



<https://linkedin.com/in/prafulpatel16>



<https://github.com/prafulpatel16>

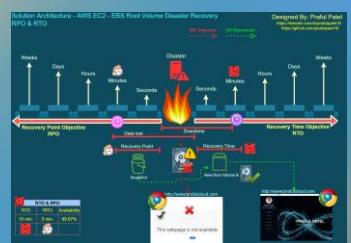
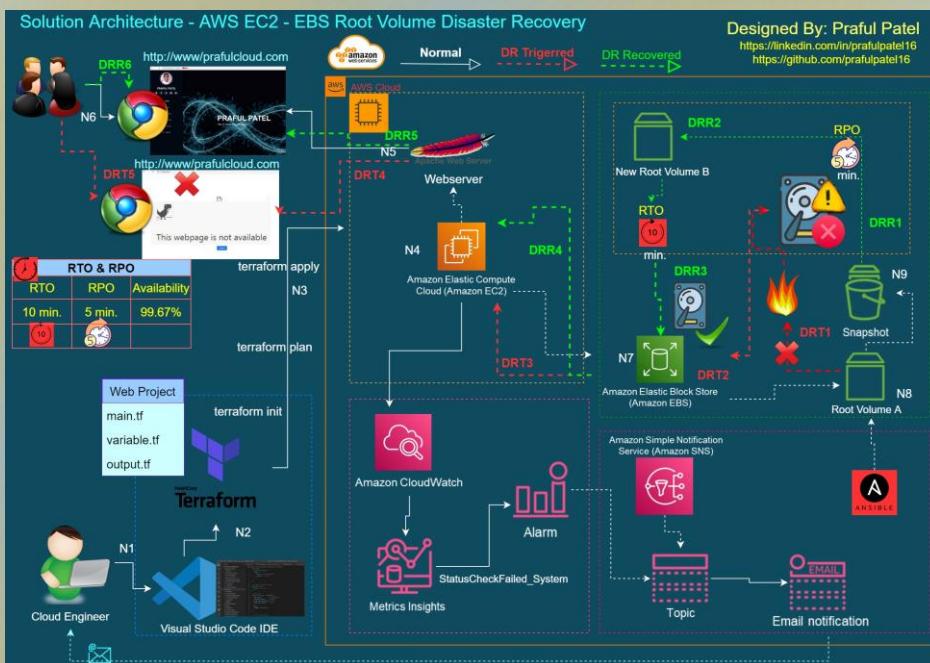


<https://medium.com/@prafulpatel16>



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

SOLUTION DESIGN & IMPLEMENTATION BY: PRAFUL PATEL



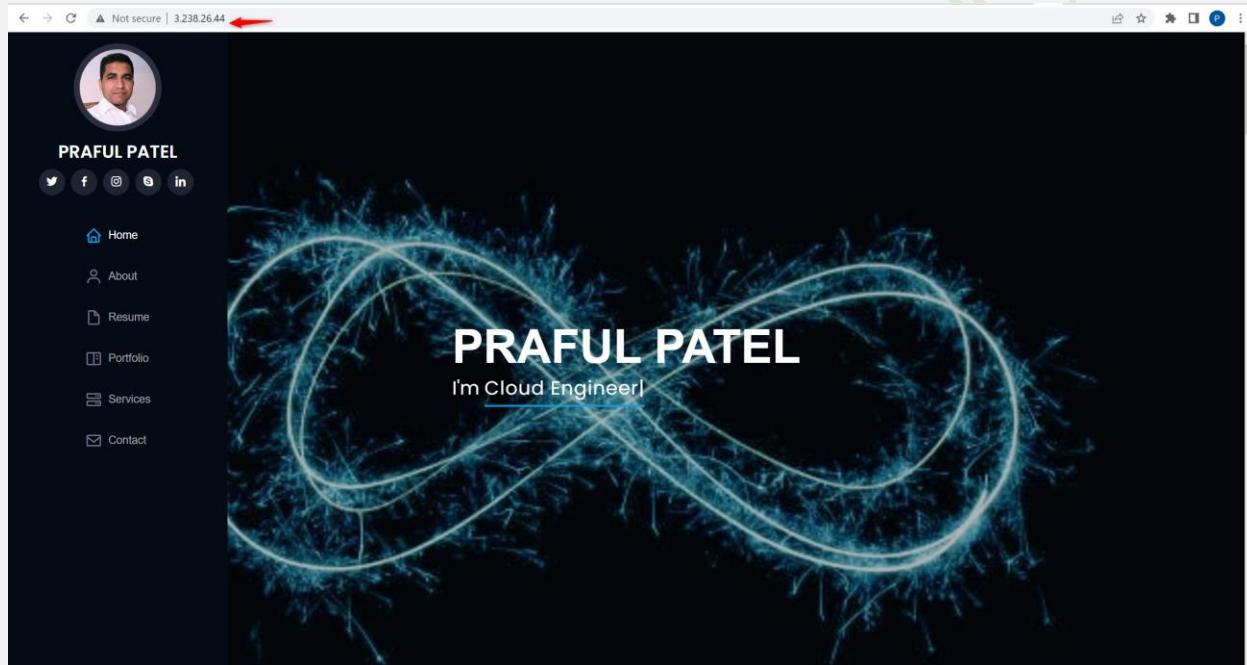
Date: June 21, 2022

➤ **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 configured 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.

RTO & RPO		
RTO	RPO	Availability
10 min.	5 min.	99.67%
* 	* 	*

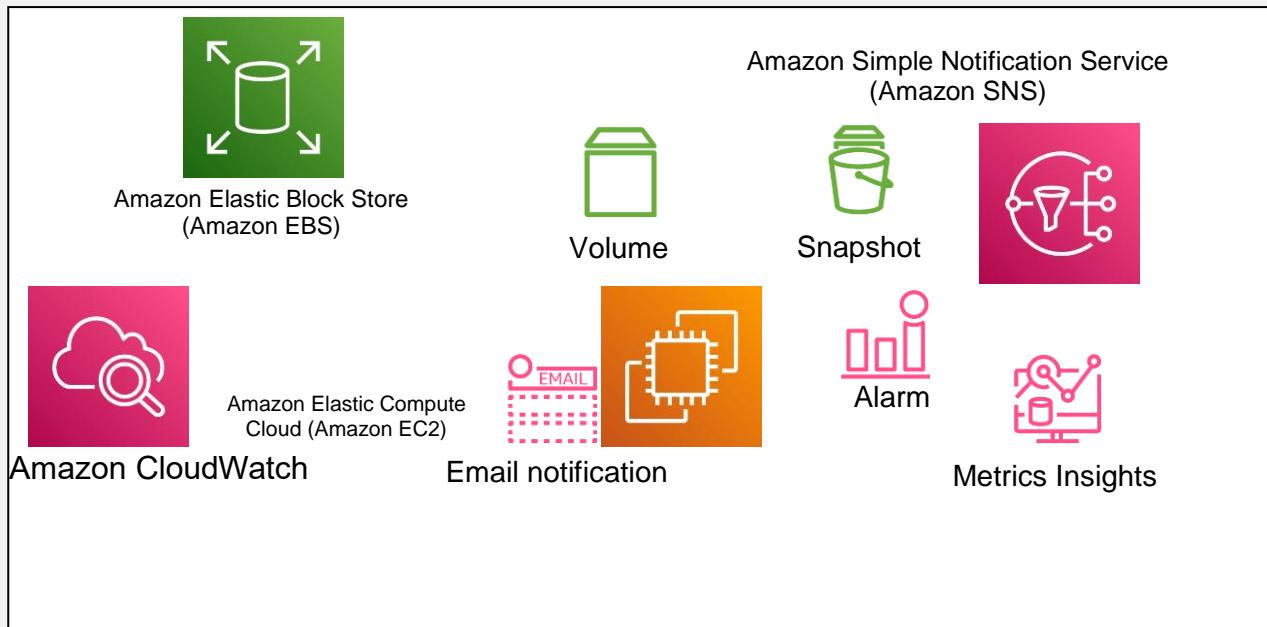
This project demonstrates an experience of designing and implementing of Disaster recovery scenario which can fulfil the defined business RPO & RTO requirements, along with cloudwatch monitoring and SNS notification system.

#### ➤ 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



**AWS** offers four levels of DR support across a spectrum of complexity and time



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

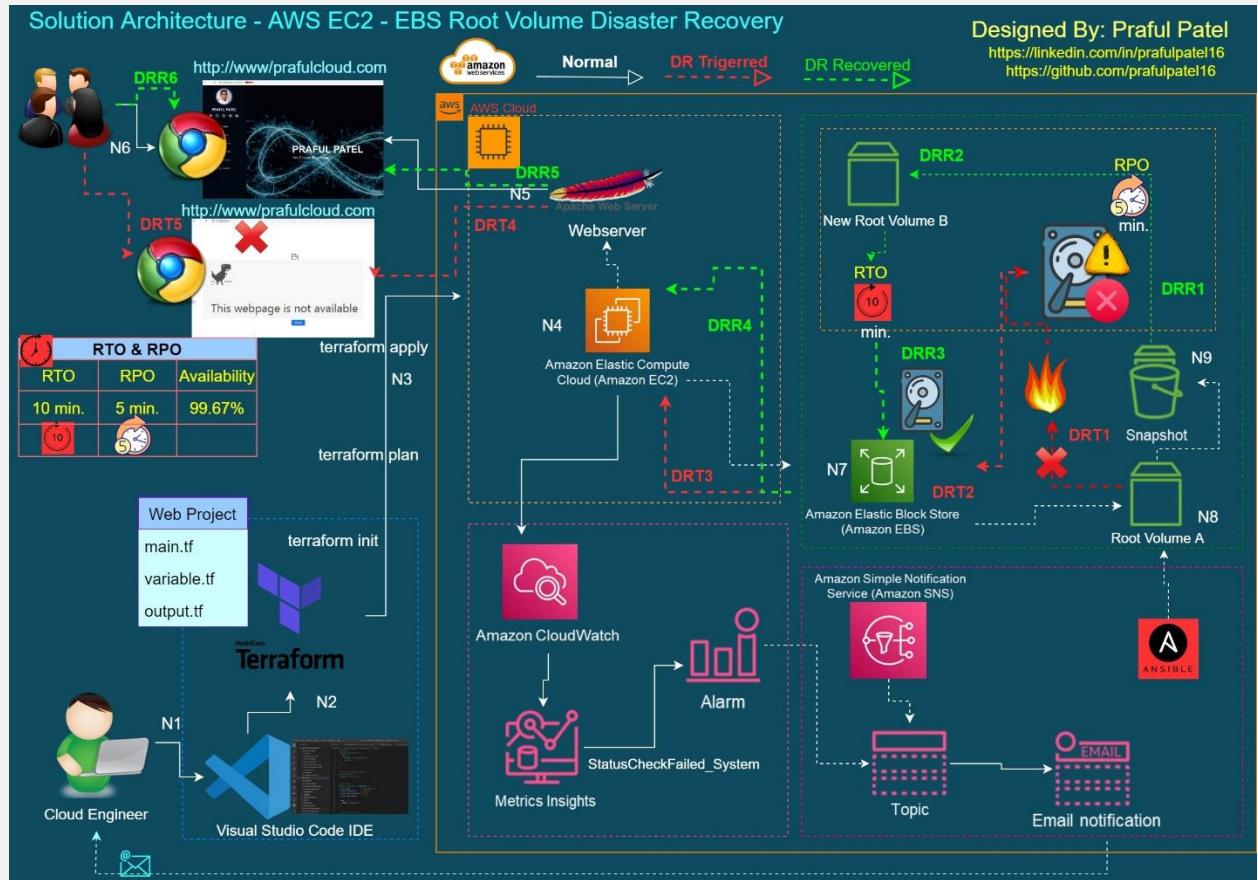
Instance status checks monitor the software and network configuration of your individual instance. Amazon EC2 checks the health of the instance by sending an address resolution protocol (ARP) request to the network interface (NIC). These checks detect problems that require your involvement to repair. 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:

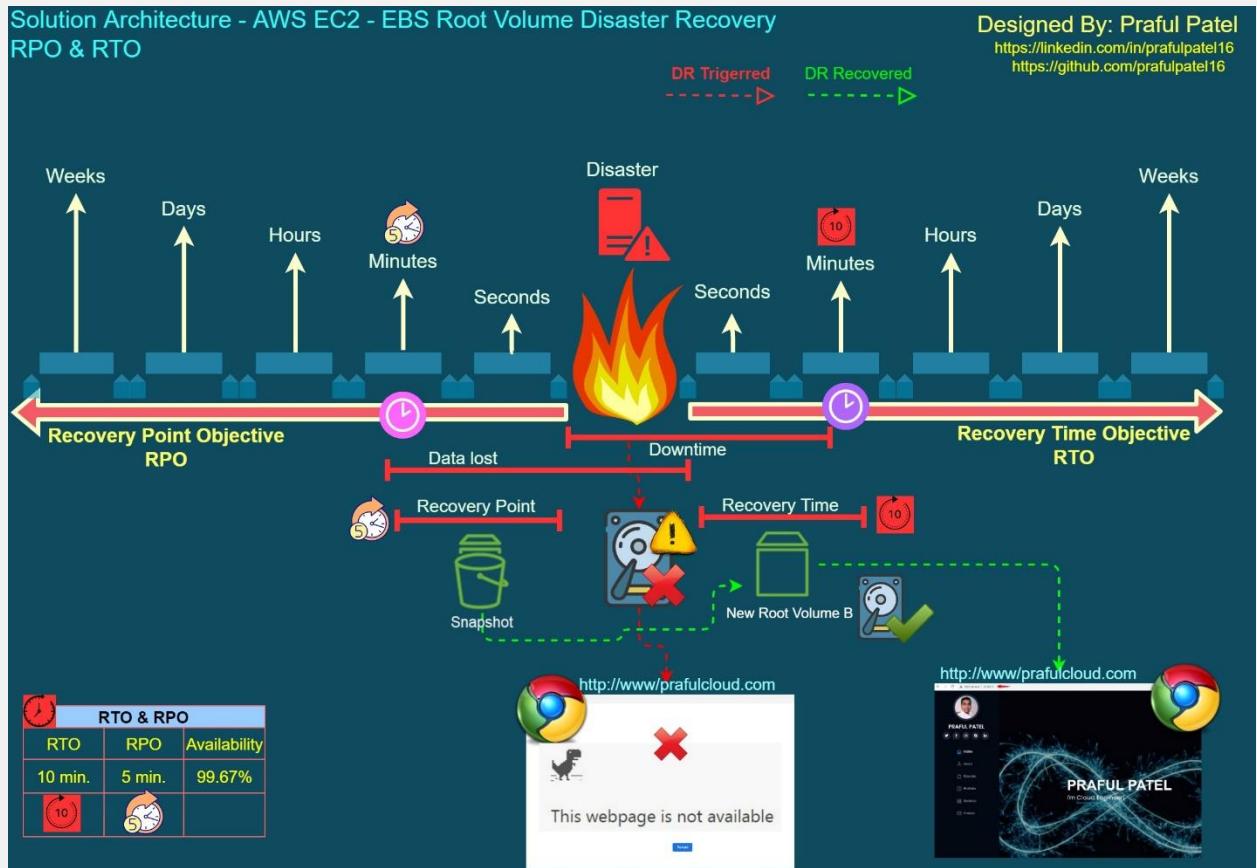
- Failed system status checks
- Incorrect networking or startup configuration
- Exhausted memory
- Corrupted file system
- Incompatible kernel

➤ **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 snapshot 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

The screenshot shows the AWS IAM 'Users' page. A message box at the top says: "New feature to generate a policy based on CloudTrail events. AWS uses your CloudTrail events to identify the services and actions used and generate a least privileged policy that you can attach to this user." The 'test-user' details are shown: User ARN: arn:aws:iam::500942944689:user/test-user, Path: /, Creation time: 2022-06-17 16:19 CST. The 'Permissions' tab is selected, showing one policy applied: 'AmazonEC2FullAccess' (AWS managed policy). Below this, there's a section for generating a policy based on CloudTrail events, with a 'Generate policy' button.

**new\_user\_credentials (3).csv**

	A	B	C	D	E	F	G	H	I	J	K
1	User name	Password	Access key ID	Secret access key	Console login link						
2	test-user		AKIA	DgJxckqlfH	iKohttps://50						
3											
4											
5											

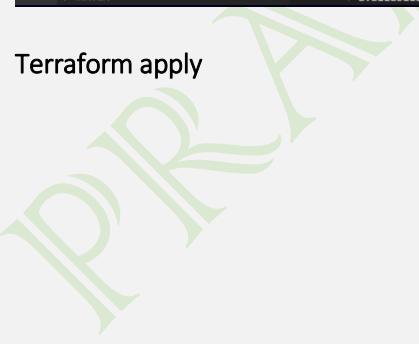
### ➤ Implementation in an Action:

- 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



Screenshot of Visual Studio Code showing the Terraform workspace. The Explorer sidebar shows various projects and files. The main editor pane displays the `main.tf` file:

```
cloud-devops > terraform-project > tf-web_project > main.tf > resource "aws_route_table" "Public_RT" > route
1 terraform {
2   required_providers {
3     aws = {
4       source  = "hashicorp/aws"
5       version = "~> 3.0"
6     }
7   }
8 }
9
10 # Configure the AWS Provider
11 provider "aws" {
12   region = "us-east-1"
13 }
14
15 # Create VPC
```

The terminal tab shows the command `terraform init` being run, resulting in the message "Terraform has been successfully initialized!"

```
- Using previously-installed hashicorp/aws v3.66.0
Terraform has been successfully initialized!
```

The output tab shows the execution plan:

```
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

# aws_instance.Web_server will be created
+ resource "aws_instance" "Web_server" {
    + ami                               = "ami-04902260ca3d33422"
    + arn                               = (known after apply)
    + associate_public_ip_address      = true
    + availability_zone                = (known after apply)
```

## Terraform apply

# AWS PROJECT: AWS-EBS ROOT VOLUME DISASTER RECOVERY, MONITORING & NOTIFICATION SOLUTION DESIGN & IMPLEMENTATION BY: PRAFUL PATEL

**Screenshot 1: Terraform Plan Phase**

```

cloud-devops > terraform-project > tf-web_project > main.tf > resource "aws_route_table" "Public_RT" > route
1   terraform {
2     required_providers {
3       aws = {
4         source  = "hashicorp/aws"
5         version = "~> 3.0"
6       }
7     }
8   }
9

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.

devops@DESKTOP-M660IAH:/mnt/d/cloud-devops/terraform-project$ terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

# aws_instance.Web_server will be created
+ resource "aws_instance" "Web_server" {
  + ami                               = "ami-04982260ca3d33422"
  + arn                             = (known after apply)
  + associate_public_ip_address      = true
  + availability_zone                = (known after apply)
  + cpu_core_count                   = (known after apply)
  + cpu_threads_per_core            = (known after apply)
  + disable_api_termination        = (known after apply)
  + ebs_optimized                   = (known after apply)
  + get_password_data              = false
  + host_id                         = (known after apply)
  + id                             = (known after apply)
  + ...
}

```

**Screenshot 2: Terraform Apply Phase**

```

cloud-devops > terraform-project > tf-web_project > main.tf > resource "aws_route_table" "Public_RT" > route
1   terraform {
2     required_providers {
3       aws = {
4         source  = "hashicorp/aws"
5         version = "~> 3.0"
6       }
7     }
8   }
9

Plan: 7 to add, 0 to change, 0 to destroy.

Changes to Outputs:
+ instance_id          = (known after apply)
+ instance_public_dns  = (known after apply)
+ instance_public_ip   = (known after apply)
+ ip                   = "aws_instance.web_server.public_ip"

Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

aws_vpc.PrafulVPC: Creating...
aws_vpc.PrafulVPC: Still creating... [10s elapsed]
aws_vpc.PrafulVPC: Creation complete after 15s [id=vpc-0aee16cdeb9dadcf19]
aws_internet_gateway.Praful_IGW: Creating...
aws_subnet.PublicSubnet: Creating...
aws_security_group.Public_SG: Creating...
aws_internet_gateway.Praful_IGW: Creation complete after 1s [id=igw-0da440783104ea2fa]
aws_route_table.Public_RT: Creating...
aws_route_table.Public_RT: Still creating... [10s elapsed]
aws_route_table.Public_RT: Creation complete after 2s [id=subnet-074dad35f3ee79387]
aws_route_table_association.a: Creating...
aws_security_group.Public_SG: Creation complete after 4s [id=sg-00778d37e050cf0ef]
aws_instance.Web_server: Creating...
aws_route_table_association.a: Creation complete after 1s [id=rtbassoc-0ac839771d078de89]
aws_instance.Web_server: Still creating... [10s elapsed]

```

Apply complete

## AWS PROJECT: AWS-EBS ROOT VOLUME DISASTER RECOVERY, MONITORING & NOTIFICATION SOLUTION DESIGN & IMPLEMENTATION BY: PRAFUL PATEL

```

1 terraform {
2   required_providers {
3     aws = {
4       source  = "hashicorp/aws"
5       version = "~> 3.0"
6     }
7   }
8 }

```

PROBLEMS    23    OUTPUT    TERMINAL    AZURE    JUPITER    DEBUG CONSOLE

Enter a value: yes

```

aws_vpc.PrafulVPC: Creating...
aws_vpc.PrafulVPC: Still creating... [10s elapsed]
aws_vpc.PrafulVPC: Creation complete after 15s [id=vpc-0ae16cdeb9dadc19]
aws_internet_gateway.Praful_IGW: Creating...
aws_subnet.PublicSubnet: Creating...
aws_security_group.Public_SG: Creating...
aws_internet_gateway.Praful_IGW: Creation complete after 1s [id=igw-0da440703104ea2fa]
aws_route_table.Public_RT: Creating...
aws_subnet.PublicSubnet: Creation complete after 2s [id=subnet-074dad35f3e079387]
aws_route_table.Public_RT: Creation complete after 2s [id=rtb-015bb36f188aa0f7f]
aws_route_table_association.a: Creating...
aws_security_group.Public_SG: Creation complete after 4s [id=sg-00778d37e050cf0ef]
aws_instance.Web_server: Creating...
aws_route_table_association.a: Creation complete after 1s [id=rtbassoc-0ac839771d078de89]
aws_instance.Web_server: Still creating... [10s elapsed]
aws_instance.Web_server: Still creating... [20s elapsed]
aws_instance.Web_server: Still creating... [30s elapsed]
aws_instance.Web_server: Creation complete after 36s [id=i-015f78b741c07a7e4]

Apply complete! Resources: 7 added, 0 changed, 0 destroyed.

Outputs:
```

```

instance_id = "i-015f78b741c07a7e4" 1
instance_public_dns = "ec2-3-238-26-44.compute-1.amazonaws.com" 2
instance_public_ip = "3.238.26.44" 3
ip = "aws_instance.web_server.public_ip" 4

```

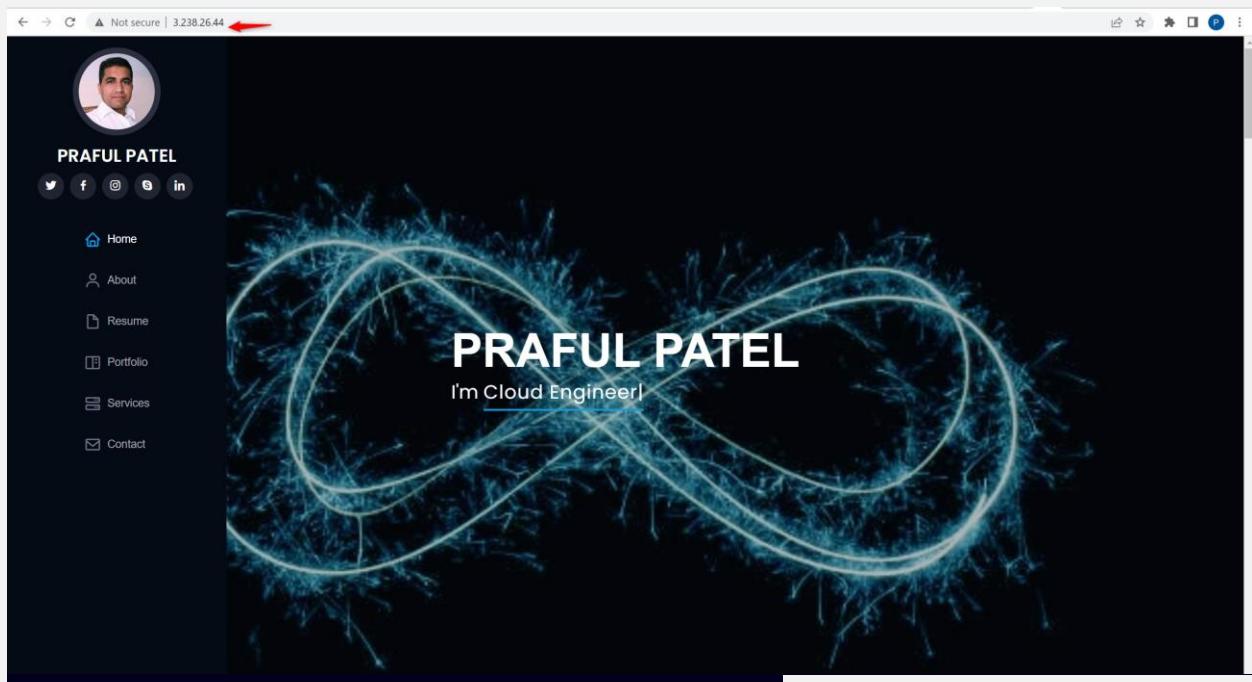
devops@DESKTOP-M660IAH:/mnt/d/cloud-devops/terraform-project/tf-web\_project\$

Verify from aws console if ec2 instance is launched

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4
web_Server	i-015f78b741c07a7e4	Running	t2.micro	Initializing	No alarms	us-east-1f	3.238.26.44

Verify that web application is accessible from browser

AWS PROJECT: AWS-EBS ROOT VOLUME DISASTER RECOVERY, MONITORING & NOTIFICATION  
SOLUTION DESIGN & IMPLEMENTATION BY: PRAFUL PATEL



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

Push source code to github

```
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
```

## AWS PROJECT: AWS-EBS ROOT VOLUME DISASTER RECOVERY, MONITORING & NOTIFICATION SOLUTION DESIGN & IMPLEMENTATION BY: PRAFUL PATEL

```
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

prafulpatel16 / terraform-projects-aws Public

Code Issues Pull requests Actions Projects Wiki Insights Settings

master terraform-projects-aws / tf-web\_project /

prafulpatel16 updated code 40c40b2 4 minutes ago History

..

.gitignore	updated code	4 minutes ago
.terraform.lock.hcl	updated code	4 minutes ago
README.md	updated code	4 minutes ago
app_remove.sh	updated code	4 minutes ago
main.tf	updated code	4 minutes ago
outputs.tf	updated code	4 minutes ago
user-data-apache.sh	updated code	4 minutes ago

README.md

```
cloudwatch alarm trigger: aws cloudwatch set-alarm-state
--alarm-name "Web-Server-SystemFailed"
--state-value ALARM
--state-reason "Simulate an EC2 HW failure"
```

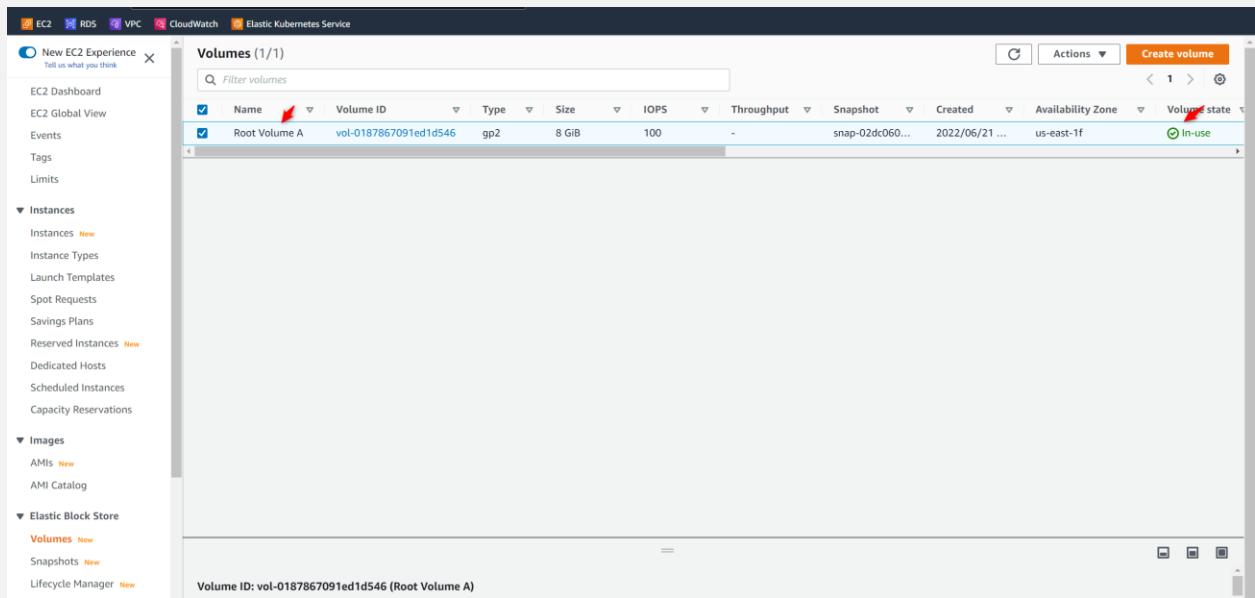
### Phase 2: Take a snapshot of root volume

- Go to snapshots and take snapshot of existing root volume A
- Verify that snapshot process is complete

Volume

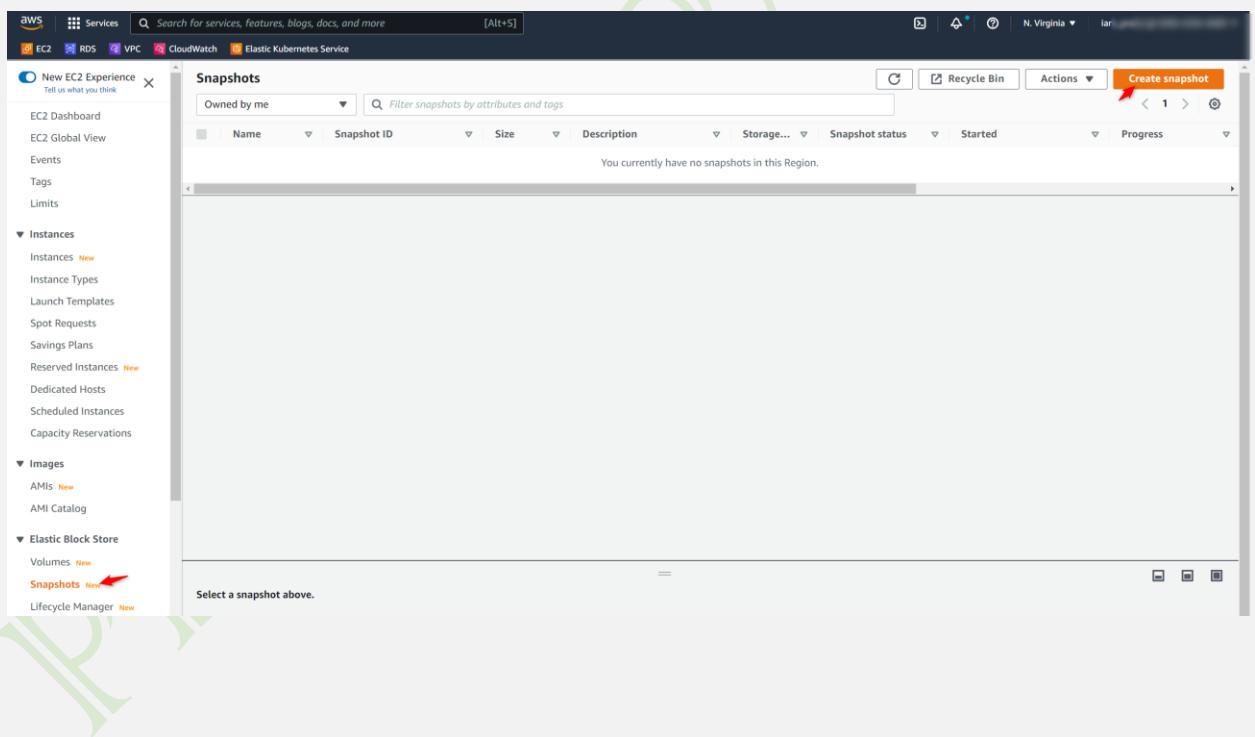
Go to Volume

# AWS PROJECT: AWS-EBS ROOT VOLUME DISASTER RECOVERY, MONITORING & NOTIFICATION SOLUTION DESIGN & IMPLEMENTATION BY: PRAFUL PATEL



The screenshot shows the AWS EC2 Volumes page. The left sidebar includes links for EC2 Dashboard, Instances, Images, and Elastic Block Store (with Volumes and Snapshots). The main content displays a table titled 'Volumes (1/1)' with one entry: 'Root Volume A' (Volume ID: vol-0187867091ed1d546, Type: gp2, Size: 8 GiB, IOPS: 100, Throughput: -, Snapshot: snap-02dc060..., Created: 2022/06/21 ..., Availability Zone: us-east-1f, Volume state: In-use). A red arrow points to the 'Name' column header.

## Take a snapshot



The screenshot shows the AWS EC2 Snapshots page. The left sidebar includes links for EC2 Dashboard, Instances, Images, and Elastic Block Store (with Volumes and Snapshots). The main content displays a table titled 'Snapshots' with the message 'You currently have no snapshots in this Region.' A red arrow points to the 'Schemas' link in the left sidebar.

# AWS PROJECT: AWS-EBS ROOT VOLUME DISASTER RECOVERY, MONITORING & NOTIFICATION SOLUTION DESIGN & IMPLEMENTATION BY: PRAFUL PATEL

**Create snapshot** Info

Create a point-in-time snapshot of an EBS volume and use it as a baseline for new volumes or for data backup. You can create snapshots from an individual volume, or you can create multi-volume snapshots from all of the volumes attached to an instance.

**Snapshot settings**

Resource type Info

**Volume**  
Create a snapshot from a specific volume.

**Instance**  
Create multi-volume snapshots from an instance.

Volume ID  
The volume from which to create the snapshot.

vol-0187867091ed1d546

Q | Select a volume

vol-0187867091ed1d546  
Root Volume A

Encryption Info  
Not encrypted

**Tags** Info

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

No tags associated with the resource.

Add tag

You can add 50 more tags.

**Snapshot settings**

Resource type Info

**Volume**  
Create a snapshot from a specific volume.

**Instance**  
Create multi-volume snapshots from an instance.

Volume ID  
The volume from which to create the snapshot.

vol-0187867091ed1d546

Description  
Add a description for your snapshot.

snapshot of root volume A

255 characters maximum

Encryption Info  
Not encrypted

**Tags** Info

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

Key Value - optional

Q Name X Q web X Remove

Add tag

You can add 49 more tags.

Cancel **Create snapshot**

**Snapshot complete**

## AWS PROJECT: AWS-EBS ROOT VOLUME DISASTER RECOVERY, MONITORING & NOTIFICATION SOLUTION DESIGN & IMPLEMENTATION BY: PRAFUL PATEL

The screenshot shows the AWS CloudWatch Snapshots console. On the left, there's a navigation sidebar with links for EC2, RDS, VPC, CloudWatch, and Elastic Kubernetes Service. Under the 'Solutions' section, there's a link to 'AWS EBS Root Volume Disaster Recovery'. The main area displays a table titled 'Snapshots (1)'. The table has columns for Name, Snapshot ID, Size, Description, Storage Type, Started, Progress, and Status. The single entry is 'snapshot-root-volume-A' with Snapshot ID 'snap-0b486efaa5b4e2655', 8 GiB size, and a description 'snapshot of root volume A'. The status is 'Completed' with a green checkmark, and the progress is 'Available (100%)'. A red arrow points to the 'Name' column header.

OR

- ➊ 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

The screenshot shows a Visual Studio Code workspace titled 'cloud-devops (Workspace)'. The Explorer sidebar shows files like 'playbook.yml', 'ec2\_snapshot.yml' (which is currently selected and highlighted with a red arrow), 'README.md', 'iam\_group\_csv.yml', and 'iam\_group.yml'. The 'ec2\_snapshot.yml' file is open in the editor, displaying an Ansible playbook. The first few lines of the code are:

```

# Type: Ansible Task
# Purpose: Take a snapshot of EC2 root volume
# Date & Time: June 22, 2022
# Author: Praful Patel
# -----
- name: Creating EC2 root volume snapshot
  ec2_snapshot:
    aws_region: us-east-1
    instance_id: i-083f1f58ef33d05d
    device_name: /dev/xvda
    description: snapshot of /web-server data taken 2013/11/22 12:18:32
    snapshot_tags:
      frequency: hourly
      source: /root_volume_data

```

Gather the instance and volume information manually

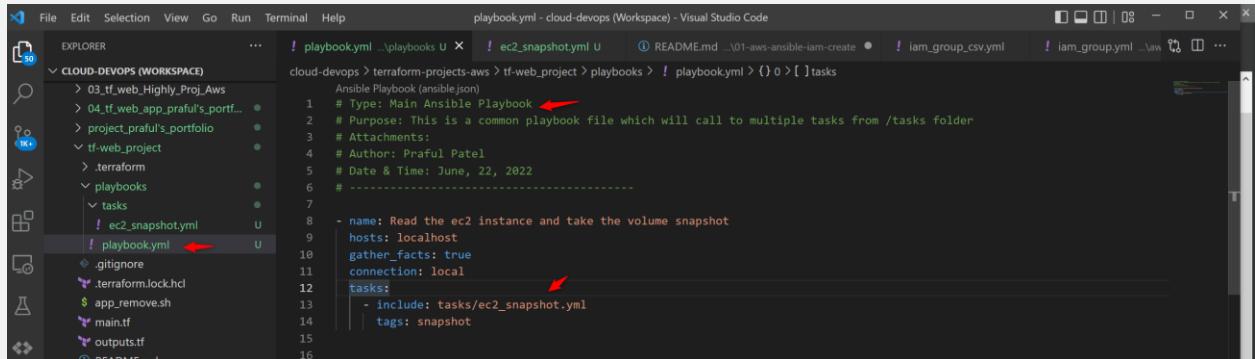
**aws\_region:**

**Instance id:**

# AWS PROJECT: AWS-EBS ROOT VOLUME DISASTER RECOVERY, MONITORING & NOTIFICATION SOLUTION DESIGN & IMPLEMENTATION BY: PRAFUL PATEL

Device\_name:

Write main snapshot yaml file

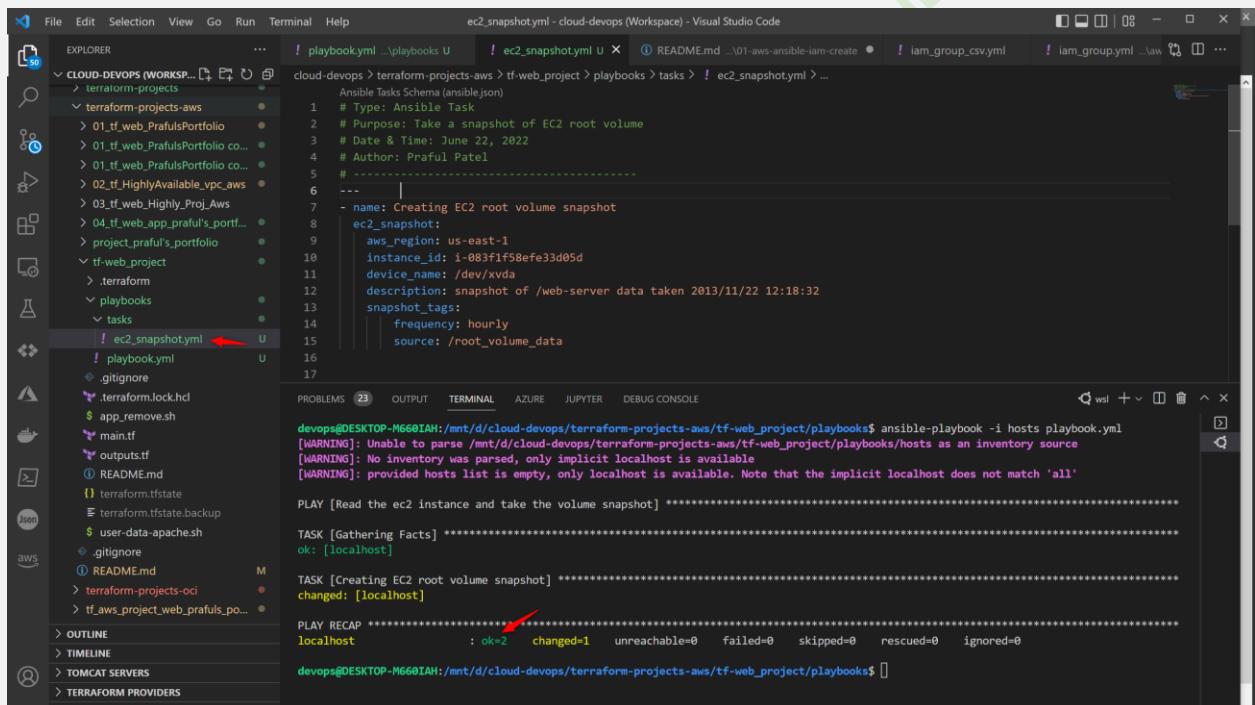


```

File Edit Selection View Go Run Terminal Help
playbook.yml - cloud-devops (Workspace) - Visual Studio Code
cloud-devops > terraform-project-aws > tf-web_project > playbooks > playbook.yml > {} 0 [ ] tasks
    Ansible Playbook (ansible.json)
    1 # Type: Main Ansible Playbook
    2 # Purpose: This is a common playbook file which will call to multiple tasks from /tasks folder
    3 # Attachments:
    4 # Author: Praful Patel
    5 # Date & Time: June, 22, 2022
    6 #
    7
    8 - name: Read the ec2 instance and take the volume snapshot
    9 hosts: localhost
   10 gather_facts: true
   11 connection: local
   12 tasks:
   13     - include: tasks/ec2_snapshot.yml
   14         tags: snapshot

```

Run Ansible playbook file



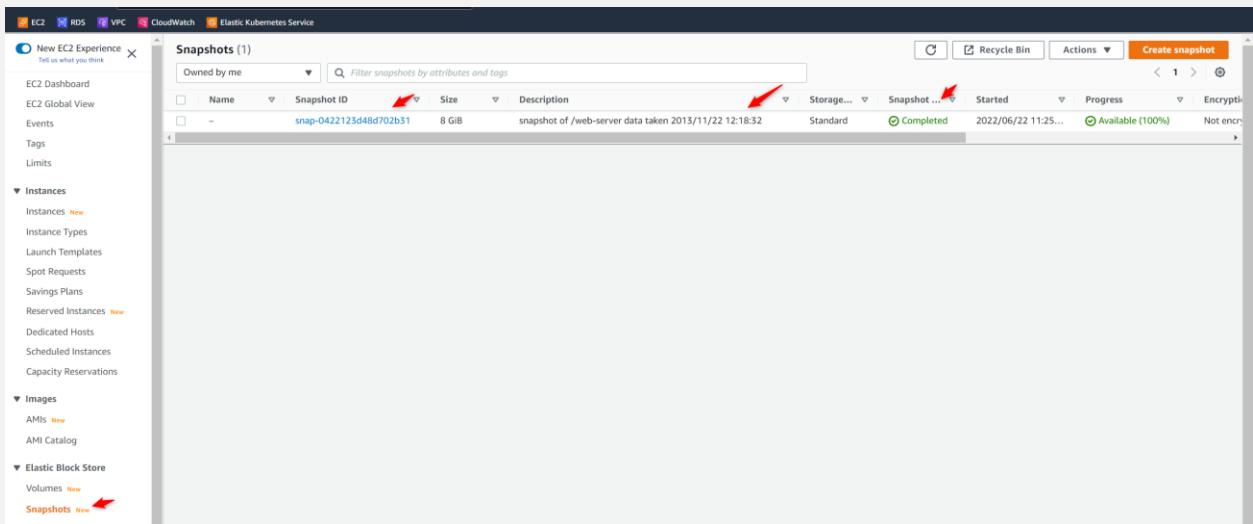
```

File Edit Selection View Go Run Terminal Help
ec2_snapshot.yml - cloud-devops (Workspace) - Visual Studio Code
cloud-devops > terraform-project-aws > tf-web_project > playbooks > tasks > ec2_snapshot.yml > ...
Ansible Tasks Schema (ansible.json)
1 # Type: Ansible Task
2 # Purpose: Take a snapshot of EC2 root volume
3 # Date & Time: June 22, 2022
4 # Author: Praful Patel
5 #
6 --- | ...
7 - name: Creating EC2 root volume snapshot
8   ec2_snapshot:
9     aws_region: us-east-1
10    instance_id: i-083f1f58ef8e33d05d
11    device_name: /dev/xvda
12    description: snapshot of /web-server data taken 2013/11/22 12:18:32
13    snapshot_tags:
14      frequency: hourly
15      source: /root_volume_data
16
17
PROBLEMS 23 OUTPUT TERMINAL AZURE JUPYTER DEBUG CONSOLE
devops@DESKTOP-M660IAH:/mnt/d/cloud-devops/terraform-project-aws/tf-web_project/playbooks$ ansible-playbook -i hosts playbook.yml
[WARNING]: Unable to parse /mnt/d/cloud-devops/terraform-project-aws/tf-web_project/playbooks/hosts as an inventory source
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost does not match 'all'
PLAY [Read the ec2 instance and take the volume snapshot] ****
TASK [Gathering Facts] ****
ok: [localhost]
TASK [Creating EC2 root volume snapshot] ****
changed: [localhost]
PLAY RECAP ****
localhost : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
devops@DESKTOP-M660IAH:/mnt/d/cloud-devops/terraform-project-aws/tf-web_project/playbooks$ 

```

Verify from the AWS console that snapshot has been created.

# AWS PROJECT: AWS-EBS ROOT VOLUME DISASTER RECOVERY, MONITORING & NOTIFICATION SOLUTION DESIGN & IMPLEMENTATION BY: PRAFUL PATEL



- 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

# AWS PROJECT: AWS-EBS ROOT VOLUME DISASTER RECOVERY, MONITORING & NOTIFICATION SOLUTION DESIGN & IMPLEMENTATION BY: PRAFUL PATEL

The figure consists of three vertically stacked screenshots of the AWS CloudWatch Metrics interface.

**Screenshot 1: CloudWatch Alarms Overview**

This screenshot shows the CloudWatch Alarms overview page. The top navigation bar includes links for EC2, RDS, VPC, CloudWatch, and Elastic Kubernetes Service. The left sidebar has sections for Favorites and recent dashboards, Alarms (with 0 alarms), Logs (Log groups, Logs Insights), and Metrics. The main area displays a table titled "Alarms (0)" with columns for Name, State, Last state update, Conditions, and Actions. A red arrow points to the "Create alarm" button in the top right corner.

**Screenshot 2: Create alarm - Step 1: Specify metric and conditions**

This screenshot shows the first step of creating a new alarm. It's titled "Specify metric and conditions". On the left, a sidebar lists steps: Step 1 (Specify metric and conditions), Step 2 (Configure actions), Step 3 (Add name and description), and Step 4 (Preview and create). The main area is titled "Metric" and contains a "Graph" section with a "Select metric" button, which is highlighted with a red arrow. Buttons for "Cancel" and "Next" are at the bottom right.

**Screenshot 3: Select metric**

This screenshot shows the "Select metric" dialog. It features a graph preview with a single line at value 1. The x-axis shows time from 17:45 to 20:30. Below the graph, it says "Your CloudWatch graph is empty. Select some metrics to appear here." At the bottom, there are tabs for "Browse", "Query", "Graphed metrics" (which is selected), "Options", and "Source". A search bar at the top says "Search for any metric, dimension or resource id". Below the tabs, a table titled "Metrics (384)" lists various metrics with their counts: EBS (63), EC2 (102), Events (6), Lambda (16), Logs (9), S3 (8), SNS (4), and Usage (176). A red arrow points to the "EC2" metric in the first row.

# AWS PROJECT: AWS-EBS ROOT VOLUME DISASTER RECOVERY, MONITORING & NOTIFICATION SOLUTION DESIGN & IMPLEMENTATION BY: PRAFUL PATEL

The screenshot shows the AWS CloudWatch Metrics console interface. At the top, there are navigation links for EC2, RDS, VPC, CloudWatch, and Elastic Kubernetes Service. Below the navigation bar, a search bar contains the text "Select metric". A message in the center of the screen says "Untitled graph" and "Your CloudWatch graph is empty. Select some metrics to appear here." Below this, there are tabs for "Browse", "Query", "Graphed metrics", "Options", and "Source". A "Metrics (102)" section shows a search bar with "All > EC2" and a search input. A red arrow points to the "Per-Instance Metrics" link. To the right of the search bar, there are buttons for "Graph with SQL" and "Graph search". At the bottom of the metrics list, there is a table with columns: "Instance name (102)", "InstanceId", and "Metric name". One row is selected, highlighted with a blue border and a circled '1'. The "Metric name" column for this row shows "StatusCheckFailed\_System". A red arrow points to this specific metric name. In the bottom right corner of the metrics table, there is a "Select metric" button.

# AWS PROJECT: AWS-EBS ROOT VOLUME DISASTER RECOVERY, MONITORING & NOTIFICATION SOLUTION DESIGN & IMPLEMENTATION BY: PRAFUL PATEL

**Specify metric and conditions**

**Metric**

Graph: This alarm will trigger when the blue line goes above the red line for 1 datapoints within 1 minute.

Count: 1

Namespace: AWS/EC2

Metric name: StatusCheckFailed\_System

InstanceId: i-015f78b741c07a7e4

Instance name: web\_Server

Statistic: Average

Period: 1 minute

**Conditions**

Period: 1 minute

Threshold type: Static (1)

Anomaly detection: Use a band as a threshold

Whenever StatusCheckFailed\_System is... (2)

Greater than threshold: > threshold (3)

Greater/Equal: >= threshold (4)

Lower/Equal: <= threshold

Lower than threshold: < threshold

than... Define the threshold value: 80 (3)

Additional configuration

Datapoints to alarm: Define the number of datapoints within the evaluation period that must be breaching to cause the alarm to go to ALARM state. (4)

Missing data treatment: How to treat missing data when evaluating the alarm. Treat missing data as missing (5)

Cancel Next

Create a New SNS Topic

# AWS PROJECT: AWS-EBS ROOT VOLUME DISASTER RECOVERY, MONITORING & NOTIFICATION SOLUTION DESIGN & IMPLEMENTATION BY: PRAFUL PATEL

**Configure actions**

**Notification**

Alarm state trigger  
Define the alarm state that will trigger this action.

**1**  In alarm  
The metric or expression is outside of the defined threshold.

OK  
The metric or expression is within the defined threshold.

Insufficient data  
The alarm has just started or not enough data is available.

Remove

Send a notification to the following SNS topic  
Define the SNS (Simple Notification Service) topic that will receive the notification.

Select an existing SNS topic

**2**  Create new topic

Use topic ARN to notify other accounts

Create a new topic...  
The topic name must be unique.

**3** WebServer\_CloudWatch\_Alarms\_Topic

SNS topic names can contain only alphanumeric characters, hyphens (-) and underscores (\_).

Email endpoints that will receive the notification...  
Add a comma-separated list of email addresses. Each address will be added as a subscription to the topic above.

**4** aysterer1@gmail.com  
user1@example.com, user2@example.com

**5** Create topic  
Add notification

Go to EC2 Action

Choose: Recover this Instance

**EC2 action**

Alarm state trigger  
Define the alarm state that will trigger this action.

**1**  In alarm  
The metric or expression is outside of the defined threshold.

OK  
The metric or expression is within the defined threshold.

Insufficient data  
The alarm has just started or not enough data is available.

Remove

Take the following action...  
Define what will happen to the EC2 instance with the Instance ID i-015f78b741c07a7e4 when this alarm is triggered.

**Recover this instance**  
You can only recover certain EC2 instance types. See documentation

Stop this instance  
You can only stop an instance if it is backed by an EBS volume. AWS will use the existing Service Linked Role (AWSServiceRoleForCloudWatchEvents) to perform this action. Show IAM policy document

Terminate this instance  
You will not be able to terminate this instance if termination protection is enabled. AWS will use the existing Service Linked Role (AWSServiceRoleForCloudWatchEvents) to perform this action. Show IAM policy document

**2**  Reboot this instance  
An instance reboot is equivalent to an operating system reboot. AWS will use the existing Service Linked Role (AWSServiceRoleForCloudWatchEvents) to perform this action. Show IAM policy document

Add EC2 action

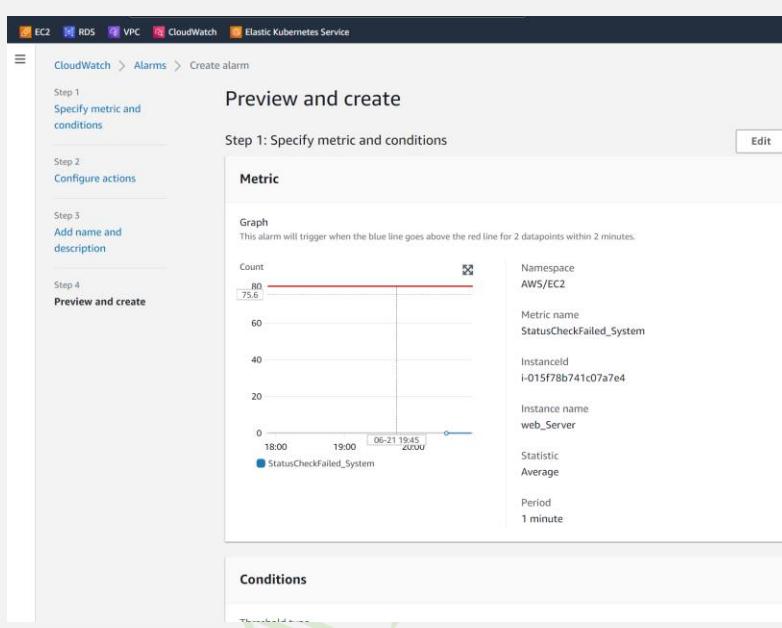
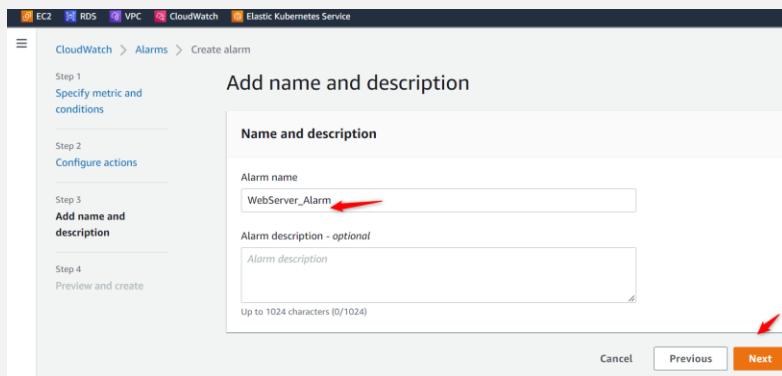
**Systems Manager action** Info

This action will create an Incident or Opsitem in Systems Manager when the alarm is **In alarm** state.

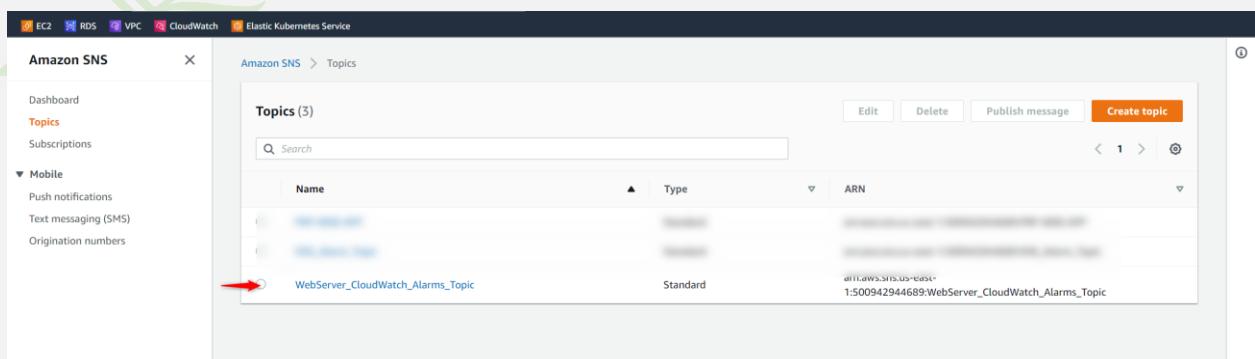
Add Systems Manager action

Cancel Previous Next **3**

# AWS PROJECT: AWS-EBS ROOT VOLUME DISASTER RECOVERY, MONITORING & NOTIFICATION SOLUTION DESIGN & IMPLEMENTATION BY: PRAFUL PATEL



Go to SNS Service to confirm



# AWS PROJECT: AWS-EBS ROOT VOLUME DISASTER RECOVERY, MONITORING & NOTIFICATION SOLUTION DESIGN & IMPLEMENTATION BY: PRAFUL PATEL

The screenshot shows the AWS SNS console. On the left, there's a navigation sidebar with links like EC2, RDS, VPC, CloudWatch, and Elastic Kubernetes Service. Under the 'Topics' section, 'WebServer\_CloudWatch\_Alarms\_Topic' is selected. The main panel displays the 'Details' for this topic, including its name, ARN, and type. Below this, the 'Subscriptions' tab is active, showing one subscription entry:

ID	Endpoint	Status	Protocol
Pending confirmation	aystester1@gmail.com	Pending confirmation	EMAIL

A red arrow points to the 'Pending confirmation' status of the subscription.

## Confirm Subscription

Go to Gmail and grab the url

The screenshot shows a Gmail inbox with 2,069 messages. An email from 'AWS Notifications <no-reply@sns.amazonaws.com>' is selected. The subject is 'AWS Notification - Subscription Confirmation'. The email body contains the following text:

You have chosen to subscribe to the topic:  
arn:aws:sns:us-east-1:500942944689:WebServer\_CloudWatch\_Alarms\_Topic

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

Please do not reply directly to this email. If you wish to remove yourself from receiving all future SNS subscription confirmation requests please send an email to [sns-optout](#)

A red arrow points to the 'Confirm subscription' link.

Click to confirm subscription

The screenshot shows a confirmation page for an AWS Simple Notification Service (SNS) subscription. The title is 'Simple Notification Service'. The main message is 'Subscription confirmed!'. It states: 'You have successfully subscribed.' and 'Your subscription's id is: arn:aws:sns:us-east-1:500942944689:WebServer\_CloudWatch\_Alarms\_Topic:10503112-7fc6-416d-ab19-1bae802ca5d2'. At the bottom, it says 'If it was not your intention to subscribe, [click here to unsubscribe](#)'.

# AWS PROJECT: AWS-EBS ROOT VOLUME DISASTER RECOVERY, MONITORING & NOTIFICATION SOLUTION DESIGN & IMPLEMENTATION BY: PRAFUL PATEL

**Amazon SNS**

**Topics**

**Subscriptions**

**Mobile**

**Push notifications**

**Text messaging (SMS)**

**Origination numbers**

**Details**

**Name:** WebServer\_CloudWatch\_Alarms\_Topic

**ARN:** arn:aws:sns:us-east-1:500942944689:WebServer\_CloudWatch\_Alarms\_Topic

**Type:** Standard

**Display name:** -

**Topic owner:** 500942944689

**Subscriptions (1)**

ID	Endpoint	Status	Protocol
10503112-7fc6-416d-ab19-1bae802ca5d2	aystest1@gmail.com	Confirmed	EMAIL

**Access policy** | **Delivery retry policy (HTTP/S)** | **Delivery status logging** | **Encryption** | **Tags**

Go to Cloudwatch and observe the Alarm status

**CloudWatch**

**Alarms**

**All alarms**

**Logs**

**Metrics**

**CloudWatch**

**Alarms**

**All alarms**

**Logs**

**Metrics**

**CloudWatch**

**Alarms**

**WebServer\_Alarm**

**Graph**

**StatusCheckFailed\_System**

StatusCheckFailed\_System >= 80 for 2 datapoints within 2 minutes

Count: 80

18:00 18:15 18:30 18:45 19:00 19:15 19:30 19:45 20:00 20:15 20:30 20:45 21:00

StatusCheckFailed\_System

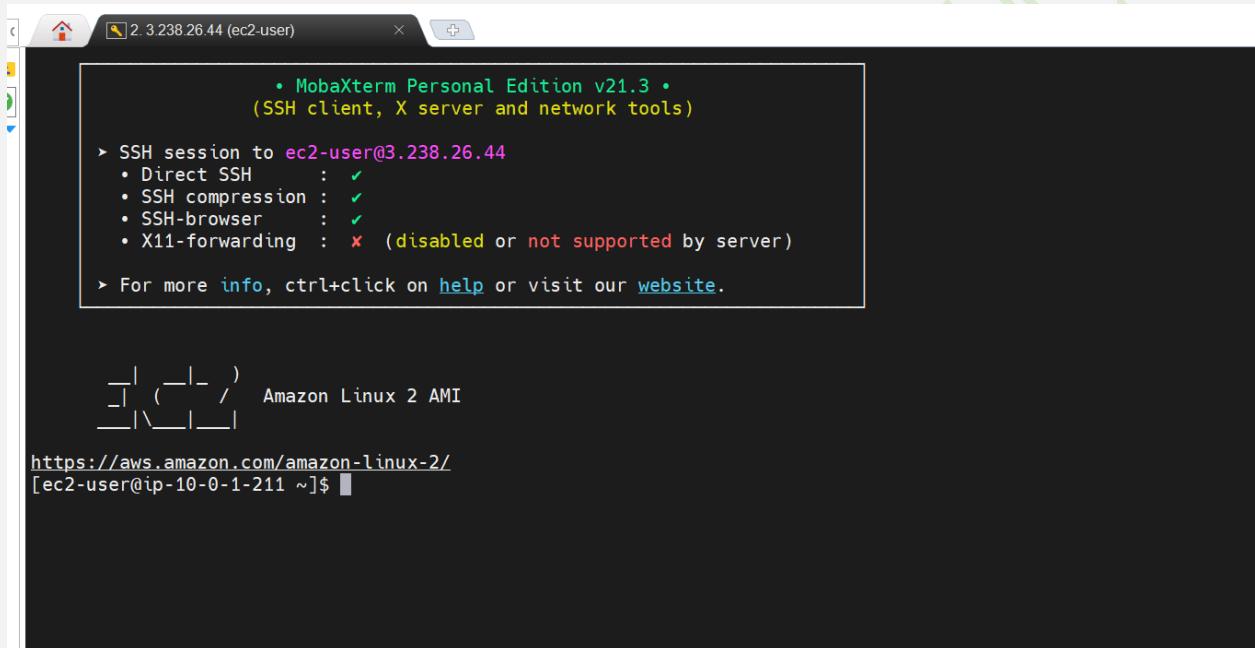
● In alarm ● OK ● Insufficient data

**Actions**

**Details**

- 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



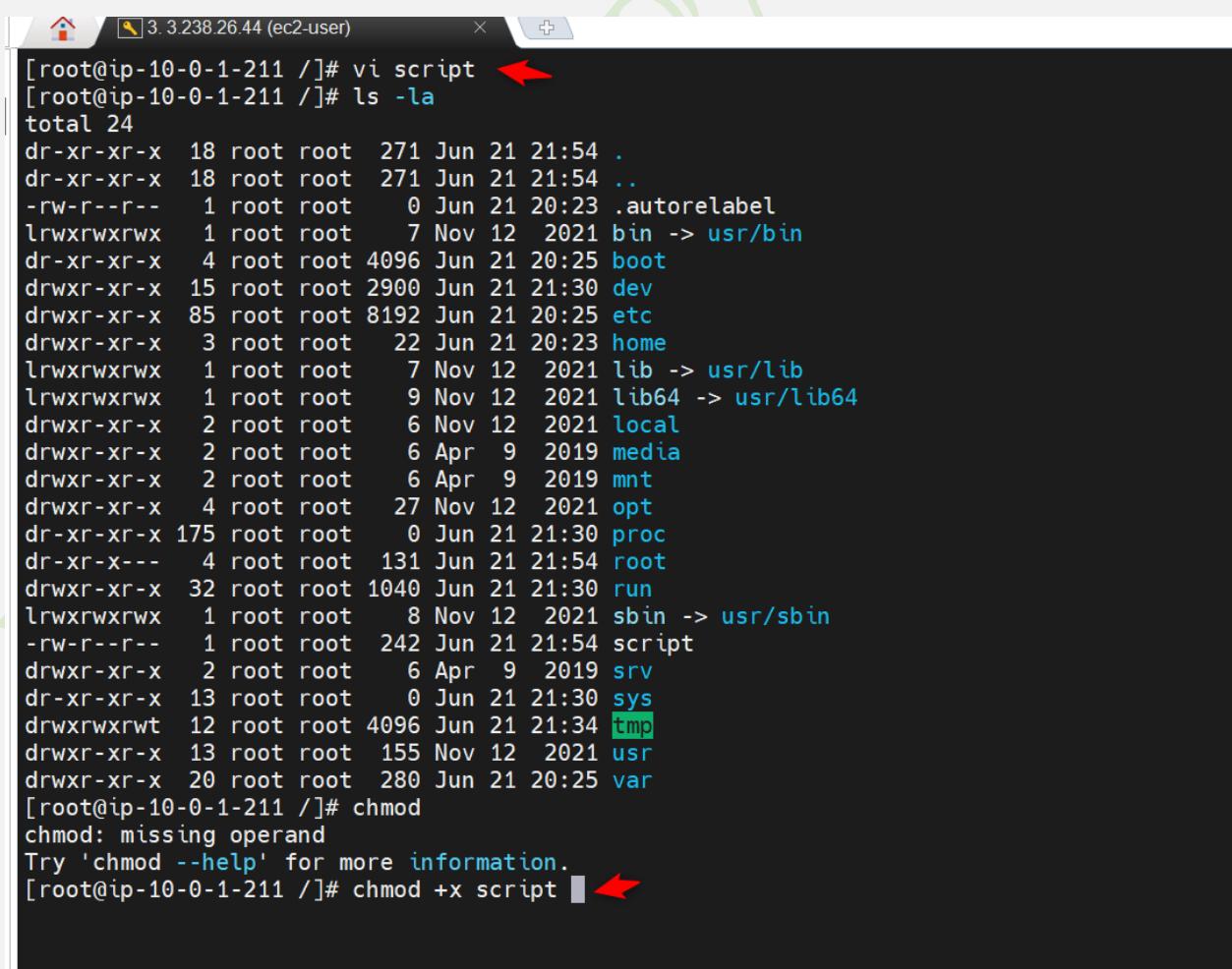
```
• MobaXterm Personal Edition v21.3 •  
(SSH client, X server and network tools)  
► SSH session to ec2-user@3.238.26.44  
• Direct SSH : ✓  
• SSH compression : ✓  
• SSH-browser : ✓  
• X11-forwarding : ✘ (disabled or not supported by server)  
► For more info, ctrl+click on help or visit our website.  
  
_ _|_ _|_ / Amazon Linux 2 AMI  
_ _|_ _|_|  
https://aws.amazon.com/amazon-linux-2/  
[ec2-user@ip-10-0-1-211 ~]$
```

AWS configure with new user into EC2 machine

Aws configure

```
• MobaXterm Personal Edition v21.3 •  
(SSH client, X server and network tools)  
  
➤ SSH session to ec2-user@3.238.26.44  
• Direct SSH : ✓  
• SSH compression : ✓  
• SSH-browser : ✓  
• X11-forwarding : ✘ (disabled or not supported by server)  
  
➤ For more info, ctrl+click on help or visit our website.  
  
_ _|_ _|_ / Amazon Linux 2 AMI  
_ _\_|_ _|  
  
https://aws.amazon.com/amazon-linux-2/  
[ec2-user@ip-10-0-1-211 ~]$ aws configure ①  
AWS Access Key ID [None]: AKI ②  
AWS Secret Access Key [None]: DgJx ③  
Default region name [None]: us-east-1 ④  
Default output format [None]:  
[ec2-user@ip-10-0-1-211 ~]$
```

Simulate Web Server app failure by removing application code:



The screenshot shows a terminal window titled "3.238.26.44 (ec2-user)". The terminal content is as follows:

```
[root@ip-10-0-1-211 /]# vi script ←  
[root@ip-10-0-1-211 /]# ls -la  
total 24  
dr-xr-xr-x 18 root root 271 Jun 21 21:54 .  
dr-xr-xr-x 18 root root 271 Jun 21 21:54 ..  
-rw-r--r-- 1 root root 0 Jun 21 20:23 .autorelabel  
lrwxrwxrwx 1 root root 7 Nov 12 2021 bin -> usr/bin  
dr-xr-xr-x 4 root root 4096 Jun 21 20:25 boot  
drwxr-xr-x 15 root root 2900 Jun 21 21:30 dev  
drwxr-xr-x 85 root root 8192 Jun 21 20:25 etc  
drwxr-xr-x 3 root root 22 Jun 21 20:23 home  
lrwxrwxrwx 1 root root 7 Nov 12 2021 lib -> usr/lib  
lrwxrwxrwx 1 root root 9 Nov 12 2021 lib64 -> usr/lib64  
drwxr-xr-x 2 root root 6 Nov 12 2021 local  
drwxr-xr-x 2 root root 6 Apr 9 2019 media  
drwxr-xr-x 2 root root 6 Apr 9 2019 mnt  
drwxr-xr-x 4 root root 27 Nov 12 2021 opt  
dr-xr-xr-x 175 root root 0 Jun 21 21:30 proc  
dr-xr-x--- 4 root root 131 Jun 21 21:54 root  
drwxr-xr-x 32 root root 1040 Jun 21 21:30 run  
lrwxrwxrwx 1 root root 8 Nov 12 2021 sbin -> usr/sbin  
-rw-r--r-- 1 root root 242 Jun 21 21:54 script  
drwxr-xr-x 2 root root 6 Apr 9 2019 srv  
dr-xr-xr-x 13 root root 0 Jun 21 21:30 sys  
drwxrwxrwt 12 root root 4096 Jun 21 21:34 tmp  
drwxr-xr-x 13 root root 155 Nov 12 2021 usr  
drwxr-xr-x 20 root root 280 Jun 21 20:25 var  
[root@ip-10-0-1-211 /]# chmod  
chmod: missing operand  
Try 'chmod --help' for more information.  
[root@ip-10-0-1-211 /]# chmod +x script ←
```

### Create an app remove script

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows files in the "CLOUD-DEVOPS (WORKSPACE)" folder, including "04\_tf\_web\_Portfolio copy", "05\_tf\_web\_Portfolio\_Single", "tf-web\_project", ".terraform", ".terraform.lock.hcl", "app\_remove.sh" (highlighted with a red arrow), "main.tf", "outputs.tf", "README.md", "terraform.tfstate", "terraform.tfstate.backup", and "user-data-apache.sh".
- Code Editor:** Displays the content of "app\_remove.sh" with the following code:

```

#!/bin/bash
# Purpose: To remove all files and folders from the web server root directory
# Author: Praful Patel
# Date & Time: June 21, 2022
# -----
rm -rfv /var/www/html/*
cd /var/www/html/
ls -la
echo "FILE and FOLDER REMOVED SUCCESSFULLY FROM ROOT DIRECTORY"

```
- Status Bar:** Shows file names and status for "main.tf", "README.md", "app\_remove.sh", and "user-data-apache.sh".

### Run the script

The screenshot shows a terminal window with the following session:

```

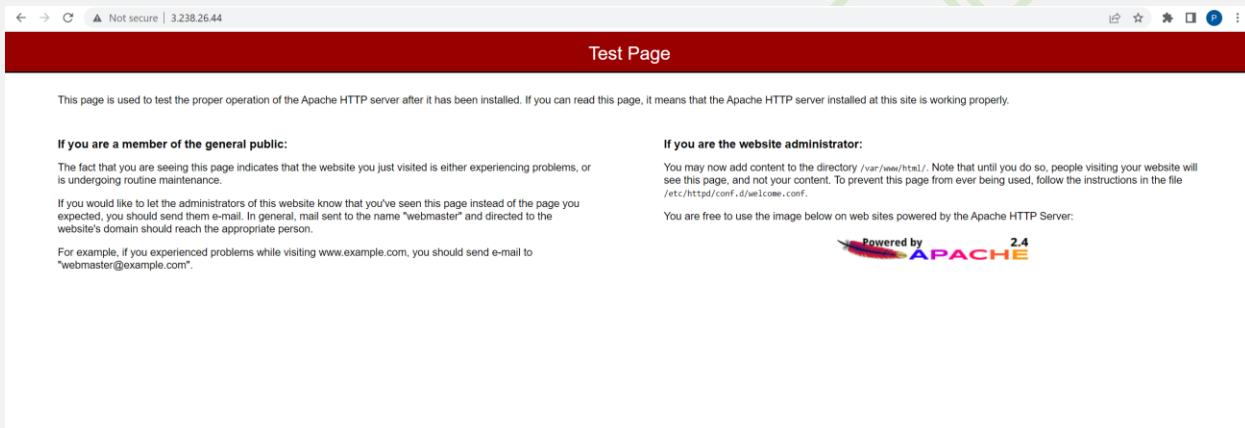
[root@ip-10-0-1-211 ~]# vi script
[root@ip-10-0-1-211 ~]# clear
[root@ip-10-0-1-211 ~]# ./script
removed '/var/www/html/Readme.txt'
removed '/var/www/html/assets/css/style.css'
removed directory: '/var/www/html/assets/css'
removed '/var/www/html/assets/img/apple-touch-icon.png'
removed '/var/www/html/assets/img/favicon.png'
removed '/var/www/html/assets/img/hero-bg.jpg'
removed '/var/www/html/assets/img/portfolio/portfolio-1.jpg'
removed '/var/www/html/assets/img/portfolio/portfolio-2.jpg'
removed '/var/www/html/assets/img/portfolio/portfolio-3.jpg'
removed '/var/www/html/assets/img/portfolio/portfolio-4.jpg'
removed '/var/www/html/assets/img/portfolio/portfolio-5.jpg'
removed '/var/www/html/assets/img/portfolio/portfolio-6.jpg'
removed '/var/www/html/assets/img/portfolio/portfolio-7.jpg'
removed '/var/www/html/assets/img/portfolio/portfolio-8.jpg'
removed '/var/www/html/assets/img/portfolio/portfolio-9.jpg'
removed directory: '/var/www/html/assets/img/portfolio'
removed '/var/www/html/assets/img/profile-img.jpg'
removed '/var/www/html/assets/img/testimonials/testimonials-1.jpg'
removed '/var/www/html/assets/img/testimonials/testimonials-2.jpg'
removed '/var/www/html/assets/img/testimonials/testimonials-3.jpg'
removed '/var/www/html/assets/img/testimonials/testimonials-4.jpg'
removed '/var/www/html/assets/img/testimonials/testimonials-5.jpg'
removed directory: '/var/www/html/assets/img/testimonials'
removed directory: '/var/www/html/assets/img'
removed '/var/www/html/assets/js/main.js'
removed directory: '/var/www/html/assets/js'
removed '/var/www/html/assets/vendor/aos/aos.css'
removed '/var/www/html/assets/vendor/aos/aos.js'
removed directory: '/var/www/html/assets/vendor/aos'
removed '/var/www/html/assets/vendor/bootstrap-icons/bootstrap-icons.css'
removed '/var/www/html/assets/vendor/bootstrap-icons/bootstrap-icons.json'
removed '/var/www/html/assets/vendor/bootstrap-icons/fonts/bootstrap-icons.woff'
removed '/var/www/html/assets/vendor/bootstrap-icons/fonts/bootstrap-icons.woff2'
removed directory: '/var/www/html/assets/vendor/bootstrap-icons/fonts'
removed '/var/www/html/assets/vendor/bootstrap-icons/index.html'
removed directory: '/var/www/html/assets/vendor/bootstrap-icons'
removed '/var/www/html/assets/vendor/bootstrap/css/bootstrap-grid.css'
removed '/var/www/html/assets/vendor/bootstrap/css/bootstrap-grid.css.map'
removed '/var/www/html/assets/vendor/bootstrap/css/bootstrap-grid.min.css'
removed '/var/www/html/assets/vendor/bootstrap/css/bootstrap-grid.min.css.map'
removed '/var/www/html/assets/vendor/bootstrap/css/bootstrap-grid rtl.css'
removed '/var/www/html/assets/vendor/bootstrap/css/bootstrap-grid rtl.css.map'
removed '/var/www/html/assets/vendor/bootstrap/css/bootstrap-grid rtl.min.css'

```

```
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 \
```

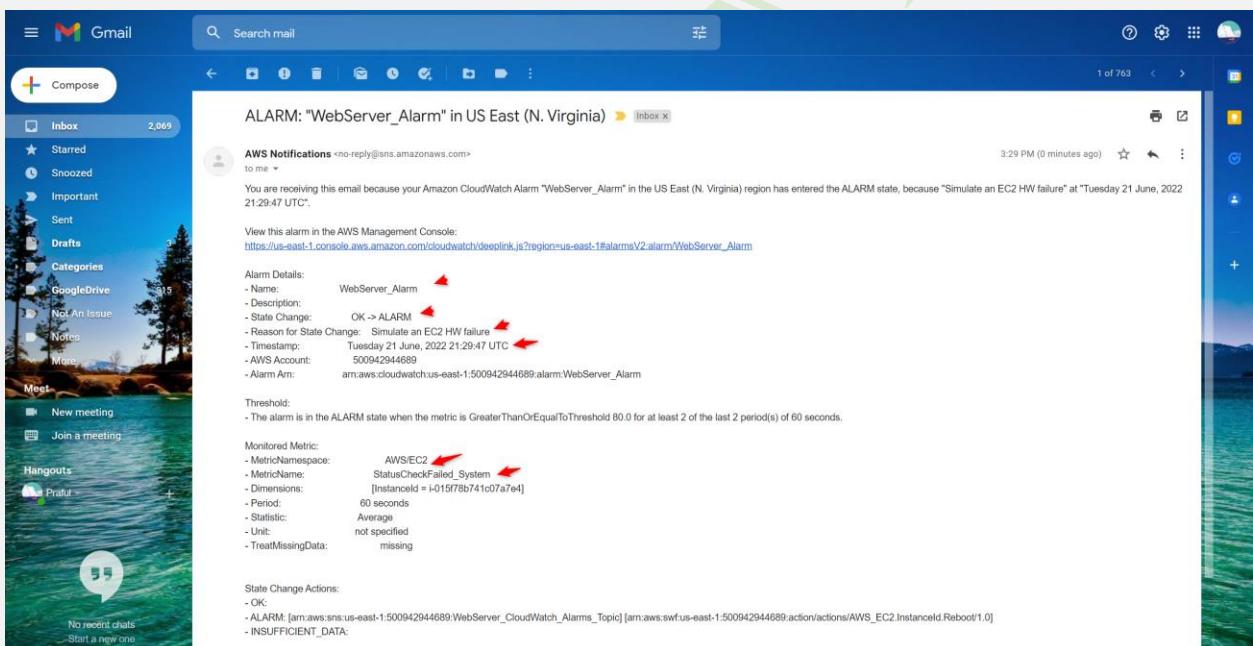
## AWS PROJECT: AWS-EBS ROOT VOLUME DISASTER RECOVERY, MONITORING & NOTIFICATION SOLUTION DESIGN & IMPLEMENTATION BY: PRAFUL PATEL

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

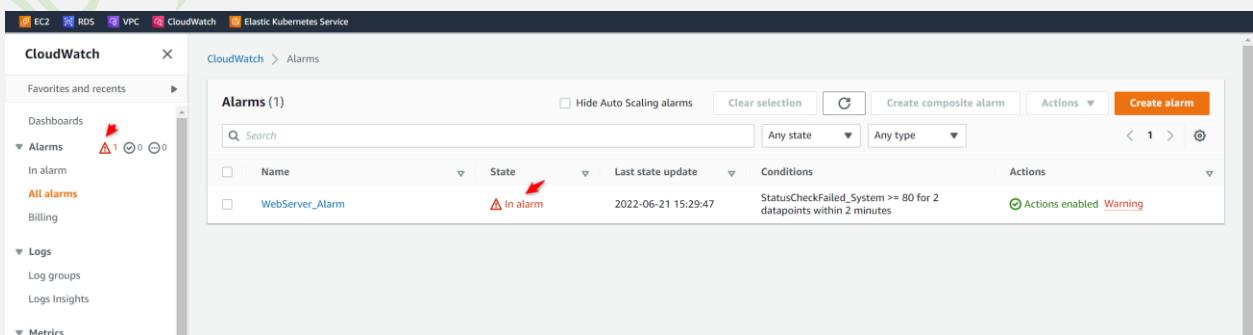
```
[root@ip-10-0-1-211 ~]# aws cloudwatch set-alarm-state \
> --alarm-name "WebServer_Alarm" \
> --state-value ALARM \
> --state-reason "Simulate an EC2 HW failure"
[root@ip-10-0-1-211 ~]#
Session terminated, killing shell... ...killed.
Terminated
[ec2-user@ip-10-0-1-211 ~]$  
Remote side unexpectedly closed network connection

k c [root@ip-10-0-1-211 ~]# aws cloudwatch set-alarm-state \
> --alarm-name "WebServer_Alarm" \
> --state-value ALARM \
> --state-reason "Simulate an EC2 HW failure"
[root@ip-10-0-1-211 ~]#
```

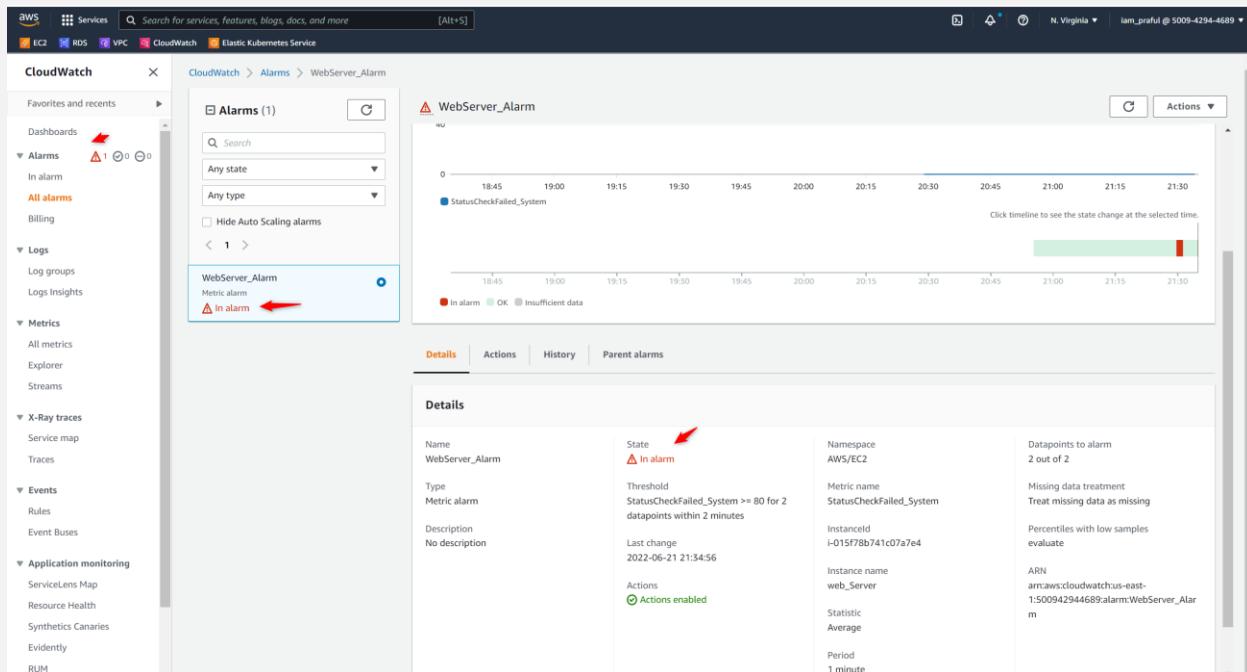
### Email Received



### Cloudwatch Alarm status changed to In-Alarm

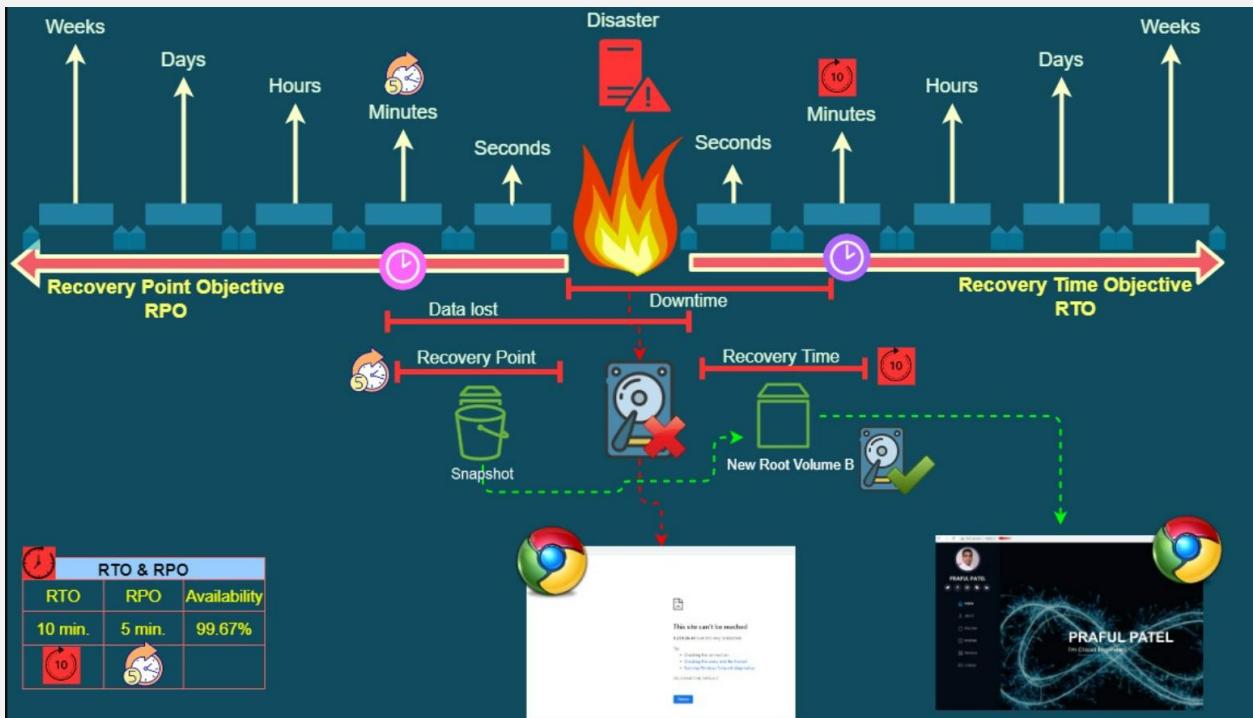


# AWS PROJECT: AWS-EBS ROOT VOLUME DISASTER RECOVERY, MONITORING & NOTIFICATION SOLUTION DESIGN & IMPLEMENTATION BY: PRAFUL PATEL



## 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 snapshot 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

The screenshot shows the AWS IAM console with the search bar set to 'iam'. The left sidebar lists services like EC2, RDS, VPC, CloudWatch, and Elastic Kubernetes Service. Under the 'Elastic Block Store' section, the 'Snapshots' tab is selected, showing a single snapshot named 'snapshot-root-volume-A' with a status of 'Available (100%)'. The right pane displays the IAM user creation process, with the 'Create New User' step completed, showing the user name 'iam\_praful' and a green checkmark indicating success.

Attach new volume to WebServer EC2 instance

Go to EC2

Replace root volume

# AWS PROJECT: AWS-EBS ROOT VOLUME DISASTER RECOVERY, MONITORING & NOTIFICATION SOLUTION DESIGN & IMPLEMENTATION BY: PRAFUL PATEL

The screenshot shows the AWS EC2 Instances page. A single instance, 'web\_Server' (Instance ID: i-015f78b741c07a7e4), is listed as 'Running'. A context menu is open over the instance, with the 'Replace root volume' option highlighted and circled with a red arrow.

EC2 > Instances > i-015f78b741c07a7e4 > Replace root volume

## Replace root volume [Info](#)

Restore the root volume for the selected instance from a specific snapshot, or restore it to its launch state. Data stored on instance store volumes, and networking and IAM configuration is retained.

**Root volume details**

Instance ID

[i-015f78b741c07a7e4 \(web\\_Server\)](#)

Root volume ID

[vol-0187867091ed1d546](#)

**Snapshot**

Specify a snapshot to restore the root volume to that state. Or omit the snapshot to restore the root volume to its initial launch state.

[snap-0b486efaa5b4e2655](#)   
snapshot of root volume A

All snapshots searched

No tags associated with the resource.

[Add new tag](#)

You can add up to 50 more tags.

[Cancel](#) [Create replacement task](#)

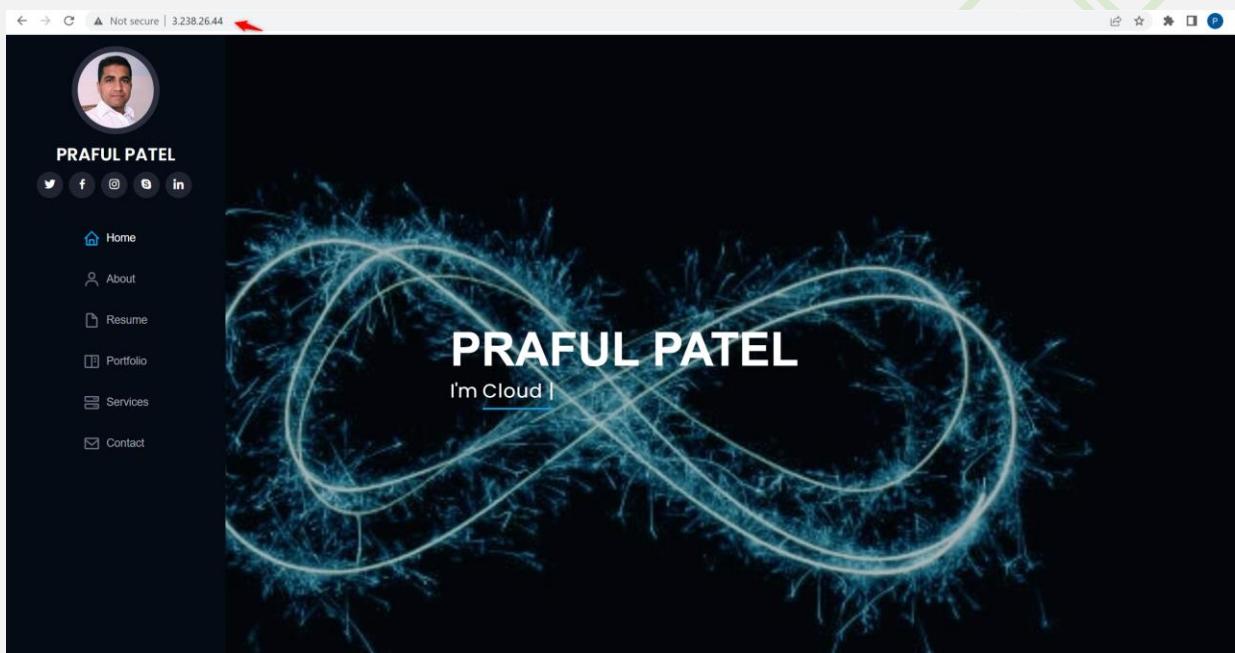
New Volume created and attached to ec2 and “In-Use” and older one is in “Available”

AWS PROJECT: AWS-EBS ROOT VOLUME DISASTER RECOVERY, MONITORING & NOTIFICATION  
SOLUTION DESIGN & IMPLEMENTATION BY: PRAFUL PATEL

Volumes (2)

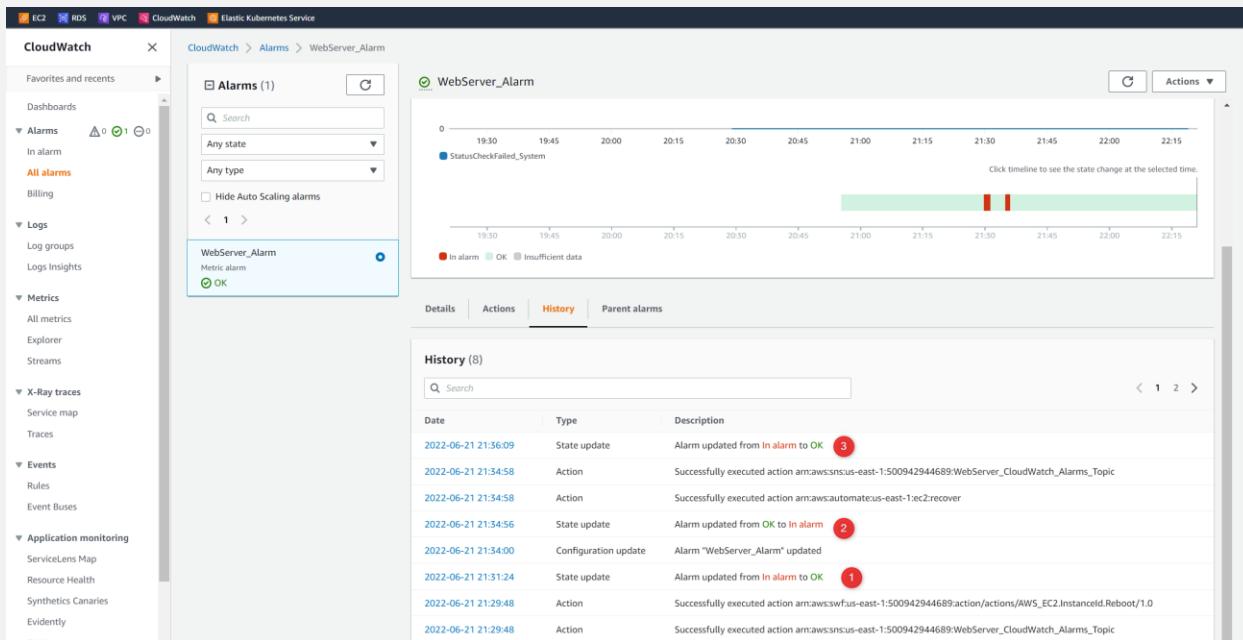
Name	Volume ID	Type	Size	IOPS	Snapshot	Created	Availability Zone	Volume state	
Root Volume A	vol-0187867091ed1d546	gp2	8 GiB	100	-	snap-02dc060ccaf62803f	2022/06/21 ...	us-east-1f	Available
-	vol-09999d4c710b4554b	gp2	8 GiB	100	-	snap-0b486efaa5b4e2655	2022/06/21 ...	us-east-1f	In-use

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

## AWS PROJECT: AWS-EBS ROOT VOLUME DISASTER RECOVERY, MONITORING & NOTIFICATION SOLUTION DESIGN & IMPLEMENTATION BY: PRAFUL PATEL



Congratulations!!!! 🎉

# AWS PROJECT: AWS-EBS ROOT VOLUME DISASTER RECOVERY, MONITORING & NOTIFICATION SOLUTION DESIGN & IMPLEMENTATION BY: PRAFUL PATEL

## Clean up Project:

### Terraform destroy

```

main.tf - cloud-devops (Workspace) - Visual Studio Code
File Edit Selection View Go Run Terminal Help
main.tf - main.tf ...tf-web_project U x README.md ...tf-web_project U $ app_remove.sh U $ user-data-apache.sh ...tf-web_project U
cloud-devops > terraform-project > tf-web_project > main.tf > resource "aws_route_table" "Public_RT" > route
1  terraform {
2    required_providers {
3      aws = {
4        source  = "hashicorp/aws"
5        version = "~> 3.0"
6      }
7    }
8  }
9
10 # Configure the AWS Provider
11 provider "aws" {
12   region = "us-east-1"
13 }
14
15 # Create VPC
aws_instance.Web_server: Still creating... [10s elapsed]
aws_instance.Web_server: Still creating... [20s elapsed]
aws_instance.Web_server: Still creating... [30s elapsed]
aws_instance.Web_server: Creation complete after 36s [id=i-015f78b741c07a7e4]

Apply complete! Resources: 7 added, 0 changed, 0 destroyed.

Outputs:
instance_id = "i-015f78b741c07a7e4"
instance_public_dns = "ec2-3-238-26-44.compute-1.amazonaws.com"
instance_public_ip = "3.238.26.44"
ip = "aws_instance.web_server.public_ip"
devops@DESKTOP-M660IAH:/mnt/d/cloud-devops/terraform-project$ terraform destroy
aws_vpc.PrafulVPC: Refreshing state... [id=vpc-0ae16cdeb9adc19]
aws_internet_gateway.Praful_IGW: Refreshing state... [id=igw-0da440703104ea2fa]
aws_subnet.PublicSubnet: Refreshing state... [id=subnet-074dad35f3ee79387]
aws_security_group.Public_SG: Refreshing state... [id=sg-00778d37e050cf0ef]
aws_route_table.Public_RT: Refreshing state... [id=rtb-015b36f188aa0f7f]
aws_instance.Web_server: Refreshing state... [id=i-015f78b741c07a7e4]
aws_route_table_association.a: Refreshing state... [id=rtbassoc-0ac839771d078de89]

Note: Objects have changed outside of Terraform

```

```

main.tf - cloud-devops (Workspace) - Visual Studio Code
File Edit Selection View Go Run Terminal Help
main.tf - main.tf ...tf-web_project U x README.md ...tf-web_project U $ app_remove.sh U $ user-data-apache.sh ...tf-web_project U
cloud-devops > terraform-project > tf-web_project > main.tf > resource "aws_route_table" "Public_RT" > route
1  terraform {
2    required_providers {
3      aws = {
4        source  = "hashicorp/aws"
5        version = "~> 3.0"
6      }
7    }
8  }
9
10 # Configure the AWS Provider
11 provider "aws" {
12   region = "us-east-1"
13 }
14
15 # Create VPC
Enter a value: yes
aws_route_table_association.a: Destroying... [id=rtbassoc-0ac839771d078de89]
aws_instance.Web_server: Destroying... [id=i-015f78b741c07a7e4]
aws_route_table_association.a: Destruction complete after 1s
aws_route_table.Public_RT: Destroying... [id=rtb-015b36f188aa0f7f]
aws_route_table.Public_RT: Destruction complete after 1s
aws_internet_gateway.Praful_IGW: Destroying... [id=igw-0da440703104ea2fa]
aws_instance.Web_server: Still destroying... [id=i-015f78b741c07a7e4, 10s elapsed]
aws_internet_gateway.Praful_IGW: Still destroying... [id=igw-0da440703104ea2fa, 10s elapsed]
aws_instance.Web_server: Still destroying... [id=i-015f78b741c07a7e4, 20s elapsed]
aws_internet_gateway.Praful_IGW: Still destroying... [id=igw-0da440703104ea2fa, 20s elapsed]
aws_instance.Web_server: Still destroying... [id=i-015f78b741c07a7e4, 30s elapsed]
aws_internet_gateway.Praful_IGW: Destruction complete after 30s
aws_instance.Web_server: Destruction complete after 32s
aws_subnet.PublicSubnet: Destroying... [id=subnet-074dad35f3ee79387]
aws_security_group.Public_SG: Destroying... [id=sg-00778d37e050cf0ef]
aws_subnet.PublicSubnet: Destruction complete after 0s
aws_security_group.Public_SG: Destruction complete after 0s
aws_vpc.PrafulVPC: Destroying... [id=vpc-0ae16cdeb9adc19]
aws_vpc.PrafulVPC: Destruction complete after 0s

Destroy complete! Resources: 7 destroyed.
devops@DESKTOP-M660IAH:/mnt/d/cloud-devops/terraform-project$ 

```

## Remove volume

Remove snapshots

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://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](https://docs.ansible.com/ansible/2.5/modules/ec2_snapshot_module.html)



Congratulations!!!! 🎉