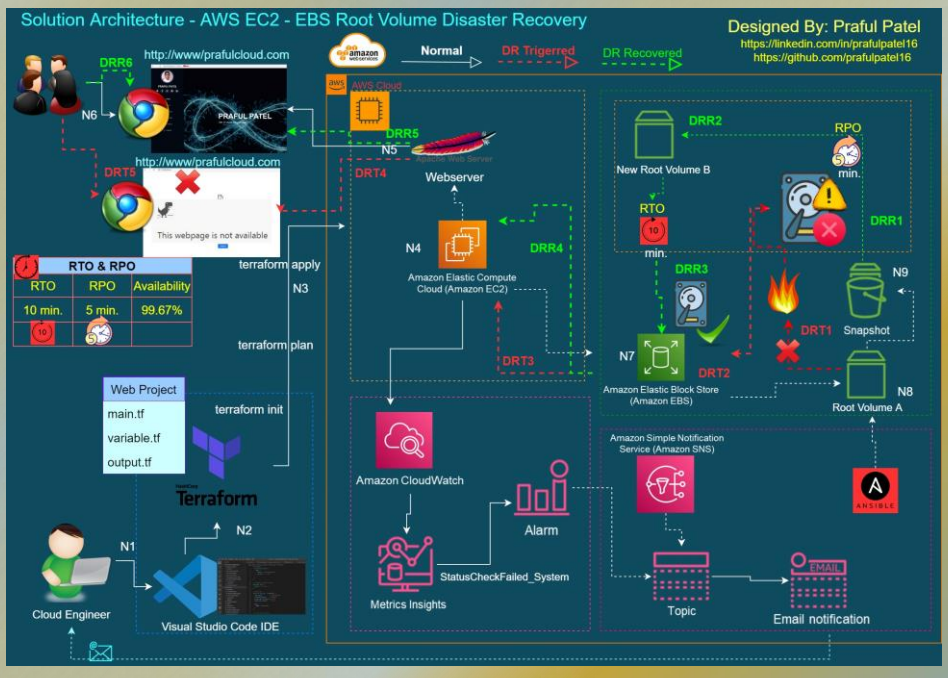
https://linkedin.com/in/prafulpatel16

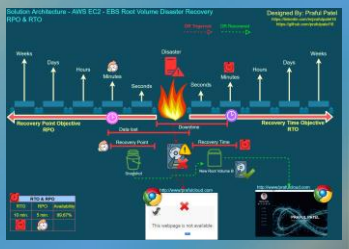
https://github.com/prafulpatel16

https://medium.com/@prafulpatel16

AWS PROJECT: AWS-EBS ROOT VOLUME DISASTER RECOVERY, MONITORING & NOTIFICATION

SOLUTION DESIGN & IMPLEMENTATION BY: PRAFUL PATEL





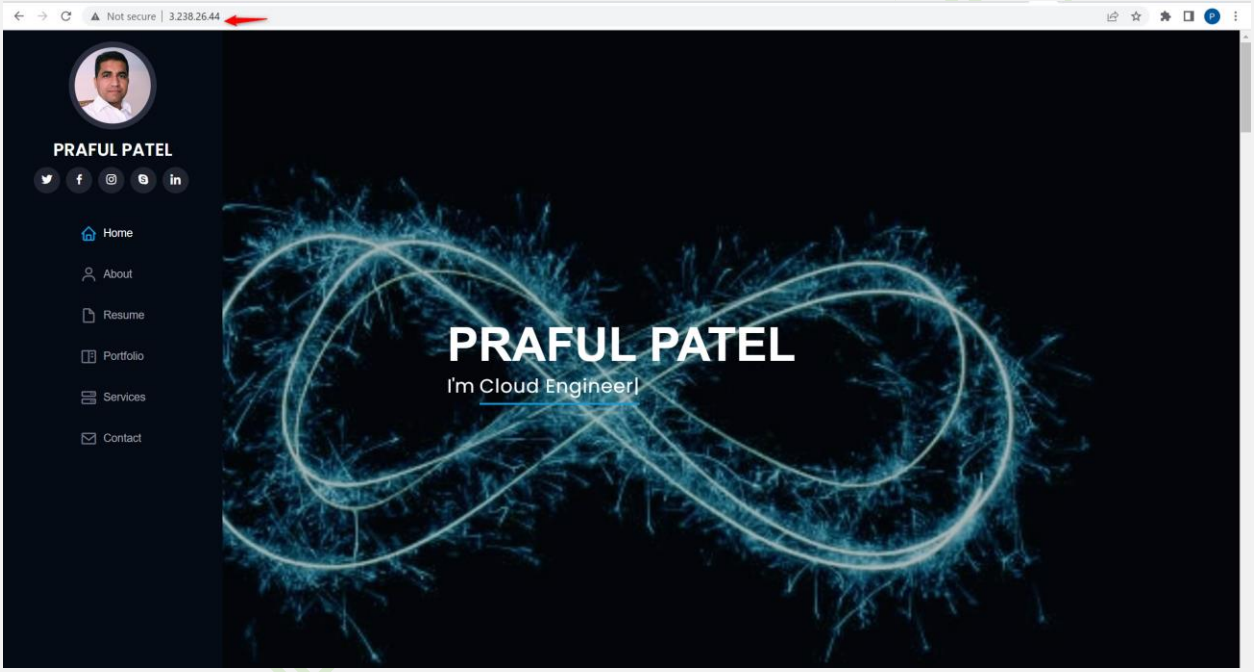
Date: June 21, 2022

- **Project:**

AWS - EBS ROOT VOLUME DISASTER RECOVERY, MONITORING & NOTIFICATION

- **Project Description:**

Application Name: Praful's webportfolio application



Cloud: AWS Cloud
Cloud Services: AWS EC2, AWS EBS, Volume, Snapshots, CloudWatch, SNS
WebServer: apache webserver

An IT services provider, PRAfect Systems Inc., is engaged in providing Cloud/DevOps & software development solutions. The company recently migrated its entire workload to the AWS Cloud. All the workload has been running on the EC2 virtual machine where application server is configured and web application is accessed through this server. They have configure the monitoring system with AWS Cloudwatch and integrated a SNS notification as well through which cloud engineer received a notification whenever there is a SystemCheck Failed for EC2 machine.

One morning cloud engineer received a System failure notification in to email, it was about EC2 machine root volume got corrupted due to some wrong system patching and hence it got system check instance failed via monitoring system.

RTO & RPO Requirements:

In order to maintain a business continuity the requirement is to maintain an RPO & RTO ratio is most critical and essential during the disaster condition.
RTO = 10 min. must meet the downtime and acceptable in order to recover the system from failure.
RPO = 05 min. must meet and system should be able to get back and recover the backup within the last 05 min.

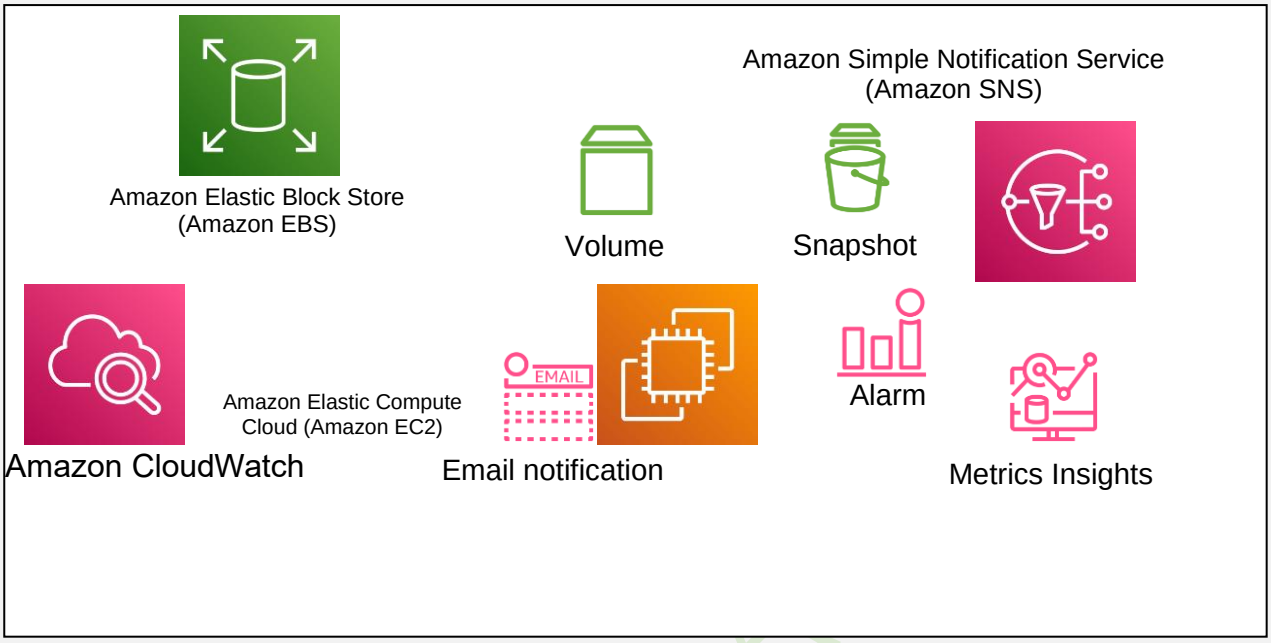
This project demonstrates an experience of designing and implementing of Disaster recovery scenario which can fulfil the defined business RPO & RTO requirements, along with cloudwatch monitoring and SNS notification system.

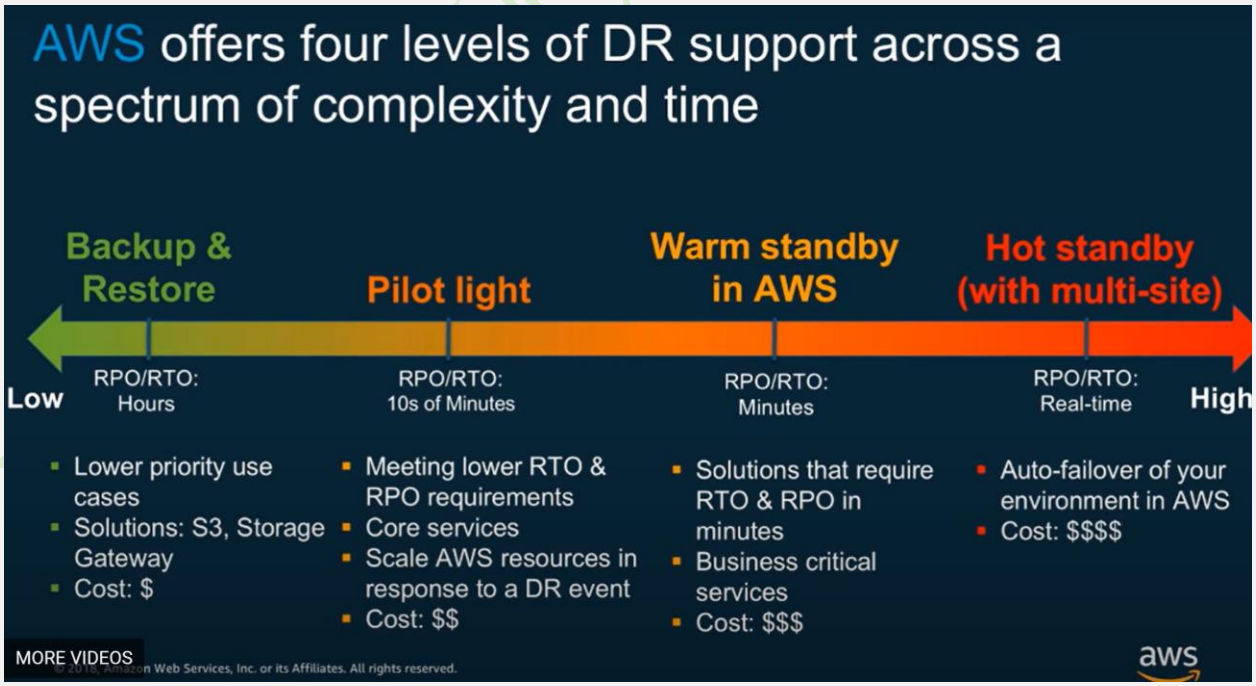
➢ Project Cost Estimation:

(Note: This cost is Not any actual cost, it's just an estimation based on high level requirement. Price may be vary based on adding and removing services based on requirement.)

➢ Tools & Technologies covered:

- AWS Cloud
- AWS Identity & Access Management (IAM)
- AWS EC2 Machine
- AWS Cloudwatch
- AWS SNS
- Terraform (Automated Cloud Provisioning Tool)
- Ansible
- Visual studio code IDE
- GitHub
- GitBash
- Draw.io





Resilience in Amazon EC2
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/disaster-recovery-resiliency.html

https://disaster-recovery.workshop.aws/en/intro/disaster-recovery.html

In addition to the AWS global infrastructure, Amazon EC2 offers the following features to support your data resiliency:

1. Copying AMIs across Regions
2. Copying EBS snapshots across Regions
3. Automating EBS-backed AMIs using Amazon Data Lifecycle Manager
4. Automating EBS snapshots using Amazon Data Lifecycle Manager
5. Maintaining the health and availability of your fleet using Amazon EC2 Auto Scaling
6. Distributing incoming traffic across multiple instances in a single Availability Zone or multiple Availability Zones using Elastic Load Balancing

- Instance status checks

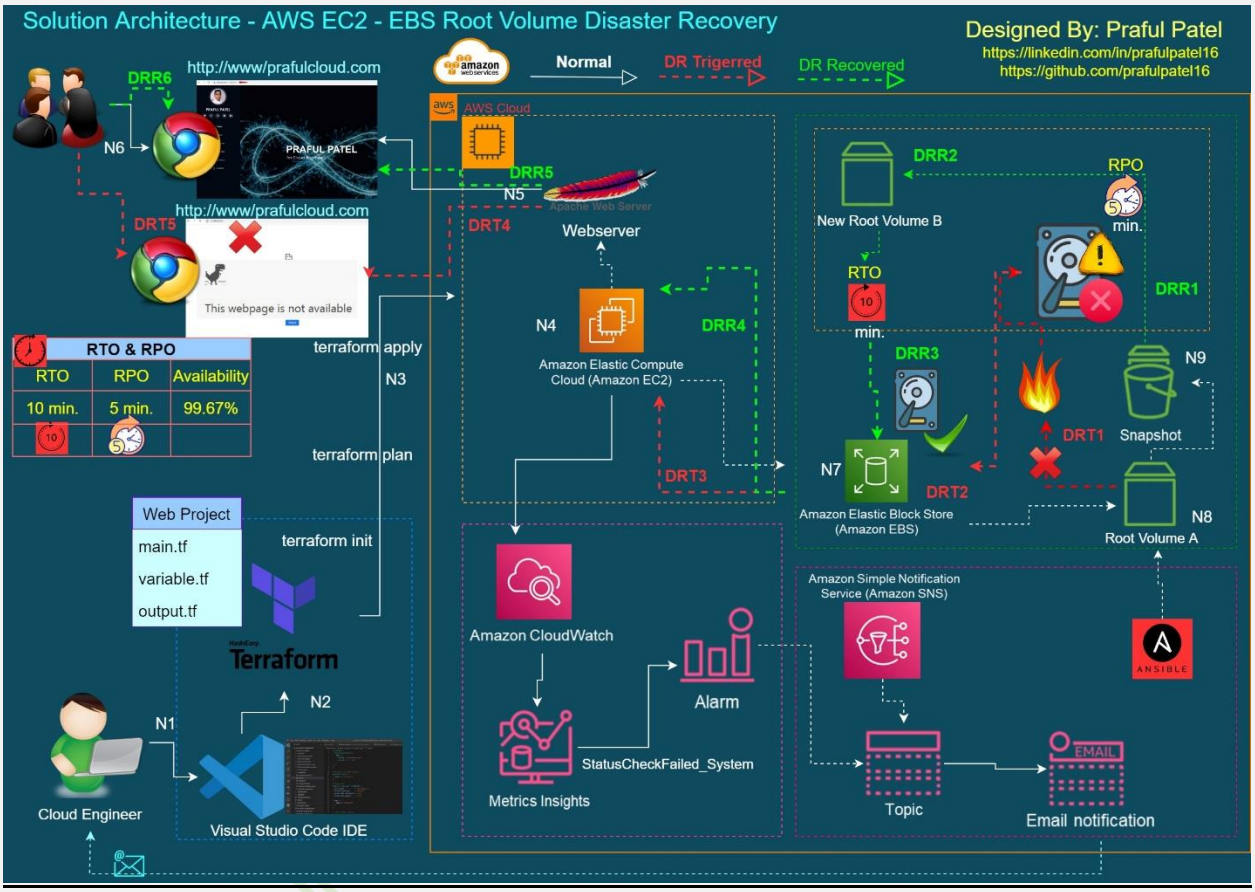
  Instance status checks monitor the software and network configuration of your individual instance. Amazon EC2 checks the health of the instance by sending an address resolution protocol (ARP) request to the network interface (NIC). These checks detect problems that require your involvement to repair. When an instance status check fails, you typically must address the problem yourself (for example, by rebooting the instance or by making instance configuration changes).

  The following are examples of problems that can cause instance status checks to fail:

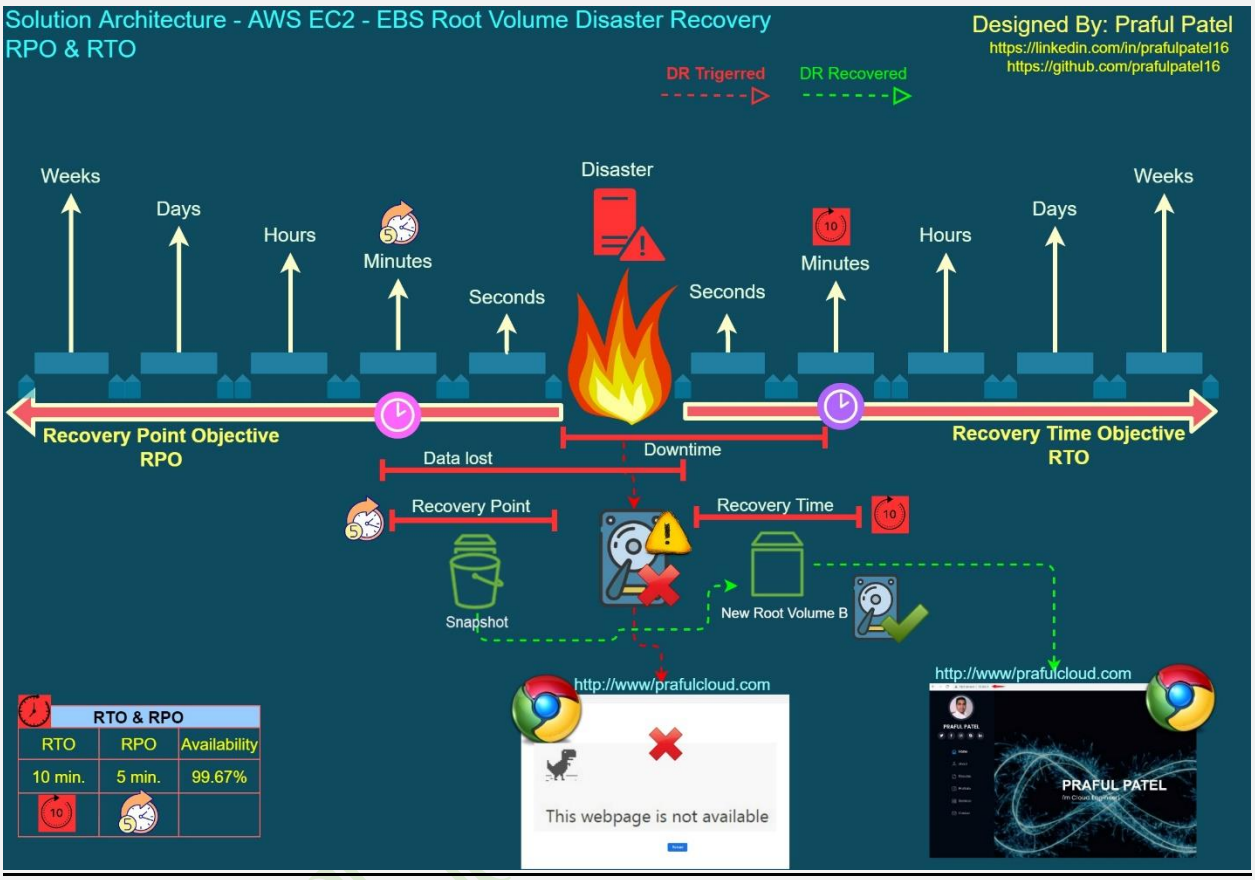
  Failed system status checks
  Incorrect networking or startup configuration
  Exhausted memory
  Corrupted file system
  Incompatible kernel

- **Solution Architecture:**

AWS EBS Root Volume Disaster Recovery by RPO & RTO Architecture



RPO & RTO Architecture



This project will be completed in following implementation phases.

- **Project implementation Phase:**

  - Phase 1: Deploy EC2 machine using terraform automation
    - Write terraform script to launch EC2 machine
    - Write main.tf, variable.tf and output.tf
    - Prepare user data webserver and application source code packages in shell script
    - Add user data file within the terraform configuration
    - Verify that web application is successfully accessed from web browser
  - Phase 2: Take a snapshot of root volume manual way.
    - Go to snapshots and take snapshot of existing root volume A
    - Verify that snapshot process is complete

  OR

  - Phase 2.1: Take a snapshot of root volume Ansible automated way.
    - Go to VS code IDE
    - Gather the instance and volume information manually
    - Write snapshot yaml file

- Run Ansible playbook file
- Verify from the AWS console that snapshot has been created.

- Phase 3: Configure Cloudwatch monitoring & SNS topic for failure notification
  - Go to Cloudwatch and create an Alarm
  - Select an EC2 metric: StatusFailedCheck_System
  - Create a new SNS Topic and provide an email address
  - Complete the cloud watch process
  - Go to SNS Topic and confirm the subscription by verifying the link

- Phase 4: Simulate and Trigger a Disaster recovery scenario.
  - Prepare manual system failure script
  - Write a script to remove an application files from the apache root directory /var/www/html/
  - Run the script from EC2 machine.
  - Verify that all application files removed
  - Verify that web application is not accessible.

- Phase 5: Simulate Cloudwatch monitoring 'In-Alarm"
  - Login to EC2 machine.
  - Become a root user
  - Configure aws configure
  - Run the cloudwatch set-alarm script to put into "In-Alarm" status
  - Go to Cloudwatch and verify that status is turned from "OK" to "In-Alarm"
  - Go to email and verify that email is received with necessary information.

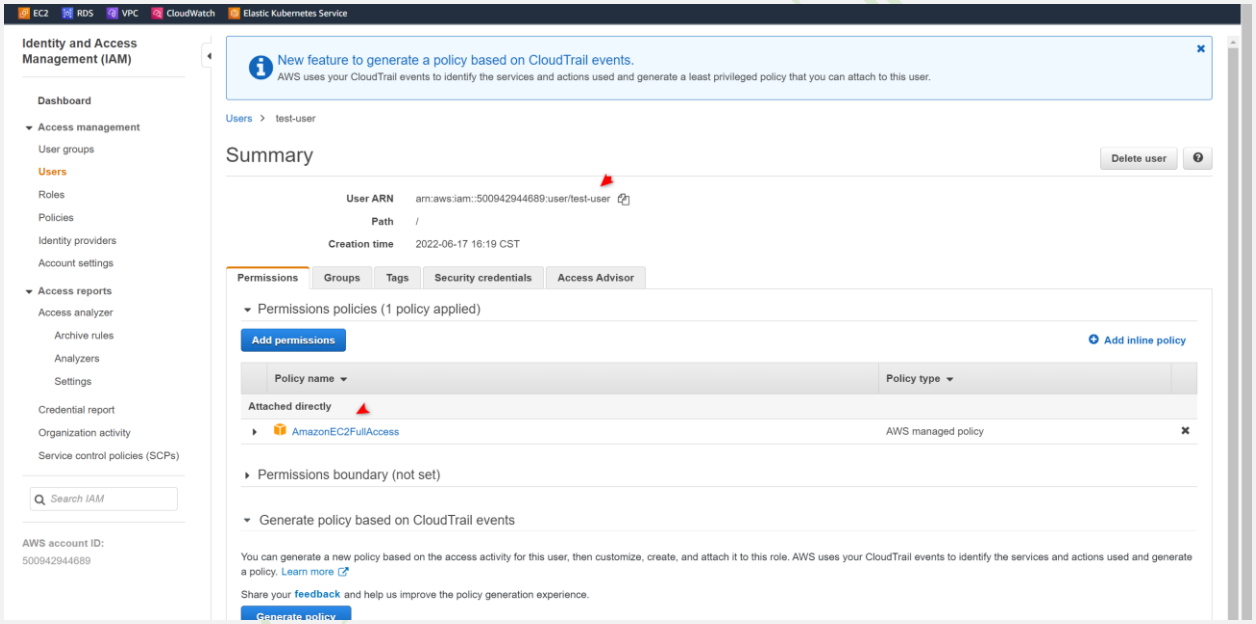
- Phase 6: Recover from Disaster condition (RPO 5 min. RTO 10 min.)
  - Go to EC2 machine
  - Go to Action – make sure that ec2 machine is running.
  - Select an option "Monitor and troubleshoot"
  - Select an option "Replace root volume"
  - Select a recent snapshot taken within last 5 minutes to meet the RPO condition
  - Attach complete the snapshot
  - Go to Volume and verify that the new snaphost volume is attached and " In-Use" status
  - Verify that old root volume is "Available" status which is no use and corrupted now.
  - Verify that web application is now accessible

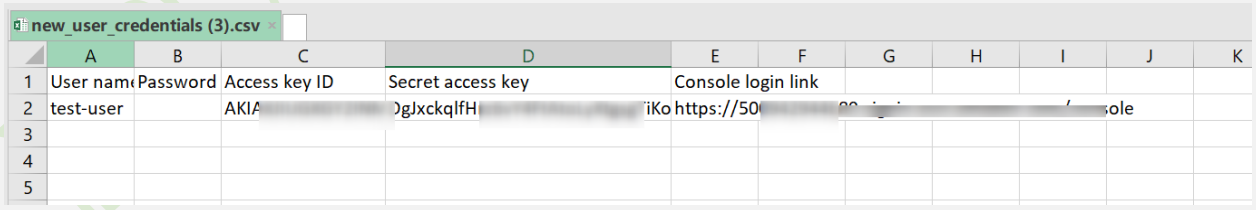
- **Pre-Requisite:**

- VS Code installed and configured in windows
- Terraform installed and configured in VS code
- AWS IAM user account with “AWSEC2FullAccess” permission
- User-data script ready for webapp source code
- bash script for application removal

AWS IAM user account with “AWSEC2FullAccess” permission

Create a New IAM user with EC2FullAccess permission with programmatic access





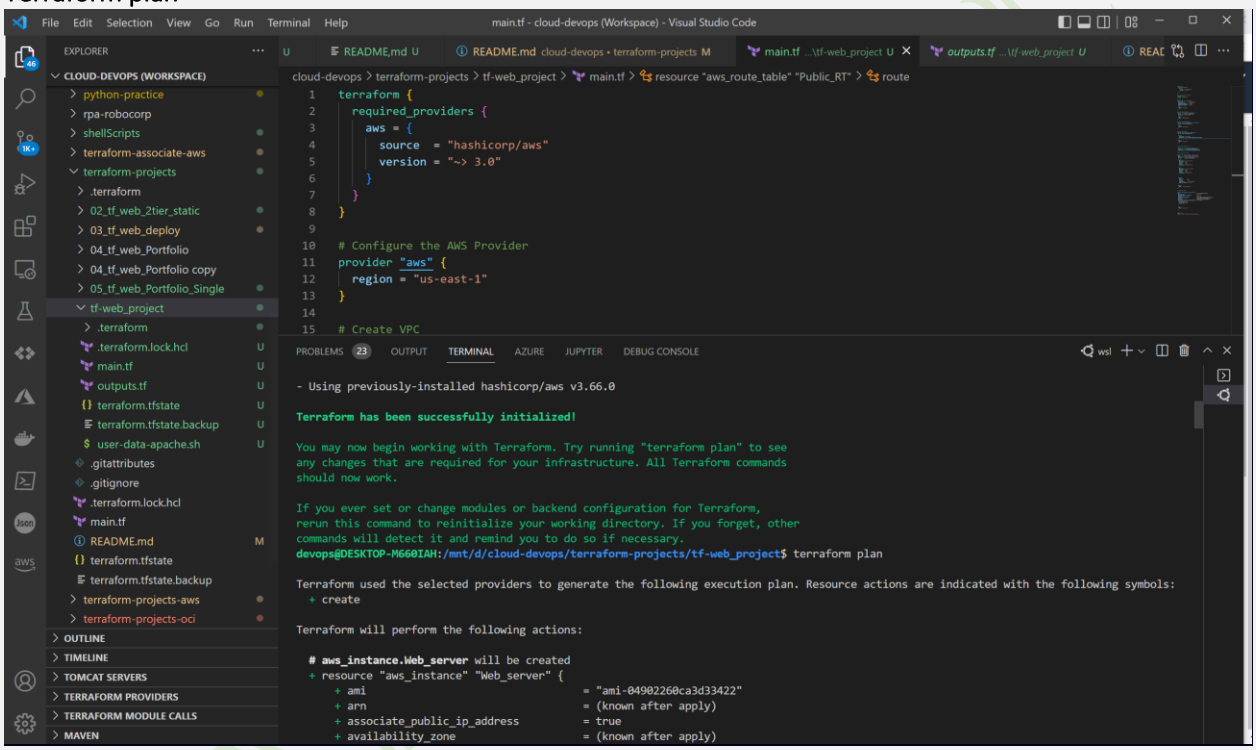
➢ Implementation in an Action:

- Phase 1: Deploy EC2 machine using terraform automation
  - Write terraform script to launch EC2 machine
  - Write main.tf, variable.tf and output.tf

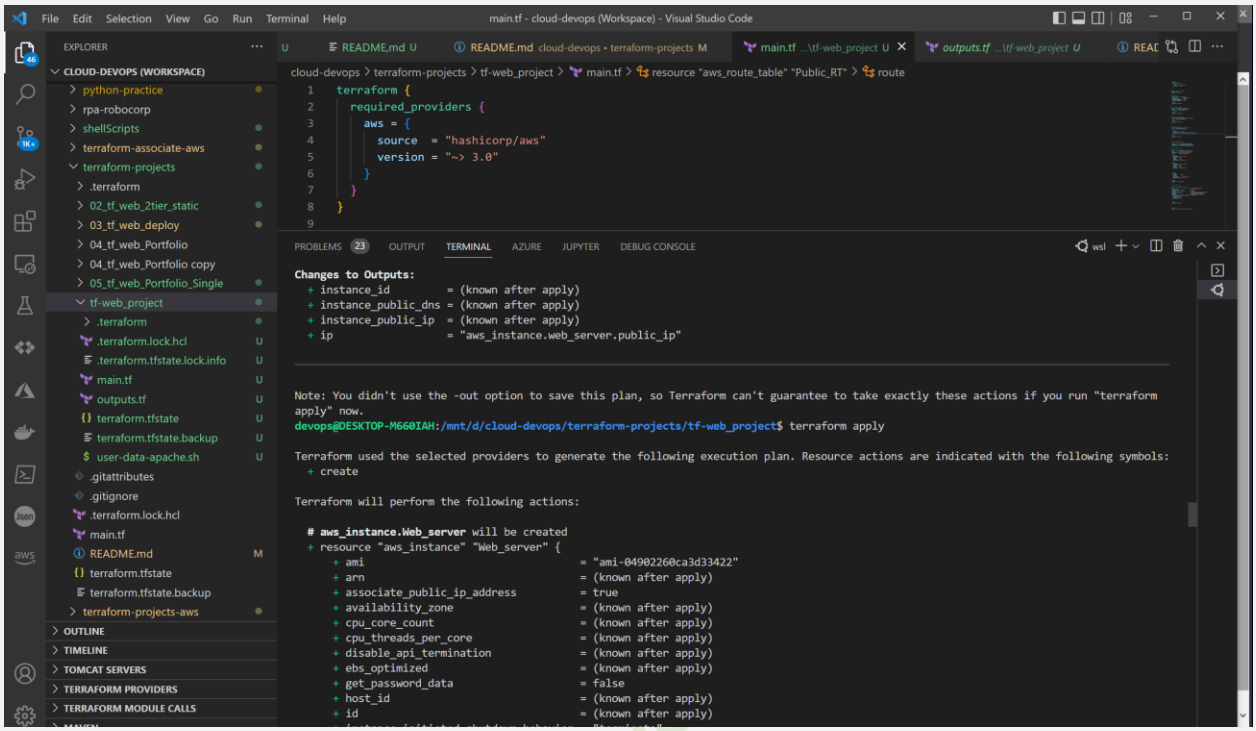
- Prepare user data webserver and application source code packages in shell script
- Add user data file within the terraform configuration
- Verify that web application is successfully accessed from web browser

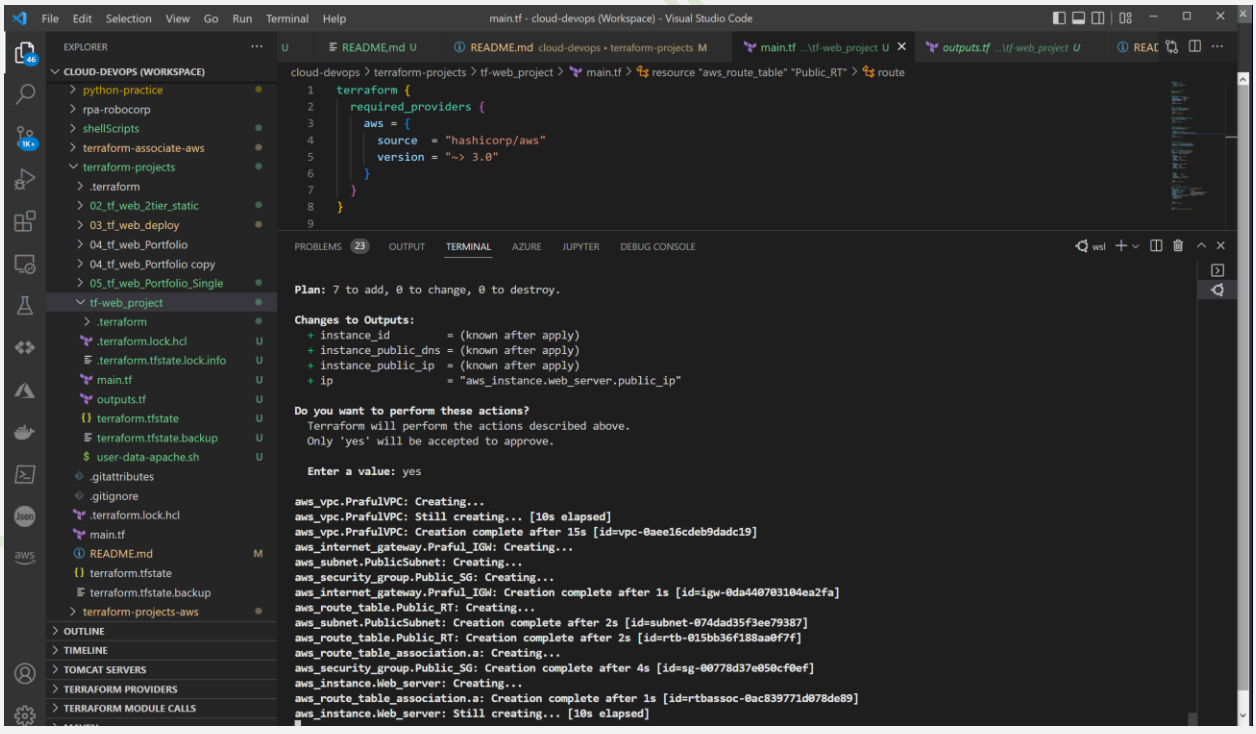
Terraform init

Terraform plan

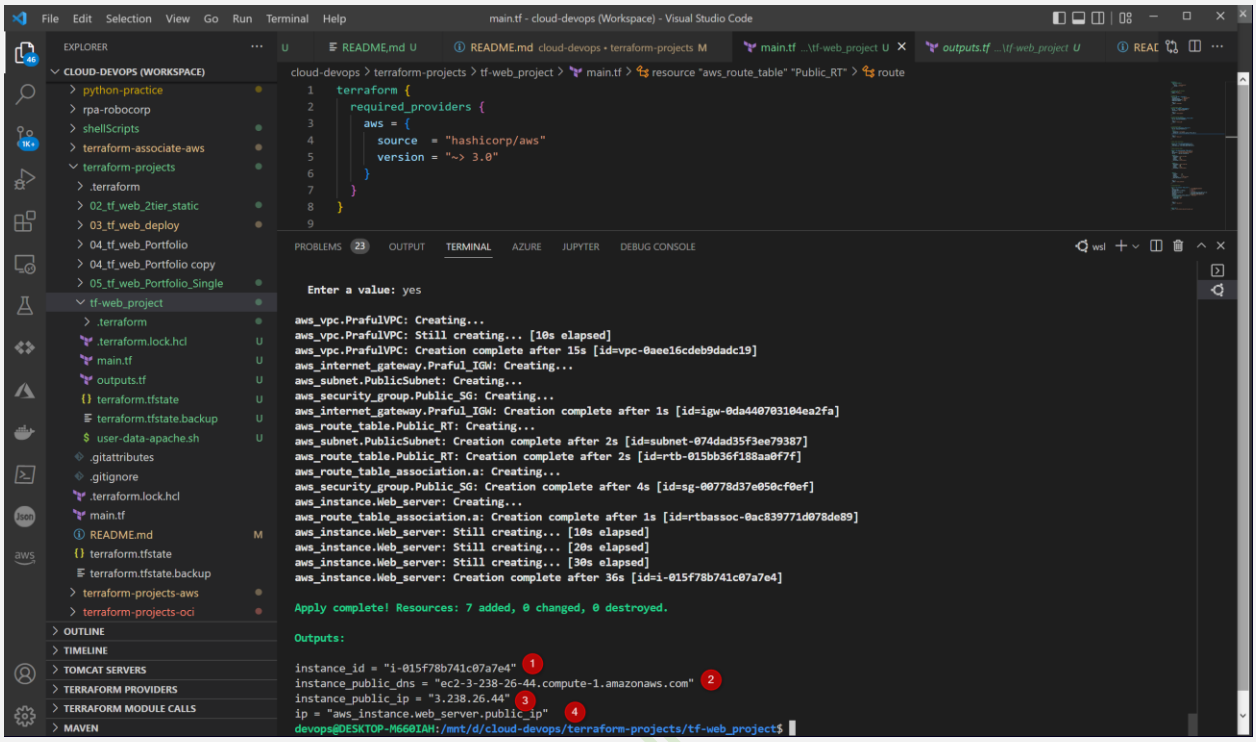


Terraform apply

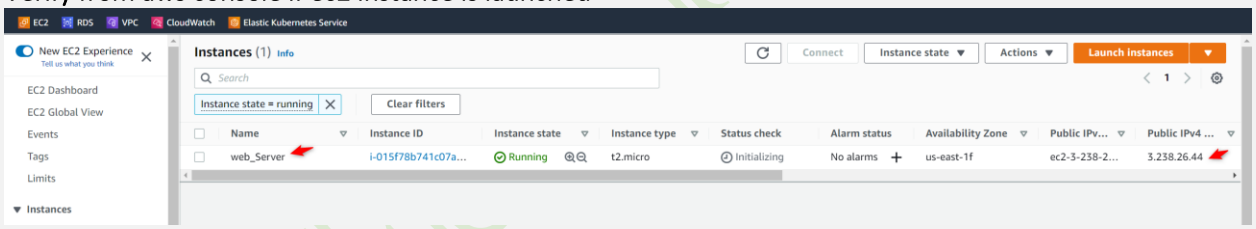




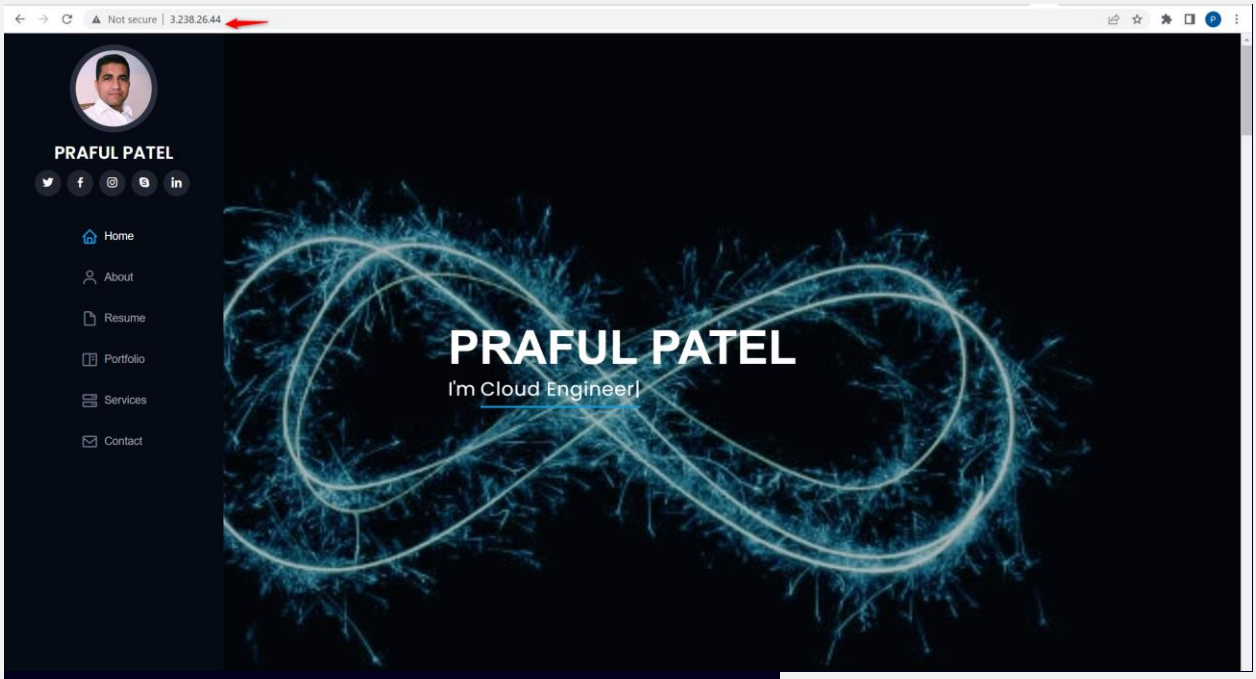
Apply complete



Verify from aws console if ec2 instance is launched



Verify that web application is accessible from browser



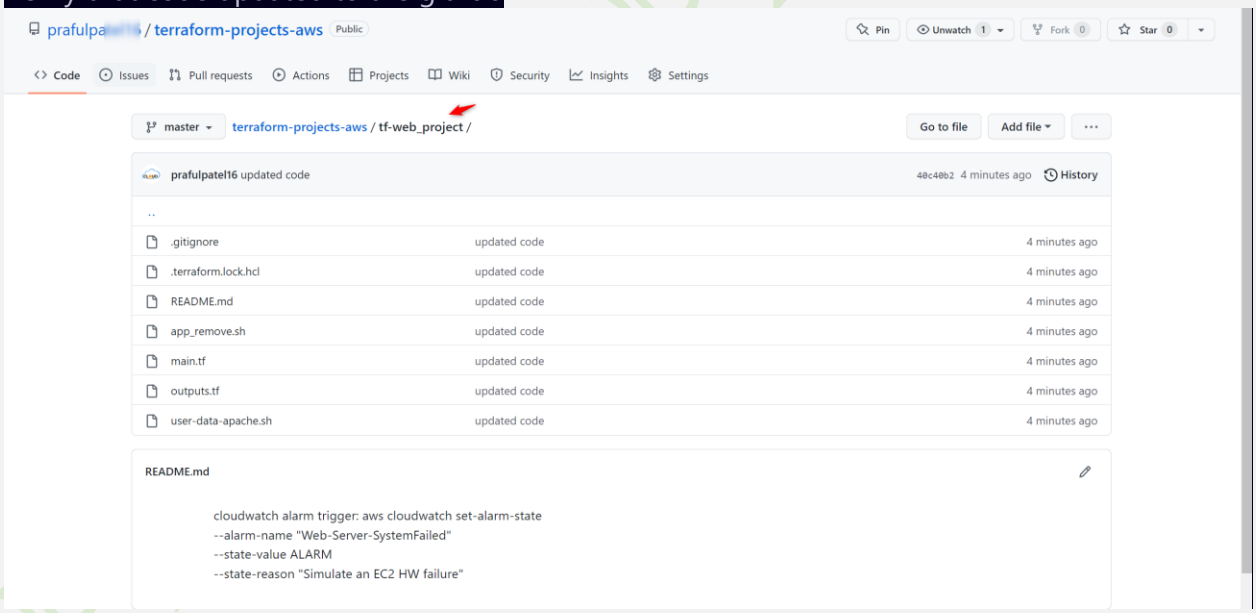
https://github.com/prafulpatel16/terraform-projects-aws.git

Push source code to github

```
MINGW64:/d/cloud-devops/terraform-projects-aws/tf-web_project

Praful@DESKTOP-M660IAH MINGW64 /d/cloud-devops/terraform-projects-aws/tf-web_project (master)
$ git add .
warning: LF will be replaced by CRLF in tf-web_project/.gitignore.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in tf-web_project/.terraform.lock.hcl.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in tf-web_project/app_remove.sh.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in tf-web_project/user-data-apache.sh.
The file will have its original line endings in your working directory

Praful@DESKTOP-M660IAH MINGW64 /d/cloud-devops/terraform-projects-aws/tf-web_project (master)
$ git commit -m "updated code"
[master 40c40b2] updated code
 7 files changed, 207 insertions(+)
 create mode 100644 tf-web_project/.gitignore
 create mode 100644 tf-web_project/.terraform.lock.hcl
 create mode 100644 tf-web_project/README.md
 create mode 100644 tf-web_project/app_remove.sh
 create mode 100644 tf-web_project/main.tf
 create mode 100644 tf-web_project/outputs.tf
 create mode 100644 tf-web_project/user-data-apache.sh
```

```
Praful@DESKTOP-M660IAH MINGW64 /d/cloud-devops/terraform-projects-aws/tf-web_project (master)
$ git push -u origin master
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 8 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (10/10), 2.90 KiB | 1.45 MiB/s, done.
Total 10 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/prafulpatel16/terraform-projects-aws.git
   f64fc6a..40c40b2  master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.

Praful@DESKTOP-M660IAH MINGW64 /d/cloud-devops/terraform-projects-aws/tf-web_project (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   ../01_tf_web_PrafulsPortfolio/outputs.tf
        modified:   ../02_tf_HighlyAvailable_vpc_aws/modules/vpc/outputs.tf
        modified:   ../02_tf_HighlyAvailable_vpc_aws/modules/vpc/variables.tf
        modified:   ../README.md
```

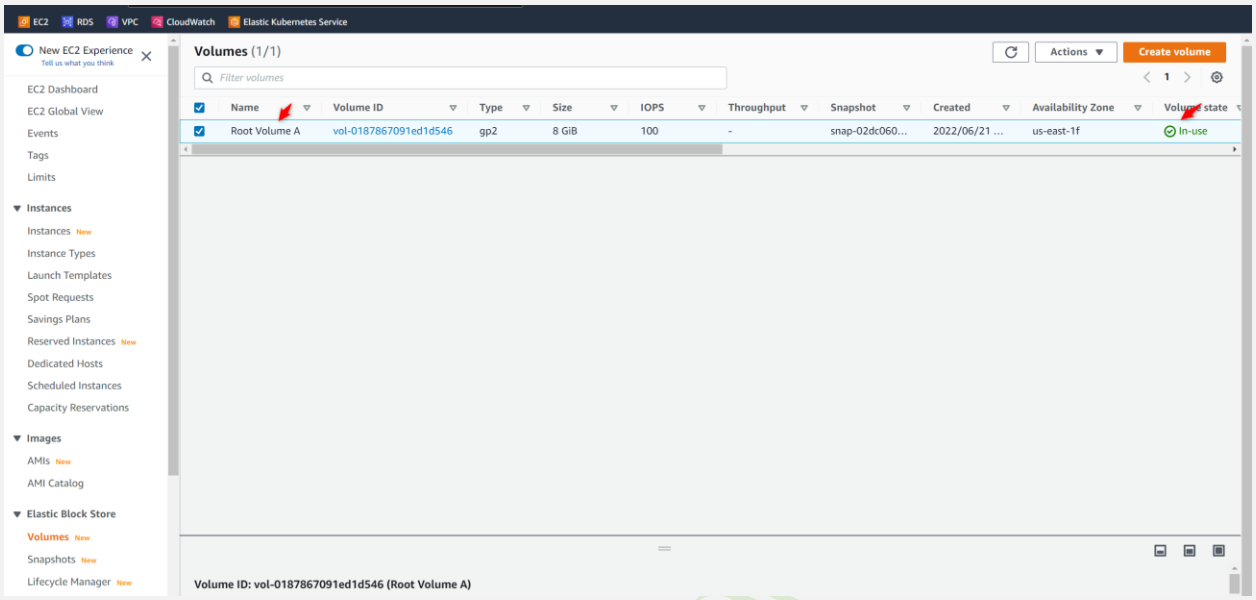
Verify that code updated to the github



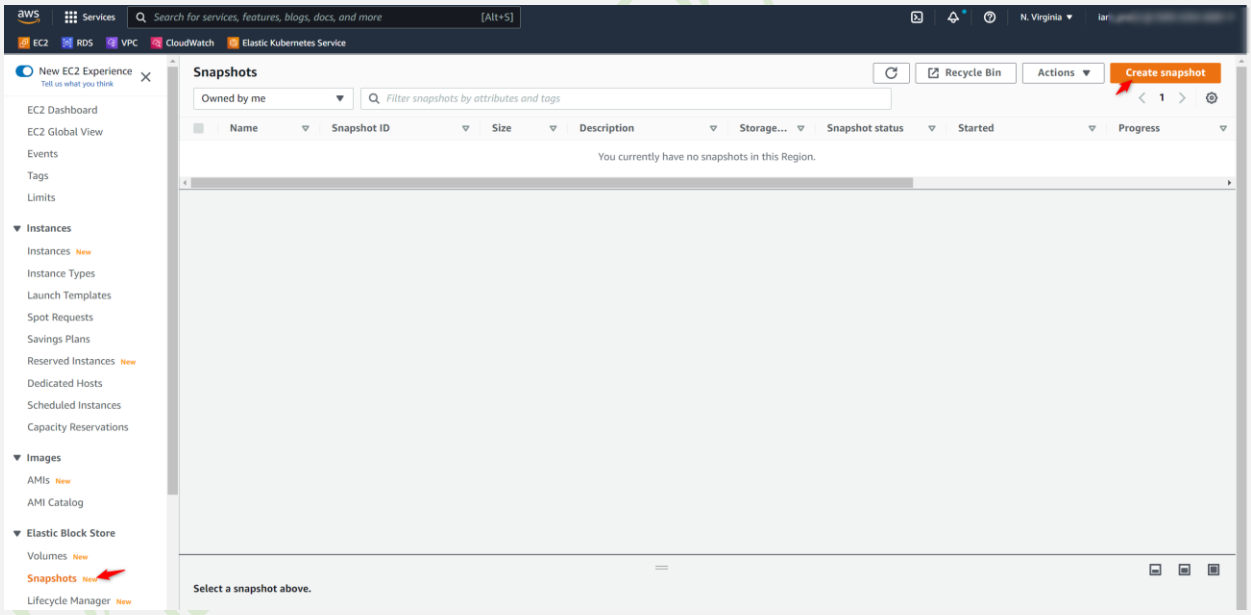
- Phase 2: Take a snapshot of root volume
  - Go to snapshots and take snapshot of existing root volume A
  - Verify that snapshot process is complete

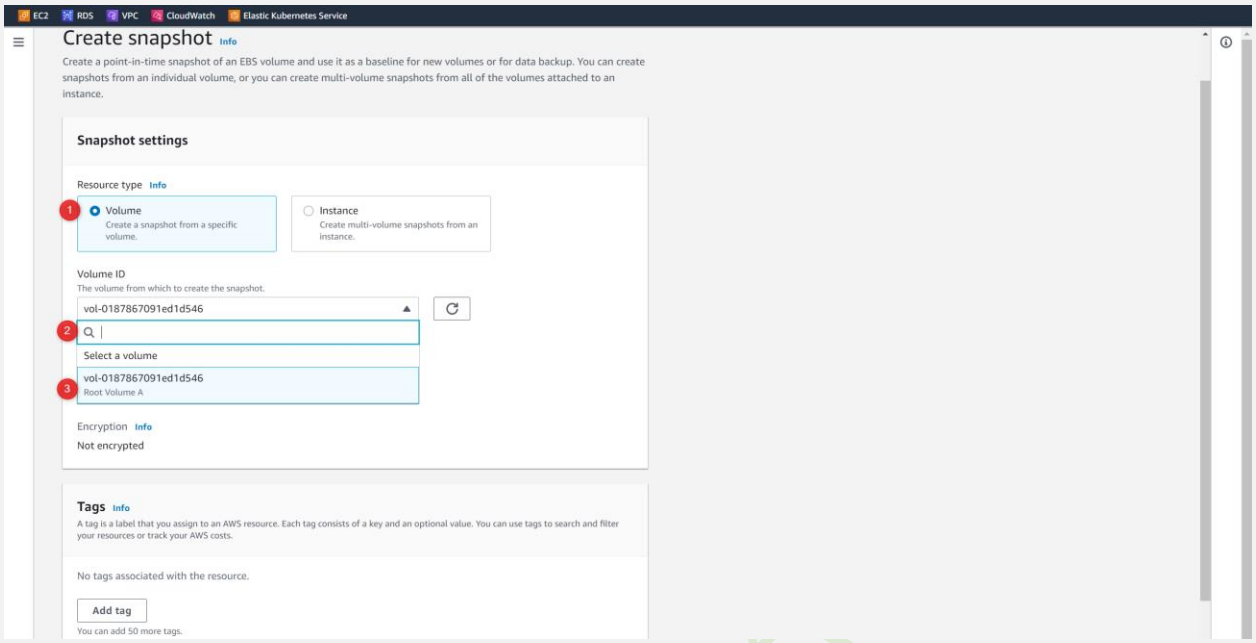
Volume

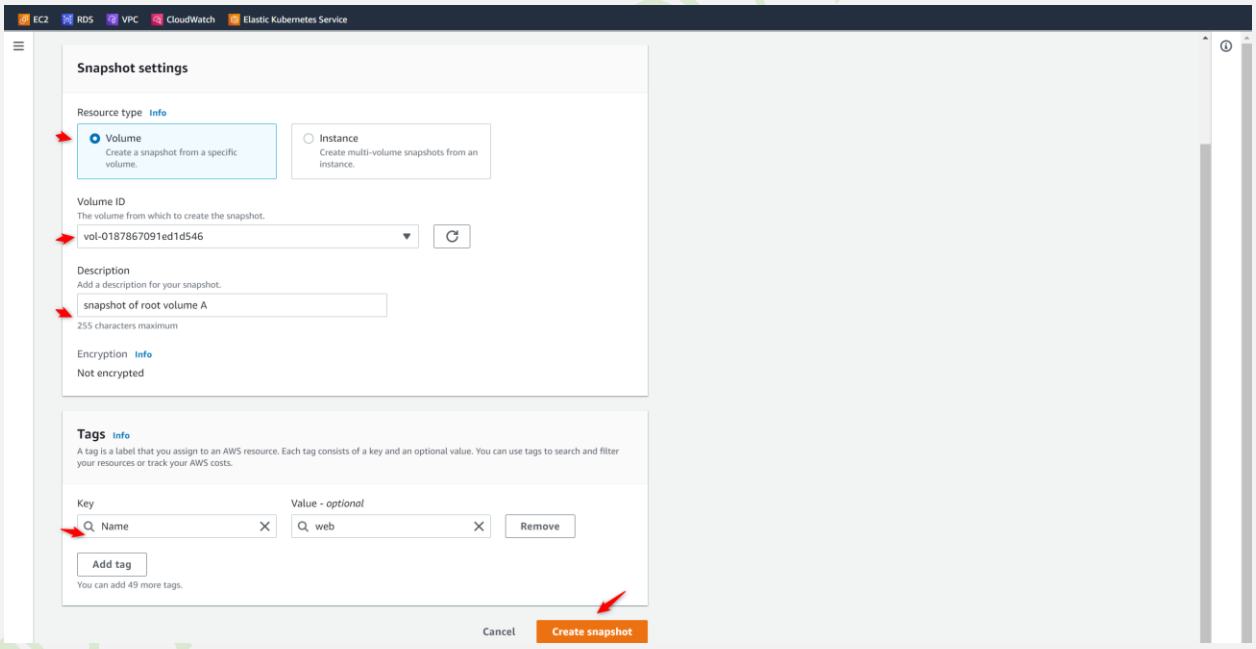
Go to Volume



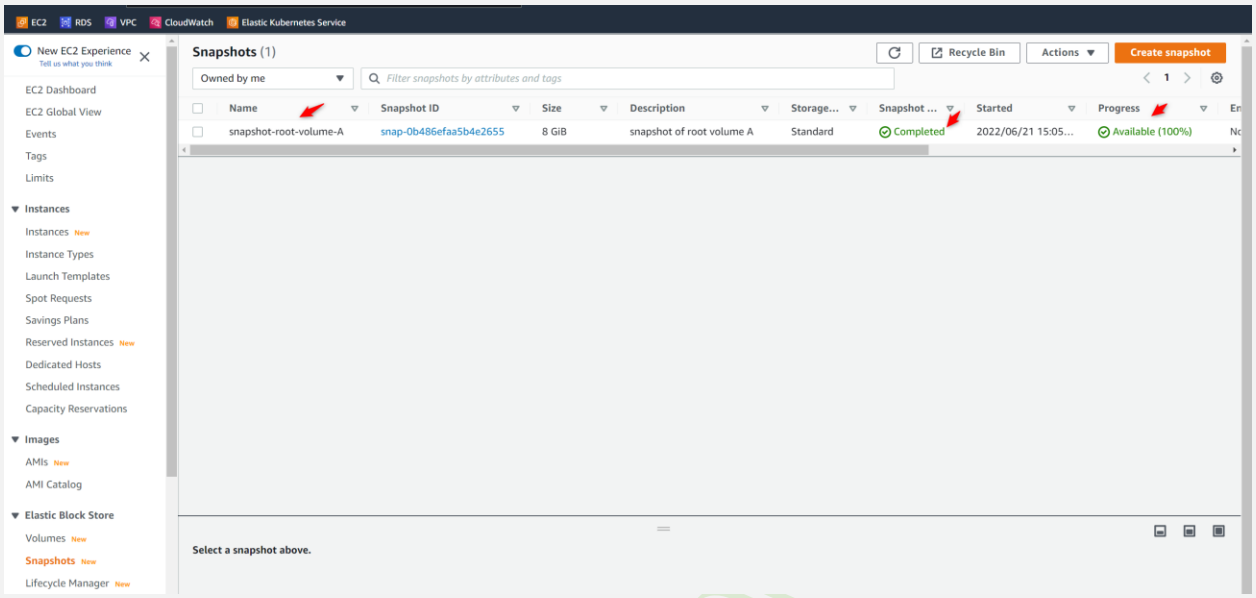
Take a snapshot







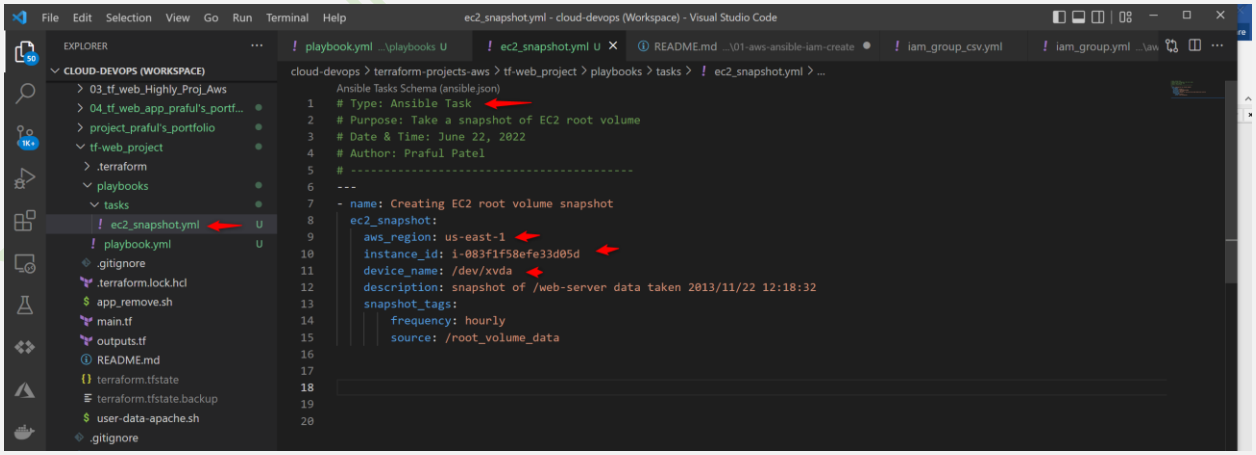
Snapshot complete



OR

- Phase 2.1: Take a snapshot of root volume Ansible automated way.
  - Go to VS code IDE
  - Gather the instance and volume information manually
  - Write snapshot yaml file
  - Run Ansible playbook file
  - Verify from the AWS console that snapshot has been created.

Go to VS code IDE



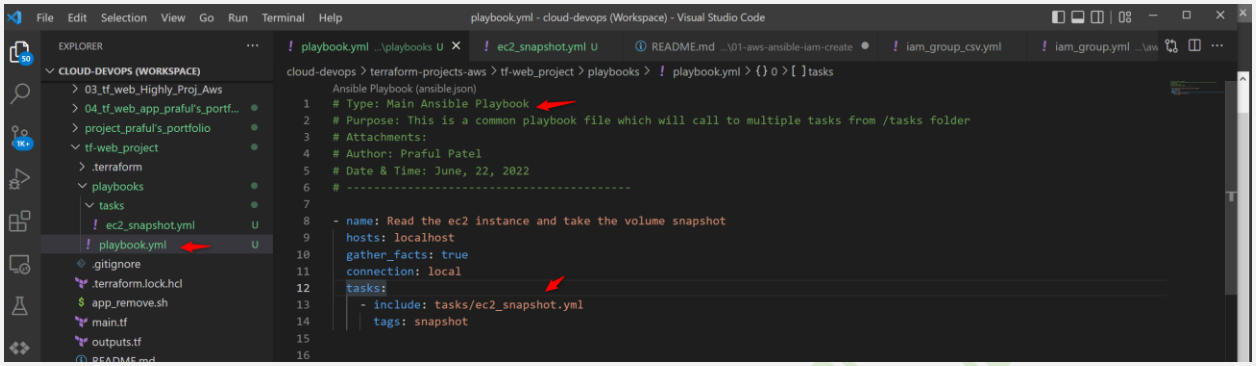
Gather the instance and volume information manually

aws_region:

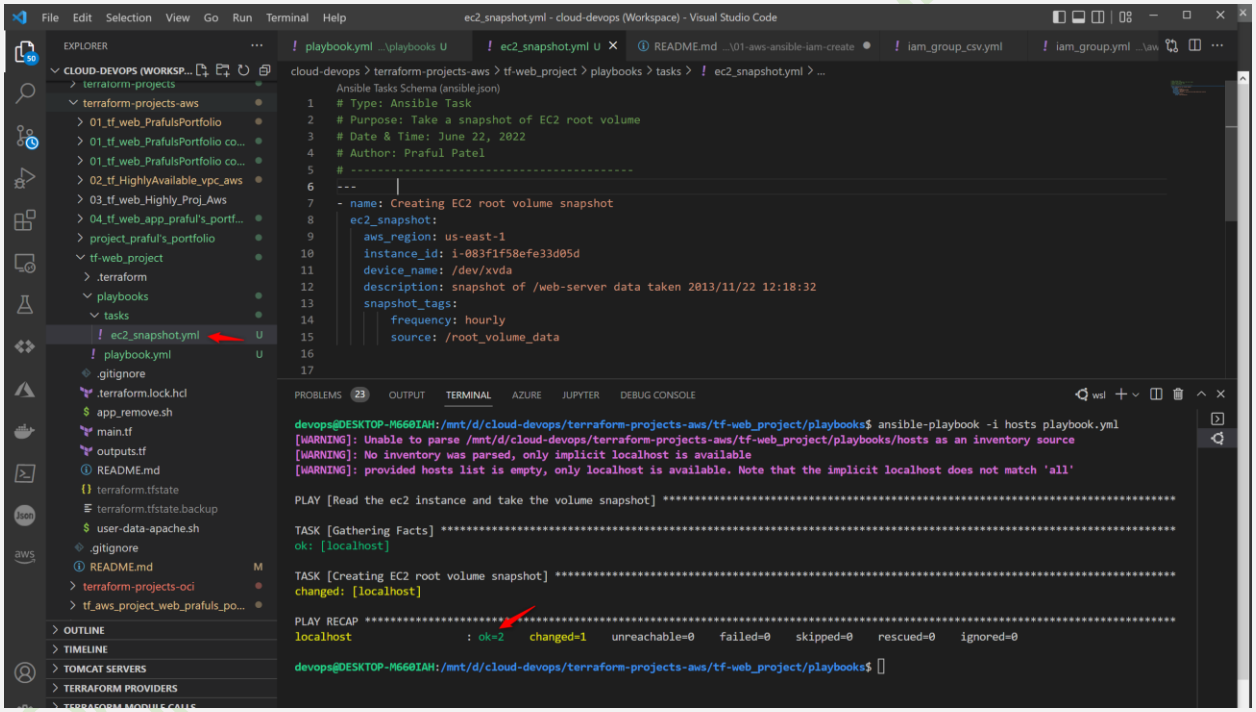
Instance id:

Device_name:

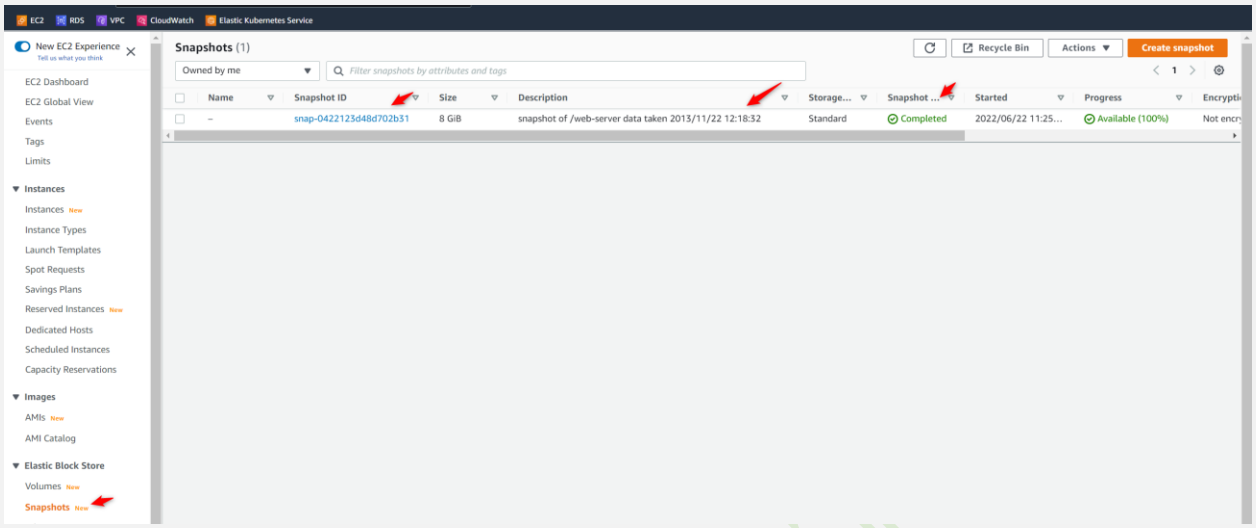
Write main snapshot yaml file



Run Ansible playbook file



Verify from the AWS console that snapshot has been created.



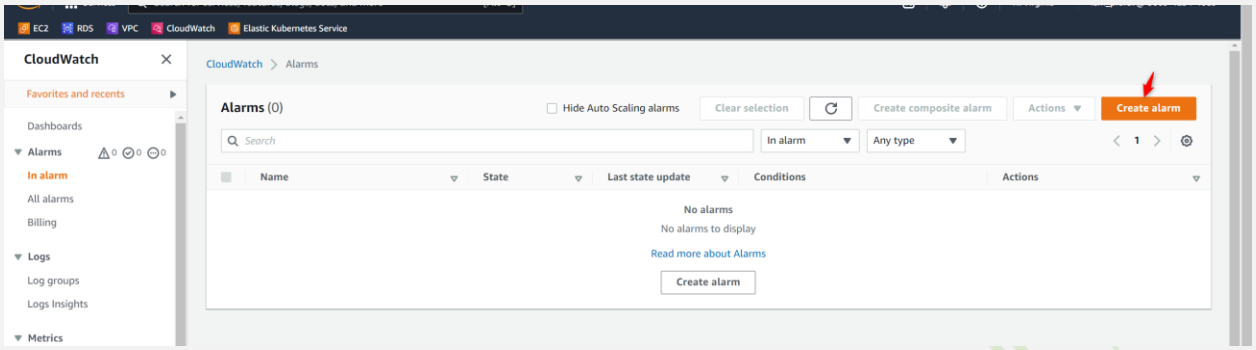
- Phase 3: Configure Cloudwatch monitoring & SNS topic for failure notification
  - Go to Cloudwatch and create an Alarm
  - Select an EC2 metric: StatusFailedCheck_System
  - Create a new SNS Topic and provide an email address
  - Complete the cloud watch process
  - Go to SNS Topic and confirm the subscription by verifying the link

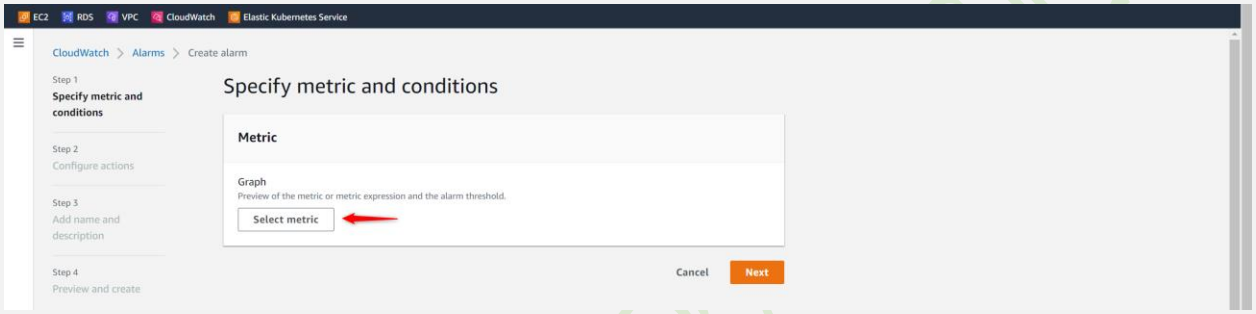
System Monitoring

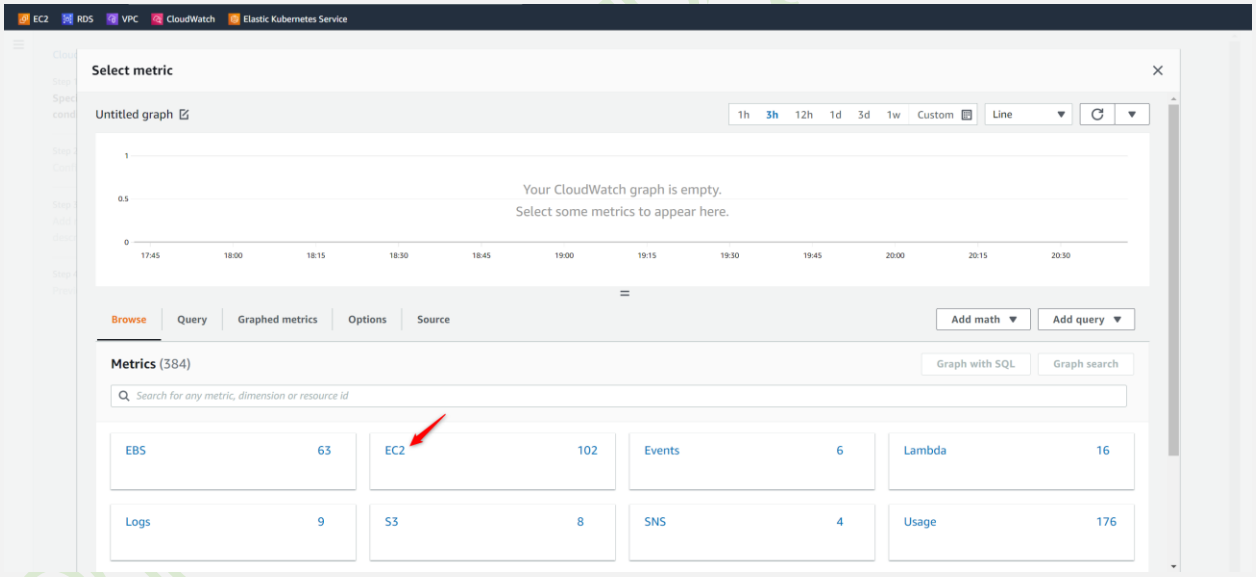
Configure Cloudwatch

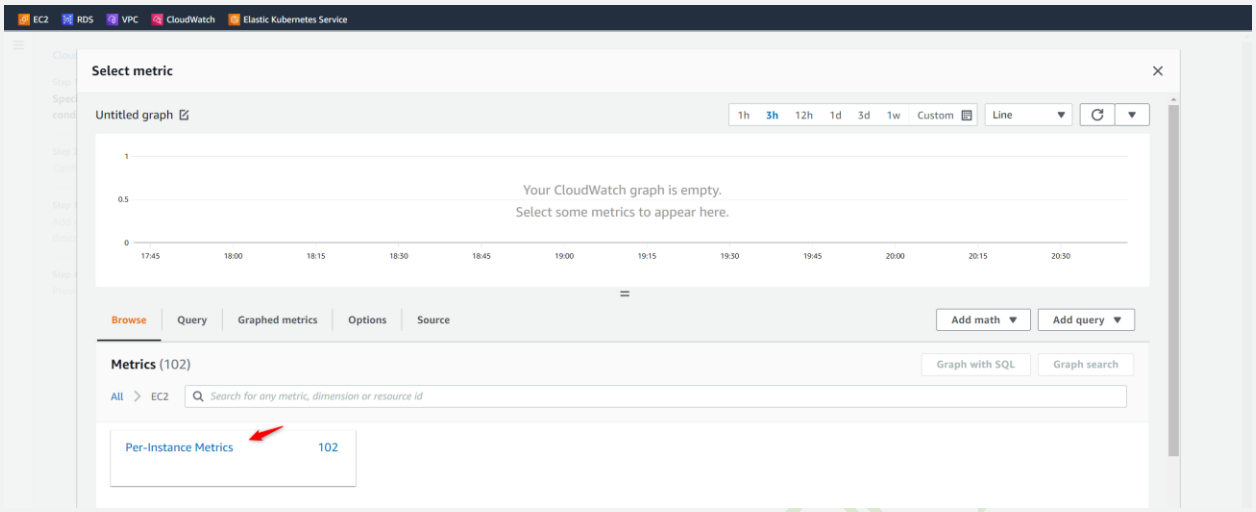
1. Go to Cloudwatch “Alarm – Create Alarm”
2. Select Metric – EC2
3. Select Per-Instance Metrics
4. Find the metric name: StatusCheckFailed_System
5. Create New SNS Topic
6. Complete the Cloudwatch process
7. Confirm SNS Subscription

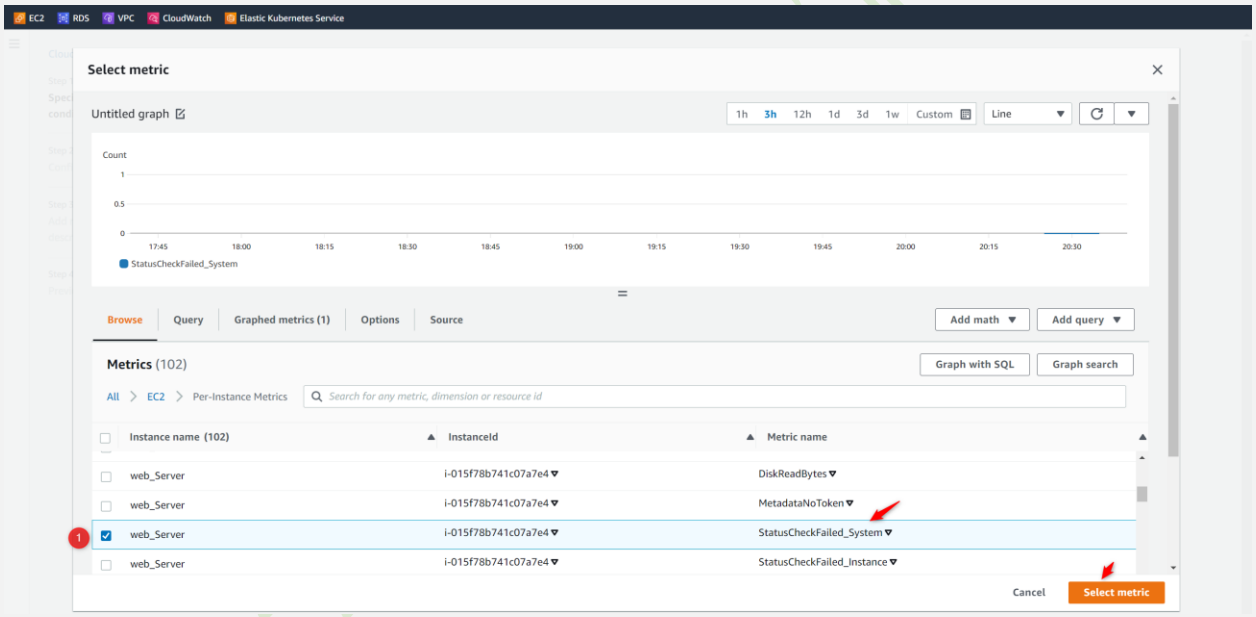
Create Alarm

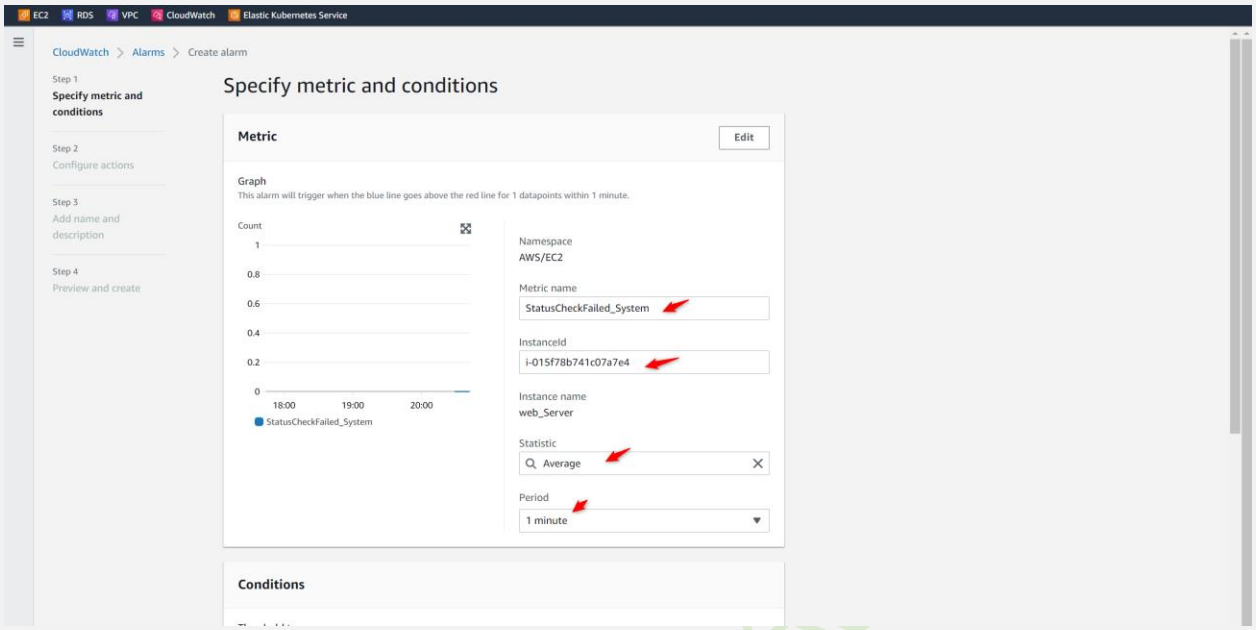

EC2
RDS
VPC
CloudWatch
Elastic Kubernetes Service
CloudWatch
Favorites and recents
Dashboards
Alarms
In alarm
All alarms
Billing
Logs
Log groups
Logs Insights
Metrics
CloudWatch > Alarms
Alarms (0)
Hide Auto Scaling alarms
Clear selection
Create composite alarm
Actions
Create alarm
Search
In alarm
Any type
Name
State
Last state update
Conditions
Actions
No alarms
No alarms to display
Read more about Alarms
Create alarm

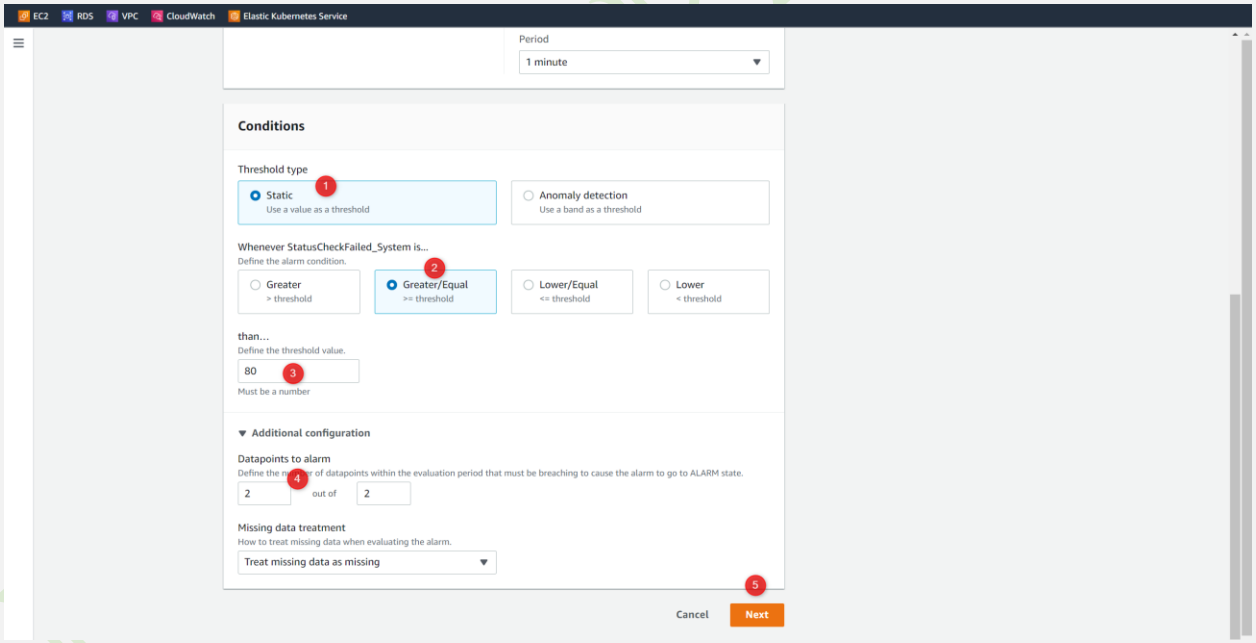

EC2
RDS
VPC
CloudWatch
Elastic Kubernetes Service
CloudWatch > Alarms > Create alarm
Step 1
Specify metric and conditions
Step 2
Configure actions
Step 3
Add name and description
Step 4
Preview and create
Specify metric and conditions
Metric
Graph
Preview of the metric or metric expression and the alarm threshold.
Select metric
Cancel
Next


EC2
RDS
VPC
CloudWatch
Elastic Kubernetes Service
Select metric
Untitled graph
1h
3h
12h
1d
3d
1w
Custom
Line
Your CloudWatch graph is empty.
Select some metrics to appear here.
Browse
Query
Graphed metrics
Options
Source
Add math
Add query
Metrics (384)
Graph with SQL
Graph search
Search for any metric, dimension or resource id
EBS 63
EC2 102
Events 6
Lambda 16
Logs 9
S3 8
SNS 4
Usage 176

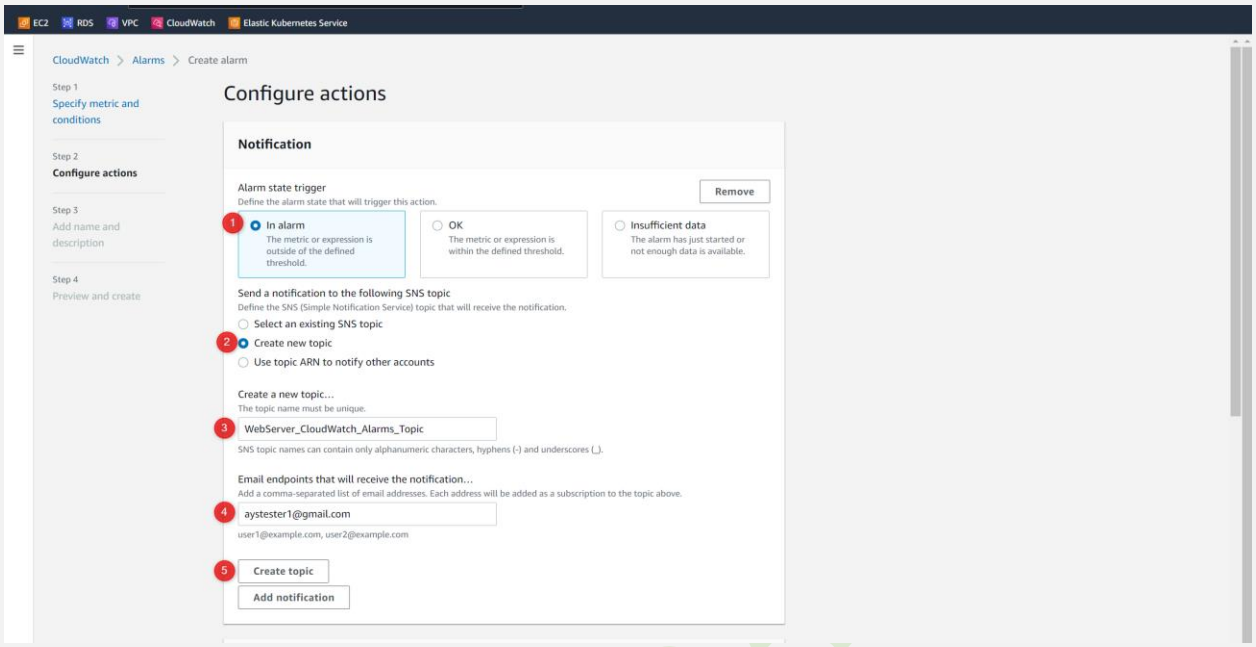

EC2
RDS
VPC
CloudWatch
Elastic Kubernetes Service
Select metric
Untitled graph
1h
3h
12h
1d
3d
1w
Custom
Line
Your CloudWatch graph is empty.
Select some metrics to appear here.
Browse
Query
Graphed metrics
Options
Source
Add math
Add query
Metrics (102)
Graph with SQL
Graph search
All
EC2
Search for any metric, dimension or resource id
Per-Instance Metrics
102


EC2
RDS
VPC
CloudWatch
Elastic Kubernetes Service
Select metric
Untitled graph
Count
StatusCheckFailed_System
Browse
Query
Graphed metrics (1)
Options
Source
Add math
Add query
Metrics (102)
Graph with SQL
Graph search
All
EC2
Per-Instance Metrics
Search for any metric, dimension or resource id
Instance name (102)
InstanceId
Metric name
web_Server
i-015f78b741c07a7e4
DiskReadBytes
MetadataNoToken
StatusCheckFailed_System
StatusCheckFailed_Instance
Cancel
Select metric



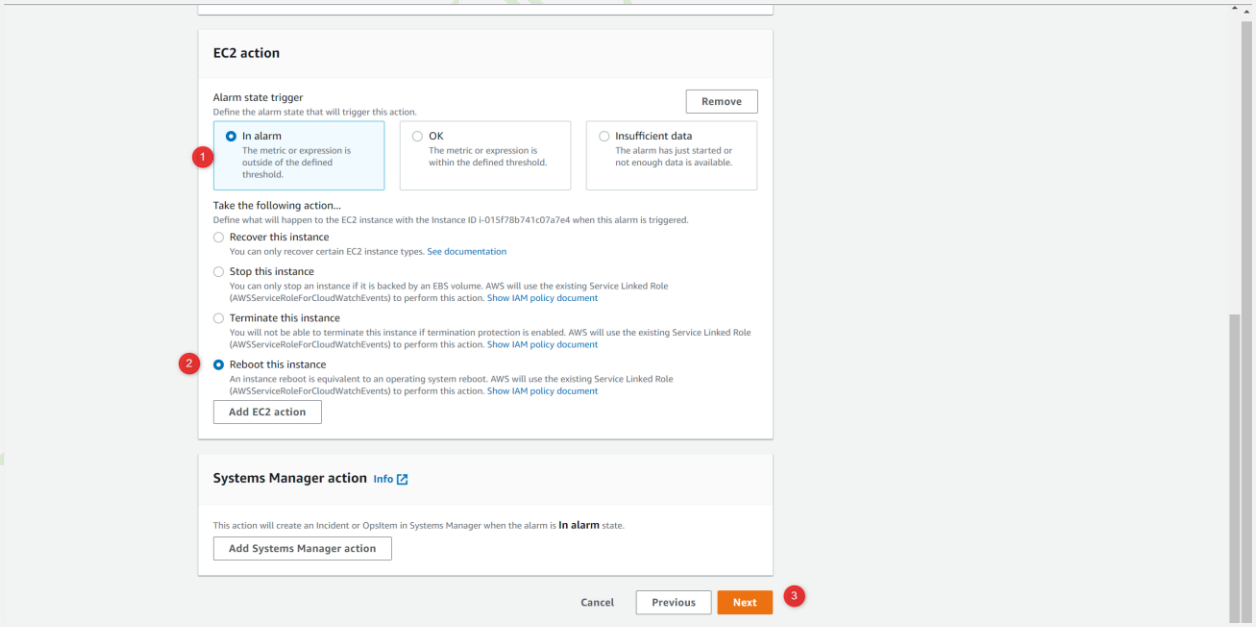


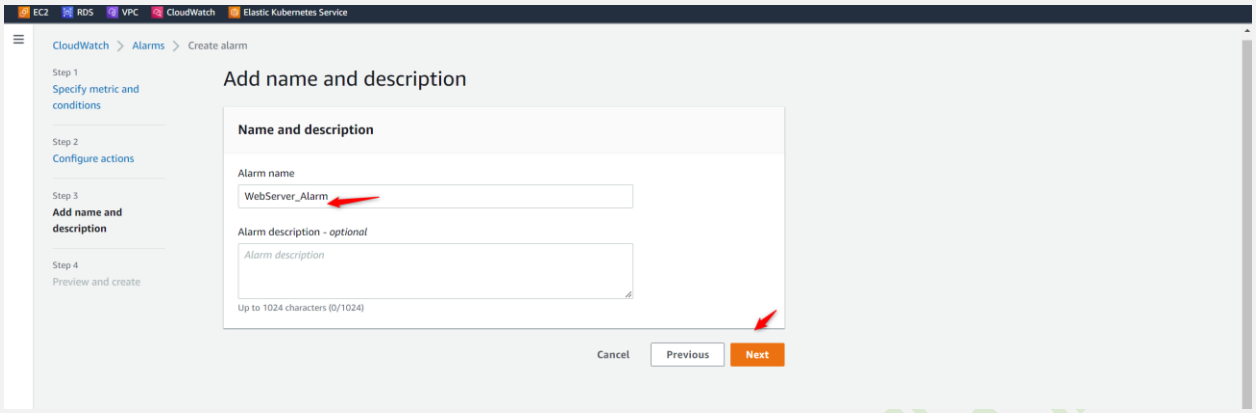
Create a New SNS Topic

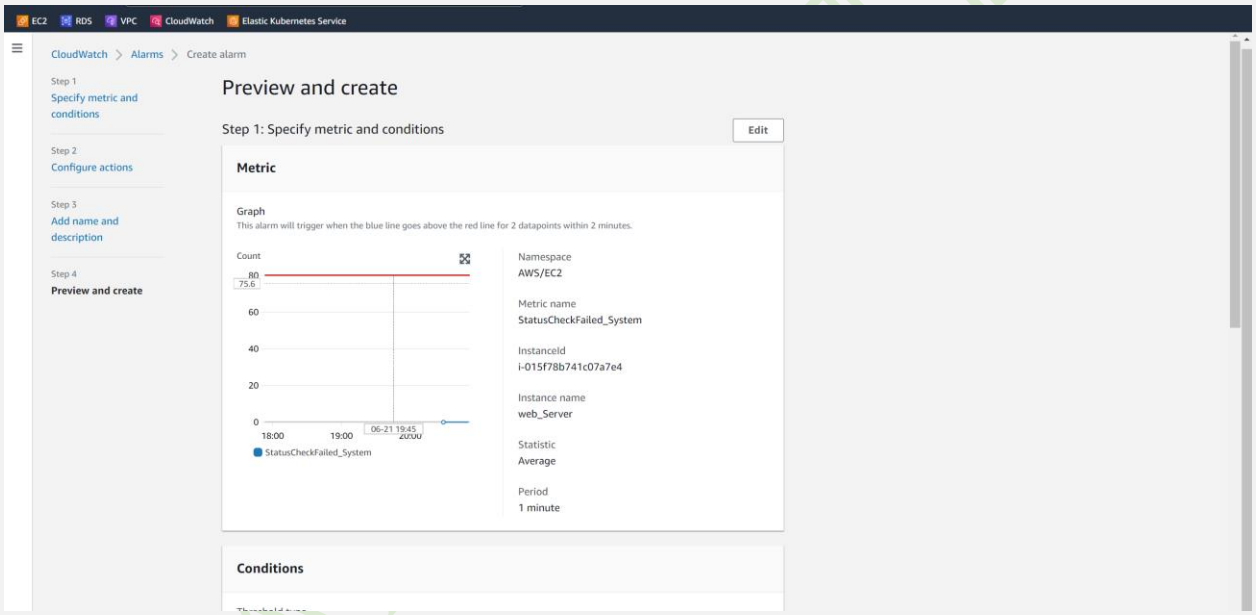


Go to EC2 Action

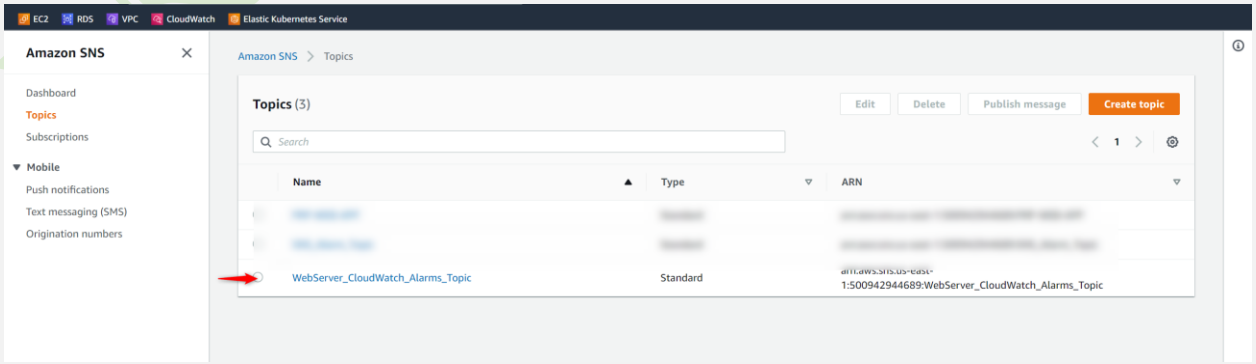
Choose: Recover this Instance

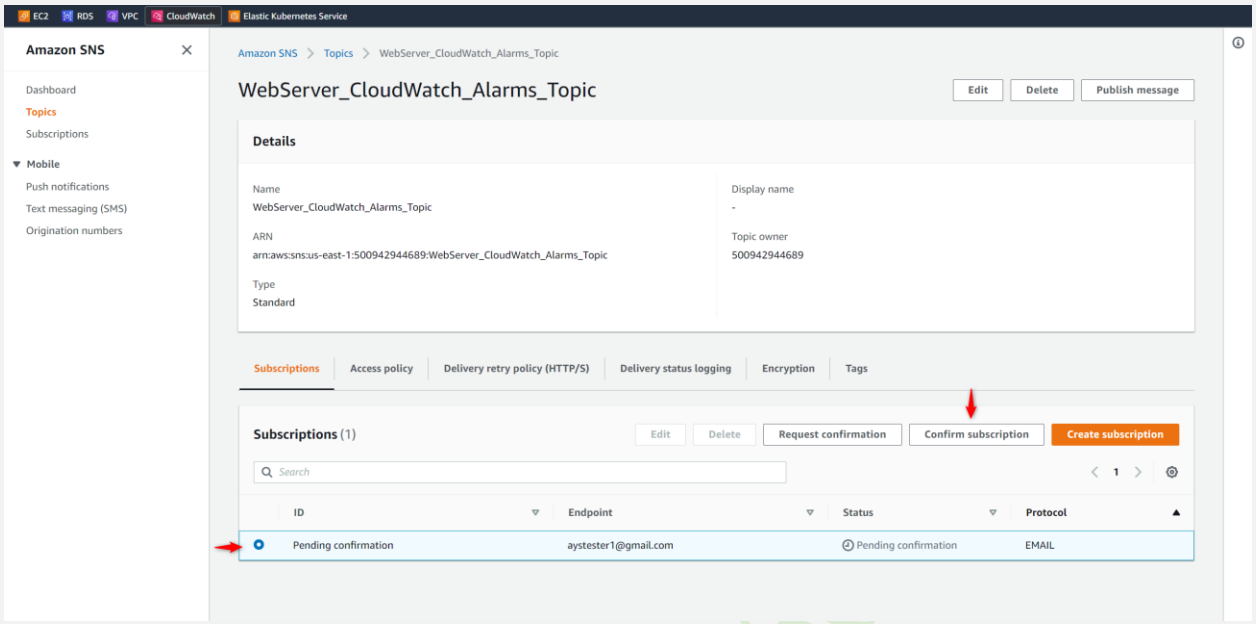






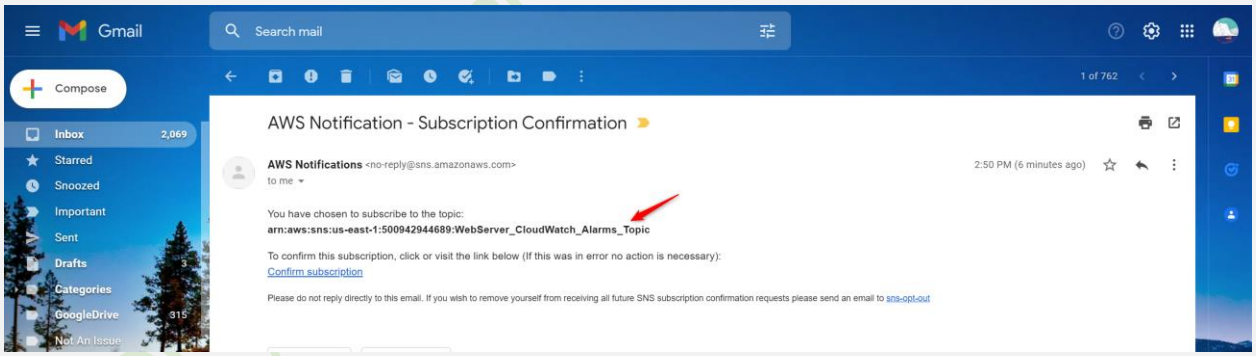
Go to SNS Service to confirm



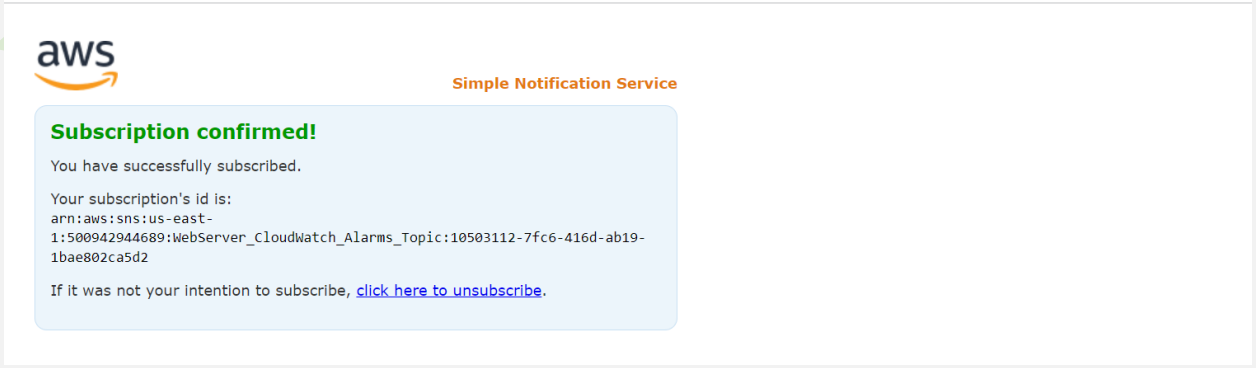


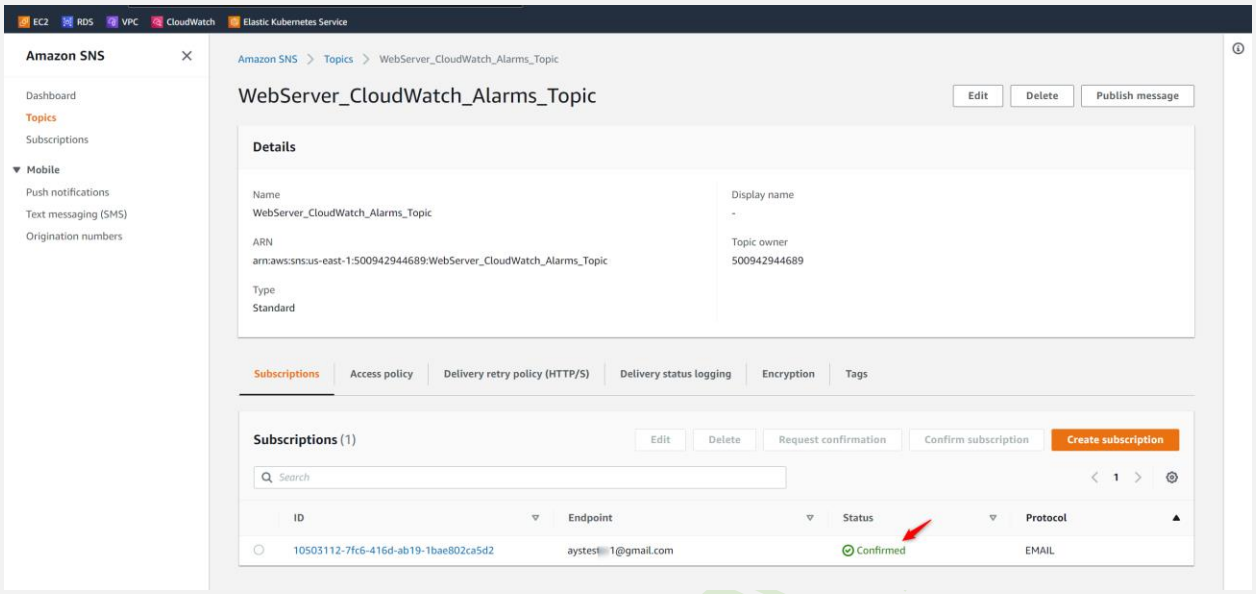
Confirm Subscription

Go to Gmail and grab the url

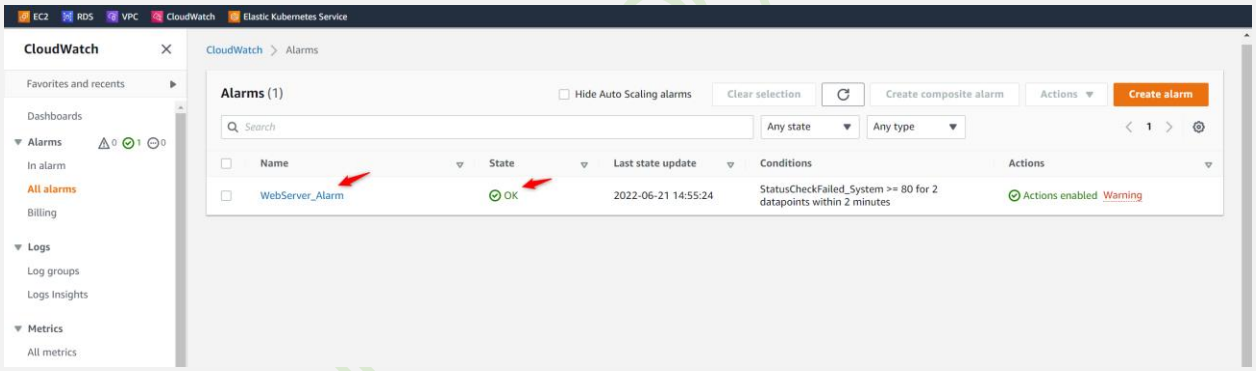


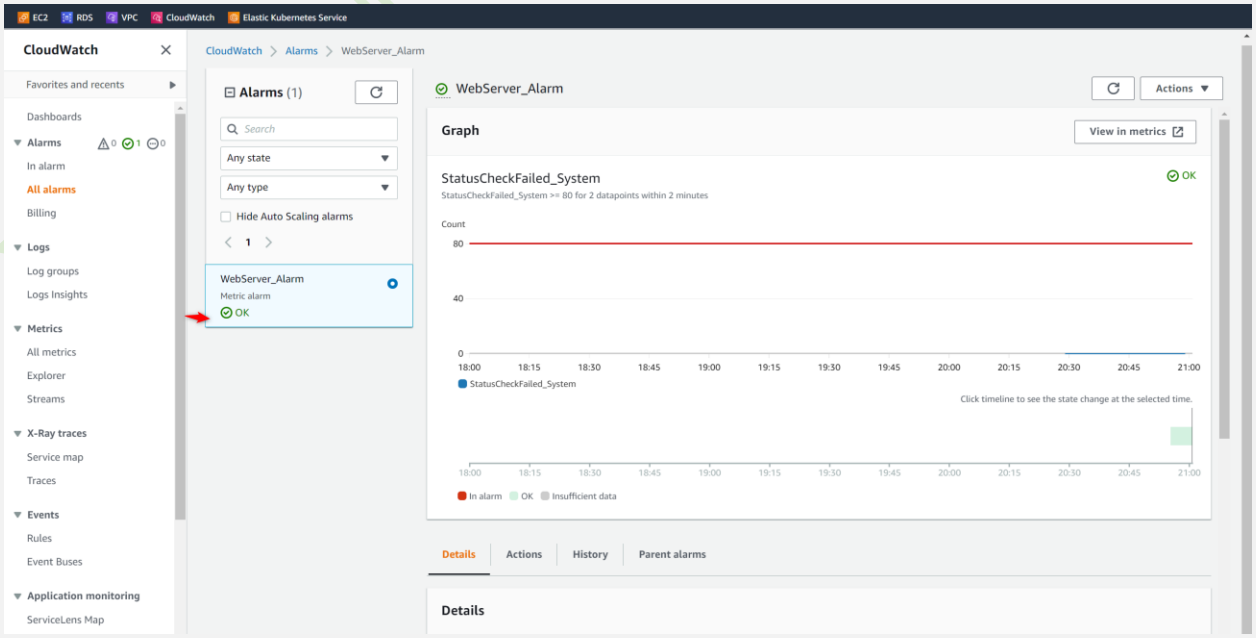
Click to confirm subscription





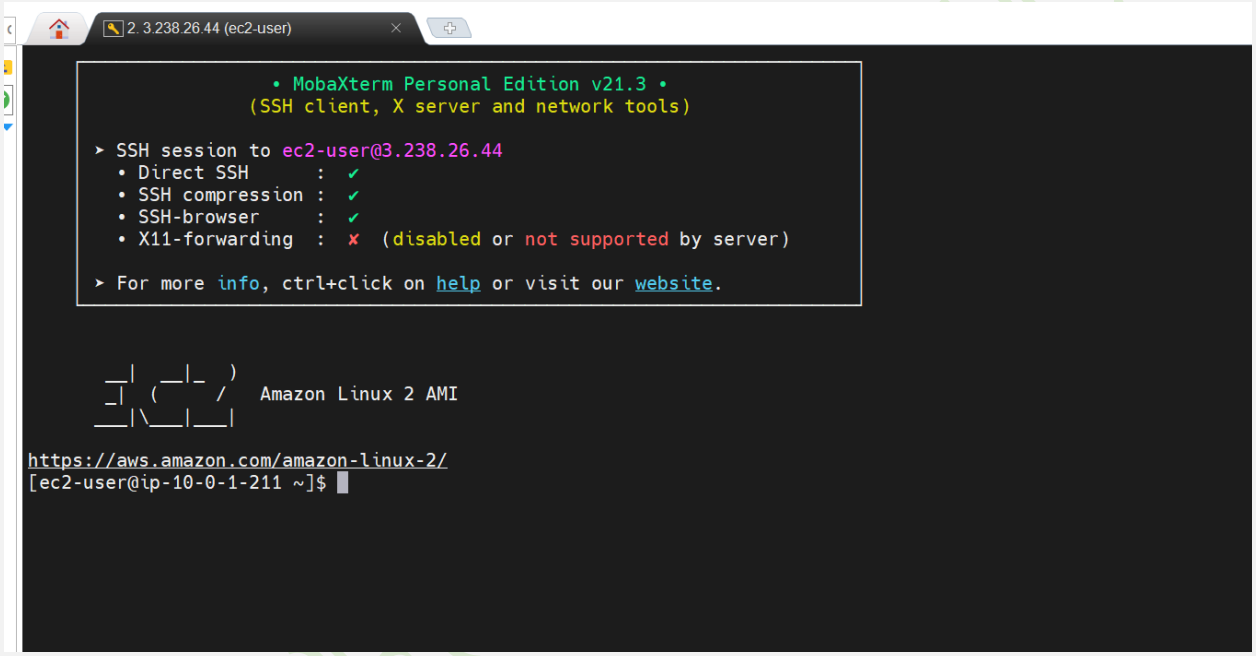
Go to Cloudwatch and observe the Alarm status





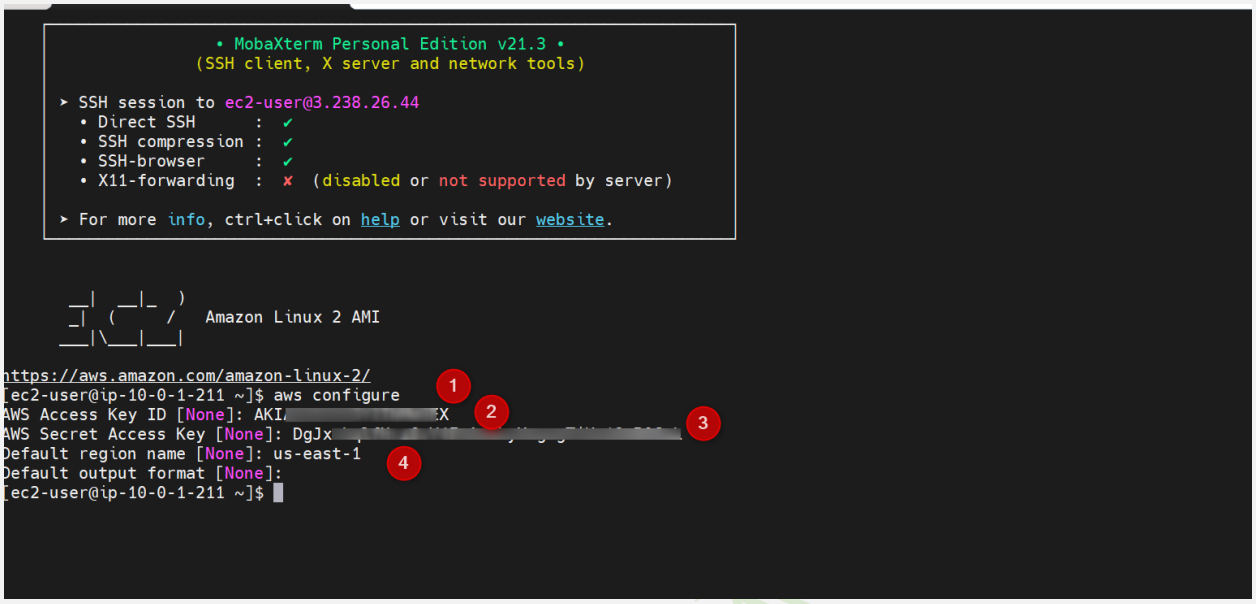
- Phase 4: Simulate and Trigger a Disaster recovery scenario.
  - Prepare manual system failure script
  - Write a script to remove an application files from the apache root directory /var/www/html/
  - Run the script from EC2 machine.
  - Verify that all application files removed
  - Verify that web application is not accessible.

Login to EC2 machine

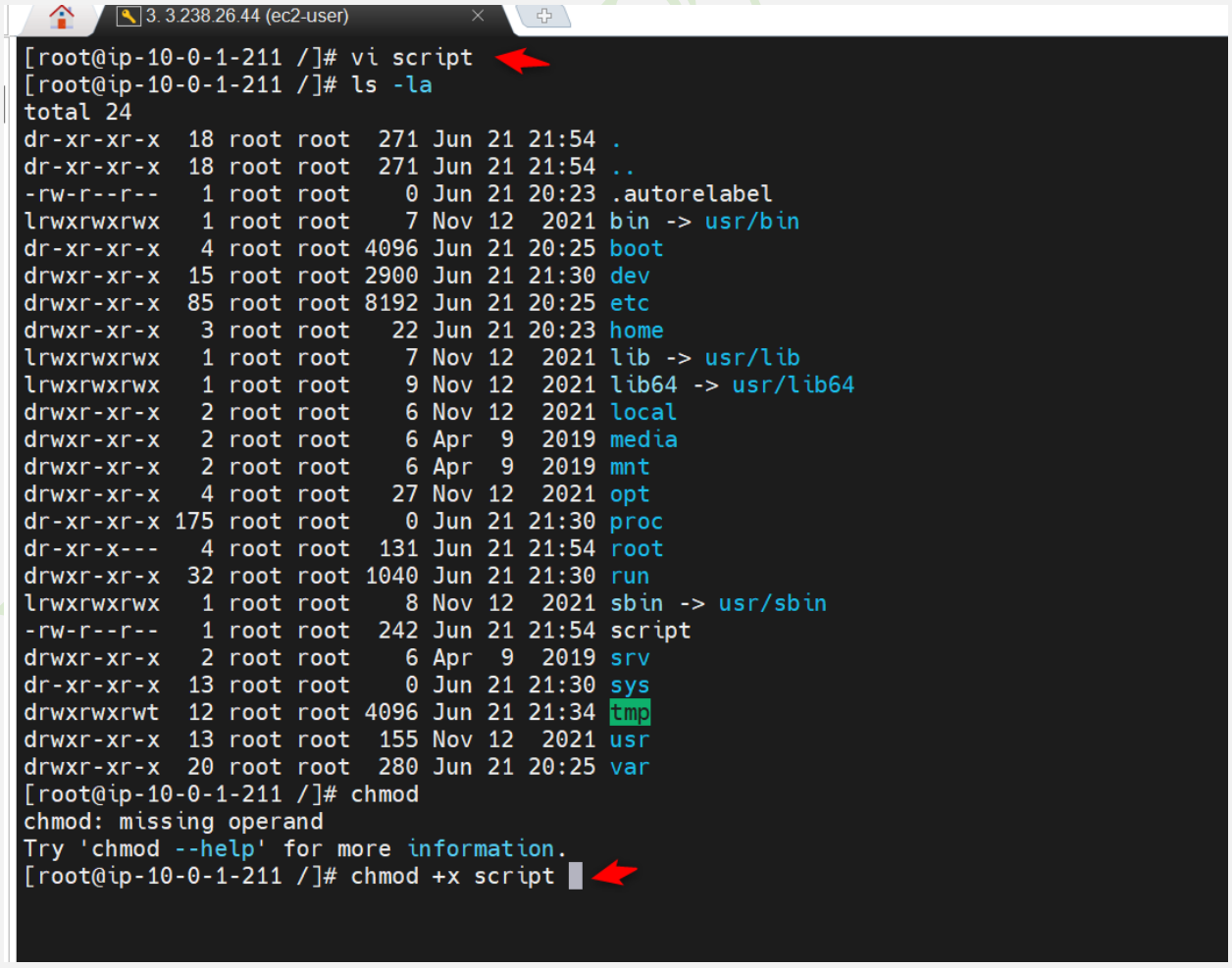


AWS configure with new user into EC2 machine

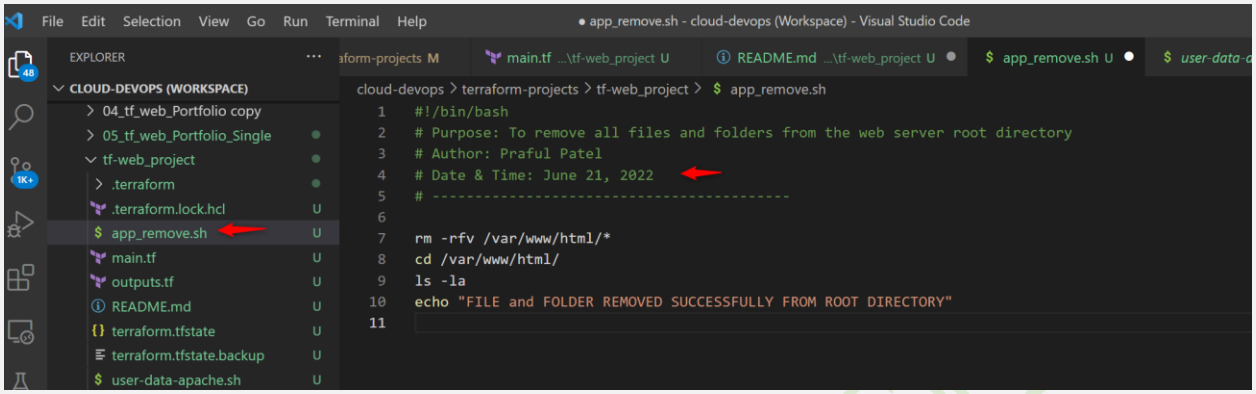
Aws configure



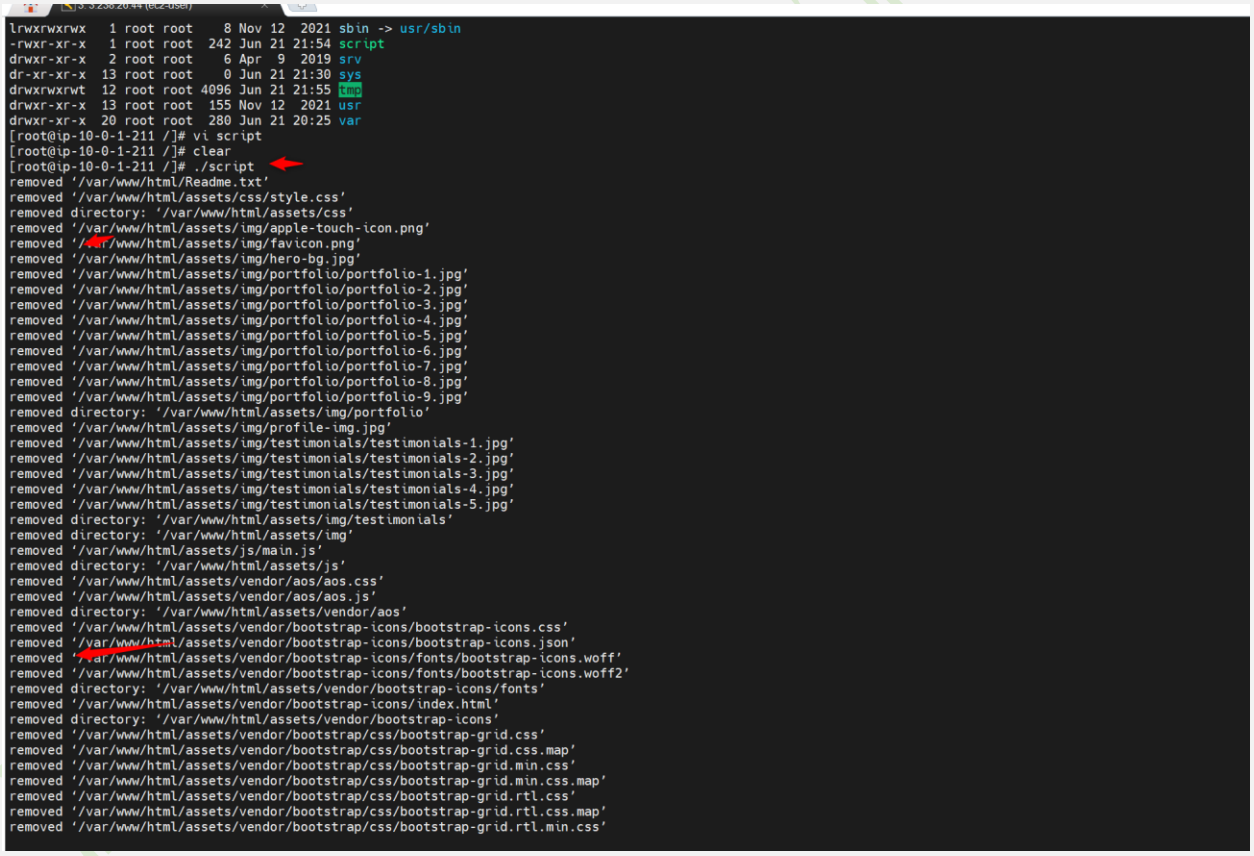
Simulate Web Server app failure by removing application code:



Create an app remove script



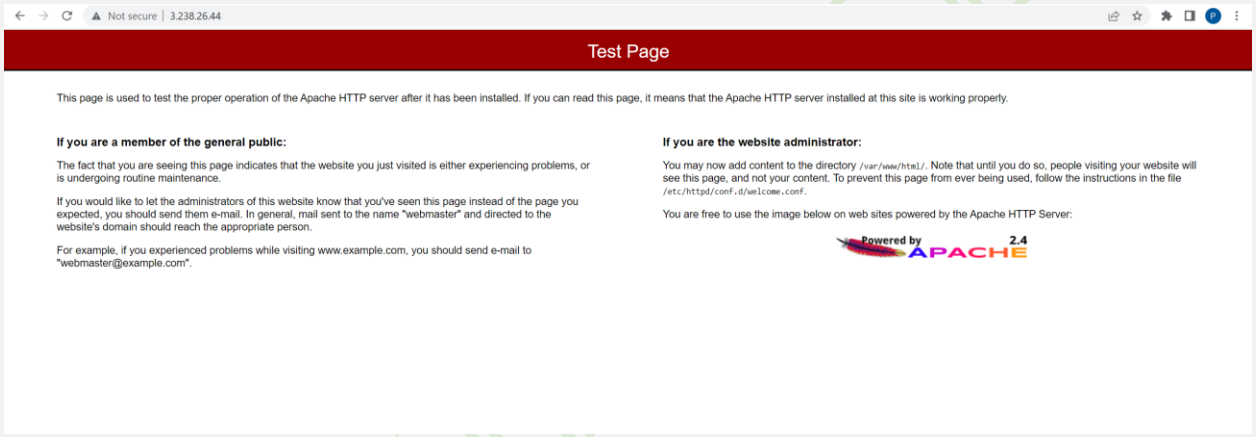
Run the script



```
removed directory: '/var/www/html/assets/vendor'
removed directory: '/var/www/html/assets'
removed '/var/www/html/changelog.txt'
removed '/var/www/html/emp.php'
removed '/var/www/html/forms/Readme.txt'
removed '/var/www/html/forms/contact.php'
removed '/var/www/html/forms/db.php'
removed '/var/www/html/forms/insert.php'
removed directory: '/var/www/html/forms'
removed '/var/www/html/inc/dbinfo.inc'
removed directory: '/var/www/html/inc'
removed '/var/www/html/index.html'
removed '/var/www/html/inner-page.html'
removed '/var/www/html/portfolio-details.html'
removed '/var/www/html/userdata.sh'
removed directory: '/var/www/html/'
ls: cannot access /var/www/html/: No such file or directory
[root@ip-10-0-1-211 /]#
```

Verify all files removed from root directory

Verified that web application removed



- Phase 5: Simulate Cloudwatch monitoring 'In-Alarm"
  - Login to EC2 machine.
  - Become a root user
  - Configure aws configure
  - Run the cloudwatch set-alarm script to put into "In-Alarm" status
  - Go to Cloudwatch and verify that status is turned from "OK" to "In-Alarm"
  - Go to email and verify that email is received with necessary information.

Simulate the EC2 webServer System Failure by CloudWatch

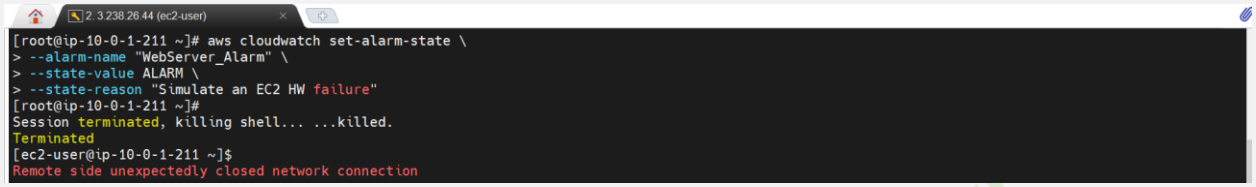
Become a root user

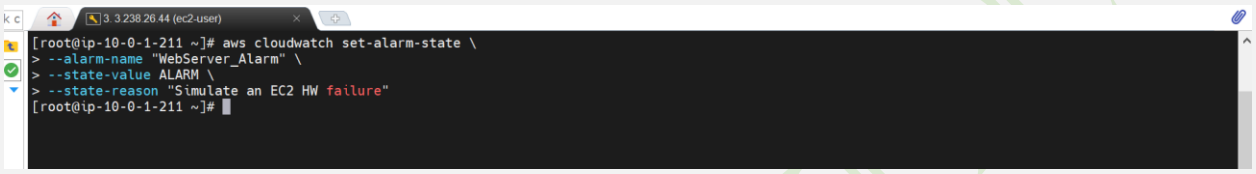
Sudo su –

Prepare a simulation script which can simulate the Alarm in status

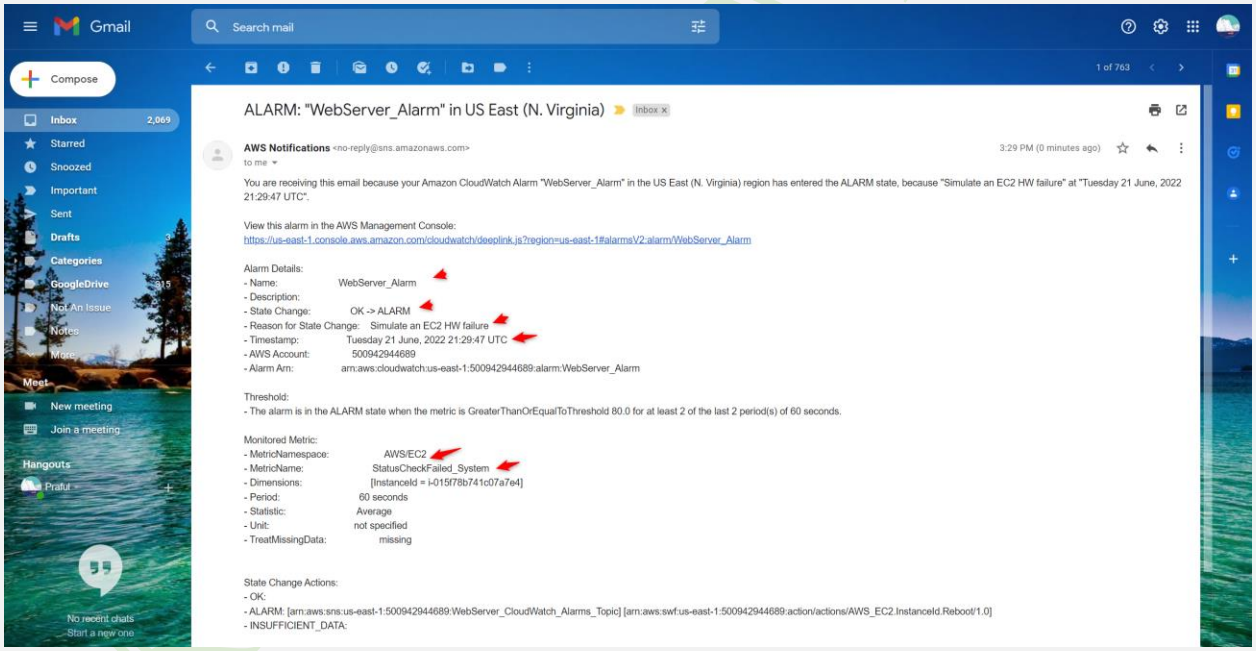
```
cloudwatch alarm trigger:
aws cloudwatch set-alarm-state \
--alarm-name "WebServer_Alarm" \
--state-value ALARM \
```

```
--state-reason "Simulate an EC2 HW failure"
```

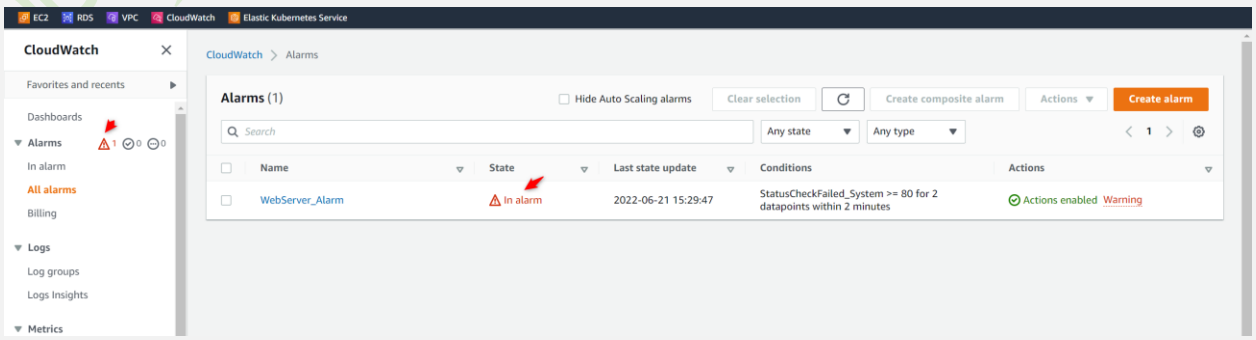


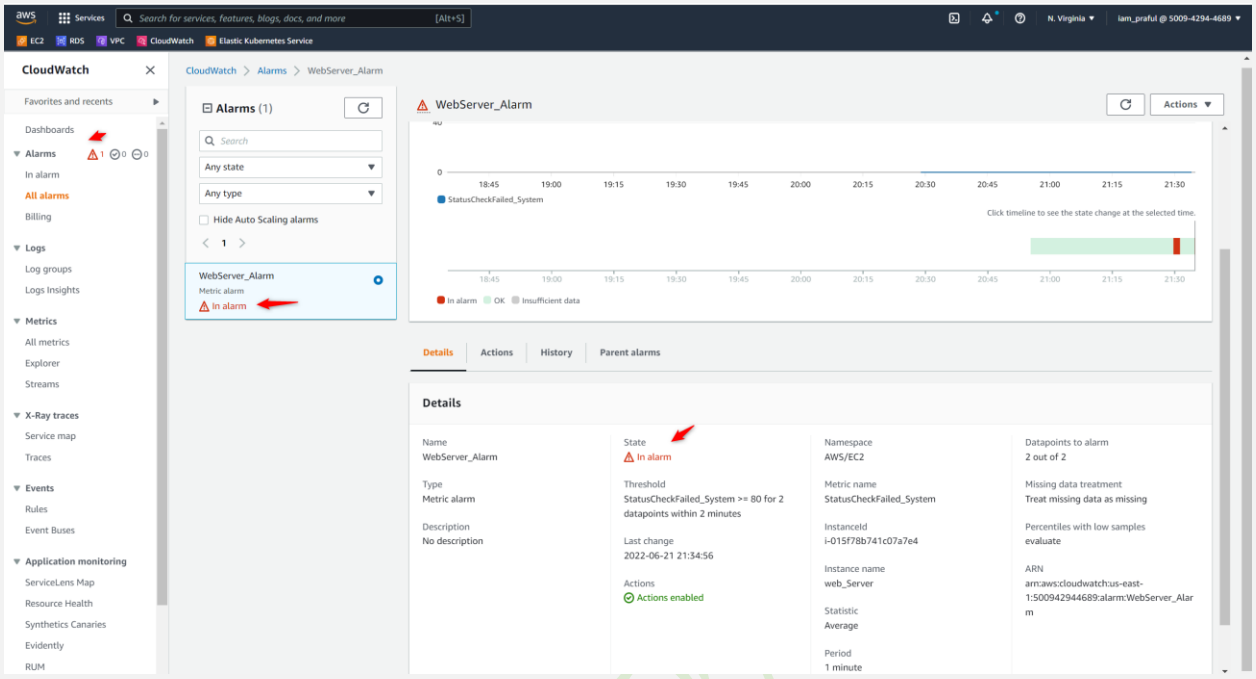


Email Received

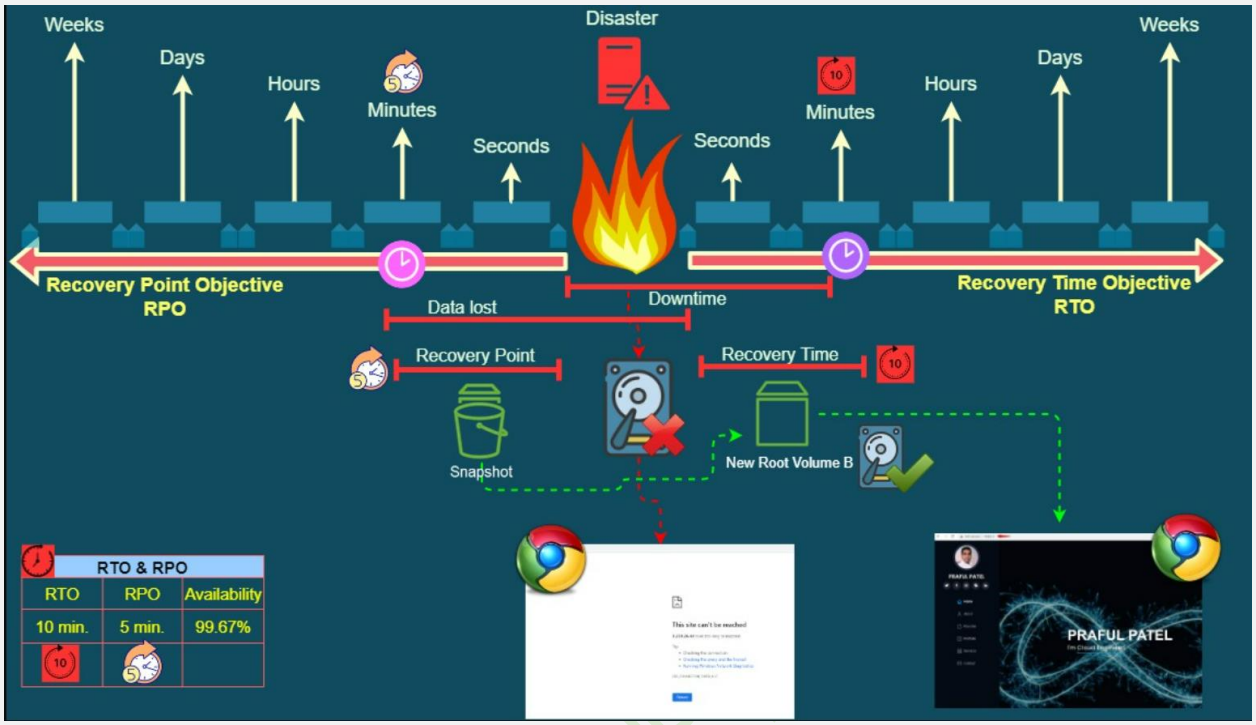


Cloudwatch Alarm status changed to In-Alarm





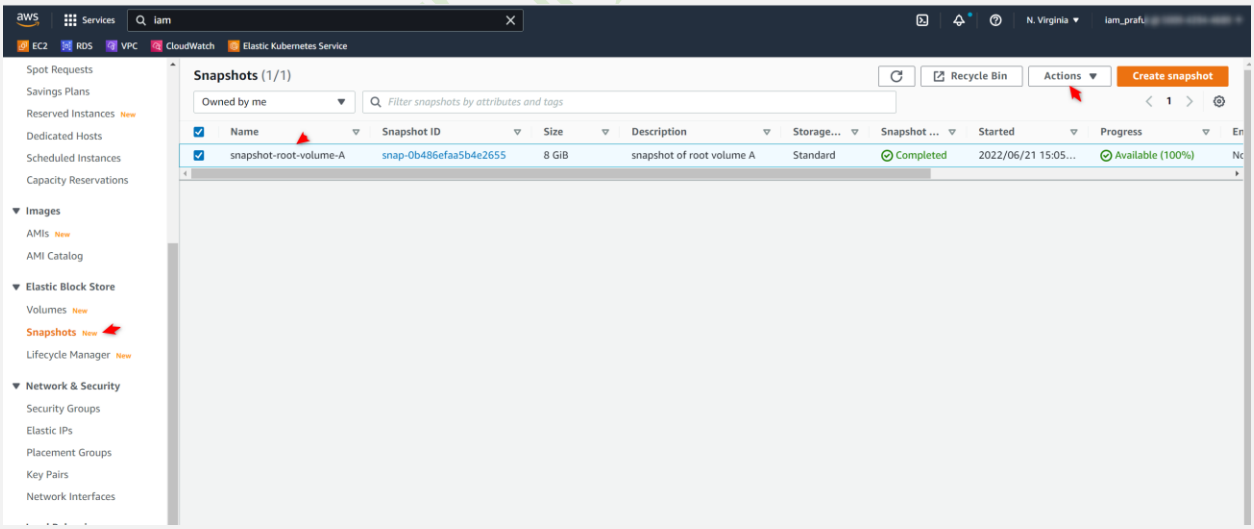
- Phase 6: Recover from Disaster condition (RPO 5 min. RTO 10 min.)
  - Go to EC2 machine
  - Go to Action – make sure that ec2 machine is running.
  - Select an option “Monitor and troubleshoot”
  - Select an option “Replace root volume”
  - Select a recent snapshot taken within last 5 minutes to meet the RPO condition
  - Attach complete the snapshot
  - Go to Volume and verify that the new snaphost volume is attached and “ In-Use” status
  - Verify that old root volume is “Available” status which is no use and corrupted now.
  - Verify that web application is now accessible



Recover Root Instance from Failure

Go to Snapshot

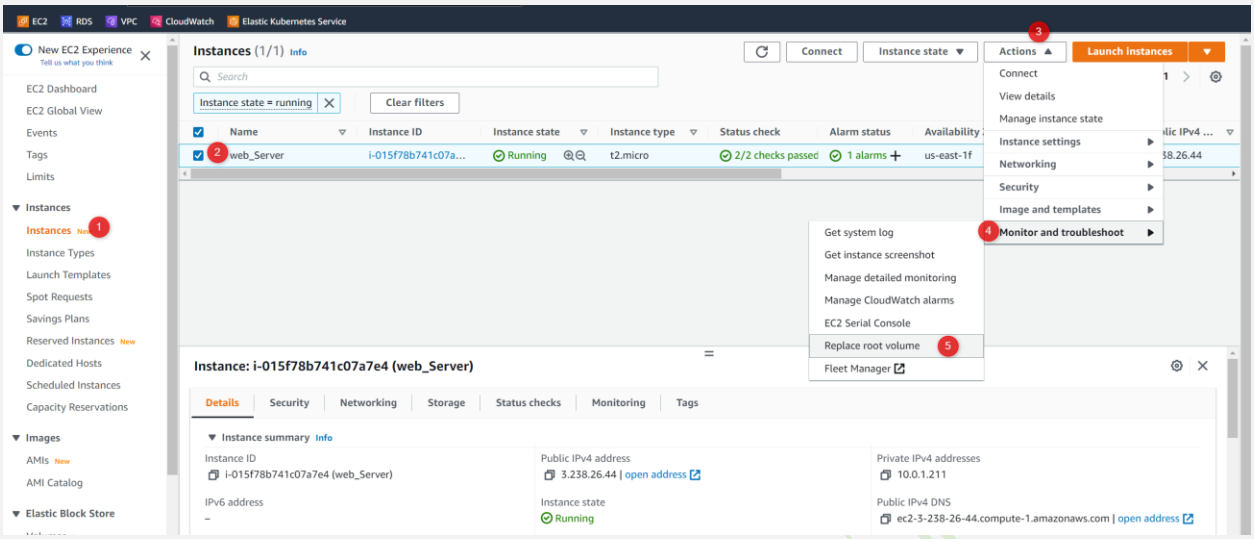
Verify that snapshot is available

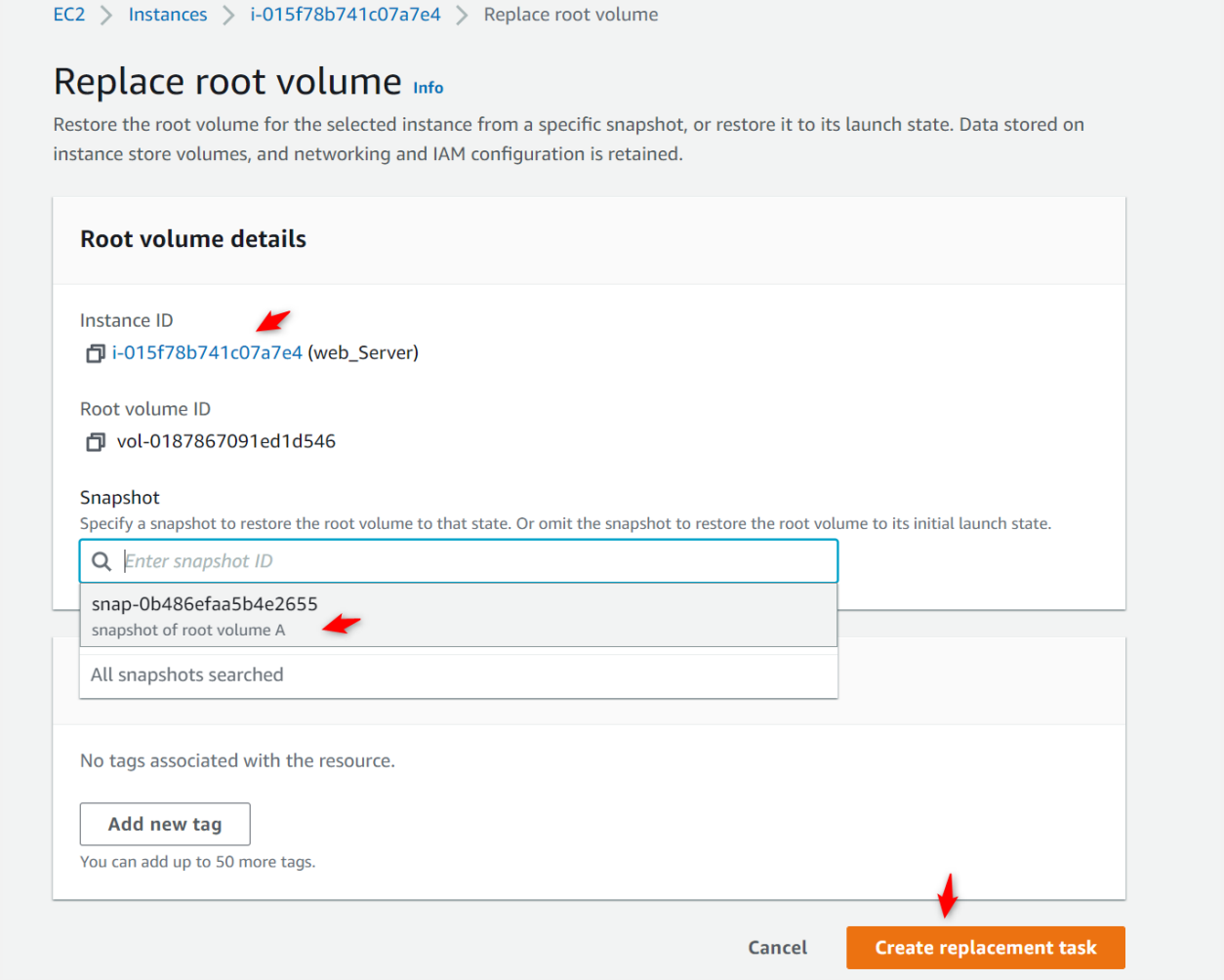


Attach new volume to WebServer EC2 instance

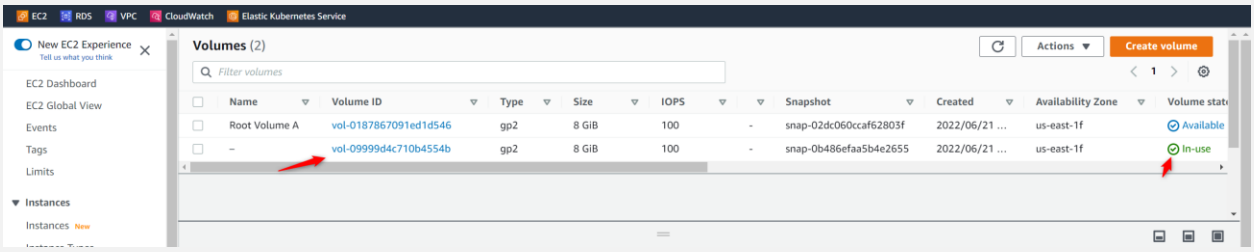
Go to EC2

Replace root volume

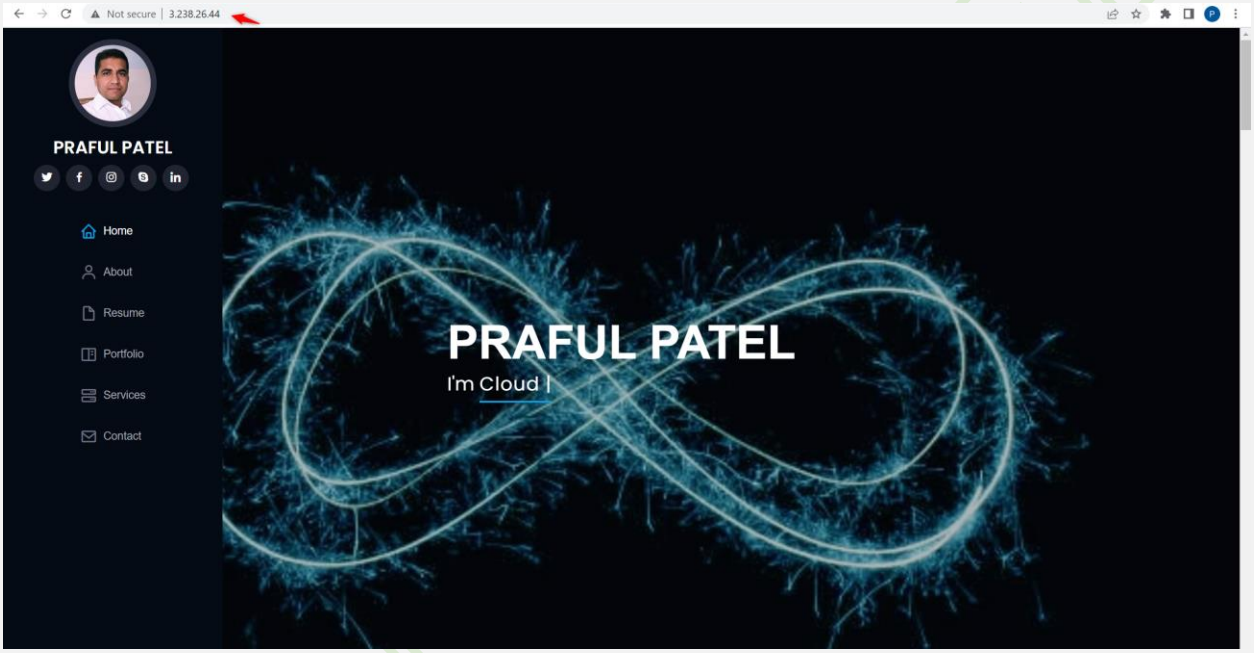




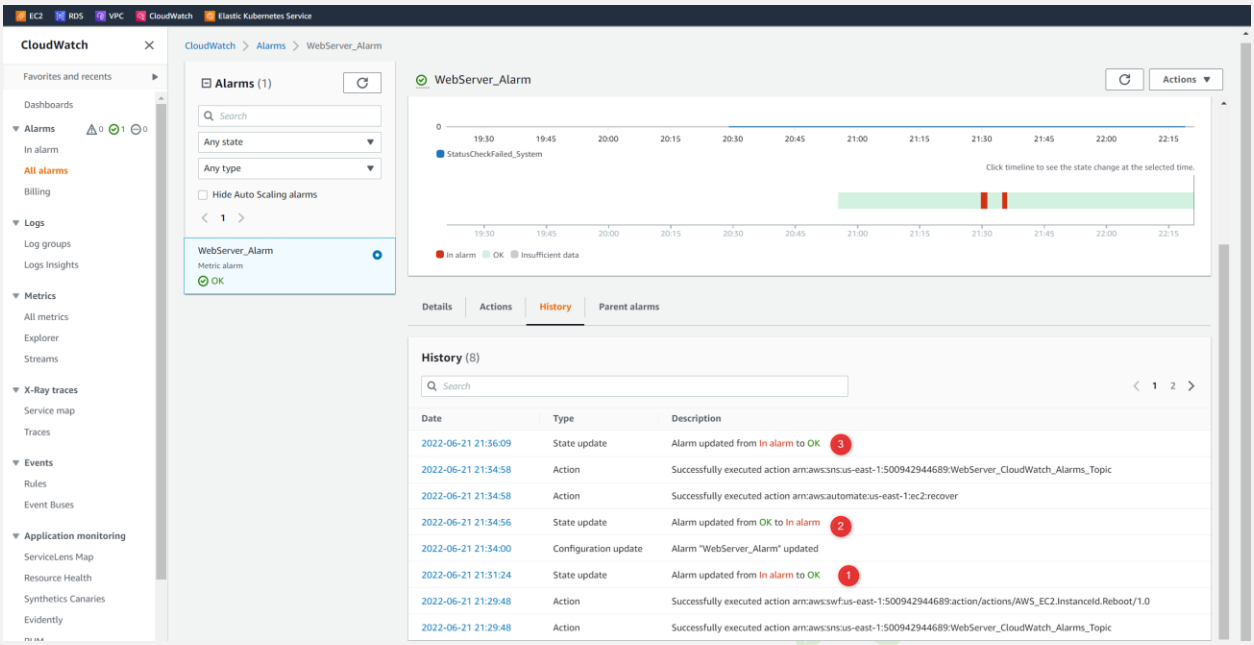
New Volume created and attached to ec2 and “In-Use” and older one is in “Available”



Go to Web browser and verify that web application is up and running again after recovery



Go to Monitoring and verify that Cloudwatch Alarm is in “OK” Status

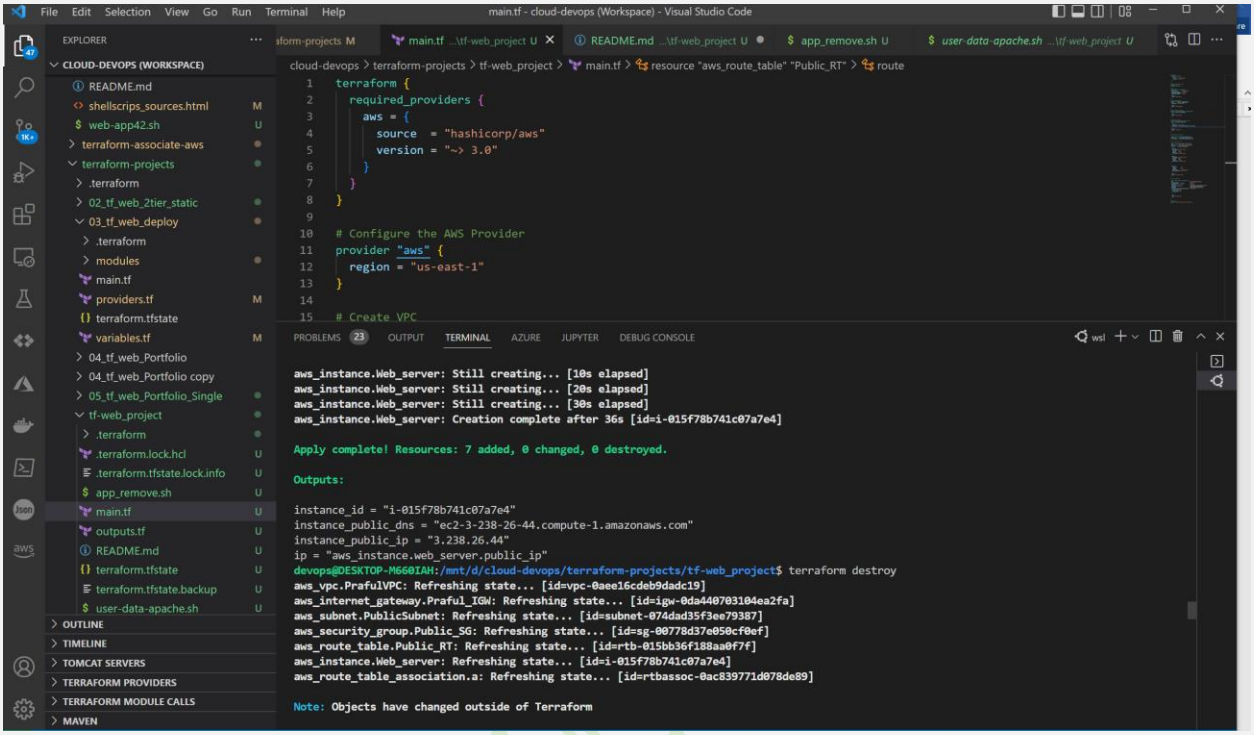


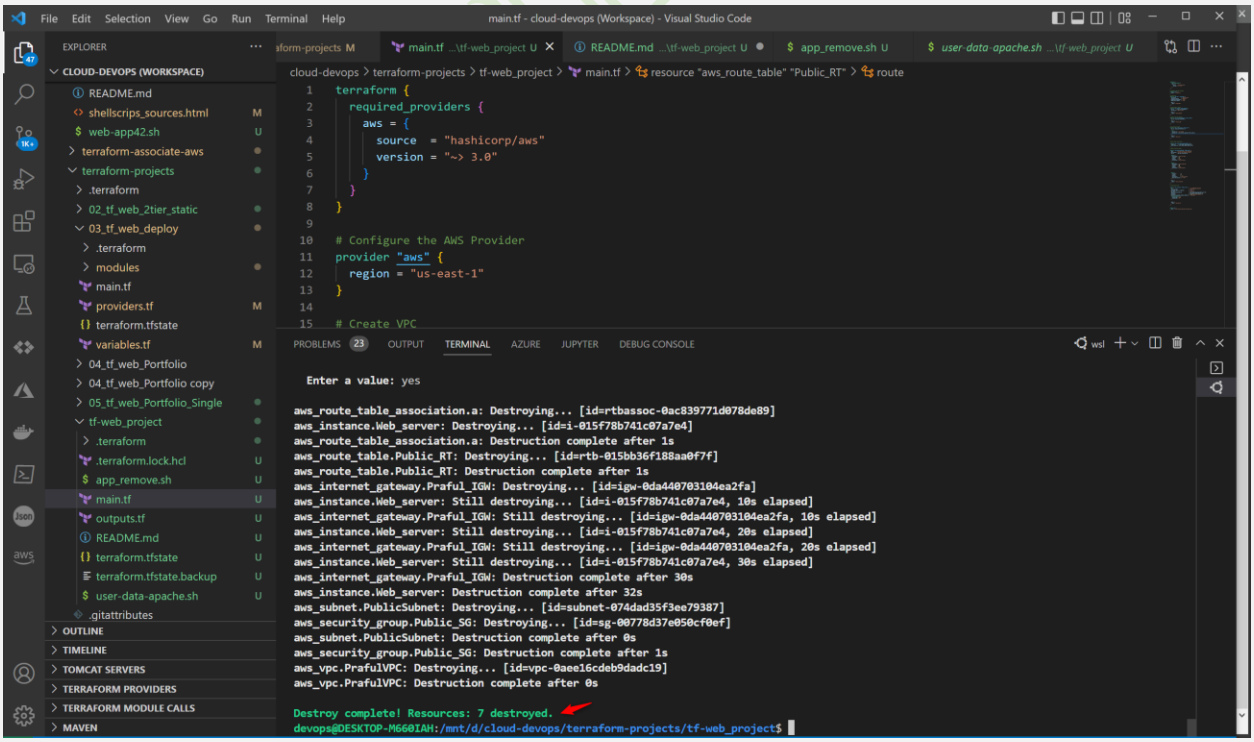
CHALLENGE

Congratulations!!!!

Clean up Project:

Terraform destroy





Remove volume

Remove snaphosts

Remove Cloudwatch Alarm

Remove SNS Topic

Resources:

https://wellarchitectedlabs.com/reliability/300_labs/300_testing_for_resiliency_of_ec2_rds_and_s3/6_failure_injection_app/
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/monitoring-system-instance-status-check.html
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/replace-root.html#view-replacement-tasks
https://github.com/terraform-aws-modules/terraform-aws-cloudwatch
https://docs.ansible.com/ansible/2.5/modules/ec2_snapshot_module.html

Congratulations!!!!