Module 2: Security Management in AWS

Demo Document 8

Title: Setup AWS Config to Evaluate Compliance For the Resources in Your Environment

Use Case:

You are the Cloud Engineer in one of the Pharma company who has multiple workloads running in their AWS account. Your IT manager wants you to ensure a proper security checks are put in place for compliance and take appropriate remediate action in case of non-compliance. You are instructed to deploy AWS Config solution that should meet following requirements.

1. **Requirement 1**
   a. Check whether your running EC2 instances are using approved AMIs.
   b. If non-compliance found, configure Remediation Action to stop the EC2 instance automatically.
   c. Identify who created the non-compliant resource using AWS CloudTrail

2. **Requirement 2**
   a. Identify if public read access enabled for any S3 bucket.
   b. If non-compliance, configure Remediation Action to send notification to Security Team
   c. Identify who created the non-compliant resource using AWS CloudTrail

Objective:

You will learn how to setup AWS Config to evaluate compliance for the resources in your environment.

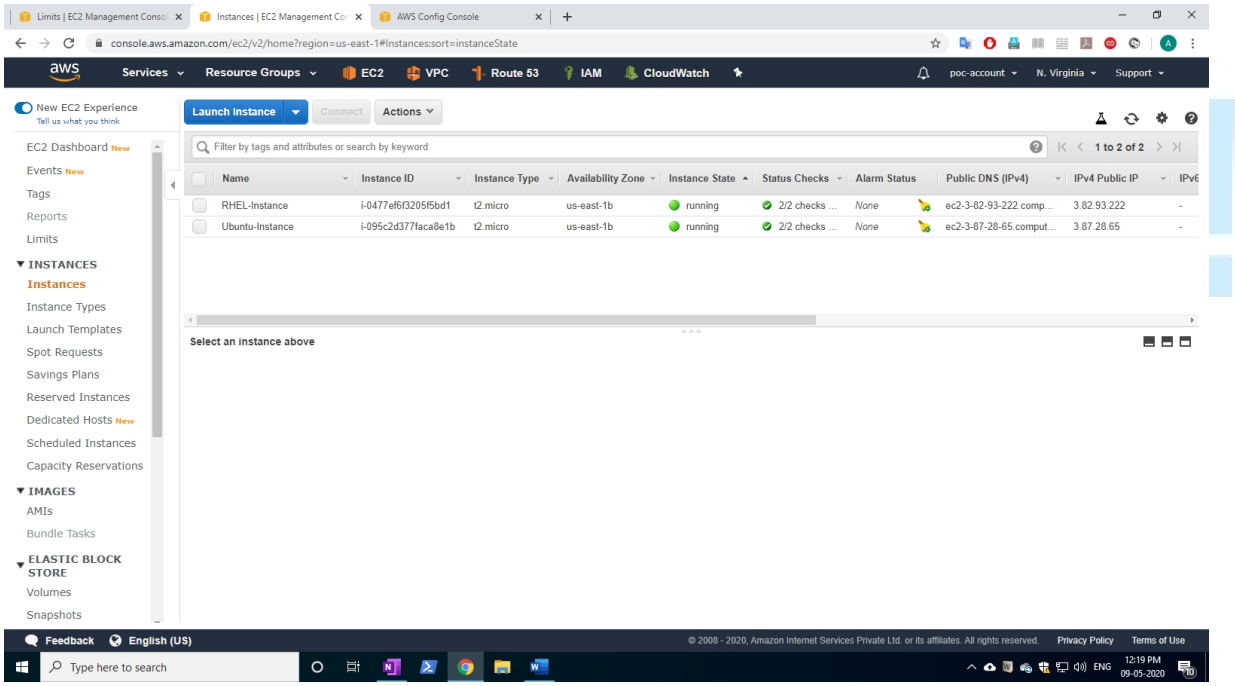
Topics Covered:

- Provision **AWS EC2** servers
- Create **IAM** role for AWS Config Remediation Automation
- Setup initial **AWS Config**
- Create **AWS Config rule** to identify non-compliance as mentioned in the requirements
- Create **Remediation Action** in AWS Config as mentioned in the requirements

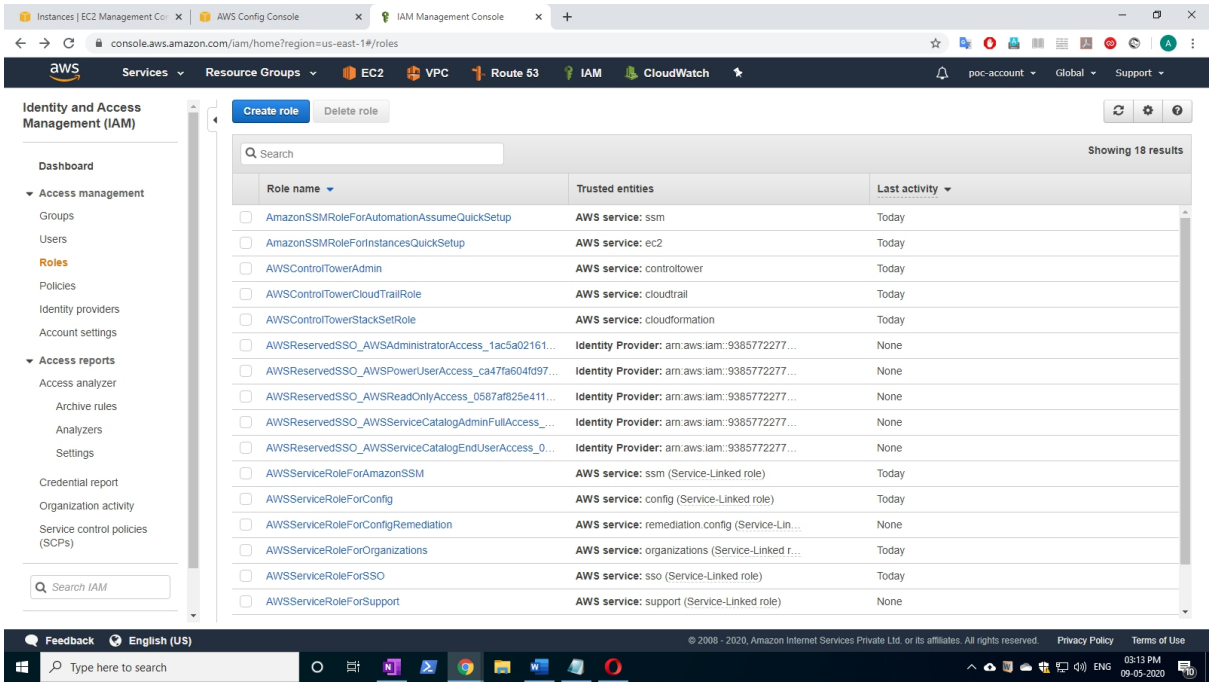
Solution Steps:

[Note: us-east-1 region is used to provision all resources in this lab]

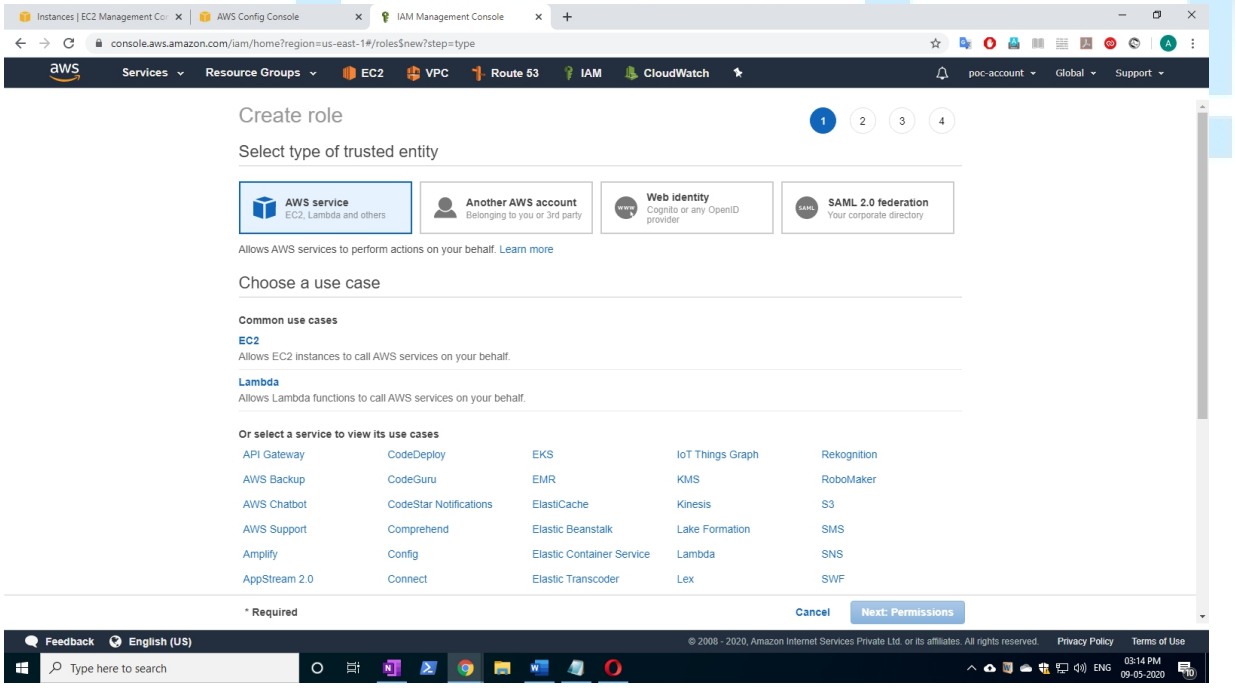
1. Create 2 EC2 instances to evaluate compliance as follows.
   - A RHEL instance, (Considering it will be is *compliant* to your upcoming rule)
   - An Ubuntu instance, (Condsidering it will be *non-compliant* to your upcoming rule)



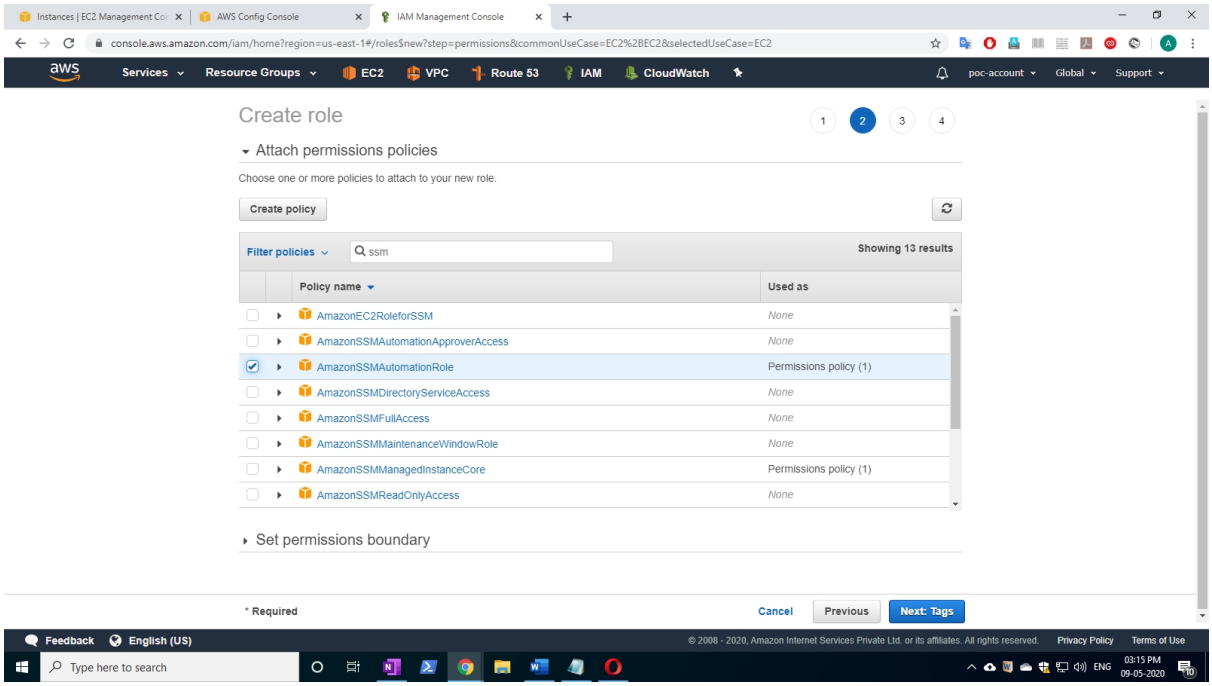
2. Create an IAM role for Automation Role
   - Navigate *IAM -> Roles -> Create role*



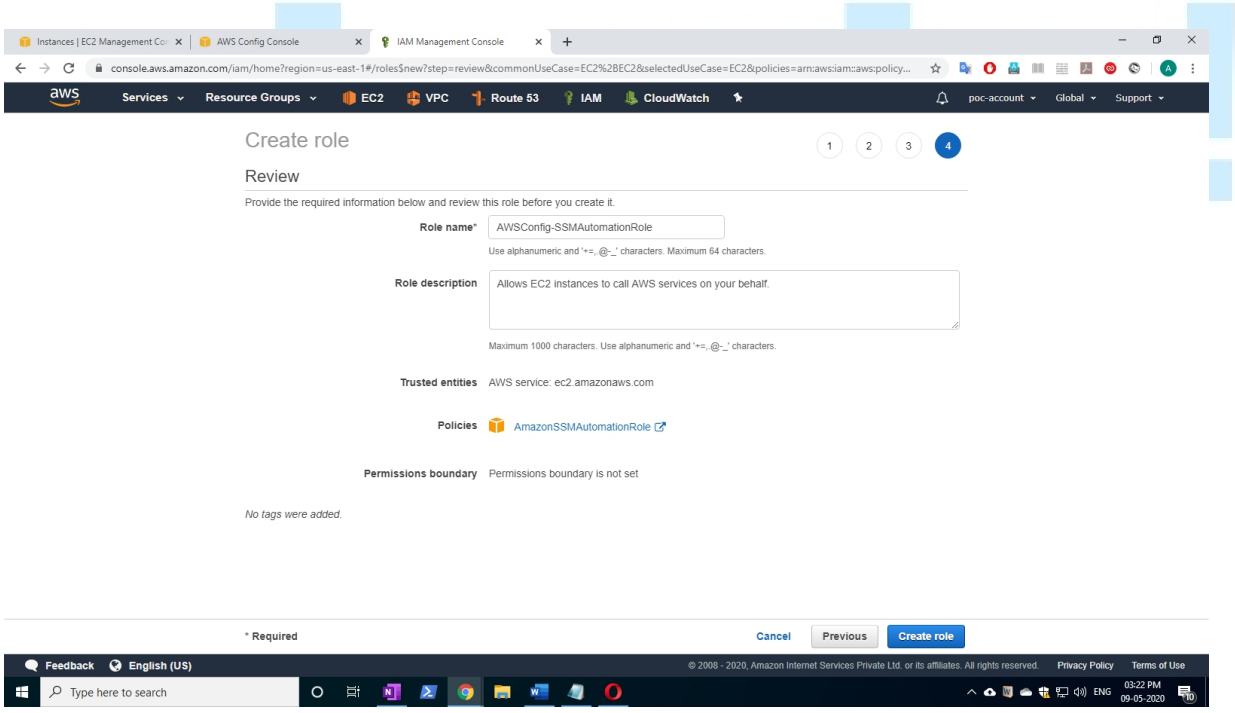
- Navigate *Create role -> AWS service -> EC2*
- Click *Next Permissions*



- Select *Attach permissions policies -> AmazonSSMAutomationRole*

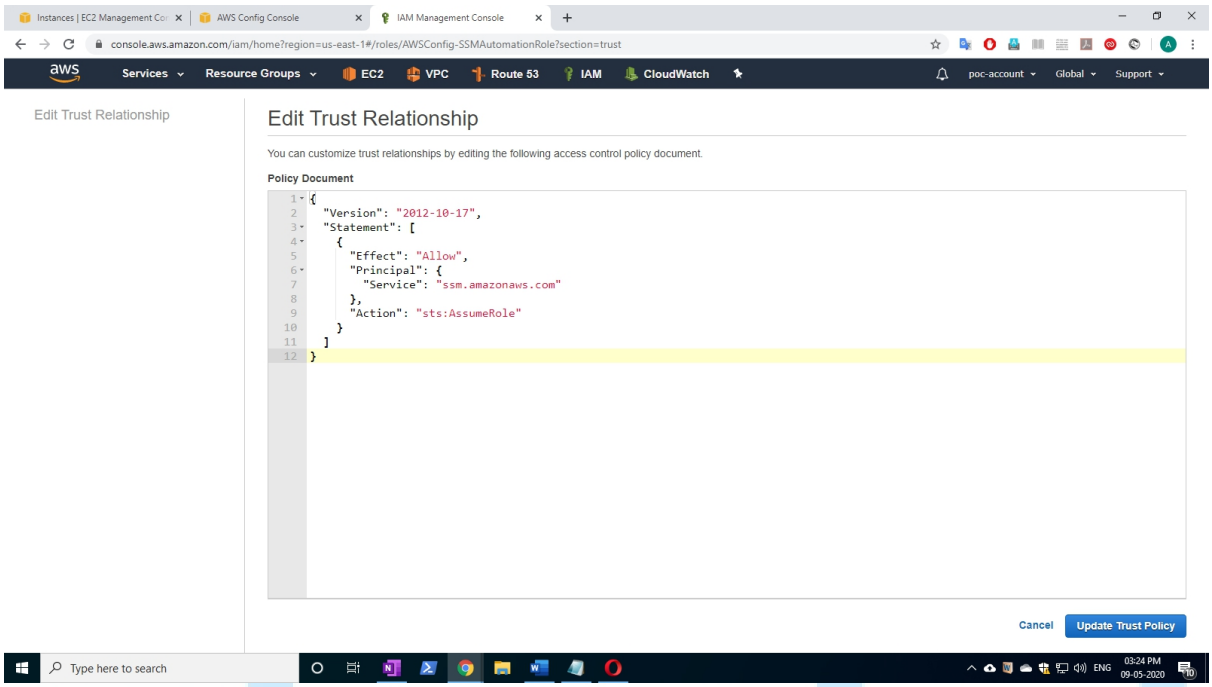


- Give a suitable name for *Role name* and click *Create*

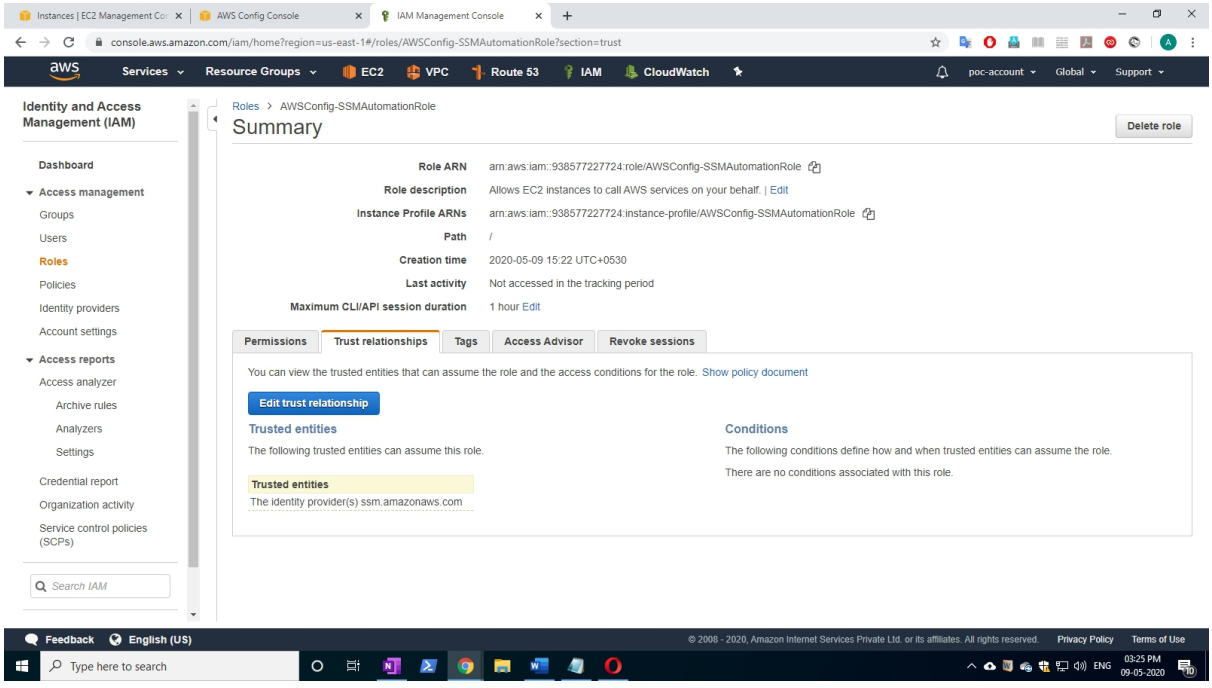


- After the IAM role is created, go to the summary section of the role
- Navigate *Trust relationship -> Edit Trust relationship*
- Replace *"ec2.amazonaws.com"* with *"ssm.amazonaws.com"*
- Click *Update Trust Policy*

[The reason why you are doing this is, AWS Config internally calls SSM Automation Document to perform Remediation Action. The SSM Automation requires IAM roles to perform Automation Actions such as EC2-Stop, EC2-Start etc.]

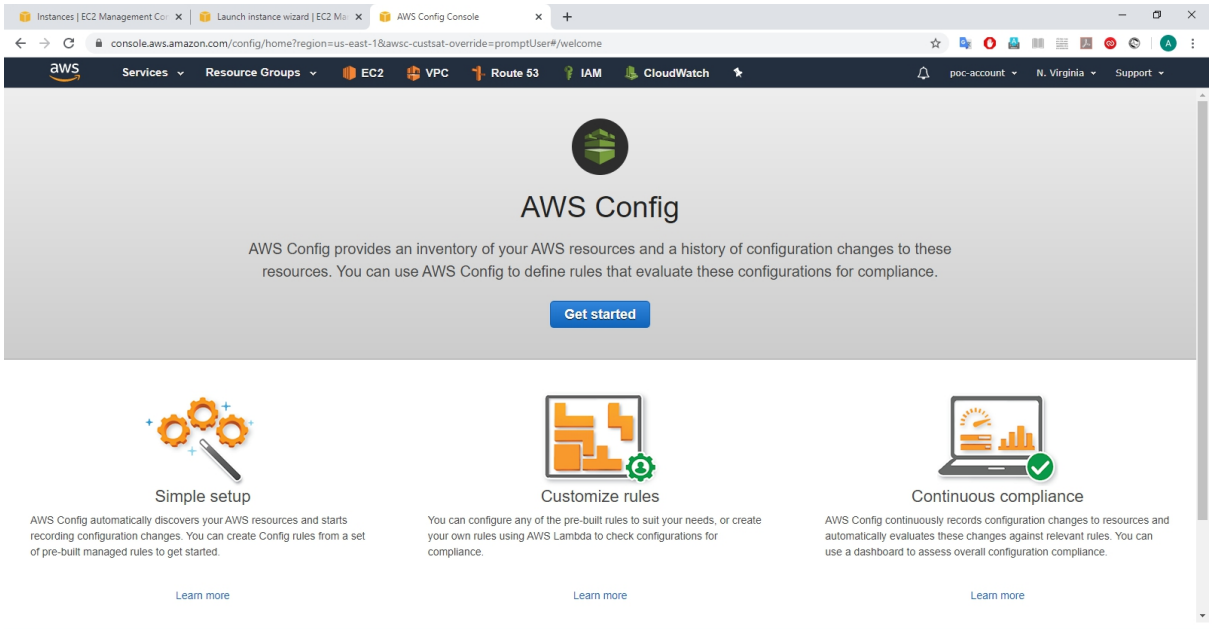


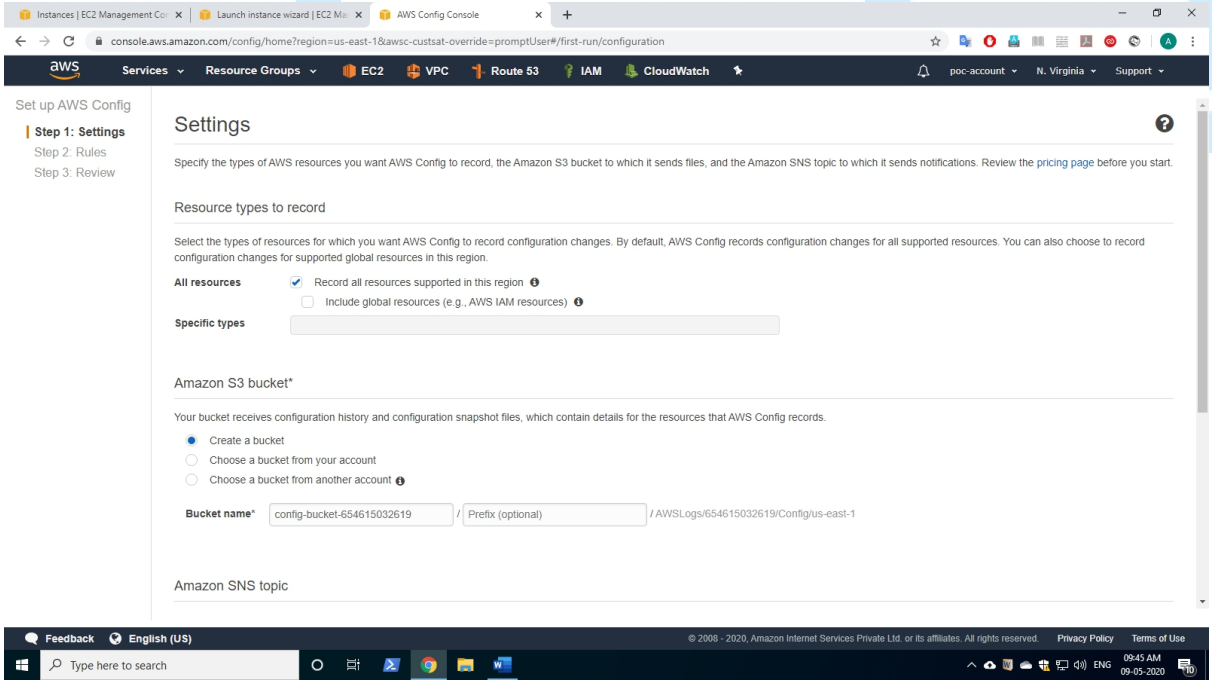
- You should see the *Trusted entities* showing *ssm.amazonaws.com* under *Trust relationship* tab of the role summary

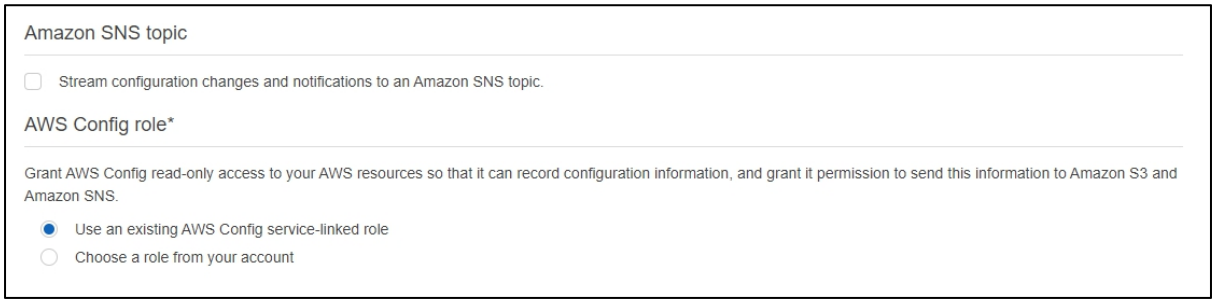


- Now IAM role is created successfully. You can move on to next section to create AWS Config service and configure the service using the IAM role created above.

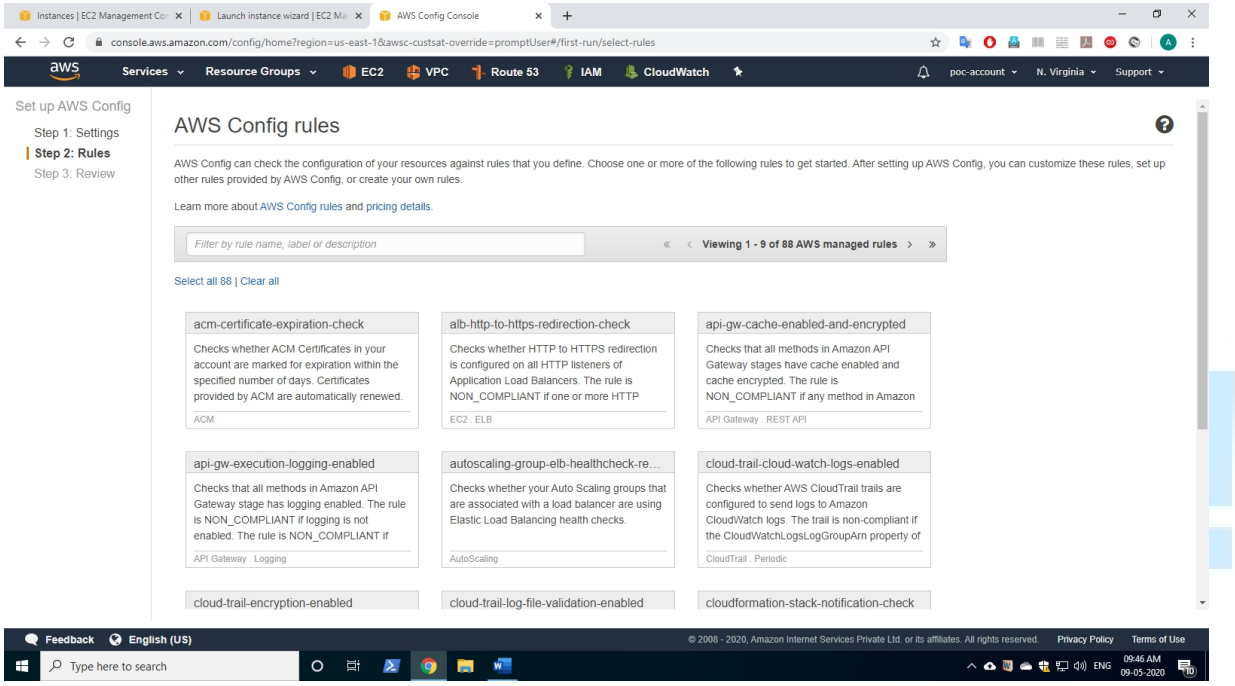
3. Setup AWS Config service
    - Launch AWS Config and click Get Started



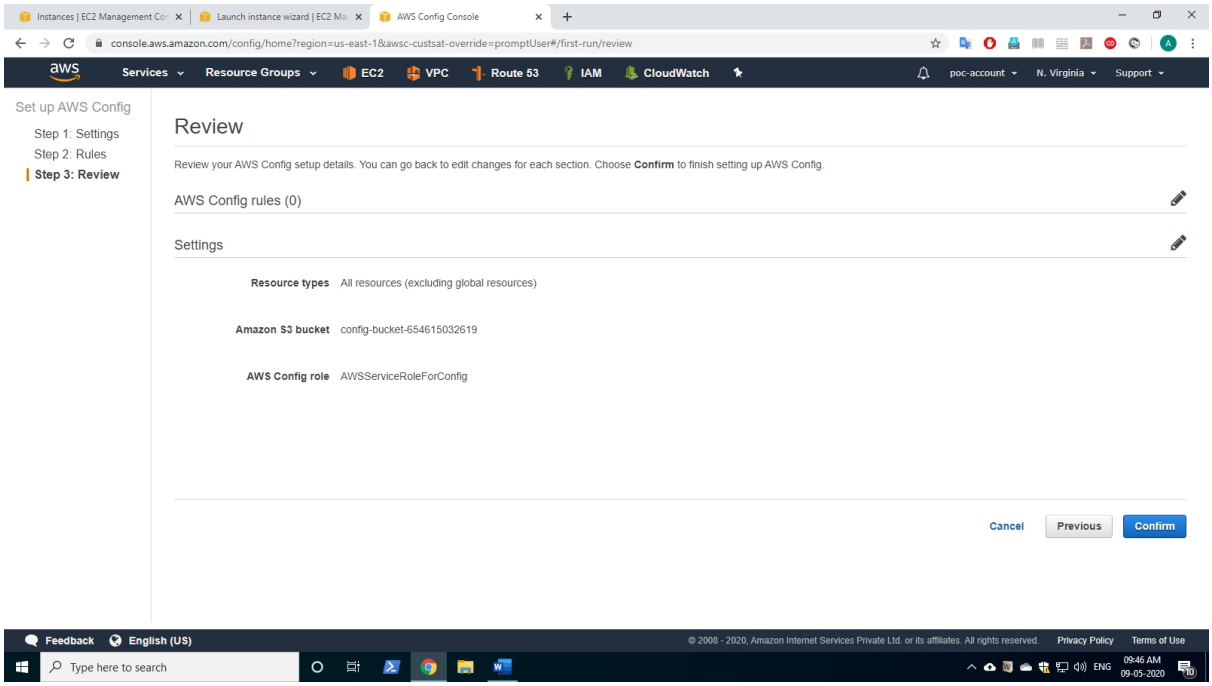




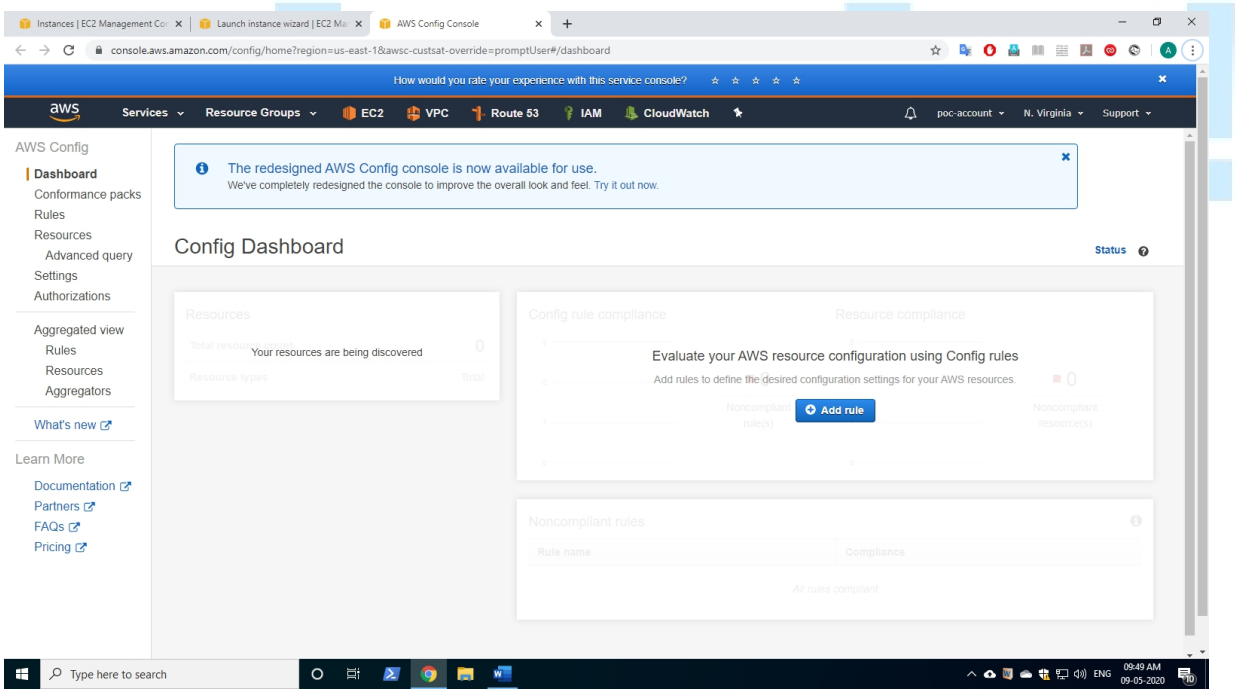
- Click *Next* in the *Rules*



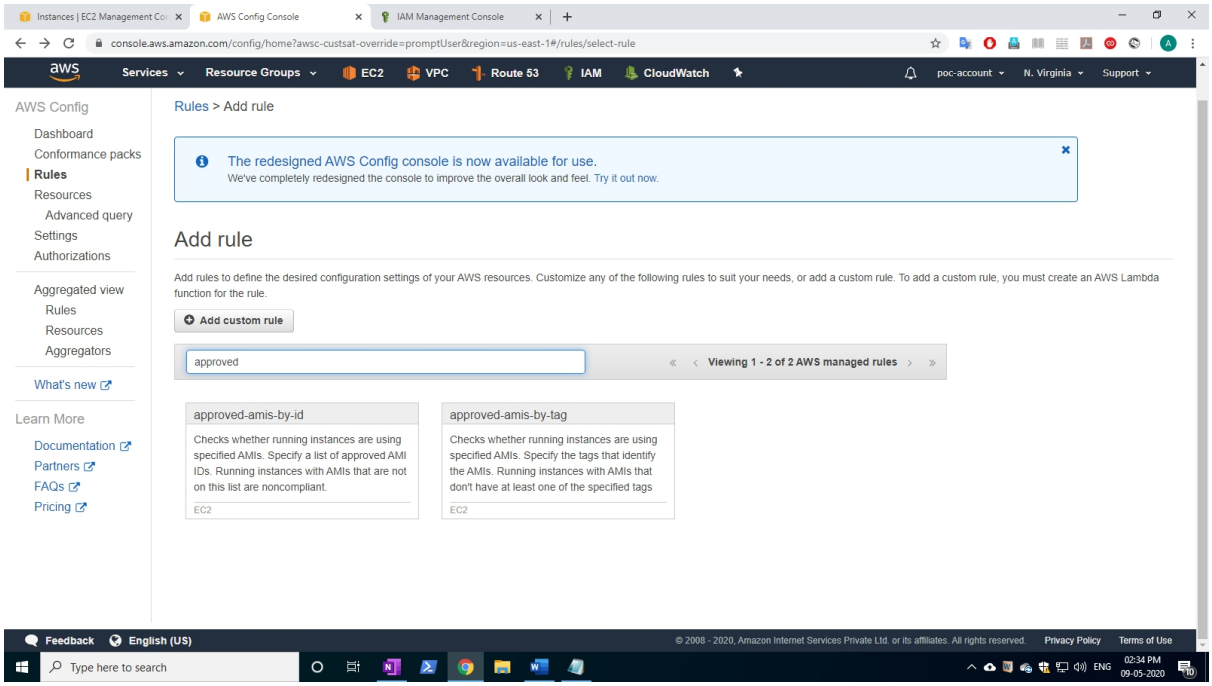
- *Review* and click *Confirm*



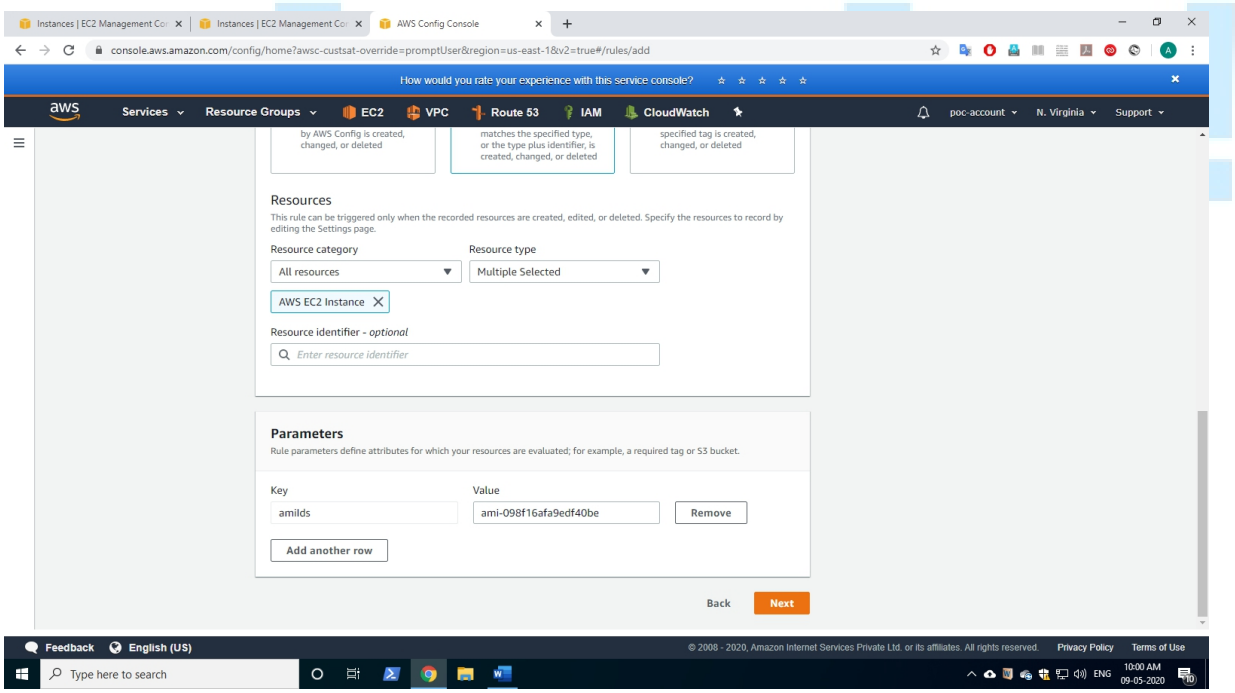
- You should see the *Dashboard* now



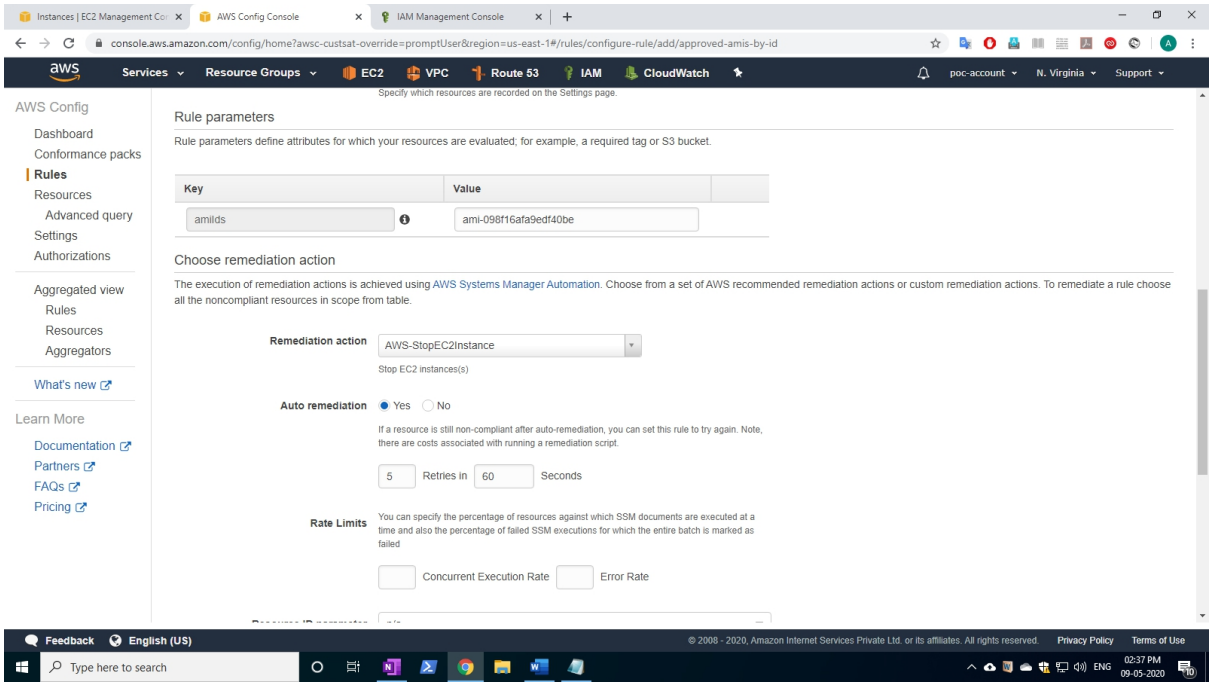
- Click *Add rule* in the dashboard. Use search bar to search and find the rule *approved-amis-by-id*



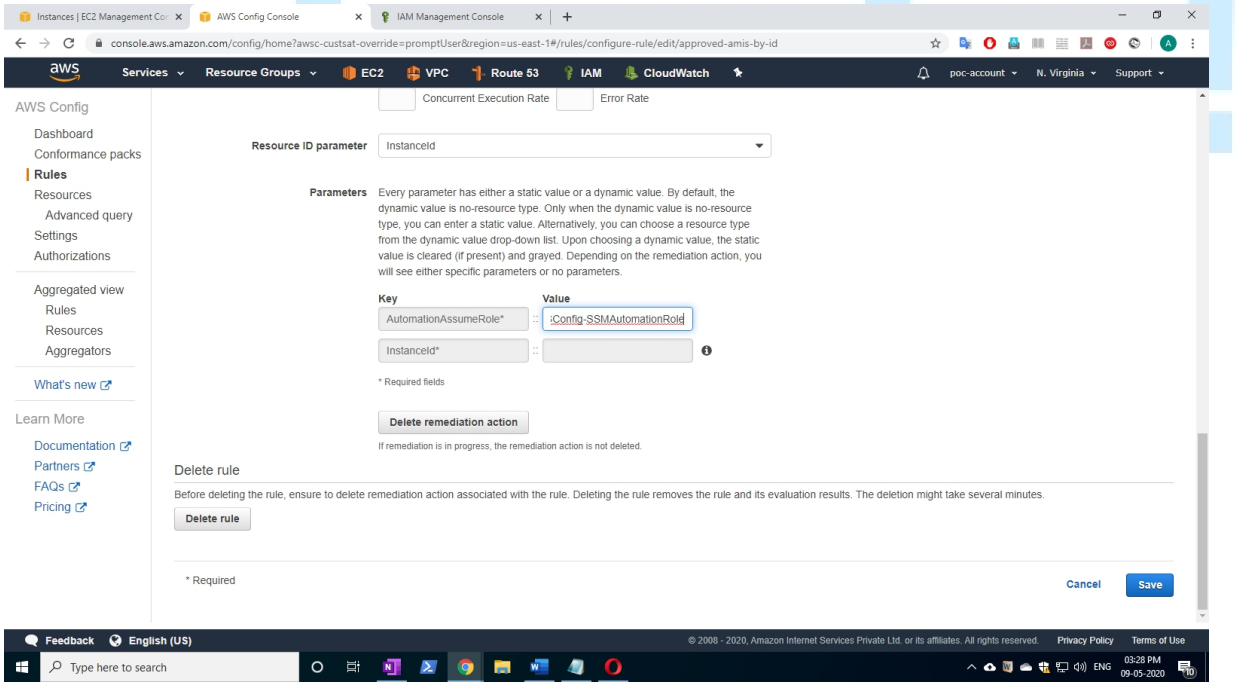
- Click *Next*



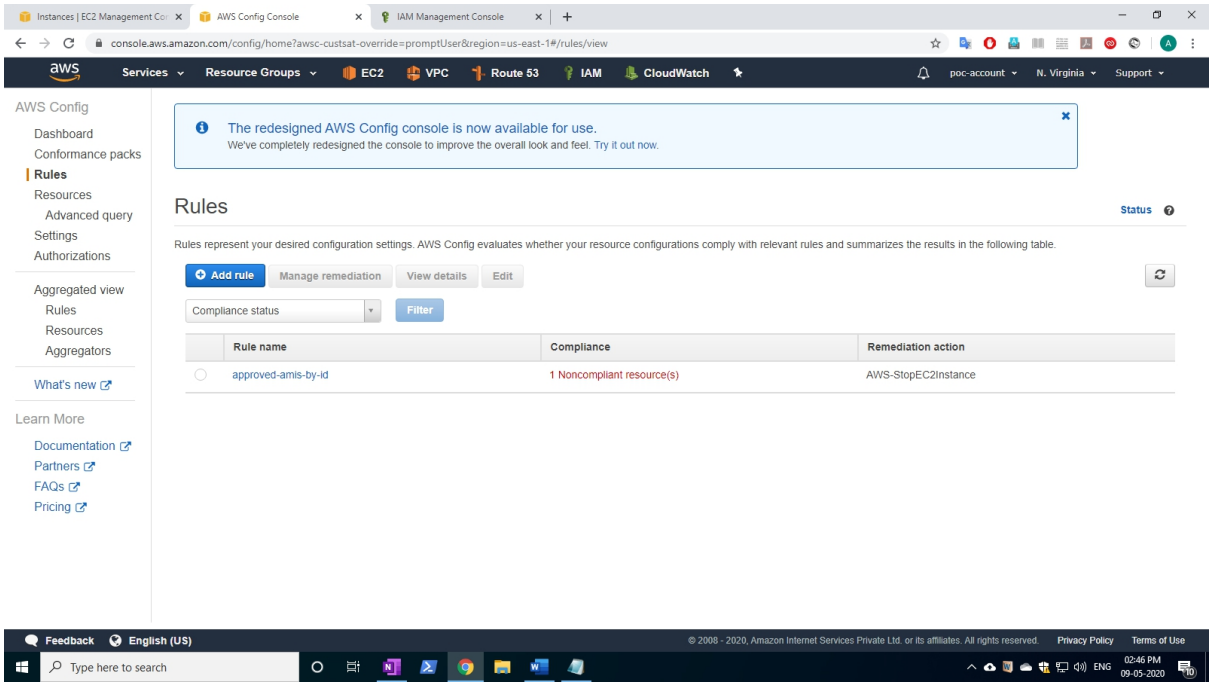
- Under Parameters section, copy and paste the ami-id you want to evaluate with.
- For this demo, we are evaluating compliance for RHEL 8 image with ami-id *ami-098f16afa9edf40be.*
- To get the AMI ID of your RHEL instance, go to EC2 Dashboard -> Click on Instances -> Select the RHEL instance. In the description section search for AMI ID.
- *Choose Remediation Action* section -> *Remediation Action -> AWS-StopEC2Instance*
- Select *Auto remediation -> Yes*



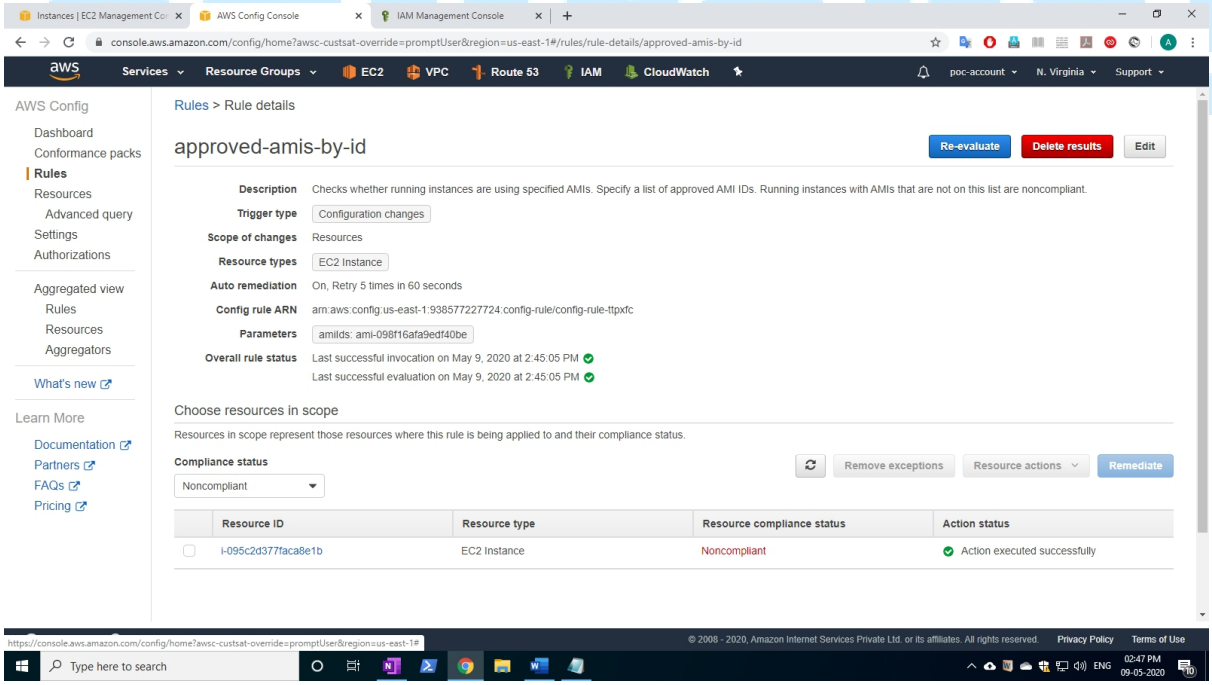
- Select *Resource ID parameter -> Instance Id*
- Fill the Value for Key *AutomationAssumeRole* with ARN of the IAM role you created earlier



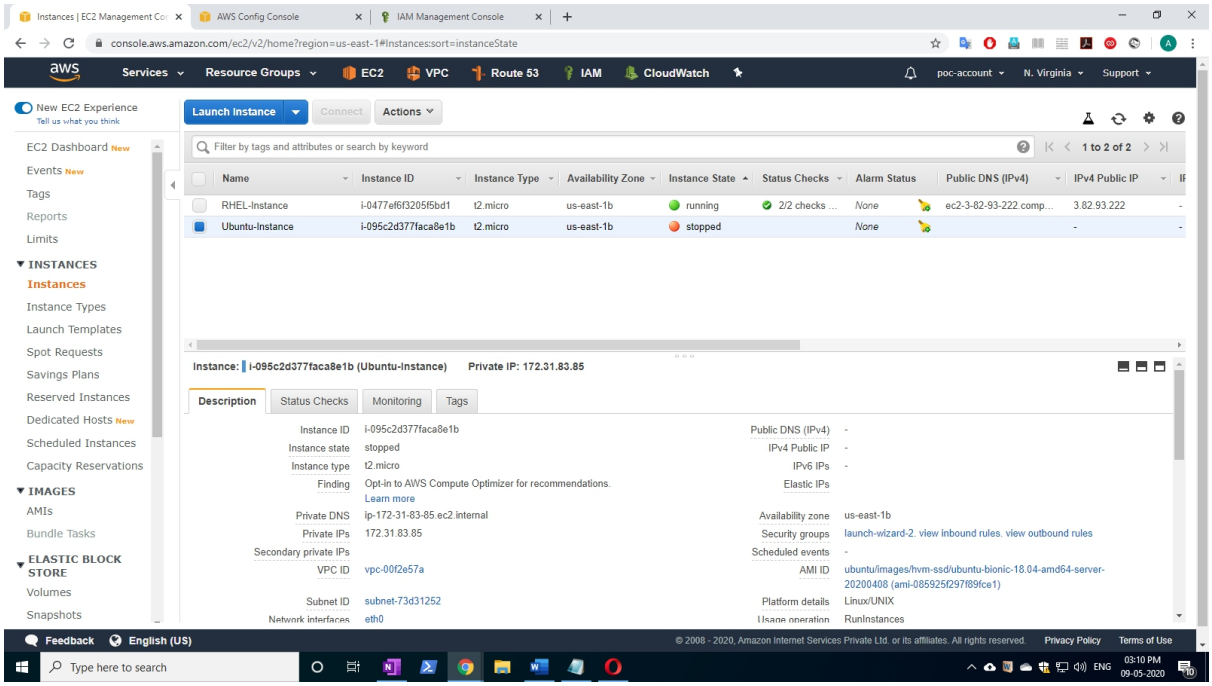
- Under *Rules* -> you can see the rule created just now



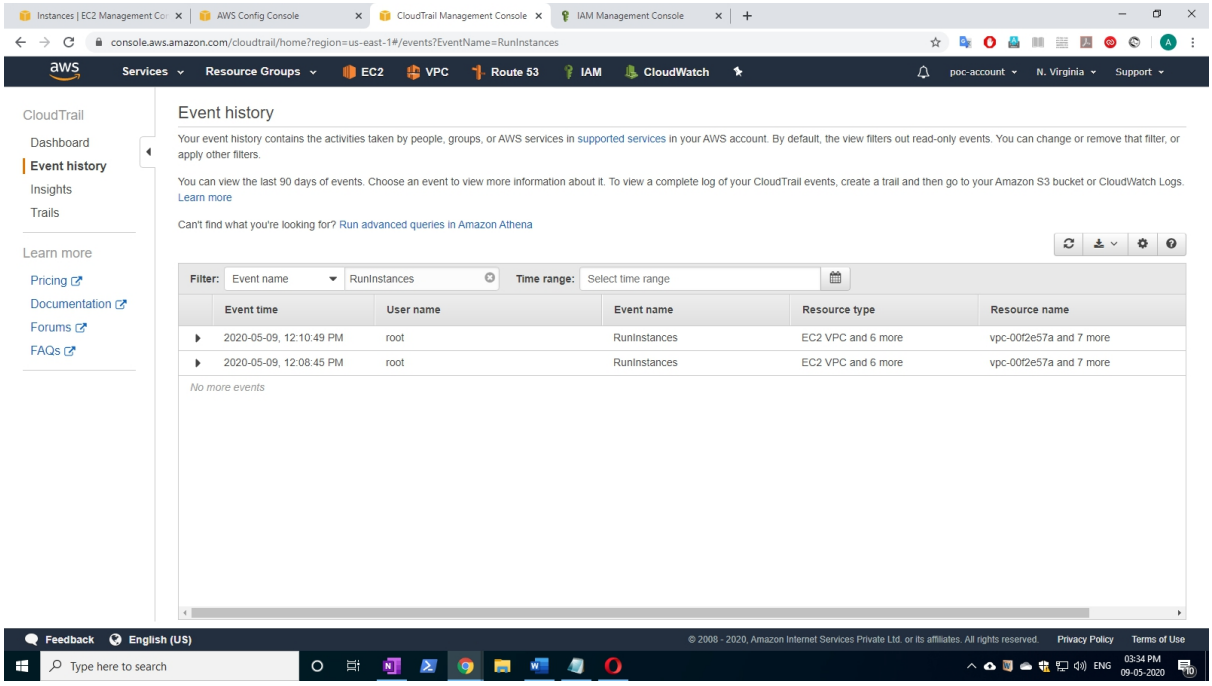
- Click the rule name, you can see the Action status for the noncompliant resource. If it shows "Action executed successfully" it means the remediation action is already taken place and the noncompliant EC2 instance is stopped.



- Go to EC2 console and verify the non-compliant EC2 instance is indeed stopped by *AWS Config Remediation Action*. Here the Ubuntu instance is non-compliant EC2 instance, so it is stopped automatically.



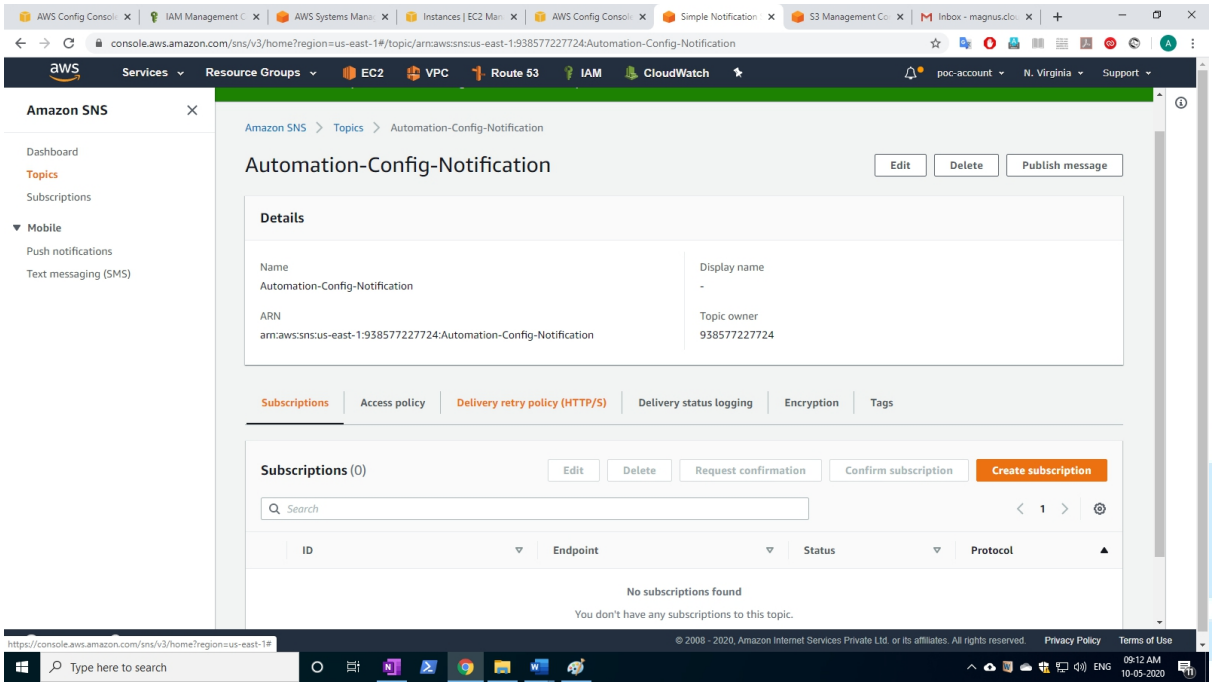
4. Identify who launched the non-compliant instance
   - Navigate to Open *CloudTrail -> Event History*
   - Filter *Event Name -> RunInstances*
   - Here you can see all the instances launched and who launched them. In below screenshot, you can see the entries for RHEL and Ubuntu instances launched and both of them are launched by user *"root"*.



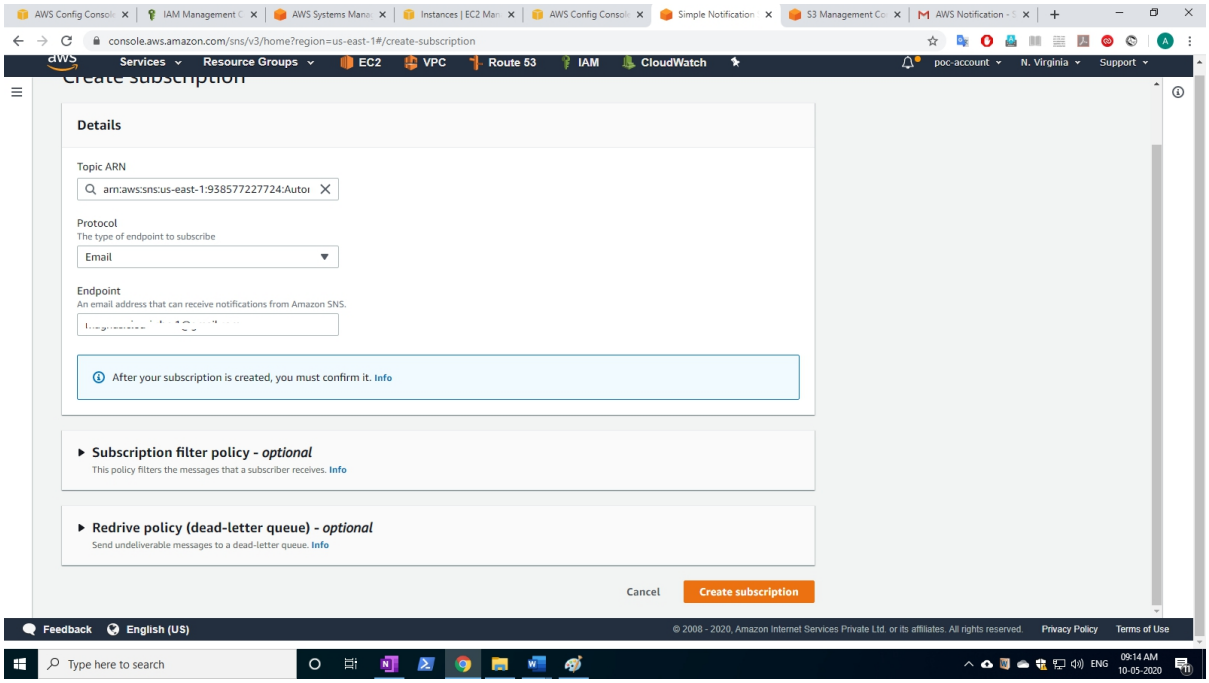
Requirement 2:

1. Create an SNS topic to send email to security team for non-compliance
    - Open *Amazon SNS* -> provide topic name as “Automation-Config-Notification” -> click *Create topic*
    - The SNS topic is successfully created as shown below.

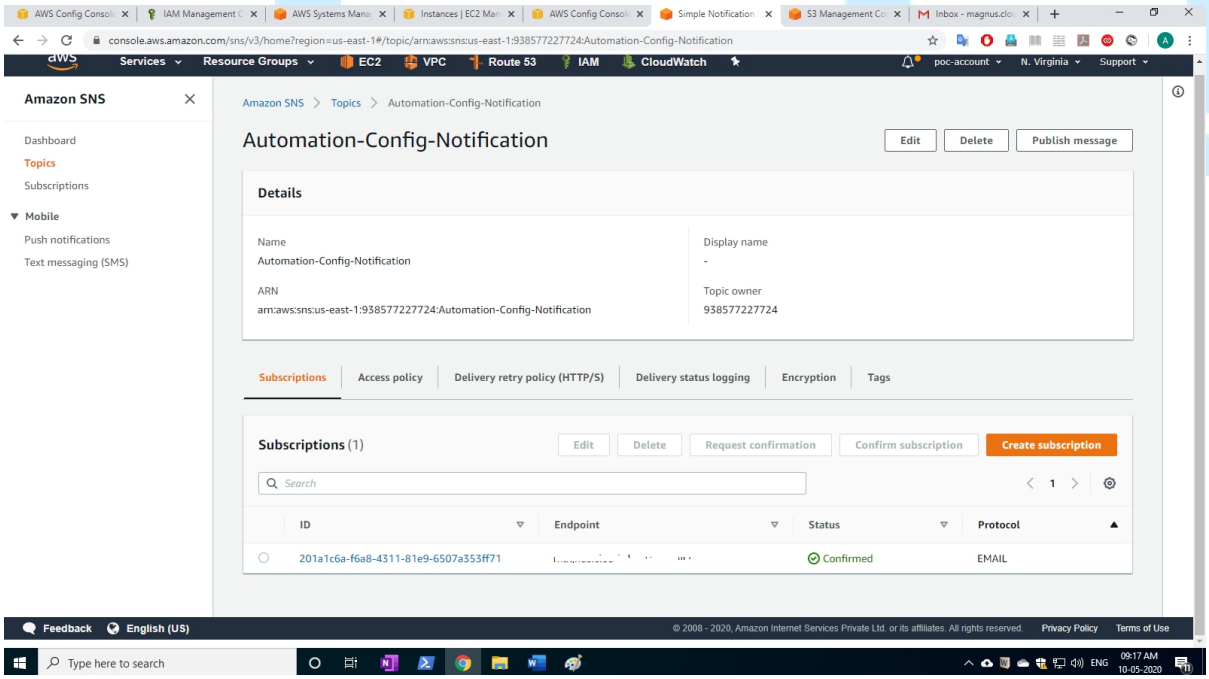
    [Note: The topic name should start with “Automation” for IAM role to be able to publish the notification as per the default permissions provided]



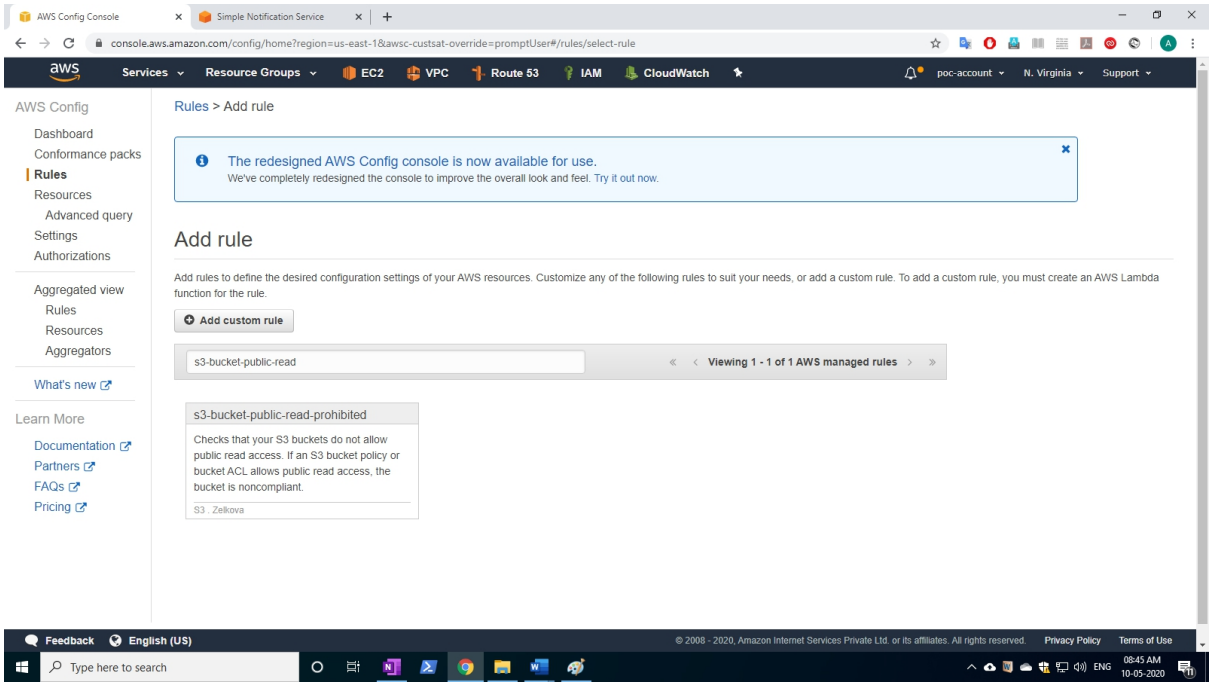
- Click *Create subscription* -> Select *Protocol as Email*
- *Endpoint - > Your email id*
- *Click Create Subscription*



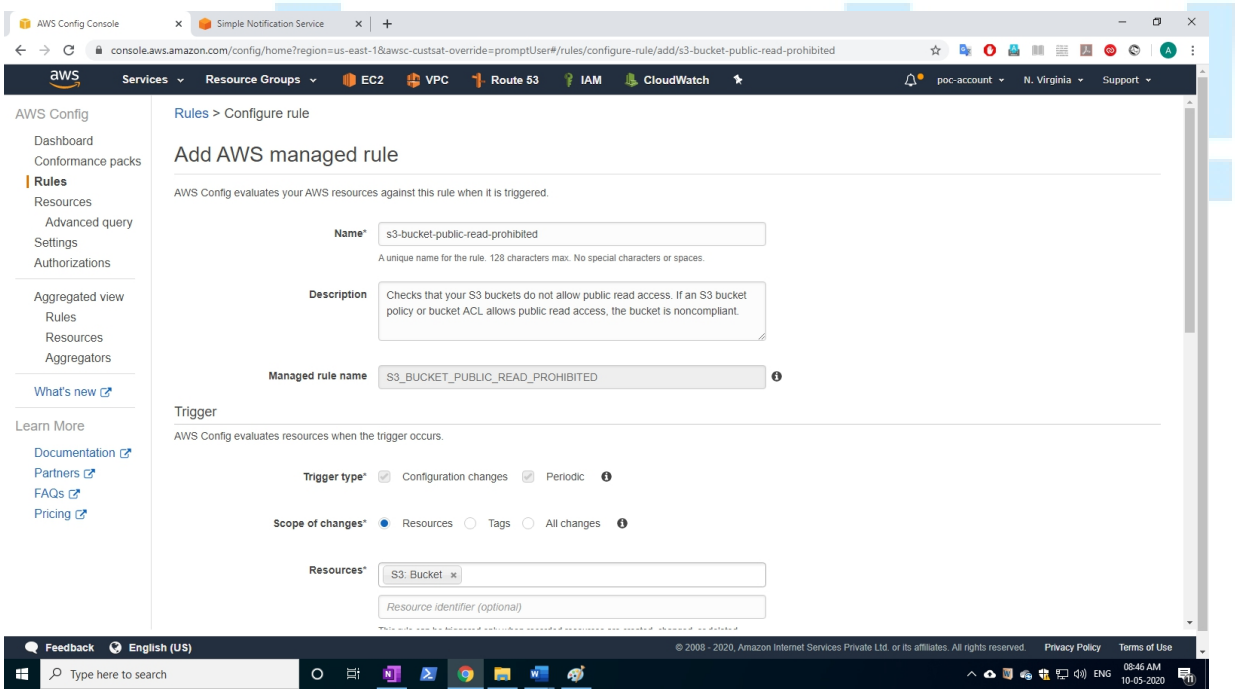
- You are will receive a notification in your email for confirmation. Click *Confirm subscription* in your email
- With this your SNS topic is fully configured and ready to send notification to your email id.



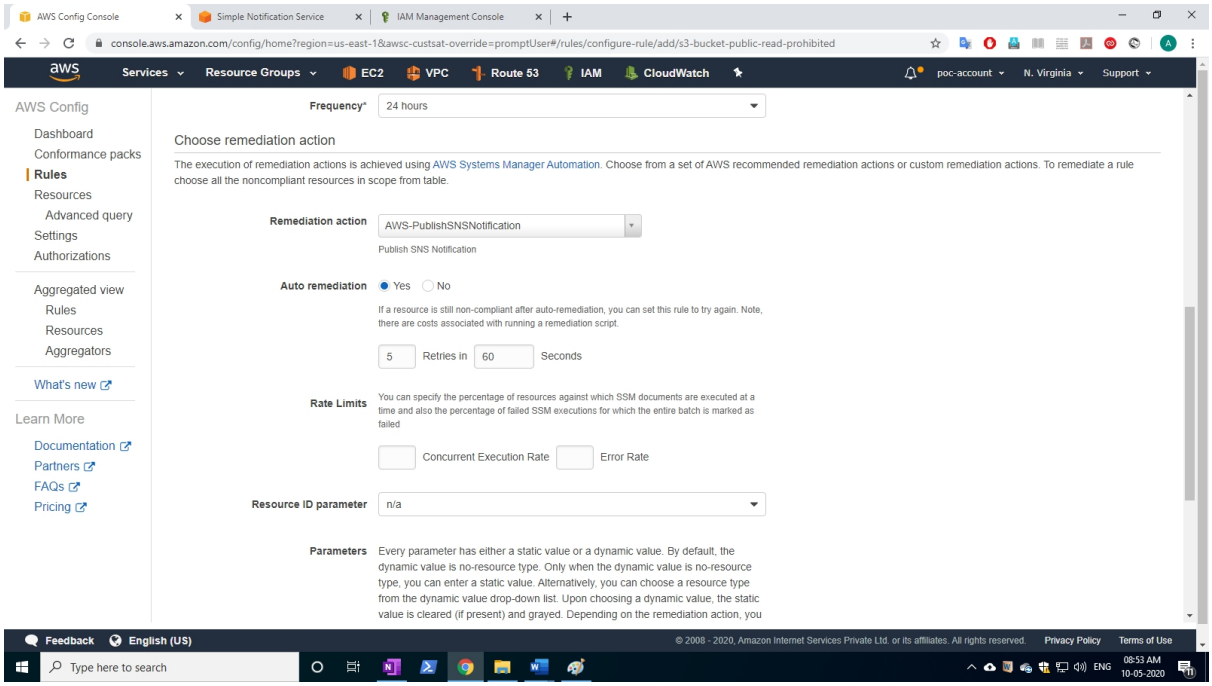
2. Create AWS Config rule for non-compliance
    - AWS Config -> Rules -> Add rule -> choose *s3-bucket-public-read-prohibited*



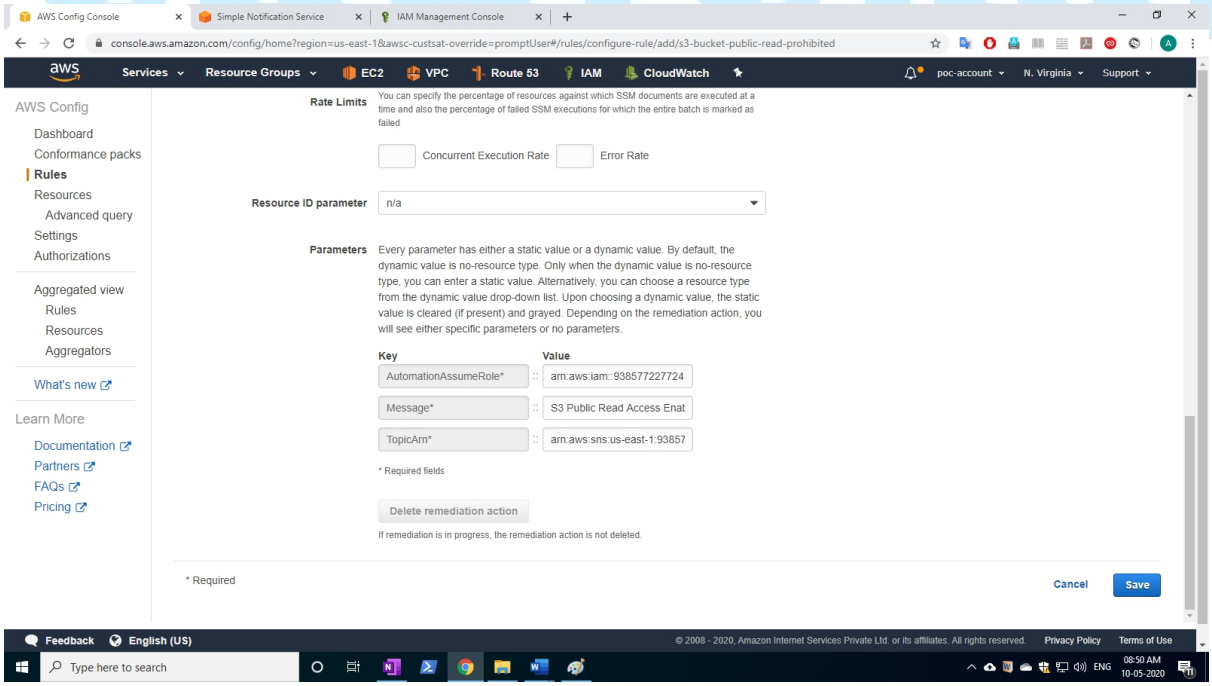
- You will see a page called “Add AWS managed rule”



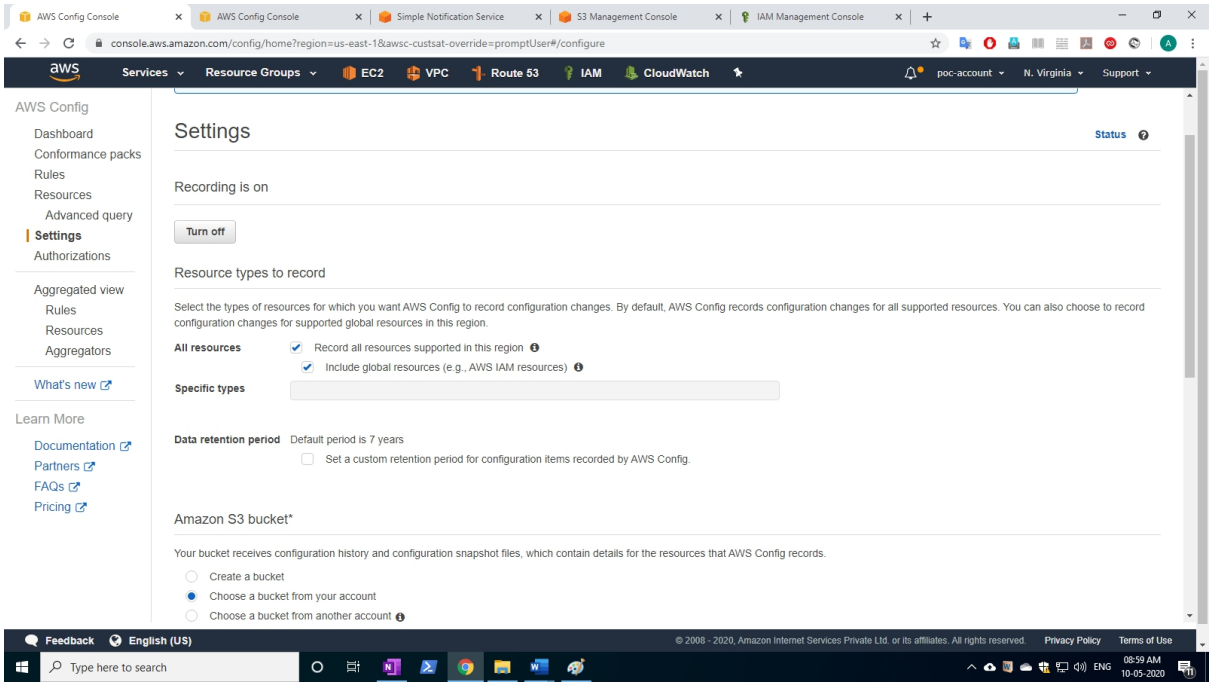
- Choose *Remediation action* -> Select *AWS-PublishSNSNotifcation*
- *Auto Remediation -> Yes*



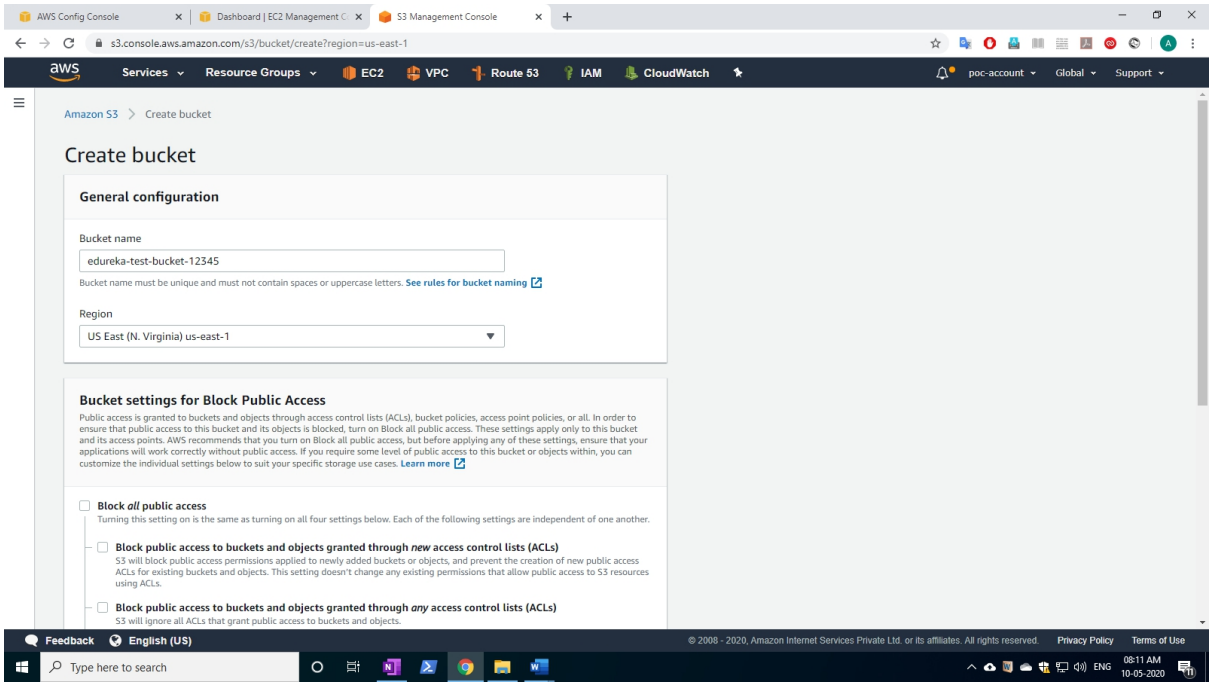
- Fill the Parameters as follows
  - *AutomationAssumeRole ->* The arn of the IAM role you created previously.
  - *Message ->* Any custom message you want to receive for non-compliance
  - *TopicArn ->* the *arn* of the SNS topic you created previously
  - Click on “Save” option

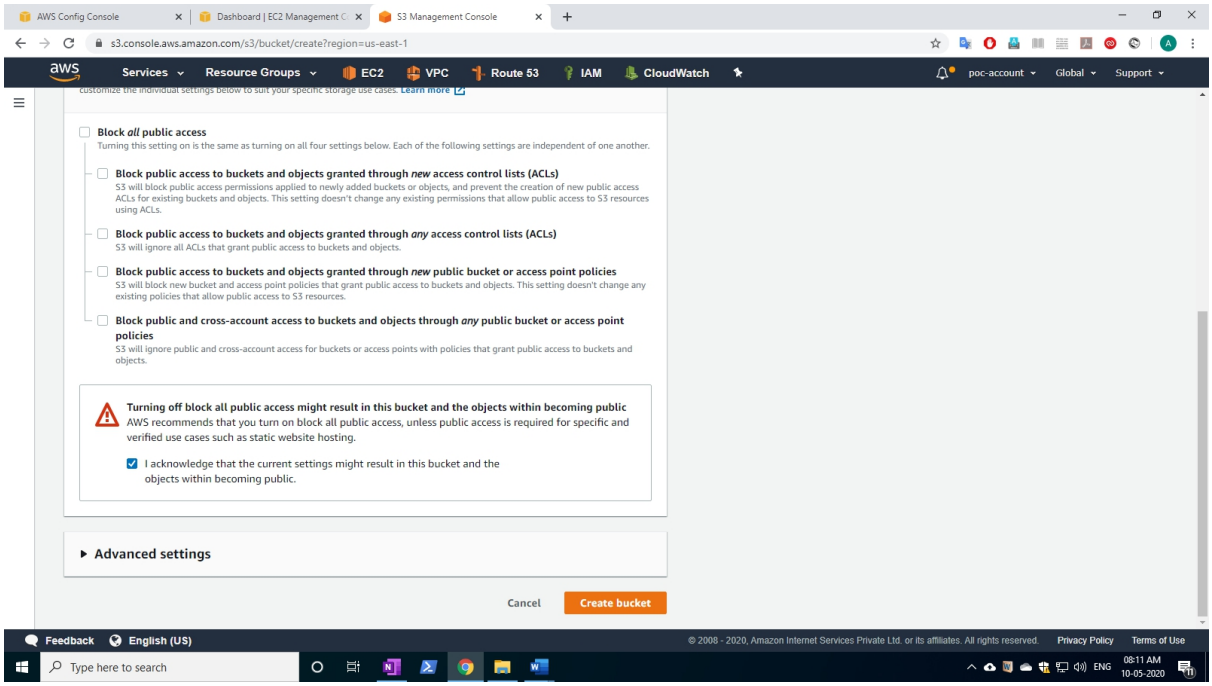


- Navigate *Settings -> Resource types to record -> All resources*
- Check *include Global resources,* and then click *Save*

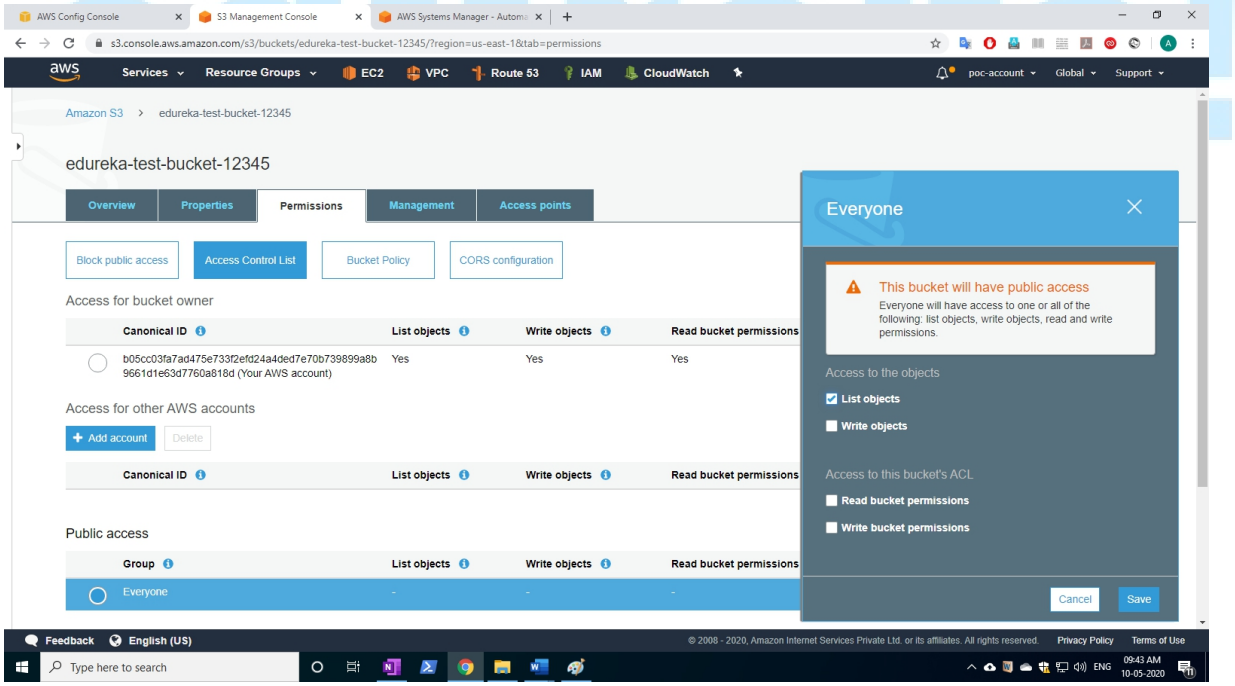


3. Create a S3 bucket with public read/write permissions
    - Navigate *S3 -> Create bucket*
    - *Bucket Name* -> any unique name as bucket names are unique across all AWS customers
    - *Region* -> us-east-1
    - Under *Bucket Settings for Public Access,* uncheck *Block all public access*
    - Click *I acknowledge* and *Create Bucket*



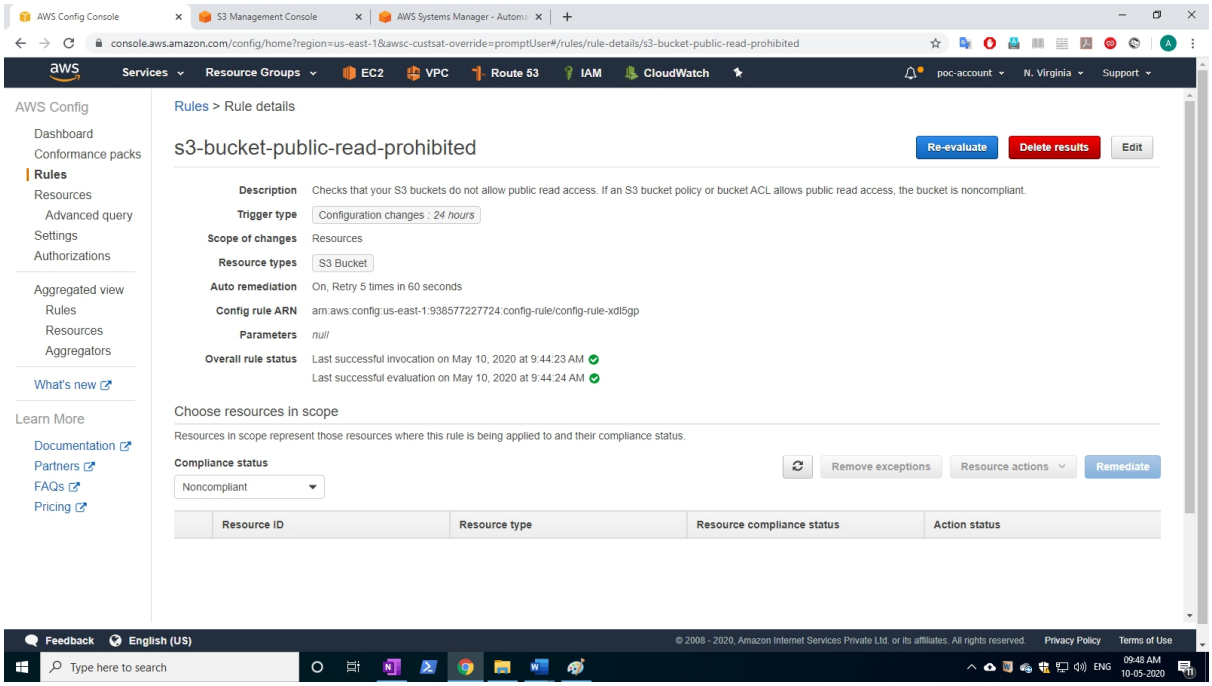


- Navigate to the newly created bucket -> Permissions -> Access Control List -> Public Access -> Check Everyone
- Check Access to the objects -> click List Objects -> Click Save

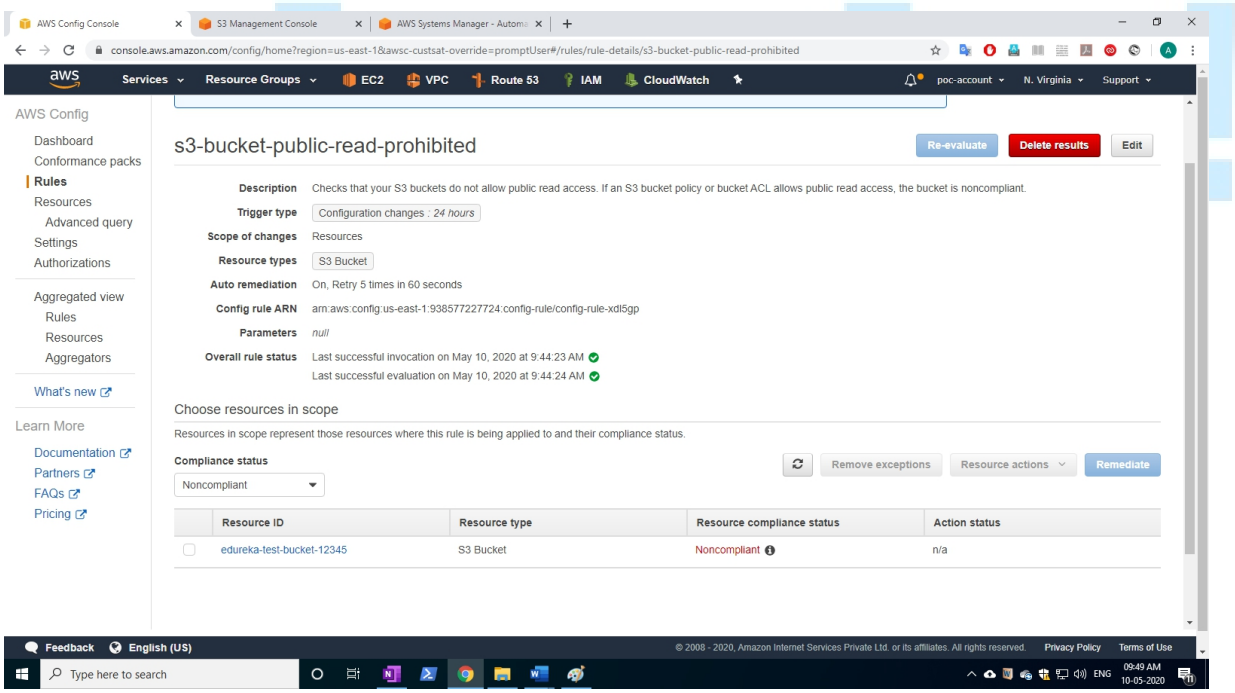


- Now the bucket is enabled with public read access

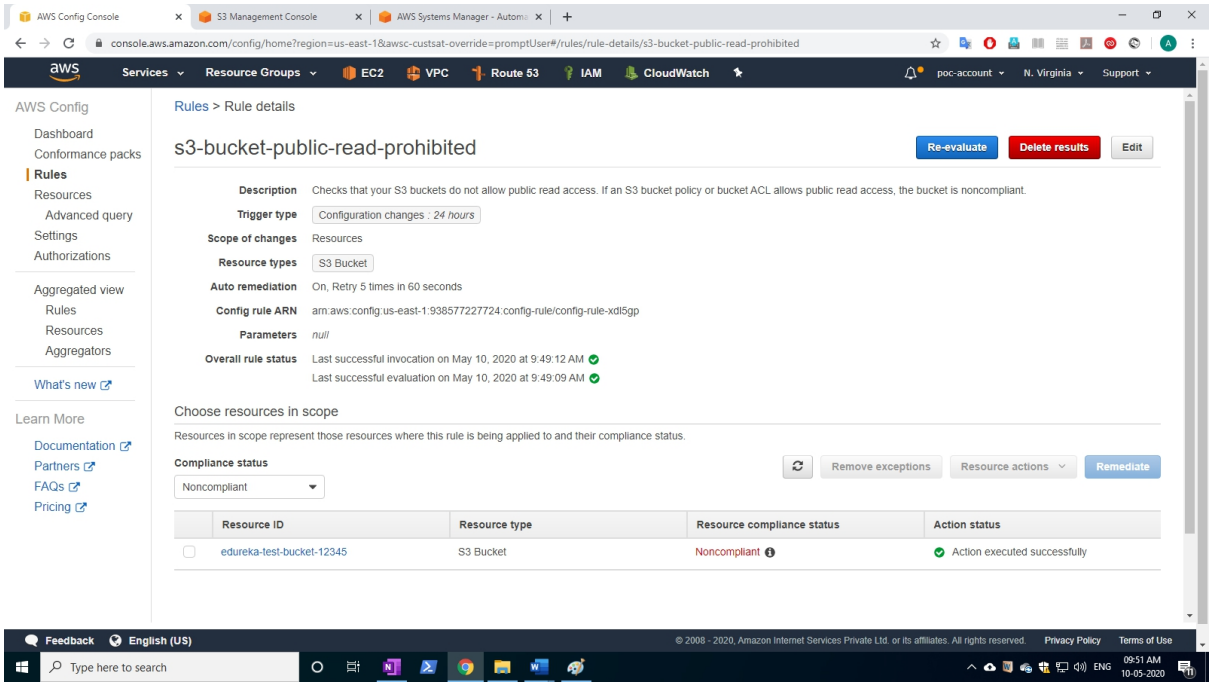
4. Verify the compliance and remediation action with AWS Config
    - Go back to AWS Config Dashboard. In the navigation pane, select Rules and select the S3 rule that you had created earlier.
    - Once the rule opens, click Re-evaluate



- Wait for a minute and then refresh. You should see the non-compliant resource as below.



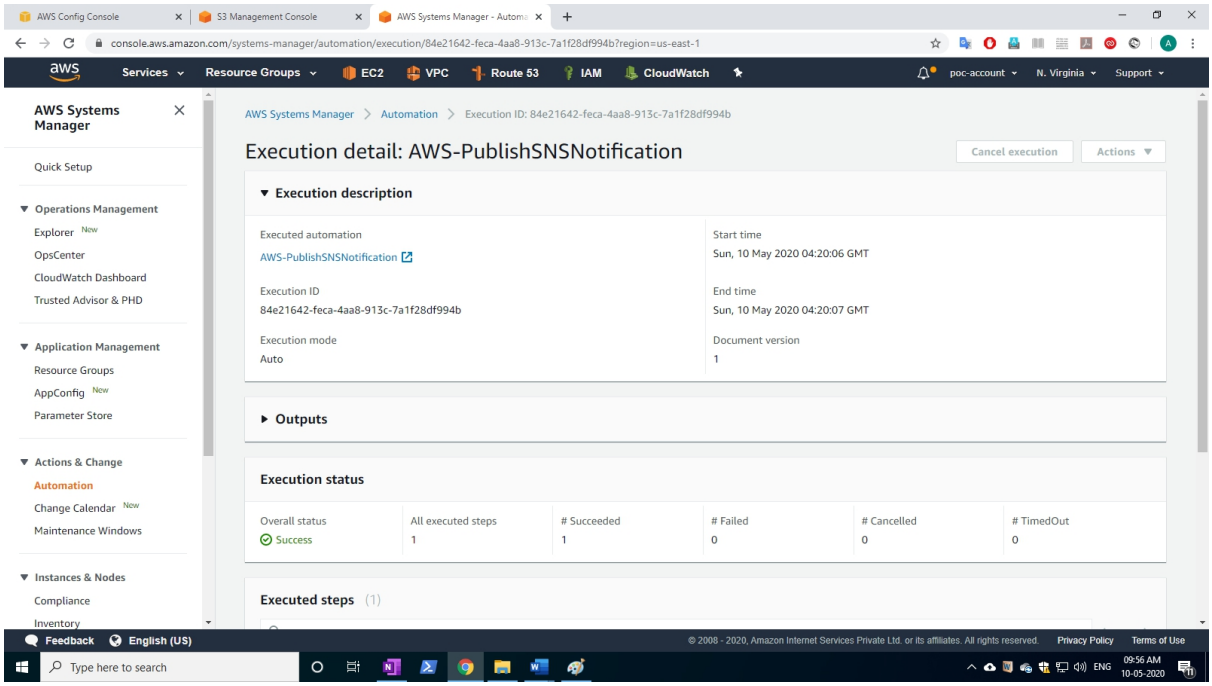
- Wait for another minute and notice Action Status changed to *Action executed successfully.*



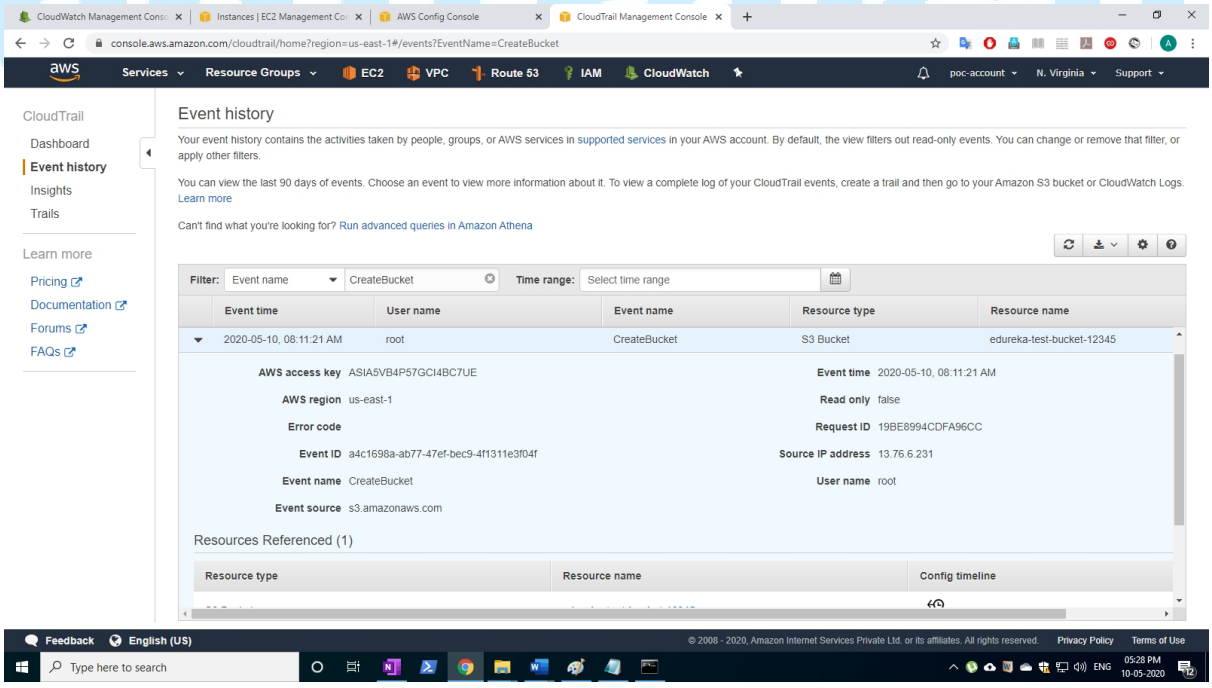
- It means the Remediation Action successfully executed and you should have received email notification as below in email mailbox.

- Navigate to *AWS Systems Manager -> Automation -> Execution ID*

- You should be able to see the details of *Remediation Automation* run by *Systems Manager* which was invoked by *AWS Config.*



5. Identify who caused non-compliance using CloudTrail
    - Navigate *CloudTrail* -> Event History -> Filter *Event Name: CreateBucket*
    - As you can see, the bucket in question was created by the user "root"



Clean Up

Delete the resources that you have created in the demo.

- This is the end of the lab.