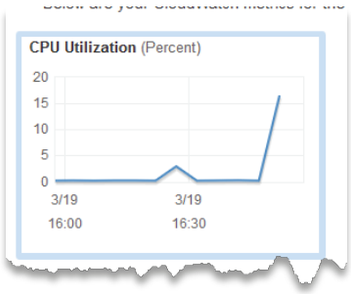
21. In the **Commands*** section, type the following command:

    stress --cpu 4 --timeout 600



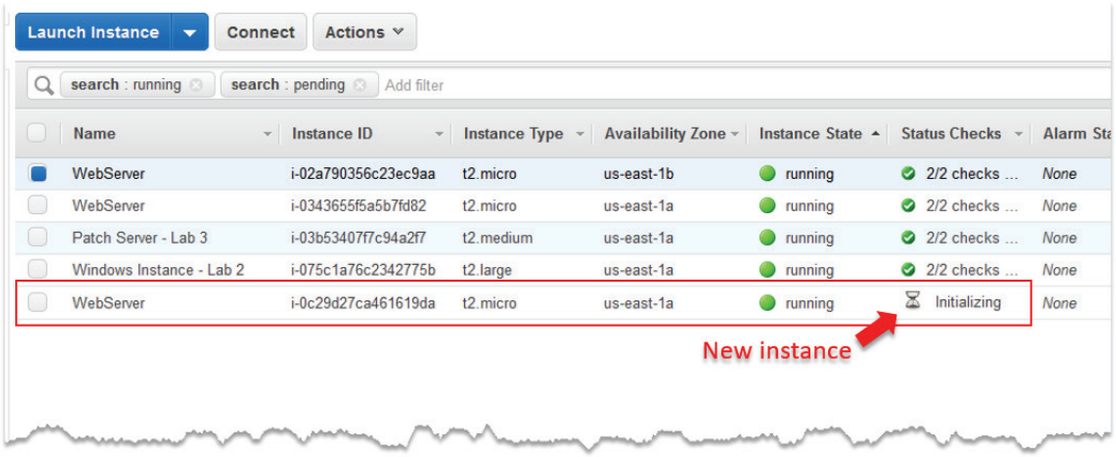
22. Click **Run**.

23. In the EC2 console, select **Instances** and select one of the **WebServer** instances. Click the **Monitoring** tab at the bottom and **refresh** the graph until you see the CPU Utilization spike.



24. At the top, **refresh** the list of instances running in your account. Continue to refresh until you see additional EC2 instances being provisioned. As configured, Auto Scaling is reacting to the CPU spike by provisioning additional resources to absorb the workload that you artificially created on the hosts.

    Note**: Auto Scale will eventually scale up to the maximum number of instances for the Auto Scaling Group, which was defined as **six (6).

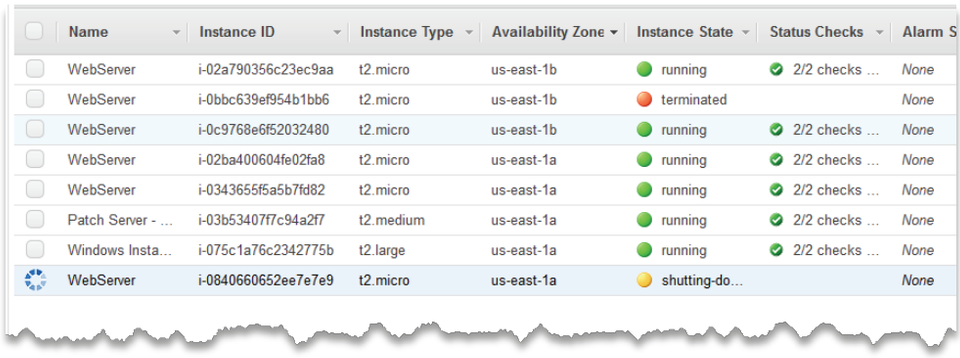


25. On the left, click **Load Balancers**. Click on the load balancer and select the **Instances** tab. Notice that the new instances are automatically being placed behind the load balancer so they can receive web requests for the application.

26. In your web browser, click **refresh** on the application webpage to see that requests are now being handled by six (6) different hosts, rather than just the original two instances that were initially provisioned.

    Now that you've seen Auto Scale react to an increase in workloads you're going to see what happens when the workload is reduced and returns to normal conditions. The command that was run against the instances earlier had a timeout, meaning that it only stressed the CPU for 5 minutes and stopped afterward. The timeout will cause the CPU to drop back down to almost 0% CPU Utilization in which the scale down policy will be executed.

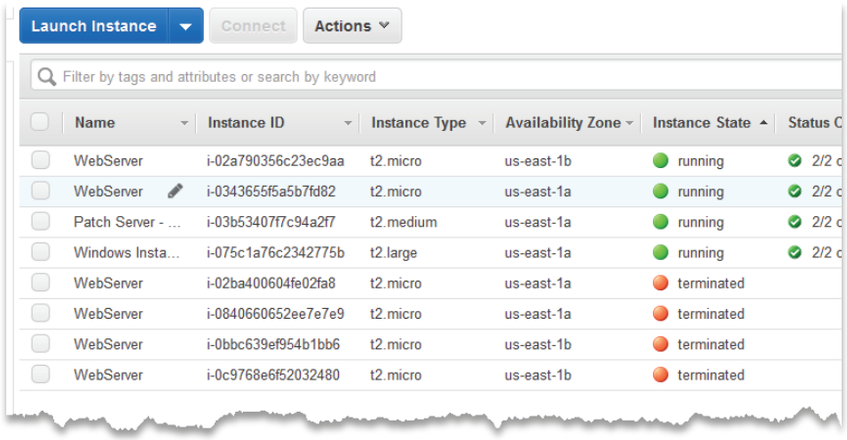
27. In the EC2 console, click **Instances**. Continue to **refresh** the list of instances. After a few minutes, you'll see that Auto Scale starts to **Shutdown and Terminate** one instance at a time.



Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm S
WebServer	i-02a790356c23ec9aa	t2.micro	us-east-1b	running	2/2 checks ...	None
WebServer	i-0bbc639ef954b1bb6	t2.micro	us-east-1b	terminated		None
WebServer	i-0c9768e6f52032480	t2.micro	us-east-1b	running	2/2 checks ...	None
WebServer	i-02ba400604fe02fa8	t2.micro	us-east-1a	running	2/2 checks ...	None
WebServer	i-0343655f5a5b7fd82	t2.micro	us-east-1a	running	2/2 checks ...	None
Patch Server - ...	i-03b53407f7c94a2f7	t2.medium	us-east-1a	running	2/2 checks ...	None
Windows Insta...	i-075c1a76c2342775b	t2.large	us-east-1a	running	2/2 checks ...	None
WebServer	i-0840660652ee7e7e9	t2.micro	us-east-1a	shutting-do...		None

Eventually, Auto Scale will **terminate** all but **two (2)** instances, which is the minimum for the application. Keep in mind that the remaining two instances serving the application will not likely be the same instances that you originally provisioned. In this case, you shouldn't care, as they are all serving the same application.

Feel free to **refresh** the webpage again to see that it's now being supported by the two remaining EC2 instances that remain from the Auto Scale event.



Name	Instance ID	Instance Type	Availability Zone	Instance State	Status C
WebServer	i-02a790356c23ec9aa	t2.micro	us-east-1b	running	2/2 c
WebServer	i-0343655f5a5b7fd82	t2.micro	us-east-1a	running	2/2 c
Patch Server - ...	i-03b53407f7c94a2f7	t2.medium	us-east-1a	running	2/2 c
Windows Insta...	i-075c1a76c2342775b	t2.large	us-east-1a	running	2/2 c
WebServer	i-02ba400604fe02fa8	t2.micro	us-east-1a	terminated	
WebServer	i-0840660652ee7e7e9	t2.micro	us-east-1a	terminated	
WebServer	i-0bbc639ef954b1bb6	t2.micro	us-east-1b	terminated	
WebServer	i-0c9768e6f52032480	t2.micro	us-east-1b	terminated	

You Did It. You are done with this Lab.

Conclusion: In this lab, participants used EC2 Auto Scale to provision a new application. Using a stress tool pre-installed on the instances, participants created artificial workload to watch how Auto Scale reacted, scaling up resources as the workload increased and scaled down when the workload decreased.

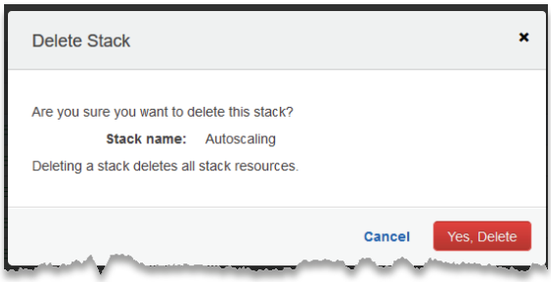
Lab 6 – Account Cleanup

Account Cleanup – At AHEAD, we understand that while learning new concepts and technology is important, leaving behind technical debt and artifacts throughout your account isn't good practice. Therefore, this lab was created to ensure AWS resources from today's Innovation Day are deleted, your account is nice and tidy, and the only thing you need to worry about is convincing your boss how to put what you've learned today to good use in your organization.

Goal – Delete all AWS resources used for today's labs.

Delete the Auto Scale Stack

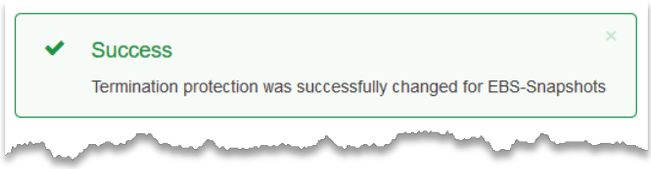
1. Navigate to the CloudFormation console and select the **AutoScaling** stack.
2. From the Actions menu, select **Delete Stack**.
3. Select **Yes, Delete**.



***Note**: Deleting this stack will terminate all EC2 instances that were deployed as part of the Auto Scaling Group. It will delete the Launch Configuration, Auto Scaling Group, the load balancer, the security group, SNS topic, and the IAM role that was created for this lab.*

Delete the EBS-Snapshots Stack

4. Back in the CloudFormation console, select the **EBS-Snapshots** stack.
5. From the actions menu, select **Change termination protection.**
6. Select **Yes, Disable**.

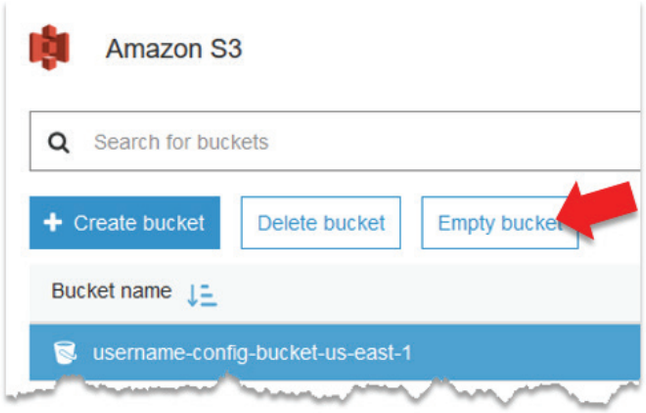


7. From the actions menu, select **Delete Stack**.

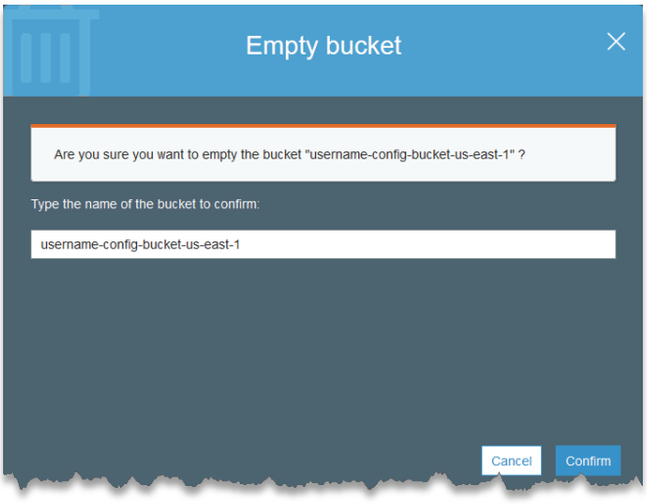
***Note**: Deleting this stack will delete both of the Lambda functions and the role associated with them.*

Deleting the AHEAD-Innovation-Days Stack

8. Navigate to the **S3** console.
9. Select the bucket used for Config logs. The bucket name should be: ***username***-config-bucket-us-east-1.
10. Click the **Empty bucket** button above the list of buckets to delete all contents.



11. **Type the name** of the bucket in the textbox to confirm that you want to empty the bucket.
12. Click **Confirm.**



13. Navigate back to the **CloudFormation** console.
14. Select the **AHEAD-Innovation-Days** stack.

    Note*: Make sure to select the main, top-level stack and not one of the nested stacks.*

15. From the actions menu, select **Delete Stack**.
16. Click **Yes, Delete**.

Cleaning up EC2 Systems Manager

17. Navigate to the EC2 console.

18. Click on Maintenance Windows in the navigation pane. Delete the **Windows1** maintenance window by selecting **Delete maintenance window** from the Actions menu.

19. Click on **Patch Baselines** in the navigation pane. Select the **AWS-DefaultPatchBaseline** and select **Modify Patch Groups** from the Actions menu.

20. Click the **X** next to Group1 to delete the associated Patch group.

21. Click **Close**.

You did it. You are done with this lab.

Conclusion: In this lab, participants cleaned up all AWS resources that were provisioned throughout today's Innovation Day.

Additional Education

Although not a requirement, we've included a few additional 'labs' for you to work on after today's Innovation Day was complete. The labs will build upon what you've learned during today's session and will further display how you can easily integrate multiple AWS services to build and accomplish amazing things. While not as detailed as the labs above, feel free to experiment with the concepts behind these 'take-a-way' labs as you continue to educate yourself and gain experience on the AWS platform. Enjoy!

Lab 1 – Billing Alerts

During the process of experimentation and learning something new, the last thing you want to worry about is what your bill will be at the end of the month. Sure, you can check your account often to ensure your bill doesn't unexpectedly increase, but why not use the AWS platform to protect yourself from an unexpected bill. With the combined use of CloudWatch and SNS, you can configure billing alarms to send out notifications when your bill hits certain thresholds.

Goal – Go beyond the guided labs above and experiment with additional services on the AWS platform.

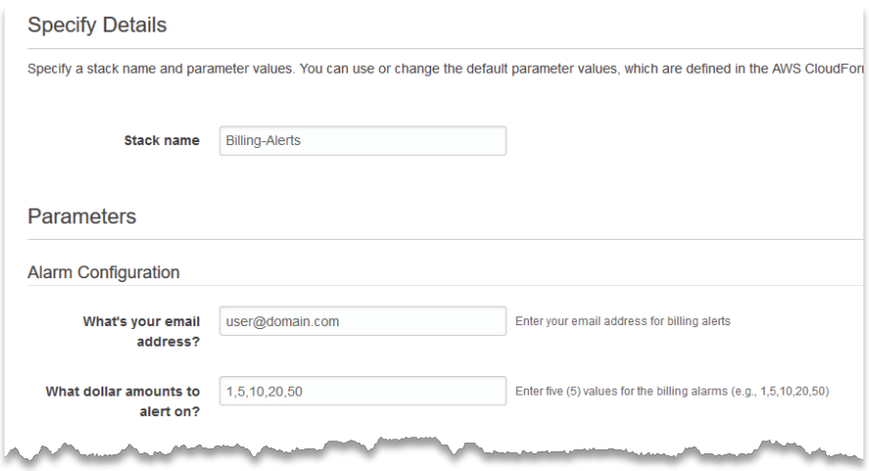
Prerequisites – A desire to learn and expand your skillset. You'll need an AWS account as well!

---

1. Using CloudFormation, **create a new stack** using the template found at the following URL:

   https://s3.us-east-2.amazonaws.com/ahead-innovation-days/aws201/billingalerts.yml

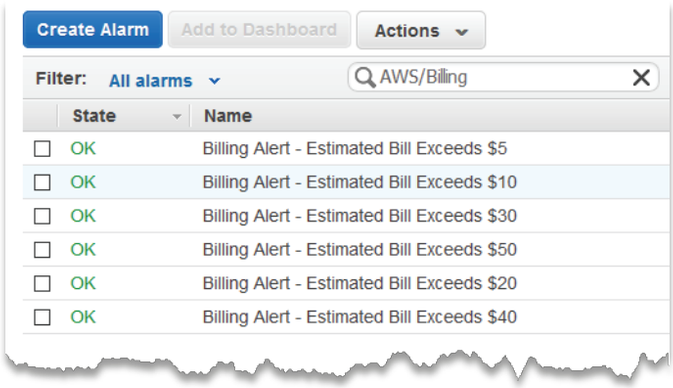
2. Enter the desired billing alert thresholds for your account. For example, if you want to be notified when your estimated monthly bill exceeds $1, $5, $10, $20, and $50, enter **1,5,10,20,50**.



3. Enter your **email address**.

4. Click **Next**, **Next**, and **Create**.

5. **Confirm your subscription** to the SNS topic using the link in the confirmation email.
6. To see the billing alerts created, navigate to the **CloudWatch** console.
7. Click on **Billing** in the navigation pane to view the new alarms created. Feel free to review or modify the alarms as needed.



8. Navigate to the **Simple Notification Service (SNS)** console.
9. Click **Topics** on the left.
10. Click the **Billing-Alerts** topic to view the email subscription already created by the template.

You did it. You are done with this lab.

Conclusion: In this lab, participants created billing alarms to ensure there are no unexpected bills from experimentation and exploration of the AWS platform.

Lab 2 – Creating a Backup Retention Schedule

In Lab 2, you deployed two AWS Lambda functions to take and delete snapshots of EBS volumes attached to your EC2 instances. However, everything you did was manual, meaning there was no scheduler that executed the Lambda functions automatically. This lab will walk you through the use of CloudWatch Events to schedule the execution of the two Lambda functions for a more comprehensive backup strategy.

---

1. Using AWS **CloudFormation**, deploy the **EBS-Snapshots** stack again. Use the same CloudFormation template used in Lab 2:

   https://s3.us-east-2.amazonaws.com/ahead-innovation-days/aws201/lab1/ebs_snapshots.yml

2. Navigate to the **CloudWatch** console. Select **Events** on the left.

3. Click **Create Rule**.

4. In the **Event Source** section, select **Schedule**.

5. Select the radio button for **Cron expression** and type the following: **0 2 * * ? ***

   ***Note**: This Cron expression states that the rule will run every day at 2:00 UTC, which is 10:00PM EST. Once you enter the expression, it will display the next 10 events to ensure your command is correct. Cron expressions are written as: Minutes | Hours | Day of Month | Month | Day of Week | Year.*

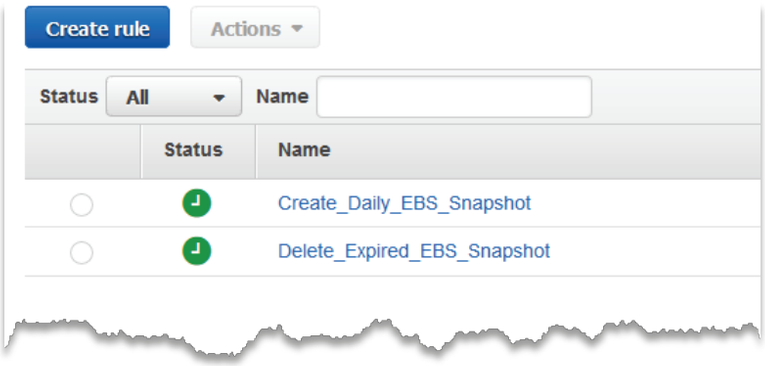
6. On the right, click the **Add target** button.

7. Select **Lambda function** from the drop-down. Select **EBS-Snapshot-Create-Function** from the Function* drop-down.

8. Click **Configure Details**.

9. Enter a name for the rule, perhaps **Create_Daily_EBS_Snapshot**. Enter a Description if you'd like.

10. Click **Create rule**.



11. Create a second rule for the **EBS-Snapshot-Delete-Function**, but use the Cron expression:

    0 1 * * ? *

    ***Note**: This will execute the EBS-Snapshot-Delete-Function every day at 1:00 UTC, which is 9:00PM EST.*

12. Give the rule the name **Delete_Expired_EBS_Snapshot**.

13. In the **EC2** console, **tag any instances** that you want snapshots with the **Backup** and **Retention** tag, as you did in Lab 2.

That's it. You are done with this lab.

Conclusion: In this lab, participants created CloudWatch Events to schedule the AWS Lambda functions to automatically execute at certain times of the day. This will help create a more comprehensive backup strategy for EC2 instances.

Lab 3 – Using Snapshots to Restore Files

In Lab 2, you took snapshots of an EC2 instance using the Lambda functions. However, you didn't do anything with the snapshots, such as restore any data. In this lab, you'll manually create a snapshot, delete some data on the EC2 instance, and then try to recover those files using the snapshot.

---

1. Using **CloudFormation**, deploy the initial stack as you did in Lab 1 using the following URL. This will deploy all the resources needed for this lab.

   https://s3.us-east-2.amazonaws.com/ahead-innovation-days/aws201/lab1/toplevel.yml

2. Navigate the **EC2** console. Click on **Volumes**.

3. Select the **Volume** attached to the **Windows Instance – Lab 2** instance. You can find information on the volumes by selecting them and viewing the information in the bottom pane.

4. Under the Action menu, select **Create Snapshot**.

5. Enter the name of **Manual Snapshot**.

6. Click on **Snapshots** in the navigation pane and **refresh** the list until the snapshot changes from a status of 🟡 Pending to 🟢 Completed.

7. In the EC2 console, select **Security Groups** in the navigation pane.

8. Select the **ahead-lab-rep-access** security group and click the **Inbound** tab at the bottom.

9. Click **Edit**.

10. Modify the existing rule so the source is **0.0.0.0/0** rather than 10.0.0.0/16.

11. Click **Save**.

12. Using **Remote Desktop**, connect to the **EC2** instance. Don't forget to grab the administrator password by decrypting it using the EC2 Key Pair. For a refresher, check out Lab 2.

13. Open **File Explorer** and view the files located in C:\AHEAD-Lab-Files.

    ***Note**: These are photos of our amazing office in downtown Chicago. Make sure to connect with your account rep to schedule some time in our office and to check out our incredible briefing center.*

14. Delete the folder **C:\AHEAD-Lab-Files**.

15. In the EC2 console, click **Snapshots**

16. Select the snapshot, and click **Create Volume** from the Actions menu.

17. Leave all the defaults and click **Create Volume**.

18. Click the volume name in the confirmation box or select Volumes in the EC2 console and **select the new volume**.

19. From the Actions Menu, select **Attach Volume**.

20. In the Instance field, select **Windows Instance – Lab 2**. Leave the device field as the default.

21. Click **Attach.**

22. Open your **Remote Desktop** connection to the instance and open **Disk Management**. This can be done by right-clicking the Start Menu and selecting Disk Management.

23. You should have a second disk listed as Disk 1.

24. Right click Disk 1 and choose **Online**.

    ***Note**: You have to click the left-side, the gray box where it says Disk 1.*

25. The disk should come **online** and Windows will likely assign it as D: drive.

26. Using **File Explorer**, browse to D: drive and see that the AHEAD-Lab-Files folder still exists, as you took the snapshot before you deleted the folder.

27. Copy the files from **D:\AHEAD-Lab-Files** back to the original location of **C:\AHEAD-Lab-Files**.

28. Now that you've restored the files, feel free to **view** those photos of the beautiful AHEAD office again!

That's it. You are done with this lab.

Conclusion: In this lab, participants took a manual snapshot of a running instance, purposely deleted files on the instance, and using the snapshot to restore the original files.

Lab Cleanup:

Don't forget to detach the additional volume, delete the volume, and delete the snapshot. Then delete the CloudFormation stack. If you need help, refer to Lab 6.

Presenter Bios

Bryan Krausen

@btkrausen
www.itdiversified.com

Bryan Krausen is a Senior Solutions Architect at AHEAD who focuses on AWS cloud and solutions. With over 17+ years of experience in enterprise IT, he has experience with a wide variety of platforms including storage, compute, networking, and cloud solutions.

Bryan is the community leader for both Louisville AWS User Group and Louisville VMUG and has presented at numerous VMUG and AWS User Group meetings across the US.

AWS Certifications

- AWS Certified DevOps Engineer - Professional
- AWS Certified Solutions Architect – Professional
- AWS Certified Advanced Networking – Specialty
- AWS Certified Solutions Architect - Associate
- AWS Certified SysOps Administrator – Associate
- AWS Certified Developer - Associate

Eric Shanks

@eric_shanks
www.theithollow.com

Eric Shanks is a Senior Solutions Architect with over 15 years of data center experience focusing on hybrid cloud and automation. He has some of the industry's highest distinctions including two VMware Certified Design Expert (VCDX #195) certifications and many others across a variety of solutions including Microsoft, Cisco, Amazon Web Services, etc.

Eric has become a community contributor through his work as a Chicago VMware Users Group (VMUG) leader, blogger, and frequent Tech Field Day delegate. He has been a keynote speaker around the country for several chapters of VMUG and at VMware's conference, VMworld. He's been part of community podcasts such as "The Datanaughts" and is also invited to cover product shows such as HPE's Discovery conference.

AWS Certifications

- AWS Certified Developer - Associate
- AWS Certified Solutions Architect - Associate
- AWS Certified SysOps Administrator - Associate
- AWS Certified DevOps Engineer - Professional
- AWS Certified Solutions Architect - Professional

Greg Thursam

@gregthursam
www.between2clouds.blog

Greg Thursam is the Solutions Principal – AWS and DevOps at AHEAD. He has over 14 years of industry experience in enterprise IT architecture including storage (deep expertise in NAS), network (SAN, WAN, LAN, etc.), security, and compute. Greg has worked for a variety of companies through his tenure, covering startups to one of the largest OEMs. He has held a variety of roles, including Storage / Server / Network / Security Administrator, Storage Architect, and Pre-Sales Engineer. His vast experience allows him to collaborate with AHEAD clients and manufacturers around the acquisition and consumption of data center technology.

AWS Certifications

- AWS Certified Developer - Associate
- AWS Certified Solutions Architect - Associate
- AWS Certified SysOps Administrator - Associate
- AWS Certified Solutions Architect - Professional

Adam Youngblood

@automation_adam
www.cloudautomation.blog

Adam Youngblood is a Cloud Specialist who helps enterprises design, architect and operate everything that encompasses private and public cloud while assisting in operational change management, showback, ESM, APM, GRC and other key features to a meaningful and fully automated enterprise cloud utilizing market leaders in ESM, automation, orchestration and configuration management.

AWS Certifications

- AWS Certified Developer - Associate
- AWS Certified Solutions Architect - Associate
- AWS Certified SysOps Administrator - Associate