AWS TerraOps - Infrastructure Automation and Scalability

Manual

By
Vivek Vashisht

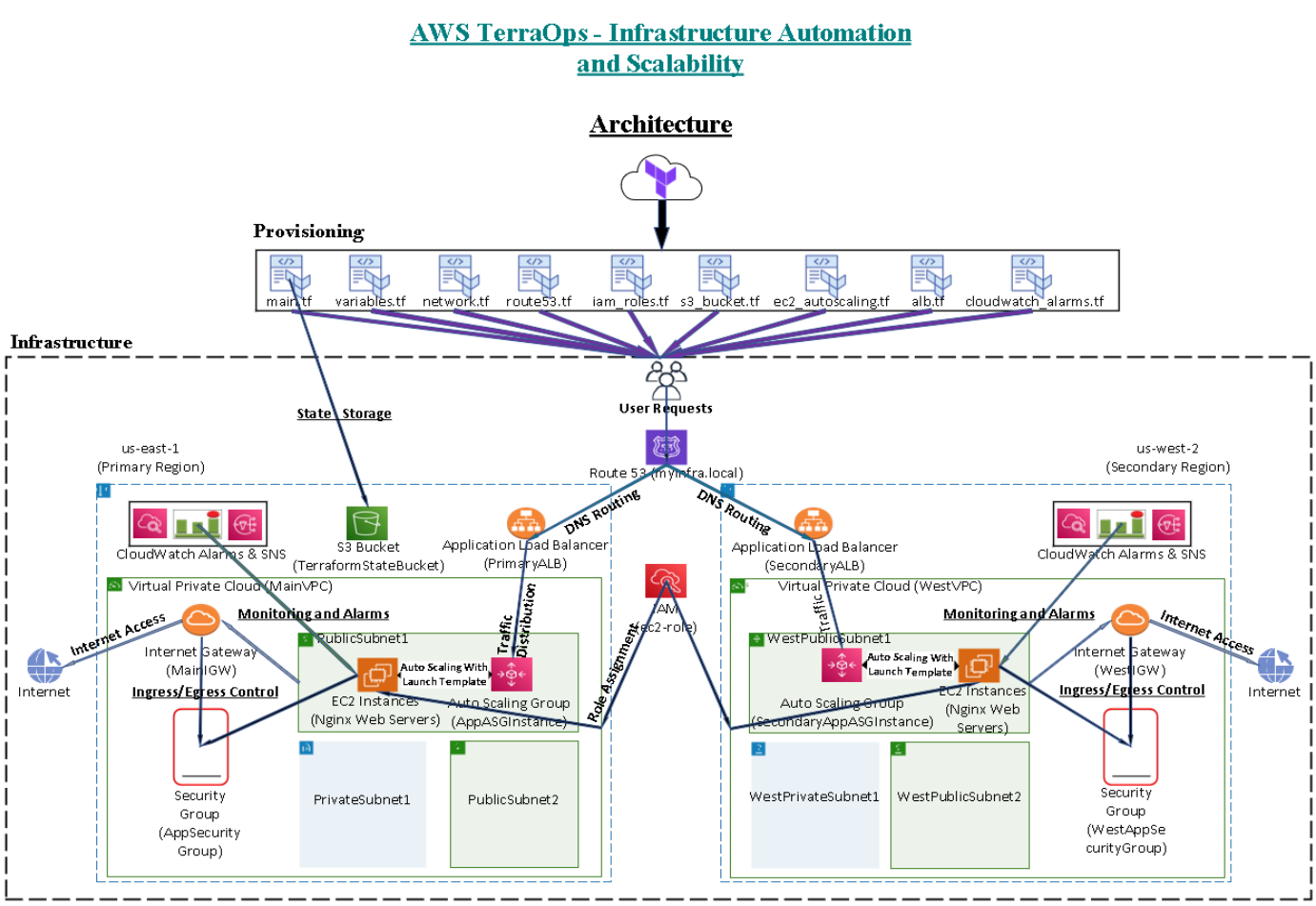
Date: 13 November 2024

Contact:-
LinkedIn: https://www.linkedin.com/in/vivek-vashisht04/

GitHub: https://github.com/vivekvashisht04/AWS-TerraOps

Table of Contents

Project Architecture Diagram



Introduction

Project Overview

The AWS TerraOps: Infrastructure Automation and Scalability project is a strategic initiative designed to automate infrastructure deployment and achieve scalable cloud solutions using AWS services. This project leverages AWS's suite of tools to create a resilient, automated, and highly available environment capable of adapting to fluctuating workloads. It emphasizes seamless scaling, effective resource management, and failover mechanisms to ensure reliability and optimal performance.

Throughout the project, I utilized various AWS services, including EC2, Auto Scaling Groups, Application Load Balancers (ALBs), Route 53, and S3 Buckets. The project demonstrates the complete lifecycle, from infrastructure provisioning and configuration management to monitoring and failover testing. Each component was meticulously implemented and documented, capturing all critical aspects of infrastructure automation and scalability within AWS.

Purpose of this Documentation

This document provides a comprehensive, step-by-step guide for replicating the AWS TerraOps project, complete with detailed instructions and screenshots. It serves as a practical resource for cloud enthusiasts, professionals, and employers interested in learning infrastructure automation and scalability practices in AWS. The primary goal is to document the project thoroughly, showcasing my technical expertise in AWS and Terraform, while providing a valuable reference for potential employers and the broader cloud community.

Prerequisites

Before starting this project, it's essential to have a foundational understanding of cloud infrastructure, networking, and automation. Additionally, the following prerequisites are recommended:

1. **AWS Account**: A valid AWS account is necessary to deploy and manage cloud resources.
2. **Basic Networking Knowledge**: Familiarity with IP addressing, subnets, and security configurations.
3. **Terraform Installed**: Knowledge of Infrastructure as Code (IaC) concepts and experience with Terraform for resource provisioning.
4. **AWS CLI (Optional)**: Familiarity with AWS CLI can be beneficial for automation and advanced configurations.
5. **Text Editor (Optional)**: A text editor, such as Visual Studio Code, may be useful for managing configuration files and scripts.

This documentation aims to showcase the steps I took to complete the project, providing an informative guide to building scalable and automated infrastructure in AWS.

Initial Setup

a) Terraform and AWS CLI Installation

To start, I ensured that **terraform** was installed on my local machine and verified its installation by running the commands:

- `terraform`
- `terraform -version`

Next, I created a folder called **Binaries** in the C drive of my local machine, moved the Terraform executable file into this folder, and set up an environment variable. To do this, I accessed the **System Properties** window, selected **Environment Variables**, chose the **PATH** variable, clicked **Edit**, and added a path to the **Binaries** folder. This setup allowed Terraform to be accessible from any command line interface on my machine.

Then, I proceeded with the **AWS CLI** installation. I verified the installation by running the command:

- `aws --version`

Afterward, I configured AWS credentials. I logged into my AWS console and ensured that my account had the **AdministratorAccess** permission. In the **IAM** section, I created an access key. Once the **Access Key ID** and **Secret Access Key** were generated, I ran the following command in CMD:

- `aws configure`

I entered the credentials, completing the AWS configuration.

b) Version Control Setup

For version control, I ensured that **Git** was installed on my local machine. I visited Git's official download page to download and install Git.

To initialize a Git repository, I created a folder named **Git**, navigated to this directory in CMD, and ran:

- `git init`

Within the **Git** folder, I created a **.gitignore** file to exclude specific files and folders, adding the following entries:

1. `.terraform/`
2. `terraform.tfstate`

3. `terraform.tfstate.backup`

I then staged and committed the files with:

- `git add .`
- `git commit -m "Initial commit with Terraform setup"`

After committing, I created a new branch named **dev** with:

- `git checkout -b dev`

To set up a remote repository, I added the GitHub repository as a remote by running:

- `git remote add origin https://github.com/VivekVashisht2004/AWS-TerraOps.git`

I then pushed the changes to the **dev** branch and authorized Git credentials:

- `git push -u origin dev`

c) Creating the Main Terraform Configuration File (main.tf)

Next, I created a **main.tf** file containing the basic Terraform configuration to initialize the provider and create an S3 bucket to store the `terraform.tfstate` file as a remote backend.

Using references from the **Terraform Registry**, **AWS Documentation**, and AI validation, I configured the **main.tf** file as follows:

```
provider "aws" {
  region = "us-east-1"
}

# S3 bucket
resource "aws_s3_bucket" "terraform_state" {
  bucket = "aws-tf-backend-vv-bucket"

  tags = {
    Name        = "TerraformStateBucket"
    Environment = "Dev"
  }
}

# Public access block for the S3 bucket
resource "aws_s3_bucket_public_access_block" "terraform_state_block" {
  bucket = aws_s3_bucket.terraform_state.id

  block_public_acls       = true
  block_public_policy     = true
  restrict_public_buckets = true
  ignore_public_acls      = true
}
```

After saving the **main.tf** file, I navigated to the **Terraform** folder and ran the following commands to initialize, plan, and apply the configuration:

- `terraform init`
- `terraform plan`
- `terraform apply -auto-approve`

Following successful initialization, planning, and application, I confirmed the resource creation by checking the **aws-tf-backend-vv-bucket** in my AWS Console.

d) Remote Backend Configuration

To complete the remote backend configuration, I updated the **main.tf** file with the following block:

```
terraform {
  backend "s3" {
    bucket = "aws-tf-backend-vv-bucket"
    key    = "terraform/dev/terraform.tfstate"
    region = "us-east-1"
  }
}
```

After saving the **main.tf** file with the new backend configuration, I re-ran:

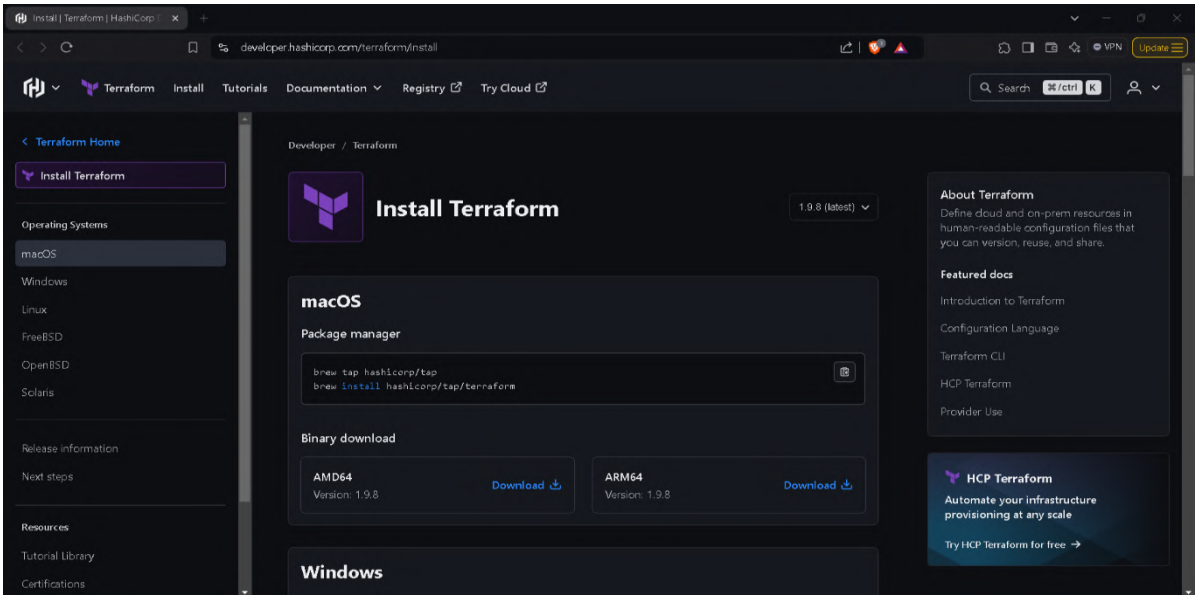
- `terraform init`

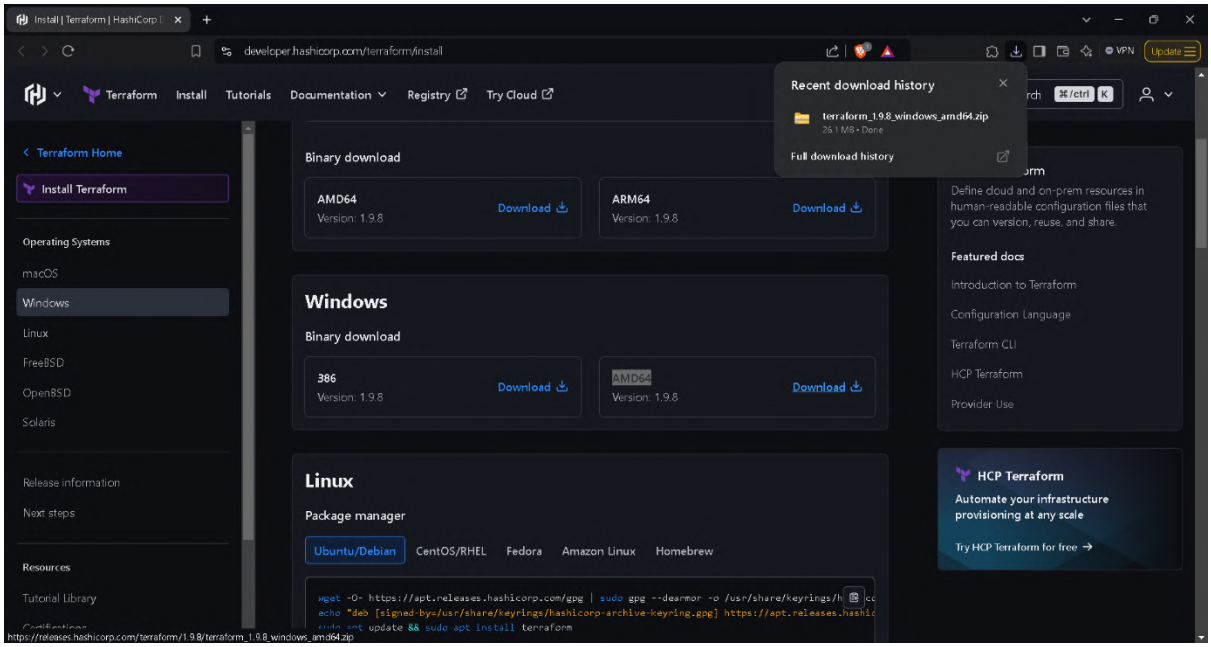
The command prompted to copy the existing state to the new backend. I confirmed, and with successful initialization, the S3 backend configuration was completed.

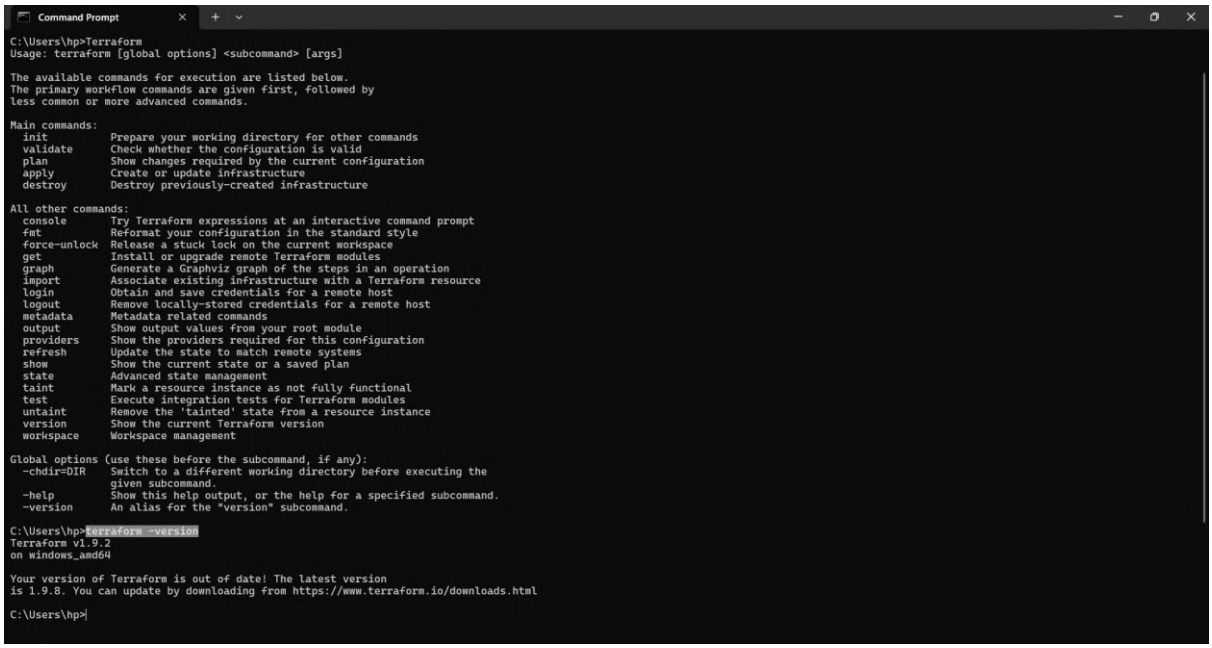
To verify, I checked the **aws-tf-backend-vv-bucket** in the AWS Console and found the `terraform.tfstate` file at the specified path **terraform/dev/terraform.tfstate**.

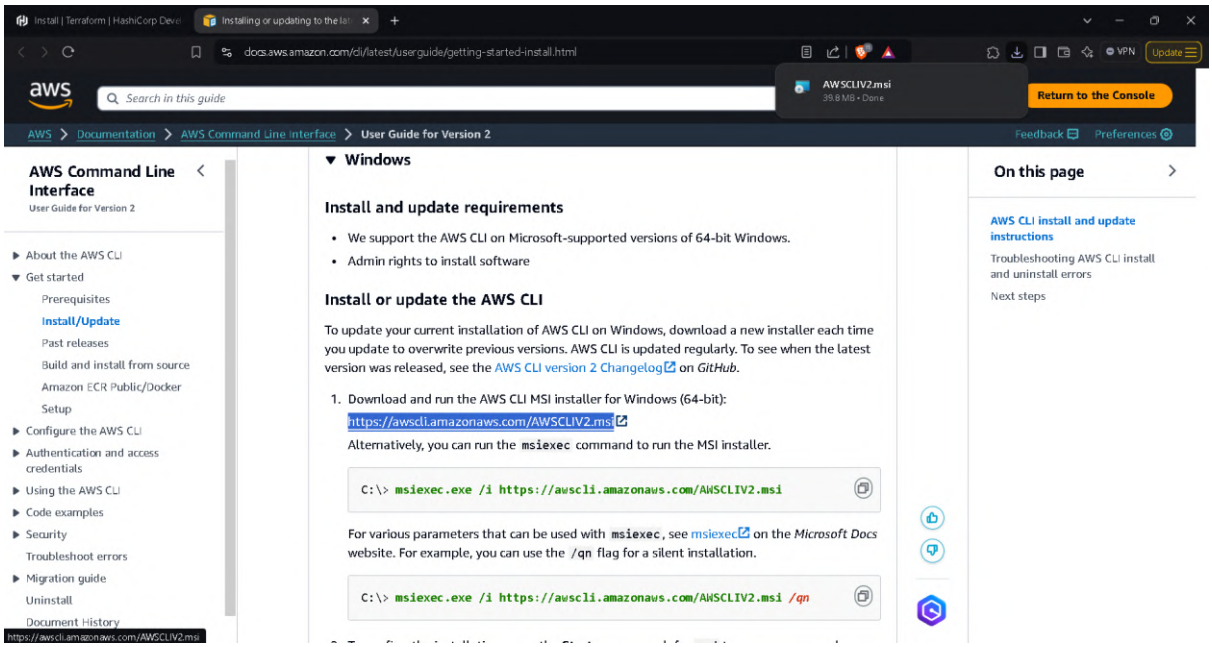
With this, the **Initial Setup** phase of the project was successfully completed.

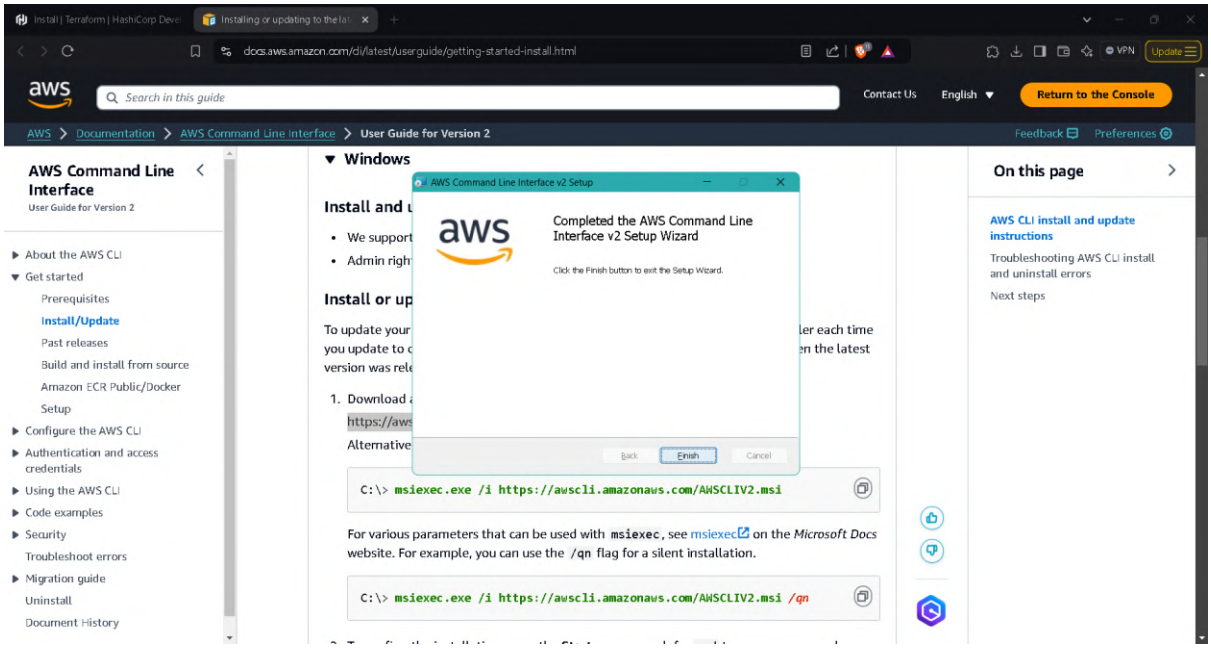
Screenshots

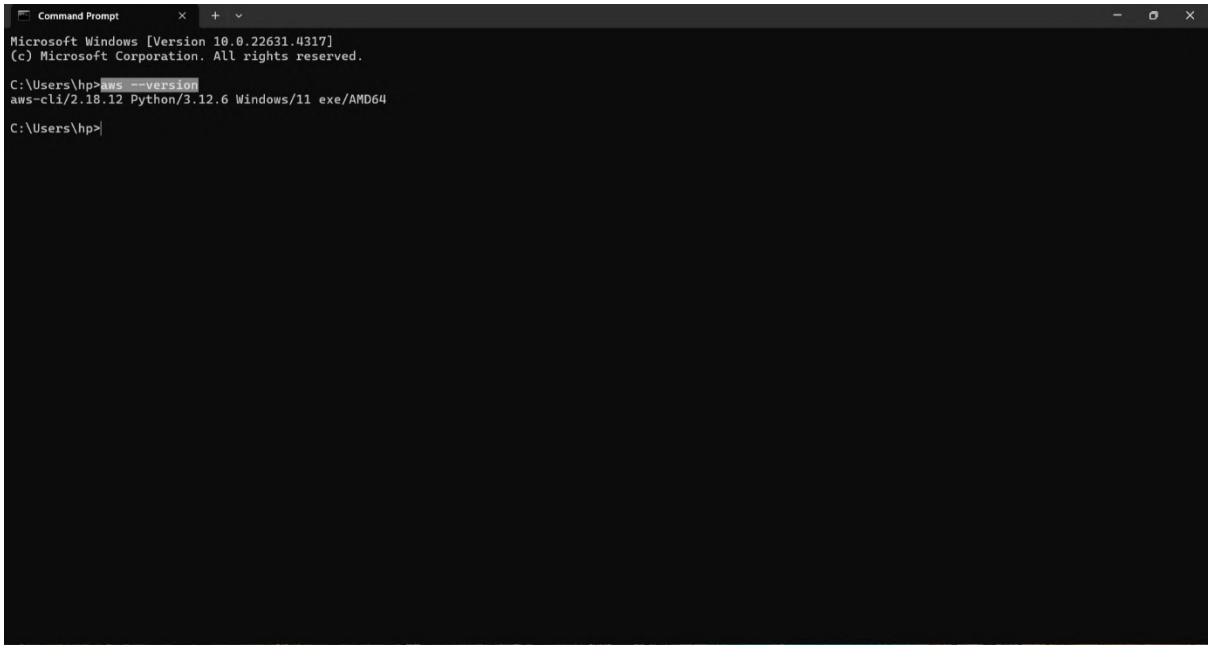


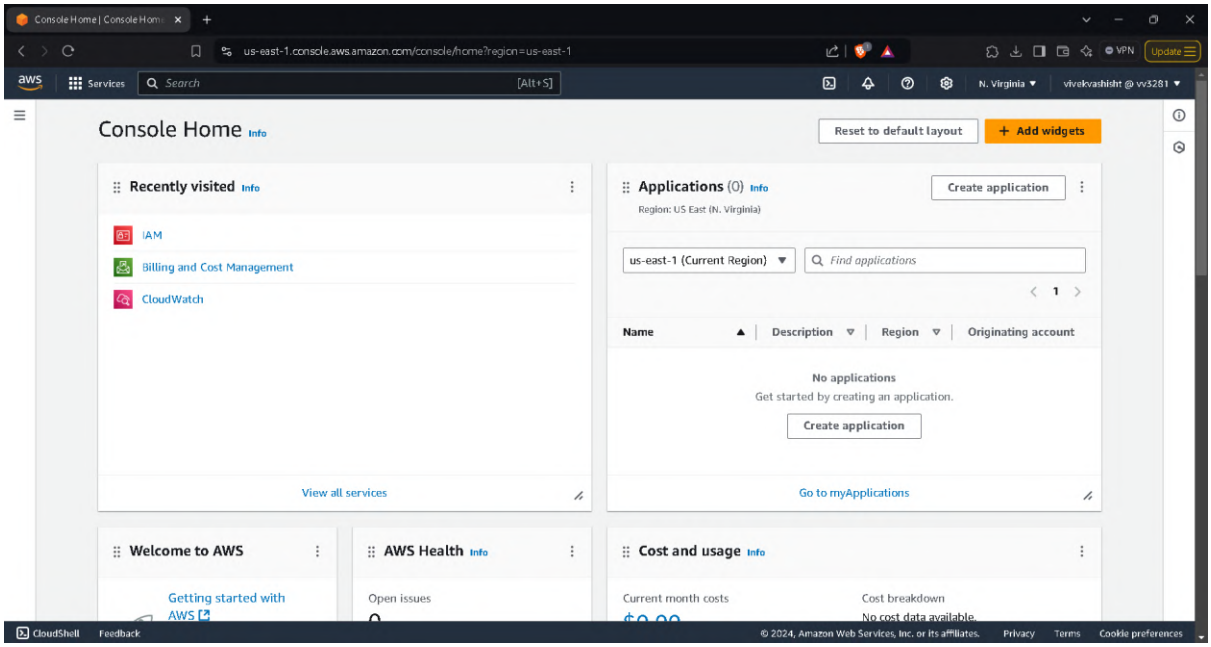

Install | Terraform | HashiCorp
developer.hashicorp.com/terraform/install
Terraform
Install
Tutorials
Documentation
Registry
Try Cloud
Recent download history
terraform_1.9.8_windows_amd64.zip
26.1 MB • Done
Full download history
Terraform Home
Install Terraform
Operating Systems
macOS
Windows
Linux
FreeBSD
OpenBSD
Solaris
Release information
Next steps
Resources
Tutorial Library
Binary download
AMD64
Version: 1.9.8
Download
ARM64
Version: 1.9.8
Download
Windows
Binary download
386
Version: 1.9.8
Download
AMD64
Version: 1.9.8
Download
Linux
Package manager
Ubuntu/Debian
CentOS/RHEL
Fedora
Amazon Linux
Homebrew
wget -O- https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o /usr/share/keyrings/h
echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashi
sudo apt update && sudo apt install terraform
Define cloud and on-prem resources in human-readable configuration files that you can version, reuse, and share.
Featured docs
Introduction to Terraform
Configuration Language
Terraform CLI
HCP Terraform
Provider Use
HCP Terraform
Automate your infrastructure provisioning at any scale
Try HCP Terraform for free
https://releases.hashicorp.com/terraform/1.9.8/terraform_1.9.8_windows_amd64.zip

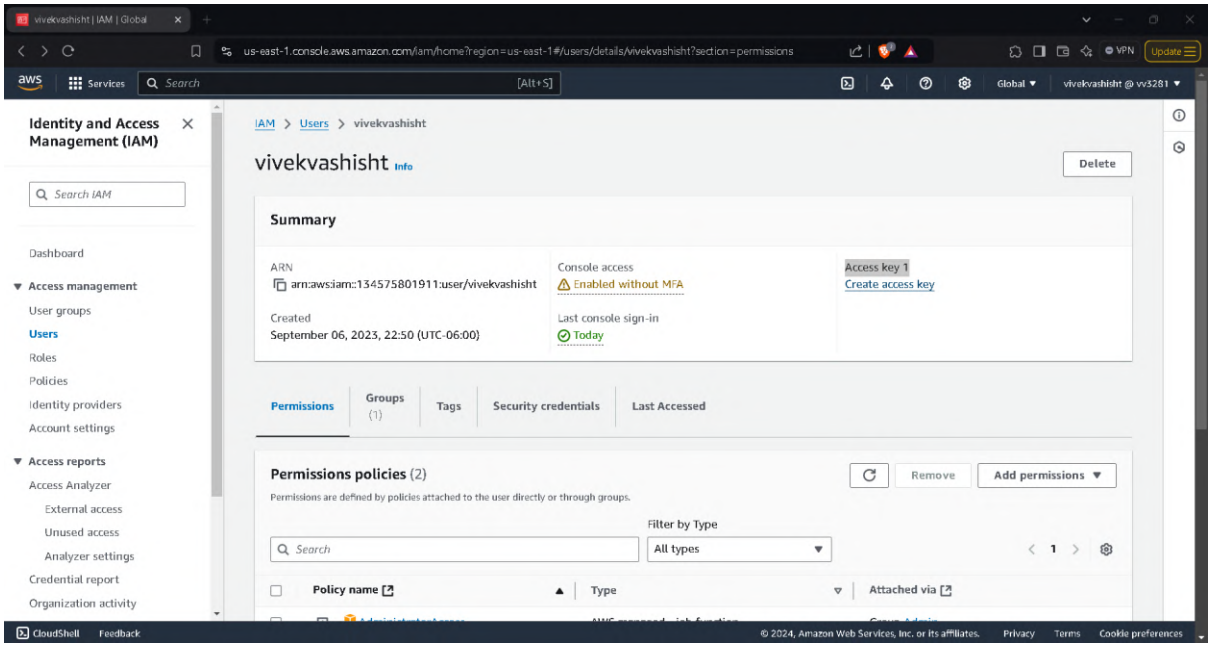

Command Prompt
C:\Users\hp>Terraform
Usage: terraform [global options] <subcommand> [args]
The available commands for execution are listed below.
The primary workflow commands are given first, followed by
less common or more advanced commands.
Main commands:
init Prepare your working directory for other commands
validate Check whether the configuration is valid
plan Show changes required by the current configuration
apply Create or update infrastructure
destroy Destroy previously-created infrastructure
All other commands:
console Try Terraform expressions at an interactive command prompt
fmt Reformat your configuration in the standard style
force-unlock Release a stuck lock on the current workspace
get Install or upgrade remote Terraform modules
graph Generate a Graphviz graph of the steps in an operation
import Associate existing infrastructure with a Terraform resource
login Obtain and save credentials for a remote host
logout Remove locally-stored credentials for a remote host
metadata Metadata related commands
output Show output values from your root module
providers Show the providers required for this configuration
refresh Update the state to match remote systems
show Show the current state or a saved plan
state Advanced state management
taint Mark a resource instance as not fully functional
test Execute integration tests for Terraform modules
untaint Remove the 'tainted' state from a resource instance
version Show the current Terraform version
workspace Workspace management
Global options (use these before the subcommand, if any):
-chdir=DIR Switch to a different working directory before executing the
given subcommand.
-help Show this help output, or the help for a specified subcommand.
-version An alias for the "version" subcommand.
C:\Users\hp>terraform -version
Terraform v1.9.2
on windows_amd64
Your version of Terraform is out of date! The latest version
is 1.9.8. You can update by downloading from https://www.terraform.io/downloads.html
C:\Users\hp>

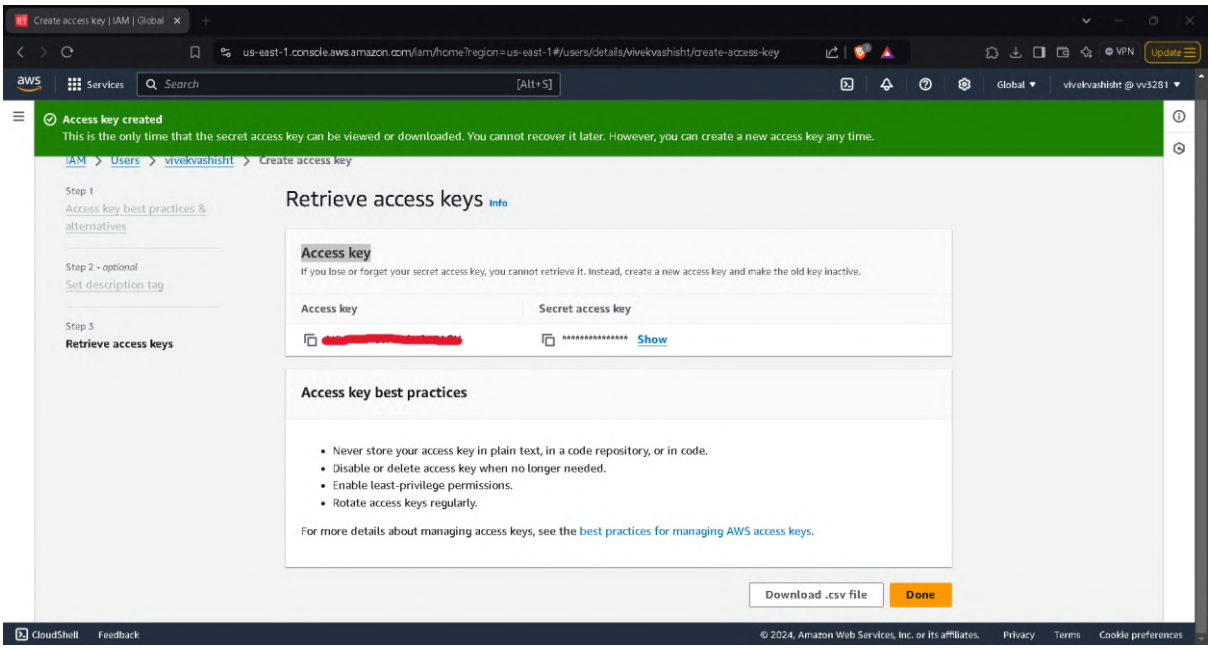

Install | Terraform | HashiCorp Deve
Installing or updating to the lat
docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html
aws
Search in this guide
AWSCLIV2.msi
39.8 MB • Done
Return to the Console
AWS > Documentation > AWS Command Line Interface > User Guide for Version 2
Feedback
Preferences
AWS Command Line Interface
User Guide for Version 2
About the AWS CLI
Get started
Prerequisites
Install/Update
Past releases
Build and install from source
Amazon ECR Public/Docker
Setup
Configure the AWS CLI
Authentication and access credentials
Using the AWS CLI
Code examples
Security
Troubleshoot errors
Migration guide
Uninstall
Document History
Windows
Install and update requirements
We support the AWS CLI on Microsoft-supported versions of 64-bit Windows.
Admin rights to install software
Install or update the AWS CLI
To update your current installation of AWS CLI on Windows, download a new installer each time you update to overwrite previous versions. AWS CLI is updated regularly. To see when the latest version was released, see the AWS CLI version 2 Changelog on GitHub.
1. Download and run the AWS CLI MSI installer for Windows (64-bit):
https://awscli.amazonaws.com/AWSCLIV2.msi
Alternatively, you can run the msiexec command to run the MSI installer.
C:\> msiexec.exe /i https://awscli.amazonaws.com/AWSCLIV2.msi
For various parameters that can be used with msiexec, see msiexec on the Microsoft Docs website. For example, you can use the /qn flag for a silent installation.
C:\> msiexec.exe /i https://awscli.amazonaws.com/AWSCLIV2.msi /qn
On this page
AWS CLI install and update instructions
Troubleshooting AWS CLI install and uninstall errors
Next steps
https://awscli.amazonaws.com/AWSCLIV2.msi

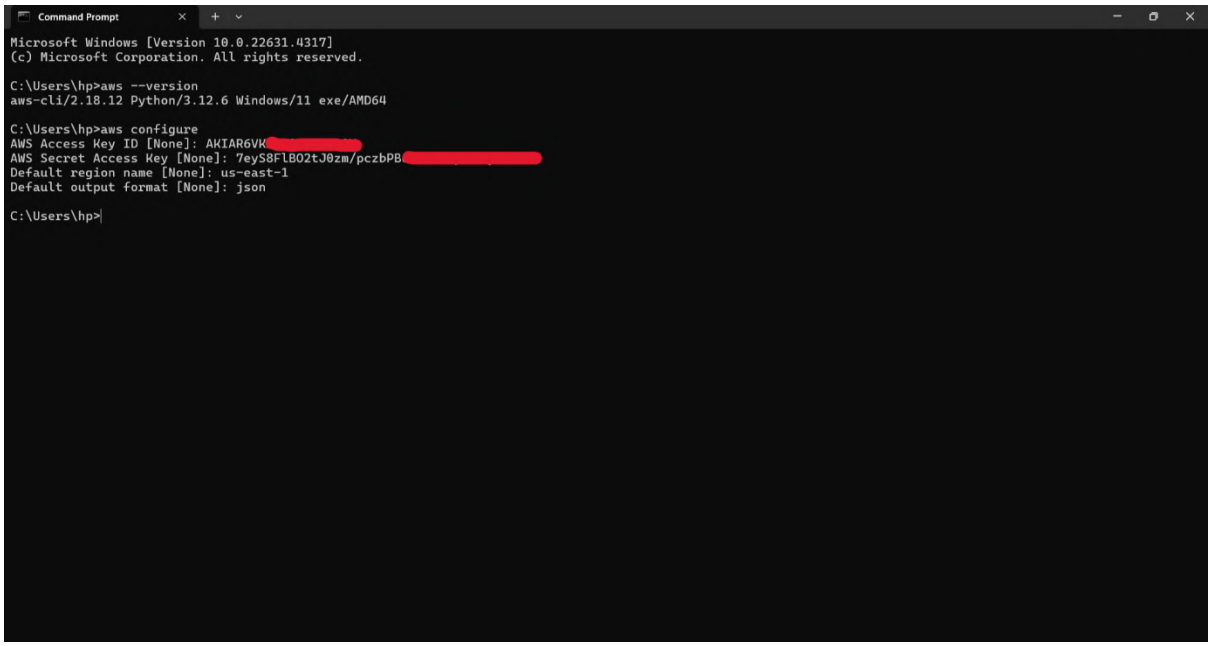

docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html
Search in this guide
Contact Us
English
Return to the Console
AWS > Documentation > AWS Command Line Interface > User Guide for Version 2
Feedback
Preferences
AWS Command Line Interface
User Guide for Version 2
About the AWS CLI
Get started
Prerequisites
Install/Update
Past releases
Build and install from source
Amazon ECR Public/Docker
Setup
Configure the AWS CLI
Authentication and access credentials
Using the AWS CLI
Code examples
Security
Troubleshoot errors
Migration guide
Uninstall
Document History
Windows
Install and u
We support
Admin righ
Install or up
To update your
you update to
version was rele
1. Download
https://aws
Alternative
AWS Command Line Interface v2 Setup
Completed the AWS Command Line Interface v2 Setup Wizard
Click the Finish button to exit the Setup Wizard.
Back
Finish
Cancel
C:\> msiexec.exe /i https://awscli.amazonaws.com/AWSCLIV2.msi
For various parameters that can be used with msiexec, see msiexec on the Microsoft Docs website. For example, you can use the /qn flag for a silent installation.
C:\> msiexec.exe /i https://awscli.amazonaws.com/AWSCLIV2.msi /qn
On this page
AWS CLI install and update instructions
Troubleshooting AWS CLI install and uninstall errors
Next steps

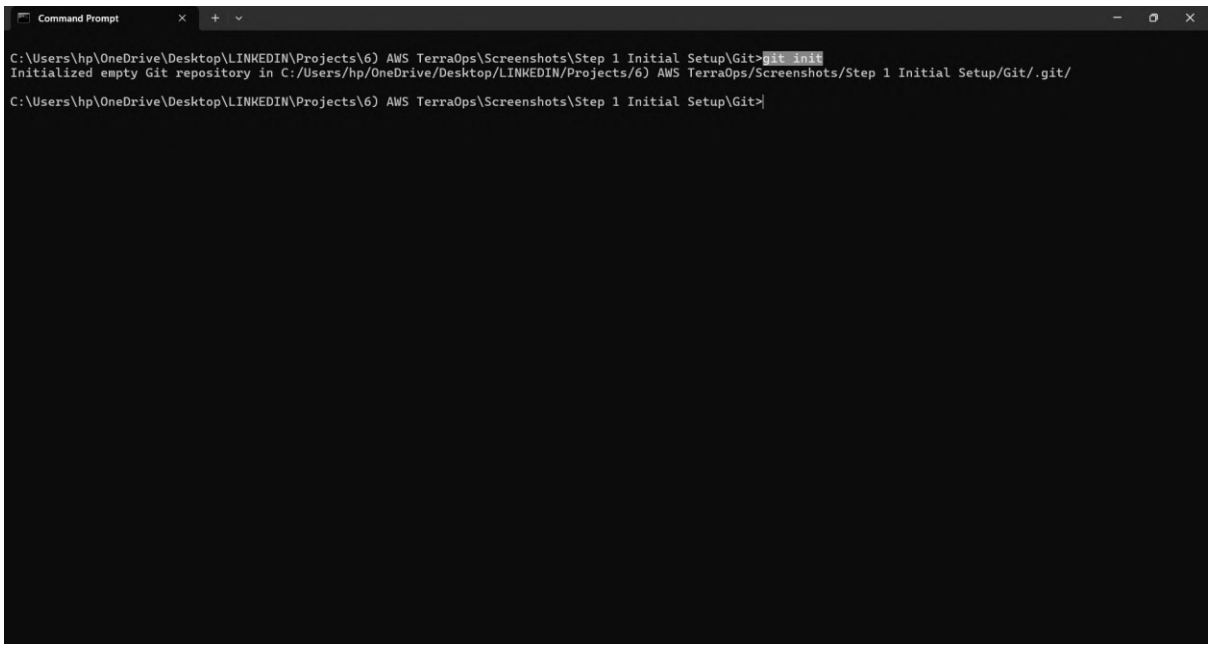

Command Prompt
Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.
C:\Users\hp>aws --version
aws-cli/2.18.12 Python/3.12.6 Windows/11 exe/AMD64
C:\Users\hp>

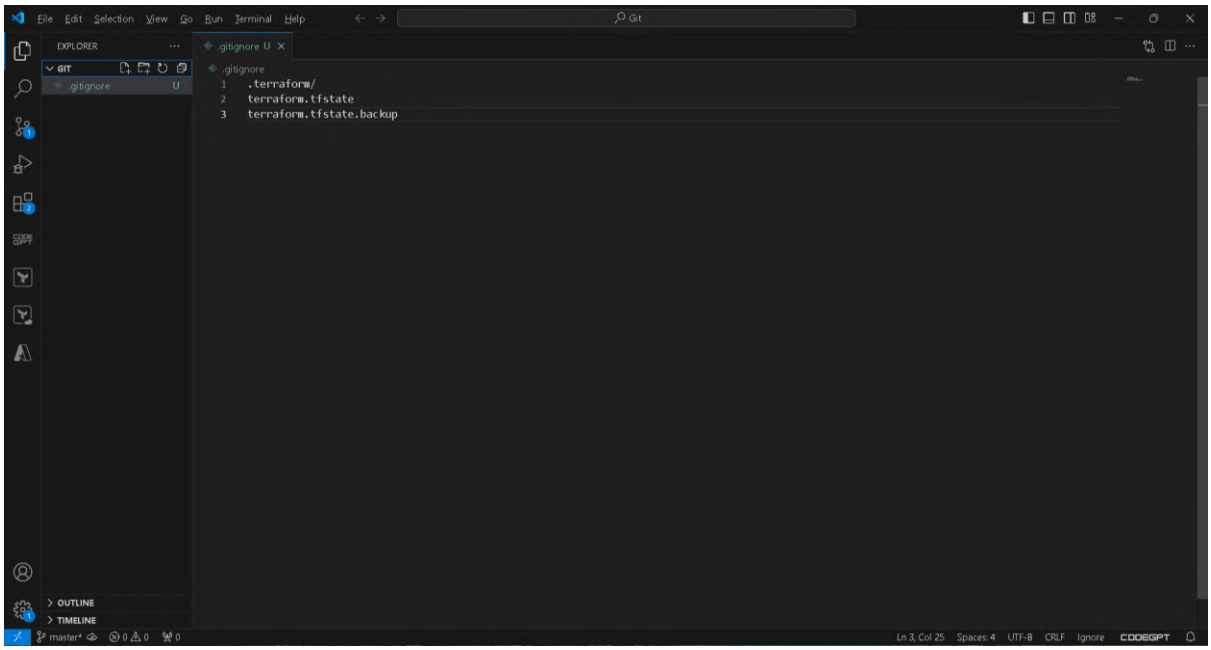

us-east-1.console.aws.amazon.com/console/home?region=us-east-1
Services
Search
[Alt+S]
N. Virginia
Console Home Info
Reset to default layout
+ Add widgets
Recently visited Info
IAM
Billing and Cost Management
CloudWatch
View all services
Applications (0) Info
Region: US East (N. Virginia)
Create application
us-east-1 (Current Region)
Find applications
Name
Description
Region
Originating account
No applications
Get started by creating an application.
Create application
Go to myApplications
Welcome to AWS
Getting started with AWS
AWS Health Info
Open issues
Cost and usage Info
Current month costs
Cost breakdown
No cost data available.
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

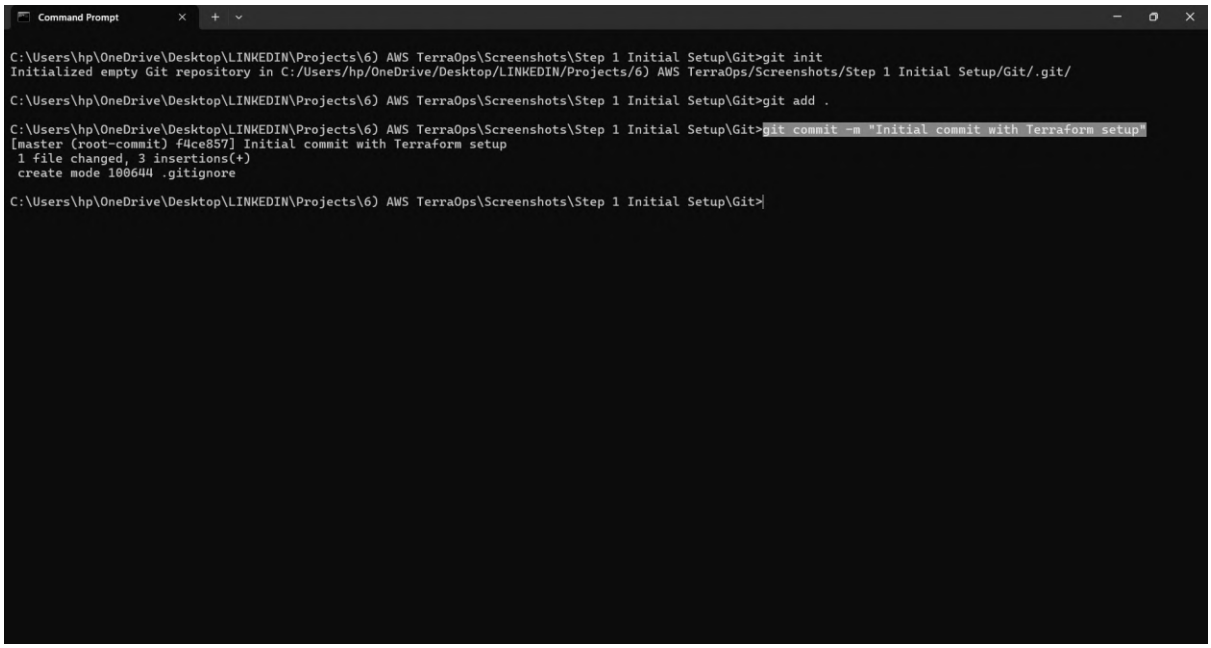

vivekvashisht | IAM | Global
us-east-1.console.aws.amazon.com/iam/home?region=us-east-1#/users/details/vivekvashisht?section=permissions
Identity and Access Management (IAM)
Search IAM
Dashboard
Access management
User groups
Users
Roles
Policies
Identity providers
Account settings
Access reports
Access Analyzer
External access
Unused access
Analyzer settings
Credential report
Organization activity
IAM > Users > vivekvashisht
vivekvashisht Info
Delete
Summary
ARN
arn:aws:iam::134575801911:user/vivekvashisht
Console access
Enabled without MFA
Access key 1
Create access key
Created
September 06, 2023, 22:50 (UTC-06:00)
Last console sign-in
Today
Permissions
Groups (1)
Tags
Security credentials
Last Accessed
Permissions policies (2)
Permissions are defined by policies attached to the user directly or through groups.
Remove
Add permissions
Filter by Type
Search
All types
Policy name
Type
Attached via
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

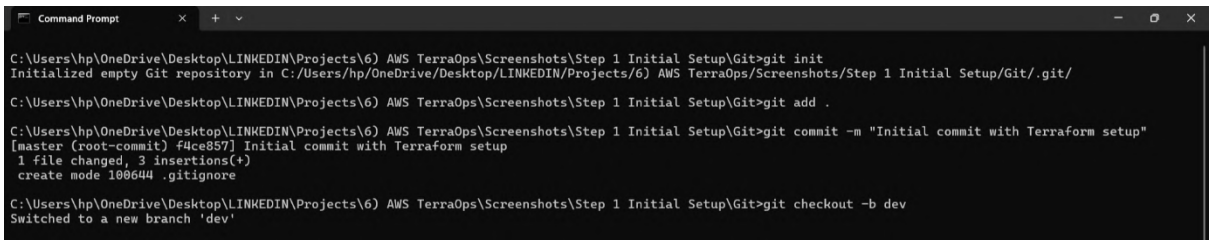

Create access key | IAM | Global
us-east-1.console.aws.amazon.com/iam/home?region=us-east-1#/users/details/vivekvashisht/create-access-key
Access key created
This is the only time that the secret access key can be viewed or downloaded. You cannot recover it later. However, you can create a new access key any time.
IAM > Users > vivekvashisht > Create access key
Step 1
Access key best practices & alternatives
Step 2 - optional
Set description tag
Step 3
Retrieve access keys
Retrieve access keys Info
Access key
If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.
Access key
Secret access key
[redacted]
********** Show
Access key best practices
Never store your access key in plain text, in a code repository, or in code.
Disable or delete access key when no longer needed.
Enable least-privilege permissions.
Rotate access keys regularly.
For more details about managing access keys, see the best practices for managing AWS access keys.
Download .csv file
Done
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

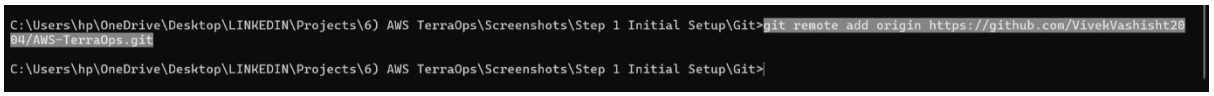

Command Prompt
Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.
C:\Users\hp>aws --version
aws-cli/2.18.12 Python/3.12.6 Windows/11 exe/AMD64
C:\Users\hp>aws configure
AWS Access Key ID [None]: AKIAR6VK[redacted]
AWS Secret Access Key [None]: 7eyS8FlBO2tJ0zm/pczbPB[redacted]
Default region name [None]: us-east-1
Default output format [None]: json
C:\Users\hp>


Command Prompt
C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Git>git init
Initialized empty Git repository in C:/Users/hp/OneDrive/Desktop/LINKEDIN/Projects/6) AWS TerraOps/Screenshots/Step 1 Initial Setup/Git/.git/
C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Git>

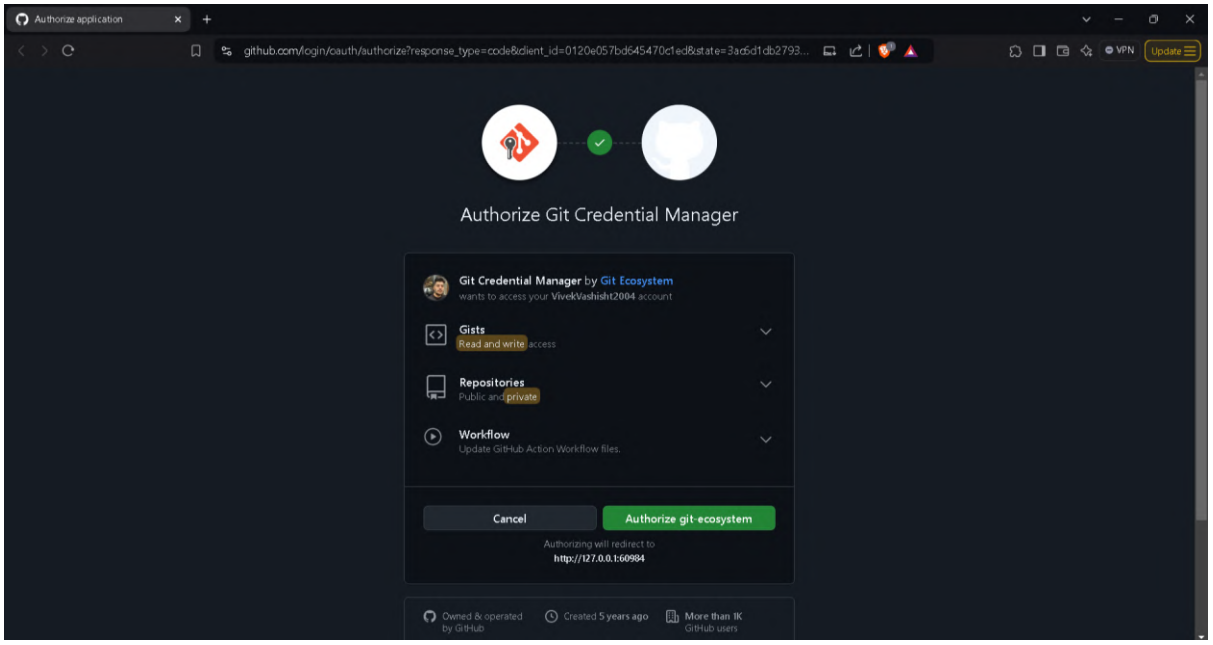

File Edit Selection View Go Run Terminal Help
Git
EXPLORER
.gitignore U
GIT
.gitignore U
.gitignore
1 .terraform/
2 terraform.tfstate
3 terraform.tfstate.backup
OUTLINE
TIMELINE
master*
Ln 3, Col 25 Spaces: 4 UTF-8 CRLF Ignore CODEGPT

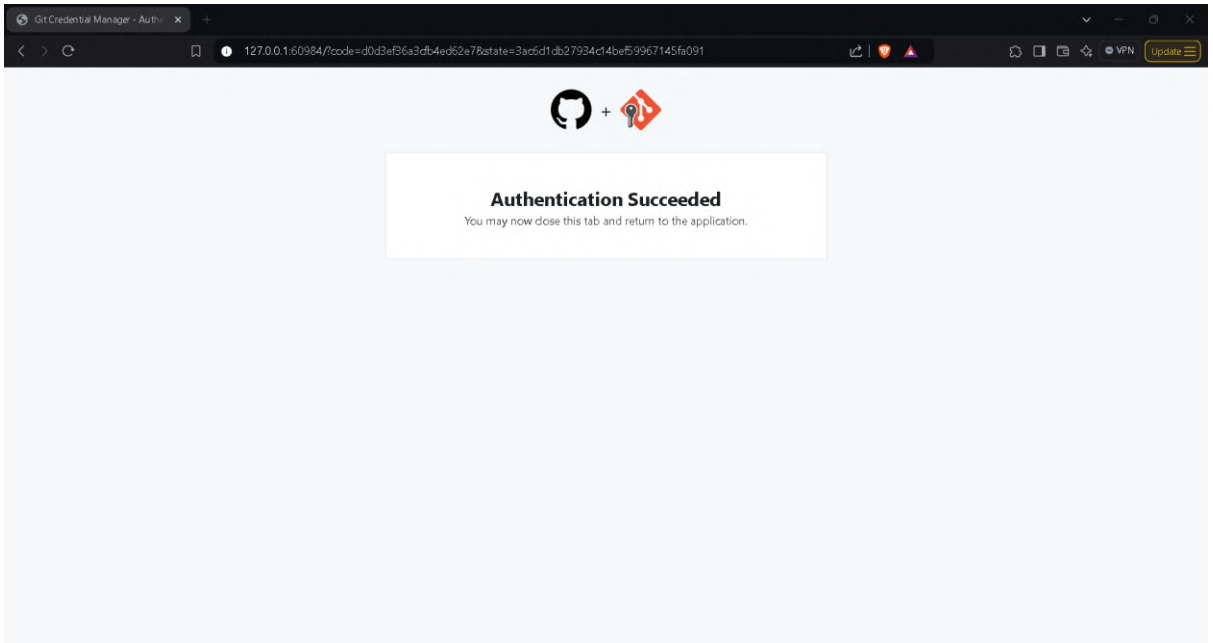

Command Prompt
C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Git>git init
Initialized empty Git repository in C:/Users/hp/OneDrive/Desktop/LINKEDIN/Projects/6) AWS TerraOps/Screenshots/Step 1 Initial Setup/Git/.git/
C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Git>git add .
C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Git>git commit -m "Initial commit with Terraform setup"
[master (root-commit) f4ce857] Initial commit with Terraform setup
1 file changed, 3 insertions(+)
create mode 100644 .gitignore
C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Git>


Command Prompt
C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Git>git init
Initialized empty Git repository in C:/Users/hp/OneDrive/Desktop/LINKEDIN/Projects/6) AWS TerraOps/Screenshots/Step 1 Initial Setup/Git/.git/
C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Git>git add .
C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Git>git commit -m "Initial commit with Terraform setup"
[master (root-commit) f4ce857] Initial commit with Terraform setup
1 file changed, 3 insertions(+)
create mode 100644 .gitignore
C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Git>git checkout -b dev
Switched to a new branch 'dev'

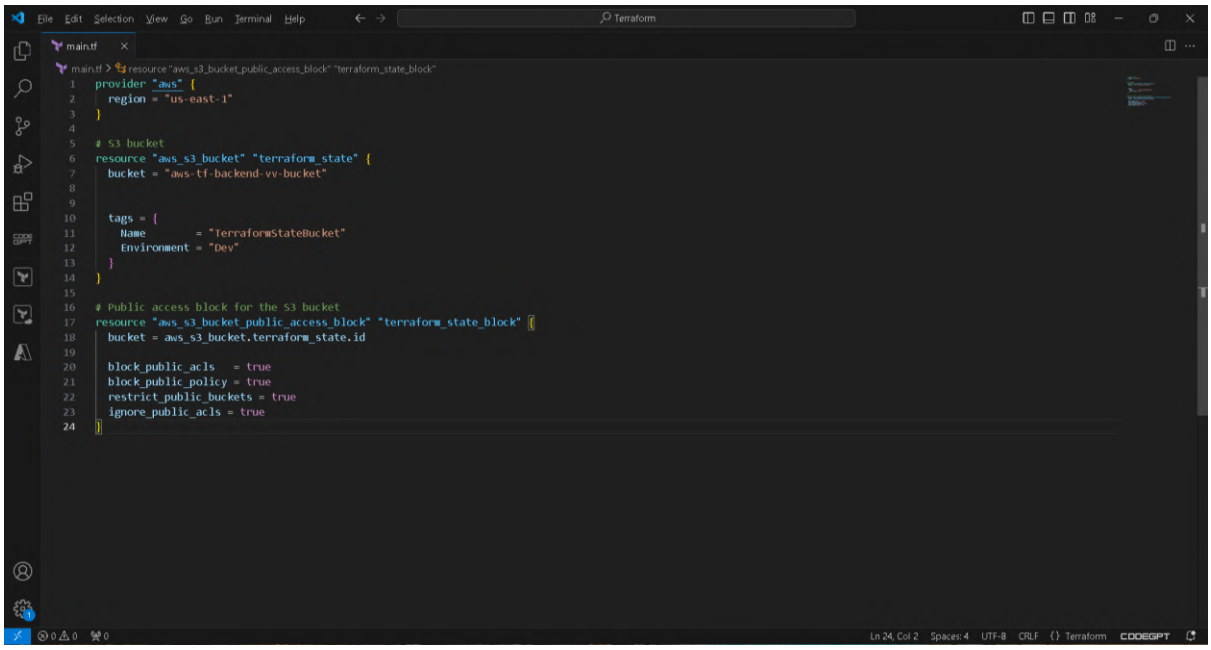

C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Git>git remote add origin https://github.com/VivekVashisht2004/AWS-TerraOps.git
C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Git>


C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Git>git push -u origin dev
info: please complete authentication in your browser...

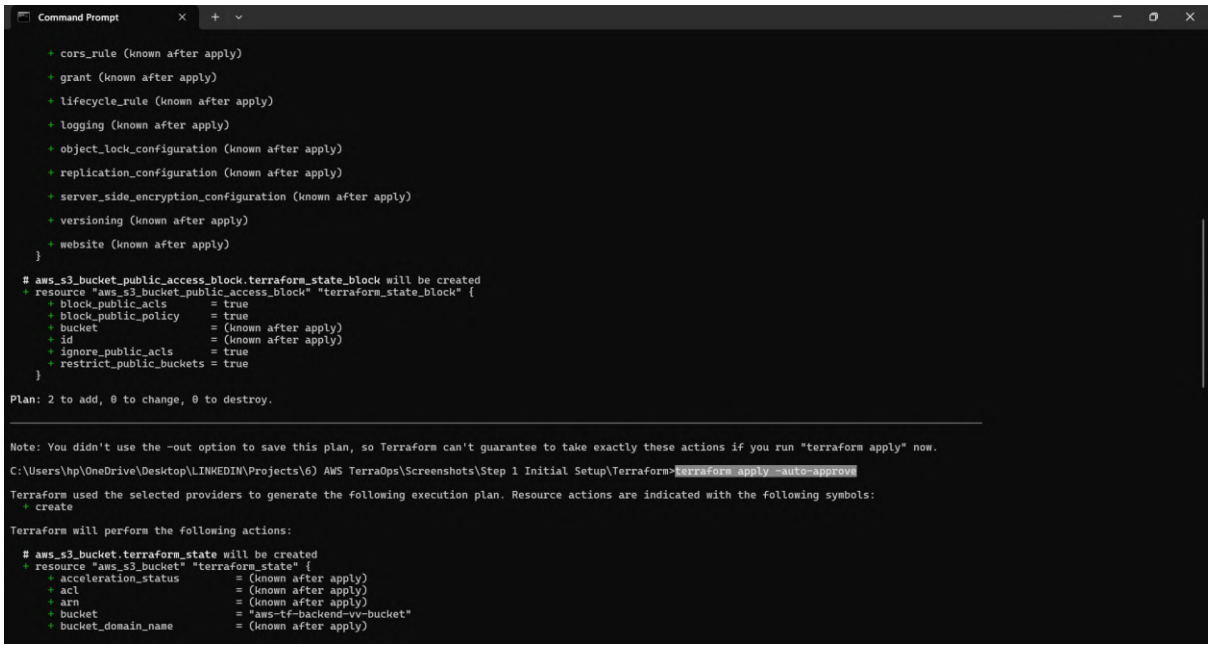

Authorize application
github.com/login/oauth/authorize?response_type=code&client_id=0120e057bd645470c1ed&state=3ac6d1db2793...
Update
Authorize Git Credential Manager
Git Credential Manager by Git Ecosystem
wants to access your VivekVashisht2004 account
Gists
Read and write access
Repositories
Public and private
Workflow
Update GitHub Action Workflow files.
Cancel
Authorize git-ecosystem
Authorizing will redirect to
http://127.0.0.1:60984
Owned & operated by GitHub
Created 5 years ago
More than 1K GitHub users

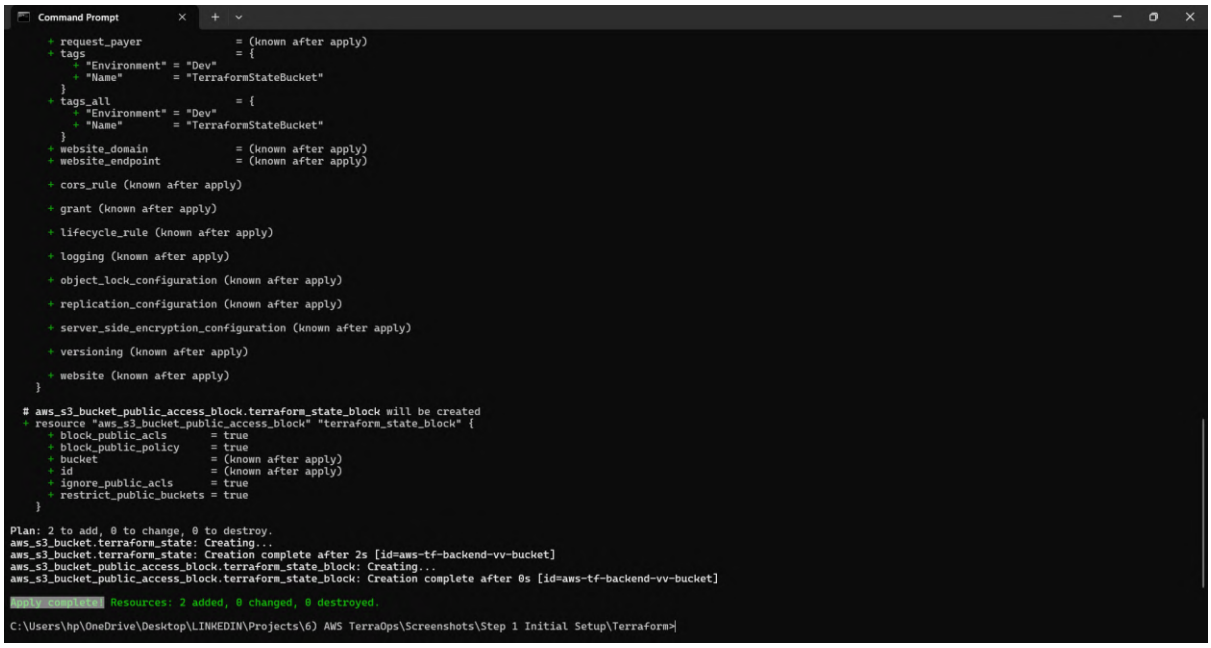

Git Credential Manager - Authe
127.0.0.1:60984/?code=d0d3ef36a3dfb4ed62e78state=3ac6d1db27934c14bef59967145fa091
Update
Authentication Succeeded
You may now close this tab and return to the application.

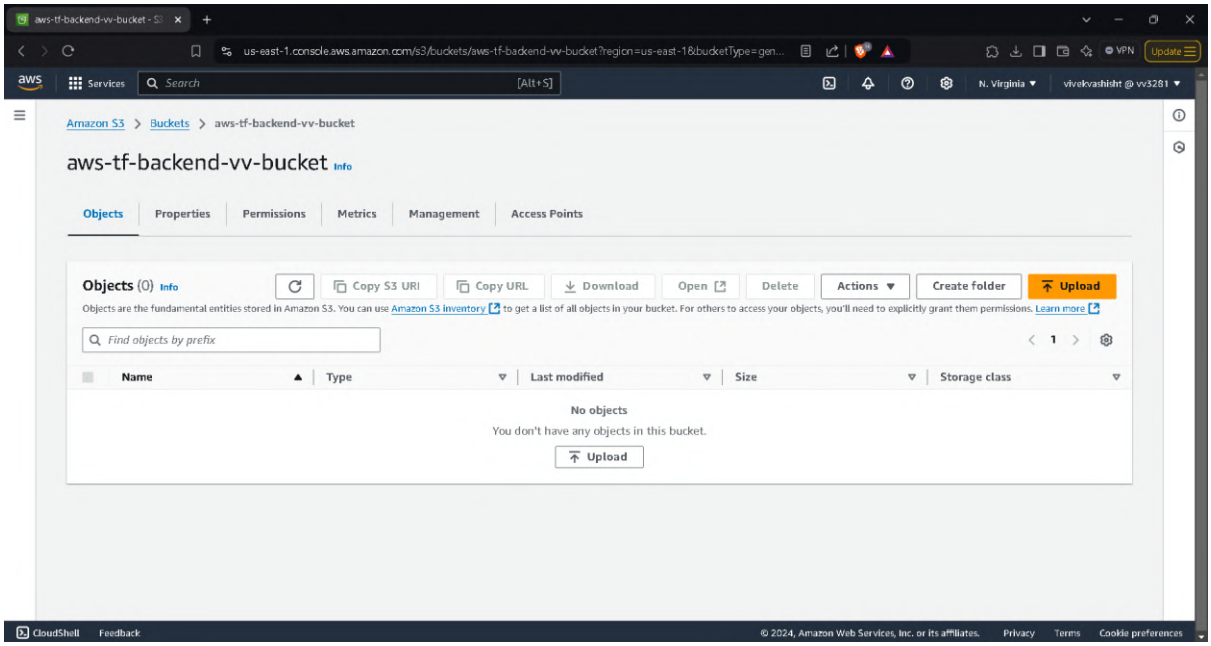
```
C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Git>git push -u origin dev
info: please complete authentication in your browser...
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 260 bytes | 260.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'dev' on GitHub by visiting:
remote:      https://github.com/VivekVashisht2004/AWS-TerraOps/pull/new/dev
remote:
To https://github.com/VivekVashisht2004/AWS-TerraOps.git
 * [new branch]      dev -> dev
branch 'dev' set up to track 'origin/dev'.

C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Git>
```

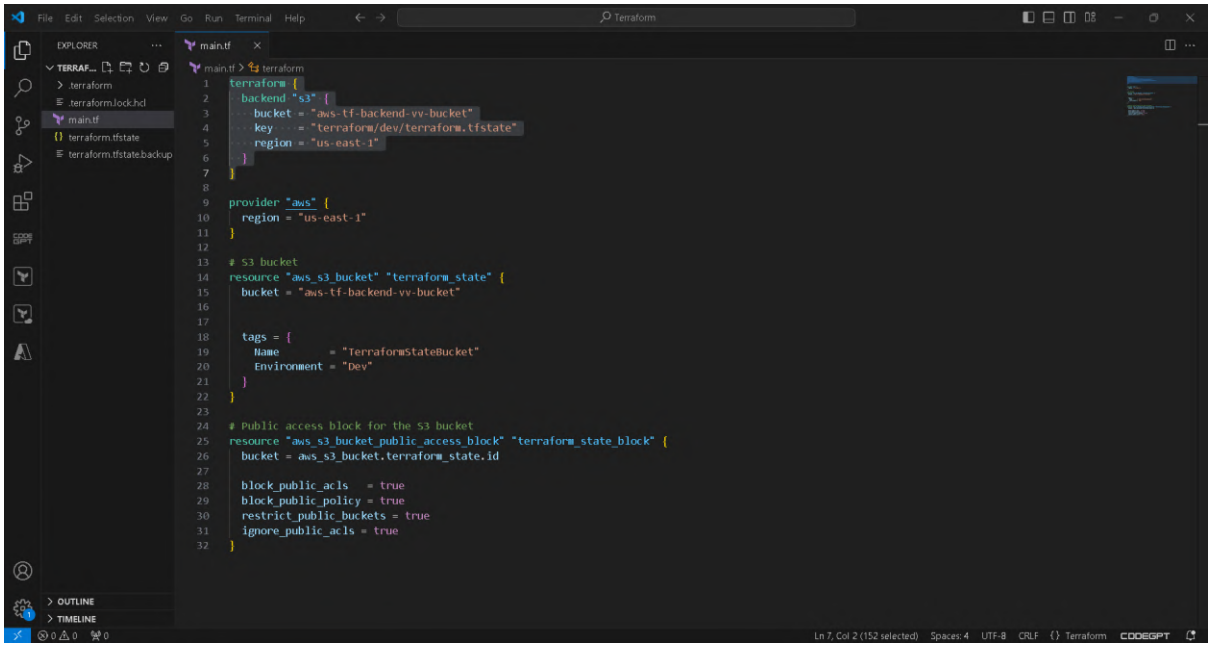



```
Command Prompt

C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.72.1

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.

C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>terraform 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_s3_bucket.terraform_state will be created
  + resource "aws_s3_bucket" "terraform_state" {
      + acceleration_status         = (known after apply)
      + acl                         = (known after apply)
      + arn                         = (known after apply)
      + bucket                      = "aws-tf-backend-vv-bucket"
      + bucket_domain_name          = (known after apply)
      + bucket_prefix               = (known after apply)
      + bucket_regional_domain_name = (known after apply)
      + force_destroy               = false
      + hosted_zone_id              = (known after apply)
      + id                          = (known after apply)
      + object_lock_enabled         = (known after apply)
      + policy                      = (known after apply)
      + region                      = (known after apply)
      + request_payer               = (known after apply)
      + tags                        = {
          + "Environment" = "Dev"
          + "Name"        = "TerraformStateBucket"
        }
      + tags_all                    = {
          + "Environment" = "Dev"
          + "Name"        = "TerraformStateBucket"
        }
      + website_domain              = (known after apply)
      + website_endpoint            = (known after apply)
```

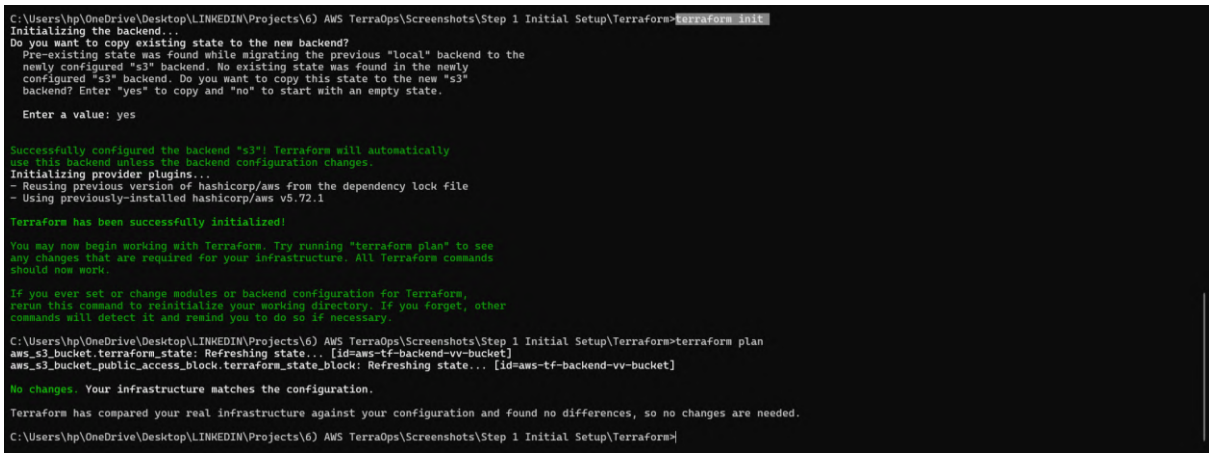

Command Prompt
+ cors_rule (known after apply)
+ grant (known after apply)
+ lifecycle_rule (known after apply)
+ logging (known after apply)
+ object_lock_configuration (known after apply)
+ replication_configuration (known after apply)
+ server_side_encryption_configuration (known after apply)
+ versioning (known after apply)
+ website (known after apply)
}
# aws_s3_bucket_public_access_block.terraform_state_block will be created
+ resource "aws_s3_bucket_public_access_block" "terraform_state_block" {
+ block_public_acls = true
+ block_public_policy = true
+ bucket = (known after apply)
+ id = (known after apply)
+ ignore_public_acls = true
+ restrict_public_buckets = true
}
Plan: 2 to add, 0 to change, 0 to destroy.
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.
C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>terraform apply -auto-approve
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_s3_bucket.terraform_state will be created
+ resource "aws_s3_bucket" "terraform_state" {
+ acceleration_status = (known after apply)
+ acl = (known after apply)
+ arn = (known after apply)
+ bucket = "aws-tf-backend-vv-bucket"
+ bucket_domain_name = (known after apply)


Command Prompt
+ request_payer = (known after apply)
+ tags = {
+ "Environment" = "Dev"
+ "Name" = "TerraformStateBucket"
}
+ tags_all = {
+ "Environment" = "Dev"
+ "Name" = "TerraformStateBucket"
}
+ website_domain = (known after apply)
+ website_endpoint = (known after apply)
+ cors_rule (known after apply)
+ grant (known after apply)
+ lifecycle_rule (known after apply)
+ logging (known after apply)
+ object_lock_configuration (known after apply)
+ replication_configuration (known after apply)
+ server_side_encryption_configuration (known after apply)
+ versioning (known after apply)
+ website (known after apply)
}
# aws_s3_bucket_public_access_block.terraform_state_block will be created
+ resource "aws_s3_bucket_public_access_block" "terraform_state_block" {
+ block_public_acls = true
+ block_public_policy = true
+ bucket = (known after apply)
+ id = (known after apply)
+ ignore_public_acls = true
+ restrict_public_buckets = true
}
Plan: 2 to add, 0 to change, 0 to destroy.
aws_s3_bucket.terraform_state: Creating...
aws_s3_bucket.terraform_state: Creation complete after 2s [id=aws-tf-backend-vv-bucket]
aws_s3_bucket_public_access_block.terraform_state_block: Creating...
aws_s3_bucket_public_access_block.terraform_state_block: Creation complete after 0s [id=aws-tf-backend-vv-bucket]
Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>

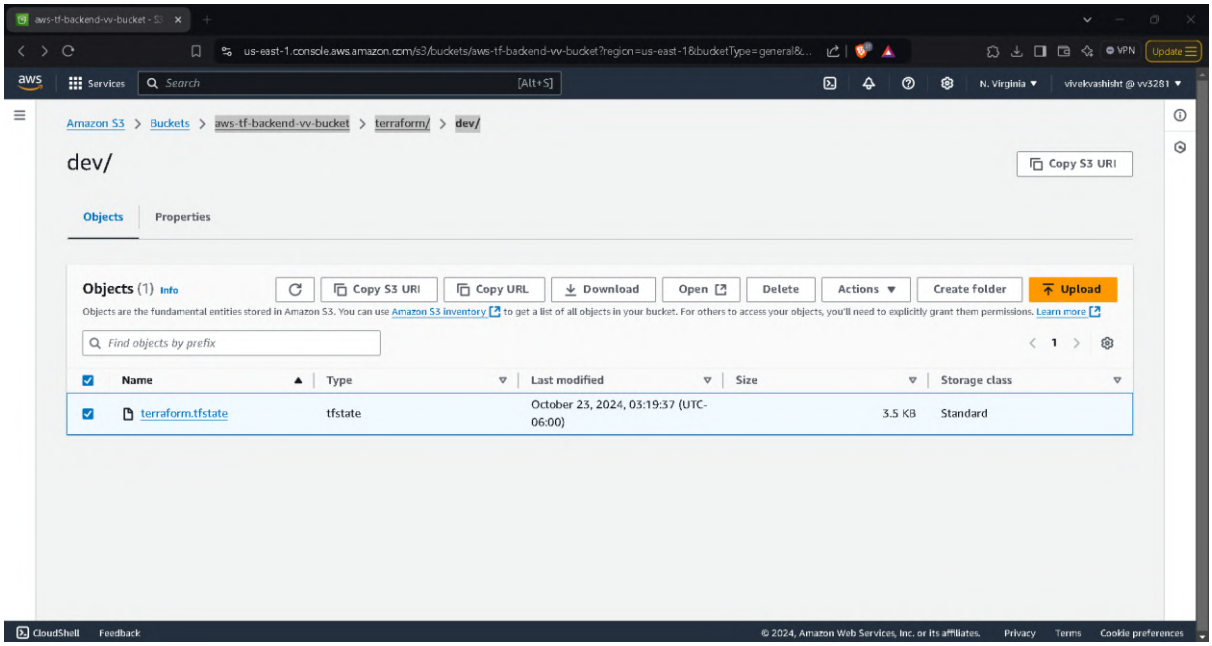

aws-tf-backend-vv-bucket - S3
us-east-1.console.aws.amazon.com/s3/buckets/aws-tf-backend-vv-bucket?region=us-east-1&bucketType=gen...
Update
aws
Services
Search
[Alt+S]
N. Virginia
vivekvashisht @ vv3281
Amazon S3 > Buckets > aws-tf-backend-vv-bucket
aws-tf-backend-vv-bucket Info
Objects
Properties
Permissions
Metrics
Management
Access Points
Objects (0) Info
Copy S3 URI
Copy URL
Download
Open
Delete
Actions
Create folder
Upload
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more
Find objects by prefix
Name
Type
Last modified
Size
Storage class
No objects
You don't have any objects in this bucket.
Upload
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences



```
terraform {
  backend "s3" {
    bucket = "aws-tf-backend-vv-bucket"
    key    = "terraform/dev/terraform.tfstate"
    region = "us-east-1"
  }
}

provider "aws" {
  region = "us-east-1"
}

# S3 bucket
resource "aws_s3_bucket" "terraform_state" {
  bucket = "aws-tf-backend-vv-bucket"


  tags = {
    Name        = "TerraformStateBucket"
    Environment = "Dev"
  }
}

# Public access block for the S3 bucket
resource "aws_s3_bucket_public_access_block" "terraform_state_block" {
  bucket = aws_s3_bucket.terraform_state.id

  block_public_acls   = true
  block_public_policy = true
  restrict_public_buckets = true
  ignore_public_acls = true
}
```



```
C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>terraform init
Initializing the backend...
Do you want to copy existing state to the new backend?
  Pre-existing state was found while migrating the previous "local" backend to the
  newly configured "s3" backend. No existing state was found in the newly
  configured "s3" backend. Do you want to copy this state to the new "s3"
  backend? Enter "yes" to copy and "no" to start with an empty state.

  Enter a value: yes


Successfully configured the backend "s3"! Terraform will automatically
use this backend unless the backend configuration changes.
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.72.1

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.

C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>terraform plan
aws_s3_bucket.terraform_state: Refreshing state... [id=aws-tf-backend-vv-bucket]
aws_s3_bucket_public_access_block.terraform_state_block: Refreshing state... [id=aws-tf-backend-vv-bucket]

No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and found no differences, so no changes are needed.

C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>
```



VPC Creation and Networking

a) Overview

In this phase of the project, I focused on creating a Virtual Private Cloud (VPC) along with its essential networking components, including subnets, an Internet Gateway, and route tables. These resources were defined in a Terraform file named `network.tf`, located in the Terraform project directory. By referencing both the Terraform Registry and AWS Documentation, I ensured each resource was configured correctly to support public and private networking.

b) VPC Setup

The first step was to define the VPC itself. I specified a CIDR block with a `/16` range, allowing for sufficient IP addresses across subnets. The VPC resource block in `network.tf` was as follows:

```
# VPC
resource "aws_vpc" "main_vpc" {
  cidr_block = "10.0.0.0/16"  # VPC with a /16 subnet range

  tags = {
    Name = "MainVPC"
  }
}
```

This created a VPC named `MainVPC` with a large address space, allowing for both public and private subnets to be defined within it.

c) Public and Private Subnet Configuration

Next, I defined both a public and a private subnet within the VPC. The public subnet was set to assign public IPs to instances, making them accessible from the Internet. The private subnet, on the other hand, was configured without public IPs, limiting access to internal networking.

```
# Public Subnet
resource "aws_subnet" "public_subnet_1" {
  vpc_id            = aws_vpc.main_vpc.id
  cidr_block        = "10.0.1.0/24"  # Public subnet's CIDR block
  availability_zone = "us-east-1a"   # Availability zone
  map_public_ip_on_launch = true     # Ensures EC2 instances in this subnet
get a public IP

  tags = {
    Name = "PublicSubnet1"
  }
}

# Private Subnet
```

```
resource "aws_subnet" "private_subnet_1" {
  vpc_id            = aws_vpc.main_vpc.id
  cidr_block        = "10.0.2.0/24"  # Private subnet's CIDR block
  availability_zone = "us-east-1a"   # Same AZ as the public subnet
  map_public_ip_on_launch = false    # Instances in this subnet will not
have public IPs

  tags = {
    Name = "PrivateSubnet1"
  }
}
```

The public subnet was assigned `10.0.1.0/24`, and the private subnet `10.0.2.0/24`, both within the `us-east-1a` availability zone. This segmentation allows for structured and secure network architecture, where only certain resources can be accessed externally.

d) Internet Gateway (IGW)

To enable internet connectivity for resources in the public subnet, I created an Internet Gateway (IGW) and attached it to the VPC.

```
# Internet Gateway (IGW)
resource "aws_internet_gateway" "igw" {
  vpc_id = aws_vpc.main_vpc.id

  tags = {
    Name = "MainIGW"
  }
}
```

The IGW, named `MainIGW`, allows traffic from the internet to reach instances within the public subnet when properly routed.

e) Public Route Table and Routing

To direct internet-bound traffic from the public subnet, I created a public route table and associated it with the subnet. The route table includes a route to forward all outbound traffic (`0.0.0.0/0`) to the Internet Gateway.

```
# Public Route Table and Route
resource "aws_route_table" "public_rt" {
  vpc_id = aws_vpc.main_vpc.id

  tags = {
    Name = "PublicRouteTable"
  }
}

resource "aws_route" "internet_access" {
  route_table_id         = aws_route_table.public_rt.id
  destination_cidr_block = "0.0.0.0/0"  # Sends traffic to the internet
  gateway_id             = aws_internet_gateway.igw.id
}

resource "aws_route_table_association" "public_subnet_assoc" {
  subnet_id      = aws_subnet.public_subnet_1.id
```

```
  route_table_id = aws_route_table.public_rt.id
}
```

This setup enables instances within `PublicSubnet1` to send and receive internet traffic through the IGW, as specified in the `PublicRouteTable`.

f) network.tf Summary

The complete `network.tf` file, with all configurations, looked as follows:

```
# VPC
resource "aws_vpc" "main_vpc" {
  cidr_block = "10.0.0.0/16"  # VPC with a /16 subnet range

  tags = {
    Name = "MainVPC"
  }
}

# Public Subnet
resource "aws_subnet" "public_subnet_1" {
  vpc_id            = aws_vpc.main_vpc.id
  cidr_block        = "10.0.1.0/24"
  availability_zone = "us-east-1a"
  map_public_ip_on_launch = true

  tags = {
    Name = "PublicSubnet1"
  }
}

# Private Subnet
resource "aws_subnet" "private_subnet_1" {
  vpc_id            = aws_vpc.main_vpc.id
  cidr_block        = "10.0.2.0/24"
  availability_zone = "us-east-1a"
  map_public_ip_on_launch = false

  tags = {
    Name = "PrivateSubnet1"
  }
}

# Internet Gateway (IGW)
resource "aws_internet_gateway" "igw" {
  vpc_id = aws_vpc.main_vpc.id

  tags = {
    Name = "MainIGW"
  }
}

# Public Route Table and Route
resource "aws_route_table" "public_rt" {
  vpc_id = aws_vpc.main_vpc.id

  tags = {
    Name = "PublicRouteTable"
  }
```

```
}

resource "aws_route" "internet_access" {
  route_table_id         = aws_route_table.public_rt.id
  destination_cidr_block = "0.0.0.0/0"
  gateway_id             = aws_internet_gateway.igw.id
}

resource "aws_route_table_association" "public_subnet_assoc" {
  subnet_id      = aws_subnet.public_subnet_1.id
  route_table_id = aws_route_table.public_rt.id
}
```

g) Execution and Verification

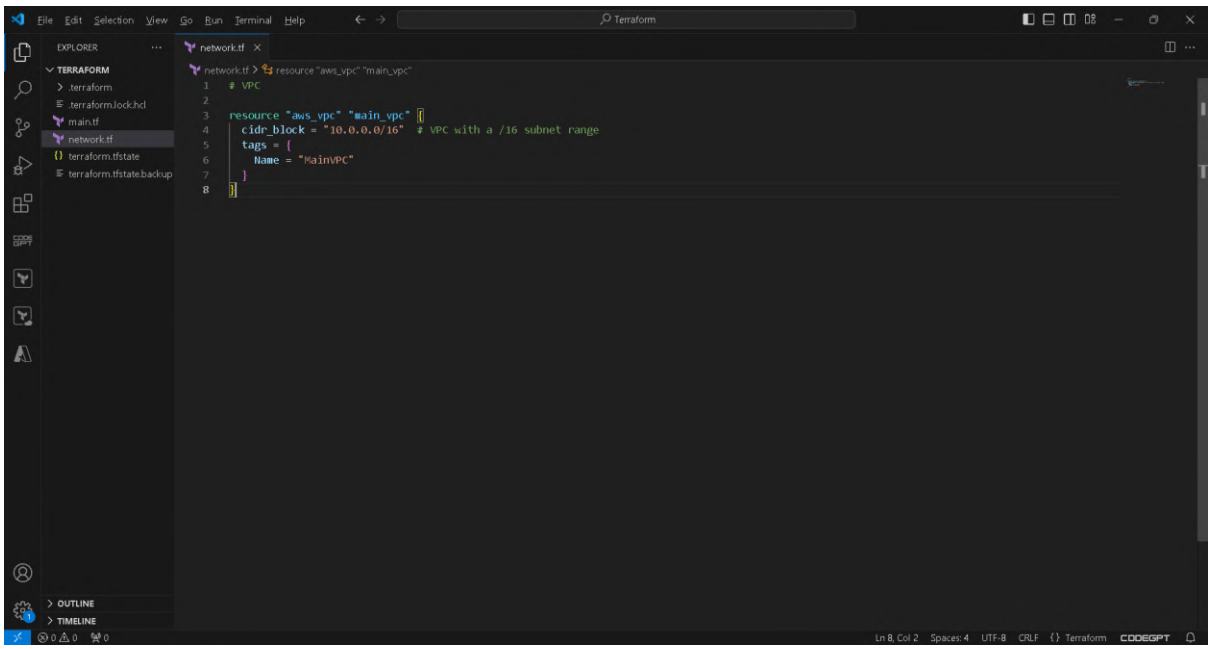
After saving `network.tf`, I proceeded with the following commands in CMD to initialize, plan, and apply the configuration:

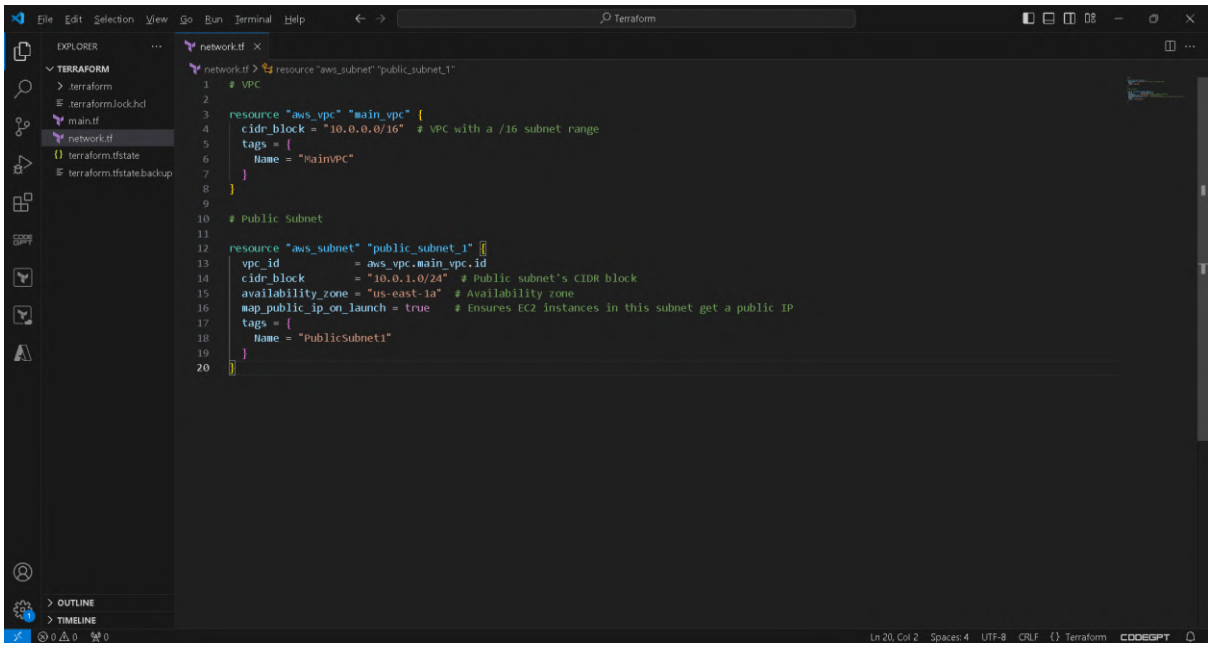
```
terraform init
terraform plan
terraform apply -auto-approve
```

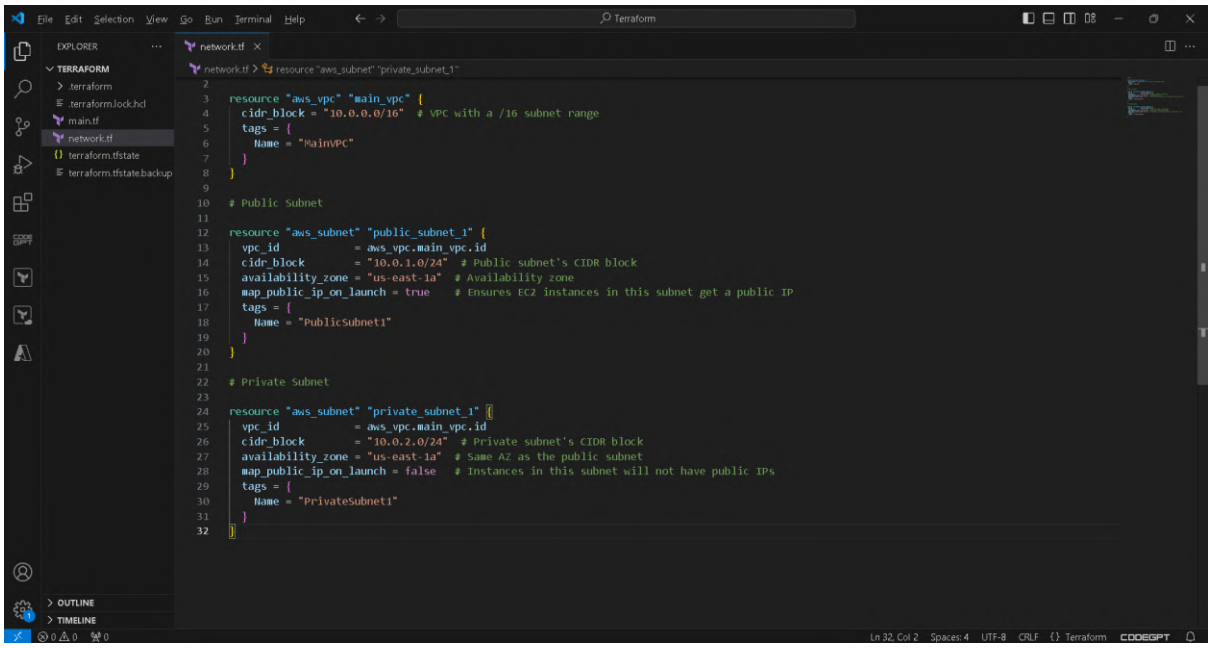
Upon confirmation and successful application, I verified the resources in the AWS Console. I checked that all resources, including `MainVPC`, `PublicSubnet1`, `PrivateSubnet1`, `MainIGW`, `PublicRouteTable`, and the associated routes and subnets, were created as defined in the `network.tf` file.

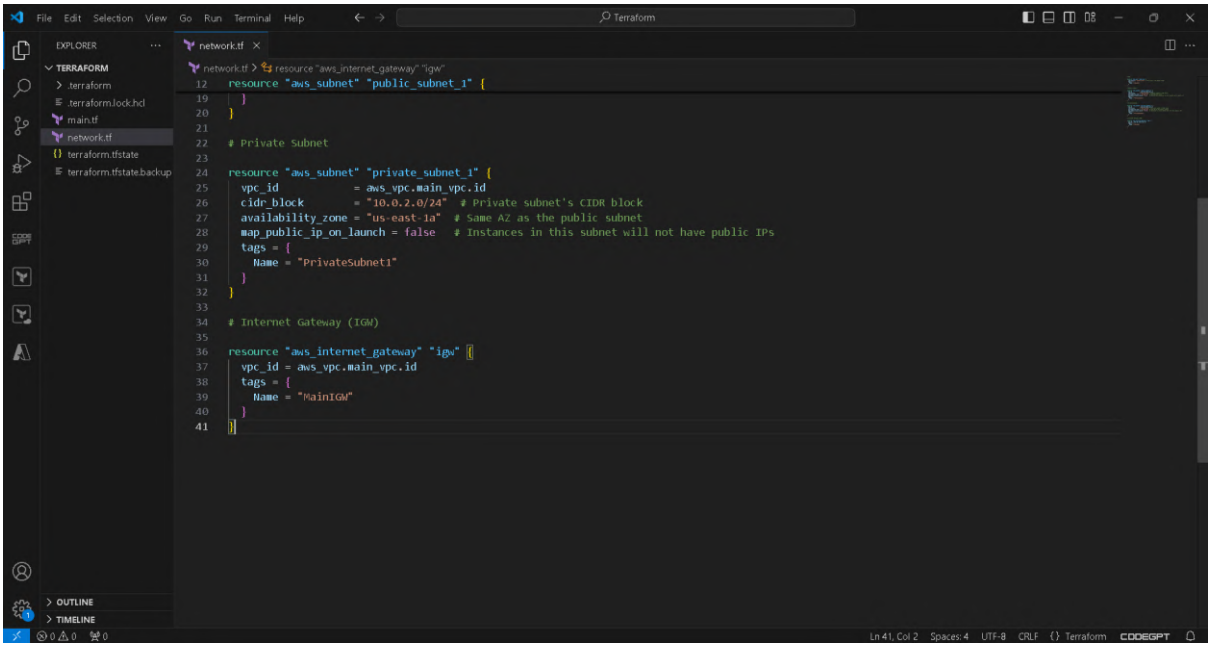
This phase successfully established the foundational networking infrastructure for the project, allowing public and private instances to operate securely within a structured network environment.

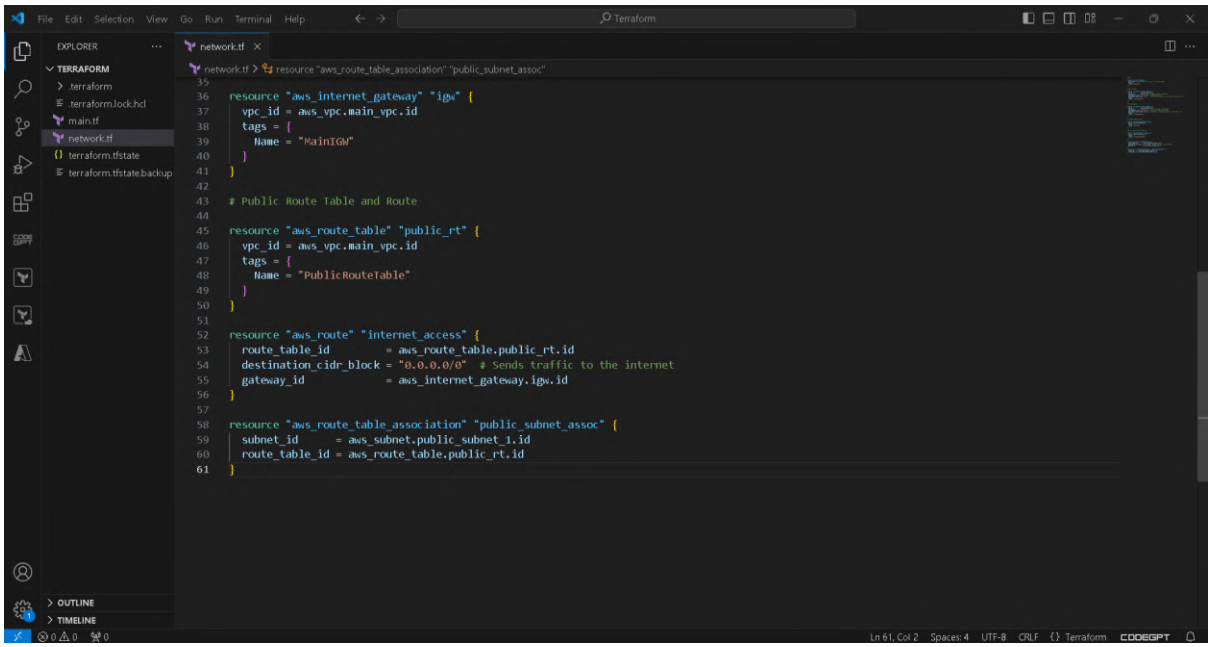
Screenshots



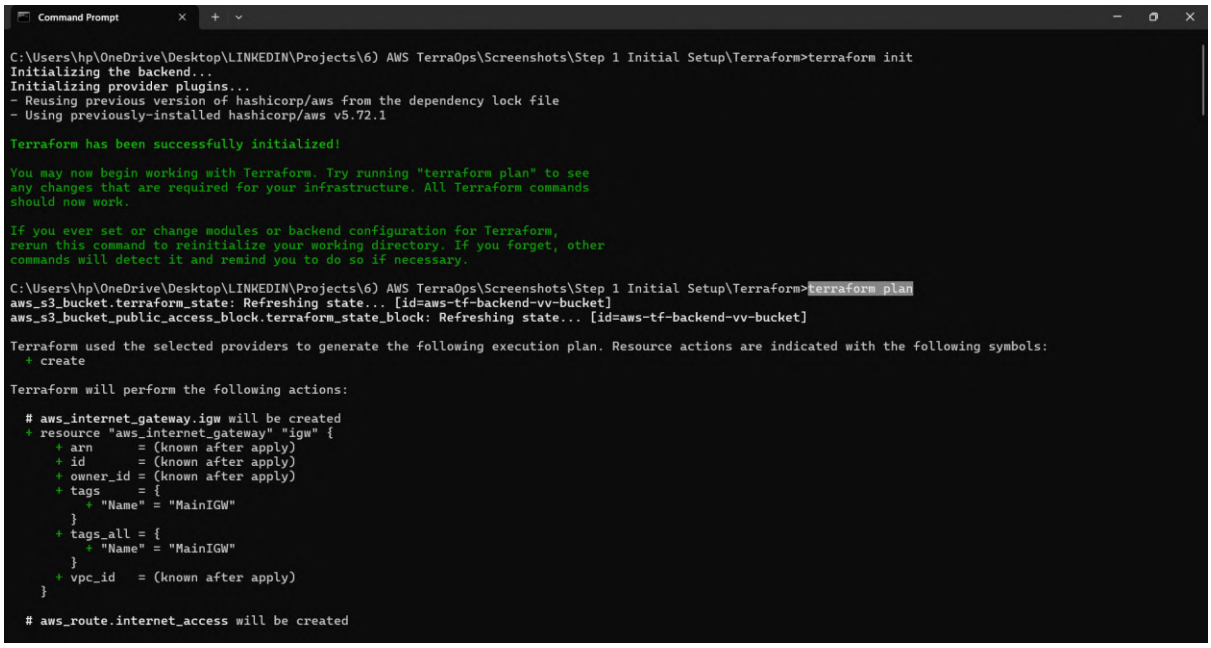

network.tf
resource "aws_subnet" "public_subnet_1"
# VPC
resource "aws_vpc" "main_vpc" {
cidr_block = "10.0.0.0/16" # VPC with a /16 subnet range
tags = {
Name = "MainVPC"
}
}
# Public Subnet
resource "aws_subnet" "public_subnet_1" {
vpc_id = aws_vpc.main_vpc.id
cidr_block = "10.0.1.0/24" # Public subnet's CIDR block
availability_zone = "us-east-1a" # Availability zone
map_public_ip_on_launch = true # Ensures EC2 instances in this subnet get a public IP
tags = {
Name = "PublicSubnet1"
}
}


network.tf
resource "aws_subnet" "private_subnet_1"
resource "aws_vpc" "main_vpc" {
cidr_block = "10.0.0.0/16" # VPC with a /16 subnet range
tags = {
Name = "MainVPC"
}
}
# Public Subnet
resource "aws_subnet" "public_subnet_1" {
vpc_id = aws_vpc.main_vpc.id
cidr_block = "10.0.1.0/24" # Public subnet's CIDR block
availability_zone = "us-east-1a" # Availability zone
map_public_ip_on_launch = true # Ensures EC2 instances in this subnet get a public IP
tags = {
Name = "PublicSubnet1"
}
}
# Private Subnet
resource "aws_subnet" "private_subnet_1" {
vpc_id = aws_vpc.main_vpc.id
cidr_block = "10.0.2.0/24" # Private subnet's CIDR block
availability_zone = "us-east-1a" # Same AZ as the public subnet
map_public_ip_on_launch = false # Instances in this subnet will not have public IPs
tags = {
Name = "PrivateSubnet1"
}
}

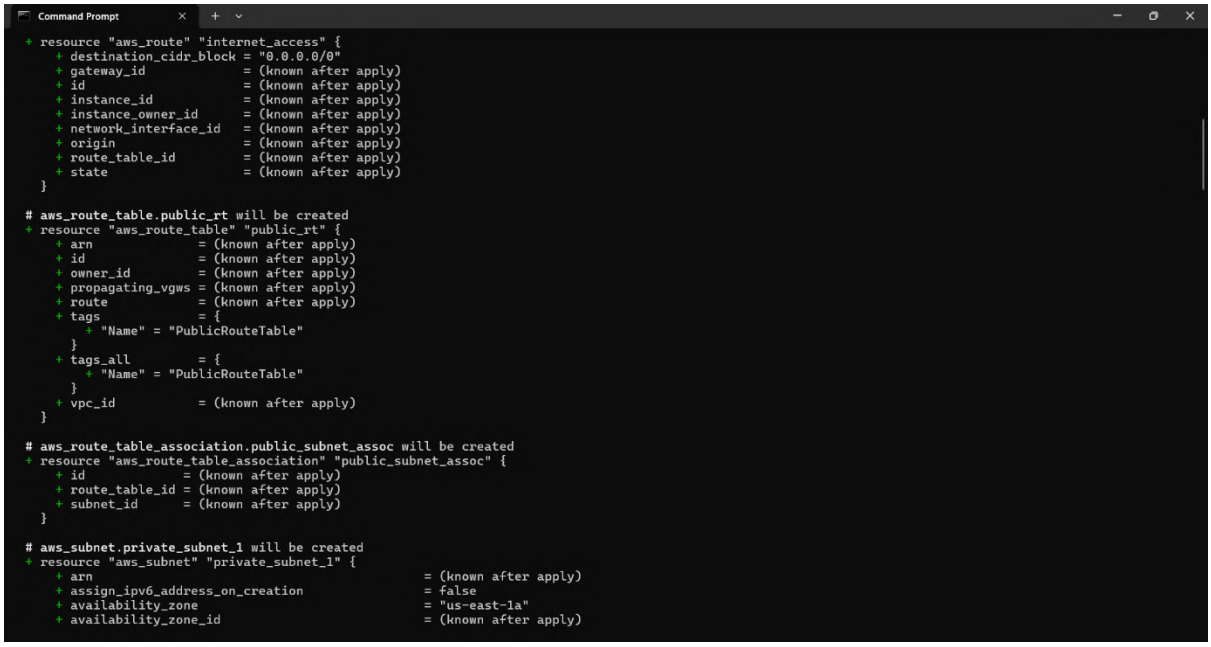

network.tf
resource "aws_internet_gateway" "igw"
resource "aws_subnet" "public_subnet_1" {
# Private Subnet
resource "aws_subnet" "private_subnet_1" {
vpc_id = aws_vpc.main_vpc.id
cidr_block = "10.0.2.0/24" # Private subnet's CIDR block
availability_zone = "us-east-1a" # Same AZ as the public subnet
map_public_ip_on_launch = false # Instances in this subnet will not have public IPs
tags = {
Name = "PrivateSubnet1"
}
}
# Internet Gateway (IGW)
resource "aws_internet_gateway" "igw" {
vpc_id = aws_vpc.main_vpc.id
tags = {
Name = "MainIGW"
}
}



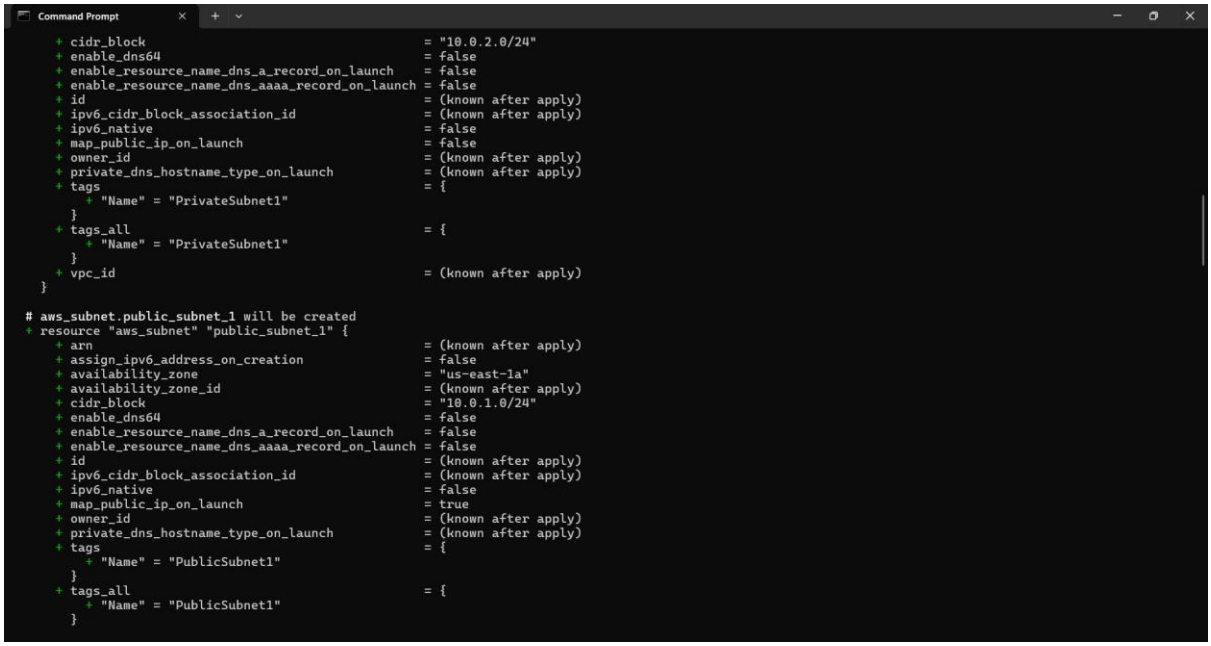
```
resource "aws_internet_gateway" "igw" {
  vpc_id = aws_vpc.main_vpc.id
  tags = {
    Name = "MainIGW"
  }
}

# Public Route Table and Route

resource "aws_route_table" "public_rt" {
  vpc_id = aws_vpc.main_vpc.id
  tags = {
    Name = "PublicRouteTable"
  }
}

resource "aws_route" "internet_access" {
  route_table_id         = aws_route_table.public_rt.id
  destination_cidr_block = "0.0.0.0/0"  # Sends traffic to the internet
  gateway_id             = aws_internet_gateway.igw.id
}

resource "aws_route_table_association" "public_subnet_assoc" {
  subnet_id      = aws_subnet.public_subnet_1.id
  route_table_id = aws_route_table.public_rt.id
}
```



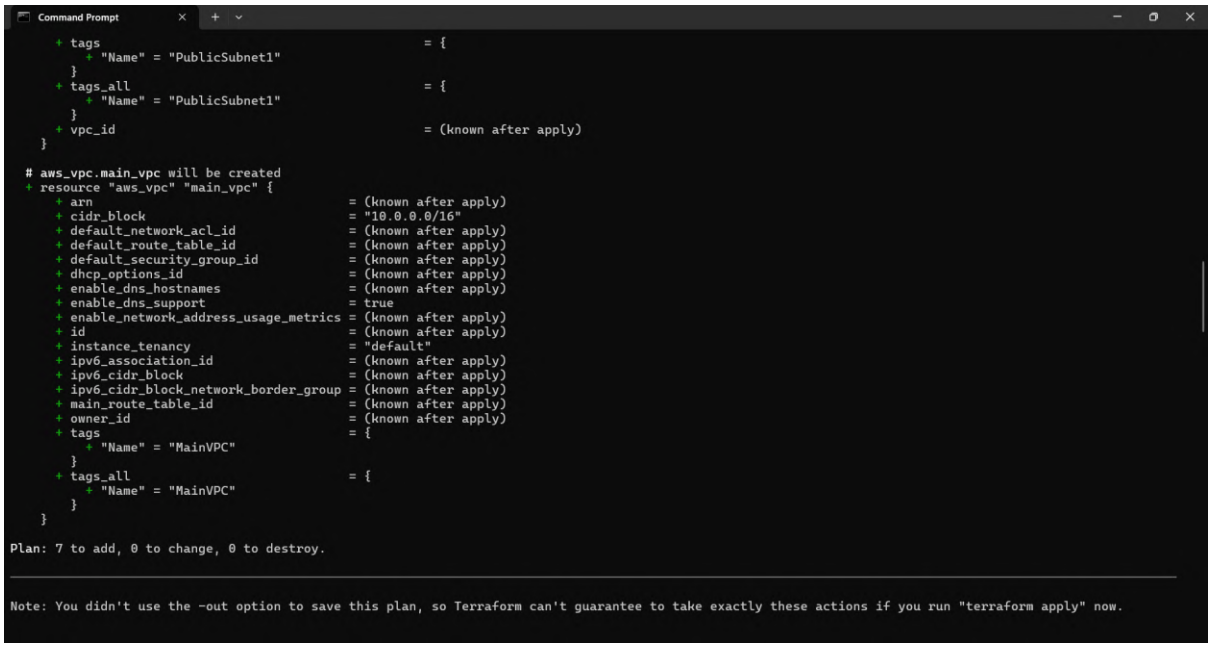
```
C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.72.1

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.

C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>terraform plan
aws_s3_bucket.terraform_state: Refreshing state... [id=aws-tf-backend-vv-bucket]
aws_s3_bucket_public_access_block.terraform_state_block: Refreshing state... [id=aws-tf-backend-vv-bucket]

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_internet_gateway.igw will be created
  + resource "aws_internet_gateway" "igw" {
      + arn      = (known after apply)
      + id       = (known after apply)
      + owner_id = (known after apply)
      + tags     = {
          + "Name" = "MainIGW"
        }
      + tags_all = {
          + "Name" = "MainIGW"
        }
      + vpc_id   = (known after apply)
    }

  # aws_route.internet_access will be created
```



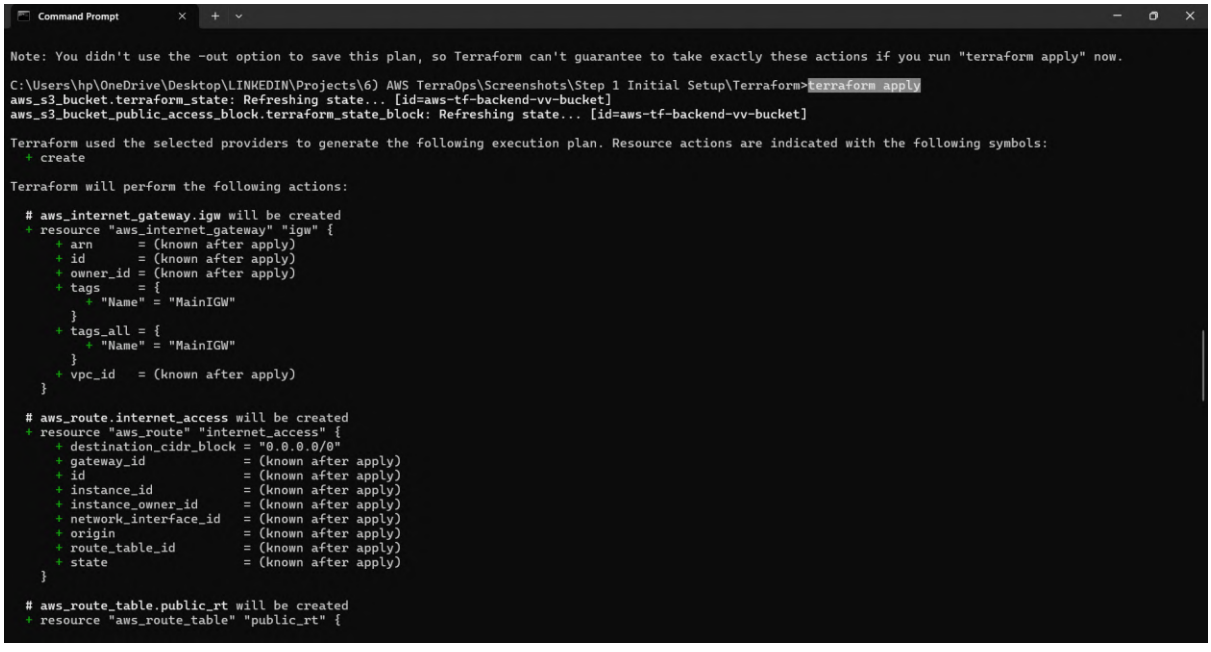
```
  + resource "aws_route" "internet_access" {
      + destination_cidr_block = "0.0.0.0/0"
      + gateway_id             = (known after apply)
      + id                     = (known after apply)
      + instance_id            = (known after apply)
      + instance_owner_id      = (known after apply)
      + network_interface_id   = (known after apply)
      + origin                 = (known after apply)
      + route_table_id         = (known after apply)
      + state                  = (known after apply)
    }

  # aws_route_table.public_rt will be created
  + resource "aws_route_table" "public_rt" {
      + arn              = (known after apply)
      + id               = (known after apply)
      + owner_id         = (known after apply)
      + propagating_vgws = (known after apply)
      + route            = (known after apply)
      + tags             = {
          + "Name" = "PublicRouteTable"
        }
      + tags_all         = {
          + "Name" = "PublicRouteTable"
        }
      + vpc_id           = (known after apply)
    }

  # aws_route_table_association.public_subnet_assoc will be created
  + resource "aws_route_table_association" "public_subnet_assoc" {
      + id             = (known after apply)
      + route_table_id = (known after apply)
      + subnet_id      = (known after apply)
    }

  # aws_subnet.private_subnet_1 will be created
  + resource "aws_subnet" "private_subnet_1" {
      + arn                                            = (known after apply)
      + assign_ipv6_address_on_creation                = false
      + availability_zone                              = "us-east-1a"
      + availability_zone_id                           = (known after apply)
```



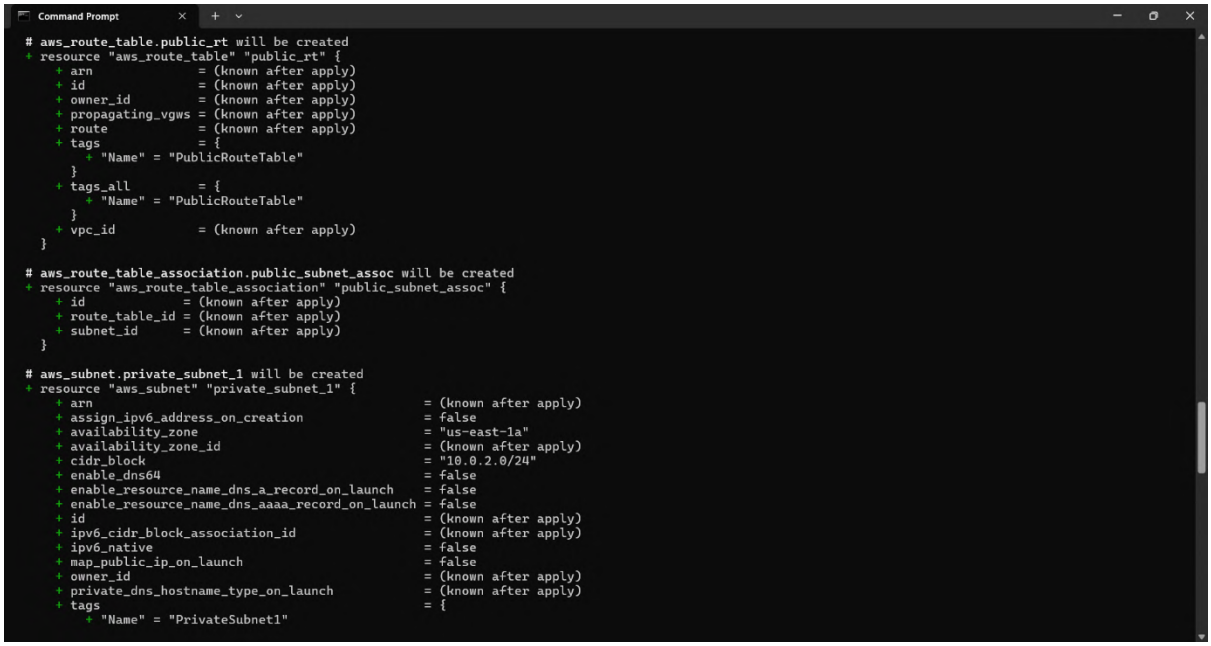
```
Command Prompt
      + cidr_block                                     = "10.0.2.0/24"
      + enable_dns64                                   = false
      + enable_resource_name_dns_a_record_on_launch    = false
      + enable_resource_name_dns_aaaa_record_on_launch = false
      + id                                             = (known after apply)
      + ipv6_cidr_block_association_id                 = (known after apply)
      + ipv6_native                                    = false
      + map_public_ip_on_launch                        = false
      + owner_id                                       = (known after apply)
      + private_dns_hostname_type_on_launch            = (known after apply)
      + tags                                           = {
          + "Name" = "PrivateSubnet1"
        }
      + tags_all                                       = {
          + "Name" = "PrivateSubnet1"
        }
      + vpc_id                                         = (known after apply)
    }

  # aws_subnet.public_subnet_1 will be created
  + resource "aws_subnet" "public_subnet_1" {
      + arn                                            = (known after apply)
      + assign_ipv6_address_on_creation                = false
      + availability_zone                              = "us-east-1a"
      + availability_zone_id                           = (known after apply)
      + cidr_block                                     = "10.0.1.0/24"
      + enable_dns64                                   = false
      + enable_resource_name_dns_a_record_on_launch    = false
      + enable_resource_name_dns_aaaa_record_on_launch = false
      + id                                             = (known after apply)
      + ipv6_cidr_block_association_id                 = (known after apply)
      + ipv6_native                                    = false
      + map_public_ip_on_launch                        = true
      + owner_id                                       = (known after apply)
      + private_dns_hostname_type_on_launch            = (known after apply)
      + tags                                           = {
          + "Name" = "PublicSubnet1"
        }
      + tags_all                                       = {
          + "Name" = "PublicSubnet1"
        }
```



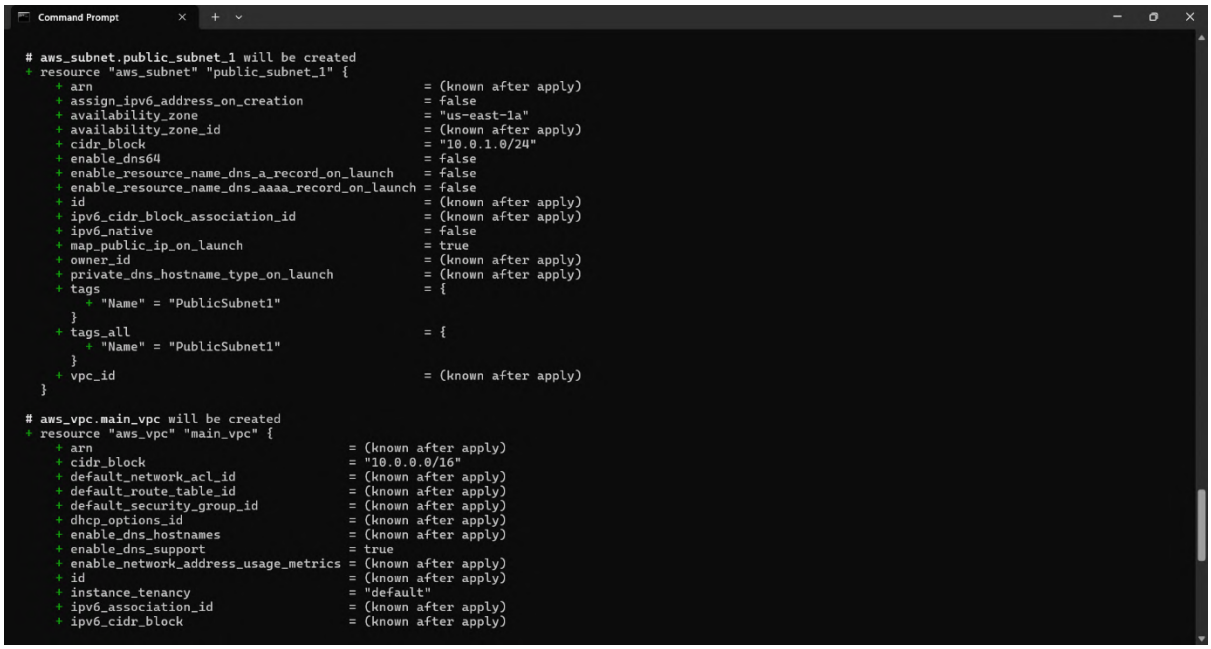
```
Command Prompt
      + tags                                           = {
          + "Name" = "PublicSubnet1"
        }
      + tags_all                                       = {
          + "Name" = "PublicSubnet1"
        }
      + vpc_id                                         = (known after apply)
    }

  # aws_vpc.main_vpc will be created
  + resource "aws_vpc" "main_vpc" {
      + arn                                = (known after apply)
      + cidr_block                         = "10.0.0.0/16"
      + default_network_acl_id             = (known after apply)
      + default_route_table_id             = (known after apply)
      + default_security_group_id          = (known after apply)
      + dhcp_options_id                    = (known after apply)
      + enable_dns_hostnames               = (known after apply)
      + enable_dns_support                 = true
      + enable_network_address_usage_metrics = (known after apply)
      + id                                 = (known after apply)
      + instance_tenancy                   = "default"
      + ipv6_association_id                = (known after apply)
      + ipv6_cidr_block                    = (known after apply)
      + ipv6_cidr_block_network_border_group = (known after apply)
      + main_route_table_id                = (known after apply)
      + owner_id                           = (known after apply)
      + tags                               = {
          + "Name" = "MainVPC"
        }
      + tags_all                           = {
          + "Name" = "MainVPC"
        }
    }

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

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.
```



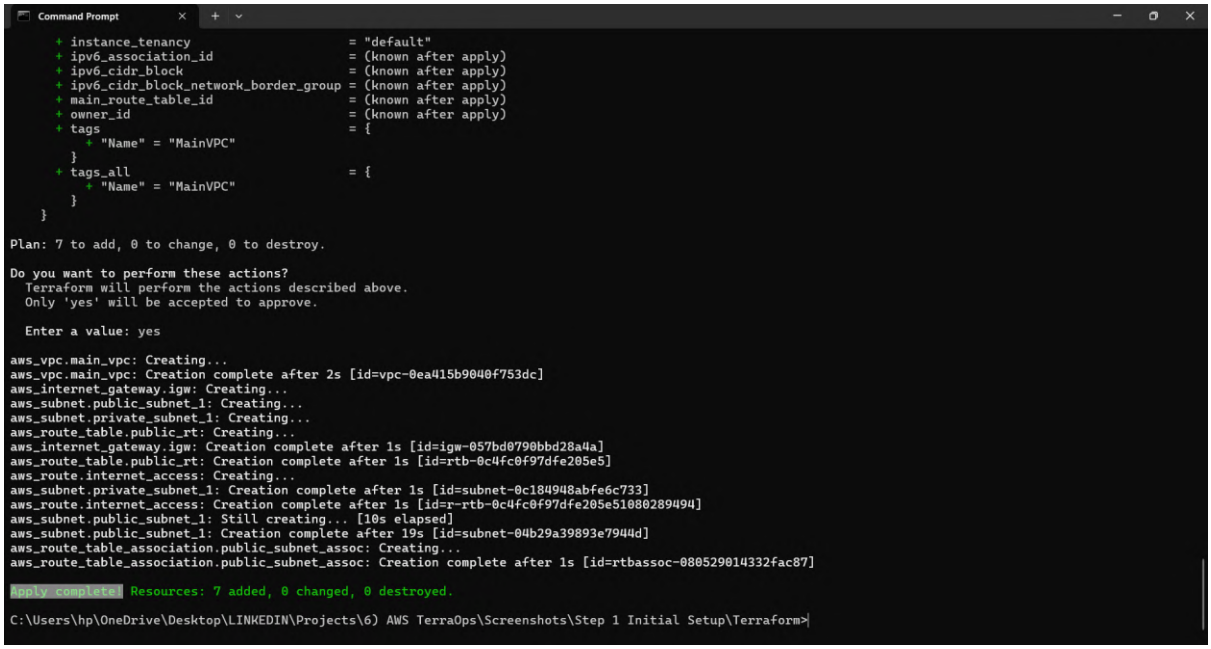
```
Command Prompt

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.

C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>terraform apply
aws_s3_bucket.terraform_state: Refreshing state... [id=aws-tf-backend-vv-bucket]
aws_s3_bucket_public_access_block.terraform_state_block: Refreshing state... [id=aws-tf-backend-vv-bucket]

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_internet_gateway.igw will be created
  + resource "aws_internet_gateway" "igw" {
      + arn      = (known after apply)
      + id       = (known after apply)
      + owner_id = (known after apply)
      + tags     = {
          + "Name" = "MainIGW"
        }
      + tags_all = {
          + "Name" = "MainIGW"
        }
      + vpc_id   = (known after apply)
    }

  # aws_route.internet_access will be created
  + resource "aws_route" "internet_access" {
      + destination_cidr_block = "0.0.0.0/0"
      + gateway_id             = (known after apply)
      + id                     = (known after apply)
      + instance_id            = (known after apply)
      + instance_owner_id      = (known after apply)
      + network_interface_id   = (known after apply)
      + origin                 = (known after apply)
      + route_table_id         = (known after apply)
      + state                  = (known after apply)
    }

  # aws_route_table.public_rt will be created
  + resource "aws_route_table" "public_rt" {
```



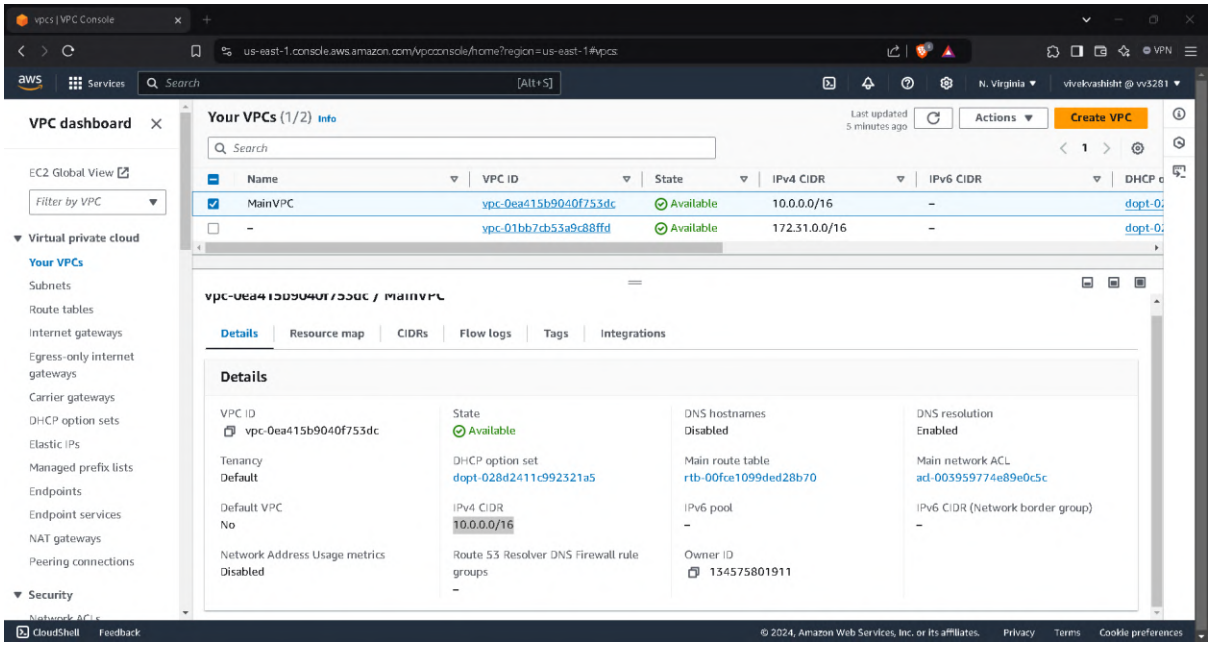
```
Command Prompt

  # aws_route_table.public_rt will be created
  + resource "aws_route_table" "public_rt" {
      + arn              = (known after apply)
      + id               = (known after apply)
      + owner_id         = (known after apply)
      + propagating_vgws = (known after apply)
      + route            = (known after apply)
      + tags             = {
          + "Name" = "PublicRouteTable"
        }
      + tags_all         = {
          + "Name" = "PublicRouteTable"
        }
      + vpc_id           = (known after apply)
    }

  # aws_route_table_association.public_subnet_assoc will be created
  + resource "aws_route_table_association" "public_subnet_assoc" {
      + id             = (known after apply)
      + route_table_id = (known after apply)
      + subnet_id      = (known after apply)
    }

  # aws_subnet.private_subnet_1 will be created
  + resource "aws_subnet" "private_subnet_1" {
      + arn                                            = (known after apply)
      + assign_ipv6_address_on_creation                = false
      + availability_zone                              = "us-east-1a"
      + availability_zone_id                           = (known after apply)
      + cidr_block                                     = "10.0.2.0/24"
      + enable_dns64                                   = false
      + enable_resource_name_dns_a_record_on_launch    = false
      + enable_resource_name_dns_aaaa_record_on_launch = false
      + id                                             = (known after apply)
      + ipv6_cidr_block_association_id                 = (known after apply)
      + ipv6_native                                    = false
      + map_public_ip_on_launch                        = false
      + owner_id                                       = (known after apply)
      + private_dns_hostname_type_on_launch            = (known after apply)
      + tags                                           = {
          + "Name" = "PrivateSubnet1"
```

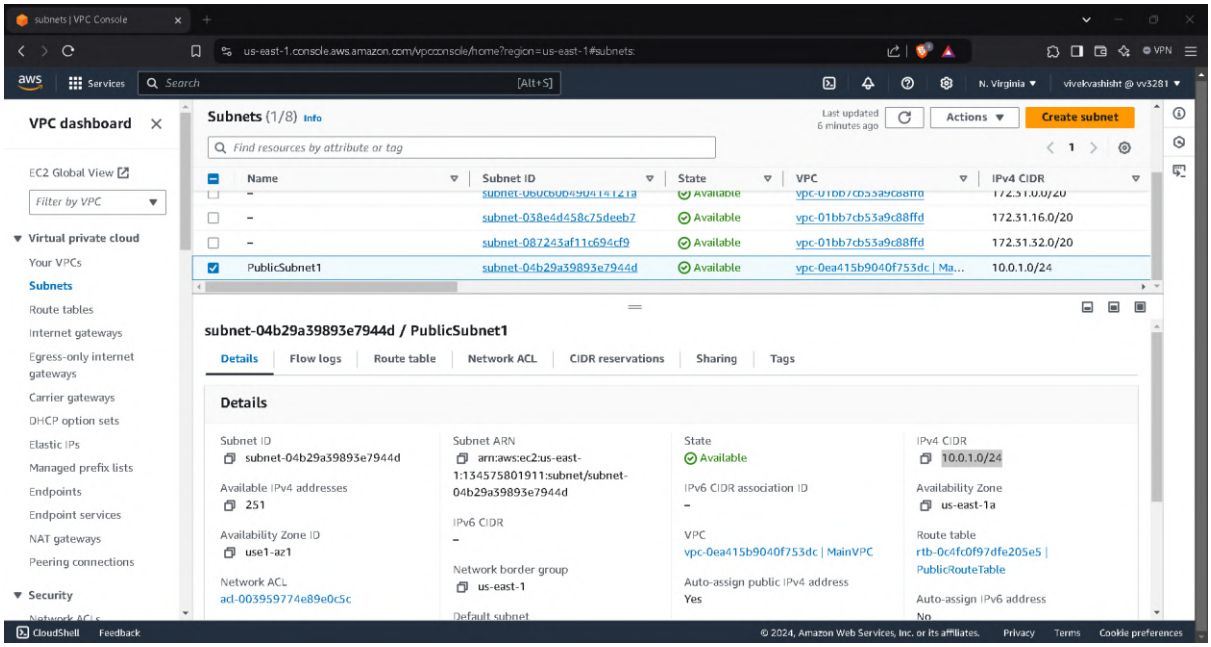


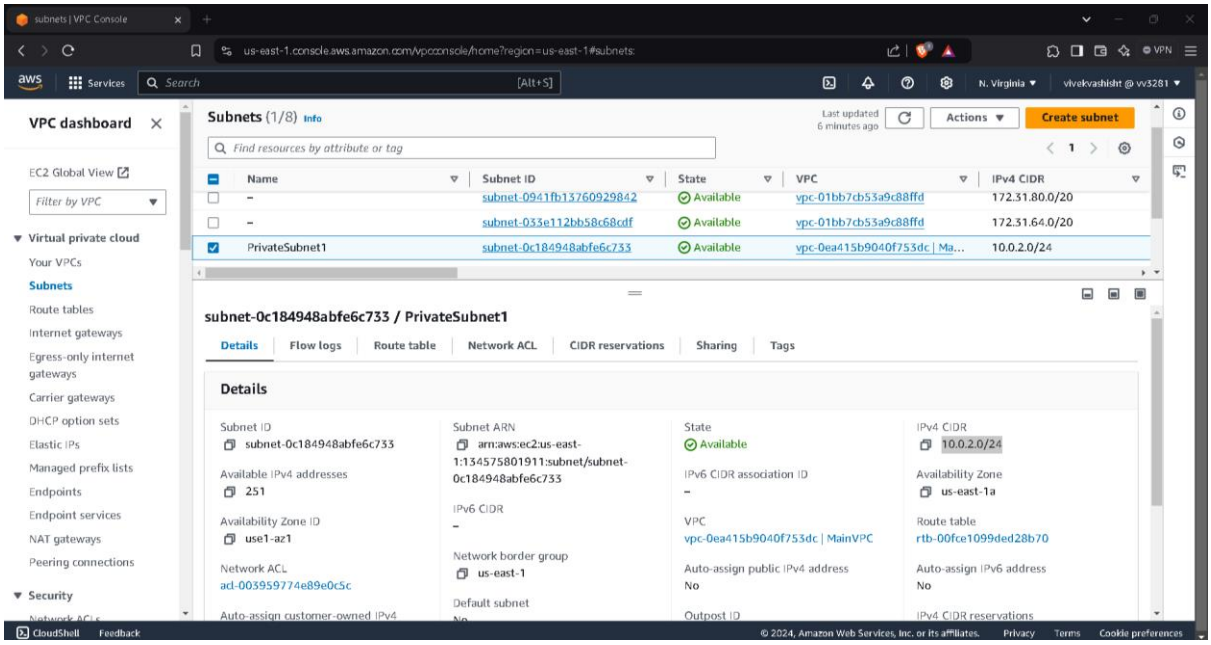
```
Command Prompt

  # aws_subnet.public_subnet_1 will be created
  + resource "aws_subnet" "public_subnet_1" {
      + arn                                            = (known after apply)
      + assign_ipv6_address_on_creation                = false
      + availability_zone                              = "us-east-1a"
      + availability_zone_id                           = (known after apply)
      + cidr_block                                     = "10.0.1.0/24"
      + enable_dns64                                   = false
      + enable_resource_name_dns_a_record_on_launch    = false
      + enable_resource_name_dns_aaaa_record_on_launch = false
      + id                                             = (known after apply)
      + ipv6_cidr_block_association_id                 = (known after apply)
      + ipv6_native                                    = false
      + map_public_ip_on_launch                        = true
      + owner_id                                       = (known after apply)
      + private_dns_hostname_type_on_launch            = (known after apply)
      + tags                                           = {
          + "Name" = "PublicSubnet1"
        }
      + tags_all                                       = {
          + "Name" = "PublicSubnet1"
        }
      + vpc_id                                         = (known after apply)
    }

  # aws_vpc.main_vpc will be created
  + resource "aws_vpc" "main_vpc" {
      + arn                                = (known after apply)
      + cidr_block                         = "10.0.0.0/16"
      + default_network_acl_id             = (known after apply)
      + default_route_table_id             = (known after apply)
      + default_security_group_id          = (known after apply)
      + dhcp_options_id                    = (known after apply)
      + enable_dns_hostnames               = (known after apply)
      + enable_dns_support                 = true
      + enable_network_address_usage_metrics = (known after apply)
      + id                                 = (known after apply)
      + instance_tenancy                   = "default"
      + ipv6_association_id                = (known after apply)
      + ipv6_cidr_block                    = (known after apply)
```

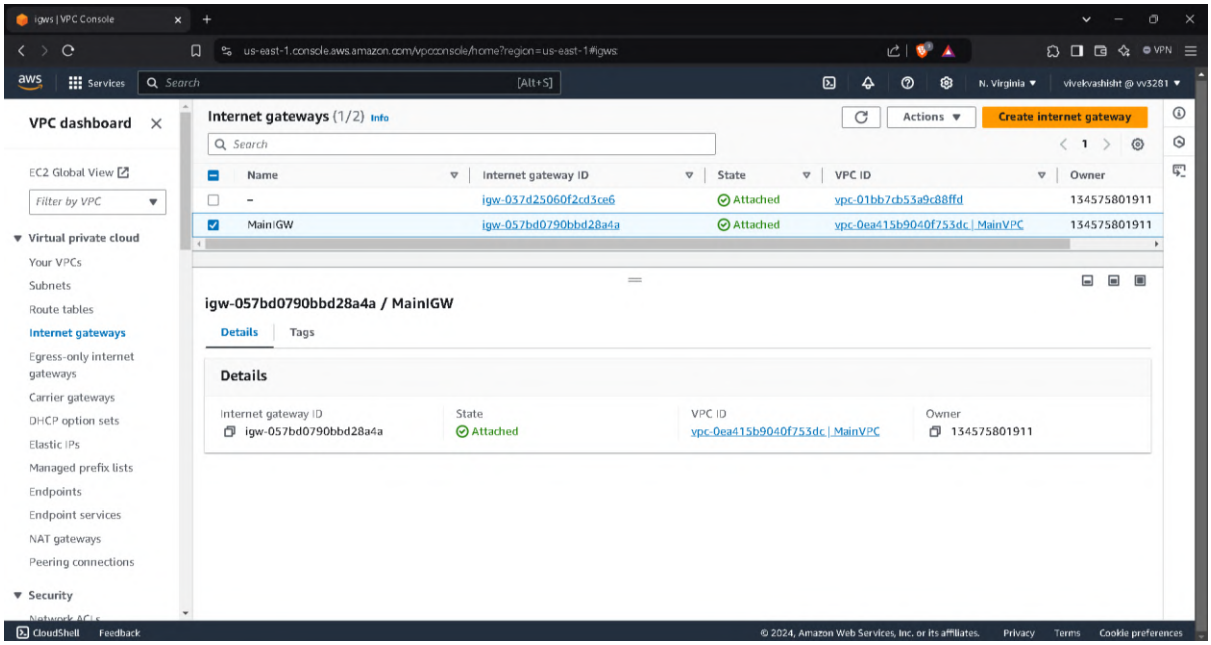


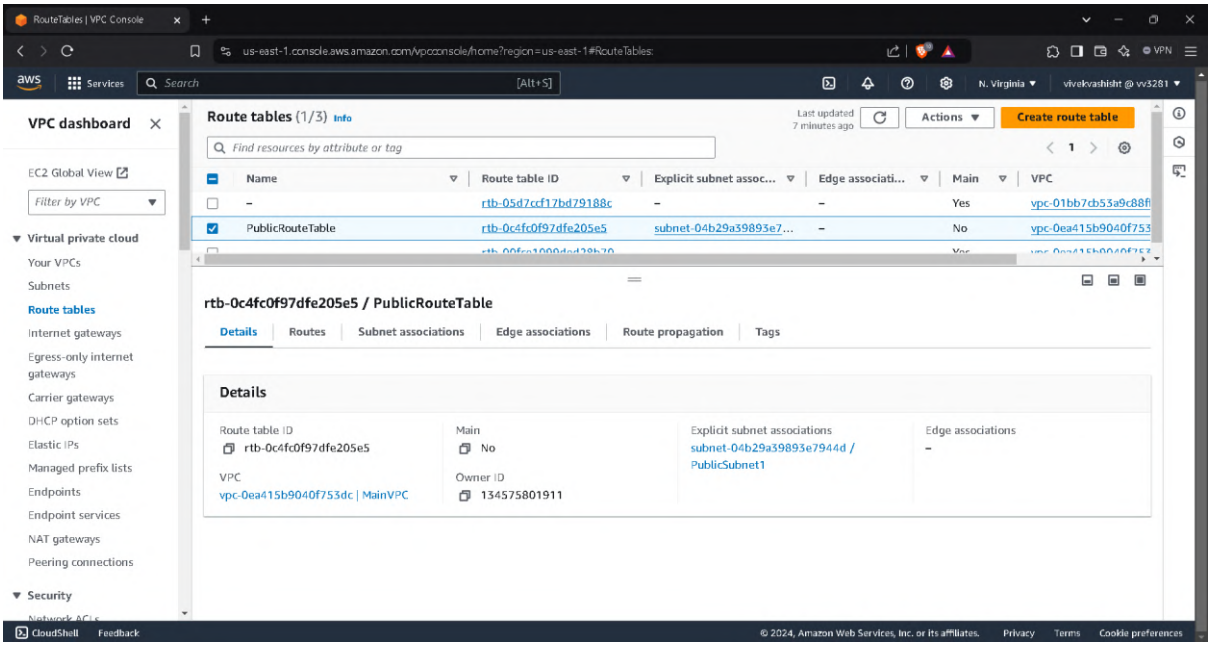
```
Command Prompt

      + instance_tenancy                   = "default"
      + ipv6_association_id                = (known after apply)
      + ipv6_cidr_block                    = (known after apply)
      + ipv6_cidr_block_network_border_group = (known after apply)
      + main_route_table_id                = (known after apply)
      + owner_id                           = (known after apply)
      + tags                               = {
          + "Name" = "MainVPC"
        }
      + tags_all                           = {
          + "Name" = "MainVPC"
        }
    }

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

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.main_vpc: Creating...
aws_vpc.main_vpc: Creation complete after 2s [id=vpc-0ea415b9040f753dc]
aws_internet_gateway.igw: Creating...
aws_subnet.public_subnet_1: Creating...
aws_subnet.private_subnet_1: Creating...
aws_route_table.public_rt: Creating...
aws_internet_gateway.igw: Creation complete after 1s [id=igw-057bd0790bbd28a4a]
aws_route_table.public_rt: Creation complete after 1s [id=rtb-0c4fc0f97dfe205e5]
aws_route.internet_access: Creating...
aws_subnet.private_subnet_1: Creation complete after 1s [id=subnet-0c184948abfe6c733]
aws_route.internet_access: Creation complete after 1s [id=r-rtb-0c4fc0f97dfe205e51080289494]
aws_subnet.public_subnet_1: Still creating... [10s elapsed]
aws_subnet.public_subnet_1: Creation complete after 19s [id=subnet-04b29a39893e7944d]
aws_route_table_association.public_subnet_assoc: Creating...
aws_route_table_association.public_subnet_assoc: Creation complete after 1s [id=rtbassoc-080529014332fac87]

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

C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>
```

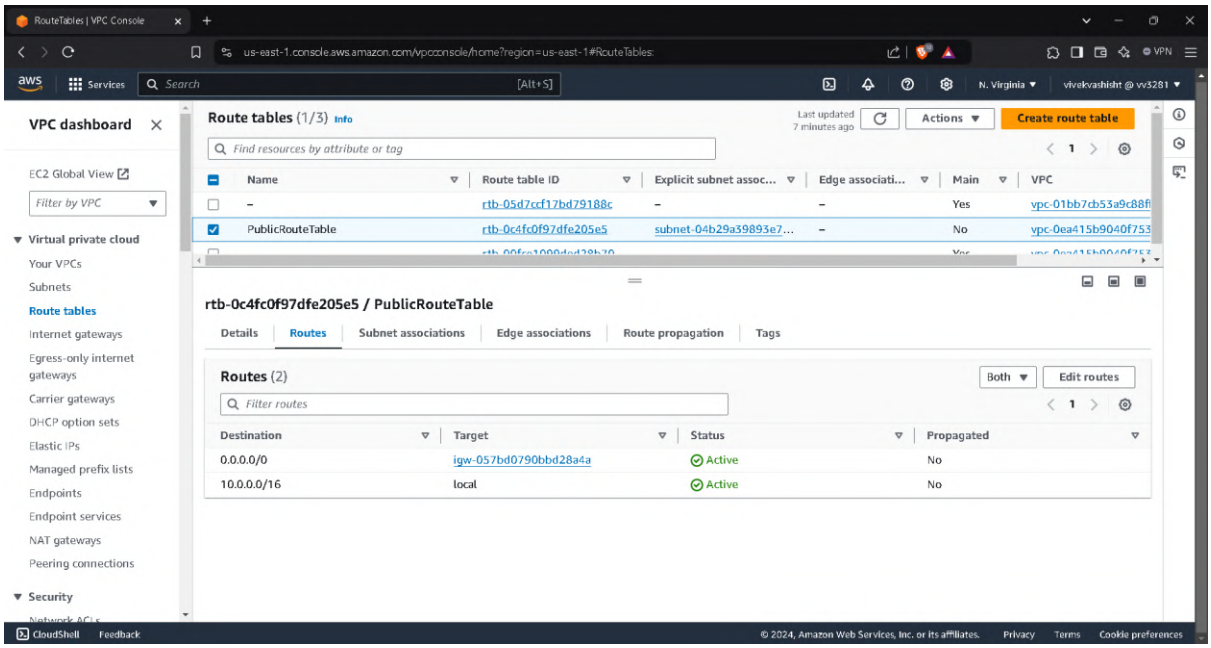

VPC dashboard
Your VPCs (1/2) Info
Name
VPC ID
State
IPv4 CIDR
IPv6 CIDR
MainVPC
vpc-0ea415b9040f753dc
Available
10.0.0.0/16
vpc-01bb7cb53a9c88ffd
172.31.0.0/16
Details
Resource map
CIDRs
Flow logs
Tags
Integrations
VPC ID
vpc-0ea415b9040f753dc
State
Available
DNS hostnames
Disabled
DNS resolution
Enabled
Tenancy
Default
DHCP option set
dopt-028d2411c992321a5
Main route table
rtb-00fce1099ded28b70
Main network ACL
acl-003959774e89e0c5c
Default VPC
No
IPv4 CIDR
10.0.0.0/16
IPv6 pool
IPv6 CIDR (Network border group)
Network Address Usage metrics
Disabled
Route 53 Resolver DNS Firewall rule groups
Owner ID
134575801911

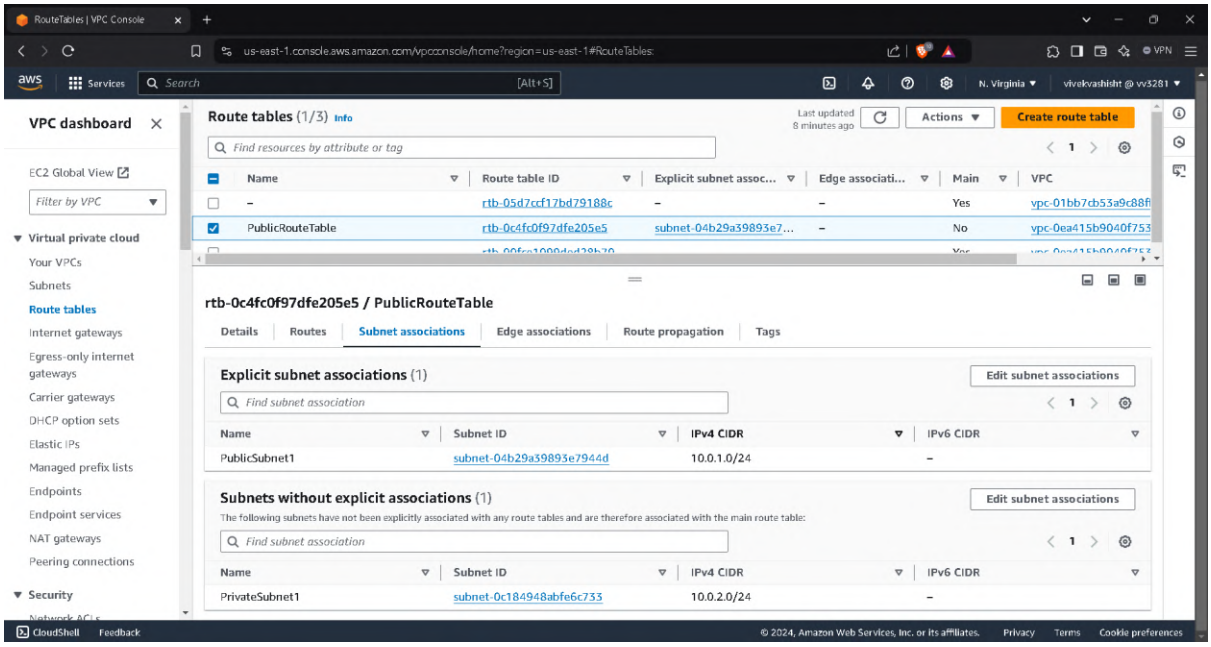

VPC dashboard
Subnets (1/8) Info
Name
Subnet ID
State
VPC
IPv4 CIDR
subnet-038e4d458c75deeb7
vpc-01bb7cb53a9c88ffd
172.31.16.0/20
subnet-087243af11c694cf9
172.31.32.0/20
PublicSubnet1
subnet-04b29a39893e7944d
vpc-0ea415b9040f753dc | Ma...
10.0.1.0/24
subnet-04b29a39893e7944d / PublicSubnet1
Details
Flow logs
Route table
Network ACL
CIDR reservations
Sharing
Tags
Subnet ID
Subnet ARN
arn:aws:ec2:us-east-1:134575801911:subnet/subnet-04b29a39893e7944d
State
Available
IPv4 CIDR
10.0.1.0/24
Available IPv4 addresses
251
IPv6 CIDR association ID
Availability Zone
us-east-1a
Availability Zone ID
use1-az1
IPv6 CIDR
VPC
vpc-0ea415b9040f753dc | MainVPC
Route table
rtb-0c4fc0f97dfe205e5 | PublicRouteTable
Network ACL
acl-003959774e89e0c5c
Network border group
us-east-1
Auto-assign public IPv4 address
Yes
Auto-assign IPv6 address
No


VPC dashboard
Subnets (1/8) Info
Name
Subnet ID
State
VPC
IPv4 CIDR
subnet-0941fb13760929842
vpc-01bb7cb53a9c88ffd
172.31.80.0/20
subnet-033e112bb58c68cdf
172.31.64.0/20
PrivateSubnet1
subnet-0c184948abfe6c733
vpc-0ea415b9040f753dc | Ma...
10.0.2.0/24
subnet-0c184948abfe6c733 / PrivateSubnet1
Details
Flow logs
Route table
Network ACL
CIDR reservations
Sharing
Tags
Subnet ID
Subnet ARN
arn:aws:ec2:us-east-1:134575801911:subnet/subnet-0c184948abfe6c733
State
Available
IPv4 CIDR
10.0.2.0/24
Available IPv4 addresses
251
IPv6 CIDR association ID
Availability Zone
us-east-1a
Availability Zone ID
use1-az1
IPv6 CIDR
VPC
vpc-0ea415b9040f753dc | MainVPC
Route table
rtb-00fce1099ded28b70
Network ACL
acl-003959774e89e0c5c
Network border group
us-east-1
Auto-assign public IPv4 address
No
Auto-assign IPv6 address
No
Default subnet
Outpost ID
IPv4 CIDR reservations


igws | VPC Console
us-east-1.console.aws.amazon.com/vpcconsole/home?region=us-east-1#igws:
Services
Search
[Alt+S]
N. Virginia
vivekvashisht @ vv3281
VPC dashboard
EC2 Global View
Filter by VPC
Virtual private cloud
Your VPCs
Subnets
Route tables
Internet gateways
Egress-only internet gateways
Carrier gateways
DHCP option sets
Elastic IPs
Managed prefix lists
Endpoints
Endpoint services
NAT gateways
Peering connections
Security
Internet gateways (1/2) Info
Actions
Create internet gateway
Name
Internet gateway ID
State
VPC ID
Owner
igw-037d25060f2cd3ce6
Attached
vpc-01bb7cb53a9c88ffd
134575801911
MainIGW
igw-057bd0790bbd28a4a
Attached
vpc-0ea415b9040f753dc | MainVPC
134575801911
igw-057bd0790bbd28a4a / MainIGW
Details
Tags
Internet gateway ID
State
VPC ID
Owner
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


RouteTables | VPC Console
us-east-1.console.aws.amazon.com/vpcconsole/home?region=us-east-1#RouteTables:
Services
Search
[Alt+S]
N. Virginia
vivekvashisht @ vv3281
VPC dashboard
Route tables (1/3) Info
Last updated 7 minutes ago
Actions
Create route table
Find resources by attribute or tag
Name
Route table ID
Explicit subnet assoc...
Edge associati...
Main
VPC
rtb-05d7ccf17bd79188c
Yes
PublicRouteTable
rtb-0c4fc0f97dfe205e5
subnet-04b29a39893e7...
No
rtb-0c4fc0f97dfe205e5 / PublicRouteTable
Details
Routes
Subnet associations
Edge associations
Route propagation
Tags
Route table ID
Main
Explicit subnet associations
Edge associations
VPC
Owner ID
subnet-04b29a39893e7944d / PublicSubnet1
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


RouteTables | VPC Console
us-east-1.console.aws.amazon.com/vpcconsole/home?region=us-east-1#RouteTables:
Services
Search
[Alt+S]
N. Virginia
vivekvashisht @ vv3281
VPC dashboard
Route tables (1/3) Info
Last updated 7 minutes ago
Actions
Create route table
Find resources by attribute or tag
Name
Route table ID
Explicit subnet assoc...
Edge associati...
Main
VPC
rtb-05d7ccf17bd79188c
Yes
PublicRouteTable
rtb-0c4fc0f97dfe205e5
subnet-04b29a39893e7...
No
rtb-0c4fc0f97dfe205e5 / PublicRouteTable
Details
Routes
Subnet associations
Edge associations
Route propagation
Tags
Routes (2)
Both
Edit routes
Filter routes
Destination
Target
Status
Propagated
0.0.0.0/0
igw-057bd0790bbd28a4a
Active
No
10.0.0.0/16
local
Active
No
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences



EC2 Instances and Auto Scaling

a) Overview

In this phase of the project, I focused on setting up EC2 instances within an Auto Scaling Group (ASG) to ensure scalability and high availability for the application. The setup includes creating a security group, a launch template with user data to install and configure NGINX and configuring scaling policies. All configurations were defined in a new Terraform file named `ec2_autoscaling.tf` in the Terraform project directory. I referenced both the Terraform Registry and AWS Documentation to ensure accuracy.

b) Security Group Configuration

To secure access to the instances, I created a security group allowing specific ingress and egress rules. This security group restricts SSH access to my IP address, allows HTTP traffic from anywhere, and permits all outbound traffic. Here's the code block for the security group:

```
# Security Group
resource "aws_security_group" "app_sg" {
  name        = "app-sg"
  description = "Allow HTTP and SSH traffic"
  vpc_id      = aws_vpc.main_vpc.id

  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
```

```
    cidr_blocks = ["142.59.28.49/32"]  # Restricted SSH to my IP only
  }

  ingress {
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]  # Allows HTTP traffic from anywhere
  }

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]  # Allows all outbound traffic
  }

  tags = {
    Name = "AppSecurityGroup"
  }
}
```

c) Launch Template with NGINX Installation

Next, I created a launch template that defines the configuration for instances launched by the ASG. The user data script installs and starts NGINX, ensuring that each instance is ready to serve web traffic immediately upon launch.

```
# Launch Template for EC2 Instances
resource "aws_launch_template" "app" {
  name_prefix   = "app-launch-template"
  image_id      = "ami-06b213ccaff8cd686"  # Amazon Linux 2 AMI
  instance_type = "t2.micro"               # Cost-effective instance type

  network_interfaces {
    security_groups = [aws_security_group.app_sg.id]
  }

  user_data = base64encode(<<-EOF
    #!/bin/bash
    sudo yum update -y
    sudo amazon-linux-extras install nginx1 -y
    sudo systemctl start nginx
    sudo systemctl enable nginx
  EOF
  )

  lifecycle {
    create_before_destroy = true
  }

  tag_specifications {
    resource_type = "instance"
    tags = {
      Name = "AppInstance"
    }
  }
}
```

d) Auto Scaling Group (ASG) Setup

The ASG is configured to maintain a minimum of 1 instance, a desired capacity of 2 instances, and a maximum of 5 instances. The ASG utilizes the launch template defined above and deploys instances in the specified subnet.

```
# Auto Scaling Group (ASG) for us-east-1
resource "aws_autoscaling_group" "app_asg" {
  desired_capacity     = 2
  min_size             = 1
  max_size             = 5
  vpc_zone_identifier  = [aws_subnet.public_subnet_1.id]

  launch_template {
    id      = aws_launch_template.app.id
    version = "$Latest"
  }

  tag {
    key                 = "Name"
    value               = "AppASGInstance"
    propagate_at_launch = true
  }

  health_check_type = "EC2"

  lifecycle {
    create_before_destroy = true
  }
}
```

e) Auto Scaling Policies

I configured scaling policies to dynamically adjust the number of instances based on demand. The scale-up policy increases the capacity by 1 instance, while the scale-down policy decreases it by 1 instance.

```
# Auto Scaling Policies

# Scaling Up Policy
resource "aws_autoscaling_policy" "scale_up" {
  name                   = "scale-up"
  scaling_adjustment     = 1  # Adds 1 instance when triggered
  adjustment_type        = "ChangeInCapacity"
  autoscaling_group_name = aws_autoscaling_group.app_asg.name
}

# Scaling Down Policy
resource "aws_autoscaling_policy" "scale_down" {
  name                   = "scale-down"
  scaling_adjustment     = -1  # Removes 1 instance when triggered
  adjustment_type        = "ChangeInCapacity"
  autoscaling_group_name = aws_autoscaling_group.app_asg.name
}
```

f) Deployment and Troubleshooting

After saving the `ec2_autoscaling.tf` file, I ran the following commands to initialize, plan, and apply the configuration:

```
terraform init
terraform plan
terraform apply -auto-approve
```

Upon verification, I encountered an issue while connecting to the EC2 instances. The EC2 Instance Connect IP range (`18.206.107.24/29`) was not authorized for SSH access. To fix this, I updated the security group in `ec2_autoscaling.tf`:

```
ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["18.206.107.24/29"]  # EC2 Instance Connect IP range for
us-east-1
}
```

After adding this rule, I re-ran `terraform plan` and `terraform apply -auto-approve` to update the security group.

g) Testing NGINX Installation

Once connected, I encountered another issue where NGINX failed to run due to an incorrect package command. I updated the user data block in the launch template:

```
user_data = base64encode(<<-EOF
  #!/bin/bash
  sudo dnf update -y
  sudo dnf install nginx -y
  sudo systemctl start nginx
  sudo systemctl enable nginx
EOF
)
```

To apply this change, I destroyed the existing launch template:

```
terraform destroy -target=aws_launch_template.app
```

Then, I re-ran `terraform apply -auto-approve` to create new instances with the updated NGINX configuration.

h) Verification

After successful deployment, I verified the NGINX installation by connecting to the EC2 instances and running:

```
sudo systemctl status nginx
```

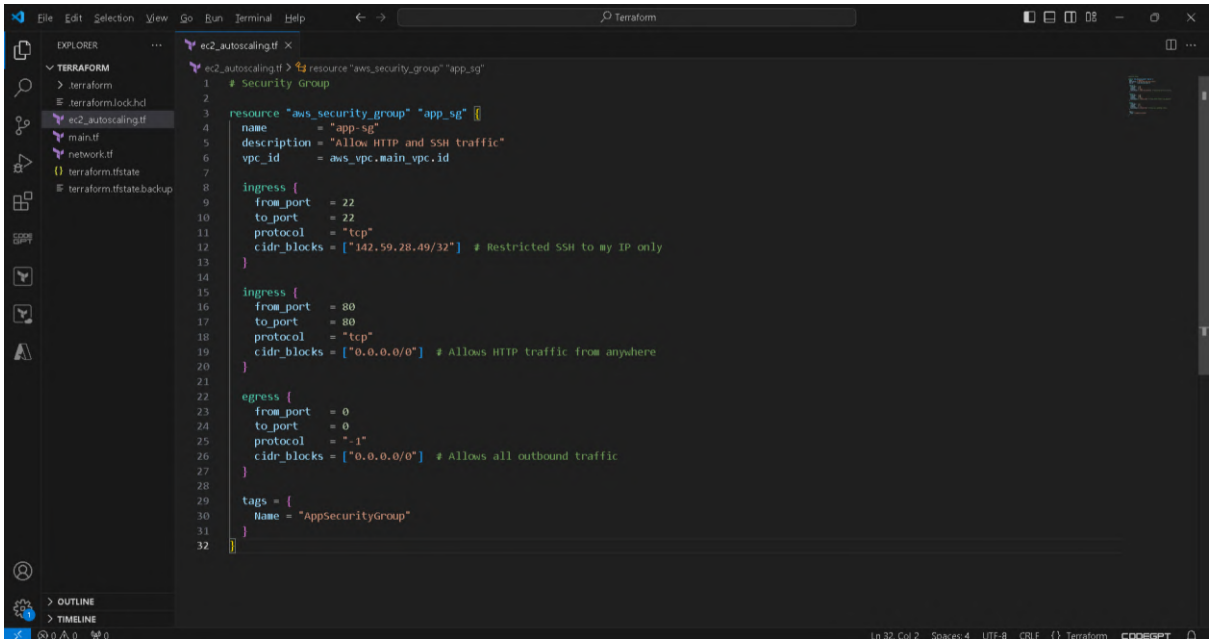
The status showed `active (running)`, confirming that NGINX was successfully installed and running. Additionally, I accessed the NGINX web page by entering the instance's public IP in my browser.

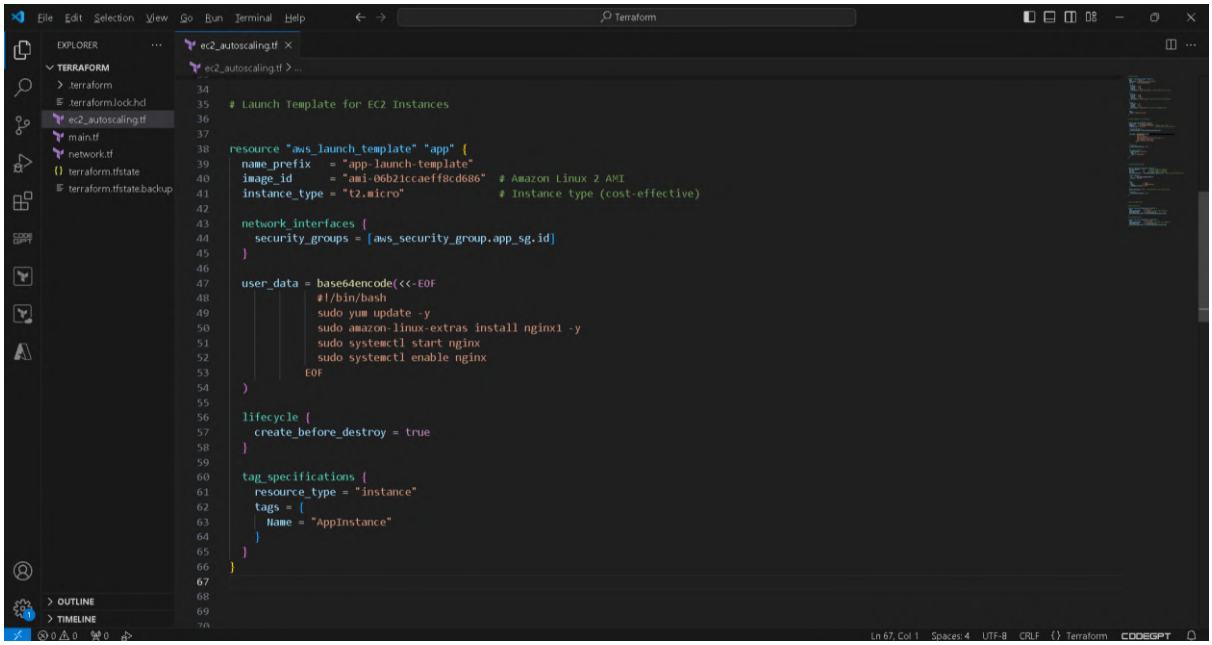
ASG Verification

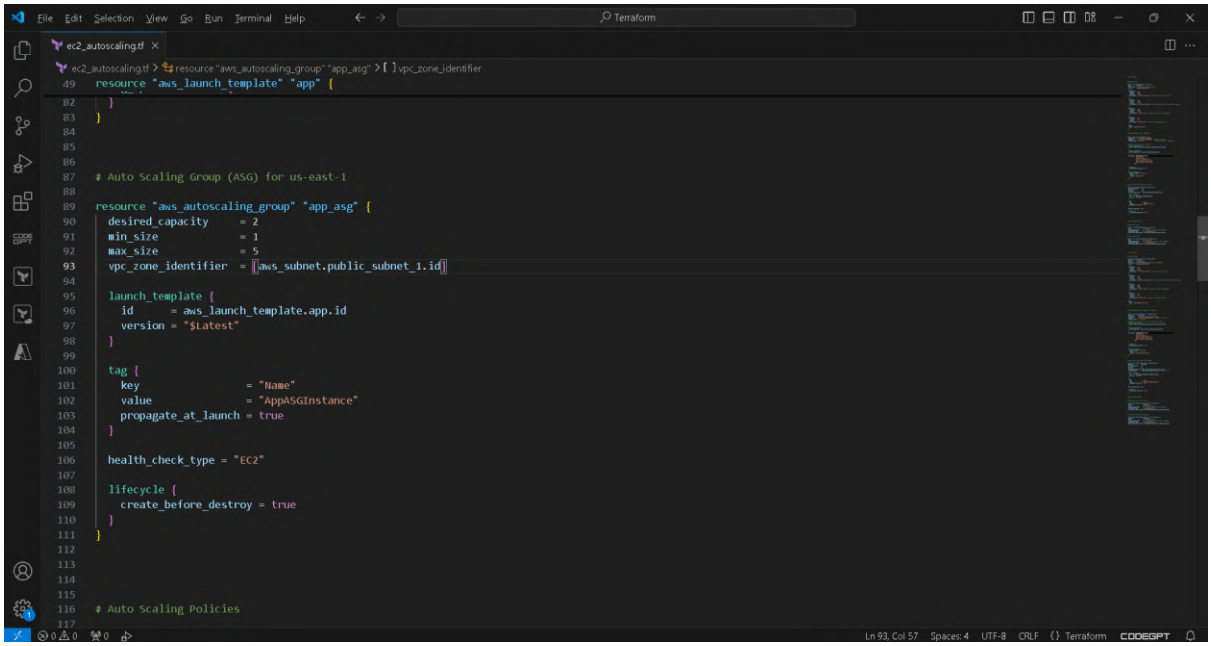
Finally, I checked the Auto Scaling Group in the AWS Console to ensure it was configured correctly with the expected settings.

This phase of the project was successfully completed, with the EC2 instances running NGINX and the ASG configured to scale resources as needed.

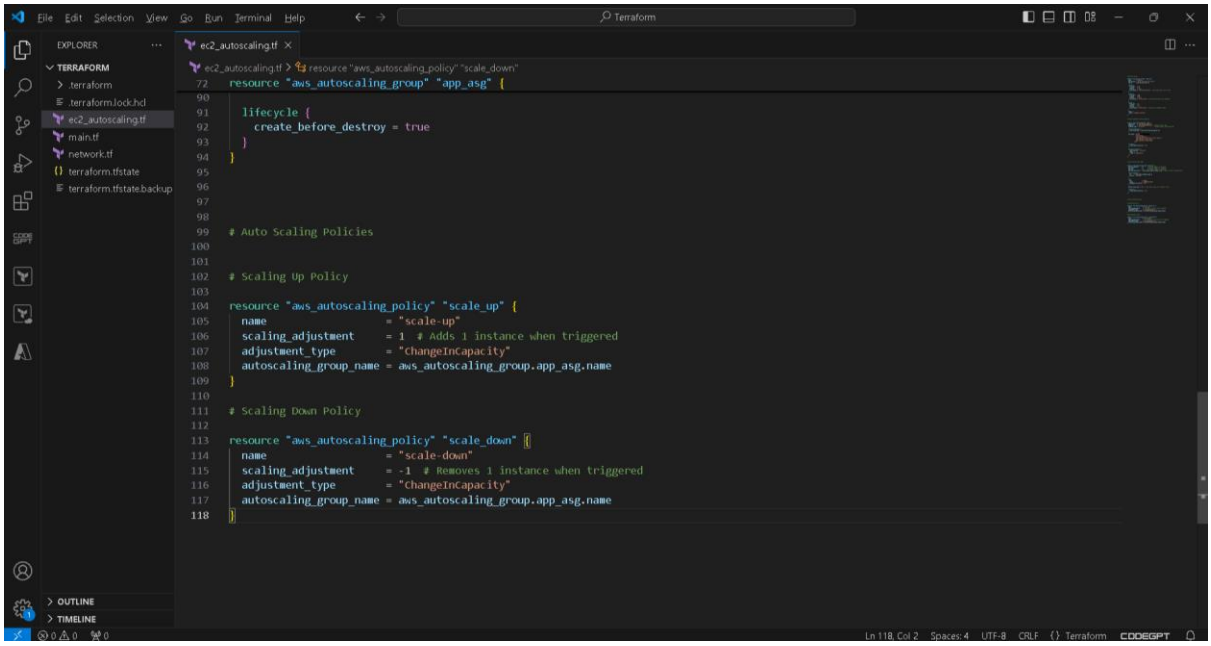
Screenshots



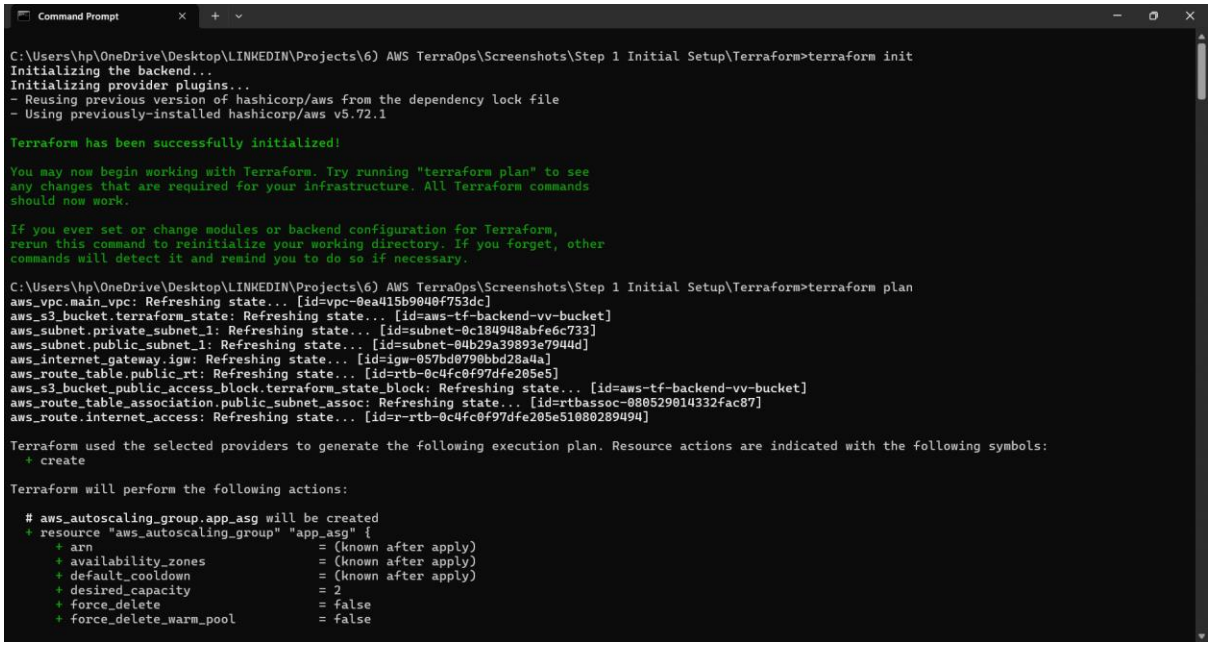




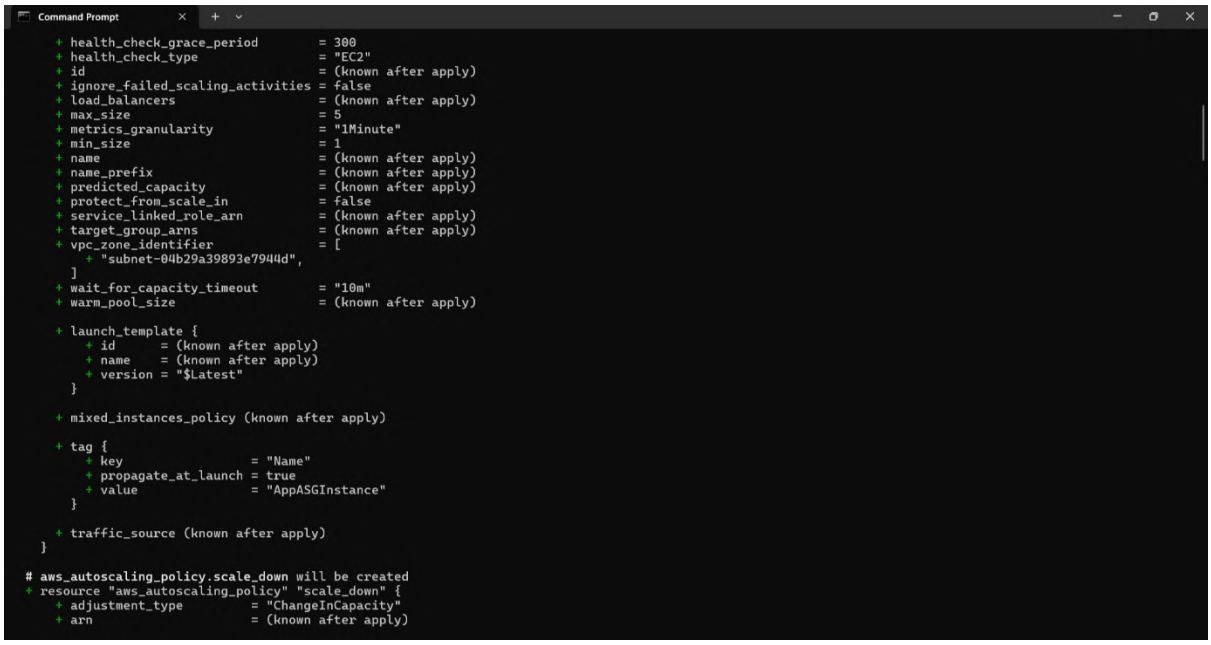
```
resource "aws_launch_template" "app" {
  }
}


# Auto Scaling Group (ASG) for us-east-1

resource "aws_autoscaling_group" "app_asg" {
  desired_capacity    = 2
  min_size            = 1
  max_size            = 5
  vpc_zone_identifier = [aws_subnet.public_subnet_1.id]

  launch_template {
    id      = aws_launch_template.app.id
    version = "$Latest"
  }

  tag {
    key                 = "Name"
    value               = "AppASGInstance"
    propagate_at_launch = true
  }

  health_check_type = "EC2"

  lifecycle {
    create_before_destroy = true
  }
}



# Auto Scaling Policies
```



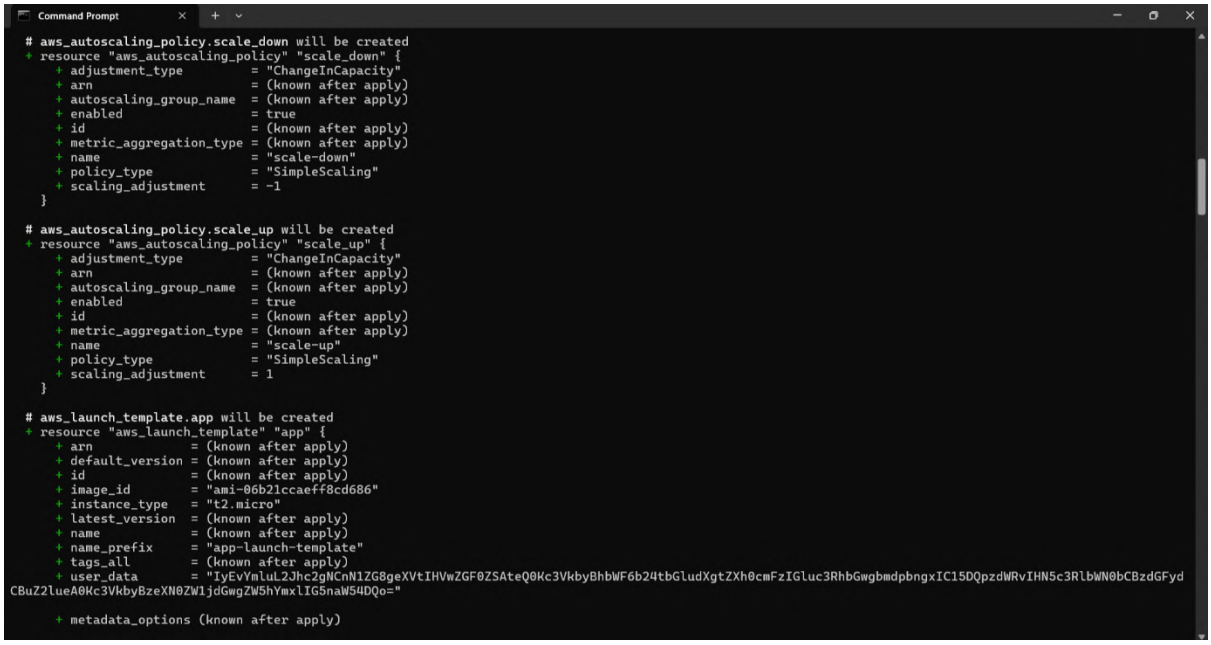
```
resource "aws_autoscaling_group" "app_asg" {

  lifecycle {
    create_before_destroy = true
  }
}



# Auto Scaling Policies


# Scaling Up Policy

resource "aws_autoscaling_policy" "scale_up" {
  name                   = "scale-up"
  scaling_adjustment     = 1  # Adds 1 instance when triggered
  adjustment_type        = "ChangeInCapacity"
  autoscaling_group_name = aws_autoscaling_group.app_asg.name
}

# Scaling Down Policy

resource "aws_autoscaling_policy" "scale_down" {
  name                   = "scale-down"
  scaling_adjustment     = -1  # Removes 1 instance when triggered
  adjustment_type        = "ChangeInCapacity"
  autoscaling_group_name = aws_autoscaling_group.app_asg.name
}
```



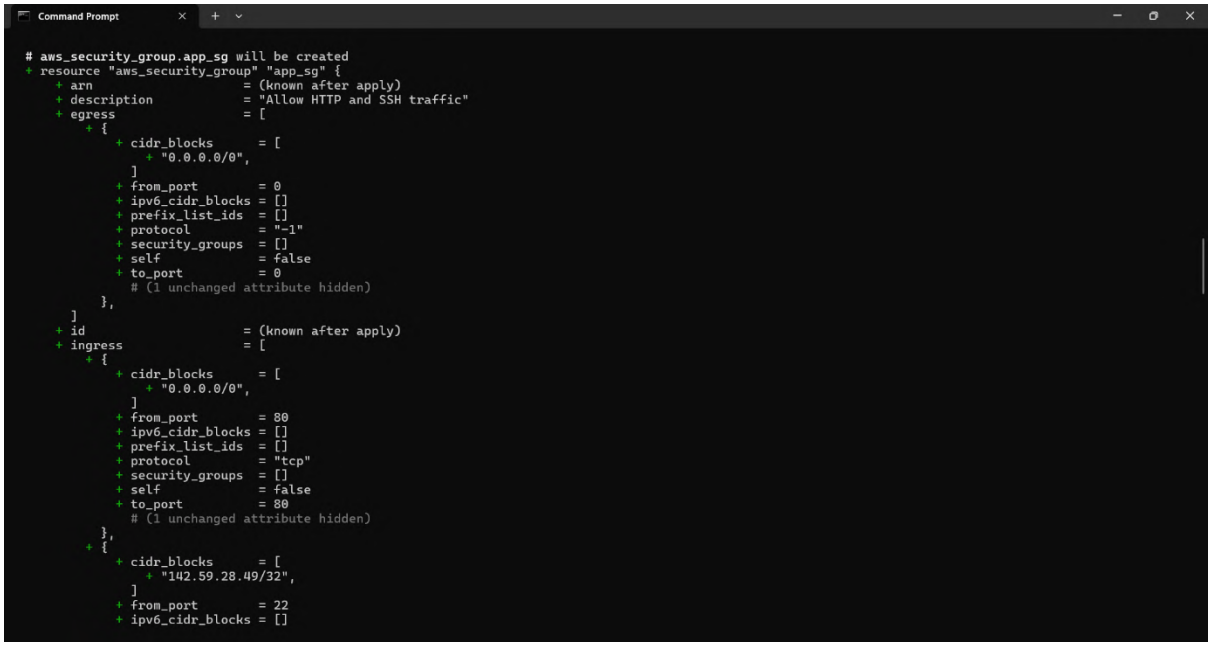
```
C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.72.1

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.

C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>terraform plan
aws_vpc.main_vpc: Refreshing state... [id=vpc-0ea415b9040f753dc]
aws_s3_bucket.terraform_state: Refreshing state... [id=aws-tf-backend-vv-bucket]
aws_subnet.private_subnet_1: Refreshing state... [id=subnet-0c184948abfe6c733]
aws_subnet.public_subnet_1: Refreshing state... [id=subnet-04b29a39893e7944d]
aws_internet_gateway.igw: Refreshing state... [id=igw-057bd0790bbd28a4a]
aws_route_table.public_rt: Refreshing state... [id=rtb-0c4fc0f97dfe205e5]
aws_s3_bucket_public_access_block.terraform_state_block: Refreshing state... [id=aws-tf-backend-vv-bucket]
aws_route_table_association.public_subnet_assoc: Refreshing state... [id=rtbassoc-080529014332fac87]
aws_route.internet_access: Refreshing state... [id=r-rtb-0c4fc0f97dfe205e51080289494]

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_autoscaling_group.app_asg will be created
  + resource "aws_autoscaling_group" "app_asg" {
      + arn                       = (known after apply)
      + availability_zones        = (known after apply)
      + default_cooldown          = (known after apply)
      + desired_capacity          = 2
      + force_delete              = false
      + force_delete_warm_pool    = false
```



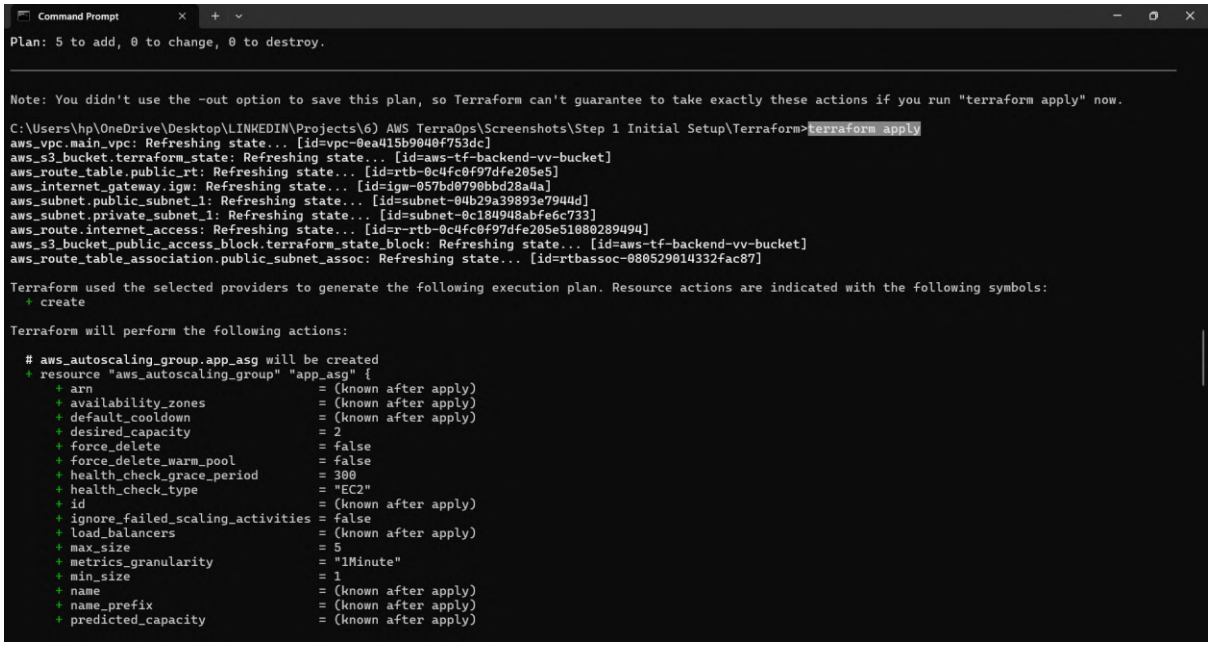
```
Command Prompt
    + health_check_grace_period        = 300
    + health_check_type                = "EC2"
    + id                               = (known after apply)
    + ignore_failed_scaling_activities = false
    + load_balancers                   = (known after apply)
    + max_size                         = 5
    + metrics_granularity              = "1Minute"
    + min_size                         = 1
    + name                             = (known after apply)
    + name_prefix                      = (known after apply)
    + predicted_capacity               = (known after apply)
    + protect_from_scale_in            = false
    + service_linked_role_arn          = (known after apply)
    + target_group_arns                = (known after apply)
    + vpc_zone_identifier              = [
        + "subnet-04b29a39893e7944d",
      ]
    + wait_for_capacity_timeout        = "10m"
    + warm_pool_size                   = (known after apply)

    + launch_template {
        + id      = (known after apply)
        + name    = (known after apply)
        + version = "$Latest"
      }

    + mixed_instances_policy (known after apply)

    + tag {
        + key                 = "Name"
        + propagate_at_launch = true
        + value               = "AppASGInstance"
      }

    + traffic_source (known after apply)
  }

  # aws_autoscaling_policy.scale_down will be created
  + resource "aws_autoscaling_policy" "scale_down" {
      + adjustment_type         = "ChangeInCapacity"
      + arn                     = (known after apply)
```

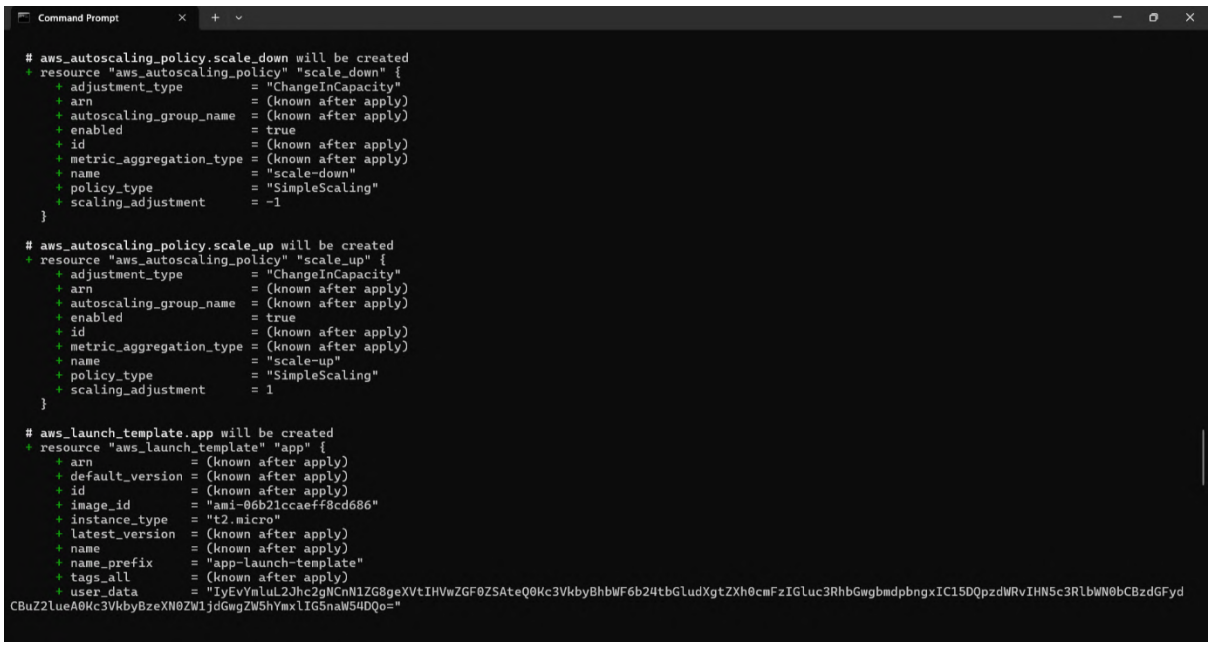


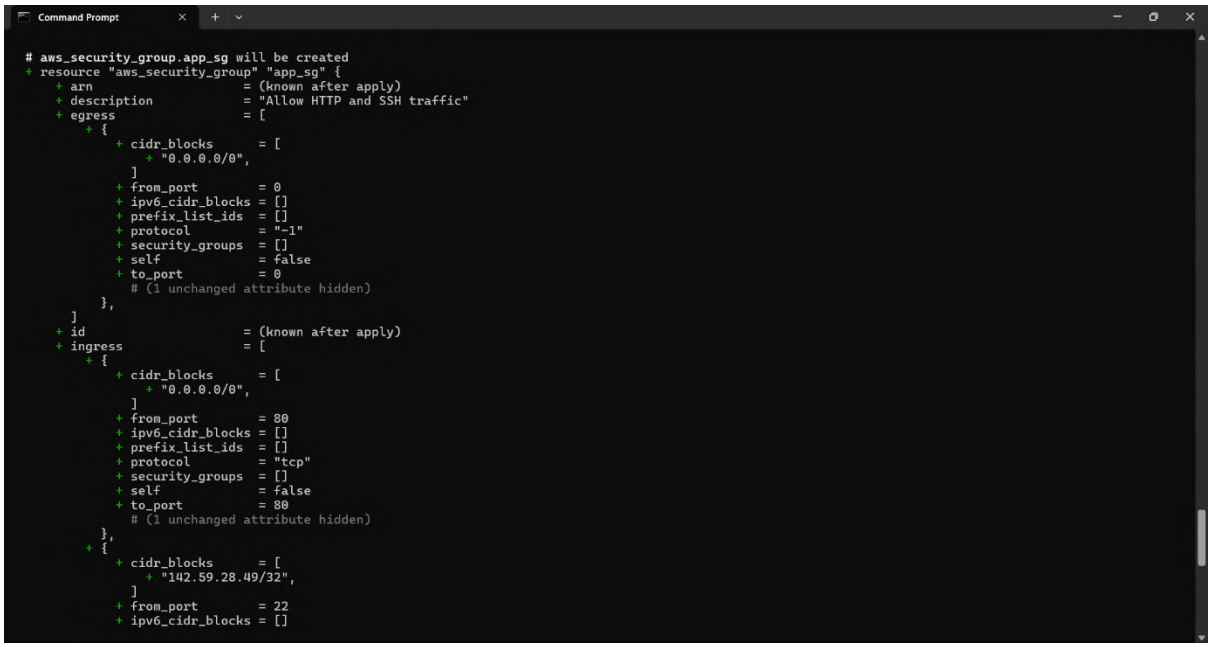
```
Command Prompt
  # aws_autoscaling_policy.scale_down will be created
  + resource "aws_autoscaling_policy" "scale_down" {
      + adjustment_type         = "ChangeInCapacity"
      + arn                     = (known after apply)
      + autoscaling_group_name  = (known after apply)
      + enabled                 = true
      + id                      = (known after apply)
      + metric_aggregation_type = (known after apply)
      + name                    = "scale-down"
      + policy_type             = "SimpleScaling"
      + scaling_adjustment      = -1
    }

  # aws_autoscaling_policy.scale_up will be created
  + resource "aws_autoscaling_policy" "scale_up" {
      + adjustment_type         = "ChangeInCapacity"
      + arn                     = (known after apply)
      + autoscaling_group_name  = (known after apply)
      + enabled                 = true
      + id                      = (known after apply)
      + metric_aggregation_type = (known after apply)
      + name                    = "scale-up"
      + policy_type             = "SimpleScaling"
      + scaling_adjustment      = 1
    }

  # aws_launch_template.app will be created
  + resource "aws_launch_template" "app" {
      + arn             = (known after apply)
      + default_version = (known after apply)
      + id              = (known after apply)
      + image_id        = "ami-06b21ccaeff8cd686"
      + instance_type   = "t2.micro"
      + latest_version  = (known after apply)
      + name            = (known after apply)
      + name_prefix     = "app-launch-template"
      + tags_all        = (known after apply)
      + user_data       = "IyEvYmluL2Jhc2gNCnN1ZG8geXVtIHVwZGF0ZSAteQ0Kc3VkbyBhbWF6b24tbGludXgtZXh0cmFzIGluc3RhbGwgbmdpbngxIC15DQpzdWRvIHN5c3RlbWN0bCBzdGFyd
CBuZ2lueA0Kc3VkbyBzeXN0ZW1jdGwgZW5hYmxlIG5naW54DQo="

      + metadata_options (known after apply)
```

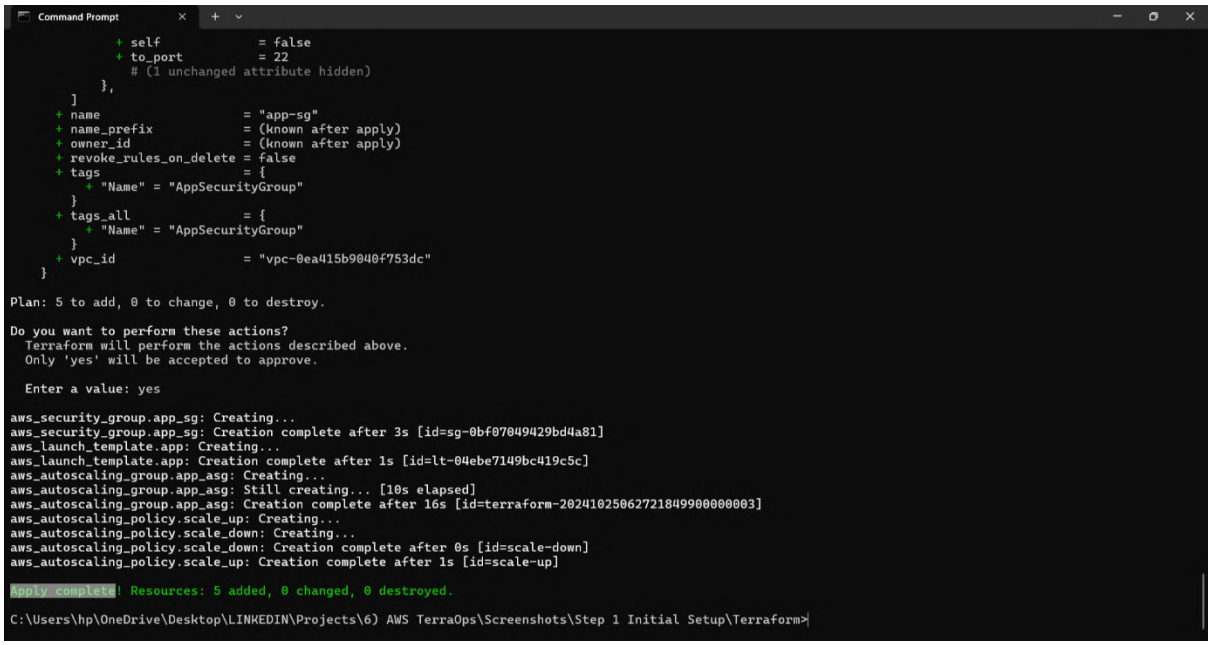


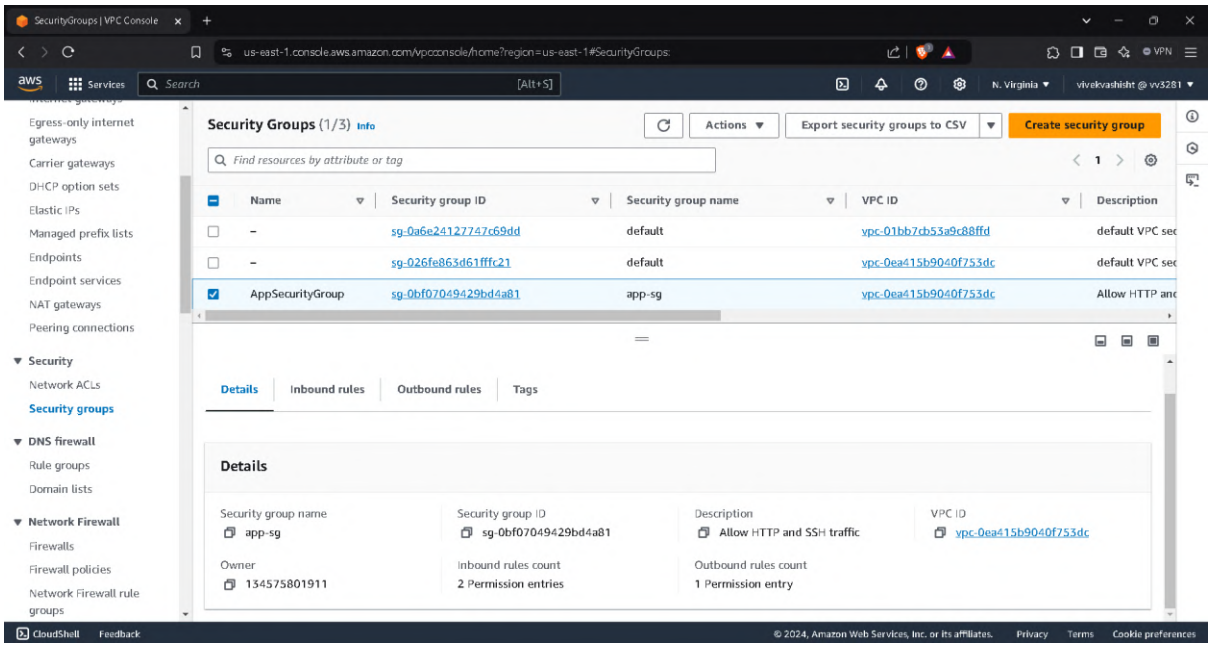
```
Command Prompt
  # aws_security_group.app_sg will be created
  + resource "aws_security_group" "app_sg" {
      + arn                    = (known after apply)
      + description            = "Allow HTTP and SSH traffic"
      + egress                 = [
          + {
              + cidr_blocks      = [
                  + "0.0.0.0/0",
                ]
              + from_port        = 0
              + ipv6_cidr_blocks = []
              + prefix_list_ids  = []
              + protocol         = "-1"
              + security_groups  = []
              + self             = false
              + to_port          = 0
                # (1 unchanged attribute hidden)
            },
        ]
      + id                     = (known after apply)
      + ingress                = [
          + {
              + cidr_blocks      = [
                  + "0.0.0.0/0",
                ]
              + from_port        = 80
              + ipv6_cidr_blocks = []
              + prefix_list_ids  = []
              + protocol         = "tcp"
              + security_groups  = []
              + self             = false
              + to_port          = 80
                # (1 unchanged attribute hidden)
            },
          + {
              + cidr_blocks      = [
                  + "142.59.28.49/32",
                ]
              + from_port        = 22
              + ipv6_cidr_blocks = []
```

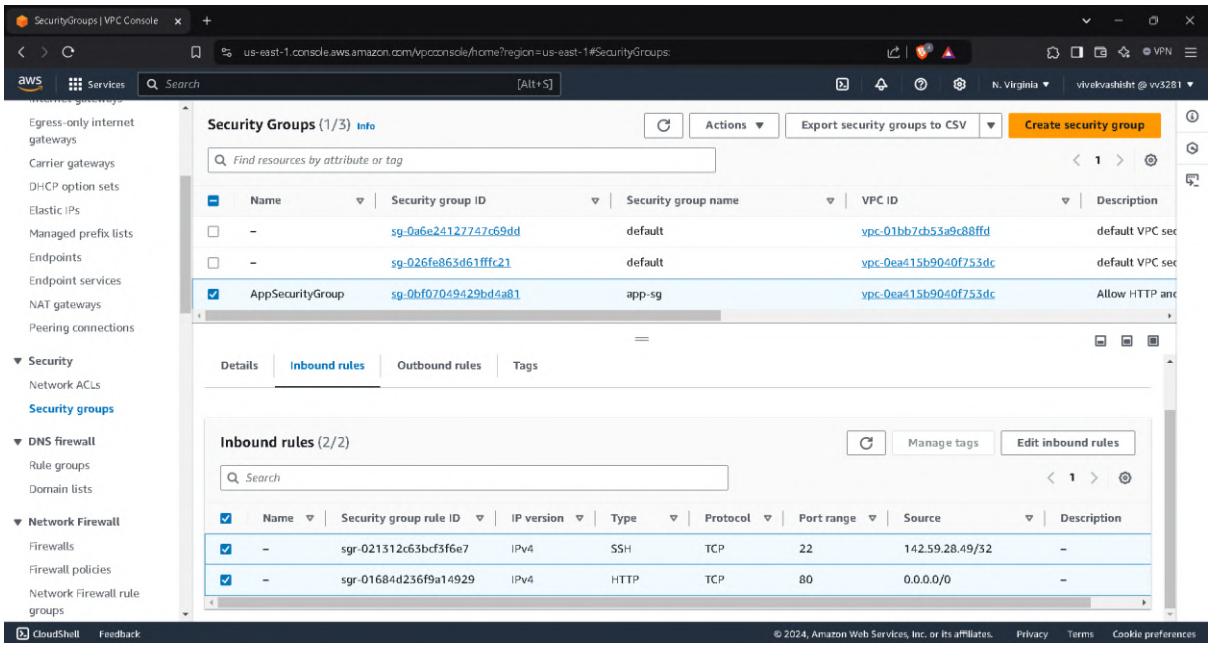

Command Prompt
Plan: 5 to add, 0 to change, 0 to destroy.
Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.
C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>terraform apply
aws_vpc.main_vpc: Refreshing state... [id=vpc-0ea415b9040f753dc]
aws_s3_bucket.terraform_state: Refreshing state... [id=aws-tf-backend-vv-bucket]
aws_route_table.public_rt: Refreshing state... [id=rtb-0c4fc0f97dfe205e5]
aws_internet_gateway.igw: Refreshing state... [id=igw-057bd0790bbd28a4a]
aws_subnet.public_subnet_1: Refreshing state... [id=subnet-04b29a39893e7944d]
aws_subnet.private_subnet_1: Refreshing state... [id=subnet-0c184948abfe6c733]
aws_route.internet_access: Refreshing state... [id=r-rtb-0c4fc0f97dfe205e51080289494]
aws_s3_bucket_public_access_block.terraform_state_block: Refreshing state... [id=aws-tf-backend-vv-bucket]
aws_route_table_association.public_subnet_assoc: Refreshing state... [id=rtbassoc-080529014332fac87]
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_autoscaling_group.app_asg will be created
+ resource "aws_autoscaling_group" "app_asg" {
+ arn = (known after apply)
+ availability_zones = (known after apply)
+ default_cooldown = (known after apply)
+ desired_capacity = 2
+ force_delete = false
+ force_delete_warm_pool = false
+ health_check_grace_period = 300
+ health_check_type = "EC2"
+ id = (known after apply)
+ ignore_failed_scaling_activities = false
+ load_balancers = (known after apply)
+ max_size = 5
+ metrics_granularity = "1Minute"
+ min_size = 1
+ name = (known after apply)
+ name_prefix = (known after apply)
+ predicted_capacity = (known after apply)

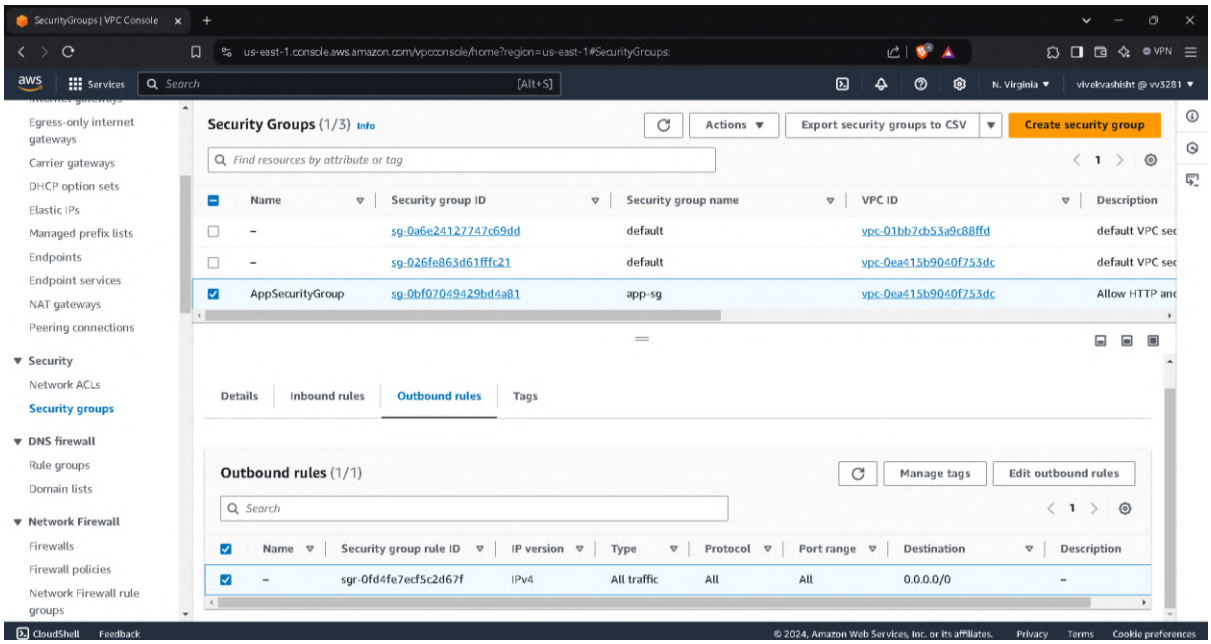

Command Prompt
# aws_autoscaling_policy.scale_down will be created
+ resource "aws_autoscaling_policy" "scale_down" {
+ adjustment_type = "ChangeInCapacity"
+ arn = (known after apply)
+ autoscaling_group_name = (known after apply)
+ enabled = true
+ id = (known after apply)
+ metric_aggregation_type = (known after apply)
+ name = "scale-down"
+ policy_type = "SimpleScaling"
+ scaling_adjustment = -1
}
# aws_autoscaling_policy.scale_up will be created
+ resource "aws_autoscaling_policy" "scale_up" {
+ adjustment_type = "ChangeInCapacity"
+ arn = (known after apply)
+ autoscaling_group_name = (known after apply)
+ enabled = true
+ id = (known after apply)
+ metric_aggregation_type = (known after apply)
+ name = "scale-up"
+ policy_type = "SimpleScaling"
+ scaling_adjustment = 1
}
# aws_launch_template.app will be created
+ resource "aws_launch_template" "app" {
+ arn = (known after apply)
+ default_version = (known after apply)
+ id = (known after apply)
+ image_id = "ami-06b21ccaeff8cd686"
+ instance_type = "t2.micro"
+ latest_version = (known after apply)
+ name = (known after apply)
+ name_prefix = "app-launch-template"
+ tags_all = (known after apply)
+ user_data = "IyEvYmluL2Jhc2gNCnN1ZG8geXVtIHVwZGF0ZSAteQ0Kc3VkbyBhbWF6b24tbGludXgtZXh0cmFzIGluc3RhbGwgbmdpbngxIC15DQpzdWRvIHN5c3RlbWN0bCBzdGFyd
CBuZ2lueA0Kc3VkbyBzeXN0ZW1jdGwgZW5hYmxlIG5naW54DQo="

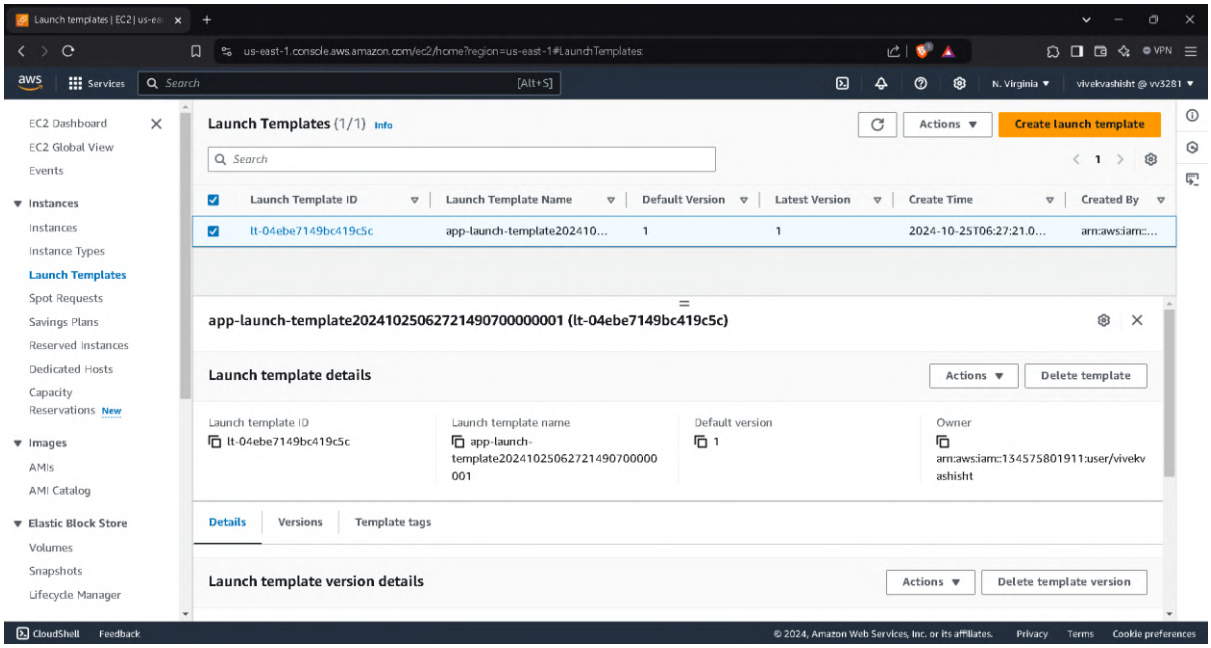

Command Prompt
# aws_security_group.app_sg will be created
+ resource "aws_security_group" "app_sg" {
+ arn = (known after apply)
+ description = "Allow HTTP and SSH traffic"
+ egress = [
+ {
+ cidr_blocks = [
+ "0.0.0.0/0",
]
+ from_port = 0
+ ipv6_cidr_blocks = []
+ prefix_list_ids = []
+ protocol = "-1"
+ security_groups = []
+ self = false
+ to_port = 0
# (1 unchanged attribute hidden)
},
]
+ id = (known after apply)
+ ingress = [
+ {
+ cidr_blocks = [
+ "0.0.0.0/0",
]
+ from_port = 80
+ ipv6_cidr_blocks = []
+ prefix_list_ids = []
+ protocol = "tcp"
+ security_groups = []
+ self = false
+ to_port = 80
# (1 unchanged attribute hidden)
},
+ {
+ cidr_blocks = [
+ "142.59.28.49/32",
]
+ from_port = 22
+ ipv6_cidr_blocks = []

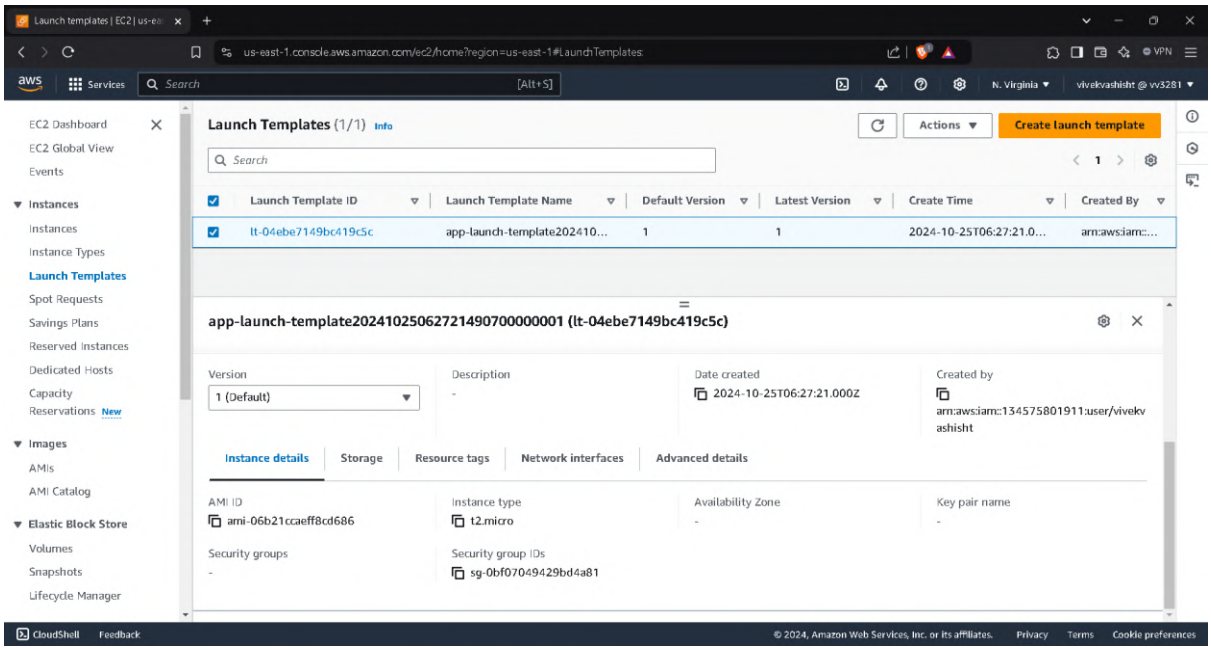

Command Prompt
+ self = false
+ to_port = 22
# (1 unchanged attribute hidden)
},
]
+ name = "app-sg"
+ name_prefix = (known after apply)
+ owner_id = (known after apply)
+ revoke_rules_on_delete = false
+ tags = {
+ "Name" = "AppSecurityGroup"
}
+ tags_all = {
+ "Name" = "AppSecurityGroup"
}
+ vpc_id = "vpc-0ea415b9040f753dc"
}
Plan: 5 to add, 0 to change, 0 to destroy.
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_security_group.app_sg: Creating...
aws_security_group.app_sg: Creation complete after 3s [id=sg-0bf07049429bd4a81]
aws_launch_template.app: Creating...
aws_launch_template.app: Creation complete after 1s [id=lt-04ebe7149bc419c5c]
aws_autoscaling_group.app_asg: Creating...
aws_autoscaling_group.app_asg: Still creating... [10s elapsed]
aws_autoscaling_group.app_asg: Creation complete after 16s [id=terraform-20241025062721849900000003]
aws_autoscaling_policy.scale_up: Creating...
aws_autoscaling_policy.scale_down: Creating...
aws_autoscaling_policy.scale_down: Creation complete after 0s [id=scale-down]
aws_autoscaling_policy.scale_up: Creation complete after 1s [id=scale-up]
Apply complete! Resources: 5 added, 0 changed, 0 destroyed.
C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>

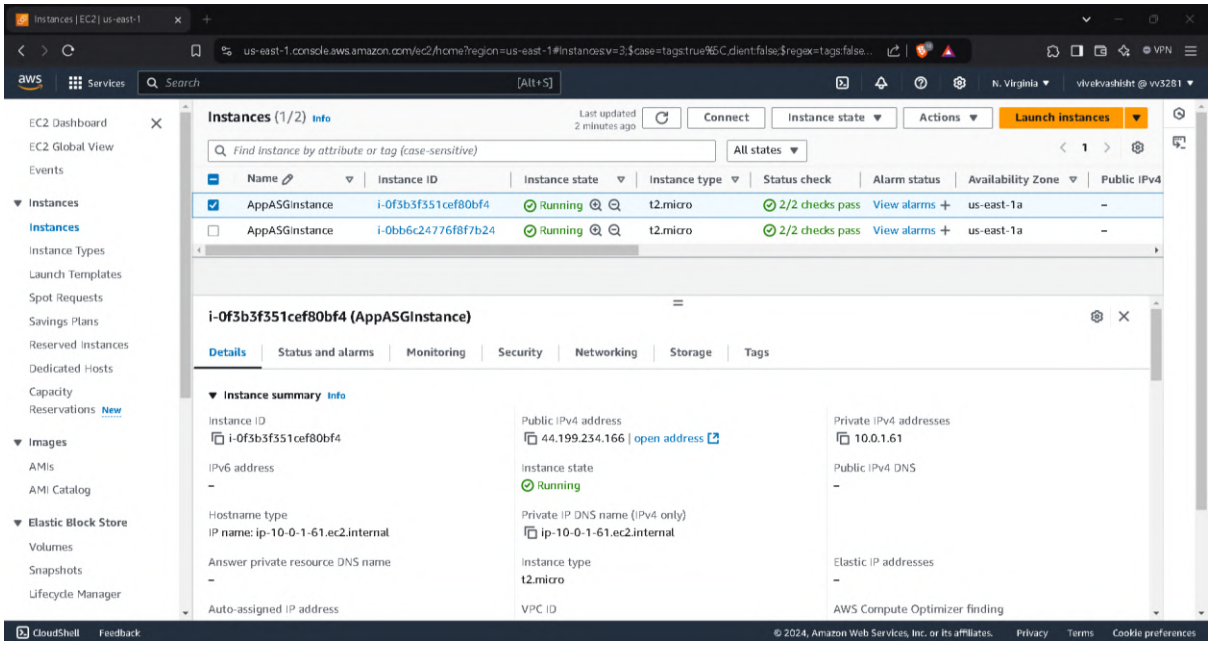

SecurityGroups | VPC Console
us-east-1.console.aws.amazon.com/vpcconsole/home?region=us-east-1#SecurityGroups:
Services
Search
[Alt+S]
N. Virginia
vivekvashisht @ vv3281
Egress-only internet gateways
Carrier gateways
DHCP option sets
Elastic IPs
Managed prefix lists
Endpoints
Endpoint services
NAT gateways
Peering connections
Security
Network ACLs
Security groups
DNS firewall
Rule groups
Domain lists
Network Firewall
Firewalls
Firewall policies
Network Firewall rule groups
Security Groups (1/3) Info
Actions
Export security groups to CSV
Create security group
Find resources by attribute or tag
Name
Security group ID
Security group name
VPC ID
Description
sg-0a6e24127747c69dd
default
vpc-01bb7cb53a9c88ffd
sg-026fe863d61fffc21
vpc-0ea415b9040f753dc
AppSecurityGroup
sg-0bf07049429bd4a81
app-sg
Allow HTTP and
Details
Inbound rules
Outbound rules
Tags
Security group name
app-sg
Security group ID
sg-0bf07049429bd4a81
Description
Allow HTTP and SSH traffic
VPC ID
Owner
134575801911
Inbound rules count
2 Permission entries
Outbound rules count
1 Permission entry
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

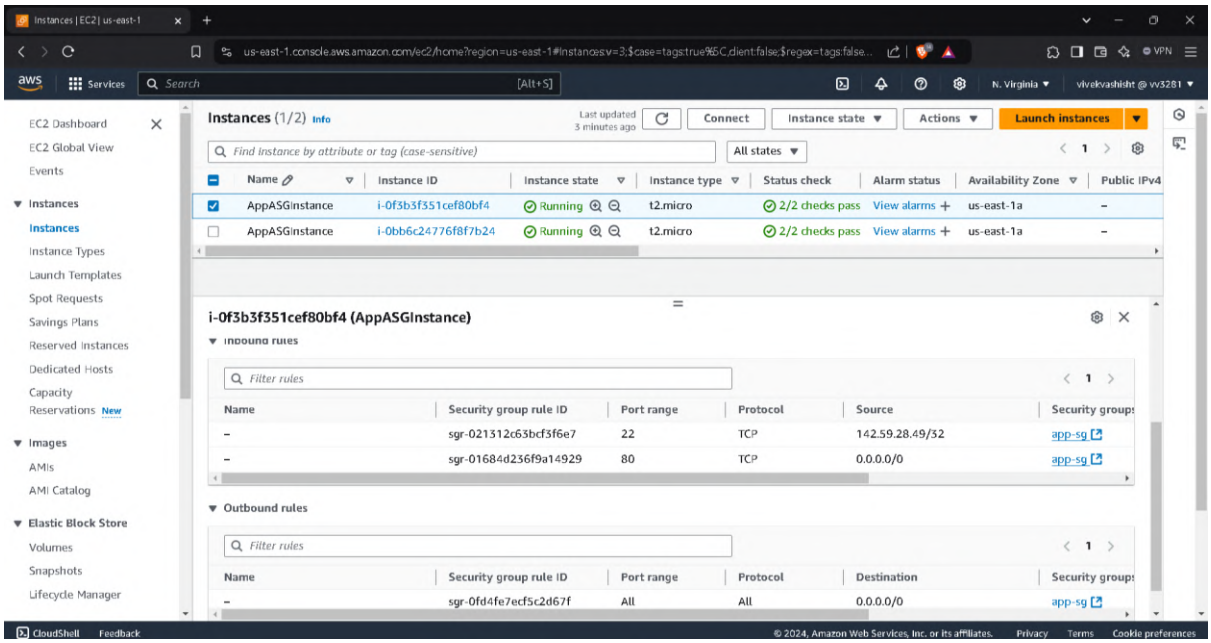

SecurityGroups | VPC Console
us-east-1.console.aws.amazon.com/vpcconsole/home?region=us-east-1#SecurityGroups:
Services
Search
[Alt+S]
N. Virginia
vivekvashisht @ vv3281
Egress-only internet gateways
Carrier gateways
DHCP option sets
Elastic IPs
Managed prefix lists
Endpoints
Endpoint services
NAT gateways
Peering connections
Security
Network ACLs
Security groups
DNS firewall
Rule groups
Domain lists
Network Firewall
Firewalls
Firewall policies
Network Firewall rule groups
Security Groups (1/3) Info
Actions
Export security groups to CSV
Create security group
Find resources by attribute or tag
Name
Security group ID
Security group name
VPC ID
Description
sg-0a6e24127747c69dd
default
vpc-01bb7cb53a9c88ffd
sg-026fe863d61fffc21
vpc-0ea415b9040f753dc
AppSecurityGroup
sg-0bf07049429bd4a81
app-sg
Allow HTTP and
Details
Inbound rules
Outbound rules
Tags
Inbound rules (2/2)
Manage tags
Edit inbound rules
Search
Name
Security group rule ID
IP version
Type
Protocol
Port range
Source
Description
sgr-021312c63bcf3f6e7
IPv4
SSH
TCP
22
142.59.28.49/32
sgr-01684d236f9a14929
IPv4
HTTP
TCP
80
0.0.0.0/0
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

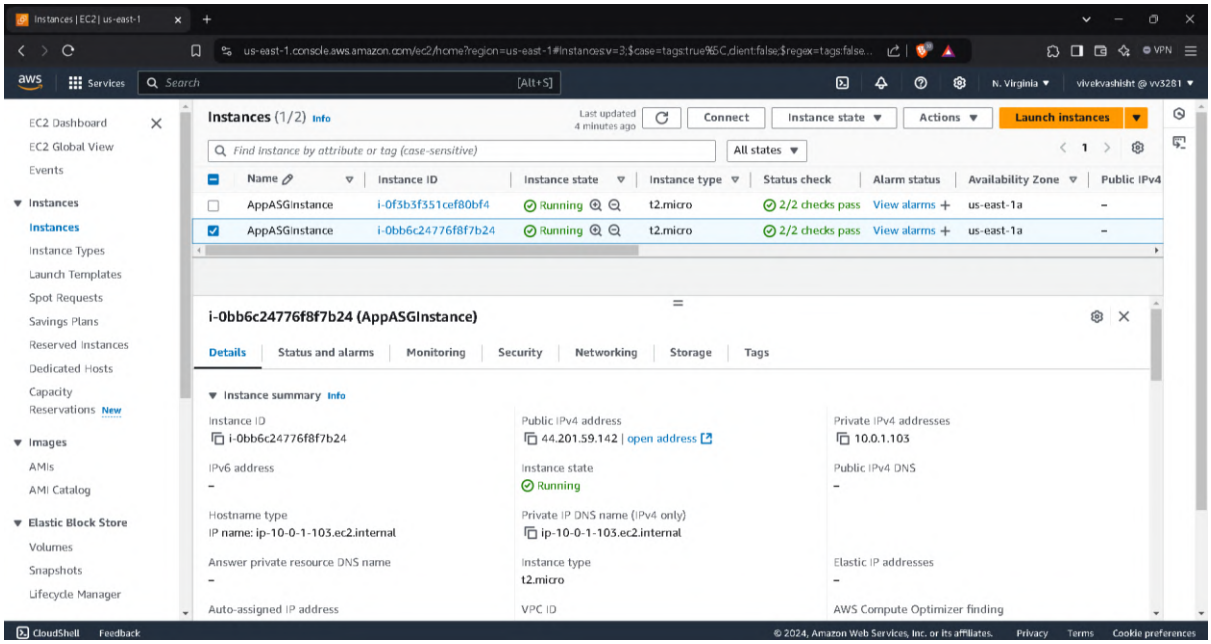

SecurityGroups | VPC Console
us-east-1.console.aws.amazon.com/vpcconsole/home?region=us-east-1#SecurityGroups:
Services
Search
[Alt+S]
N. Virginia
vivekvashisht @ vv3281
Egress-only internet gateways
Carrier gateways
DHCP option sets
Elastic IPs
Managed prefix lists
Endpoints
Endpoint services
NAT gateways
Peering connections
Security
Network ACLs
Security groups
DNS firewall
Rule groups
Domain lists
Network Firewall
Firewalls
Firewall policies
Network Firewall rule groups
Security Groups (1/3) Info
Actions
Export security groups to CSV
Create security group
Find resources by attribute or tag
Name
Security group ID
Security group name
VPC ID
Description
sg-0a6e24127747c69dd
default
vpc-01bb7cb53a9c88ffd
default VPC sec
sg-026fe863d61fffc21
vpc-0ea415b9040f753dc
AppSecurityGroup
sg-0bf07049429bd4a81
app-sg
Allow HTTP and
Details
Inbound rules
Outbound rules
Tags
Outbound rules (1/1)
Manage tags
Edit outbound rules
Search
Name
Security group rule ID
IP version
Type
Protocol
Port range
Destination
Description
sgr-0fd4fe7ecf5c2d67f
IPv4
All traffic
All
All
0.0.0.0/0
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

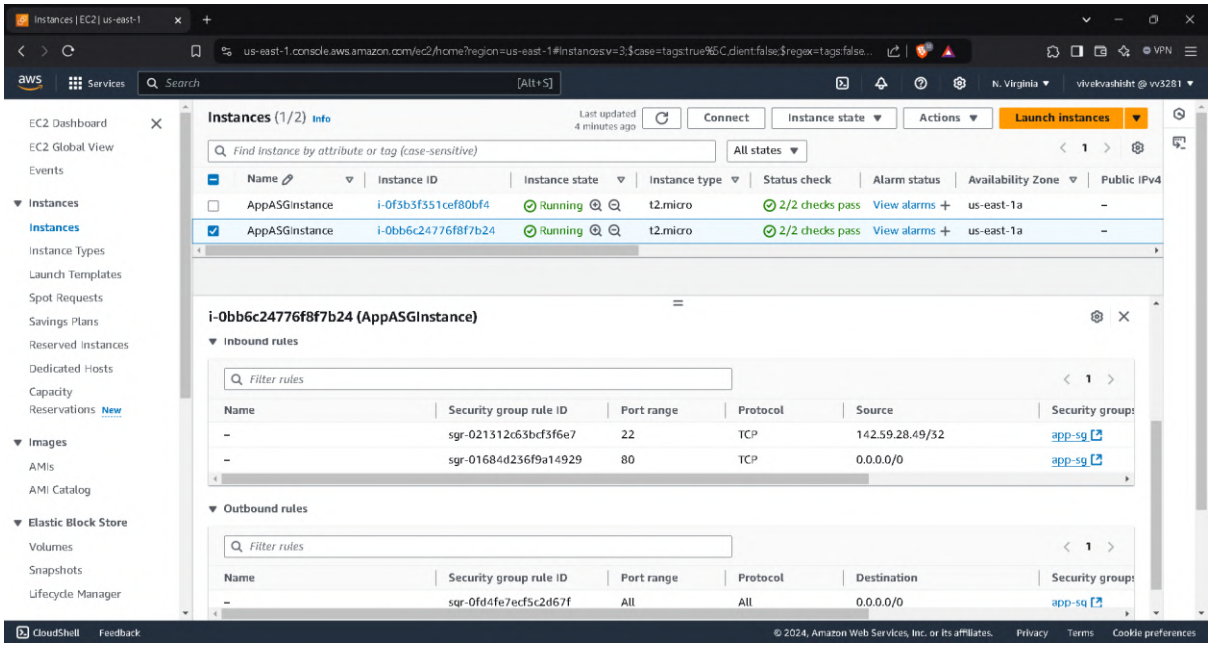

Launch templates | EC2 | us-east-1
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LaunchTemplates:
Services
Search
[Alt+S]
N. Virginia
vivekvashisht @ vv3281
EC2 Dashboard
EC2 Global View
Events
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations New
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Launch Templates (1/1) Info
Actions
Create launch template
Search
Launch Template ID
Launch Template Name
Default Version
Latest Version
Create Time
Created By
lt-04ebe7149bc419c5c
app-launch-template202410...
1
1
2024-10-25T06:27:21.0...
arn:aws:iam::...
app-launch-template20241025062721490700000001 (lt-04ebe7149bc419c5c)
Launch template details
Actions
Delete template
Launch template ID
lt-04ebe7149bc419c5c
Launch template name
app-launch-template20241025062721490700000001
Default version
1
Owner
arn:aws:iam::134575801911:user/vivekvashisht
Details
Versions
Template tags
Launch template version details
Actions
Delete template version
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

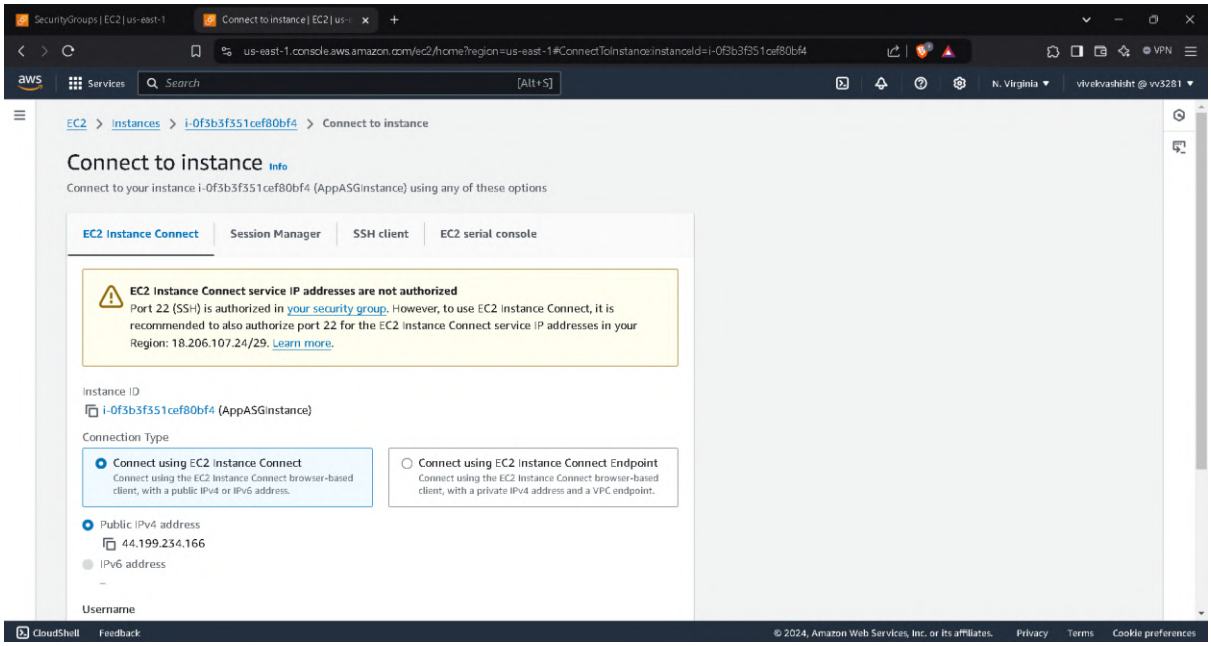

Launch templates | EC2 | us-east-1
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LaunchTemplates:
Services
Search
[Alt+S]
N. Virginia
vivekvashisht @ vv3281
EC2 Dashboard
EC2 Global View
Events
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations New
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Launch Templates (1/1) Info
Actions
Create launch template
Search
Launch Template ID
Launch Template Name
Default Version
Latest Version
Create Time
Created By
lt-04ebe7149bc419c5c
app-launch-template202410...
1
1
2024-10-25T06:27:21.0...
arn:aws:iam::...
app-launch-template20241025062721490700000001 (lt-04ebe7149bc419c5c)
Version
1 (Default)
Description
-
Date created
2024-10-25T06:27:21.000Z
Created by
arn:aws:iam::134575801911:user/vivekvashisht
Instance details
Storage
Resource tags
Network interfaces
Advanced details
AMI ID
ami-06b21ccaeff8cd686
Instance type
t2.micro
Availability Zone
-
Key pair name
-
Security groups
-
Security group IDs
sg-0bf07049429bd4a81
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

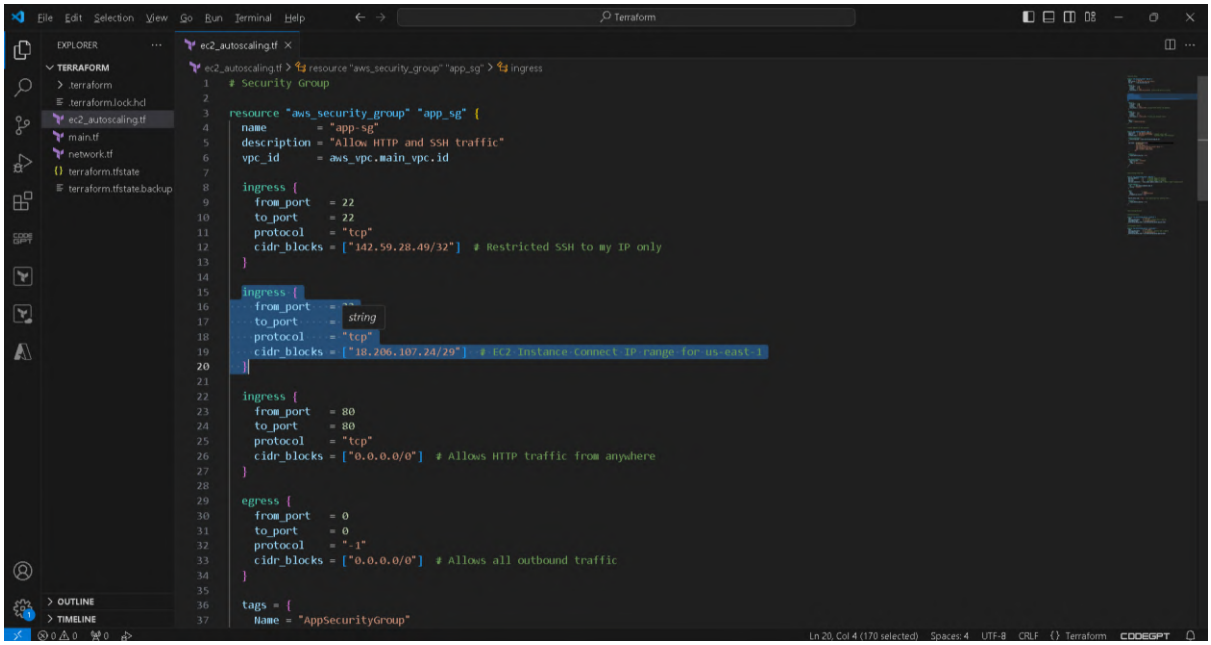

Instances | EC2 | us-east-1
Services
Search
[Alt+S]
N. Virginia
vivekvashisht @ vv3281
EC2 Dashboard
EC2 Global View
Events
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations New
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Instances (1/2) Info
Last updated 2 minutes ago
Connect
Instance state
Actions
Launch instances
Find instance by attribute or tag (case-sensitive)
All states
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
Availability Zone
Public IPv4
AppASGinstance
i-0f3b3f351cef80bf4
Running
t2.micro
2/2 checks pass
View alarms
us-east-1a
AppASGinstance
i-0bb6c24776f8f7b24
Running
t2.micro
2/2 checks pass
View alarms
us-east-1a
i-0f3b3f351cef80bf4 (AppASGInstance)
Details
Status and alarms
Monitoring
Security
Networking
Storage
Tags
Instance summary Info
Instance ID
i-0f3b3f351cef80bf4
Public IPv4 address
44.199.234.166 | open address
Private IPv4 addresses
10.0.1.61
IPv6 address
Instance state
Running
Public IPv4 DNS
Hostname type
IP name: ip-10-0-1-61.ec2.internal
Private IP DNS name (IPv4 only)
ip-10-0-1-61.ec2.internal
Answer private resource DNS name
Instance type
t2.micro
Elastic IP addresses
Auto-assigned IP address
VPC ID
AWS Compute Optimizer finding
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

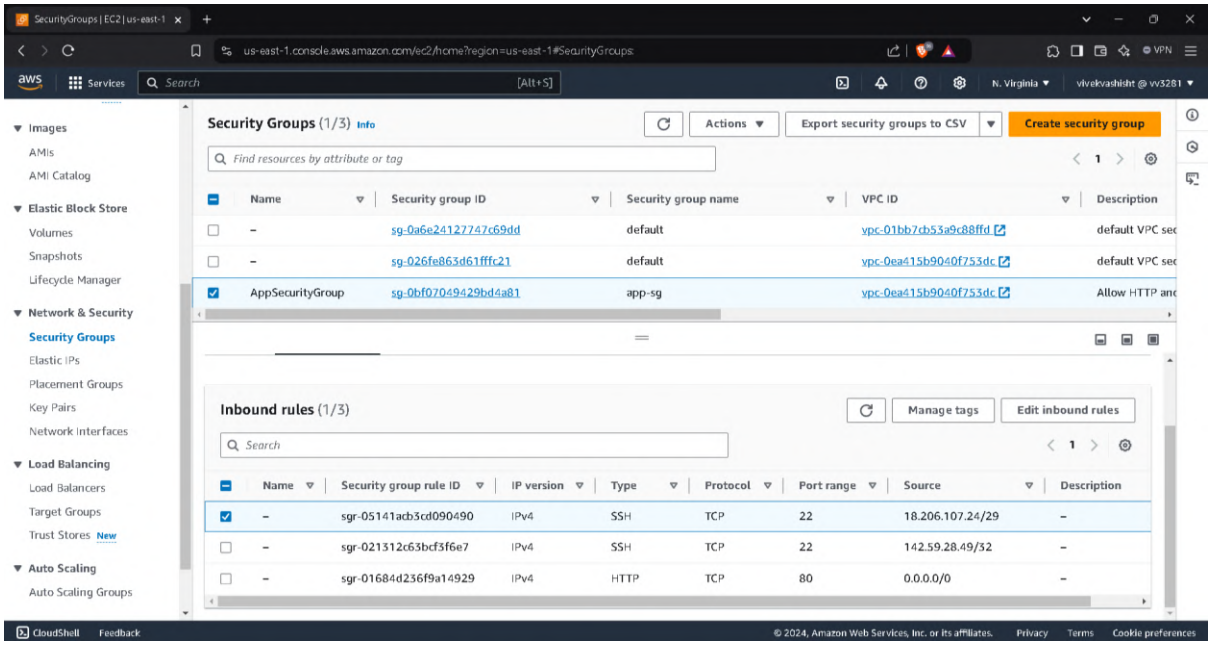

Instances | EC2 | us-east-1
Services
Search
[Alt+S]
N. Virginia
vivekvashisht @ vv3281
Instances (1/2) Info
Last updated 3 minutes ago
Connect
Instance state
Actions
Launch instances
Find instance by attribute or tag (case-sensitive)
All states
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
Availability Zone
Public IPv4
AppASGinstance
i-0f3b3f351cef80bf4
Running
t2.micro
2/2 checks pass
View alarms
us-east-1a
AppASGinstance
i-0bb6c24776f8f7b24
Running
t2.micro
2/2 checks pass
View alarms
us-east-1a
i-0f3b3f351cef80bf4 (AppASGInstance)
Inbound rules
Filter rules
Name
Security group rule ID
Port range
Protocol
Source
Security groups
sgr-021312c63bcf3f6e7
22
TCP
142.59.28.49/32
app-sg
sgr-01684d236f9a14929
80
TCP
0.0.0.0/0
app-sg
Outbound rules
Filter rules
Name
Security group rule ID
Port range
Protocol
Destination
Security groups
sgr-0fd4fe7ecf5c2d67f
All
All
0.0.0.0/0
app-sg
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

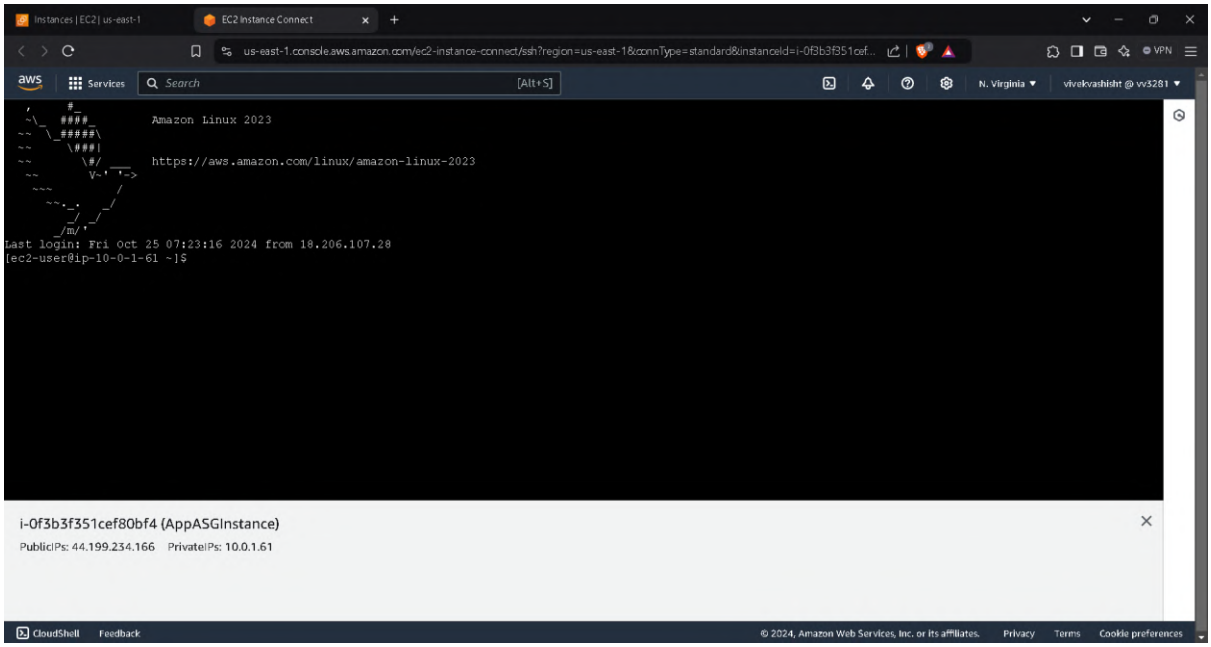

Instances | EC2 | us-east-1
Services
Search
[Alt+S]
N. Virginia
vivekvashisht @ vv3281
Instances (1/2) Info
Last updated 4 minutes ago
Connect
Instance state
Actions
Launch instances
Find instance by attribute or tag (case-sensitive)
All states
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
Availability Zone
Public IPv4
AppASGinstance
i-0f3b3f351cef80bf4
Running
t2.micro
2/2 checks pass
View alarms
us-east-1a
AppASGinstance
i-0bb6c24776f8f7b24
Running
t2.micro
2/2 checks pass
View alarms
us-east-1a
i-0bb6c24776f8f7b24 (AppASGInstance)
Details
Status and alarms
Monitoring
Security
Networking
Storage
Tags
Instance summary Info
Instance ID
i-0bb6c24776f8f7b24
Public IPv4 address
44.201.59.142 | open address
Private IPv4 addresses
10.0.1.103
IPv6 address
Instance state
Running
Public IPv4 DNS
Hostname type
IP name: ip-10-0-1-103.ec2.internal
Private IP DNS name (IPv4 only)
ip-10-0-1-103.ec2.internal
Answer private resource DNS name
Instance type
t2.micro
Elastic IP addresses
Auto-assigned IP address
VPC ID
AWS Compute Optimizer finding
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

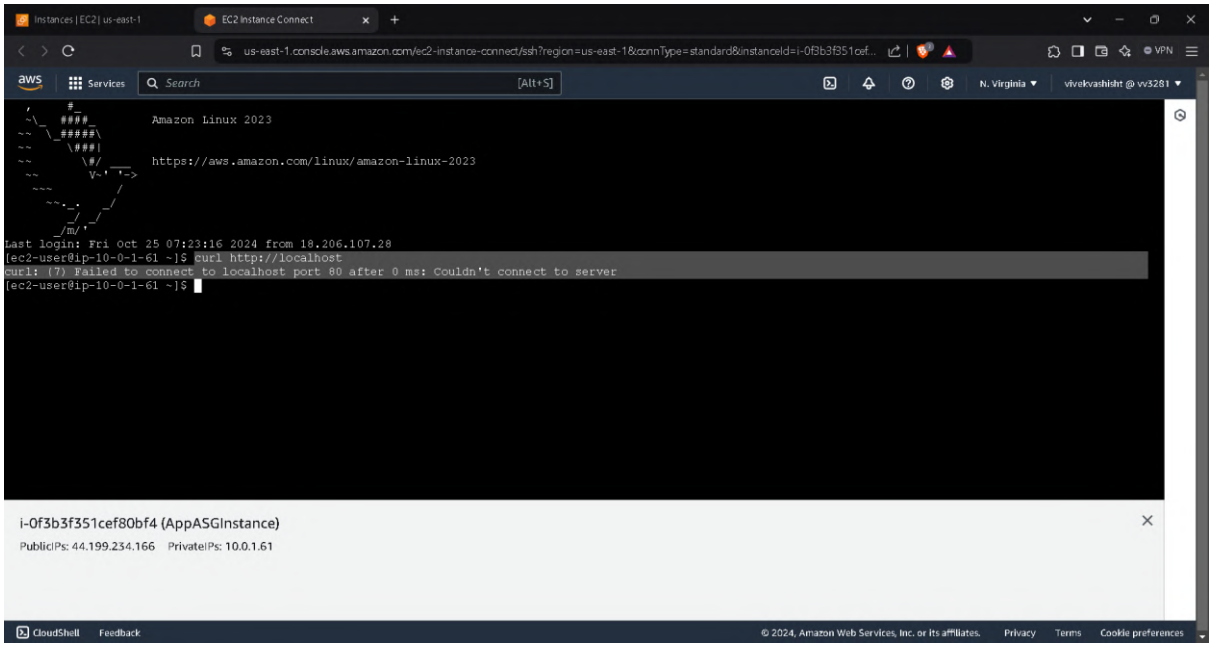

Instances | EC2 | us-east-1
Instances (1/2) Info
Connect
Instance state
Actions
Launch instances
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
Availability Zone
Public IPv4
AppASGInstance
i-0f3b3f351cef80bf4
Running
t2.micro
2/2 checks pass
View alarms
us-east-1a
AppASGInstance
i-0bb6c24776f8f7b24
Running
t2.micro
2/2 checks pass
View alarms
us-east-1a
i-0bb6c24776f8f7b24 (AppASGInstance)
Inbound rules
Name
Security group rule ID
Port range
Protocol
Source
Security groups
sgr-021312c63bcf3f6e7
22
TCP
142.59.28.49/32
app-sg
sgr-01684d236f9a14929
80
TCP
0.0.0.0/0
app-sg
Outbound rules
Destination
sgr-0fd4fe7ecf5c2d67f
All
All
0.0.0.0/0
app-sg
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

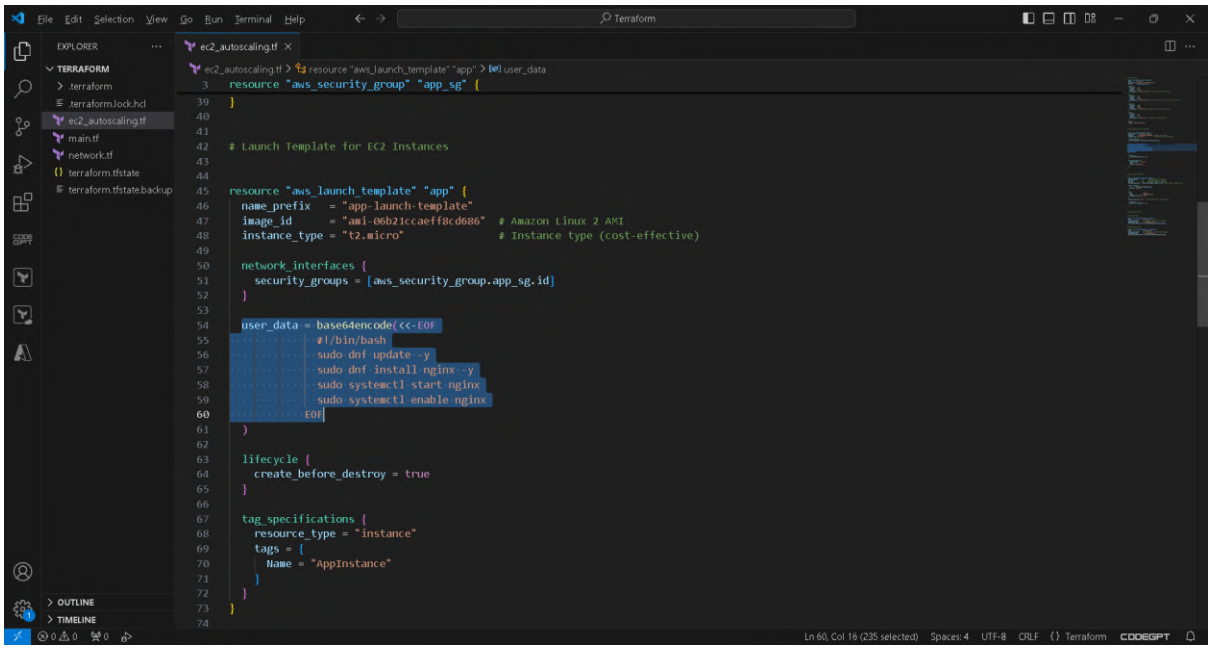

EC2 > Instances > i-0f3b3f351cef80bf4 > Connect to instance
Connect to instance Info
Connect to your instance i-0f3b3f351cef80bf4 (AppASGInstance) using any of these options
EC2 Instance Connect
Session Manager
SSH client
EC2 serial console
EC2 Instance Connect service IP addresses are not authorized
Port 22 (SSH) is authorized in your security group. However, to use EC2 Instance Connect, it is recommended to also authorize port 22 for the EC2 Instance Connect service IP addresses in your Region: 18.206.107.24/29. Learn more.
Instance ID
i-0f3b3f351cef80bf4 (AppASGInstance)
Connection Type
Connect using EC2 Instance Connect
Connect using the EC2 Instance Connect browser-based client, with a public IPv4 or IPv6 address.
Connect using EC2 Instance Connect Endpoint
Connect using the EC2 Instance Connect browser-based client, with a private IPv4 address and a VPC endpoint.
Public IPv4 address
44.199.234.166
IPv6 address
Username
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


EXPLORER
TERRAFORM
.terraform
.terraform.lock.hcl
ec2_autoscaling.tf
main.tf
network.tf
terraform.tfstate
terraform.tfstate.backup
ec2_autoscaling.tf
# Security Group
resource "aws_security_group" "app_sg" {
name = "app-sg"
description = "Allow HTTP and SSH traffic"
vpc_id = aws_vpc.main_vpc.id
ingress {
from_port = 22
to_port = 22
protocol = "tcp"
cidr_blocks = ["142.59.28.49/32"] # Restricted SSH to my IP only
}
ingress {
from_port = "22"
to_port = "22"
protocol = "tcp"
cidr_blocks = ["18.206.107.24/29"] # EC2 Instance Connect IP range for us-east-1
}
ingress {
from_port = 80
to_port = 80
protocol = "tcp"
cidr_blocks = ["0.0.0.0/0"] # Allows HTTP traffic from anywhere
}
egress {
from_port = 0
to_port = 0
protocol = "-1"
cidr_blocks = ["0.0.0.0/0"] # Allows all outbound traffic
}
tags = {
Name = "AppSecurityGroup"
string
OUTLINE
TIMELINE

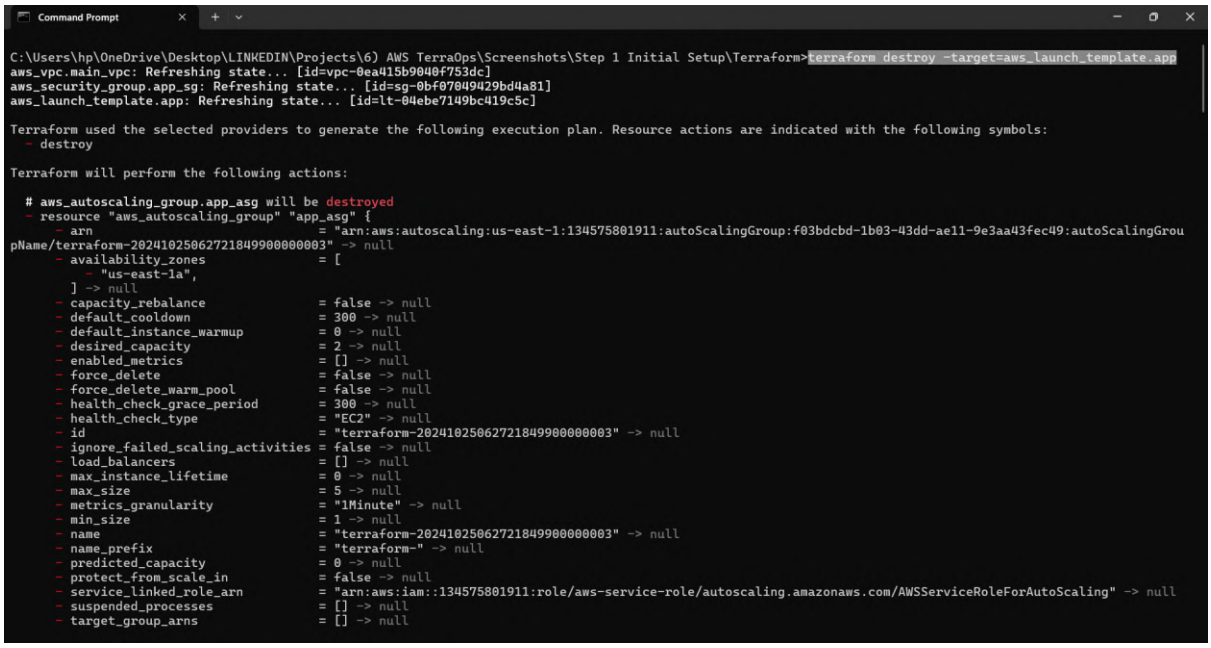

SecurityGroups | EC2 | us-east-1
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#SecurityGroups:
Services
Search
[Alt+S]
N. Virginia
vivekvashisht @ vv3281
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Network & Security
Security Groups
Elastic IPs
Placement Groups
Key Pairs
Network Interfaces
Load Balancing
Load Balancers
Target Groups
Trust Stores New
Auto Scaling
Auto Scaling Groups
Security Groups (1/3) Info
Actions
Export security groups to CSV
Create security group
Find resources by attribute or tag
Name
Security group ID
Security group name
VPC ID
Description
sg-0a6e24127747c69dd
default
vpc-01bb7cb53a9c88ffd
default VPC sec
sg-026fe863d61fffc21
vpc-0ea415b9040f753dc
AppSecurityGroup
sg-0bf07049429bd4a81
app-sg
Allow HTTP and
Inbound rules (1/3)
Manage tags
Edit inbound rules
Search
Security group rule ID
IP version
Type
Protocol
Port range
Source
sgr-05141acb3cd090490
IPv4
SSH
TCP
22
18.206.107.24/29
sgr-021312c63bcf3f6e7
142.59.28.49/32
sgr-01684d236f9a14929
HTTP
80
0.0.0.0/0
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Instances | EC2 | us-east-1
EC2 Instance Connect
us-east-1.console.aws.amazon.com/ec2-instance-connect/ssh?region=us-east-1&connType=standard&instanceId=i-0f3b3f351cef...
Services
Search
[Alt+S]
N. Virginia
vivekvashisht @ vv3281
Amazon Linux 2023
https://aws.amazon.com/linux/amazon-linux-2023
Last login: Fri Oct 25 07:23:16 2024 from 18.206.107.28
[ec2-user@ip-10-0-1-61 ~]$
i-0f3b3f351cef80bf4 (AppASGInstance)
PublicIPs: 44.199.234.166 PrivateIPs: 10.0.1.61
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

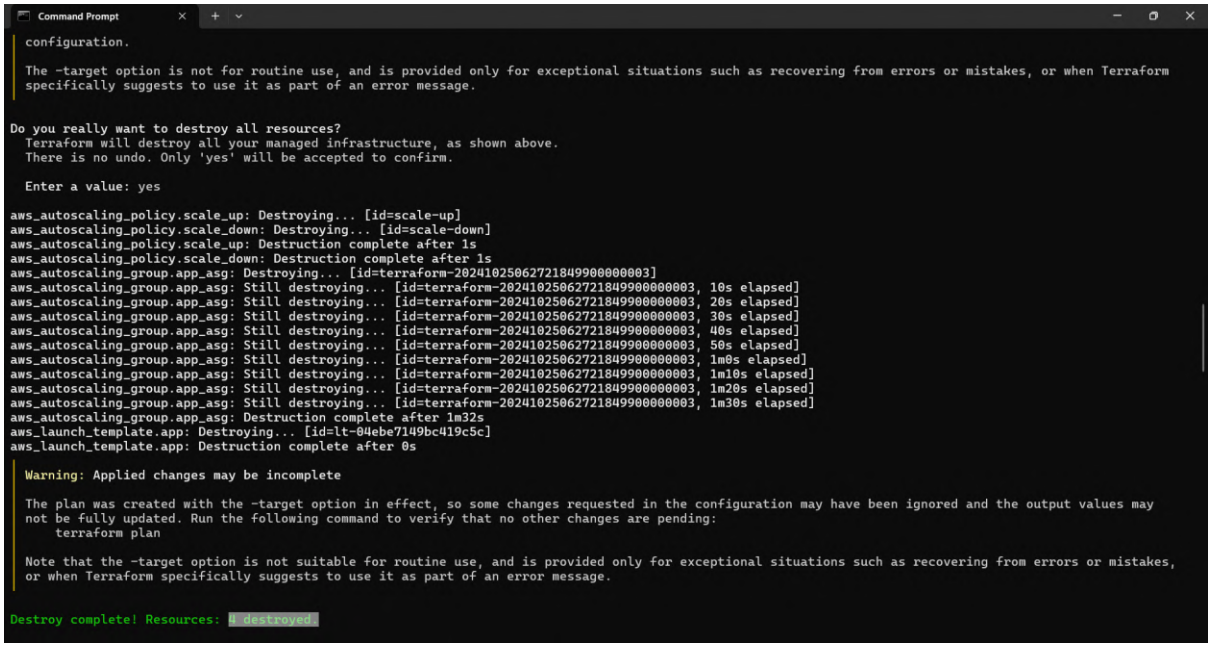

Instances | EC2 | us-east-1
EC2 Instance Connect
us-east-1.console.aws.amazon.com/ec2-instance-connect/ssh?region=us-east-1&connType=standard&instanceId=i-0f3b3f351cef...
Services
Search
[Alt+S]
N. Virginia
vivekvashisht @ vv3281
Amazon Linux 2023
https://aws.amazon.com/linux/amazon-linux-2023
Last login: Fri Oct 25 07:23:16 2024 from 18.206.107.28
[ec2-user@ip-10-0-1-61 ~]$ curl http://localhost
curl: (7) Failed to connect to localhost port 80 after 0 ms: Couldn't connect to server
[ec2-user@ip-10-0-1-61 ~]$
i-0f3b3f351cef80bf4 (AppASGInstance)
PublicIPs: 44.199.234.166 PrivateIPs: 10.0.1.61
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences



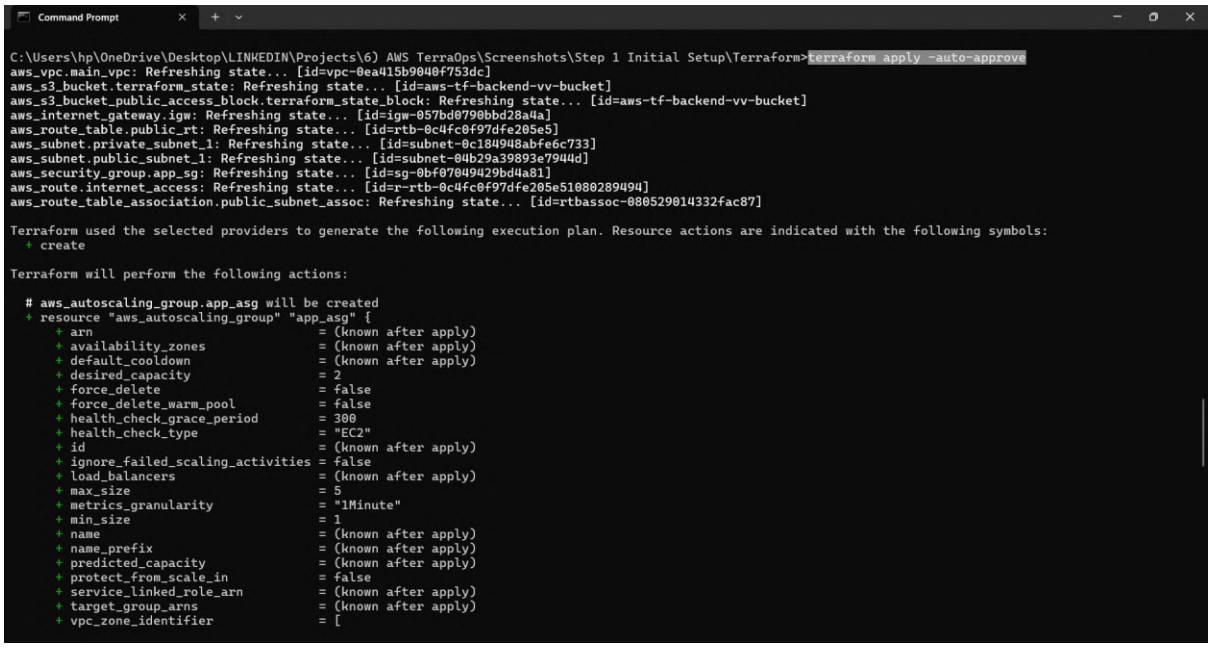
```
resource "aws_security_group" "app_sg" {
}


# Launch Template for EC2 Instances


resource "aws_launch_template" "app" {
  name_prefix   = "app-launch-template"
  image_id      = "ami-06b21ccaeff8cd686"  # Amazon Linux 2 AMI
  instance_type = "t2.micro"               # Instance type (cost-effective)

  network_interfaces {
    security_groups = [aws_security_group.app_sg.id]
  }

  user_data = base64encode(<<-EOF
              #!/bin/bash
              sudo dnf update -y
              sudo dnf install nginx -y
              sudo systemctl start nginx
              sudo systemctl enable nginx
              EOF
  )

  lifecycle {
    create_before_destroy = true
  }

  tag_specifications {
    resource_type = "instance"
    tags = {
      Name = "AppInstance"
    }
  }
}
```

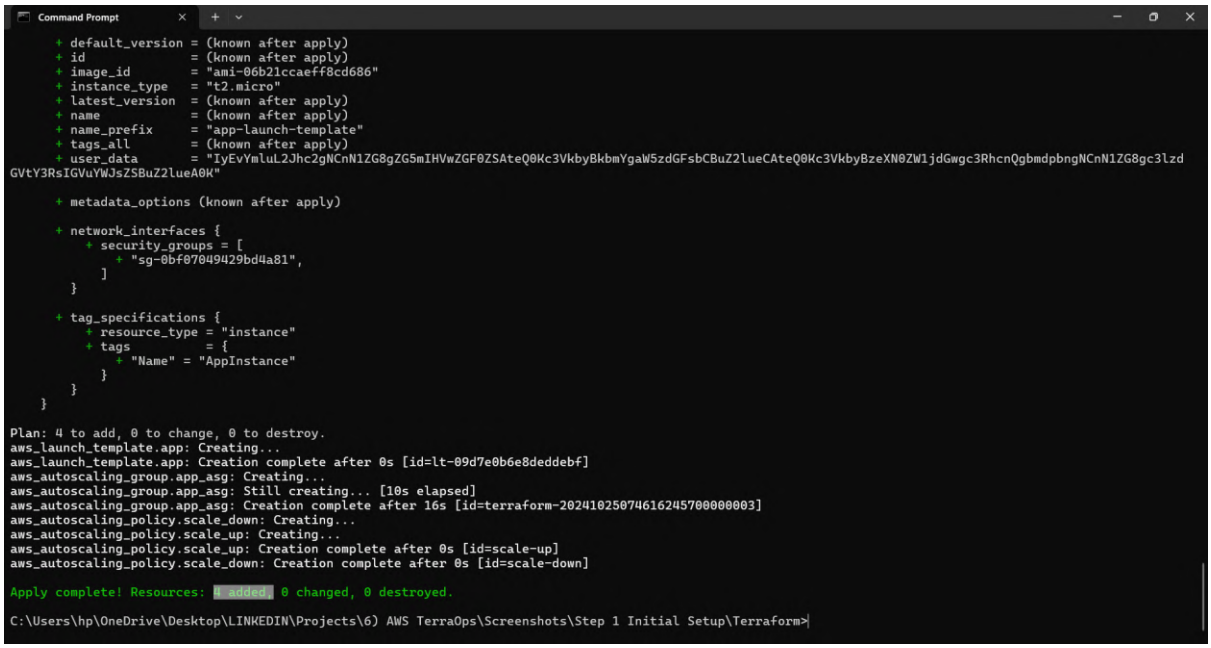


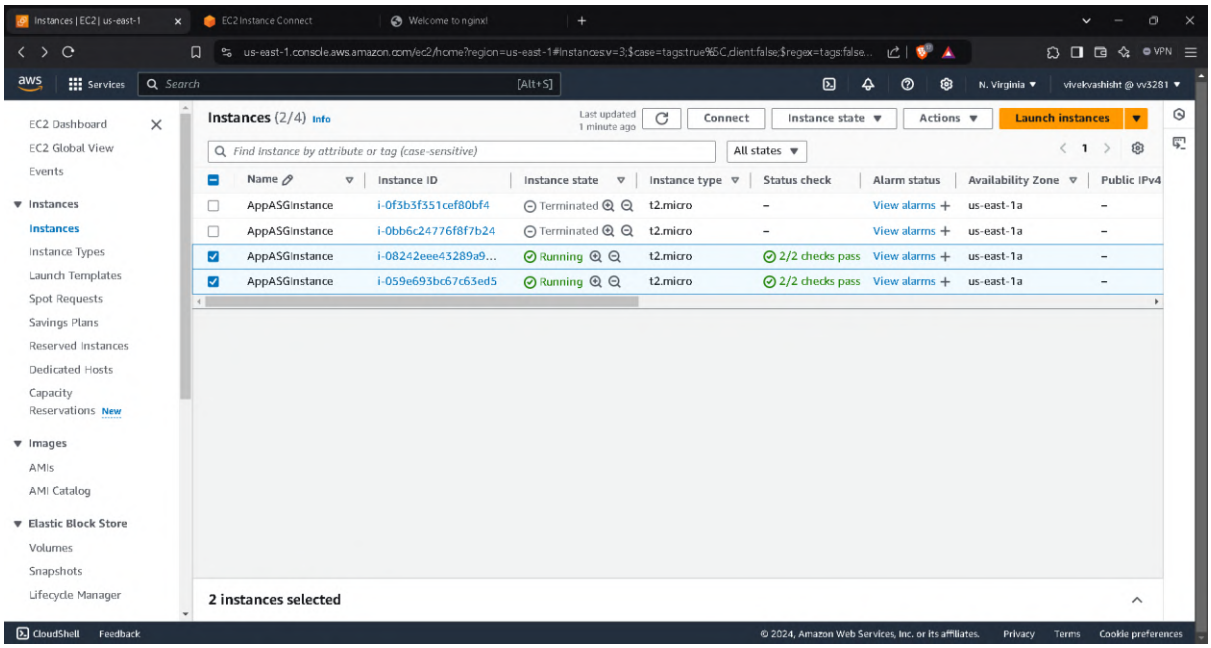
```
C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>terraform destroy -target=aws_launch_template.app
aws_vpc.main_vpc: Refreshing state... [id=vpc-0ea415b9040f753dc]
aws_security_group.app_sg: Refreshing state... [id=sg-0bf07049429bd4a81]
aws_launch_template.app: Refreshing state... [id=lt-04ebe7149bc419c5c]

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

Terraform will perform the following actions:

  # aws_autoscaling_group.app_asg will be destroyed
  - resource "aws_autoscaling_group" "app_asg" {
      - arn                       = "arn:aws:autoscaling:us-east-1:134575801911:autoScalingGroup:f03bdcbd-1b03-43dd-ae11-9e3aa43fec49:autoScalingGrou
pName/terraform-20241025062721849900000003" -> null
      - availability_zones        = [
          - "us-east-1a",
        ] -> null
      - capacity_rebalance        = false -> null
      - default_cooldown          = 300 -> null
      - default_instance_warmup   = 0 -> null
      - desired_capacity          = 2 -> null
      - enabled_metrics           = [] -> null
      - force_delete              = false -> null
      - force_delete_warm_pool    = false -> null
      - health_check_grace_period = 300 -> null
      - health_check_type         = "EC2" -> null
      - id                        = "terraform-20241025062721849900000003" -> null
      - ignore_failed_scaling_activities = false -> null
      - load_balancers            = [] -> null
      - max_instance_lifetime     = 0 -> null
      - max_size                  = 5 -> null
      - metrics_granularity       = "1Minute" -> null
      - min_size                  = 1 -> null
      - name                      = "terraform-20241025062721849900000003" -> null
      - name_prefix               = "terraform-" -> null
      - predicted_capacity        = 0 -> null
      - protect_from_scale_in     = false -> null
      - service_linked_role_arn   = "arn:aws:iam::134575801911:role/aws-service-role/autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling" -> null
      - suspended_processes       = [] -> null
      - target_group_arns         = [] -> null
```

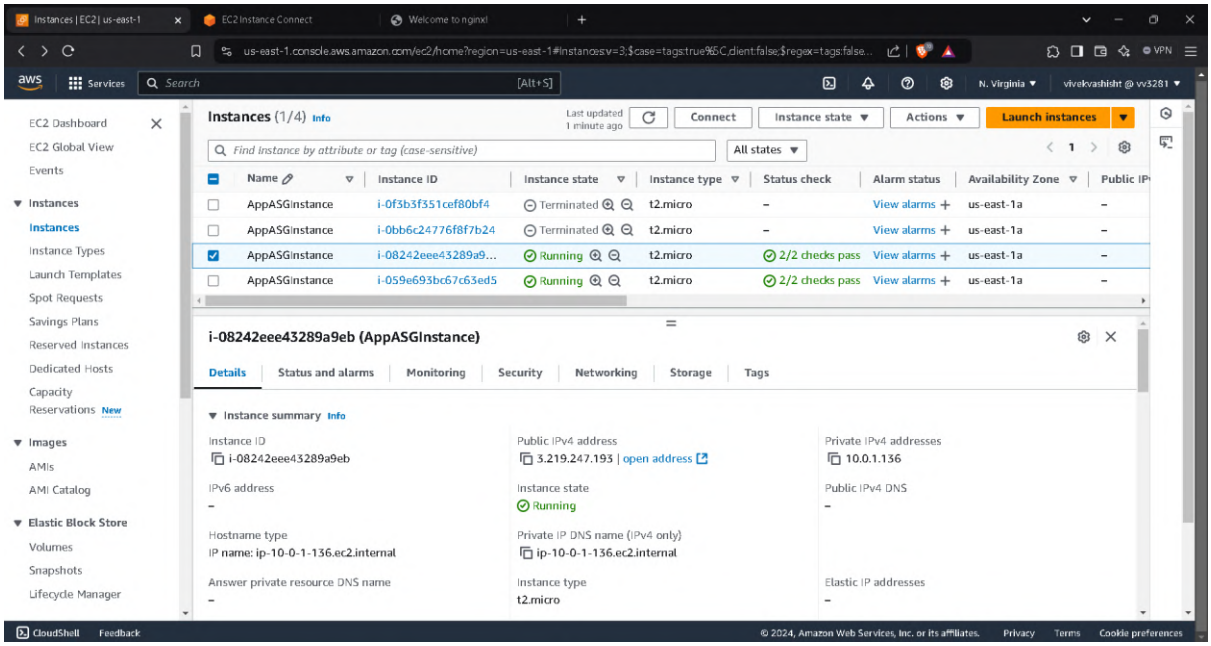


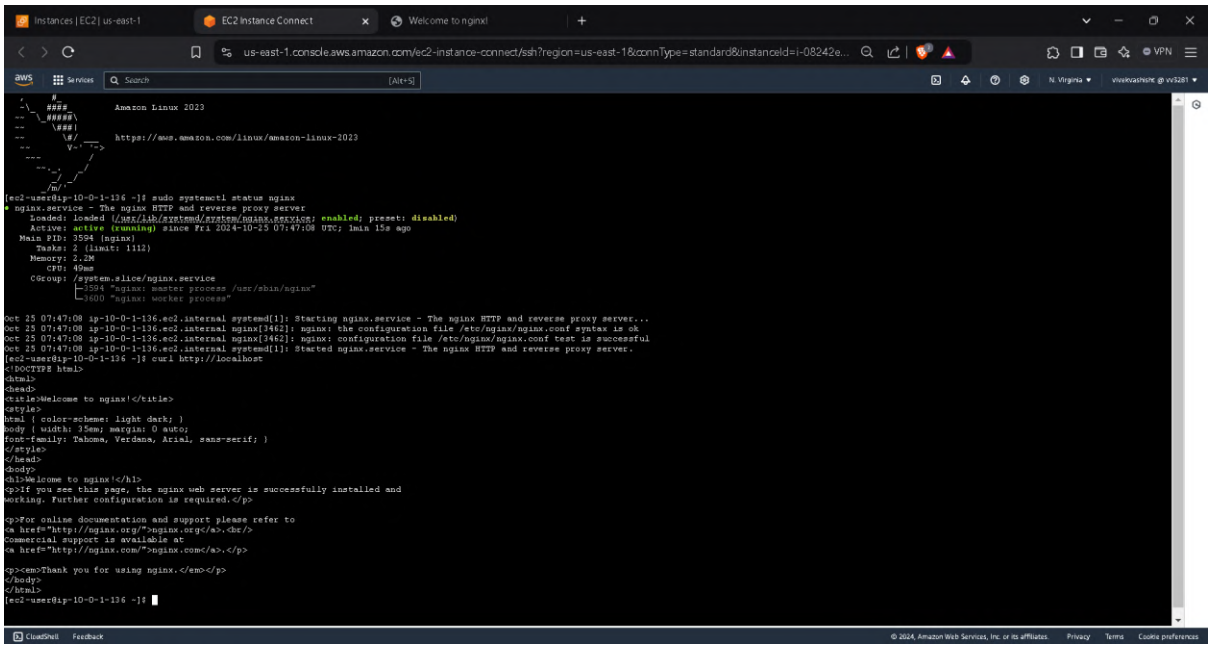
```
  configuration.

  The -target option is not for routine use, and is provided only for exceptional situations such as recovering from errors or mistakes, or when Terraform
  specifically suggests to use it as part of an error message.


Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

aws_autoscaling_policy.scale_up: Destroying... [id=scale-up]
aws_autoscaling_policy.scale_down: Destroying... [id=scale-down]
aws_autoscaling_policy.scale_up: Destruction complete after 1s
aws_autoscaling_policy.scale_down: Destruction complete after 1s
aws_autoscaling_group.app_asg: Destroying... [id=terraform-20241025062721849900000003]
aws_autoscaling_group.app_asg: Still destroying... [id=terraform-20241025062721849900000003, 10s elapsed]
aws_autoscaling_group.app_asg: Still destroying... [id=terraform-20241025062721849900000003, 20s elapsed]
aws_autoscaling_group.app_asg: Still destroying... [id=terraform-20241025062721849900000003, 30s elapsed]
aws_autoscaling_group.app_asg: Still destroying... [id=terraform-20241025062721849900000003, 40s elapsed]
aws_autoscaling_group.app_asg: Still destroying... [id=terraform-20241025062721849900000003, 50s elapsed]
aws_autoscaling_group.app_asg: Still destroying... [id=terraform-20241025062721849900000003, 1m0s elapsed]
aws_autoscaling_group.app_asg: Still destroying... [id=terraform-20241025062721849900000003, 1m10s elapsed]
aws_autoscaling_group.app_asg: Still destroying... [id=terraform-20241025062721849900000003, 1m20s elapsed]
aws_autoscaling_group.app_asg: Still destroying... [id=terraform-20241025062721849900000003, 1m30s elapsed]
aws_autoscaling_group.app_asg: Destruction complete after 1m32s
aws_launch_template.app: Destroying... [id=lt-04ebe7149bc419c5c]
aws_launch_template.app: Destruction complete after 0s

  Warning: Applied changes may be incomplete

  The plan was created with the -target option in effect, so some changes requested in the configuration may have been ignored and the output values may
  not be fully updated. Run the following command to verify that no other changes are pending:
      terraform plan

  Note that the -target option is not suitable for routine use, and is provided only for exceptional situations such as recovering from errors or mistakes,
  or when Terraform specifically suggests to use it as part of an error message.

Destroy complete! Resources: 4 destroyed.
```

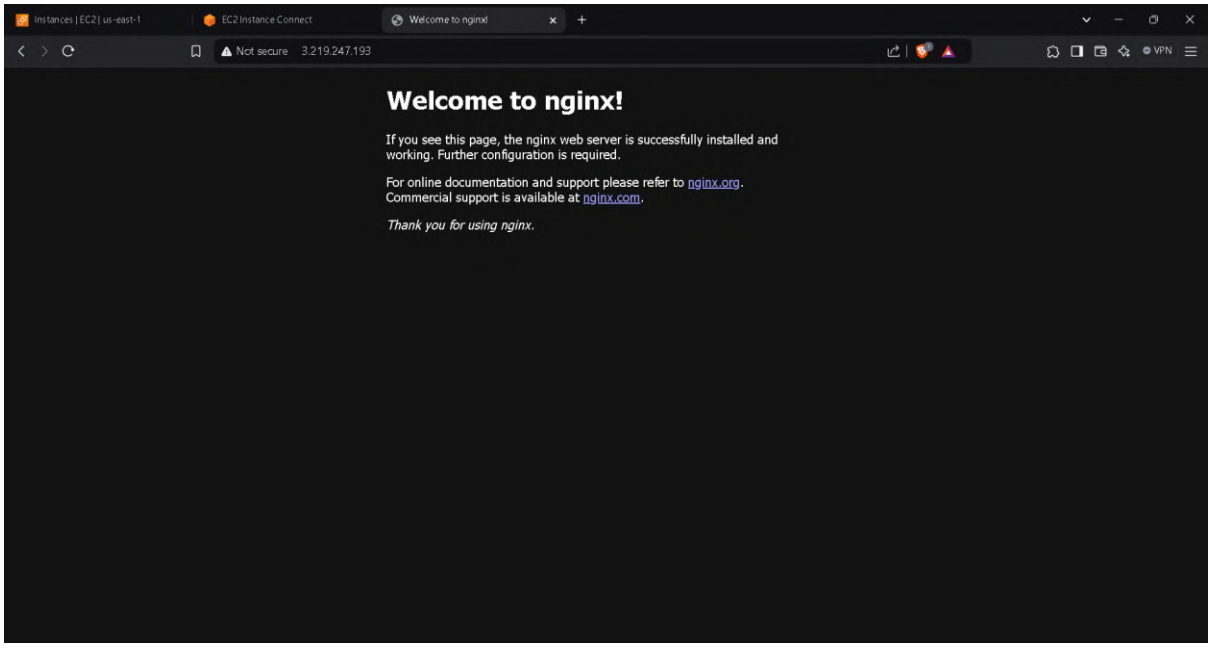

Command Prompt
C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>terraform apply -auto-approve
aws_vpc.main_vpc: Refreshing state... [id=vpc-0ea415b9040f753dc]
aws_s3_bucket.terraform_state: Refreshing state... [id=aws-tf-backend-vv-bucket]
aws_s3_bucket_public_access_block.terraform_state_block: Refreshing state... [id=aws-tf-backend-vv-bucket]
aws_internet_gateway.igw: Refreshing state... [id=igw-057bd0790bbd28a4a]
aws_route_table.public_rt: Refreshing state... [id=rtb-0c4fc0f97dfe205e5]
aws_subnet.private_subnet_1: Refreshing state... [id=subnet-0c184948abfe6c733]
aws_subnet.public_subnet_1: Refreshing state... [id=subnet-04b29a39893e7944d]
aws_security_group.app_sg: Refreshing state... [id=sg-0bf07049429bd4a81]
aws_route.internet_access: Refreshing state... [id=r-rtb-0c4fc0f97dfe205e51080289494]
aws_route_table_association.public_subnet_assoc: Refreshing state... [id=rtbassoc-080529014332fac87]
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_autoscaling_group.app_asg will be created
+ resource "aws_autoscaling_group" "app_asg" {
+ arn = (known after apply)
+ availability_zones = (known after apply)
+ default_cooldown = (known after apply)
+ desired_capacity = 2
+ force_delete = false
+ force_delete_warm_pool = false
+ health_check_grace_period = 300
+ health_check_type = "EC2"
+ id = (known after apply)
+ ignore_failed_scaling_activities = false
+ load_balancers = (known after apply)
+ max_size = 5
+ metrics_granularity = "1Minute"
+ min_size = 1
+ name = (known after apply)
+ name_prefix = (known after apply)
+ predicted_capacity = (known after apply)
+ protect_from_scale_in = false
+ service_linked_role_arn = (known after apply)
+ target_group_arns = (known after apply)
+ vpc_zone_identifier = [

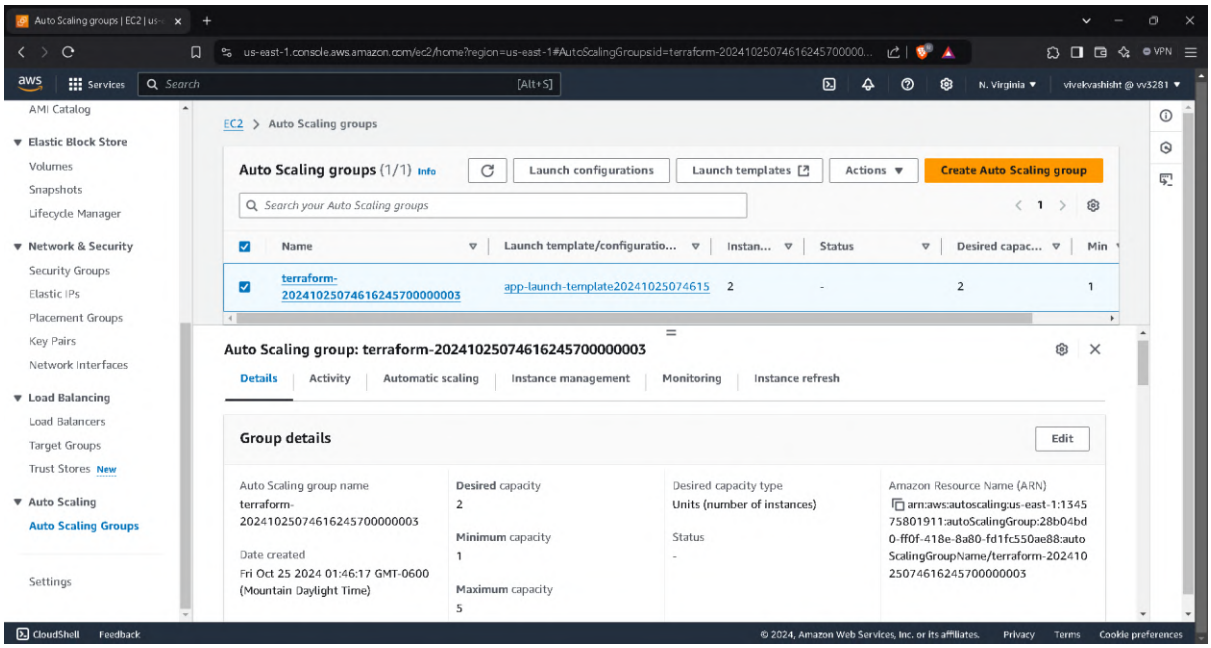

Command Prompt
+ default_version = (known after apply)
+ id = (known after apply)
+ image_id = "ami-06b21ccaeff8cd686"
+ instance_type = "t2.micro"
+ latest_version = (known after apply)
+ name = (known after apply)
+ name_prefix = "app-launch-template"
+ tags_all = (known after apply)
+ user_data = "IyEvYmluL2Jhc2gNCnN1ZG8gZG5mIHVwZGF0ZSAteQ0Kc3VkbyBkbmYgaW5zdGFsbCBuZ2lueCAteQ0Kc3VkbyBzeXN0ZW1jdGwgc3RhcnQgbmdpbngNCnN1ZG8gc3lzdGVtY3RsIGVuYWJsZSBuZ2lueA0K"
+ metadata_options (known after apply)
+ network_interfaces {
+ security_groups = [
+ "sg-0bf07049429bd4a81",
]
}
+ tag_specifications {
+ resource_type = "instance"
+ tags = {
+ "Name" = "AppInstance"
}
}
}
Plan: 4 to add, 0 to change, 0 to destroy.
aws_launch_template.app: Creating...
aws_launch_template.app: Creation complete after 0s [id=lt-09d7e0b6e8deddebf]
aws_autoscaling_group.app_asg: Creating...
aws_autoscaling_group.app_asg: Still creating... [10s elapsed]
aws_autoscaling_group.app_asg: Creation complete after 16s [id=terraform-20241025074616245700000003]
aws_autoscaling_policy.scale_down: Creating...
aws_autoscaling_policy.scale_up: Creating...
aws_autoscaling_policy.scale_up: Creation complete after 0s [id=scale-up]
aws_autoscaling_policy.scale_down: Creation complete after 0s [id=scale-down]
Apply complete! Resources: 4 added, 0 changed, 0 destroyed.
C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>


Instances | EC2 | us-east-1
EC2 Instance Connect
Welcome to nginx!
Services
Search
[Alt+S]
N. Virginia
vivekvashisht @ vv3281
EC2 Dashboard
EC2 Global View
Events
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations New
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Instances (2/4) Info
Last updated 1 minute ago
Connect
Instance state
Actions
Launch instances
Find instance by attribute or tag (case-sensitive)
All states
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
Availability Zone
Public IPv4
AppASGinstance i-0f3b3f351cef80bf4 Terminated t2.micro – View alarms us-east-1a –
AppASGinstance i-0bb6c24776f8f7b24 Terminated t2.micro – View alarms us-east-1a –
AppASGinstance i-08242eee43289a9... Running t2.micro 2/2 checks pass View alarms us-east-1a –
AppASGinstance i-059e693bc67c63ed5 Running t2.micro 2/2 checks pass View alarms us-east-1a –
2 instances selected
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Instances | EC2 | us-east-1
EC2 Instance Connect
Welcome to nginx!
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#instances:v=3;$case=tags:true%5C,client:false;$regex=tags:false...
aws
Services
Search
[Alt+S]
N. Virginia
vivekvashisht @ vv5281
EC2 Dashboard
EC2 Global View
Events
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations New
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Instances (1/4) Info
Last updated 1 minute ago
Connect
Instance state
Actions
Launch instances
Find instance by attribute or tag (case-sensitive)
All states
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
Availability Zone
Public IP
AppASGInstance i-0f3b3f351cef80bf4 Terminated t2.micro – View alarms us-east-1a –
AppASGInstance i-0bb6c24776f8f7b24 Terminated t2.micro – View alarms us-east-1a –
AppASGInstance i-08242eee43289a9... Running t2.micro 2/2 checks pass View alarms us-east-1a –
AppASGInstance i-059e693bc67c63ed5 Running t2.micro 2/2 checks pass View alarms us-east-1a –
i-08242eee43289a9eb (AppASGInstance)
Details
Status and alarms
Monitoring
Security
Networking
Storage
Tags
Instance summary Info
Instance ID
i-08242eee43289a9eb
Public IPv4 address
3.219.247.193 | open address
Private IPv4 addresses
10.0.1.136
IPv6 address
–
Instance state
Running
Public IPv4 DNS
–
Hostname type
IP name: ip-10-0-1-136.ec2.internal
Private IP DNS name (IPv4 only)
ip-10-0-1-136.ec2.internal
Answer private resource DNS name
–
Instance type
t2.micro
Elastic IP addresses
–
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Instances | EC2 | us-east-1
EC2 Instance Connect
Welcome to nginx!
us-east-1.console.aws.amazon.com/ec2-instance-connect/ssh?region=us-east-1&connType=standard&instanceId=i-08242e...
Amazon Linux 2023
https://aws.amazon.com/linux/amazon-linux-2023
[ec2-user@ip-10-0-1-136 ~]$ sudo systemctl status nginx
● nginx.service - The nginx HTTP and reverse proxy server
Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: disabled)
Active: active (running) since Fri 2024-10-25 07:47:08 UTC; 1min 15s ago
Main PID: 3594 (nginx)
Tasks: 2 (limit: 1112)
Memory: 2.2M
CPU: 49ms
CGroup: /system.slice/nginx.service
├─3594 "nginx: master process /usr/sbin/nginx"
└─3600 "nginx: worker process"
Oct 25 07:47:08 ip-10-0-1-136.ec2.internal systemd[1]: Starting nginx.service - The nginx HTTP and reverse proxy server...
Oct 25 07:47:08 ip-10-0-1-136.ec2.internal nginx[3462]: nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
Oct 25 07:47:08 ip-10-0-1-136.ec2.internal nginx[3462]: nginx: configuration file /etc/nginx/nginx.conf test is successful
Oct 25 07:47:08 ip-10-0-1-136.ec2.internal systemd[1]: Started nginx.service - The nginx HTTP and reverse proxy server.
[ec2-user@ip-10-0-1-136 ~]$ curl http://localhost
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.
For online documentation and support please refer to
nginx.org.
Commercial support is available at
nginx.com.
Thank you for using nginx.
</body>
</html>
[ec2-user@ip-10-0-1-136 ~]$
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Instances | EC2 | us-east-1
EC2 Instance Connect
Welcome to nginx!
Not secure 3.219.247.193
Welcome to nginx!
If you see this page, the nginx web server is successfully installed and working. Further configuration is required.
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.
Thank you for using nginx.



IAM Roles and S3 Integration

a) Overview

The objective of this phase was to set up IAM roles for EC2 instances to enable them to access S3 buckets securely, following AWS best practices. This setup involved defining IAM roles, updating instance configurations, and creating an additional S3 bucket for backups. Each step was carefully crafted to ensure the correct permissions and organization of resources in Terraform.

b) IAM Role Definition

To begin, I created a new file named `iam_roles.tf` in the Terraform project folder. This file defines an IAM role that allows EC2 instances to assume permissions required to access S3 resources. Here's the code for the IAM role, which includes a policy attachment to allow S3 read-only access:

```
# IAM Role for EC2 to assume
resource "aws_iam_role" "ec2_role" {
  name = "ec2-role"

  assume_role_policy = <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
```

```
}
EOF
}

# Attachment of S3 read-only policy to the role
resource "aws_iam_role_policy_attachment" "ec2_s3_readonly" {
  role       = aws_iam_role.ec2_role.name
  policy_arn = "arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess"
}
```

This configuration defines the EC2 role and attaches the `AmazonS3ReadOnlyAccess` policy to ensure EC2 instances can only read from S3, following the principle of least privilege.

c) Updating `ec2_autoscaling.tf` for IAM Role Attachment

Next, I updated the `ec2_autoscaling.tf` file to include the IAM role for the EC2 instances by defining an instance profile and attaching it to the launch template:

```
# IAM instance profile for EC2
resource "aws_iam_instance_profile" "ec2_instance_profile" {
  name = "ec2-instance-profile"
  role = aws_iam_role.ec2_role.name
}

# Attach the IAM role to EC2 instances
iam_instance_profile {
  name = aws_iam_instance_profile.ec2_instance_profile.name
}
```

This instance profile links the IAM role with EC2 instances, granting them the necessary permissions to access S3 resources.

d) S3 Bucket Configuration for Backups

For better resource organization and compliance with Terraform principles, I moved the configuration of the S3 bucket used for Terraform state storage from `main.tf` to a new file, `s3_bucket.tf`. This file also defines a new S3 bucket specifically for storing backups. The updated code for `s3_bucket.tf` is as follows:

```
# S3 bucket for storing Terraform state
resource "aws_s3_bucket" "terraform_state" {
  bucket = "aws-tf-backend-vv-bucket"

  tags = {
    Name        = "TerraformStateBucket"
    Environment = "Dev"
  }
}

# Public access block for the S3 bucket
resource "aws_s3_bucket_public_access_block" "terraform_state_block" {
  bucket = aws_s3_bucket.terraform_state.id

  block_public_acls       = true
  block_public_policy     = true
  restrict_public_buckets = true
```

```
  ignore_public_acls      = true
}

# S3 bucket for backups
resource "aws_s3_bucket" "backup_bucket" {
  bucket = "terra-backup-vv-project-bucket"

  tags = {
    Name        = "BackupBucket"
    Environment = "Dev"
  }
}

# Public access block for the backup S3 bucket
resource "aws_s3_bucket_public_access_block" "backup_bucket_block" {
  bucket = aws_s3_bucket.backup_bucket.id

  block_public_acls       = true
  block_public_policy     = true
  restrict_public_buckets = true
  ignore_public_acls      = true
}
```

The S3 bucket `terra-backup-vv-project-bucket` is configured for backup storage with public access blocked for security. Both S3 buckets have access controls to ensure they are secure and compliant.

e) Applying the Changes

After configuring `iam_roles.tf` and `s3_bucket.tf`, I followed these steps to apply the changes:

1. **Destroying the Existing Launch Template** – Since the updated instance profile would only take effect on new instances, I first destroyed the current launch template:

   ```
   terraform destroy -target=aws_launch_template.app
   ```

2. **Re-initializing and Applying Terraform** – I then ran the following commands to initialize, plan, and apply the updated configurations:

   ```
   terraform init
   terraform plan
   terraform apply -auto-approve
   ```

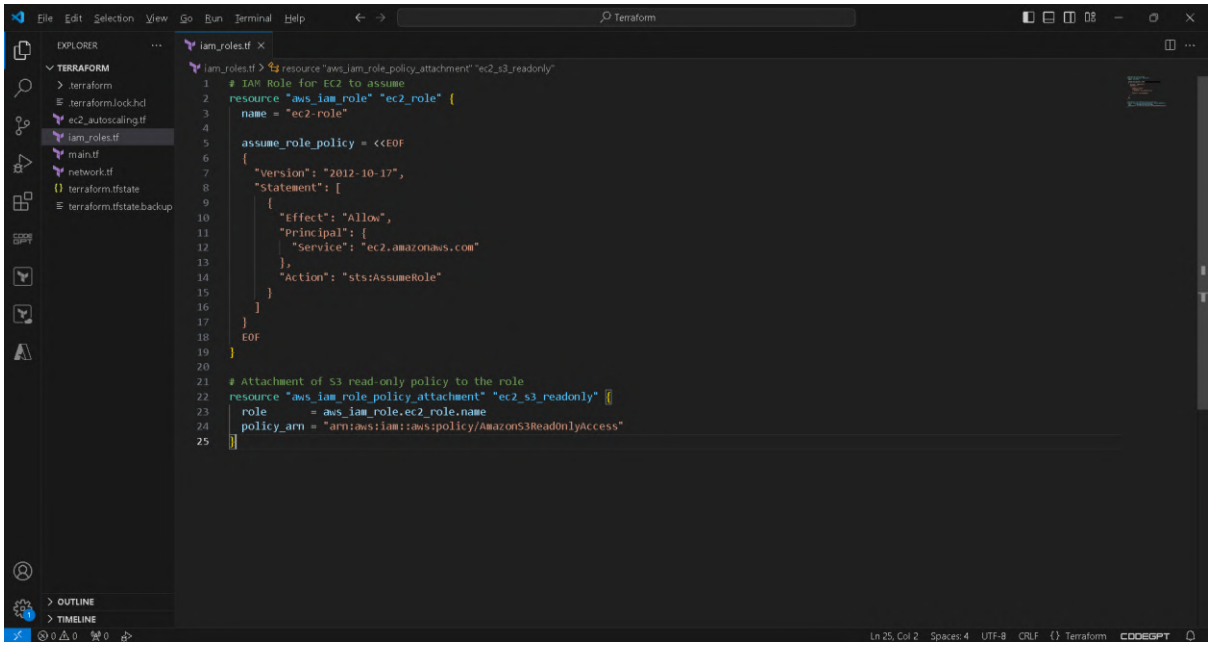
f) Verification

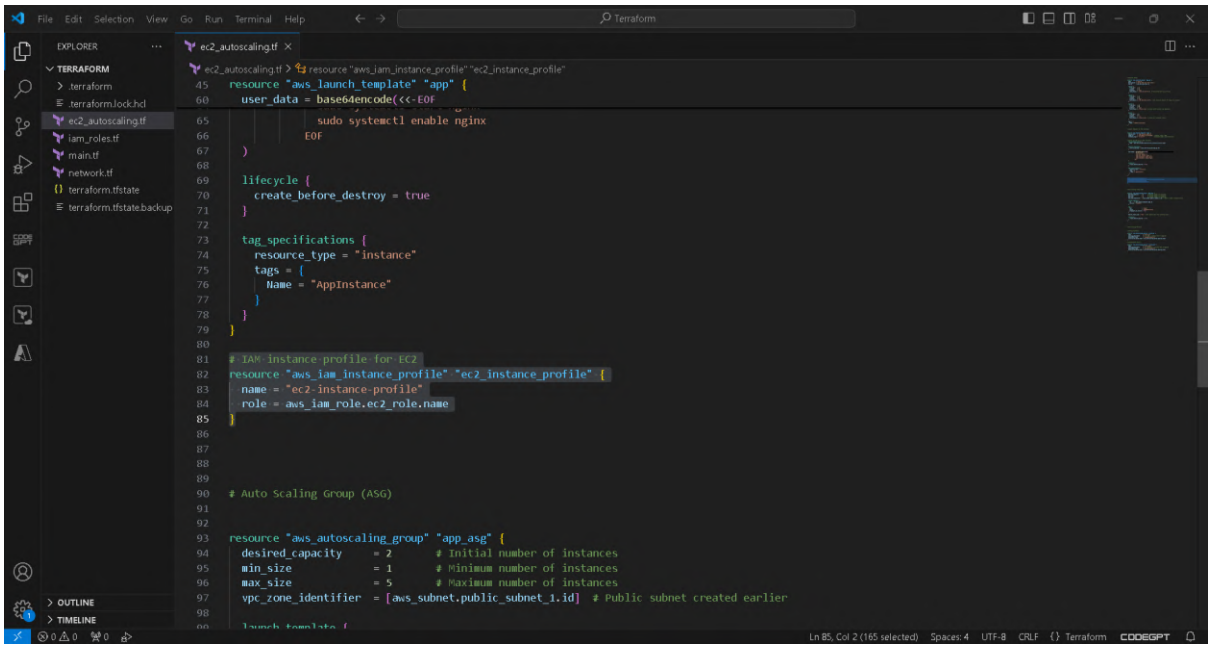
After a successful apply, I verified the resources in the AWS Console:

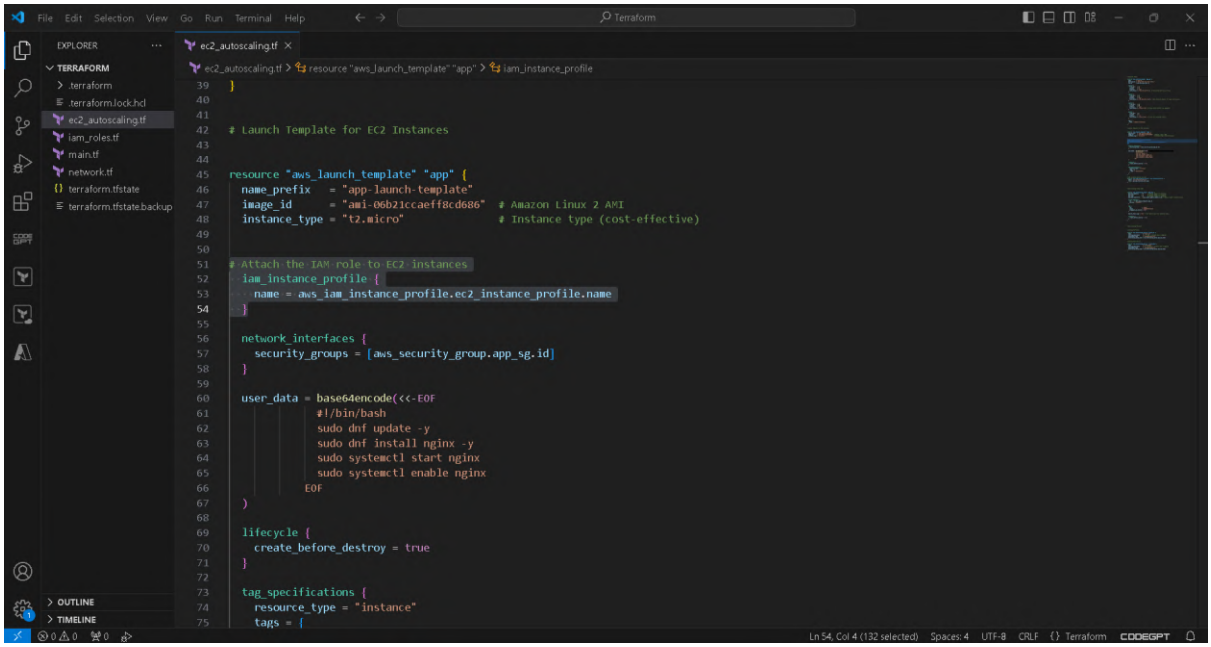
- **IAM Role and Policy** – The `ec2-role` with `AmazonS3ReadOnlyAccess` was successfully created and attached to the EC2 instances.
- **EC2 Instances** – Each instance had the `ec2-role` attached, allowing them to access the S3 bucket with read-only permissions.
- **S3 Buckets** – The `terra-backup-vv-project-bucket` was created and configured as expected, in addition to the Terraform state bucket.

This phase of the project was successfully completed with the EC2 instances securely configured to access S3 resources, following least-privilege access principles and terraform best practices.

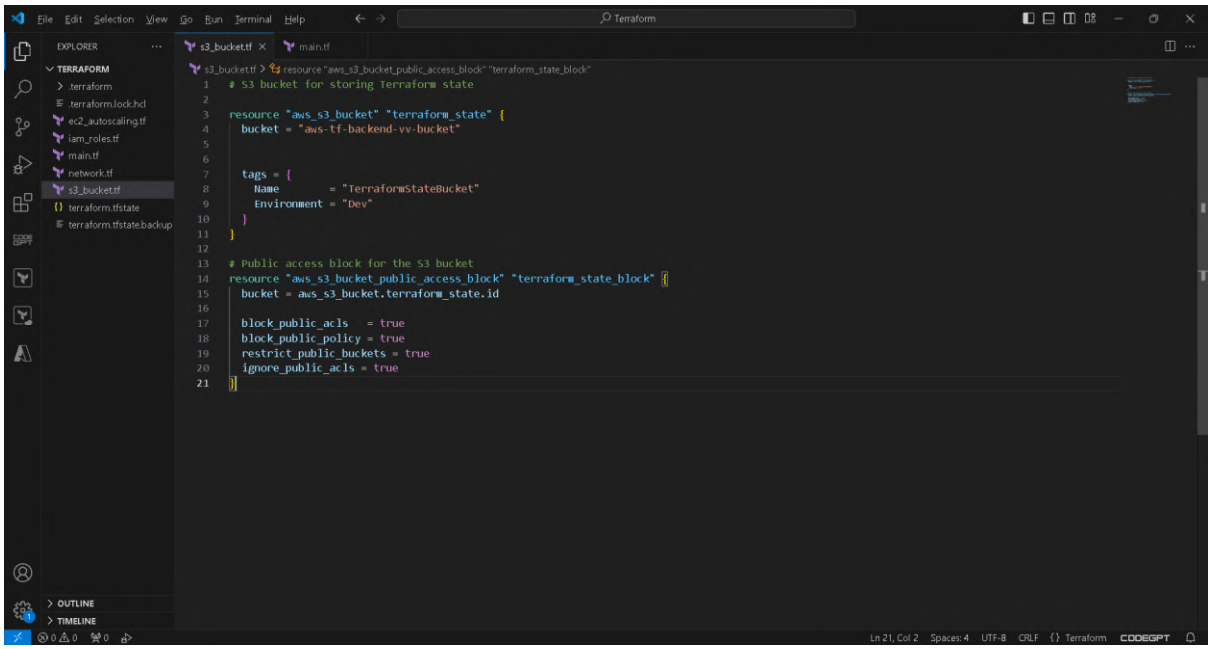
Screenshots



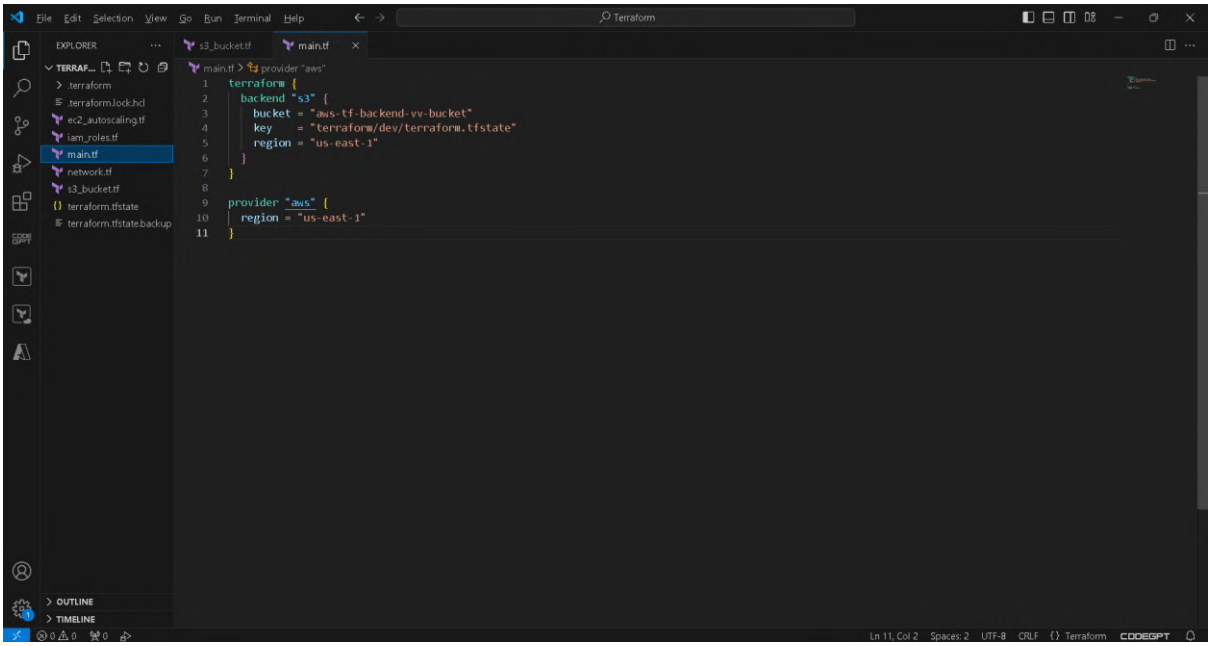




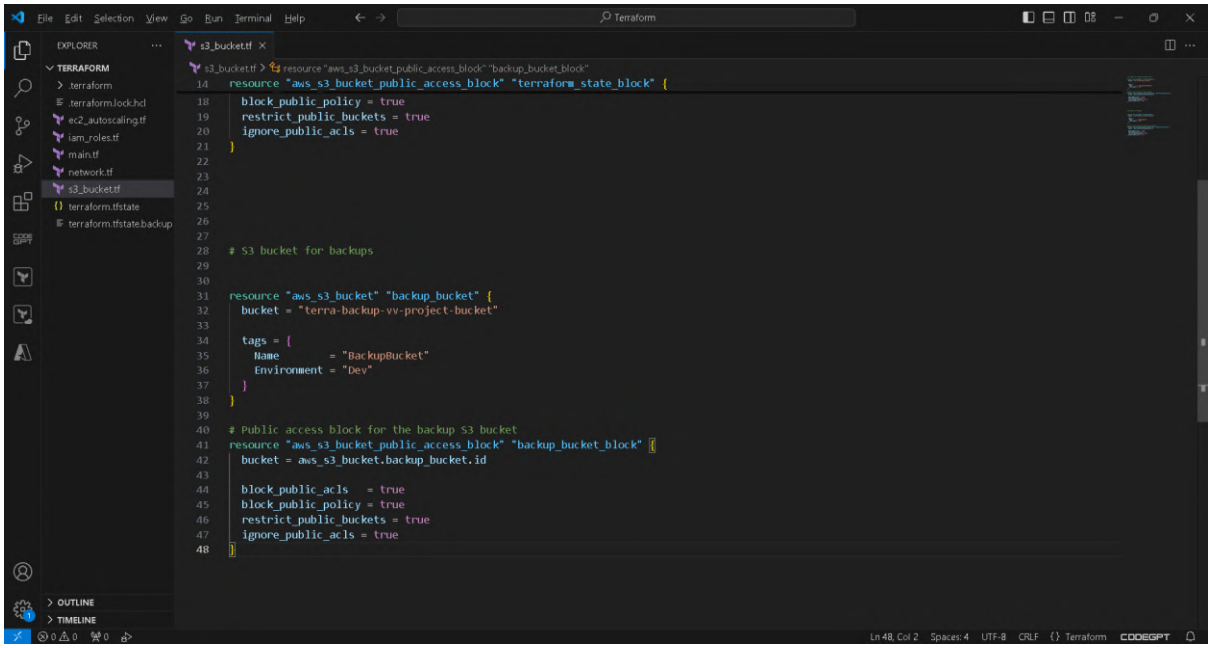
```
}


# Launch Template for EC2 Instances


resource "aws_launch_template" "app" {
  name_prefix   = "app-launch-template"
  image_id      = "ami-06b21ccaeff8cd686"  # Amazon Linux 2 AMI
  instance_type = "t2.micro"               # Instance type (cost-effective)


# Attach the IAM role to EC2 instances
  iam_instance_profile {
    name = aws_iam_instance_profile.ec2_instance_profile.name
  }

  network_interfaces {
    security_groups = [aws_security_group.app_sg.id]
  }

  user_data = base64encode(<<-EOF
              #!/bin/bash
              sudo dnf update -y
              sudo dnf install nginx -y
              sudo systemctl start nginx
              sudo systemctl enable nginx
              EOF
  )

  lifecycle {
    create_before_destroy = true
  }

  tag_specifications {
    resource_type = "instance"
    tags = {
```

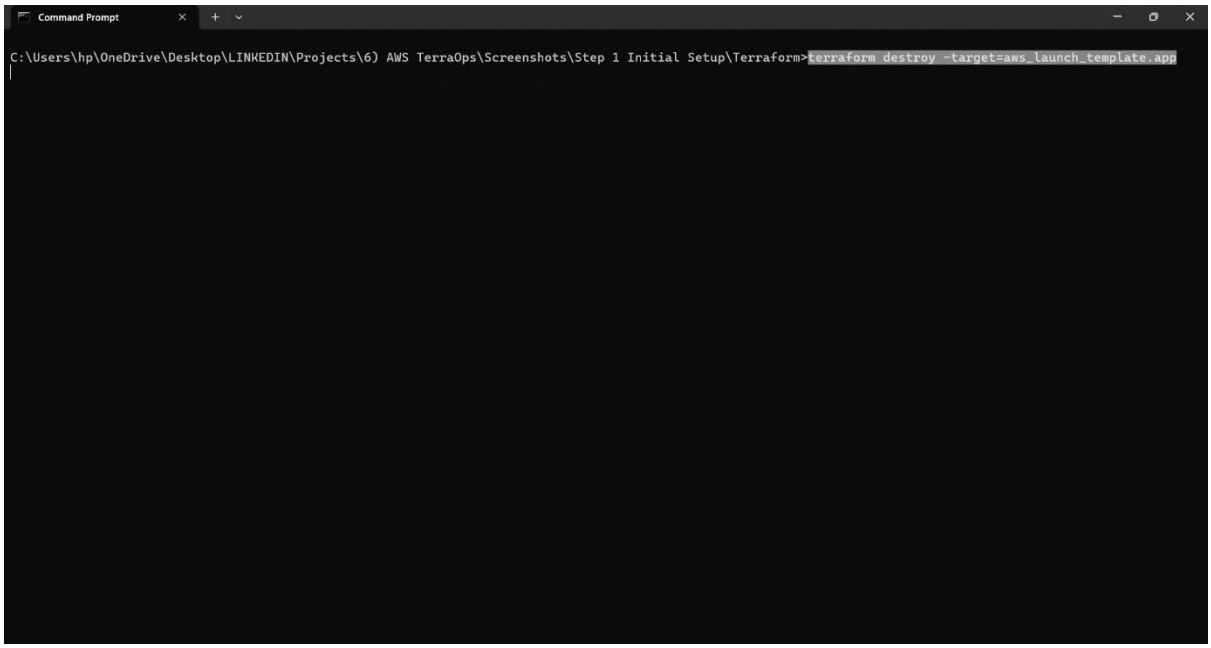


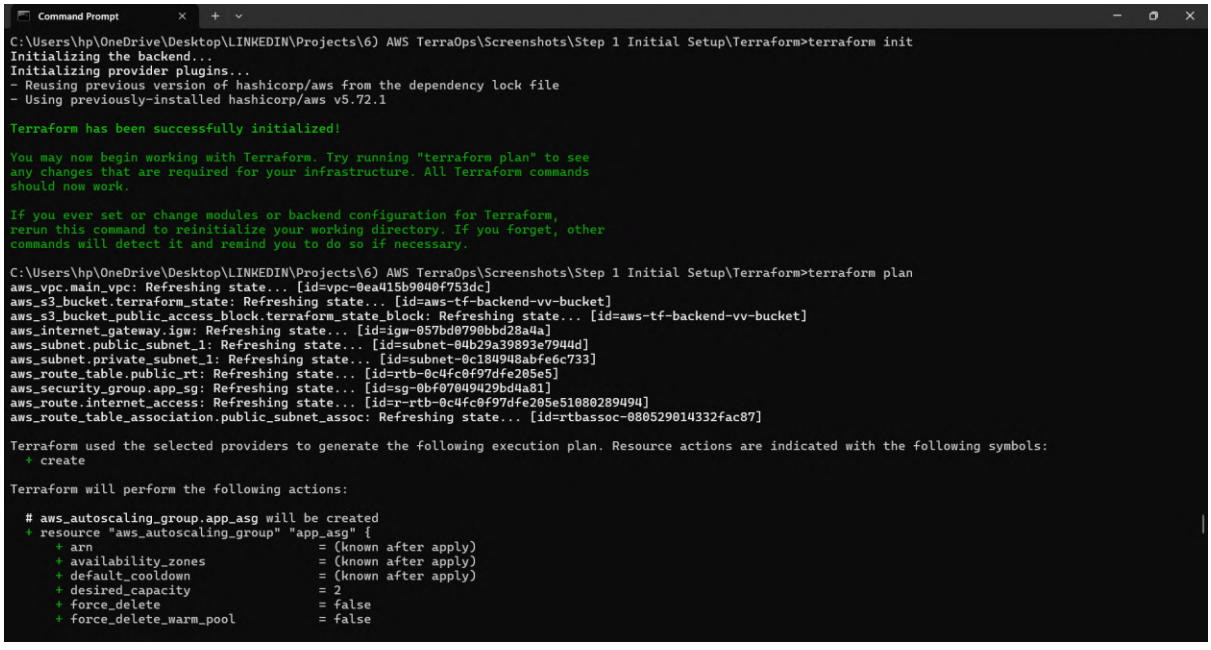
```
# S3 bucket for storing Terraform state

resource "aws_s3_bucket" "terraform_state" {
  bucket = "aws-tf-backend-vv-bucket"


  tags = {
    Name        = "TerraformStateBucket"
    Environment = "Dev"
  }
}

# Public access block for the S3 bucket
resource "aws_s3_bucket_public_access_block" "terraform_state_block" {
  bucket = aws_s3_bucket.terraform_state.id

  block_public_acls   = true
  block_public_policy = true
  restrict_public_buckets = true
  ignore_public_acls = true
}
```

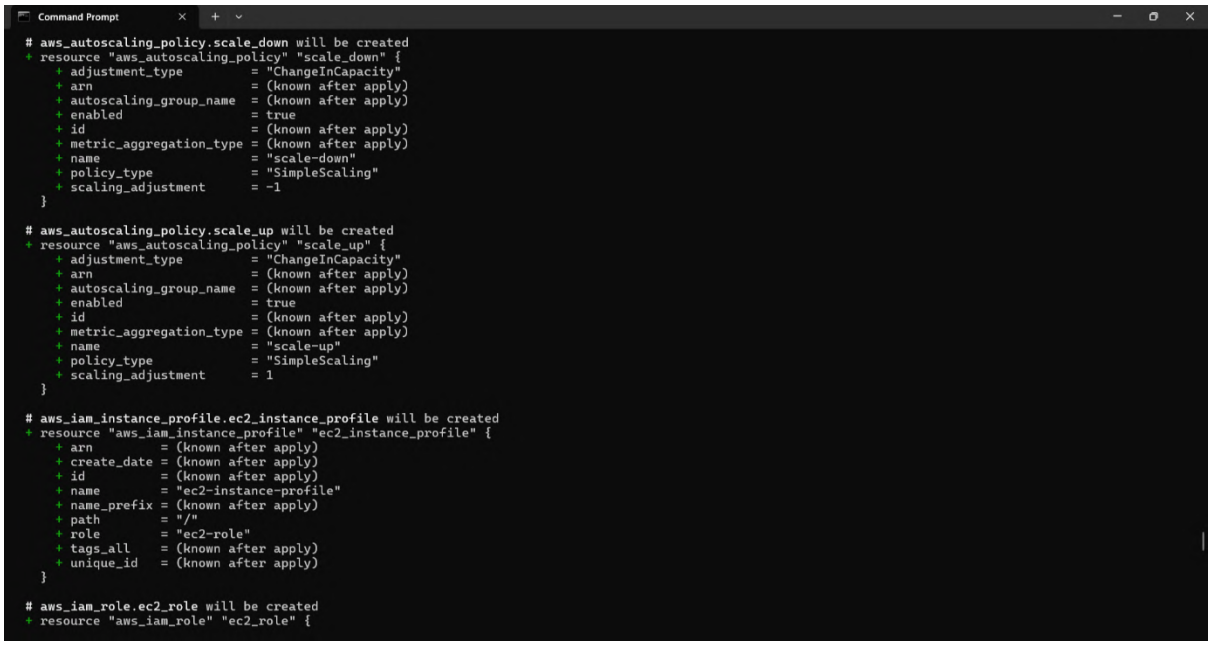


```
terraform {
  backend "s3" {
    bucket = "aws-tf-backend-vv-bucket"
    key    = "terraform/dev/terraform.tfstate"
    region = "us-east-1"
  }
}

provider "aws" {
  region = "us-east-1"
}
```

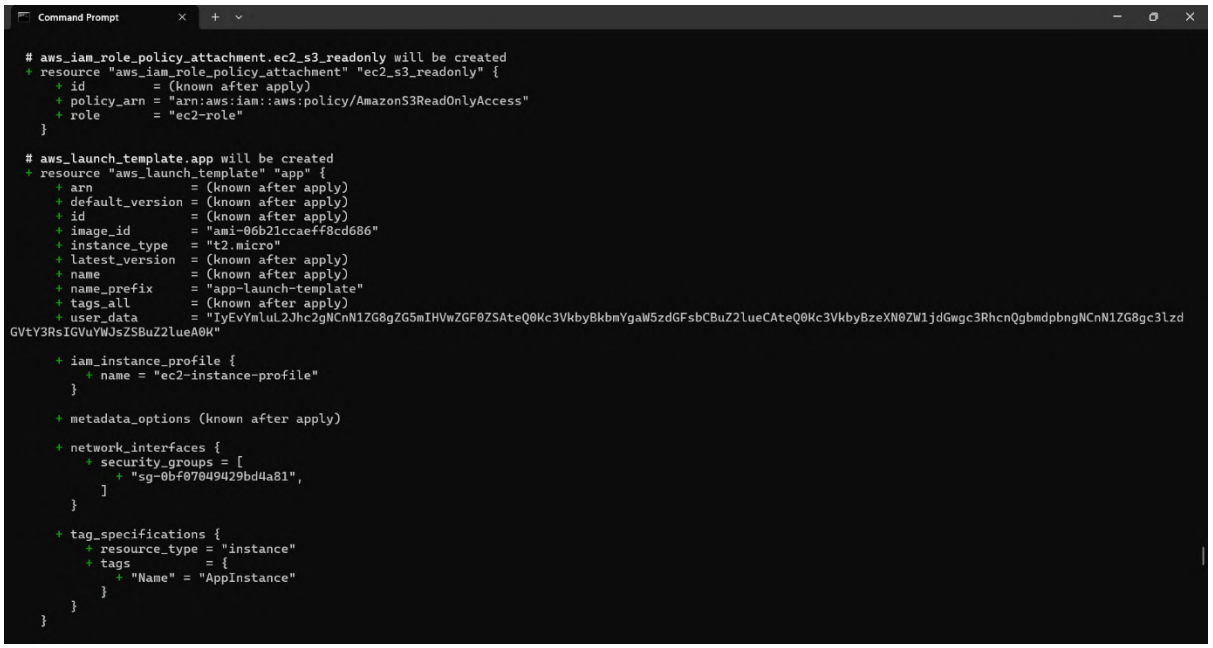

File Edit Selection View Go Run Terminal Help
Terraform
EXPLORER
TERRAFORM
.terraform
.terraform.lock.hcl
ec2_autoscaling.tf
iam_roles.tf
main.tf
network.tf
s3_bucket.tf
terraform.tfstate
terraform.tfstate.backup
OUTLINE
TIMELINE
s3_bucket.tf
s3_bucket.tf > resource "aws_s3_bucket_public_access_block" "backup_bucket_block"
14 resource "aws_s3_bucket_public_access_block" "terraform_state_block" {
18 block_public_policy = true
19 restrict_public_buckets = true
20 ignore_public_acls = true
21 }
28 # S3 bucket for backups
31 resource "aws_s3_bucket" "backup_bucket" {
32 bucket = "terra-backup-vv-project-bucket"
34 tags = {
35 Name = "BackupBucket"
36 Environment = "Dev"
37 }
38 }
40 # Public access block for the backup S3 bucket
41 resource "aws_s3_bucket_public_access_block" "backup_bucket_block" {
42 bucket = aws_s3_bucket.backup_bucket.id
44 block_public_acls = true
45 block_public_policy = true
46 restrict_public_buckets = true
47 ignore_public_acls = true
48 }
Ln 48, Col 2 Spaces: 4 UTF-8 CRLF {} Terraform CODEGPT


Command Prompt
C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>terraform destroy -target=aws_launch_template.app

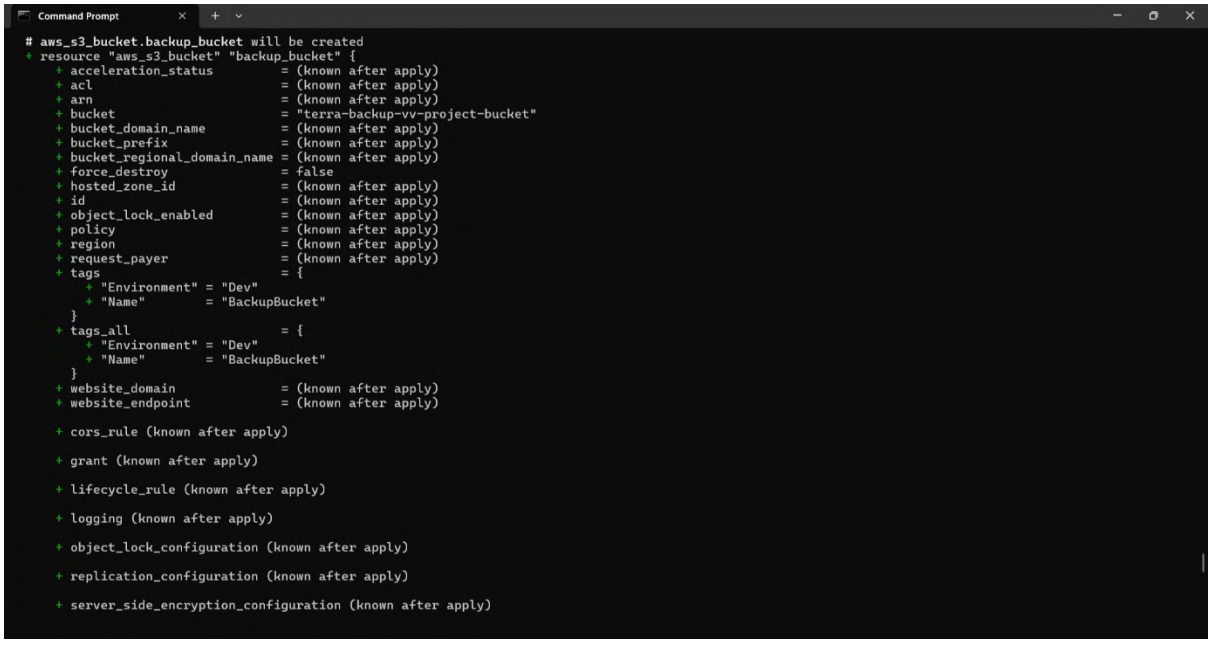

Command Prompt
C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.72.1
Terraform has been successfully initialized!
You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.
If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>terraform plan
aws_vpc.main_vpc: Refreshing state... [id=vpc-0ea415b9040f753dc]
aws_s3_bucket.terraform_state: Refreshing state... [id=aws-tf-backend-vv-bucket]
aws_s3_bucket_public_access_block.terraform_state_block: Refreshing state... [id=aws-tf-backend-vv-bucket]
aws_internet_gateway.igw: Refreshing state... [id=igw-057bd0790bbd28a4a]
aws_subnet.public_subnet_1: Refreshing state... [id=subnet-04b29a39893e7944d]
aws_subnet.private_subnet_1: Refreshing state... [id=subnet-0c184948abfe6c733]
aws_route_table.public_rt: Refreshing state... [id=rtb-0c4fc0f97dfe205e5]
aws_security_group.app_sg: Refreshing state... [id=sg-0bf07049429bd4a81]
aws_route.internet_access: Refreshing state... [id=r-rtb-0c4fc0f97dfe205e51080289494]
aws_route_table_association.public_subnet_assoc: Refreshing state... [id=rtbassoc-080529014332fac87]
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_autoscaling_group.app_asg will be created
+ resource "aws_autoscaling_group" "app_asg" {
+ arn = (known after apply)
+ availability_zones = (known after apply)
+ default_cooldown = (known after apply)
+ desired_capacity = 2
+ force_delete = false
+ force_delete_warm_pool = false



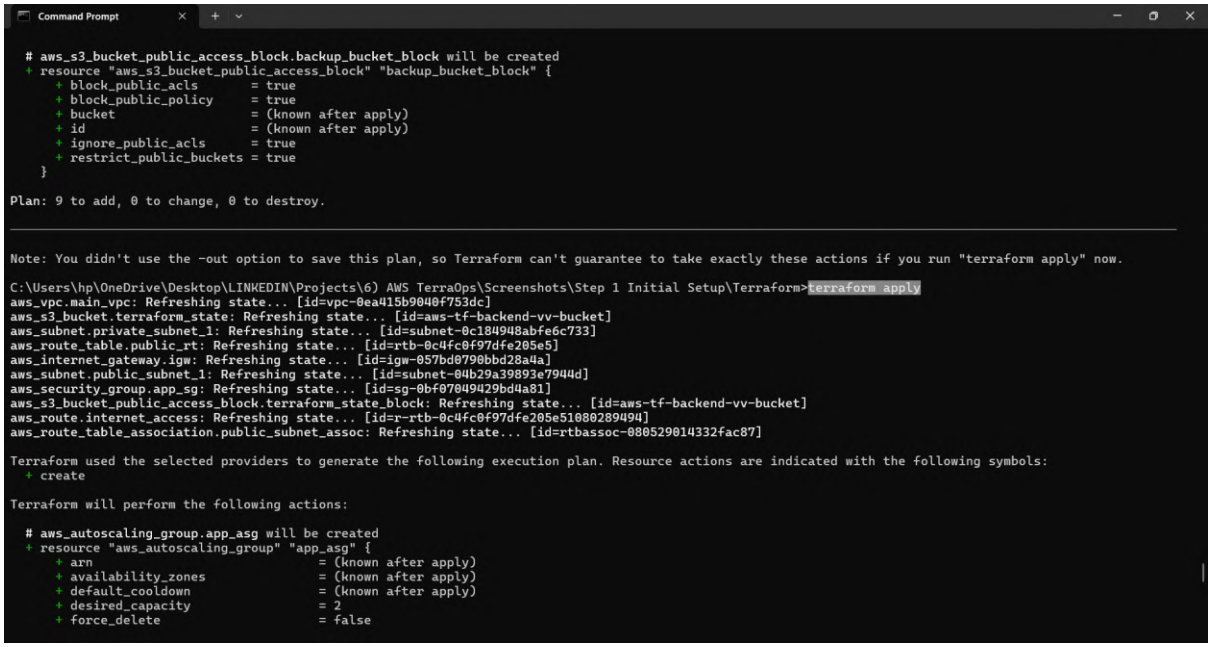
```
Command Prompt

  # aws_autoscaling_policy.scale_down will be created
  + resource "aws_autoscaling_policy" "scale_down" {
      + adjustment_type         = "ChangeInCapacity"
      + arn                     = (known after apply)
      + autoscaling_group_name  = (known after apply)
      + enabled                 = true
      + id                      = (known after apply)
      + metric_aggregation_type = (known after apply)
      + name                    = "scale-down"
      + policy_type             = "SimpleScaling"
      + scaling_adjustment      = -1
    }

  # aws_autoscaling_policy.scale_up will be created
  + resource "aws_autoscaling_policy" "scale_up" {
      + adjustment_type         = "ChangeInCapacity"
      + arn                     = (known after apply)
      + autoscaling_group_name  = (known after apply)
      + enabled                 = true
      + id                      = (known after apply)
      + metric_aggregation_type = (known after apply)
      + name                    = "scale-up"
      + policy_type             = "SimpleScaling"
      + scaling_adjustment      = 1
    }

  # aws_iam_instance_profile.ec2_instance_profile will be created
  + resource "aws_iam_instance_profile" "ec2_instance_profile" {
      + arn         = (known after apply)
      + create_date = (known after apply)
      + id          = (known after apply)
      + name        = "ec2-instance-profile"
      + name_prefix = (known after apply)
      + path        = "/"
      + role        = "ec2-role"
      + tags_all    = (known after apply)
      + unique_id   = (known after apply)
    }

  # aws_iam_role.ec2_role will be created
  + resource "aws_iam_role" "ec2_role" {
```

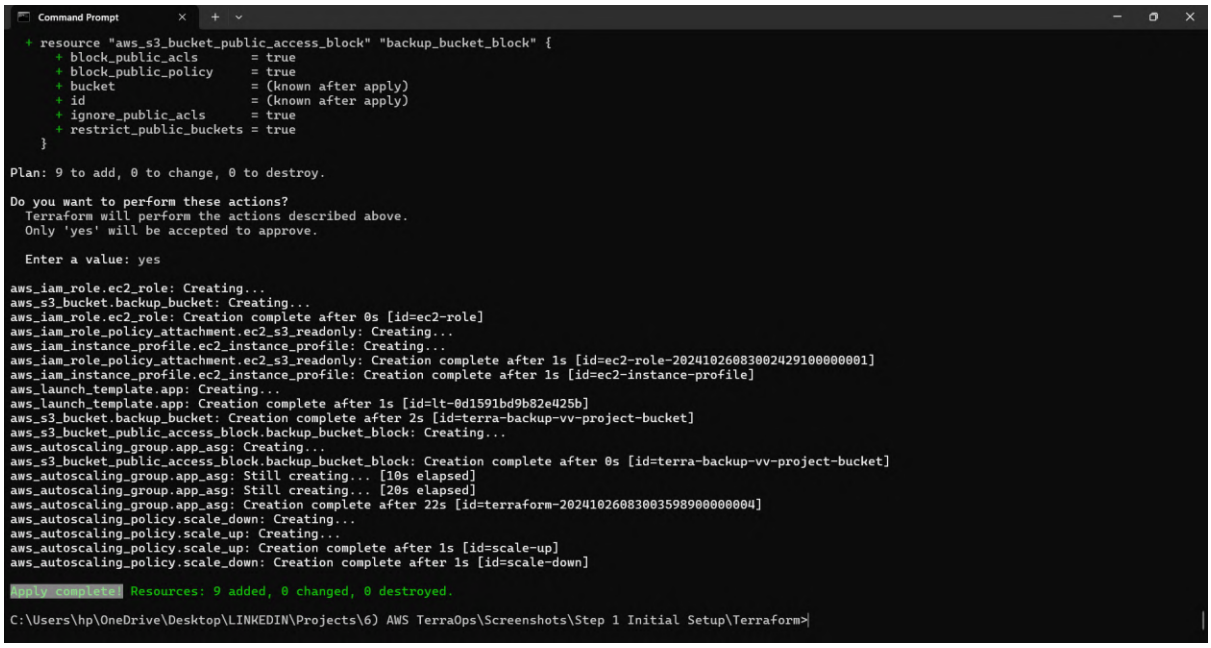


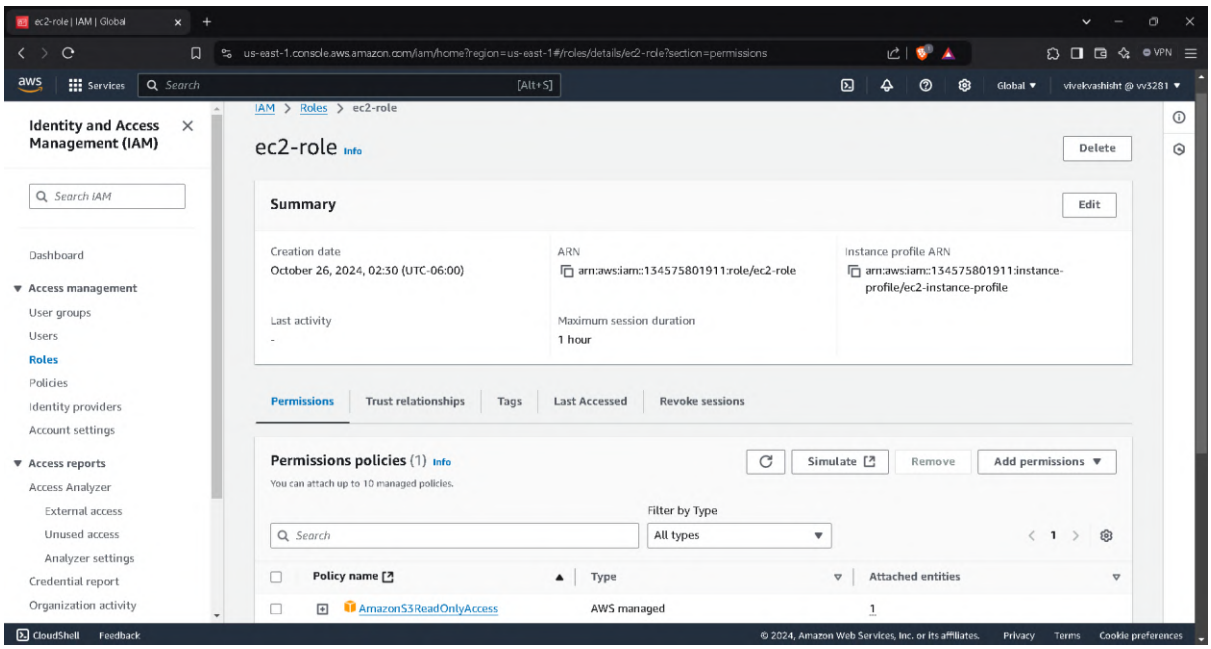
```
Command Prompt

  # aws_iam_role_policy_attachment.ec2_s3_readonly will be created
  + resource "aws_iam_role_policy_attachment" "ec2_s3_readonly" {
      + id         = (known after apply)
      + policy_arn = "arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess"
      + role       = "ec2-role"
    }

  # aws_launch_template.app will be created
  + resource "aws_launch_template" "app" {
      + arn             = (known after apply)
      + default_version = (known after apply)
      + id              = (known after apply)
      + image_id        = "ami-06b21ccaeff8cd686"
      + instance_type   = "t2.micro"
      + latest_version  = (known after apply)
      + name            = (known after apply)
      + name_prefix     = "app-launch-template"
      + tags_all        = (known after apply)
      + user_data       = "IyEvYmluL2Jhc2gNCnN1ZG8gZG5mIHVwZGF0ZSAteQ0Kc3VkbyBkbmYgaW5zdGFsbCBuZ2lueCAteQ0Kc3VkbyBzeXN0ZW1jdGwgc3RhcnQgbmdpbngNCnN1ZG8gc3lzdGVtY3RsIGVuYWJsZSBuZ2lueA0K"

      + iam_instance_profile {
          + name = "ec2-instance-profile"
        }

      + metadata_options (known after apply)

      + network_interfaces {
          + security_groups = [
              + "sg-0bf07049429bd4a81",
            ]
        }

      + tag_specifications {
          + resource_type = "instance"
          + tags          = {
              + "Name" = "AppInstance"
            }
        }
    }
```

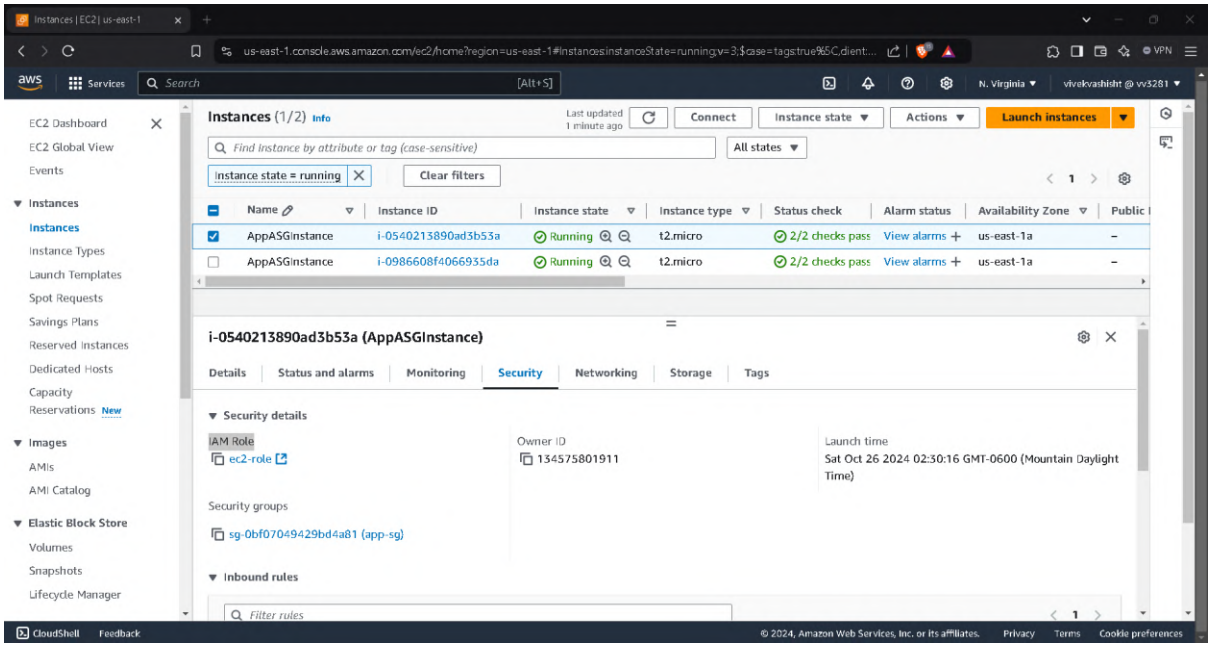


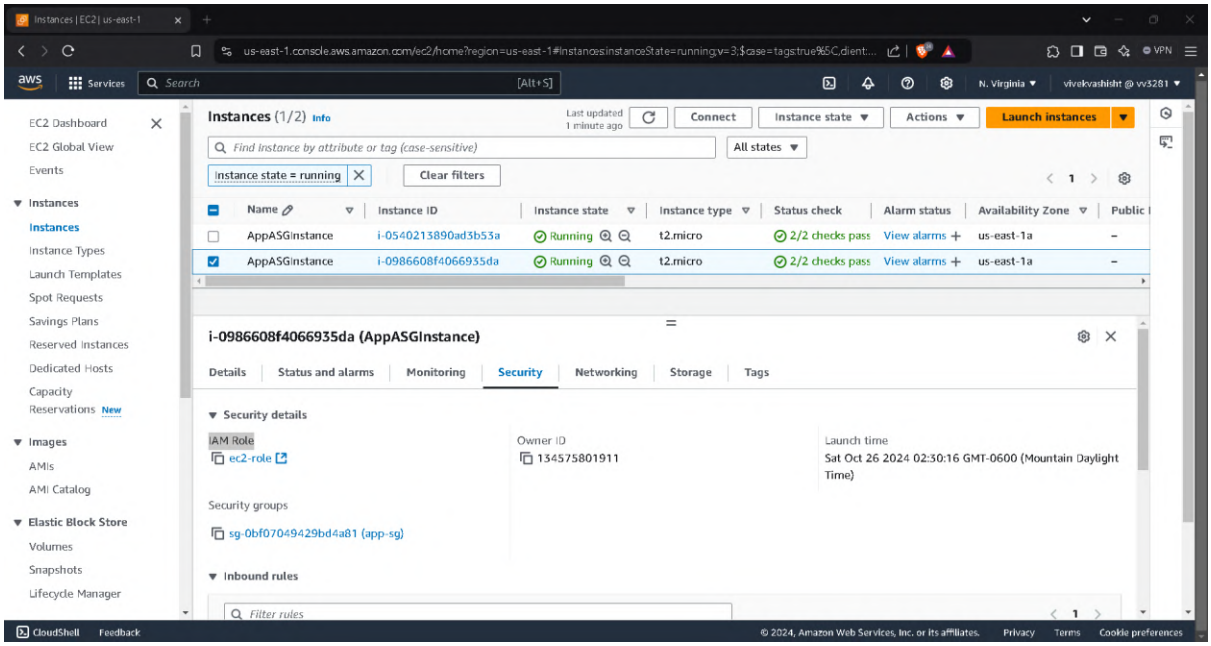
```
Command Prompt

  # aws_s3_bucket.backup_bucket will be created
  + resource "aws_s3_bucket" "backup_bucket" {
      + acceleration_status         = (known after apply)
      + acl                         = (known after apply)
      + arn                         = (known after apply)
      + bucket                      = "terra-backup-vv-project-bucket"
      + bucket_domain_name          = (known after apply)
      + bucket_prefix               = (known after apply)
      + bucket_regional_domain_name = (known after apply)
      + force_destroy               = false
      + hosted_zone_id              = (known after apply)
      + id                          = (known after apply)
      + object_lock_enabled         = (known after apply)
      + policy                      = (known after apply)
      + region                      = (known after apply)
      + request_payer               = (known after apply)
      + tags                        = {
          + "Environment" = "Dev"
          + "Name"        = "BackupBucket"
        }
      + tags_all                    = {
          + "Environment" = "Dev"
          + "Name"        = "BackupBucket"
        }
      + website_domain              = (known after apply)
      + website_endpoint            = (known after apply)

      + cors_rule (known after apply)

      + grant (known after apply)

      + lifecycle_rule (known after apply)

      + logging (known after apply)

      + object_lock_configuration (known after apply)

      + replication_configuration (known after apply)

      + server_side_encryption_configuration (known after apply)
```

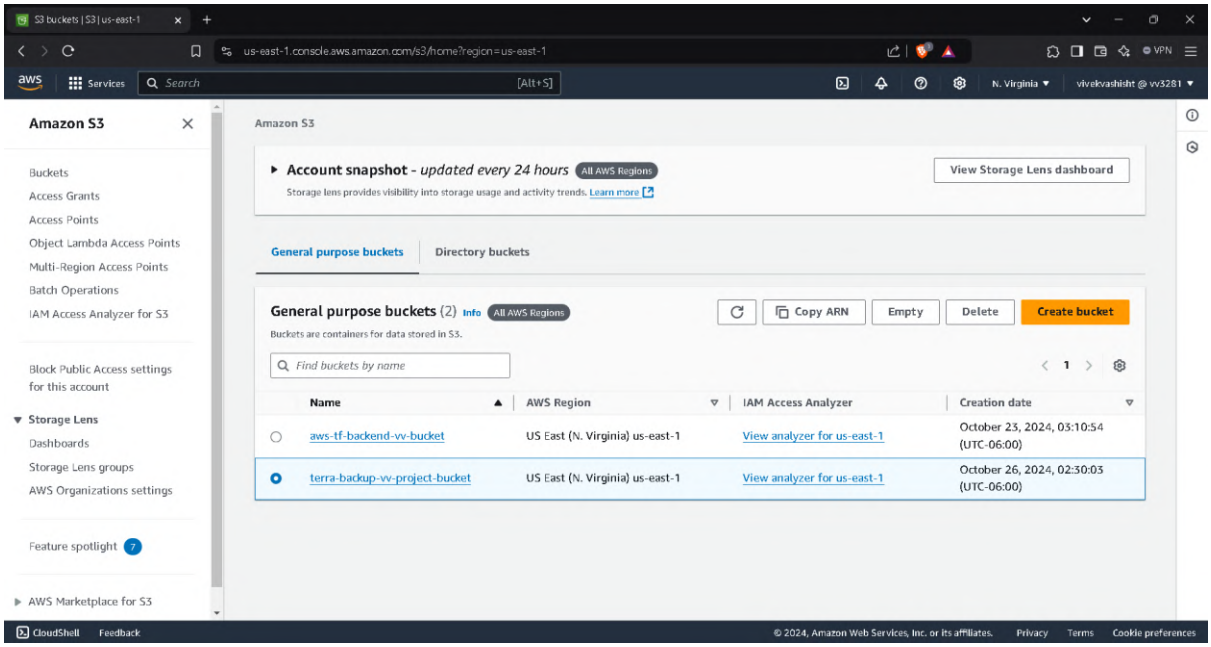

Command Prompt
# aws_s3_bucket_public_access_block.backup_bucket_block will be created
+ resource "aws_s3_bucket_public_access_block" "backup_bucket_block" {
+ block_public_acls = true
+ block_public_policy = true
+ bucket = (known after apply)
+ id = (known after apply)
+ ignore_public_acls = true
+ restrict_public_buckets = true
}
Plan: 9 to add, 0 to change, 0 to destroy.
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.
C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>terraform apply
aws_vpc.main_vpc: Refreshing state... [id=vpc-0ea415b9040f753dc]
aws_s3_bucket.terraform_state: Refreshing state... [id=aws-tf-backend-vv-bucket]
aws_subnet.private_subnet_1: Refreshing state... [id=subnet-0c184948abfe6c733]
aws_route_table.public_rt: Refreshing state... [id=rtb-0c4fc0f97dfe205e5]
aws_internet_gateway.igw: Refreshing state... [id=igw-057bd0790bbd28a4a]
aws_subnet.public_subnet_1: Refreshing state... [id=subnet-04b29a39893e7944d]
aws_security_group.app_sg: Refreshing state... [id=sg-0bf07049429bd4a81]
aws_s3_bucket_public_access_block.terraform_state_block: Refreshing state... [id=aws-tf-backend-vv-bucket]
aws_route.internet_access: Refreshing state... [id=r-rtb-0c4fc0f97dfe205e510802894094]
aws_route_table_association.public_subnet_assoc: Refreshing state... [id=rtbassoc-080529014332fac87]
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_autoscaling_group.app_asg will be created
+ resource "aws_autoscaling_group" "app_asg" {
+ arn = (known after apply)
+ availability_zones = (known after apply)
+ default_cooldown = (known after apply)
+ desired_capacity = 2
+ force_delete = false

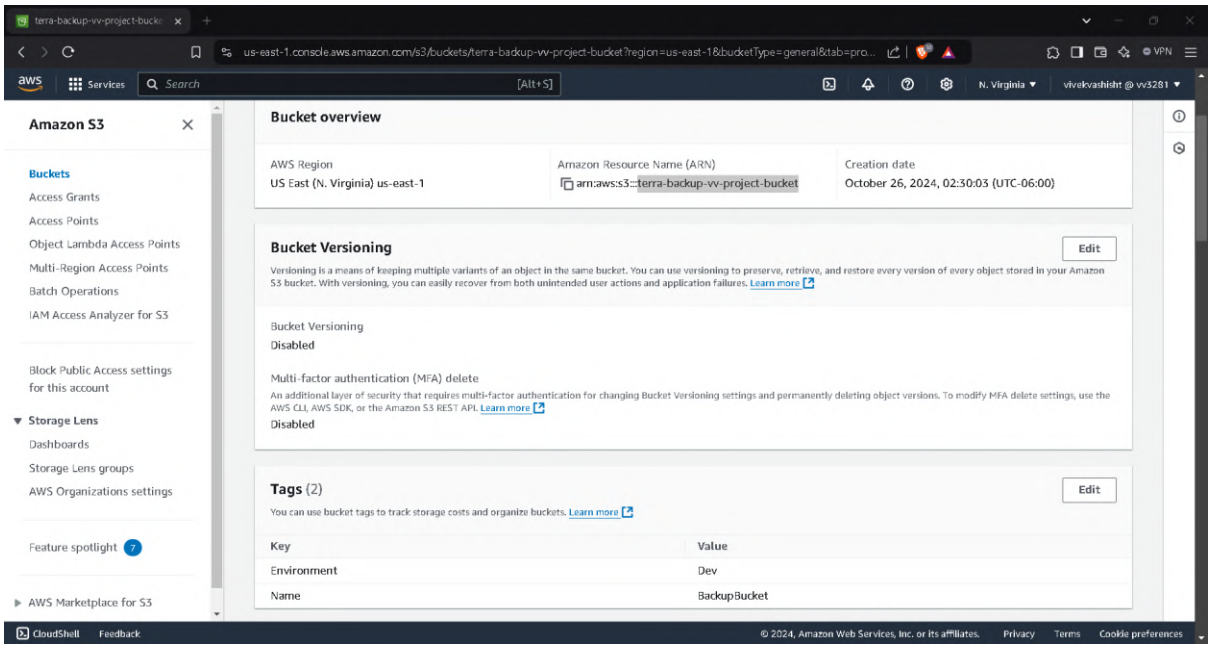

Command Prompt
+ resource "aws_s3_bucket_public_access_block" "backup_bucket_block" {
+ block_public_acls = true
+ block_public_policy = true
+ bucket = (known after apply)
+ id = (known after apply)
+ ignore_public_acls = true
+ restrict_public_buckets = true
}
Plan: 9 to add, 0 to change, 0 to destroy.
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_iam_role.ec2_role: Creating...
aws_s3_bucket.backup_bucket: Creating...
aws_iam_role.ec2_role: Creation complete after 0s [id=ec2-role]
aws_iam_role_policy_attachment.ec2_s3_readonly: Creating...
aws_iam_instance_profile.ec2_instance_profile: Creating...
aws_iam_role_policy_attachment.ec2_s3_readonly: Creation complete after 1s [id=ec2-role-20241026083002429100000001]
aws_iam_instance_profile.ec2_instance_profile: Creation complete after 1s [id=ec2-instance-profile]
aws_launch_template.app: Creating...
aws_launch_template.app: Creation complete after 1s [id=lt-0d1591bd9b82e425b]
aws_s3_bucket.backup_bucket: Creation complete after 2s [id=terra-backup-vv-project-bucket]
aws_s3_bucket_public_access_block.backup_bucket_block: Creating...
aws_autoscaling_group.app_asg: Creating...
aws_s3_bucket_public_access_block.backup_bucket_block: Creation complete after 0s [id=terra-backup-vv-project-bucket]
aws_autoscaling_group.app_asg: Still creating... [10s elapsed]
aws_autoscaling_group.app_asg: Still creating... [20s elapsed]
aws_autoscaling_group.app_asg: Creation complete after 22s [id=terraform-20241026083003598900000004]
aws_autoscaling_policy.scale_down: Creating...
aws_autoscaling_policy.scale_up: Creating...
aws_autoscaling_policy.scale_up: Creation complete after 1s [id=scale-up]
aws_autoscaling_policy.scale_down: Creation complete after 1s [id=scale-down]
Apply complete! Resources: 9 added, 0 changed, 0 destroyed.
C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>


ec2-role | IAM | Global
us-east-1.console.aws.amazon.com/iam/home?region=us-east-1#/roles/details/ec2-role?section=permissions
Identity and Access Management (IAM)
Search IAM
Dashboard
Access management
User groups
Users
Roles
Policies
Identity providers
Account settings
Access reports
Access Analyzer
External access
Unused access
Analyzer settings
Credential report
Organization activity
IAM > Roles > ec2-role
ec2-role Info
Delete
Summary
Edit
Creation date
October 26, 2024, 02:30 (UTC-06:00)
ARN
arn:aws:iam::134575801911:role/ec2-role
Instance profile ARN
arn:aws:iam::134575801911:instance-profile/ec2-instance-profile
Last activity
-
Maximum session duration
1 hour
Permissions
Trust relationships
Tags
Last Accessed
Revoke sessions
Permissions policies (1) Info
You can attach up to 10 managed policies.
Simulate
Remove
Add permissions
Filter by Type
Search
All types
Policy name
Type
Attached entities
AmazonS3ReadOnlyAccess
AWS managed
1
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Instances | EC2 | us-east-1
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#Instances:instanceState=running;v=3;$case=tags:true%5C,client:...
Services
Search
[Alt+S]
N. Virginia
vivekvashisht @ vv3281
EC2 Dashboard
EC2 Global View
Events
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations New
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Instances (1/2) Info
Last updated 1 minute ago
Connect
Instance state
Actions
Launch instances
Find instance by attribute or tag (case-sensitive)
All states
instance state = running
Clear filters
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
Availability Zone
Public
AppASGInstance
i-0540213890ad3b53a
Running
t2.micro
2/2 checks pass
View alarms
us-east-1a
AppASGInstance
i-0986608f4066935da
Running
t2.micro
2/2 checks pass
View alarms
us-east-1a
i-0540213890ad3b53a (AppASGInstance)
Details
Status and alarms
Monitoring
Security
Networking
Storage
Tags
Security details
IAM Role
ec2-role
Owner ID
134575801911
Launch time
Sat Oct 26 2024 02:30:16 GMT-0600 (Mountain Daylight Time)
Security groups
sg-0bf07049429bd4a81 (app-sg)
Inbound rules
Filter rules
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Instances | EC2 | us-east-1
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#Instances:instanceState=running;v=3;$case=tags:true%5C,client:...
Services
Search
[Alt+S]
N. Virginia
vivekvashisht @ vv3281
EC2 Dashboard
EC2 Global View
Events
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations New
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Instances (1/2) Info
Last updated 1 minute ago
Connect
Instance state
Actions
Launch instances
Find instance by attribute or tag (case-sensitive)
All states
instance state = running
Clear filters
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
Availability Zone
Public
AppASGInstance
i-0540213890ad3b53a
Running
t2.micro
2/2 checks pass
View alarms
us-east-1a
AppASGInstance
i-0986608f4066935da
Running
t2.micro
2/2 checks pass
View alarms
us-east-1a
i-0986608f4066935da (AppASGInstance)
Details
Status and alarms
Monitoring
Security
Networking
Storage
Tags
Security details
IAM Role
ec2-role
Owner ID
134575801911
Launch time
Sat Oct 26 2024 02:30:16 GMT-0600 (Mountain Daylight Time)
Security groups
sg-0bf07049429bd4a81 (app-sg)
Inbound rules
Filter rules
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


S3 buckets | S3 | us-east-1
us-east-1.console.aws.amazon.com/s3/home?region=us-east-1
Services
Search
[Alt+S]
N. Virginia
vivekvashisht @ vv3281
Amazon S3
Buckets
Access Grants
Access Points
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
IAM Access Analyzer for S3
Block Public Access settings for this account
Storage Lens
Dashboards
Storage Lens groups
AWS Organizations settings
Feature spotlight 7
AWS Marketplace for S3
Amazon S3
Account snapshot - updated every 24 hours
All AWS Regions
Storage lens provides visibility into storage usage and activity trends. Learn more
View Storage Lens dashboard
General purpose buckets
Directory buckets
General purpose buckets (2) Info
All AWS Regions
Buckets are containers for data stored in S3.
Copy ARN
Empty
Delete
Create bucket
Find buckets by name
Name
AWS Region
IAM Access Analyzer
Creation date
aws-tf-backend-vv-bucket
US East (N. Virginia) us-east-1
View analyzer for us-east-1
October 23, 2024, 03:10:54 (UTC-06:00)
terra-backup-vv-project-bucket
US East (N. Virginia) us-east-1
View analyzer for us-east-1
October 26, 2024, 02:30:03 (UTC-06:00)
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences



Multi-Region Deployment and Failover

a) Overview

In this phase, I set up a multi-region deployment with a failover mechanism. By establishing resources in two AWS regions, `us-east-1` (primary) and `us-west-2` (secondary), I ensured the application would remain available in case of failure in the primary region. To support this, I configured an Application Load Balancer (ALB) in each region and implemented route tables, VPCs, subnets, and Internet Gateways.

b) Configuring Providers for Multi-Region Deployment

I updated the `main.tf` file to define an additional provider alias for `us-west-2`. This allows us to deploy resources in two regions within the same Terraform configuration.

```
# Adding provider for us-west-2
provider "aws" {
  alias  = "west"
  region = "us-west-2"
}
```

The complete `main.tf` now includes both `us-east-1` and `us-west-2` configurations:

```
terraform {
  backend "s3" {
    bucket = "aws-tf-backend-vv-bucket"
    key    = "terraform/dev/terraform.tfstate"
    region = "us-east-1"
  }
```

```
}

provider "aws" {
  region = "us-east-1"
}

provider "aws" {
  alias  = "west"
  region = "us-west-2"
}
```

c) Network Configuration in us-east-1

In `network.tf`, I added a second public subnet for the primary region (`us-east-1`). Two public subnets are required to set up the Application Load Balancer for high availability.

```
# Additional Public Subnet in us-east-1b for ALB
resource "aws_subnet" "public_subnet_2" {
  vpc_id                  = aws_vpc.main_vpc.id
  cidr_block              = "10.0.4.0/24"
  availability_zone       = "us-east-1b"
  map_public_ip_on_launch = true

  tags = {
    Name = "PublicSubnet2"
  }
}
```

I also modified the CIDR block of the private subnet in `us-east-1a` to `10.0.3.0/24` to avoid any overlapping issues.

d) Configuring Network Resources in us-west-2

Following the setup for `us-east-1`, I configured similar resources for the `us-west-2` region. This includes creating a VPC, public and private subnets, Internet Gateway, and route tables. These resources allow the application to function independently within `us-west-2`, ensuring that traffic can be routed to this region if the primary region fails.

Here's the final `network.tf` configuration for multi-region deployment:

```
# VPC and Subnets for us-east-1
resource "aws_vpc" "main_vpc" {
  cidr_block = "10.0.0.0/16"

  tags = {
    Name = "MainVPC"
  }
}

resource "aws_subnet" "public_subnet_1" {
  vpc_id                  = aws_vpc.main_vpc.id
  cidr_block              = "10.0.1.0/24"
  availability_zone       = "us-east-1a"
  map_public_ip_on_launch = true

  tags = {
```

```
    Name = "PublicSubnet1"
  }
}

resource "aws_subnet" "public_subnet_2" {
  vpc_id                  = aws_vpc.main_vpc.id
  cidr_block              = "10.0.4.0/24"
  availability_zone       = "us-east-1b"
  map_public_ip_on_launch = true

  tags = {
    Name = "PublicSubnet2"
  }
}

resource "aws_subnet" "private_subnet_1" {
  vpc_id                  = aws_vpc.main_vpc.id
  cidr_block              = "10.0.3.0/24"
  availability_zone       = "us-east-1a"
  map_public_ip_on_launch = false

  tags = {
    Name = "PrivateSubnet1"
  }
}

# VPC and Subnets for us-west-2
resource "aws_vpc" "west_vpc" {
  provider = aws.west
  cidr_block = "10.1.0.0/16"

  tags = {
    Name = "WestVPC"
  }
}

resource "aws_subnet" "west_public_subnet_1" {
  provider = aws.west
  vpc_id = aws_vpc.west_vpc.id
  cidr_block = "10.1.1.0/24"
  availability_zone = "us-west-2a"
  map_public_ip_on_launch = true

  tags = {
    Name = "WestPublicSubnet1"
  }
}

resource "aws_subnet" "west_public_subnet_2" {
  provider = aws.west
  vpc_id = aws_vpc.west_vpc.id
  cidr_block = "10.1.2.0/24"
  availability_zone = "us-west-2b"
  map_public_ip_on_launch = true

  tags = {
    Name = "WestPublicSubnet2"
  }
}

resource "aws_subnet" "west_private_subnet_1" {
```

```
  provider = aws.west
  vpc_id = aws_vpc.west_vpc.id
  cidr_block = "10.1.3.0/24"
  availability_zone = "us-west-2a"

  tags = {
    Name = "WestPrivateSubnet1"
  }
}
```

e) Internet Gateways and Route Tables

I defined Internet Gateways for both regions and associated each with its corresponding VPC. For each public subnet in both regions, I configured route tables to allow outbound internet access through the Internet Gateway.

```
# Internet Gateway for us-east-1
resource "aws_internet_gateway" "igw" {
  vpc_id = aws_vpc.main_vpc.id
  tags = {
    Name = "MainIGW"
  }
}

# Public Route Table and Route for us-east-1
resource "aws_route_table" "public_rt" {
  vpc_id = aws_vpc.main_vpc.id
  tags = {
    Name = "PublicRouteTable"
  }
}

resource "aws_route" "internet_access" {
  route_table_id = aws_route_table.public_rt.id
  destination_cidr_block = "0.0.0.0/0"
  gateway_id = aws_internet_gateway.igw.id
}

resource "aws_route_table_association" "public_subnet_assoc_1" {
  subnet_id = aws_subnet.public_subnet_1.id
  route_table_id = aws_route_table.public_rt.id
}

resource "aws_route_table_association" "public_subnet_assoc_2" {
  subnet_id = aws_subnet.public_subnet_2.id
  route_table_id = aws_route_table.public_rt.id
}

# Internet Gateway and Route Table for us-west-2
resource "aws_internet_gateway" "west_igw" {
  provider = aws.west
  vpc_id = aws_vpc.west_vpc.id
  tags = {
    Name = "WestIGW"
  }
}

resource "aws_route_table" "west_public_rt" {
  provider = aws.west
```

```
  vpc_id = aws_vpc.west_vpc.id
  tags = {
    Name = "WestPublicRouteTable"
  }
}

resource "aws_route" "west_internet_access" {
  provider = aws.west
  route_table_id = aws_route_table.west_public_rt.id
  destination_cidr_block = "0.0.0.0/0"
  gateway_id = aws_internet_gateway.west_igw.id
}

resource "aws_route_table_association" "west_public_subnet_assoc_1" {
  provider = aws.west
  subnet_id = aws_subnet.west_public_subnet_1.id
  route_table_id = aws_route_table.west_public_rt.id
}

resource "aws_route_table_association" "west_public_subnet_assoc_2" {
  provider = aws.west
  subnet_id = aws_subnet.west_public_subnet_2.id
  route_table_id = aws_route_table.west_public_rt.id
}
```

f) Application Load Balancer (ALB) Setup

I created an ALB in each region to distribute incoming traffic across instances in the Auto Scaling Group. Each ALB is associated with two public subnets in its region, enabling it to handle high availability across different Availability Zones.

The `alb.tf` configuration for the primary ALB in `us-east-1` and the secondary ALB in `us-west-2` is as follows:

```
# Application Load Balancer (ALB) (us-east-1)
resource "aws_lb" "primary_alb" {
  name               = "primary-alb"
  internal           = false
  load_balancer_type = "application"
  security_groups    = [aws_security_group.app_sg.id] # Attach the security
group
  subnets            = [aws_subnet.public_subnet_1.id,
aws_subnet.public_subnet_2.id]

  enable_deletion_protection = false
  idle_timeout               = 400

  tags = {
    Name = "PrimaryALB"
  }
}

# ALB Target Group for us-east-1
resource "aws_lb_target_group" "primary" {
  name     = "primary-target-group"
  port     = 80
  protocol = "HTTP"
  vpc_id   = aws_vpc.main_vpc.id
```

```
  health_check {
    path                = "/"
    interval            = 30
    timeout             = 5
    healthy_threshold   = 2
    unhealthy_threshold = 2
  }

  tags = {
    Name = "PrimaryTargetGroup"
  }
}

# ALB Listener for HTTP traffic
resource "aws_lb_listener" "http" {
  load_balancer_arn = aws_lb.primary_alb.arn
  port              = 80
  protocol          = "HTTP"

  default_action {
    type             = "forward"
    target_group_arn = aws_lb_target_group.primary.arn
  }

  tags = {
    Name = "PrimaryALBListener"
  }
}

# Application Load Balancer (ALB) for us-west-2
resource "aws_lb" "secondary_alb" {
  provider           = aws.west
  name               = "secondary-alb"
  internal           = false
  load_balancer_type = "application"
  security_groups    = [aws_security_group.west_app_sg.id]
  subnets            = [aws_subnet.west_public_subnet_1.id,
aws_subnet.west_public_subnet_2.id]

  enable_deletion_protection = false
  idle_timeout               = 400

  tags = {
    Name = "SecondaryALB"
  }
}

# ALB Target Group for us-west-2
resource "aws_lb_target_group" "secondary" {
  provider = aws.west
  name     = "secondary-target-group"
  port     = 80
  protocol = "HTTP"
  vpc_id   = aws_vpc.west_vpc.id

  health_check {
    path                = "/"
    interval            = 30
    timeout             = 5
    healthy_threshold   = 2
    unhealthy_threshold = 2
```

```
  }

  tags = {
    Name = "SecondaryTargetGroup"
  }
}

# ALB Listener for HTTP traffic in us-west-2
resource "aws_lb_listener" "secondary_http" {
  provider          = aws.west
  load_balancer_arn = aws_lb.secondary_alb.arn
  port              = 80
  protocol          = "HTTP"

  default_action {
    type             = "forward"
    target_group_arn = aws_lb_target_group.secondary.arn
  }

  tags = {
    Name = "SecondaryALBListener"
  }
}
```

g) Auto Scaling Group (ASG) Update for us-east-1 Region

After saving the `alb.tf` file, I proceeded to update the `Auto Scaling Group (ASG)` configuration for the `us-east-1` region in the `ec2_autoscaling.tf` file. Here, I added `public_subnet_2` to the `vpc_zone_identifier` to ensure that the ASG is configured to deploy instances across multiple subnets for high availability. Additionally, I included the `target_group_arns` configuration, attaching the ASG to the `primary` ALB Target Group, and set `health_check_grace_period` and `health_check_type` to use ELB-based health checks.

```
# Auto Scaling Group (ASG) for us-east-1
resource "aws_autoscaling_group" "app_asg" {
  desired_capacity     = 2
  min_size             = 1
  max_size             = 5
  vpc_zone_identifier  = [aws_subnet.public_subnet_1.id,
aws_subnet.public_subnet_2.id] # Attach to ALB Target Group
  target_group_arns    = [aws_lb_target_group.primary.arn]

  health_check_grace_period = 300  # Grace period
  health_check_type         = "ELB"  # Change to ELB-based health check

  launch_template {
    id      = aws_launch_template.app.id
    version = "$Latest"
  }

  tag {
    key                 = "Name"
    value               = "AppASGInstance"
    propagate_at_launch = true
  }
```

```
  lifecycle {
    create_before_destroy = true
  }
}
```

With these changes, the ASG in `us-east-1` is now configured to manage instances across both public subnets (`public_subnet_1` and `public_subnet_2`) and integrated with the Application Load Balancer (ALB) for efficient traffic distribution.

h) Multi-Region Resource Configuration in us-west-2

Following the `us-east-1` configuration, I replicated the necessary configurations for the `us-west-2` region, defining the Security Group, Launch Template, Auto Scaling Group (ASG), and Auto Scaling Policies. This ensures the project's high availability and resilience across both `us-east-1` and `us-west-2` regions.

Security Group for `us-west-2`

The security group for `us-west-2`, named `west-app-sg`, was created to manage access to instances in this region. This security group includes rules to allow SSH access from a specified IP, as well as HTTP access for web traffic.

```
# Security Group
resource "aws_security_group" "west_app_sg" {
  provider    = aws.west
  name        = "west-app-sg"
  description = "Allow HTTP and SSH traffic in us-west-2"
  vpc_id      = aws_vpc.west_vpc.id

  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["142.59.28.49/32"]  # Restrict SSH to your IP
  }

  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["18.237.140.160/29"]  # EC2 Instance Connect IP range
for us-west-2
  }

  ingress {
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]  # Allow HTTP traffic from anywhere
  }

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
```

```
    cidr_blocks = ["0.0.0.0/0"]  # Allow all outbound traffic
  }

  tags = {
    Name = "WestAppSecurityGroup"
  }
}
```

Launch Template for EC2 Instances in `us-west-2`

The launch template in `us-west-2` enables the ASG to provision instances with a pre-defined configuration. This template includes user data to install and start NGINX upon instance initialization, ensuring each instance is ready to serve traffic.

```
# Launch Template for EC2 Instances in us-west-2
resource "aws_launch_template" "secondary_app" {
  provider     = aws.west
  name_prefix  = "secondary-app-launch-template"
  image_id     = "ami-07c5ecd84908c59d5"  # Amazon Linux AMI
  instance_type = "t2.micro"

  # Attachment of the IAM role to EC2 instances in us-west-2
  iam_instance_profile {
    name = aws_iam_instance_profile.ec2_instance_profile.name
  }

  network_interfaces {
    security_groups = [aws_security_group.west_app_sg.id]
  }

  user_data = base64encode(<<-EOF
    #!/bin/bash
    sudo dnf update -y
    sudo dnf install nginx -y
    sudo systemctl start nginx
    sudo systemctl enable nginx
  EOF
  )

  lifecycle {
    create_before_destroy = true
  }

  tag_specifications {
    resource_type = "instance"
    tags = {
      Name = "SecondaryAppInstance"
    }
  }
}
```

Auto Scaling Group (ASG) for `us-west-2`

For high availability, I created an ASG in `us-west-2` to manage instance deployment across public subnets and attach to the secondary ALB target group.

```
# Auto Scaling Group (ASG) for us-west-2
resource "aws_autoscaling_group" "secondary_asg" {
```

```
  provider           = aws.west
  desired_capacity   = 2
  min_size           = 1
  max_size           = 5
  vpc_zone_identifier = [aws_subnet.west_public_subnet_1.id,
aws_subnet.west_public_subnet_2.id]  # Attach to ALB Target Group
  target_group_arns  = [aws_lb_target_group.secondary.arn]

  health_check_grace_period = 300  # Grace period
  health_check_type         = "ELB"  # Use ELB-based health check for ALB

  launch_template {
    id      = aws_launch_template.secondary_app.id
    version = "$latest"
  }

  tag {
    key                 = "Name"
    value               = "SecondaryAppASGInstance"
    propagate_at_launch = true
  }

  lifecycle {
    create_before_destroy = true
  }
}
```

Auto Scaling Policies for `us-west-2`

To enable dynamic scaling based on demand, I created scale-up and scale-down policies for `us-west-2`. These policies adjust the ASG capacity by one instance based on the load.

```
# Auto Scaling Policies

# Scaling Up Policy for us-west-2
resource "aws_autoscaling_policy" "west_scale_up" {
  provider               = aws.west
  name                   = "west-scale-up"
  scaling_adjustment     = 1  # Adds 1 instance when triggered
  adjustment_type        = "ChangeInCapacity"
  autoscaling_group_name  = aws_autoscaling_group.secondary_asg.name
}

# Scaling Down Policy for us-west-2
resource "aws_autoscaling_policy" "west_scale_down" {
  provider               = aws.west
  name                   = "west-scale-down"
  scaling_adjustment     = -1  # Removes 1 instance when triggered
  adjustment_type        = "ChangeInCapacity"
  autoscaling_group_name  = aws_autoscaling_group.secondary_asg.name
}
```

i) IAM Role Configuration and Permissions Update

After saving the `ec2_autoscaling.tf` file, I updated the `iam_roles.tf` file to ensure that EC2 instances had the necessary permissions to interact with AWS services required for this multi-region deployment.

```
# IAM Role for EC2 to assume
resource "aws_iam_role" "ec2_role" {
  name = "ec2-role"

  assume_role_policy = <<EOF
  {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Principal": {
          "Service": "ec2.amazonaws.com"
        },
        "Action": "sts:AssumeRole"
      }
    ]
  }
  EOF
}

# IAM instance profile for EC2
resource "aws_iam_instance_profile" "ec2_instance_profile" {
  name = "ec2-instance-profile"
  role = aws_iam_role.ec2_role.name
}

# Attach required permissions
resource "aws_iam_role_policy_attachment" "ec2_full_access" {
  role       = aws_iam_role.ec2_role.name
  policy_arn = "arn:aws:iam::aws:policy/AmazonEC2FullAccess"
}

resource "aws_iam_role_policy_attachment" "autoscaling_full_access" {
  role       = aws_iam_role.ec2_role.name
  policy_arn = "arn:aws:iam::aws:policy/AutoScalingFullAccess"
}

resource "aws_iam_role_policy_attachment" "elb_full_access" {
  role       = aws_iam_role.ec2_role.name
  policy_arn = "arn:aws:iam::aws:policy/ElasticLoadBalancingFullAccess"
}

# Attachment of S3 read-only policy to the role
resource "aws_iam_role_policy_attachment" "ec2_s3_readonly" {
  role       = aws_iam_role.ec2_role.name
  policy_arn = "arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess"
}
```

j) Route 53 Configuration for Failover

To implement failover between the two regions, I created `route53.tf`, which defines the Route 53 hosted zone and configures failover routing between the primary (us-east-1) and secondary (us-west-2) regions. I set up health checks for each ALB to monitor availability and automatically switch traffic to the secondary region if needed.

```
# Route 53 Hosted Zone
resource "aws_route53_zone" "main" {
  name = "myinfra.local"  # A fake domain for testing purposes
}
```

```
# Primary Route 53 Record (us-east-1)
resource "aws_route53_record" "primary" {
  zone_id = aws_route53_zone.main.zone_id
  name    = "app.myinfra.local"
  type    = "A"

  alias {
    name                   = aws_lb.primary_alb.dns_name
    zone_id                = aws_lb.primary_alb.zone_id
    evaluate_target_health = true
  }

  set_identifier           = "Primary"
  failover_routing_policy {
    type = "PRIMARY"
  }
}

# Secondary Route 53 Record (us-west-2)
resource "aws_route53_record" "secondary" {
  zone_id = aws_route53_zone.main.zone_id
  name    = "app.myinfra.local"
  type    = "A"

  alias {
    name                   = aws_lb.secondary_alb.dns_name
    zone_id                = aws_lb.secondary_alb.zone_id
    evaluate_target_health = true
  }

  set_identifier           = "Secondary"
  failover_routing_policy {
    type = "SECONDARY"
  }
}

# Route 53 Health Checks

# Primary Health Check (us-east-1)
resource "aws_route53_health_check" "primary_health_check" {
  fqdn              = aws_lb.primary_alb.dns_name  # Use ALB DNS Name
  type              = "HTTP"
  port              = 80
  request_interval  = 30
  failure_threshold = 3
}

# Secondary Health Check (us-west-2)
resource "aws_route53_health_check" "secondary_health_check" {
  fqdn              = aws_lb.secondary_alb.dns_name  # Use ALB DNS Name
  type              = "HTTP"
  port              = 80
  request_interval  = 30
  failure_threshold = 3
}
```

k) Verification and Testing of Resources

After saving `route53.tf`, I ran the following commands to verify the deployment:

```
terraform init
terraform plan
terraform apply -auto-approve
```

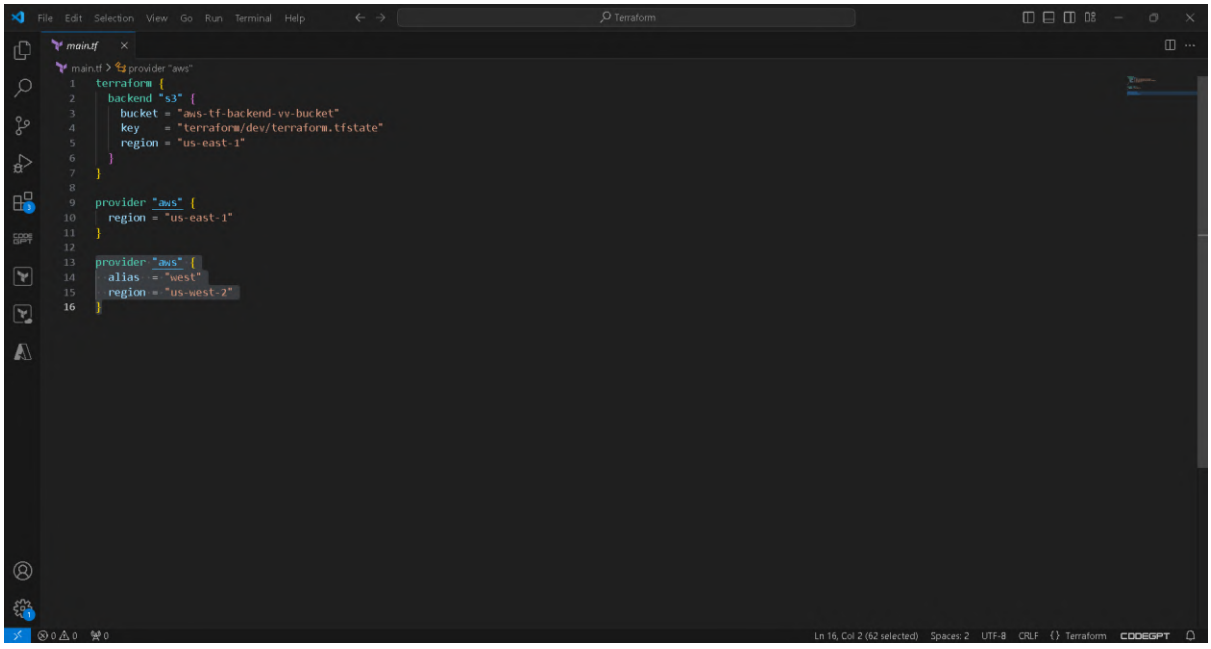
The plan showed 33 resources to add, confirming the configurations for this phase. After successful deployment, I checked the AWS Console to verify each of the resources as expected:

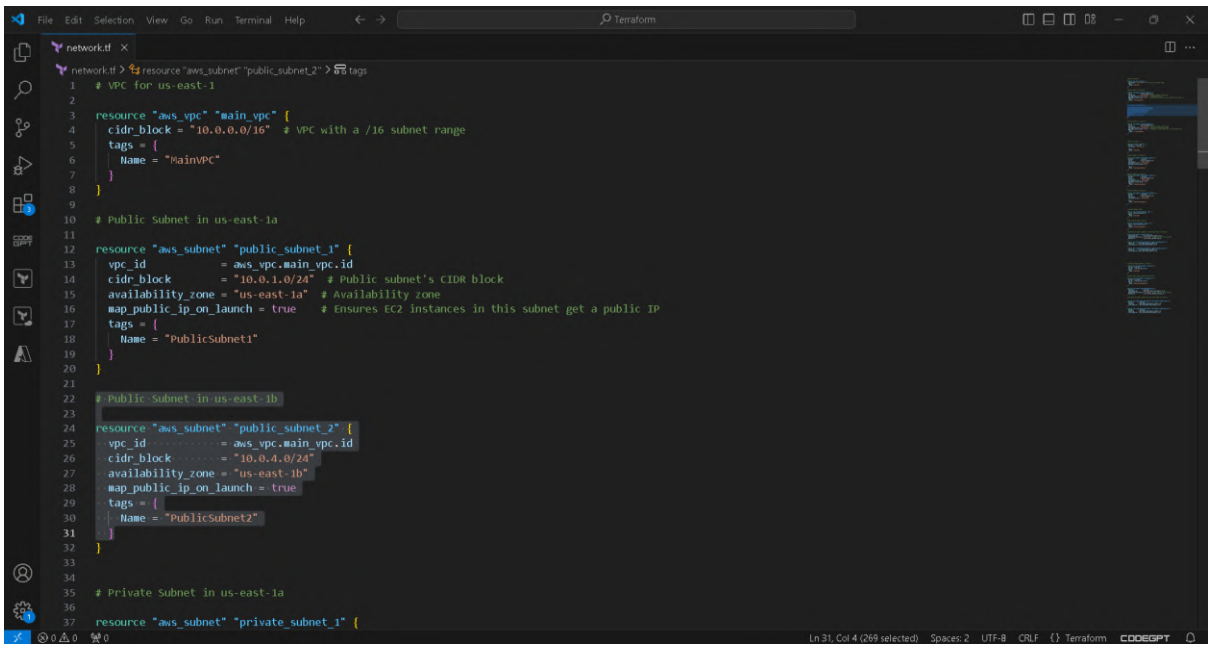
- **VPCs and Subnets**: Verified the `WestVPC`, `WestPublicSubnet1`, `WestPublicSubnet2`, `WestPrivateSubnet1`, and the route tables configured in both regions.
- **Internet Gateways and Route Tables**: Confirmed the presence and configuration of `WestIGW` and associated public route tables.
- **Security Groups**: Verified `WestAppSecurityGroup` and its inbound/outbound rules.
- **Instances**: Observed that 2 instances were running in `us-west-2` as expected.
- **Application Load Balancers**: Verified both `primary-alb` and `secondary-alb`, including listeners, target groups, and health checks.
- **Route 53**: Confirmed that the `myinfra.local` Route 53 hosted zone was created with failover routing configured between `us-east-1` and `us-west-2`, with healthy status for both health checks.

With all 33 resources created and verified as expected, this phase of the project was successfully completed.

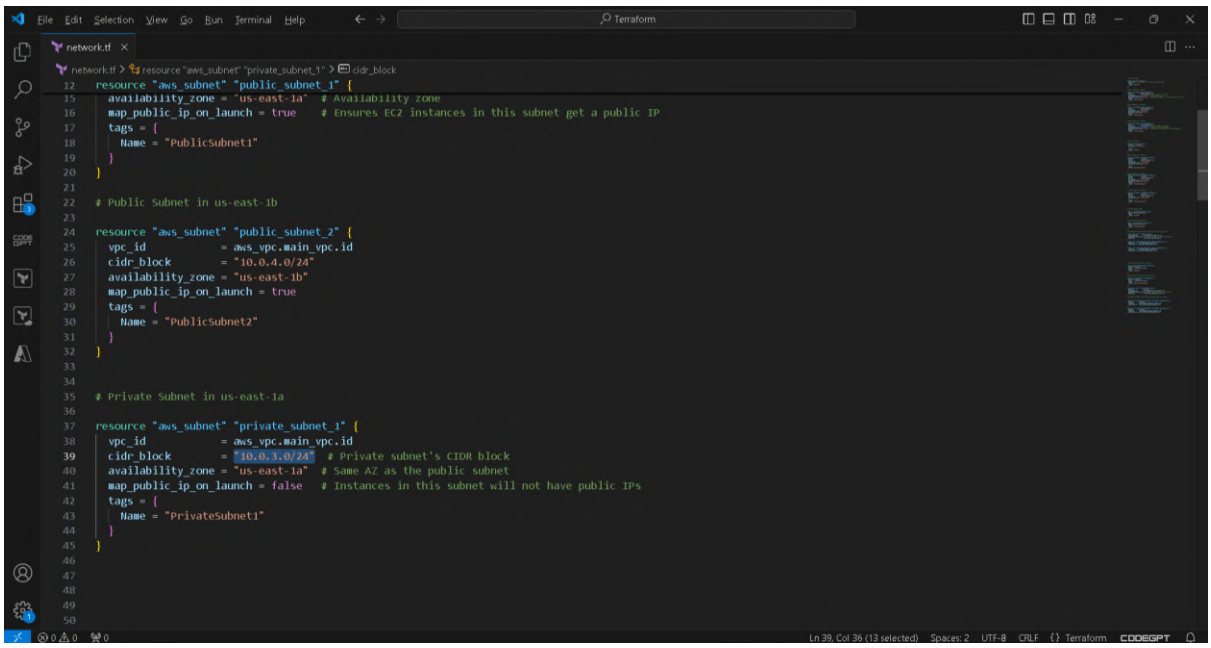
This completes the multi-region deployment and failover setup, ensuring that the application can seamlessly handle failover between `us-east-1` and `us-west-2` in the event of an outage in the primary region.

Screenshots

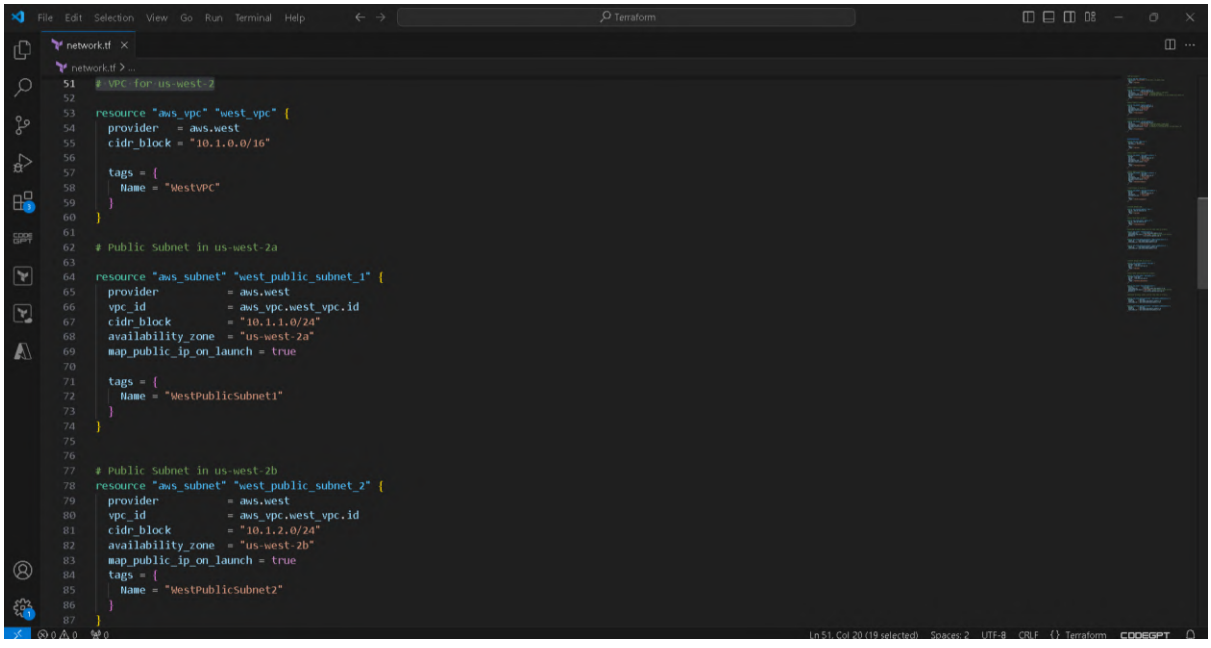




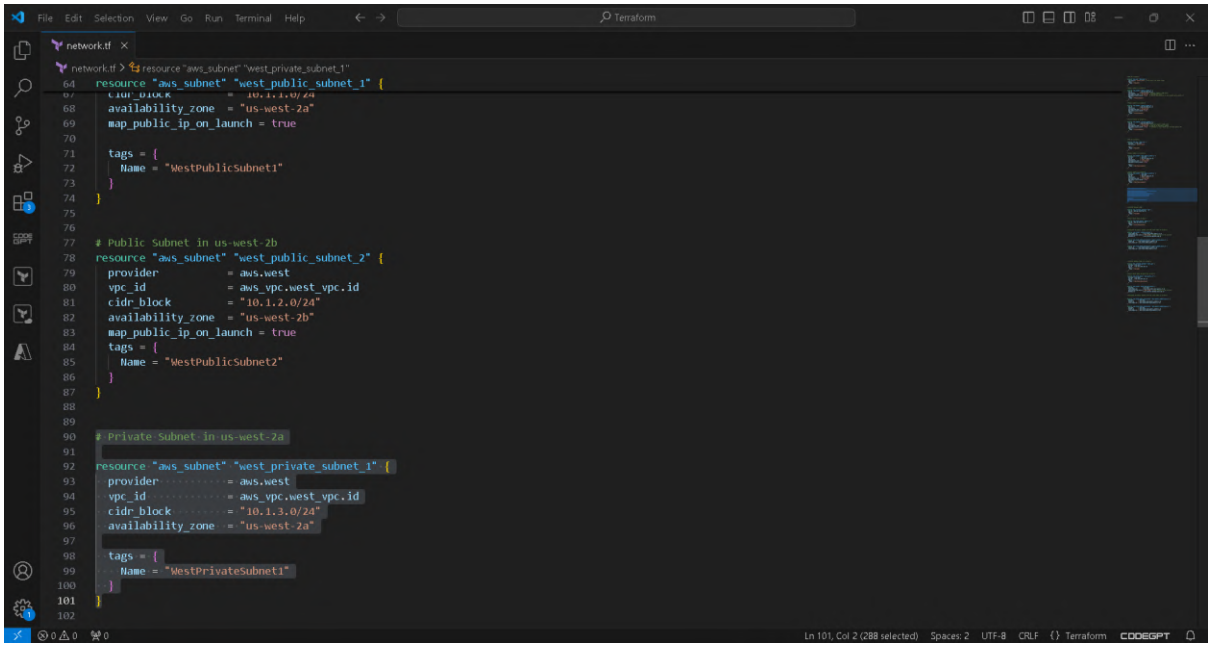
```
# VPC for us-east-1

resource "aws_vpc" "main_vpc" {
  cidr_block = "10.0.0.0/16"  # VPC with a /16 subnet range
  tags = {
    Name = "MainVPC"
  }
}

# Public Subnet in us-east-1a

resource "aws_subnet" "public_subnet_1" {
  vpc_id            = aws_vpc.main_vpc.id
  cidr_block        = "10.0.1.0/24"  # Public subnet's CIDR block
  availability_zone = "us-east-1a"  # Availability zone
  map_public_ip_on_launch = true    # Ensures EC2 instances in this subnet get a public IP
  tags = {
    Name = "PublicSubnet1"
  }
}

# Public Subnet in us-east-1b

resource "aws_subnet" "public_subnet_2" {
  vpc_id            = aws_vpc.main_vpc.id
  cidr_block        = "10.0.4.0/24"
  availability_zone = "us-east-1b"
  map_public_ip_on_launch = true
  tags = {
    Name = "PublicSubnet2"
  }
}


# Private Subnet in us-east-1a

resource "aws_subnet" "private_subnet_1" {
```



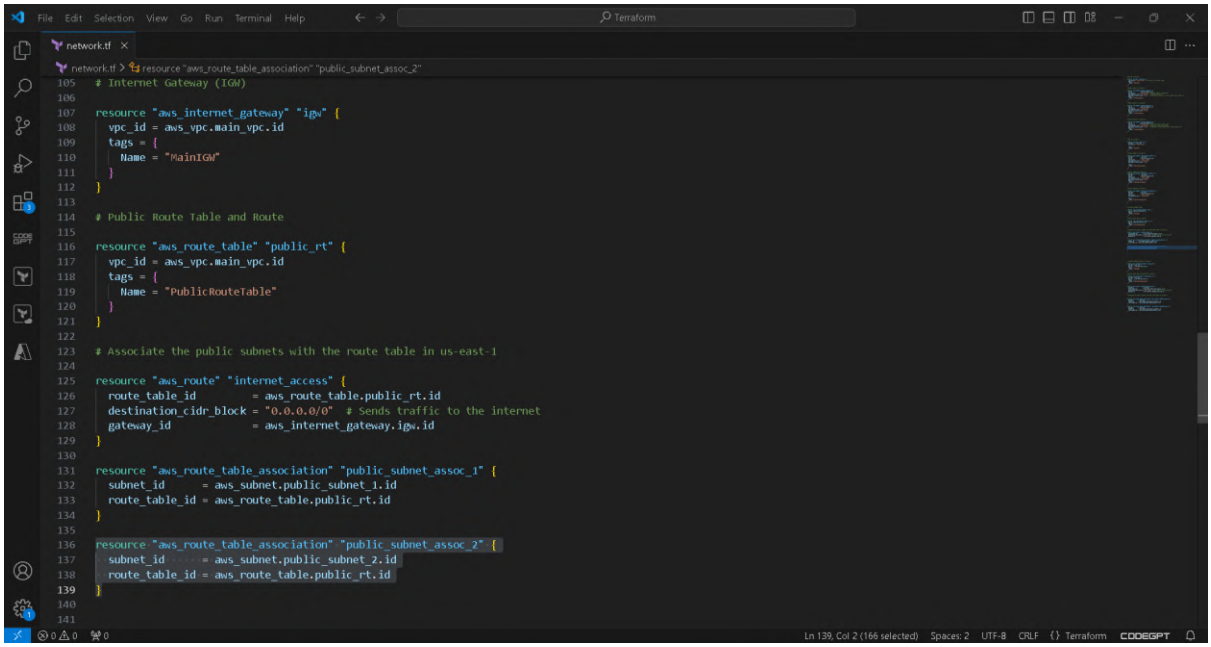
```
resource "aws_subnet" "public_subnet_1" {
  availability_zone = "us-east-1a"  # Availability zone
  map_public_ip_on_launch = true    # Ensures EC2 instances in this subnet get a public IP
  tags = {
    Name = "PublicSubnet1"
  }
}

# Public Subnet in us-east-1b

resource "aws_subnet" "public_subnet_2" {
  vpc_id            = aws_vpc.main_vpc.id
  cidr_block        = "10.0.4.0/24"
  availability_zone = "us-east-1b"
  map_public_ip_on_launch = true
  tags = {
    Name = "PublicSubnet2"
  }
}


# Private Subnet in us-east-1a

resource "aws_subnet" "private_subnet_1" {
  vpc_id            = aws_vpc.main_vpc.id
  cidr_block        = "10.0.3.0/24"  # Private subnet's CIDR block
  availability_zone = "us-east-1a"  # Same AZ as the public subnet
  map_public_ip_on_launch = false   # Instances in this subnet will not have public IPs
  tags = {
    Name = "PrivateSubnet1"
  }
}
```



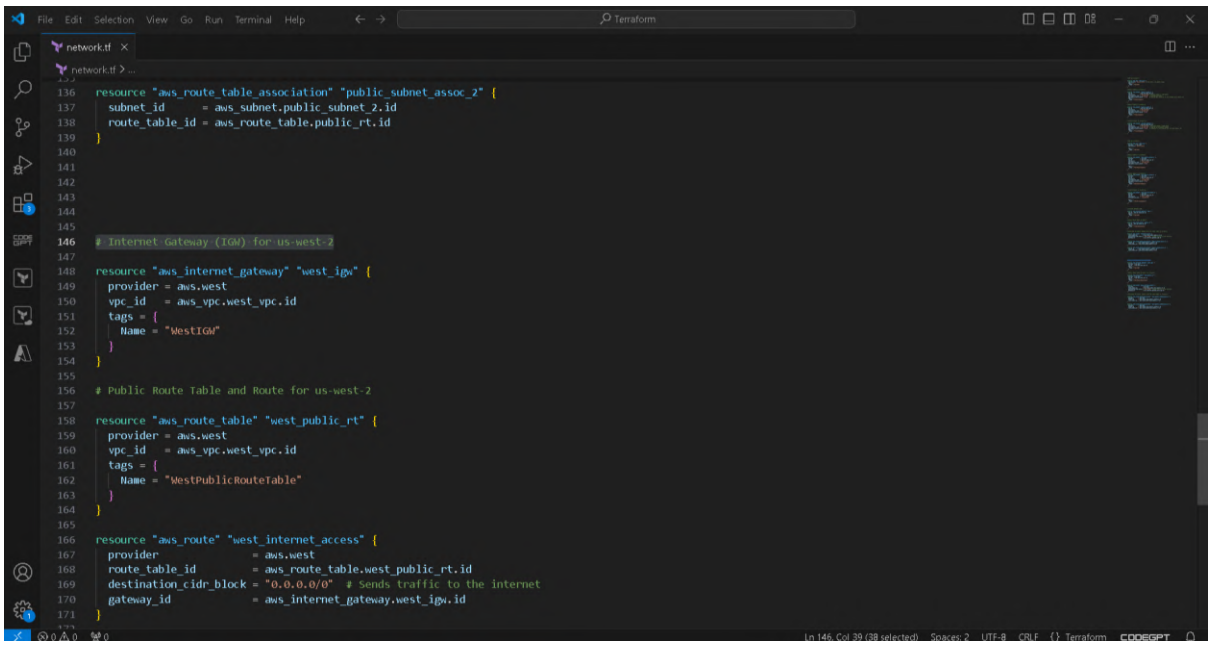
```
# VPC for us-west-2

resource "aws_vpc" "west_vpc" {
  provider   = aws.west
  cidr_block = "10.1.0.0/16"

  tags = {
    Name = "WestVPC"
  }
}

# Public Subnet in us-west-2a

resource "aws_subnet" "west_public_subnet_1" {
  provider          = aws.west
  vpc_id            = aws_vpc.west_vpc.id
  cidr_block        = "10.1.1.0/24"
  availability_zone  = "us-west-2a"
  map_public_ip_on_launch = true

  tags = {
    Name = "WestPublicSubnet1"
  }
}


# Public Subnet in us-west-2b
resource "aws_subnet" "west_public_subnet_2" {
  provider          = aws.west
  vpc_id            = aws_vpc.west_vpc.id
  cidr_block        = "10.1.2.0/24"
  availability_zone  = "us-west-2b"
  map_public_ip_on_launch = true
  tags = {
    Name = "WestPublicSubnet2"
  }
}
```



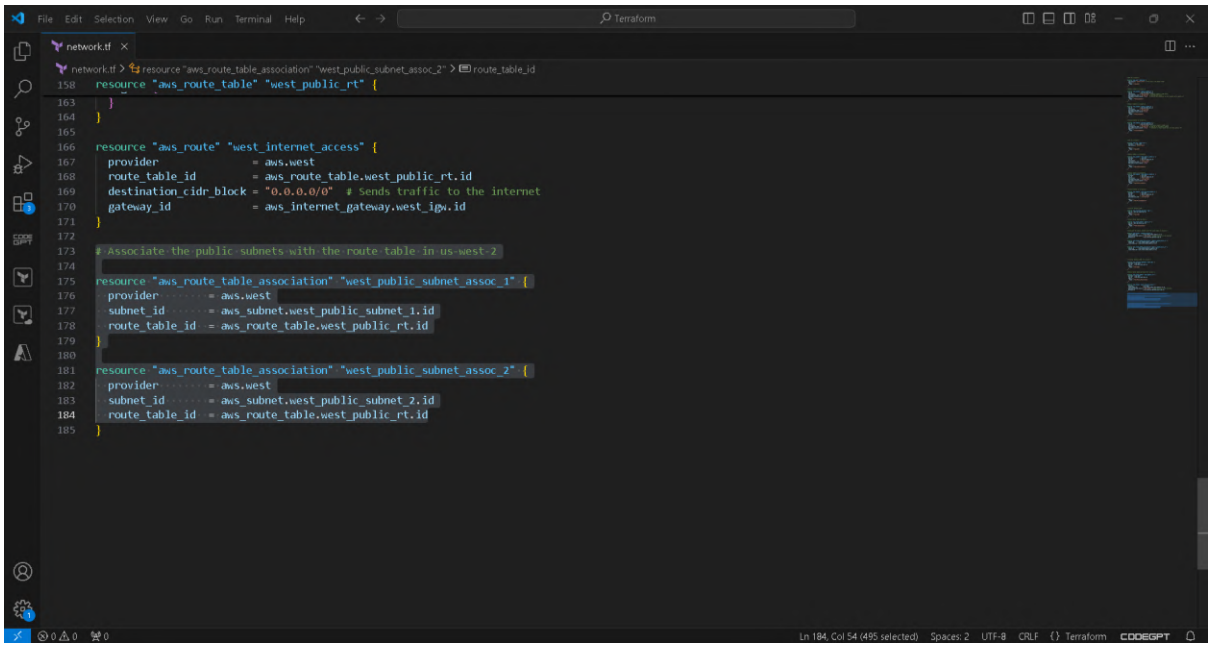
```
resource "aws_subnet" "west_public_subnet_1" {
  cidr_block        = "10.1.1.0/24"
  availability_zone = "us-west-2a"
  map_public_ip_on_launch = true

  tags = {
    Name = "WestPublicSubnet1"
  }
}


# Public Subnet in us-west-2b
resource "aws_subnet" "west_public_subnet_2" {
  provider          = aws.west
  vpc_id            = aws_vpc.west_vpc.id
  cidr_block        = "10.1.2.0/24"
  availability_zone = "us-west-2b"
  map_public_ip_on_launch = true
  tags = {
    Name = "WestPublicSubnet2"
  }
}


# Private Subnet in us-west-2a

resource "aws_subnet" "west_private_subnet_1" {
  provider          = aws.west
  vpc_id            = aws_vpc.west_vpc.id
  cidr_block        = "10.1.3.0/24"
  availability_zone = "us-west-2a"

  tags = {
    Name = "WestPrivateSubnet1"
  }
}
```



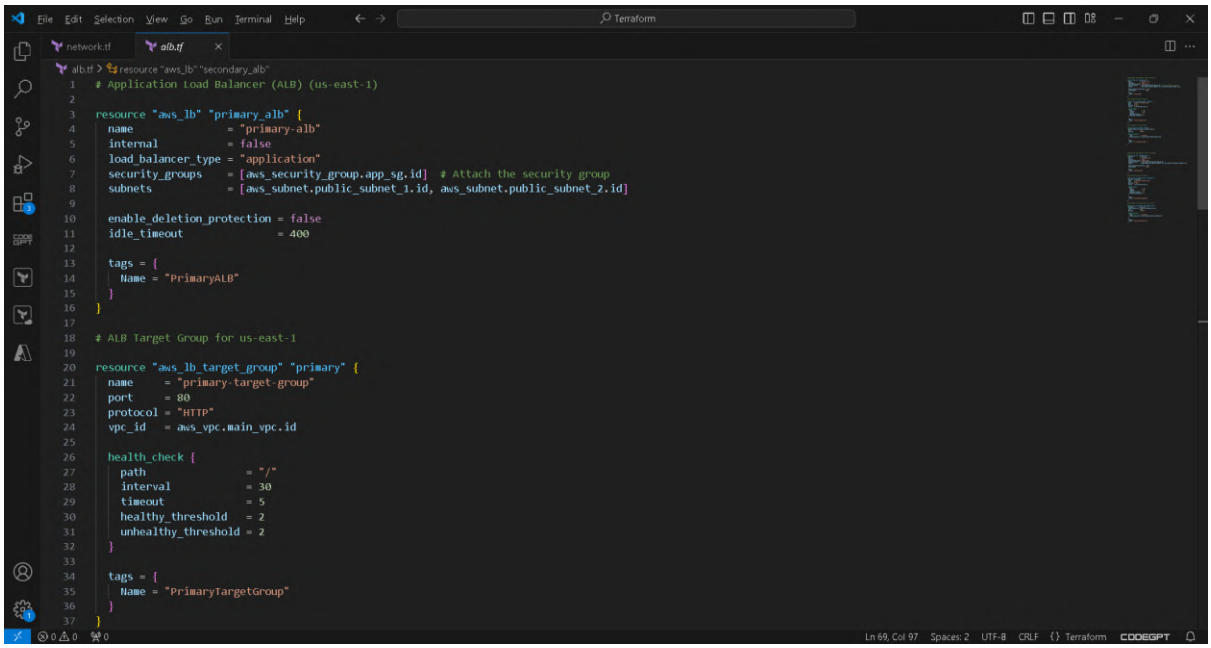
```
# Internet Gateway (IGW)

resource "aws_internet_gateway" "igw" {
  vpc_id = aws_vpc.main_vpc.id
  tags = {
    Name = "MainIGW"
  }
}

# Public Route Table and Route

resource "aws_route_table" "public_rt" {
  vpc_id = aws_vpc.main_vpc.id
  tags = {
    Name = "PublicRouteTable"
  }
}

# Associate the public subnets with the route table in us-east-1

resource "aws_route" "internet_access" {
  route_table_id         = aws_route_table.public_rt.id
  destination_cidr_block = "0.0.0.0/0"  # Sends traffic to the internet
  gateway_id             = aws_internet_gateway.igw.id
}

resource "aws_route_table_association" "public_subnet_assoc_1" {
  subnet_id      = aws_subnet.public_subnet_1.id
  route_table_id = aws_route_table.public_rt.id
}

resource "aws_route_table_association" "public_subnet_assoc_2" {
  subnet_id      = aws_subnet.public_subnet_2.id
  route_table_id = aws_route_table.public_rt.id
}
```



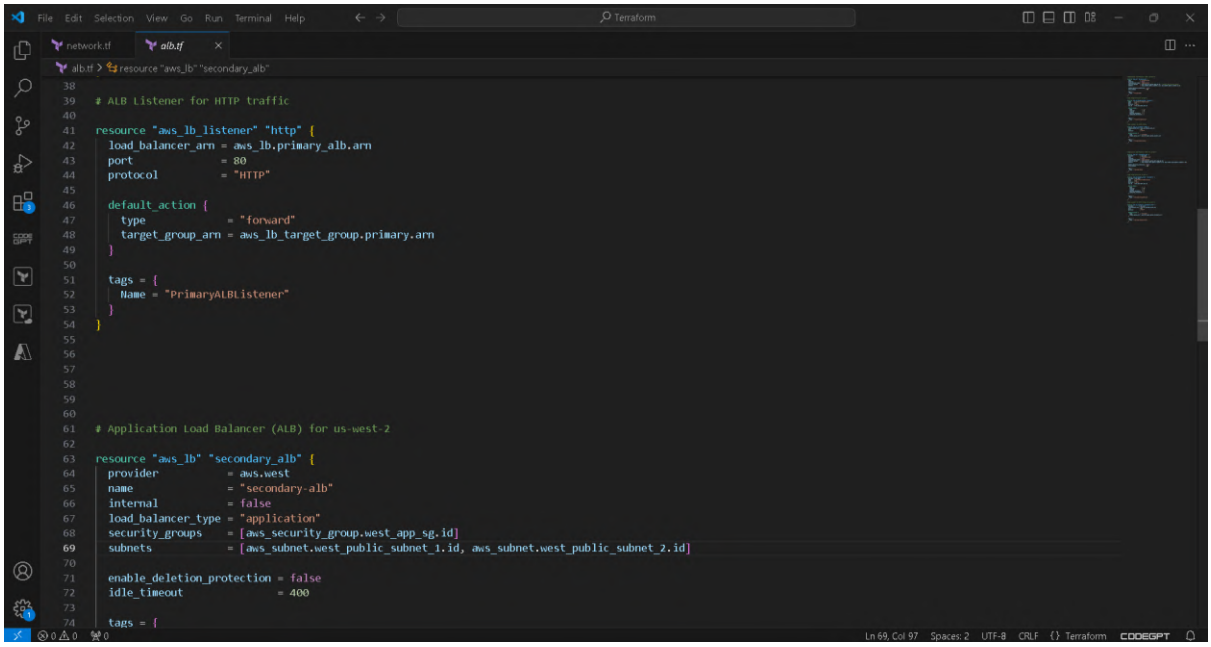
```
resource "aws_route_table_association" "public_subnet_assoc_2" {
  subnet_id      = aws_subnet.public_subnet_2.id
  route_table_id = aws_route_table.public_rt.id
}




# Internet Gateway (IGW) for us-west-2

resource "aws_internet_gateway" "west_igw" {
  provider = aws.west
  vpc_id   = aws_vpc.west_vpc.id
  tags = {
    Name = "WestIGW"
  }
}

# Public Route Table and Route for us-west-2

resource "aws_route_table" "west_public_rt" {
  provider = aws.west
  vpc_id   = aws_vpc.west_vpc.id
  tags = {
    Name = "WestPublicRouteTable"
  }
}

resource "aws_route" "west_internet_access" {
  provider               = aws.west
  route_table_id         = aws_route_table.west_public_rt.id
  destination_cidr_block = "0.0.0.0/0"  # Sends traffic to the internet
  gateway_id             = aws_internet_gateway.west_igw.id
}
```



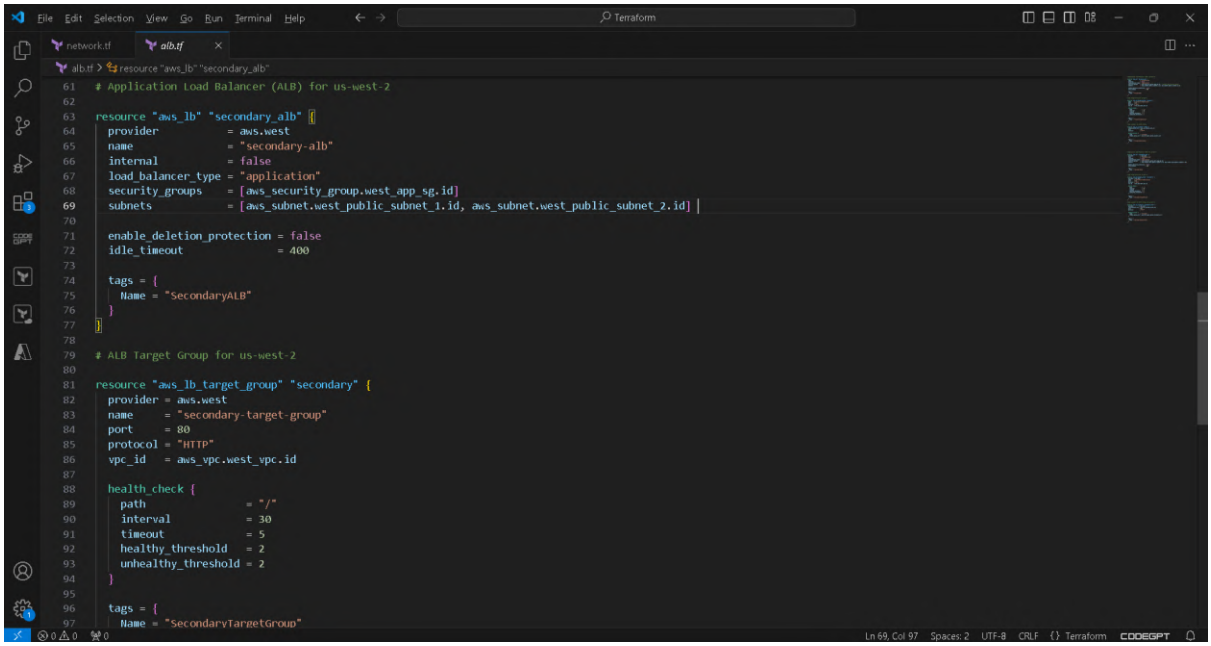
```
resource "aws_route_table" "west_public_rt" {
  }
}

resource "aws_route" "west_internet_access" {
  provider               = aws.west
  route_table_id         = aws_route_table.west_public_rt.id
  destination_cidr_block = "0.0.0.0/0"  # Sends traffic to the internet
  gateway_id             = aws_internet_gateway.west_igw.id
}

# Associate the public subnets with the route table in us-west-2

resource "aws_route_table_association" "west_public_subnet_assoc_1" {
  provider       = aws.west
  subnet_id      = aws_subnet.west_public_subnet_1.id
  route_table_id = aws_route_table.west_public_rt.id
}

resource "aws_route_table_association" "west_public_subnet_assoc_2" {
  provider       = aws.west
  subnet_id      = aws_subnet.west_public_subnet_2.id
  route_table_id = aws_route_table.west_public_rt.id
}
```



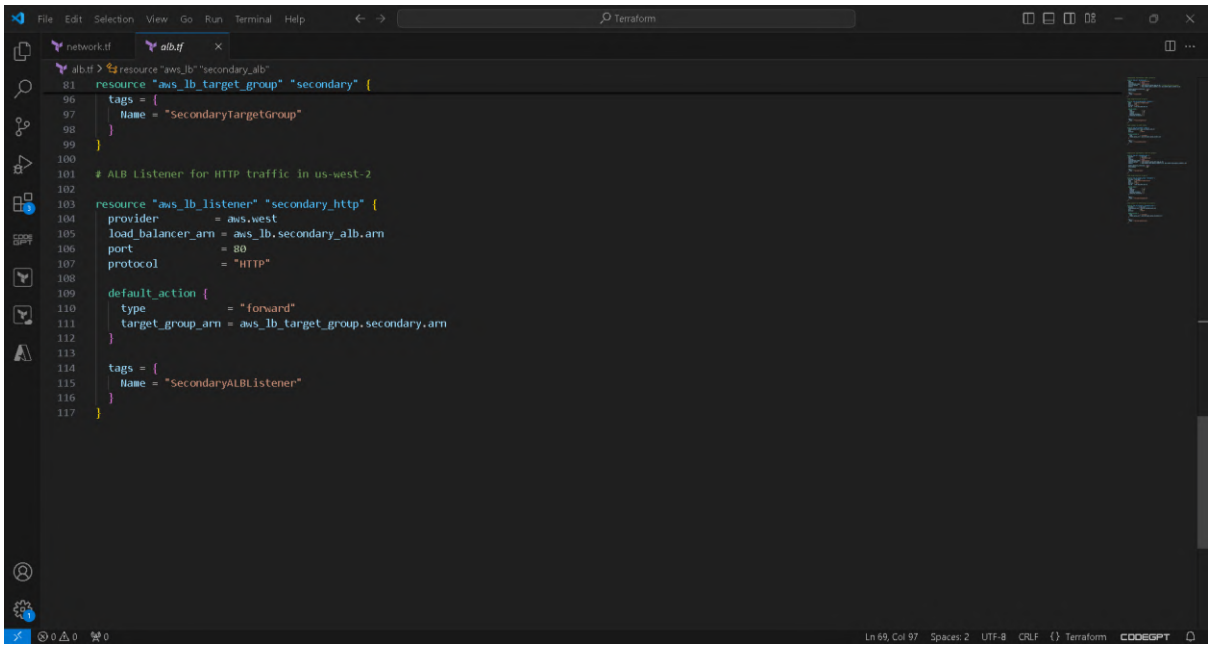
```
# Application Load Balancer (ALB) (us-east-1)

resource "aws_lb" "primary_alb" {
  name               = "primary-alb"
  internal           = false
  load_balancer_type = "application"
  security_groups    = [aws_security_group.app_sg.id]  # Attach the security group
  subnets            = [aws_subnet.public_subnet_1.id, aws_subnet.public_subnet_2.id]

  enable_deletion_protection = false
  idle_timeout               = 400

  tags = {
    Name = "PrimaryALB"
  }
}

# ALB Target Group for us-east-1

resource "aws_lb_target_group" "primary" {
  name     = "primary-target-group"
  port     = 80
  protocol = "HTTP"
  vpc_id   = aws_vpc.main_vpc.id

  health_check {
    path                = "/"
    interval            = 30
    timeout             = 5
    healthy_threshold   = 2
    unhealthy_threshold = 2
  }

  tags = {
    Name = "PrimaryTargetGroup"
  }
}
```



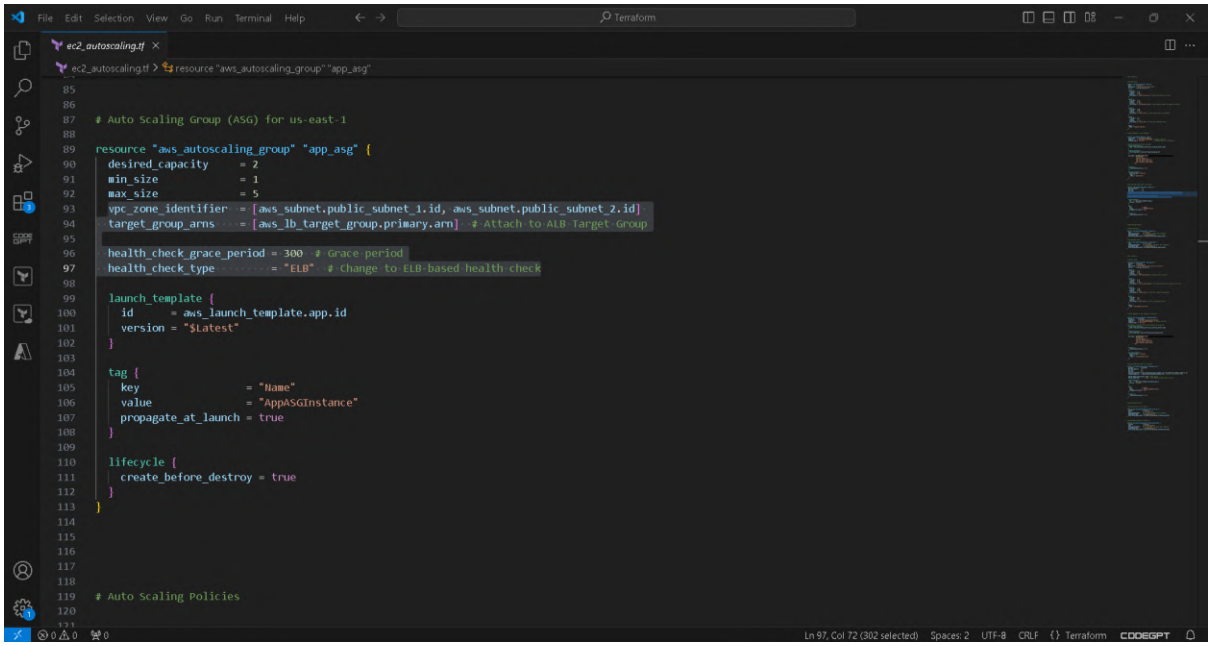
```
# ALB Listener for HTTP traffic

resource "aws_lb_listener" "http" {
  load_balancer_arn = aws_lb.primary_alb.arn
  port              = 80
  protocol          = "HTTP"

  default_action {
    type             = "forward"
    target_group_arn = aws_lb_target_group.primary.arn
  }

  tags = {
    Name = "PrimaryALBListener"
  }
}




# Application Load Balancer (ALB) for us-west-2

resource "aws_lb" "secondary_alb" {
  provider           = aws.west
  name               = "secondary-alb"
  internal           = false
  load_balancer_type = "application"
  security_groups    = [aws_security_group.west_app_sg.id]
  subnets            = [aws_subnet.west_public_subnet_1.id, aws_subnet.west_public_subnet_2.id]

  enable_deletion_protection = false
  idle_timeout               = 400

  tags = {
```



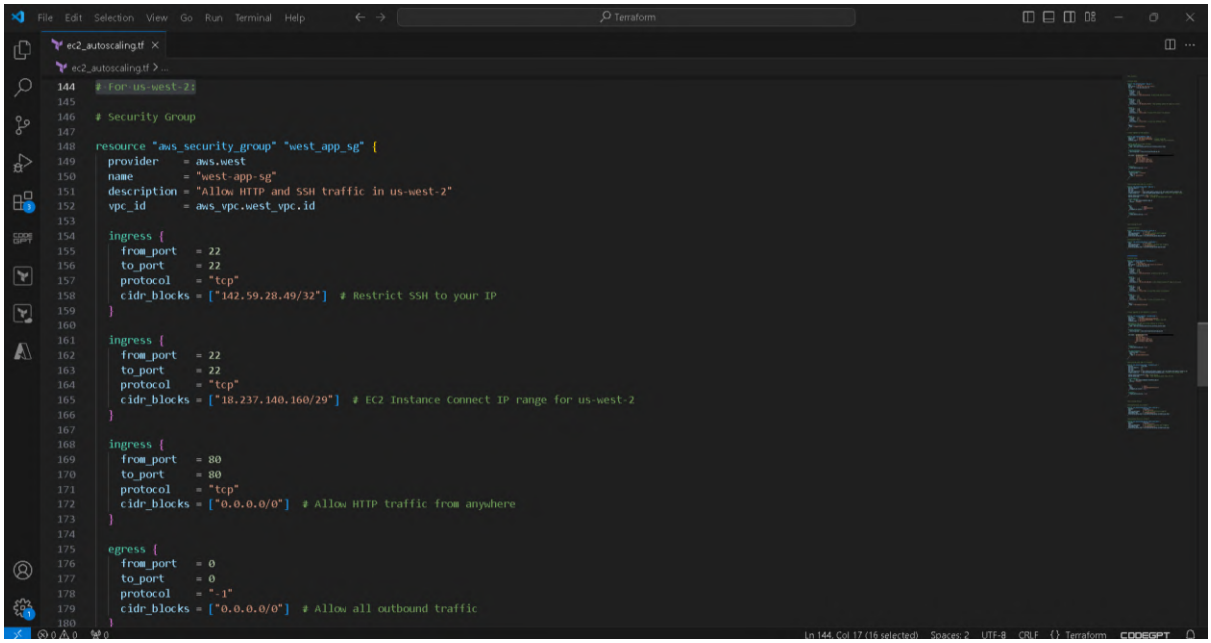
```
# Application Load Balancer (ALB) for us-west-2

resource "aws_lb" "secondary_alb" {
  provider           = aws.west
  name               = "secondary-alb"
  internal           = false
  load_balancer_type = "application"
  security_groups    = [aws_security_group.west_app_sg.id]
  subnets            = [aws_subnet.west_public_subnet_1.id, aws_subnet.west_public_subnet_2.id]

  enable_deletion_protection = false
  idle_timeout               = 400

  tags = {
    Name = "SecondaryALB"
  }
}

# ALB Target Group for us-west-2

resource "aws_lb_target_group" "secondary" {
  provider = aws.west
  name     = "secondary-target-group"
  port     = 80
  protocol = "HTTP"
  vpc_id   = aws_vpc.west_vpc.id

  health_check {
    path                = "/"
    interval            = 30
    timeout             = 5
    healthy_threshold   = 2
    unhealthy_threshold = 2
  }

  tags = {
    Name = "SecondaryTargetGroup"
```



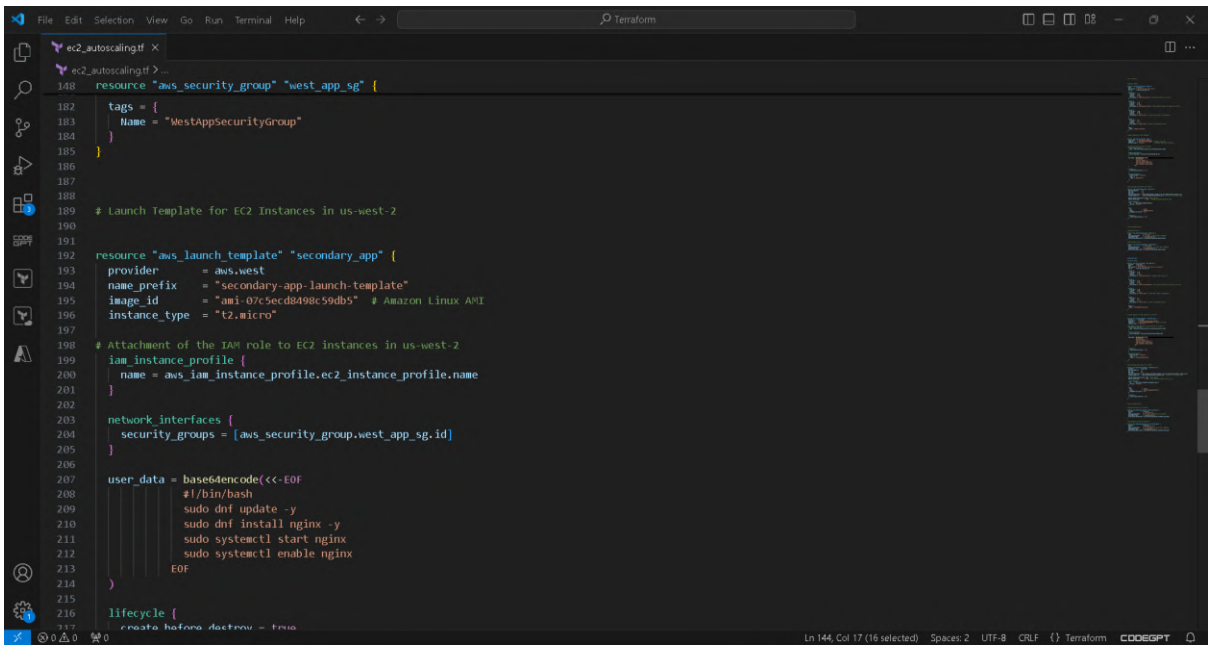
```
resource "aws_lb_target_group" "secondary" {
  tags = {
    Name = "SecondaryTargetGroup"
  }
}

# ALB Listener for HTTP traffic in us-west-2

resource "aws_lb_listener" "secondary_http" {
  provider          = aws.west
  load_balancer_arn = aws_lb.secondary_alb.arn
  port              = 80
  protocol          = "HTTP"

  default_action {
    type             = "forward"
    target_group_arn = aws_lb_target_group.secondary.arn
  }

  tags = {
    Name = "SecondaryALBListener"
  }
}
```



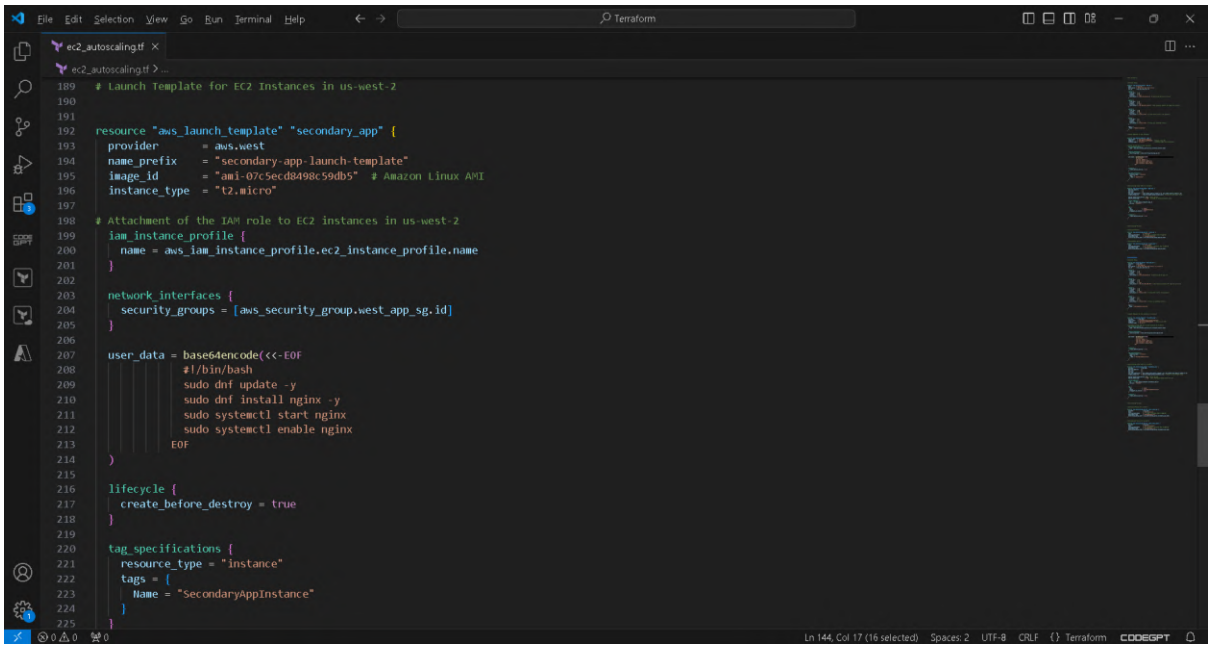
```
# Auto Scaling Group (ASG) for us-east-1

resource "aws_autoscaling_group" "app_asg" {
  desired_capacity    = 2
  min_size            = 1
  max_size            = 5
  vpc_zone_identifier = [aws_subnet.public_subnet_1.id, aws_subnet.public_subnet_2.id]
  target_group_arns   = [aws_lb_target_group.primary.arn]  # Attach to ALB Target Group

  health_check_grace_period = 300  # Grace period
  health_check_type         = "ELB"  # Change to ELB-based health check

  launch_template {
    id      = aws_launch_template.app.id
    version = "$Latest"
  }

  tag {
    key                 = "Name"
    value               = "AppASGInstance"
    propagate_at_launch = true
  }

  lifecycle {
    create_before_destroy = true
  }
}

# Auto Scaling Policies
```



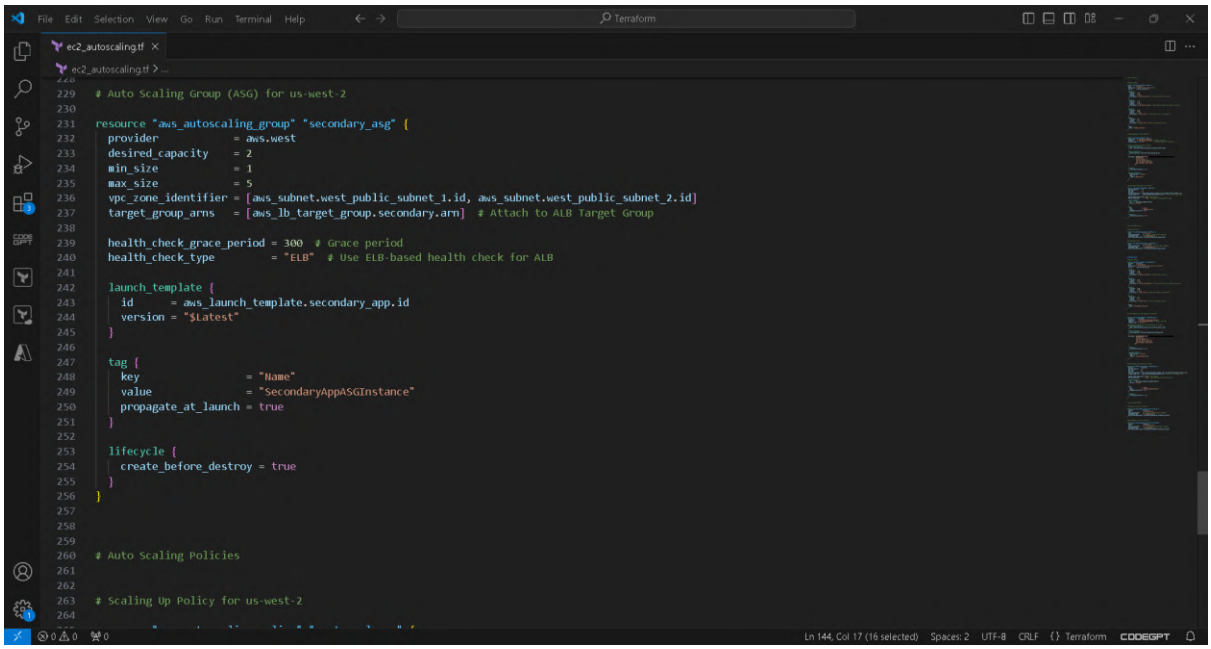
```
# For us-west-2a

# Security Group

resource "aws_security_group" "west_app_sg" {
  provider    = aws.west
  name        = "west-app-sg"
  description = "Allow HTTP and SSH traffic in us-west-2"
  vpc_id      = aws_vpc.west_vpc.id

  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["142.59.28.49/32"]  # Restrict SSH to your IP
  }

  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["18.237.140.160/29"]  # EC2 Instance Connect IP range for us-west-2
  }

  ingress {
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]  # Allow HTTP traffic from anywhere
  }

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]  # Allow all outbound traffic
  }
```



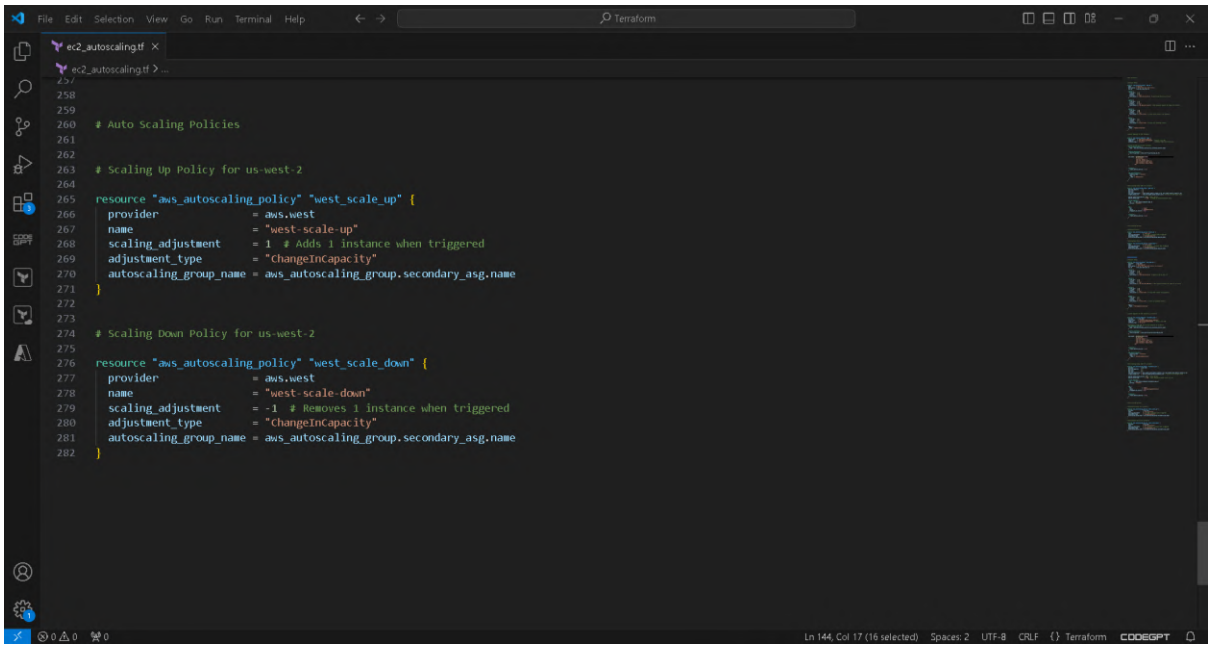
```
resource "aws_security_group" "west_app_sg" {
  tags = {
    Name = "WestAppSecurityGroup"
  }
}


# Launch Template for EC2 Instances in us-west-2


resource "aws_launch_template" "secondary_app" {
  provider       = aws.west
  name_prefix    = "secondary-app-launch-template"
  image_id       = "ami-07c5ecd8498c59db5"  # Amazon Linux AMI
  instance_type  = "t2.micro"

# Attachment of the IAM role to EC2 instances in us-west-2
  iam_instance_profile {
    name = aws_iam_instance_profile.ec2_instance_profile.name
  }

  network_interfaces {
    security_groups = [aws_security_group.west_app_sg.id]
  }

  user_data = base64encode(<<-EOF
              #!/bin/bash
              sudo dnf update -y
              sudo dnf install nginx -y
              sudo systemctl start nginx
              sudo systemctl enable nginx
              EOF
  )

  lifecycle {
    create_before_destroy = true
```



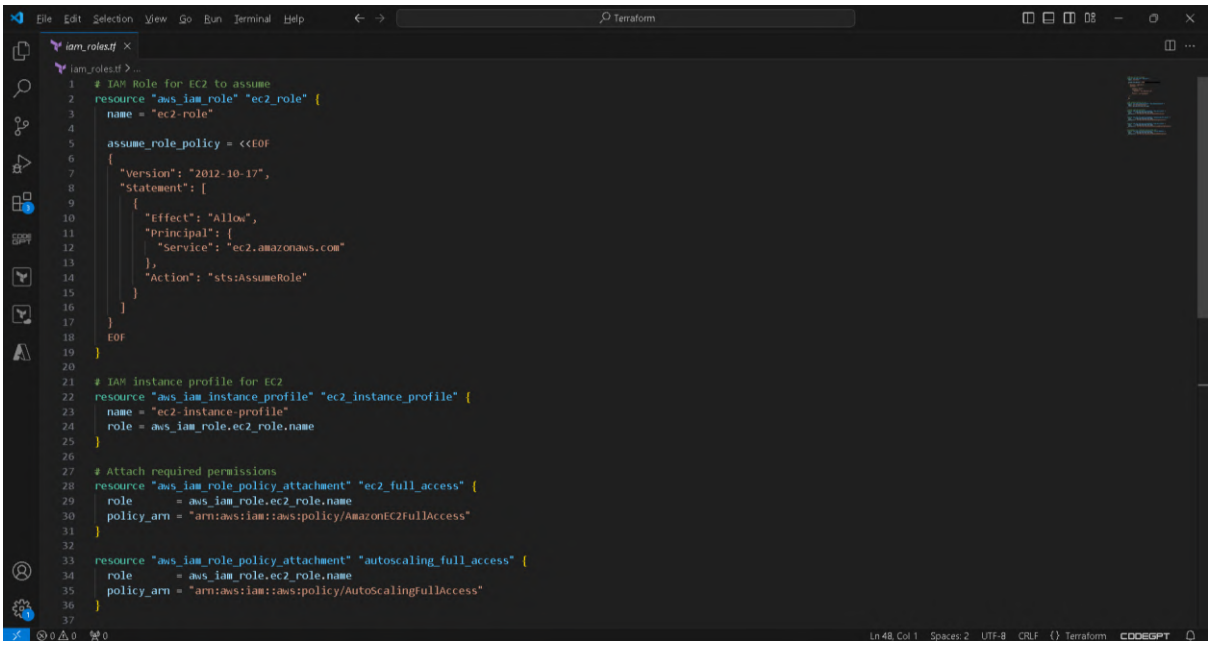
```
# Launch Template for EC2 Instances in us-west-2


resource "aws_launch_template" "secondary_app" {
  provider       = aws.west
  name_prefix    = "secondary-app-launch-template"
  image_id       = "ami-07c5ecd8498c59db5"  # Amazon Linux AMI
  instance_type  = "t2.micro"

# Attachment of the IAM role to EC2 instances in us-west-2
  iam_instance_profile {
    name = aws_iam_instance_profile.ec2_instance_profile.name
  }

  network_interfaces {
    security_groups = [aws_security_group.west_app_sg.id]
  }

  user_data = base64encode(<<-EOF
              #!/bin/bash
              sudo dnf update -y
              sudo dnf install nginx -y
              sudo systemctl start nginx
              sudo systemctl enable nginx
              EOF
  )

  lifecycle {
    create_before_destroy = true
  }

  tag_specifications {
    resource_type = "instance"
    tags = {
      Name = "SecondaryAppInstance"
    }
  }
```



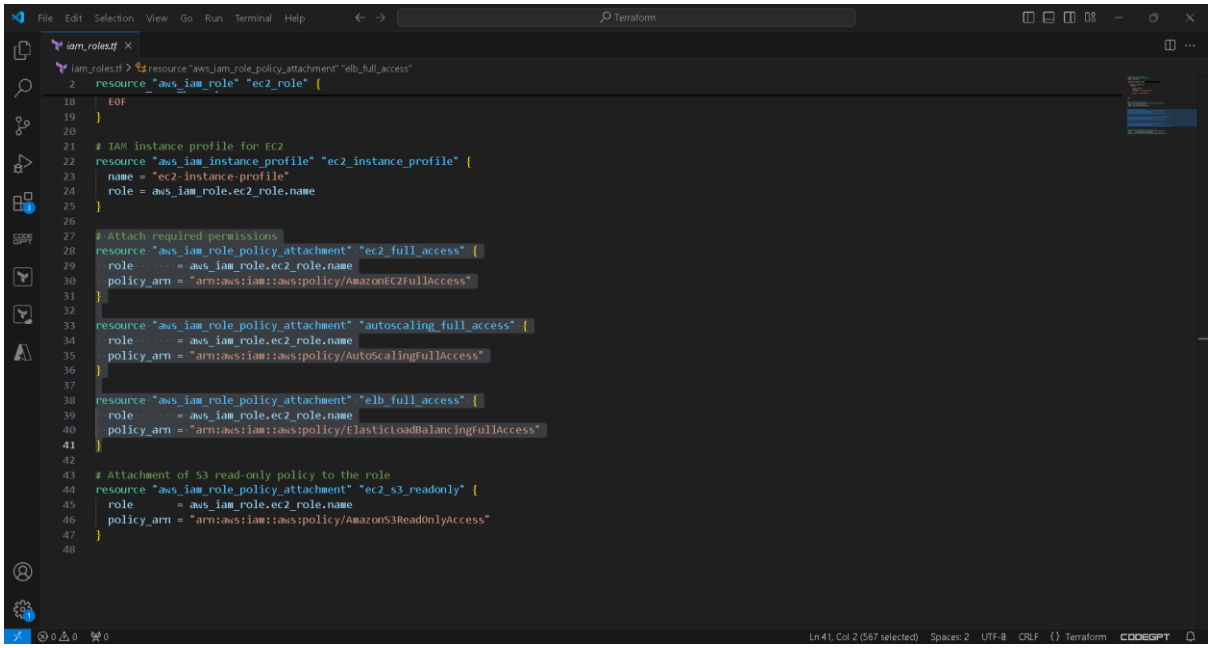
```
# Auto Scaling Group (ASG) for us-west-2

resource "aws_autoscaling_group" "secondary_asg" {
  provider            = aws.west
  desired_capacity    = 2
  min_size            = 1
  max_size            = 5
  vpc_zone_identifier = [aws_subnet.west_public_subnet_1.id, aws_subnet.west_public_subnet_2.id]
  target_group_arns   = [aws_lb_target_group.secondary.arn]  # Attach to ALB Target Group

  health_check_grace_period = 300  # Grace period
  health_check_type         = "ELB"  # Use ELB-based health check for ALB

  launch_template {
    id      = aws_launch_template.secondary_app.id
    version = "$Latest"
  }

  tag {
    key                 = "Name"
    value               = "SecondaryAppASGInstance"
    propagate_at_launch = true
  }

  lifecycle {
    create_before_destroy = true
  }
}



# Auto Scaling Policies


# Scaling Up Policy for us-west-2
```



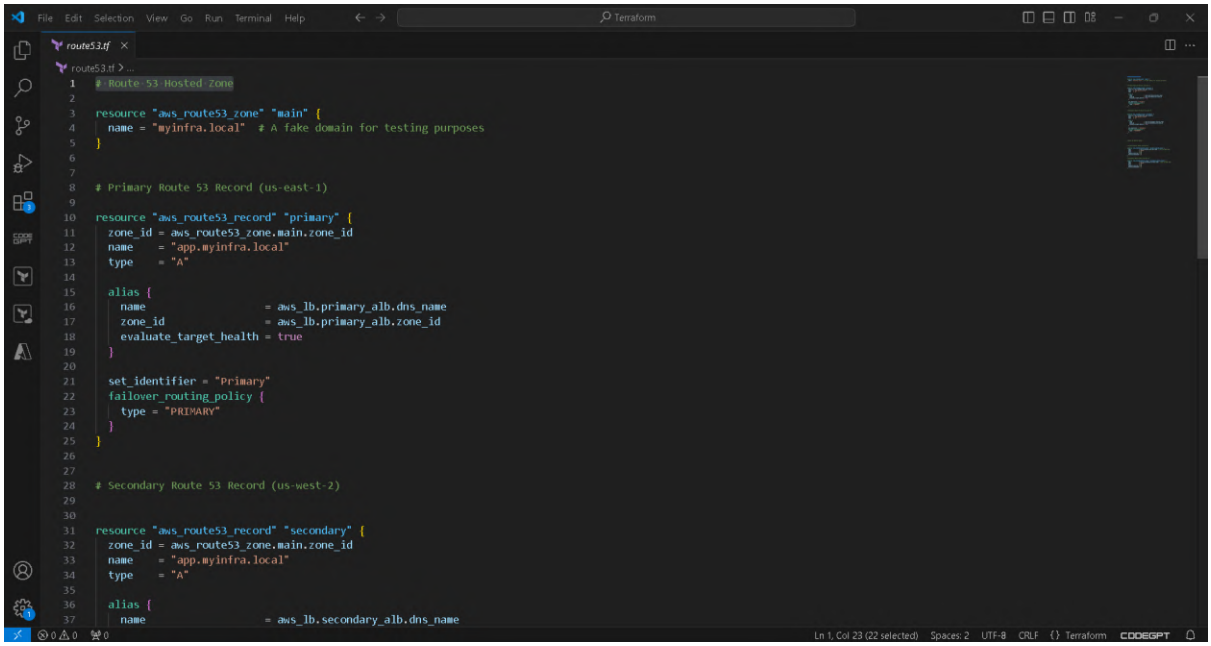
```
# Auto Scaling Policies


# Scaling Up Policy for us-west-2

resource "aws_autoscaling_policy" "west_scale_up" {
  provider               = aws.west
  name                   = "west-scale-up"
  scaling_adjustment     = 1  # Adds 1 instance when triggered
  adjustment_type        = "ChangeInCapacity"
  autoscaling_group_name = aws_autoscaling_group.secondary_asg.name
}


# Scaling Down Policy for us-west-2

resource "aws_autoscaling_policy" "west_scale_down" {
  provider               = aws.west
  name                   = "west-scale-down"
  scaling_adjustment     = -1  # Removes 1 instance when triggered
  adjustment_type        = "ChangeInCapacity"
  autoscaling_group_name = aws_autoscaling_group.secondary_asg.name
}
```



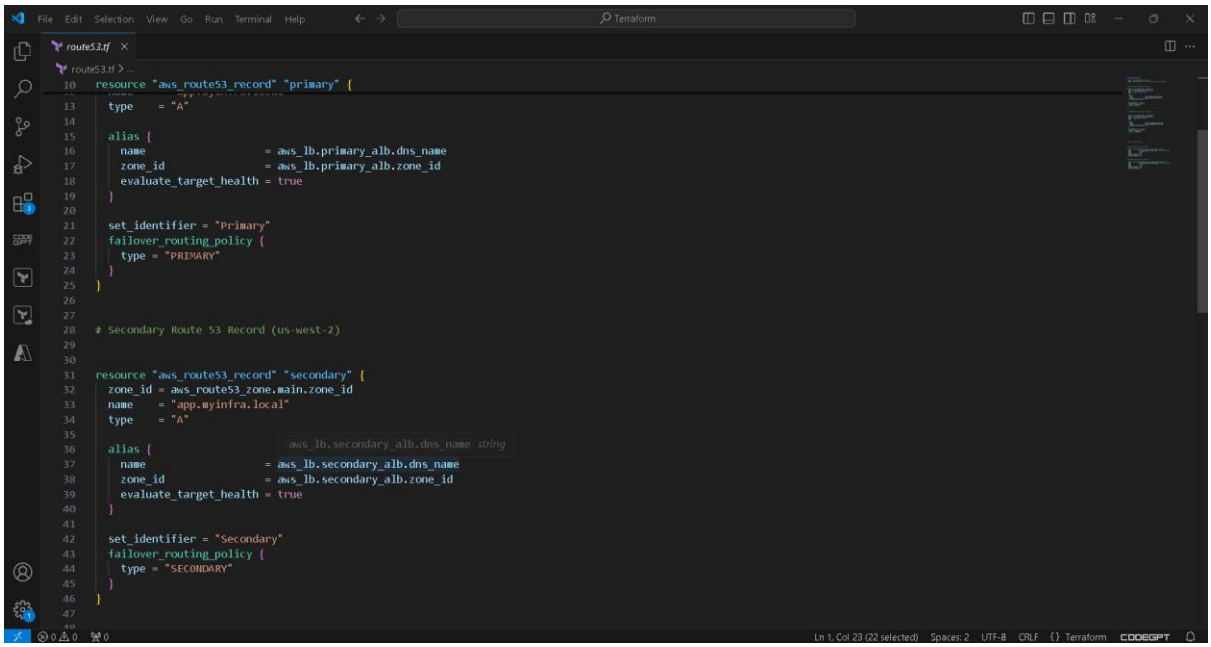
```
# IAM Role for EC2 to assume
resource "aws_iam_role" "ec2_role" {
  name = "ec2-role"

  assume_role_policy = <<EOF
  {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Principal": {
          "Service": "ec2.amazonaws.com"
        },
        "Action": "sts:AssumeRole"
      }
    ]
  }
  EOF
}

# IAM instance profile for EC2
resource "aws_iam_instance_profile" "ec2_instance_profile" {
  name = "ec2-instance-profile"
  role = aws_iam_role.ec2_role.name
}

# Attach required permissions
resource "aws_iam_role_policy_attachment" "ec2_full_access" {
  role       = aws_iam_role.ec2_role.name
  policy_arn = "arn:aws:iam::aws:policy/AmazonEC2FullAccess"
}

resource "aws_iam_role_policy_attachment" "autoscaling_full_access" {
  role       = aws_iam_role.ec2_role.name
  policy_arn = "arn:aws:iam::aws:policy/AutoScalingFullAccess"
}
```



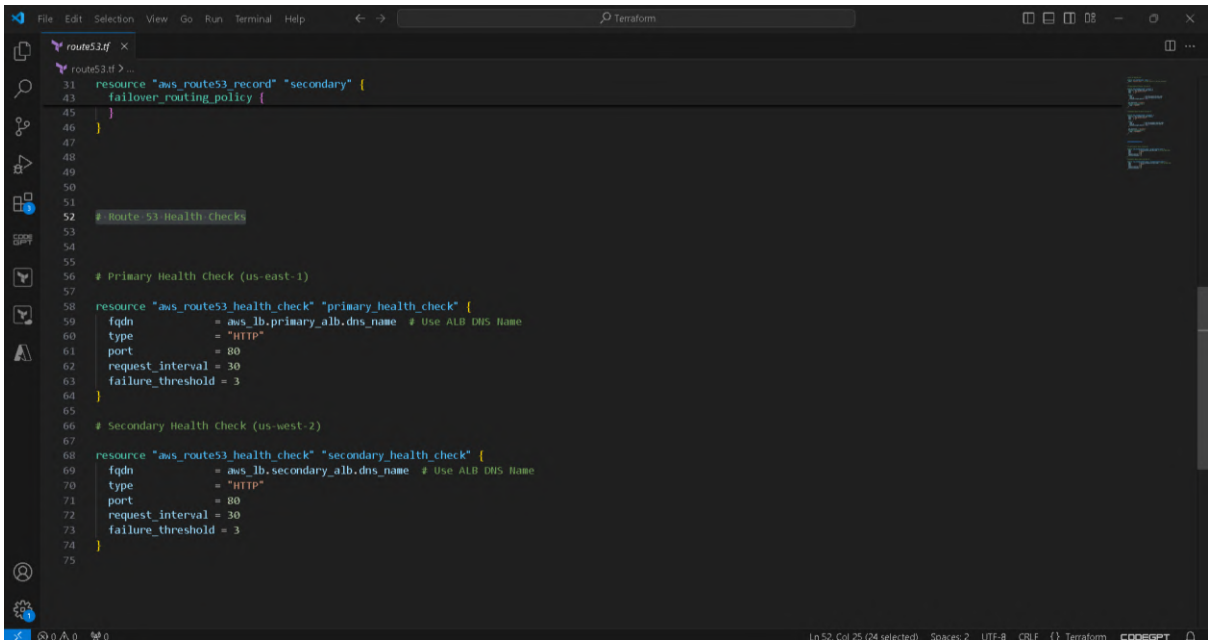
```
resource "aws_iam_role" "ec2_role" {
  EOF
}

# IAM instance profile for EC2
resource "aws_iam_instance_profile" "ec2_instance_profile" {
  name = "ec2-instance-profile"
  role = aws_iam_role.ec2_role.name
}

# Attach required permissions
resource "aws_iam_role_policy_attachment" "ec2_full_access" {
  role       = aws_iam_role.ec2_role.name
  policy_arn = "arn:aws:iam::aws:policy/AmazonEC2FullAccess"
}

resource "aws_iam_role_policy_attachment" "autoscaling_full_access" {
  role       = aws_iam_role.ec2_role.name
  policy_arn = "arn:aws:iam::aws:policy/AutoScalingFullAccess"
}

resource "aws_iam_role_policy_attachment" "elb_full_access" {
  role       = aws_iam_role.ec2_role.name
  policy_arn = "arn:aws:iam::aws:policy/ElasticLoadBalancingFullAccess"
}

# Attachment of S3 read-only policy to the role
resource "aws_iam_role_policy_attachment" "ec2_s3_readonly" {
  role       = aws_iam_role.ec2_role.name
  policy_arn = "arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess"
}
```



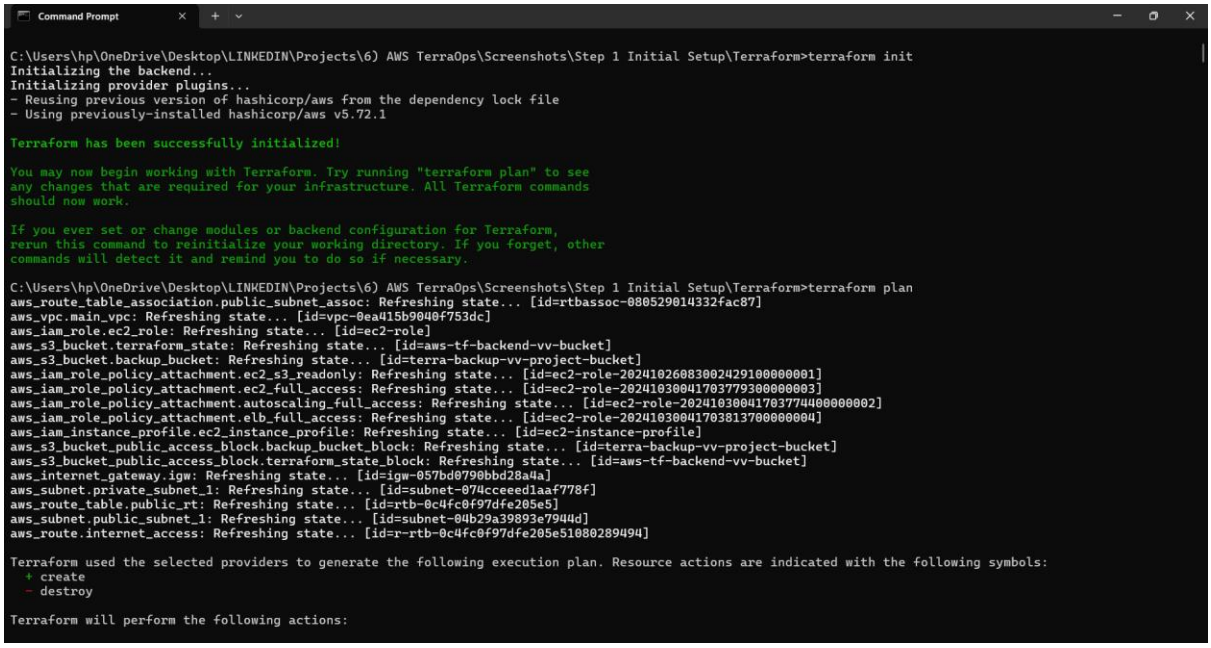
```
# Route 53 Hosted Zone

resource "aws_route53_zone" "main" {
  name = "myinfra.local"  # A fake domain for testing purposes
}


# Primary Route 53 Record (us-east-1)

resource "aws_route53_record" "primary" {
  zone_id = aws_route53_zone.main.zone_id
  name    = "app.myinfra.local"
  type    = "A"

  alias {
    name                   = aws_lb.primary_alb.dns_name
    zone_id                = aws_lb.primary_alb.zone_id
    evaluate_target_health = true
  }

  set_identifier = "Primary"
  failover_routing_policy {
    type = "PRIMARY"
  }
}


# Secondary Route 53 Record (us-west-2)


resource "aws_route53_record" "secondary" {
  zone_id = aws_route53_zone.main.zone_id
  name    = "app.myinfra.local"
  type    = "A"

  alias {
    name                   = aws_lb.secondary_alb.dns_name
```



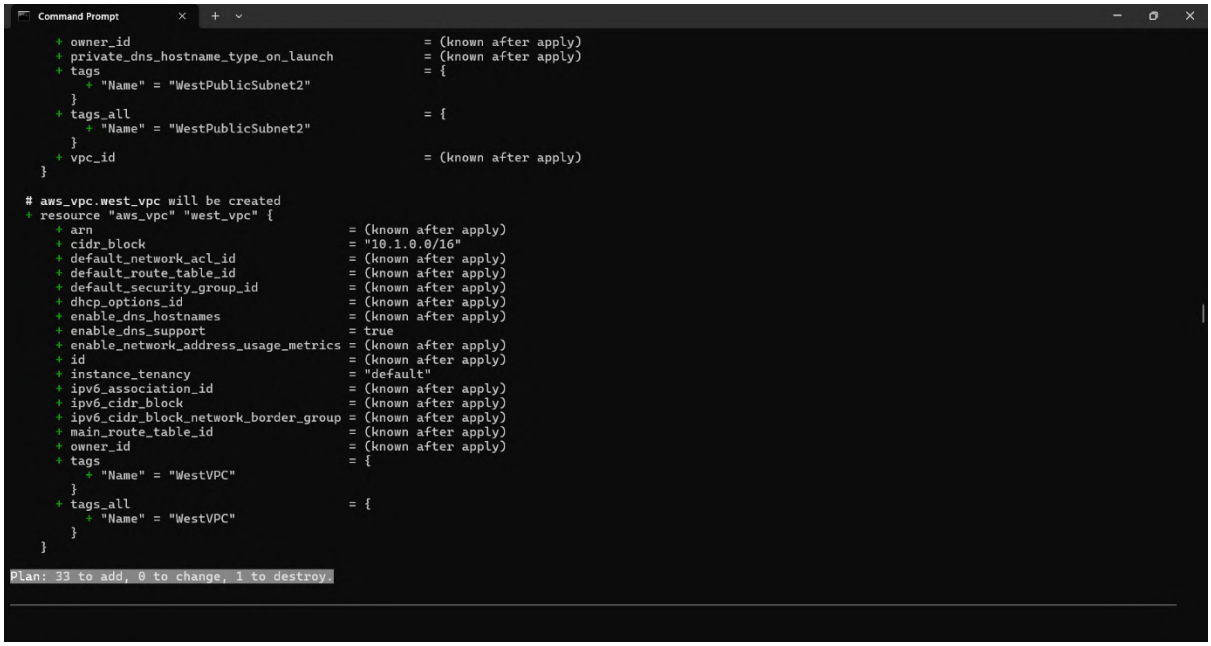
```
resource "aws_route53_record" "primary" {
  type    = "A"

  alias {
    name                   = aws_lb.primary_alb.dns_name
    zone_id                = aws_lb.primary_alb.zone_id
    evaluate_target_health = true
  }

  set_identifier = "Primary"
  failover_routing_policy {
    type = "PRIMARY"
  }
}


# Secondary Route 53 Record (us-west-2)


resource "aws_route53_record" "secondary" {
  zone_id = aws_route53_zone.main.zone_id
  name    = "app.myinfra.local"
  type    = "A"

  alias {
    name                   = aws_lb.secondary_alb.dns_name
    zone_id                = aws_lb.secondary_alb.zone_id
    evaluate_target_health = true
  }

  set_identifier = "Secondary"
  failover_routing_policy {
    type = "SECONDARY"
  }
}
```



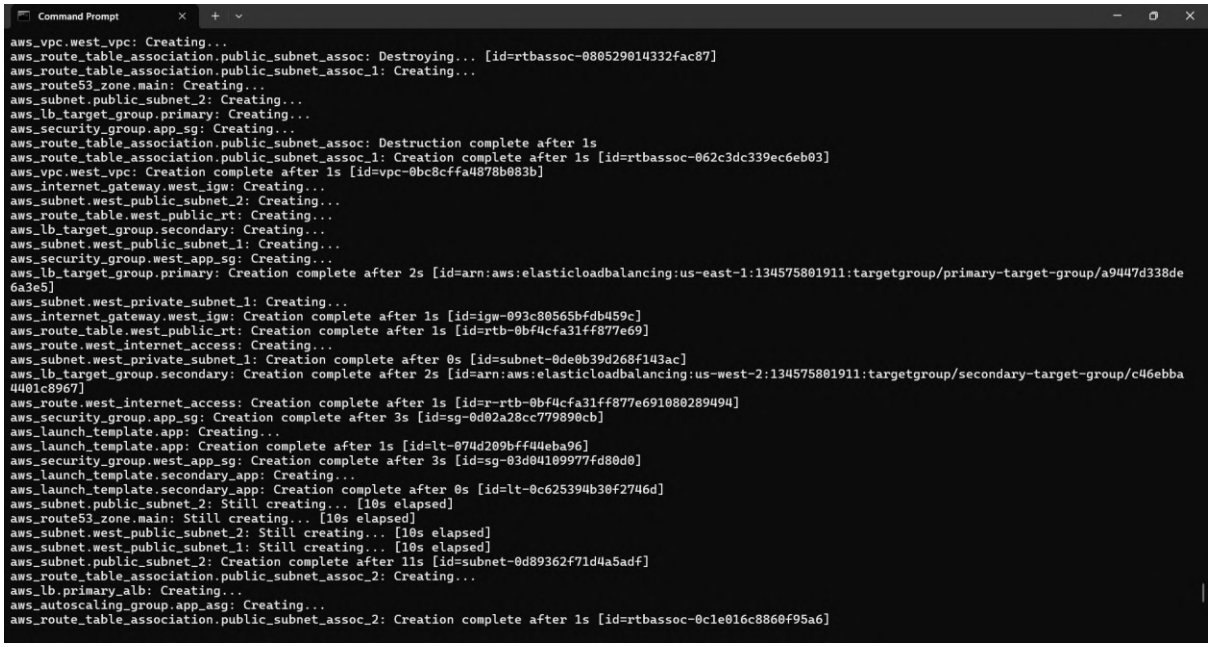
```
File  Edit  Selection  View  Go  Run  Terminal  Help        Terraform

route53.tf

route53.tf > ...
31  resource "aws_route53_record" "secondary" {
43    failover_routing_policy {
45    }
46  }
47
48
49
50
51
52  # Route 53 Health Checks
53
54
55
56  # Primary Health Check (us-east-1)
57
58  resource "aws_route53_health_check" "primary_health_check" {
59    fqdn              = aws_lb.primary_alb.dns_name  # Use ALB DNS Name
60    type              = "HTTP"
61    port              = 80
62    request_interval = 30
63    failure_threshold = 3
64  }
65
66  # Secondary Health Check (us-west-2)
67
68  resource "aws_route53_health_check" "secondary_health_check" {
69    fqdn              = aws_lb.secondary_alb.dns_name  # Use ALB DNS Name
70    type              = "HTTP"
71    port              = 80
72    request_interval = 30
73    failure_threshold = 3
74  }
75
```

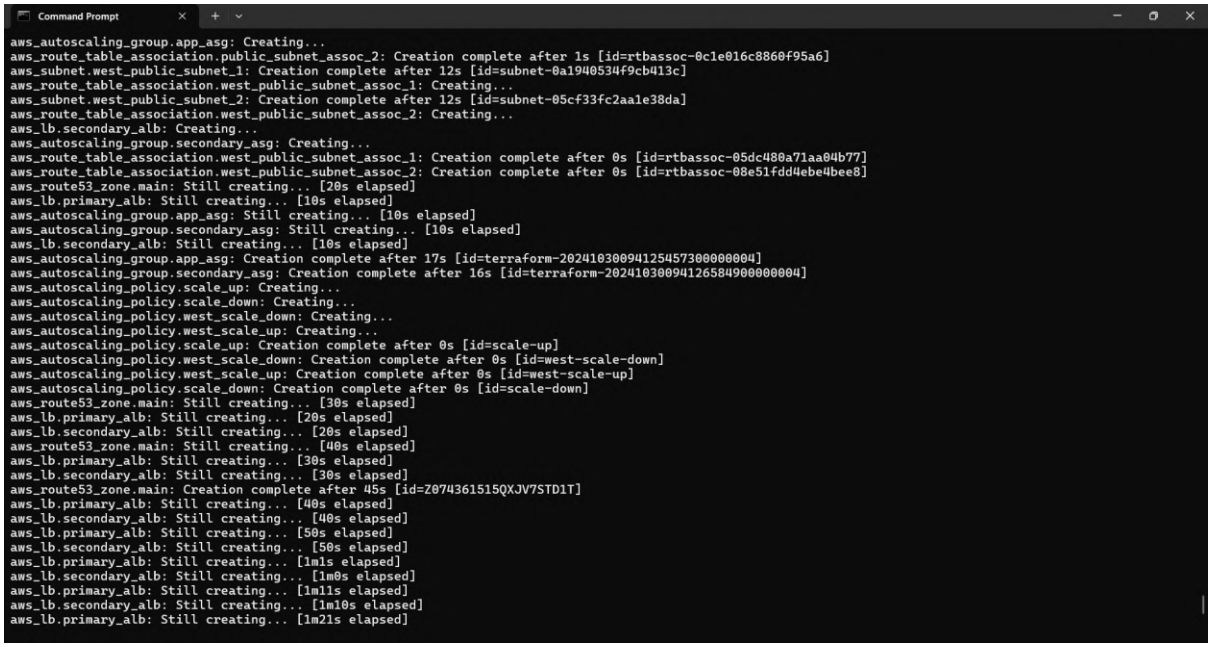


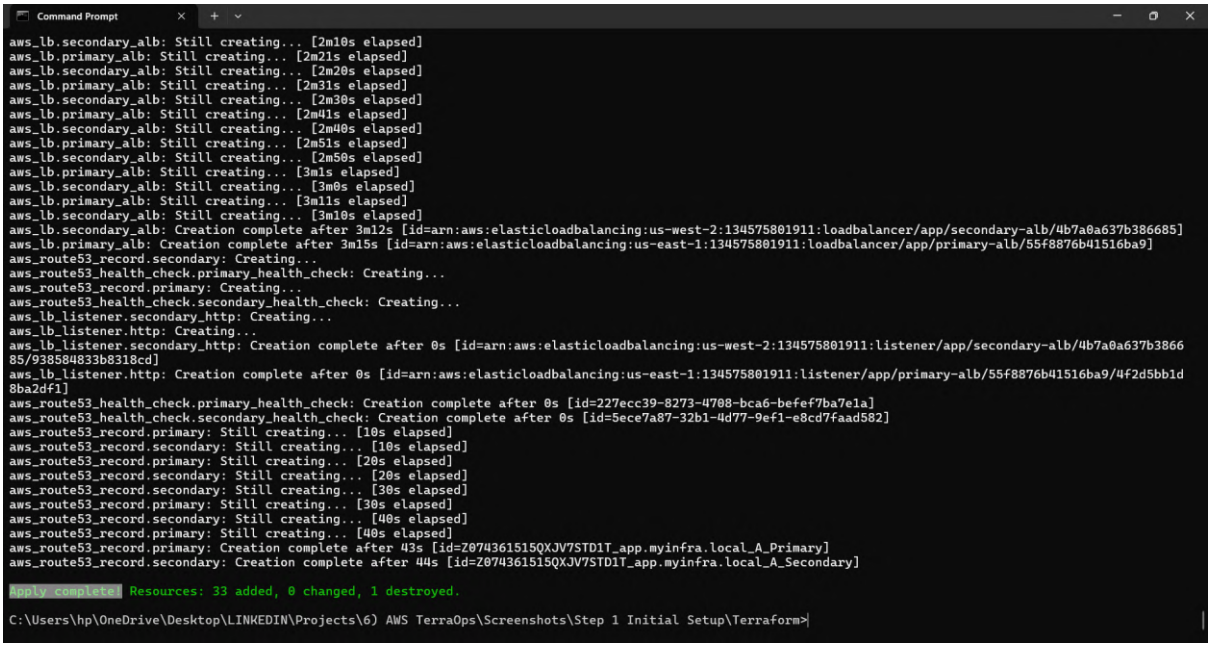
```
Command Prompt

C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.72.1

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.

C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>terraform plan
aws_route_table_association.public_subnet_assoc: Refreshing state... [id=rtbassoc-080529014332fac87]
aws_vpc.main_vpc: Refreshing state... [id=vpc-0ea415b9040f753dc]
aws_iam_role.ec2_role: Refreshing state... [id=ec2-role]
aws_s3_bucket.terraform_state: Refreshing state... [id=aws-tf-backend-vv-bucket]
aws_s3_bucket.backup_bucket: Refreshing state... [id=terra-backup-vv-project-bucket]
aws_iam_role_policy_attachment.ec2_s3_readonly: Refreshing state... [id=ec2-role-20241026083002429100000001]
aws_iam_role_policy_attachment.ec2_full_access: Refreshing state... [id=ec2-role-20241030041703779300000003]
aws_iam_role_policy_attachment.autoscaling_full_access: Refreshing state... [id=ec2-role-20241030041703774400000002]
aws_iam_role_policy_attachment.elb_full_access: Refreshing state... [id=ec2-role-20241030041703813700000004]
aws_iam_instance_profile.ec2_instance_profile: Refreshing state... [id=ec2-instance-profile]
aws_s3_bucket_public_access_block.backup_bucket_block: Refreshing state... [id=terra-backup-vv-project-bucket]
aws_s3_bucket_public_access_block.terraform_state_block: Refreshing state... [id=aws-tf-backend-vv-bucket]
aws_internet_gateway.igw: Refreshing state... [id=igw-057bd0790bbd28a4a]
aws_subnet.private_subnet_1: Refreshing state... [id=subnet-074cceeed1aaf778f]
aws_route_table.public_rt: Refreshing state... [id=rtb-0c4fc0f97dfe205e5]
aws_subnet.public_subnet_1: Refreshing state... [id=subnet-04b29a39893e7944d]
aws_route.internet_access: Refreshing state... [id=r-rtb-0c4fc0f97dfe205e51080289494]

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

Terraform will perform the following actions:
```

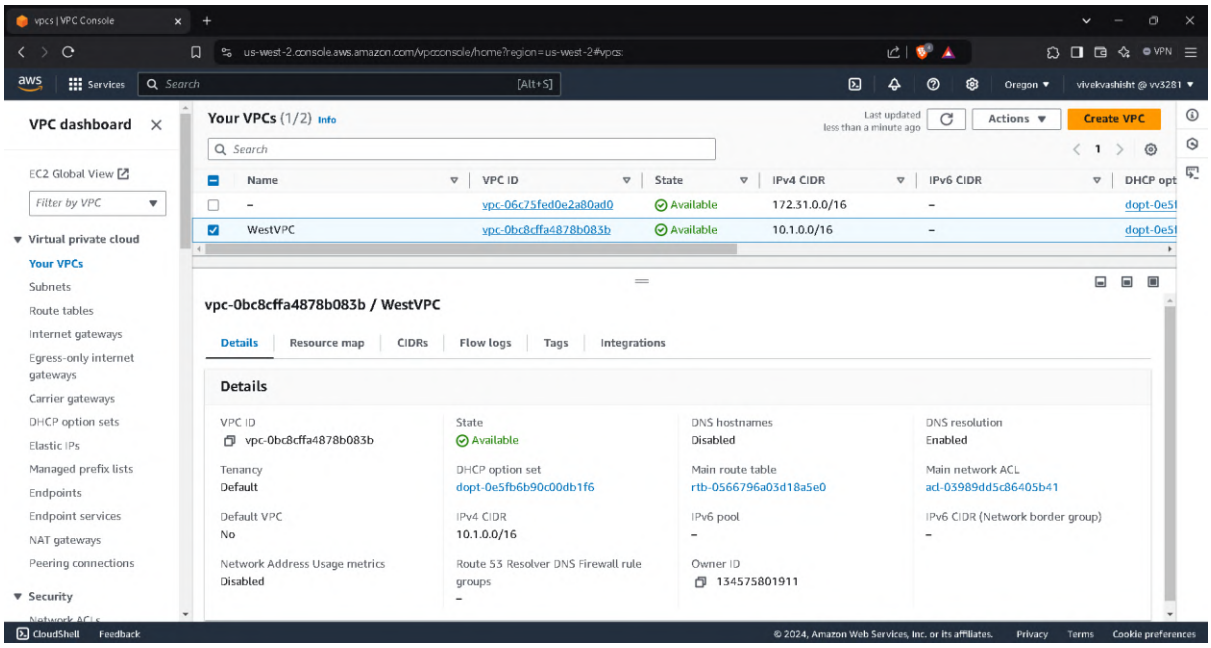


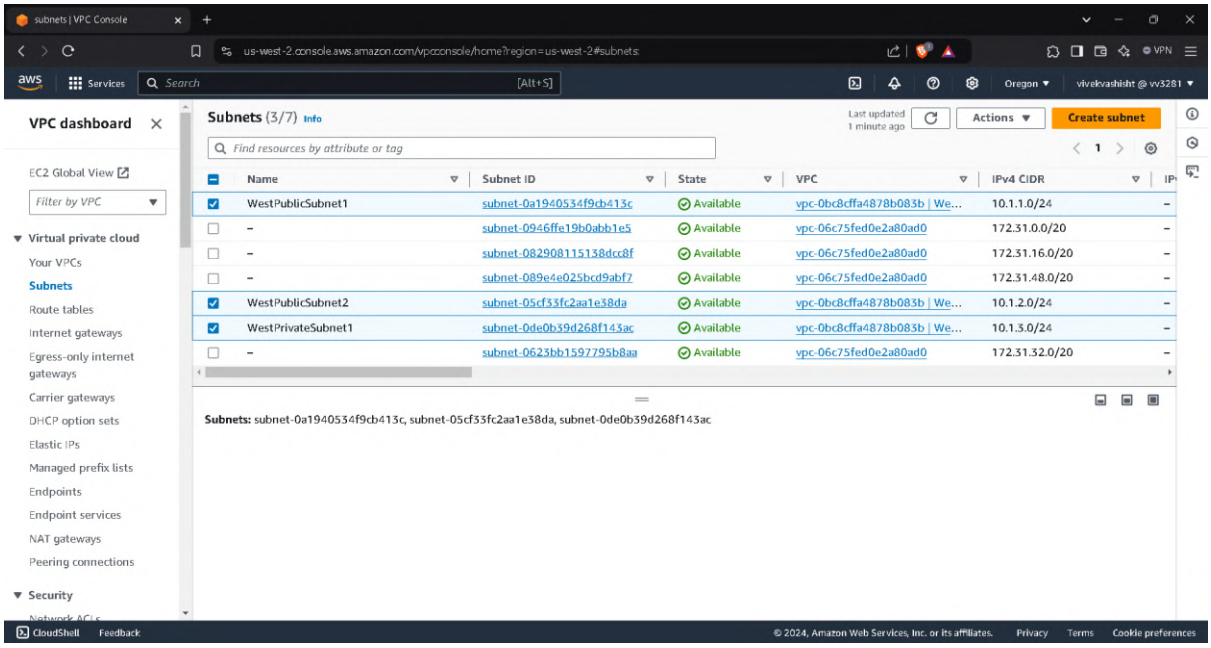
```
Command Prompt

      + owner_id                                     = (known after apply)
      + private_dns_hostname_type_on_launch          = (known after apply)
      + tags                                         = {
          + "Name" = "WestPublicSubnet2"
        }
      + tags_all                                     = {
          + "Name" = "WestPublicSubnet2"
        }
      + vpc_id                                       = (known after apply)
    }

  # aws_vpc.west_vpc will be created
  + resource "aws_vpc" "west_vpc" {
      + arn                                = (known after apply)
      + cidr_block                         = "10.1.0.0/16"
      + default_network_acl_id             = (known after apply)
      + default_route_table_id             = (known after apply)
      + default_security_group_id          = (known after apply)
      + dhcp_options_id                    = (known after apply)
      + enable_dns_hostnames               = (known after apply)
      + enable_dns_support                 = true
      + enable_network_address_usage_metrics = (known after apply)
      + id                                 = (known after apply)
      + instance_tenancy                   = "default"
      + ipv6_association_id                = (known after apply)
      + ipv6_cidr_block                    = (known after apply)
      + ipv6_cidr_block_network_border_group = (known after apply)
      + main_route_table_id                = (known after apply)
      + owner_id                           = (known after apply)
      + tags                               = {
          + "Name" = "WestVPC"
        }
      + tags_all                           = {
          + "Name" = "WestVPC"
        }
    }

Plan: 33 to add, 0 to change, 1 to destroy.
```

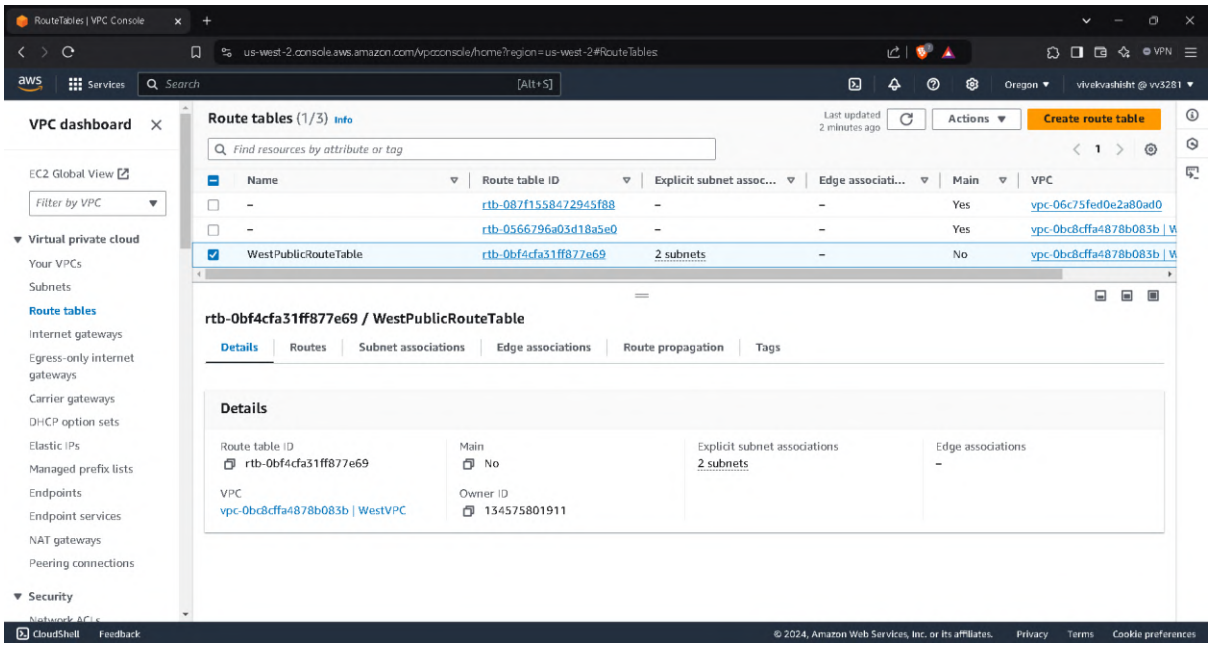

Command Prompt
aws_vpc.west_vpc: Creating...
aws_route_table_association.public_subnet_assoc: Destroying... [id=rtbassoc-080529014332fac87]
aws_route_table_association.public_subnet_assoc_1: Creating...
aws_route53_zone.main: Creating...
aws_subnet.public_subnet_2: Creating...
aws_lb_target_group.primary: Creating...
aws_security_group.app_sg: Creating...
aws_route_table_association.public_subnet_assoc: Destruction complete after 1s
aws_route_table_association.public_subnet_assoc_1: Creation complete after 1s [id=rtbassoc-062c3dc339ec6eb03]
aws_vpc.west_vpc: Creation complete after 1s [id=vpc-0bc8cffa4878b083b]
aws_internet_gateway.west_igw: Creating...
aws_subnet.west_public_subnet_2: Creating...
aws_route_table.west_public_rt: Creating...
aws_lb_target_group.secondary: Creating...
aws_subnet.west_public_subnet_1: Creating...
aws_security_group.west_app_sg: Creating...
aws_lb_target_group.primary: Creation complete after 2s [id=arn:aws:elasticloadbalancing:us-east-1:134575801911:targetgroup/primary-target-group/a9447d338de6a3e5]
aws_subnet.west_private_subnet_1: Creating...
aws_internet_gateway.west_igw: Creation complete after 1s [id=igw-093c80565bfdb459c]
aws_route_table.west_public_rt: Creation complete after 1s [id=rtb-0bf4cfa31ff877e69]
aws_route.west_internet_access: Creating...
aws_subnet.west_private_subnet_1: Creation complete after 0s [id=subnet-0de0b39d268f143ac]
aws_lb_target_group.secondary: Creation complete after 2s [id=arn:aws:elasticloadbalancing:us-west-2:134575801911:targetgroup/secondary-target-group/c46ebba4401c8967]
aws_route.west_internet_access: Creation complete after 1s [id=r-rtb-0bf4cfa31ff877e691080289494]
aws_security_group.app_sg: Creation complete after 3s [id=sg-0d02a28cc779890cb]
aws_launch_template.app: Creating...
aws_launch_template.app: Creation complete after 1s [id=lt-074d209bff44eba96]
aws_security_group.west_app_sg: Creation complete after 3s [id=sg-03d04109977fd80d0]
aws_launch_template.secondary_app: Creating...
aws_launch_template.secondary_app: Creation complete after 0s [id=lt-0c625394b30f2746d]
aws_subnet.public_subnet_2: Still creating... [10s elapsed]
aws_route53_zone.main: Still creating... [10s elapsed]
aws_subnet.west_public_subnet_2: Still creating... [10s elapsed]
aws_subnet.west_public_subnet_1: Still creating... [10s elapsed]
aws_subnet.public_subnet_2: Creation complete after 11s [id=subnet-0d89362f71d4a5adf]
aws_route_table_association.public_subnet_assoc_2: Creating...
aws_lb.primary_alb: Creating...
aws_autoscaling_group.app_asg: Creating...
aws_route_table_association.public_subnet_assoc_2: Creation complete after 1s [id=rtbassoc-0c1e016c8860f95a6]

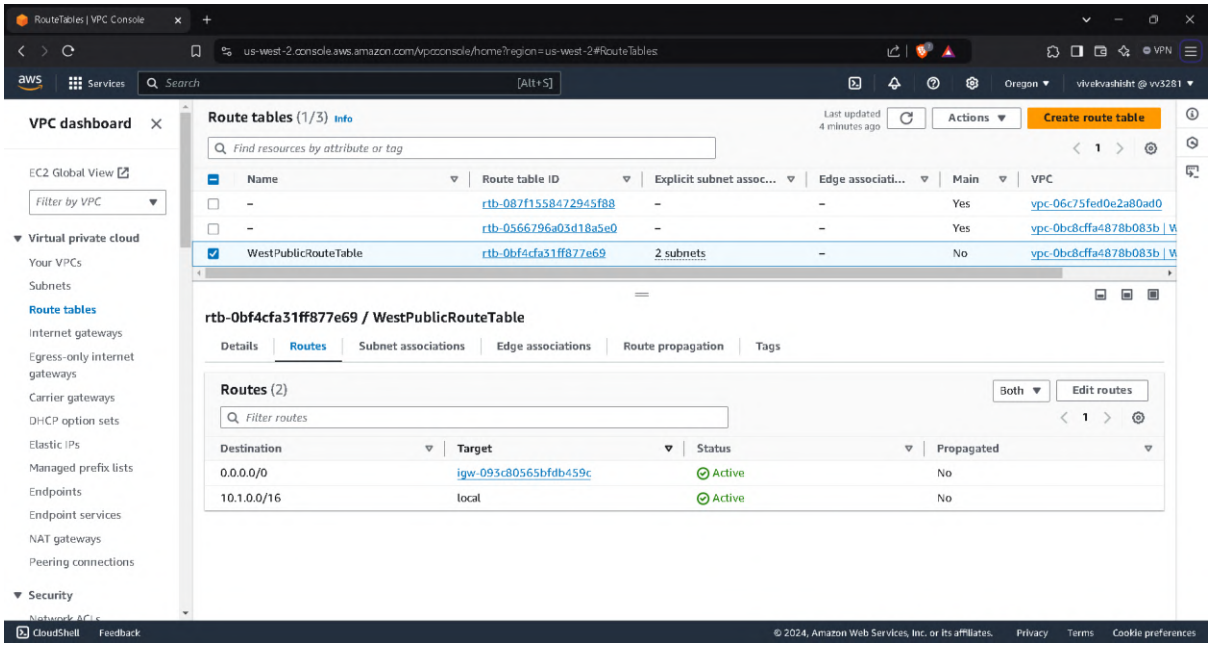

Command Prompt
aws_autoscaling_group.app_asg: Creating...
aws_route_table_association.public_subnet_assoc_2: Creation complete after 1s [id=rtbassoc-0c1e016c8860f95a6]
aws_subnet.west_public_subnet_1: Creation complete after 12s [id=subnet-0a1940534f9cb413c]
aws_route_table_association.west_public_subnet_assoc_1: Creating...
aws_subnet.west_public_subnet_2: Creation complete after 12s [id=subnet-05cf33fc2aa1e38da]
aws_route_table_association.west_public_subnet_assoc_2: Creating...
aws_lb.secondary_alb: Creating...
aws_autoscaling_group.secondary_asg: Creating...
aws_route_table_association.west_public_subnet_assoc_1: Creation complete after 0s [id=rtbassoc-05dc480a71aa04b77]
aws_route_table_association.west_public_subnet_assoc_2: Creation complete after 0s [id=rtbassoc-08e51fdd4ebe4bee8]
aws_route53_zone.main: Still creating... [20s elapsed]
aws_lb.primary_alb: Still creating... [10s elapsed]
aws_autoscaling_group.app_asg: Still creating... [10s elapsed]
aws_autoscaling_group.secondary_asg: Still creating... [10s elapsed]
aws_lb.secondary_alb: Still creating... [10s elapsed]
aws_autoscaling_group.app_asg: Creation complete after 17s [id=terraform-20241030094125457300000004]
aws_autoscaling_group.secondary_asg: Creation complete after 16s [id=terraform-20241030094126584900000004]
aws_autoscaling_policy.scale_up: Creating...
aws_autoscaling_policy.scale_down: Creating...
aws_autoscaling_policy.west_scale_down: Creating...
aws_autoscaling_policy.west_scale_up: Creating...
aws_autoscaling_policy.scale_up: Creation complete after 0s [id=scale-up]
aws_autoscaling_policy.west_scale_down: Creation complete after 0s [id=west-scale-down]
aws_autoscaling_policy.west_scale_up: Creation complete after 0s [id=west-scale-up]
aws_autoscaling_policy.scale_down: Creation complete after 0s [id=scale-down]
aws_route53_zone.main: Still creating... [30s elapsed]
aws_lb.primary_alb: Still creating... [20s elapsed]
aws_lb.secondary_alb: Still creating... [20s elapsed]
aws_route53_zone.main: Still creating... [40s elapsed]
aws_lb.primary_alb: Still creating... [30s elapsed]
aws_lb.secondary_alb: Still creating... [30s elapsed]
aws_route53_zone.main: Creation complete after 45s [id=Z0743615150XJV7STD1T]
aws_lb.primary_alb: Still creating... [40s elapsed]
aws_lb.secondary_alb: Still creating... [40s elapsed]
aws_lb.primary_alb: Still creating... [50s elapsed]
aws_lb.secondary_alb: Still creating... [50s elapsed]
aws_lb.primary_alb: Still creating... [1m1s elapsed]
aws_lb.secondary_alb: Still creating... [1m0s elapsed]
aws_lb.primary_alb: Still creating... [1m11s elapsed]
aws_lb.secondary_alb: Still creating... [1m10s elapsed]
aws_lb.primary_alb: Still creating... [1m21s elapsed]

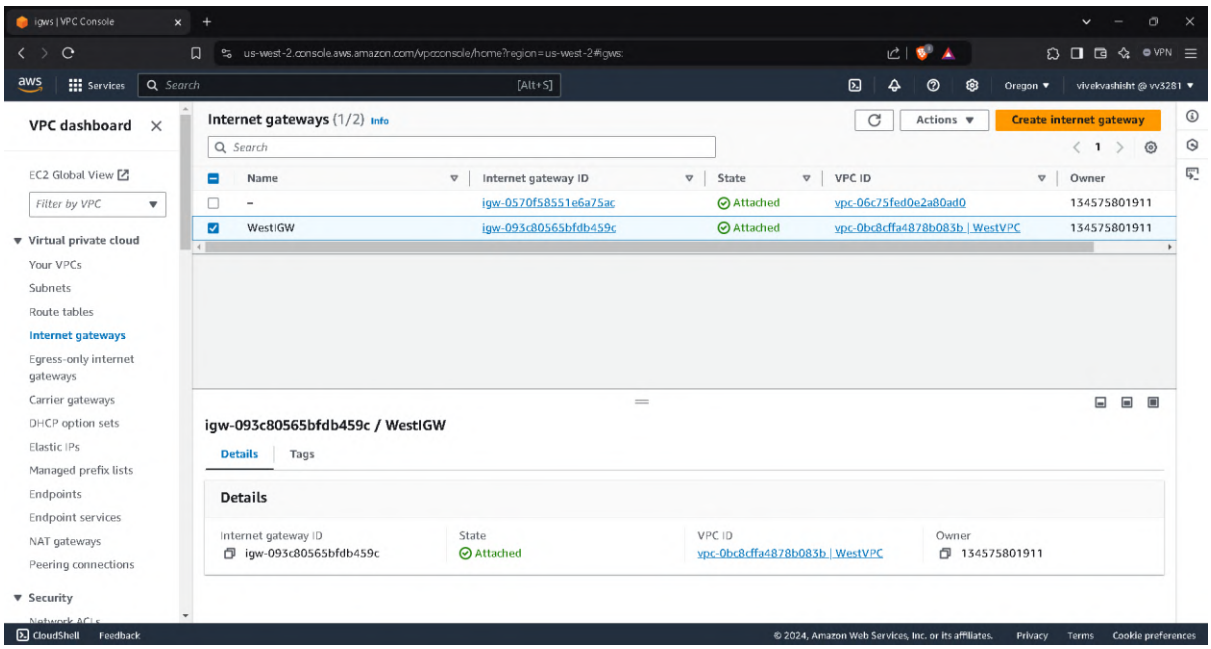

Command Prompt
aws_lb.secondary_alb: Still creating... [2m10s elapsed]
aws_lb.primary_alb: Still creating... [2m21s elapsed]
aws_lb.secondary_alb: Still creating... [2m20s elapsed]
aws_lb.primary_alb: Still creating... [2m31s elapsed]
aws_lb.secondary_alb: Still creating... [2m30s elapsed]
aws_lb.primary_alb: Still creating... [2m41s elapsed]
aws_lb.secondary_alb: Still creating... [2m40s elapsed]
aws_lb.primary_alb: Still creating... [2m51s elapsed]
aws_lb.secondary_alb: Still creating... [2m50s elapsed]
aws_lb.primary_alb: Still creating... [3m1s elapsed]
aws_lb.secondary_alb: Still creating... [3m0s elapsed]
aws_lb.primary_alb: Still creating... [3m11s elapsed]
aws_lb.secondary_alb: Still creating... [3m10s elapsed]
aws_lb.secondary_alb: Creation complete after 3m12s [id=arn:aws:elasticloadbalancing:us-west-2:134575801911:loadbalancer/app/secondary-alb/4b7a0a637b386685]
aws_lb.primary_alb: Creation complete after 3m15s [id=arn:aws:elasticloadbalancing:us-east-1:134575801911:loadbalancer/app/primary-alb/55f8876b41516ba9]
aws_route53_record.secondary: Creating...
aws_route53_health_check.primary_health_check: Creating...
aws_route53_record.primary: Creating...
aws_route53_health_check.secondary_health_check: Creating...
aws_lb_listener.secondary_http: Creating...
aws_lb_listener.http: Creating...
aws_lb_listener.secondary_http: Creation complete after 0s [id=arn:aws:elasticloadbalancing:us-west-2:134575801911:listener/app/secondary-alb/4b7a0a637b386685/938584833b8318cd]
aws_lb_listener.http: Creation complete after 0s [id=arn:aws:elasticloadbalancing:us-east-1:134575801911:listener/app/primary-alb/55f8876b41516ba9/4f2d5bb1d8ba2df1]
aws_route53_health_check.primary_health_check: Creation complete after 0s [id=227ecc39-8273-4708-bca6-befef7ba7e1a]
aws_route53_health_check.secondary_health_check: Creation complete after 0s [id=5ece7a87-32b1-4d77-9ef1-e8cd7faad582]
aws_route53_record.primary: Still creating... [10s elapsed]
aws_route53_record.secondary: Still creating... [10s elapsed]
aws_route53_record.primary: Still creating... [20s elapsed]
aws_route53_record.secondary: Still creating... [20s elapsed]
aws_route53_record.secondary: Still creating... [30s elapsed]
aws_route53_record.primary: Still creating... [30s elapsed]
aws_route53_record.secondary: Still creating... [40s elapsed]
aws_route53_record.primary: Still creating... [40s elapsed]
aws_route53_record.primary: Creation complete after 43s [id=Z0743615150XJV7STD1T_app.myinfra.local_A_Primary]
aws_route53_record.secondary: Creation complete after 44s [id=Z0743615150XJV7STD1T_app.myinfra.local_A_Secondary]
Apply complete! Resources: 33 added, 0 changed, 1 destroyed.
C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>

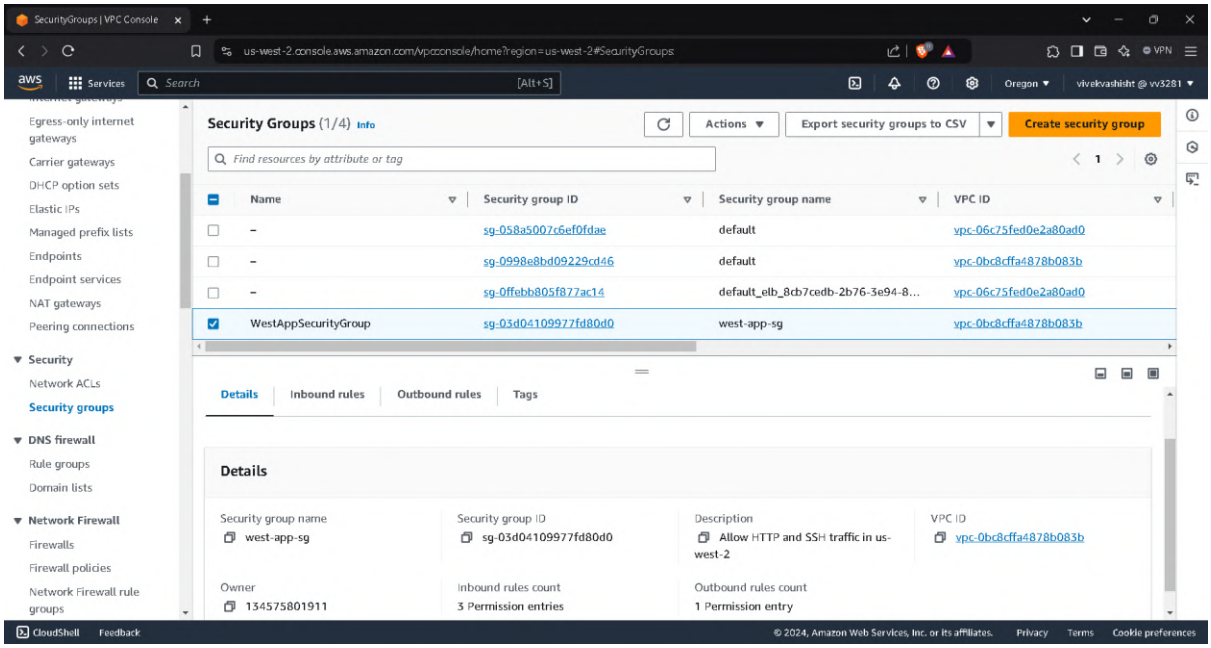

vpcs | VPC Console
us-west-2.console.aws.amazon.com/vpcconsole/home?region=us-west-2#vpcs:
VPC dashboard
Your VPCs (1/2) Info
Last updated less than a minute ago
Actions
Create VPC
Search
Name
VPC ID
State
IPv4 CIDR
IPv6 CIDR
DHCP opt
– vpc-06c75fed0e2a80ad0 Available 172.31.0.0/16 –
WestVPC vpc-0bc8cffa4878b083b Available 10.1.0.0/16 –
vpc-0bc8cffa4878b083b / WestVPC
Details
Resource map
CIDRs
Flow logs
Tags
Integrations
VPC ID vpc-0bc8cffa4878b083b
State Available
DNS hostnames Disabled
DNS resolution Enabled
Tenancy Default
DHCP option set dopt-0e5fb6b90c00db1f6
Main route table rtb-0566796a03d18a5e0
Main network ACL acl-03989dd5c86405b41
Default VPC No
IPv4 CIDR 10.1.0.0/16
IPv6 pool –
IPv6 CIDR (Network border group) –
Network Address Usage metrics Disabled
Route 53 Resolver DNS Firewall rule groups –
Owner ID 134575801911
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

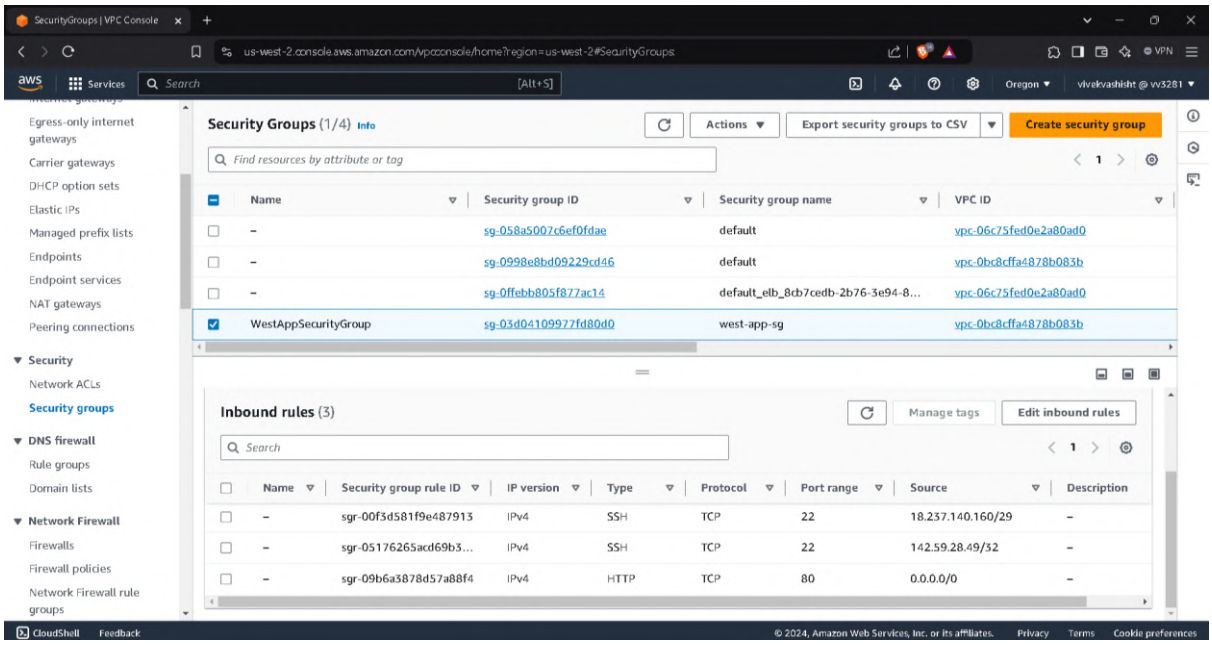

subnets | VPC Console
us-west-2.console.aws.amazon.com/vpcconsole/home?region=us-west-2#subnets:
VPC dashboard
Subnets (3/7) Info
Last updated 1 minute ago
Actions
Create subnet
Find resources by attribute or tag
Name
Subnet ID
State
VPC
IPv4 CIDR
WestPublicSubnet1 subnet-0a1940534f9cb413c Available vpc-0bc8cffa4878b083b | We... 10.1.1.0/24
– subnet-0946ffe19b0abb1e5 Available vpc-06c75fed0e2a80ad0 172.31.0.0/20
– subnet-082908115138dcc8f Available vpc-06c75fed0e2a80ad0 172.31.16.0/20
– subnet-089e4e025bcd9abf7 Available vpc-06c75fed0e2a80ad0 172.31.48.0/20
WestPublicSubnet2 subnet-05cf33fc2aa1e38da Available vpc-0bc8cffa4878b083b | We... 10.1.2.0/24
WestPrivateSubnet1 subnet-0de0b39d268f143ac Available vpc-0bc8cffa4878b083b | We... 10.1.3.0/24
– subnet-0623bb1597795b8aa Available vpc-06c75fed0e2a80ad0 172.31.32.0/20
Subnets: subnet-0a1940534f9cb413c, subnet-05cf33fc2aa1e38da, subnet-0de0b39d268f143ac
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

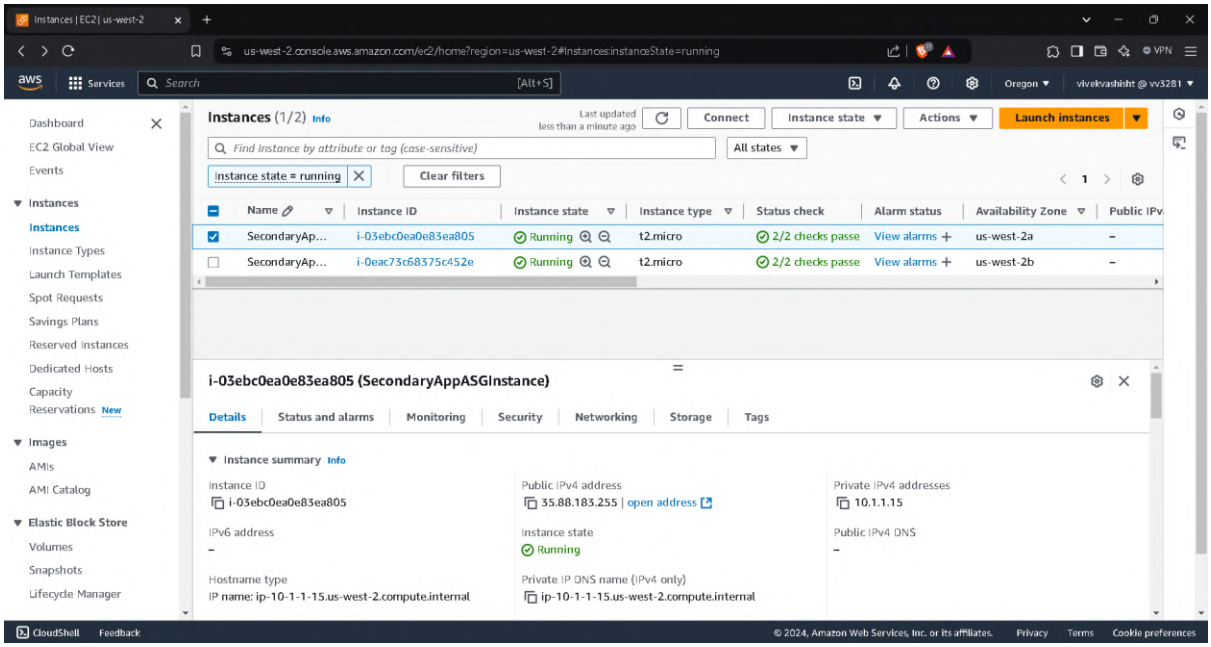

RouteTables | VPC Console
us-west-2.console.aws.amazon.com/vpcconsole/home?region=us-west-2#RouteTables:
VPC dashboard
Route tables (1/3) Info
Last updated 2 minutes ago
Actions
Create route table
Find resources by attribute or tag
Name
Route table ID
Explicit subnet assoc...
Edge associati...
Main
VPC
– rtb-087f155847294 5f88 – – Yes vpc-06c75fed0e2a80ad0
– rtb-0566796a03d18a5e0 – – Yes vpc-0bc8cffa4878b083b | W
WestPublicRouteTable rtb-0bf4cfa31ff877e69 2 subnets – No vpc-0bc8cffa4878b083b | W
rtb-0bf4cfa31ff877e69 / WestPublicRouteTable
Details
Routes
Subnet associations
Edge associations
Route propagation
Tags
Route table ID rtb-0bf4cfa31ff877e69
Main No
Explicit subnet associations 2 subnets
Edge associations –
VPC vpc-0bc8cffa4878b083b | WestVPC
Owner ID 134575801911
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

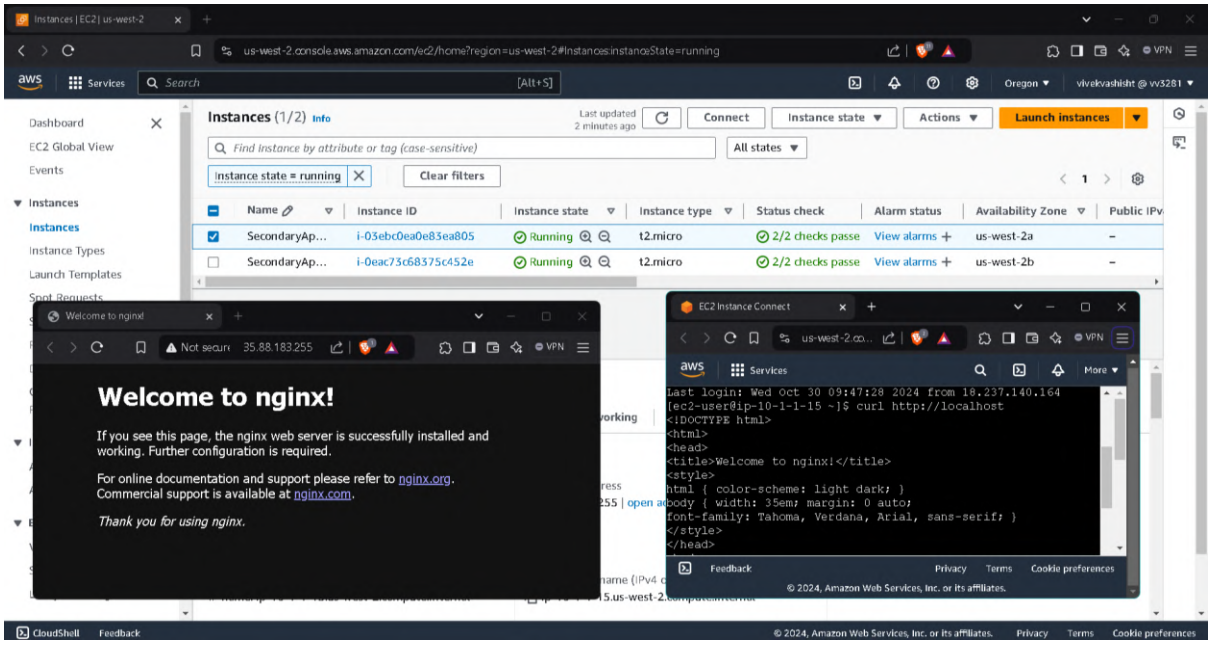

RouteTables | VPC Console
us-west-2.console.aws.amazon.com/vpcconsole/home?region=us-west-2#RouteTables:
VPC dashboard
Route tables (1/3) Info
Last updated 4 minutes ago
Actions
Create route table
Find resources by attribute or tag
Name
Route table ID
Explicit subnet assoc...
Edge associati...
Main
VPC
– rtb-087f1558472945f88 – – Yes vpc-06c75fed0e2a80ad0
– rtb-0566796a03d18a5e0 – – Yes vpc-0bc8cffa4878b083b | W
WestPublicRouteTable rtb-0bf4cfa31ff877e69 2 subnets – No vpc-0bc8cffa4878b083b | W
rtb-0bf4cfa31ff877e69 / WestPublicRouteTable
Details
Routes
Subnet associations
Edge associations
Route propagation
Tags
Routes (2)
Both
Edit routes
Filter routes
Destination
Target
Status
Propagated
0.0.0.0/0 igw-093c80565bfdb459c Active No
10.1.0.0/16 local Active No
Virtual private cloud
Your VPCs
Subnets
Route tables
Internet gateways
Egress-only internet gateways
Carrier gateways
DHCP option sets
Elastic IPs
Managed prefix lists
Endpoints
Endpoint services
NAT gateways
Peering connections
Security
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

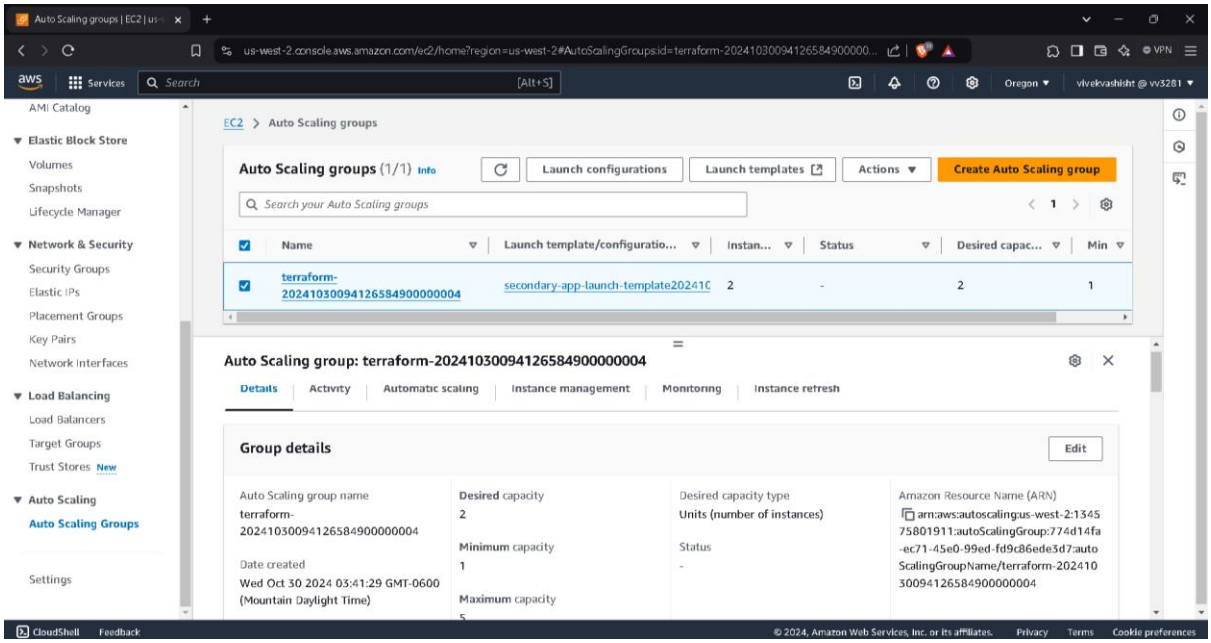

igws | VPC Console
us-west-2.console.aws.amazon.com/vpcconsole/home?region=us-west-2#igws:
VPC dashboard
Internet gateways (1/2) Info
Actions
Create internet gateway
Search
Name
Internet gateway ID
State
VPC ID
Owner
– igw-0570f58551e6a75ac Attached vpc-06c75fed0e2a80ad0 134575801911
WestIGW igw-093c80565bfdb459c Attached vpc-0bc8cffa4878b083b | WestVPC 134575801911
igw-093c80565bfdb459c / WestIGW
Details
Tags
Details
Internet gateway ID
igw-093c80565bfdb459c
State
Attached
VPC ID
vpc-0bc8cffa4878b083b | WestVPC
Owner
134575801911
Virtual private cloud
Your VPCs
Subnets
Route tables
Internet gateways
Egress-only internet gateways
Carrier gateways
DHCP option sets
Elastic IPs
Managed prefix lists
Endpoints
Endpoint services
NAT gateways
Peering connections
Security
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

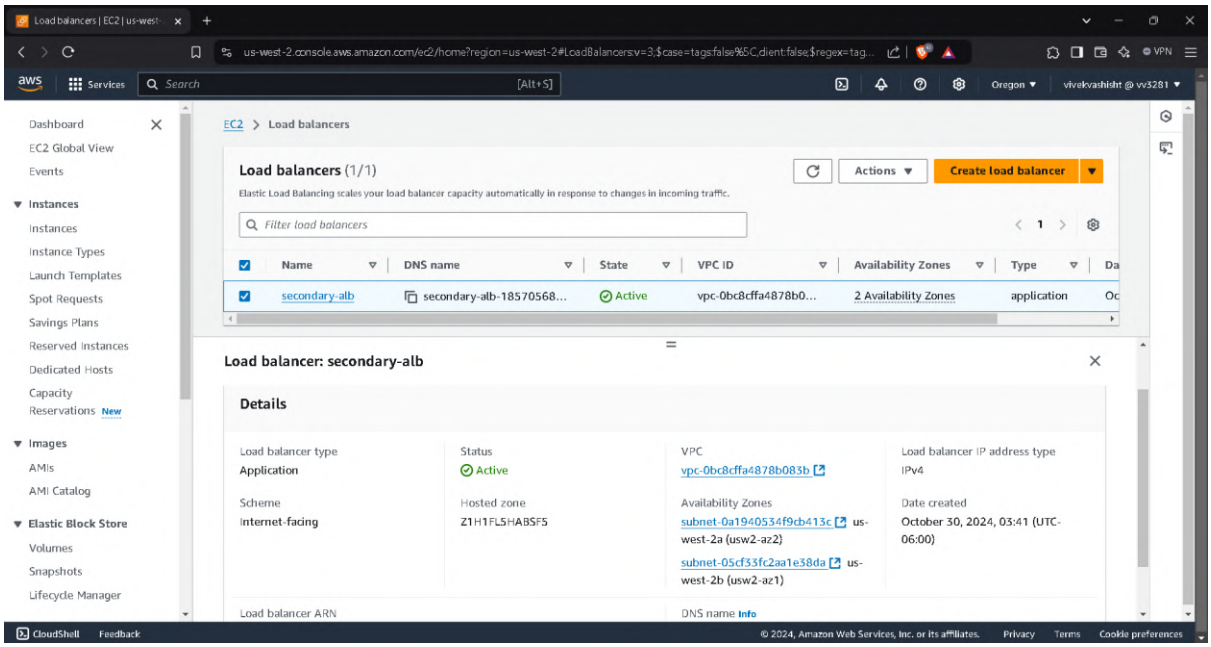

SecurityGroups | VPC Console
us-west-2.console.aws.amazon.com/vpcconsole/home?region=us-west-2#SecurityGroups:
Security Groups (1/4) Info
Actions
Export security groups to CSV
Create security group
Find resources by attribute or tag
Name
Security group ID
Security group name
VPC ID
– sg-058a5007c6ef0fdae default vpc-06c75fed0e2a80ad0
– sg-0998e8bd09229cd46 default vpc-0bc8cffa4878b083b
– sg-0ffebb805f877ac14 default_elb_8cb7cedb-2b76-3e94-8... vpc-06c75fed0e2a80ad0
WestAppSecurityGroup sg-03d04109977fd80d0 west-app-sg vpc-0bc8cffa4878b083b
Details
Inbound rules
Outbound rules
Tags
Details
Security group name
west-app-sg
Security group ID
sg-03d04109977fd80d0
Description
Allow HTTP and SSH traffic in us-west-2
VPC ID
vpc-0bc8cffa4878b083b
Owner
134575801911
Inbound rules count
3 Permission entries
Outbound rules count
1 Permission entry
Egress-only internet gateways
Carrier gateways
DHCP option sets
Elastic IPs
Managed prefix lists
Endpoints
Endpoint services
NAT gateways
Peering connections
Security
Network ACLs
Security groups
DNS firewall
Rule groups
Domain lists
Network Firewall
Firewalls
Firewall policies
Network Firewall rule groups
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

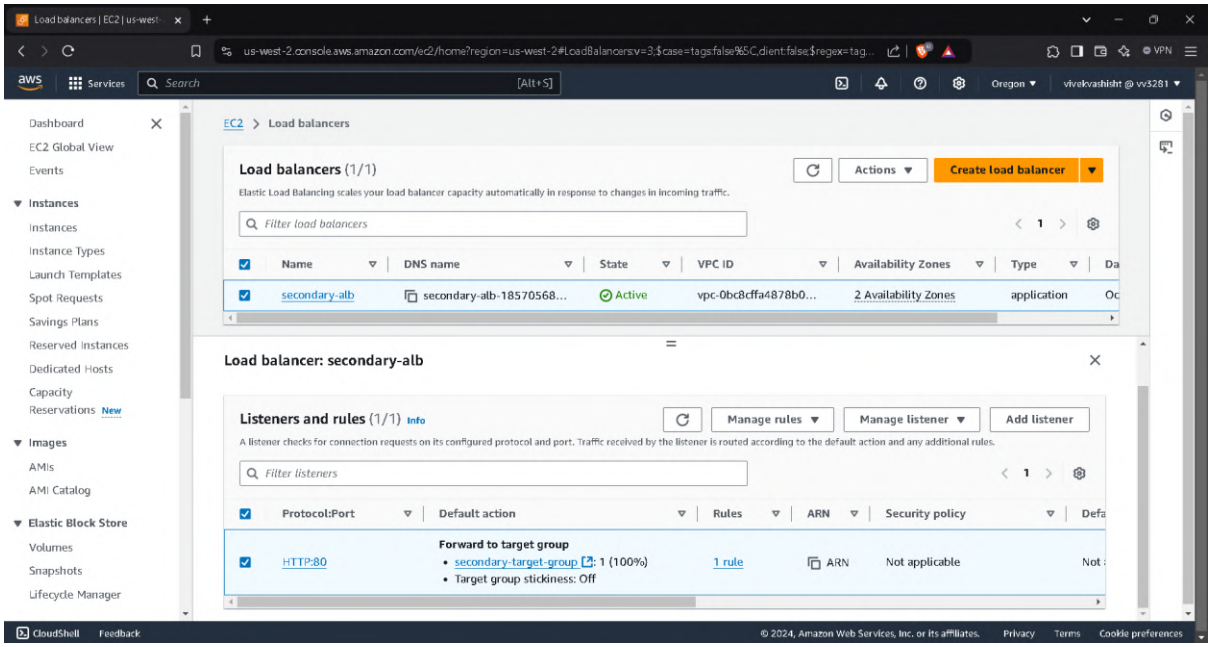

SecurityGroups | VPC Console
us-west-2.console.aws.amazon.com/vpcconsole/home?region=us-west-2#SecurityGroups:
Security Groups (1/4) Info
Actions
Export security groups to CSV
Create security group
Find resources by attribute or tag
Name
Security group ID
Security group name
VPC ID
sg-058a5007c6ef0fdae default vpc-06c75fed0e2a80ad0
sg-0998e8bd09229cd46 default vpc-0bc8cffa4878b083b
sg-0ffebb805f877ac14 default_elb_8cb7cedb-2b76-3e94-8... vpc-06c75fed0e2a80ad0
WestAppSecurityGroup sg-03d04109977fd80d0 west-app-sg vpc-0bc8cffa4878b083b
Inbound rules (3)
Manage tags
Edit inbound rules
Search
Name
Security group rule ID
IP version
Type
Protocol
Port range
Source
Description
sgr-00f3d581f9e487913 IPv4 SSH TCP 22 18.237.140.160/29
sgr-05176265acd69b3... IPv4 SSH TCP 22 142.59.28.49/32
sgr-09b6a3878d57a88f4 IPv4 HTTP TCP 80 0.0.0.0/0
Egress-only internet gateways
Carrier gateways
DHCP option sets
Elastic IPs
Managed prefix lists
Endpoints
Endpoint services
NAT gateways
Peering connections
Security
Network ACLs
Security groups
DNS firewall
Rule groups
Domain lists
Network Firewall
Firewalls
Firewall policies
Network Firewall rule groups
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

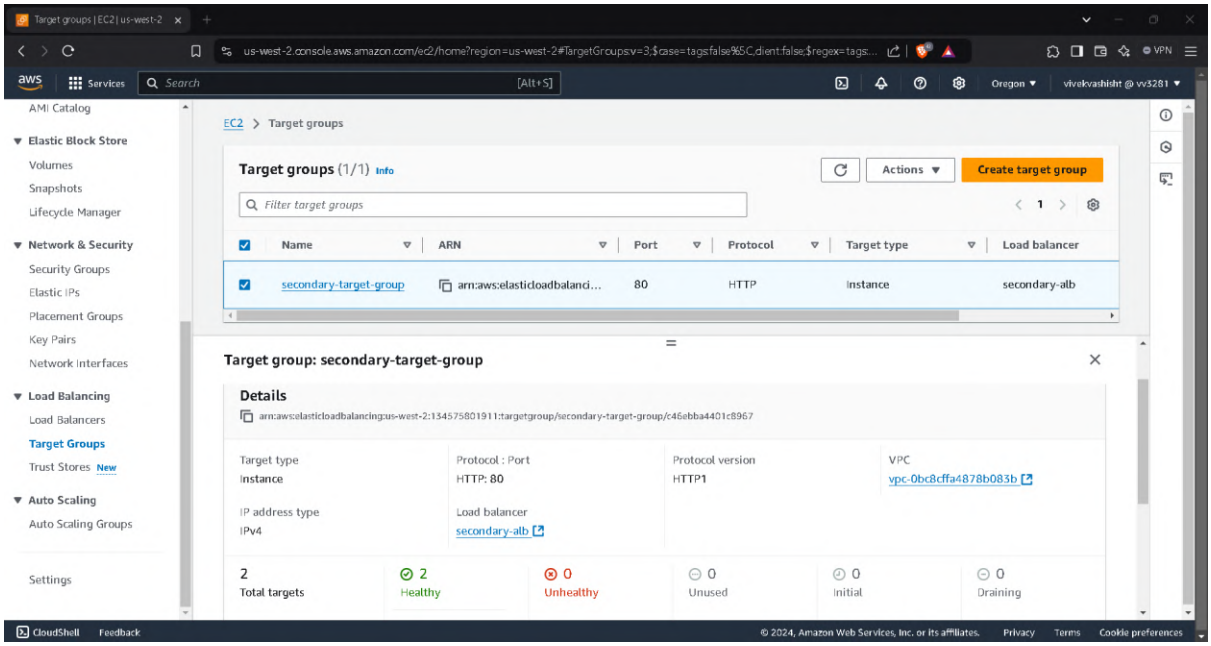

Instances | EC2 | us-west-2
us-west-2.console.aws.amazon.com/ec2/home?region=us-west-2#Instances:instanceState=running
Instances (1/2) Info
Last updated less than a minute ago
Connect
Instance state
Actions
Launch instances
Find instance by attribute or tag (case-sensitive)
All states
Instance state = running
Clear filters
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
Availability Zone
Public IPv
SecondaryAp... i-03ebc0ea0e83ea805 Running t2.micro 2/2 checks passe View alarms us-west-2a
SecondaryAp... i-0eac73c68375c452e Running t2.micro 2/2 checks passe View alarms us-west-2b
i-03ebc0ea0e83ea805 (SecondaryAppASGInstance)
Details
Status and alarms
Monitoring
Security
Networking
Storage
Tags
Instance summary Info
Instance ID
i-03ebc0ea0e83ea805
Public IPv4 address
35.88.183.255 | open address
Private IPv4 addresses
10.1.1.15
IPv6 address
-
Instance state
Running
Public IPv4 DNS
-
Hostname type
IP name: ip-10-1-1-15.us-west-2.compute.internal
Private IP DNS name (IPv4 only)
ip-10-1-1-15.us-west-2.compute.internal
Dashboard
EC2 Global View
Events
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations New
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

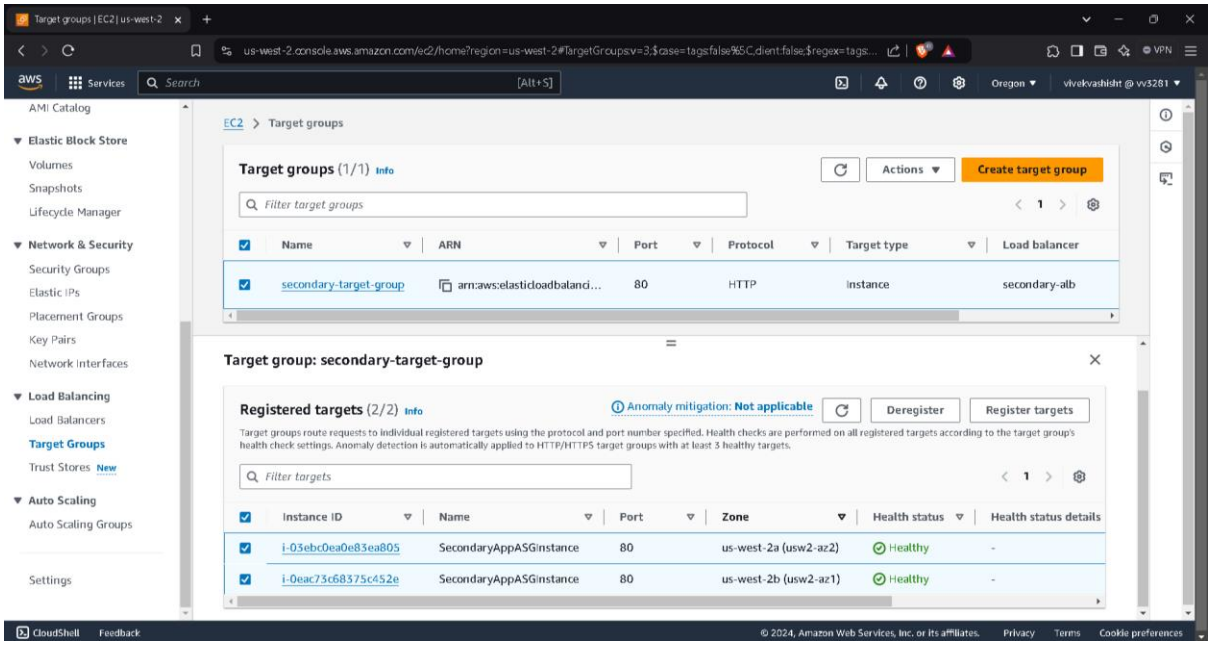

Instances | EC2 | us-west-2
Instances (1/2) Info
Last updated 2 minutes ago
Welcome to nginx!
Not secure 35.88.183.255
Welcome to nginx!
If you see this page, the nginx web server is successfully installed and working. Further configuration is required.
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.
Thank you for using nginx.
EC2 Instance Connect
us-west-2.co...
Last login: Wed Oct 30 09:47:28 2024 from 18.237.140.164
[ec2-user@ip-10-1-1-15 ~]$ curl http://localhost
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
Feedback
Privacy
Terms
Cookie preferences
© 2024, Amazon Web Services, Inc. or its affiliates.
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

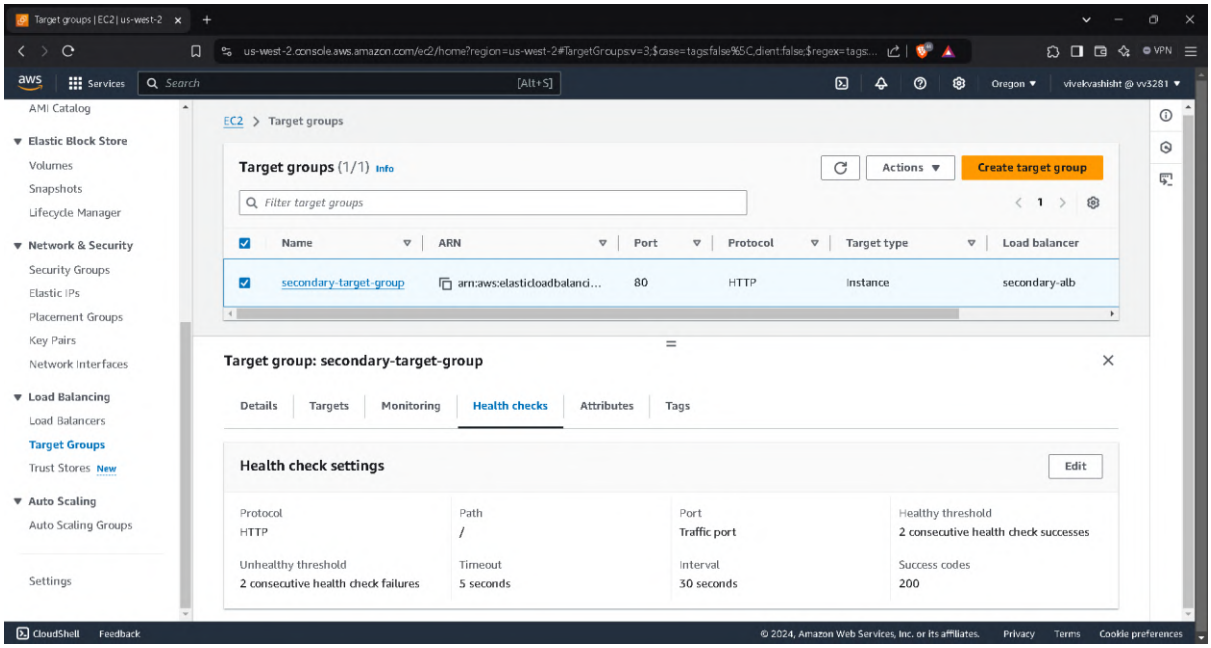

Auto Scaling groups | EC2 | us-
us-west-2.console.aws.amazon.com/ec2/home?region=us-west-2#AutoScalingGroups:id=terraform-20241030094126584900000...
Services
Search
[Alt+S]
Oregon
vivekvashisht @ vv3281
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Network & Security
Security Groups
Elastic IPs
Placement Groups
Key Pairs
Network Interfaces
Load Balancing
Load Balancers
Target Groups
Trust Stores New
Auto Scaling
Auto Scaling Groups
Settings
EC2 > Auto Scaling groups
Auto Scaling groups (1/1) Info
Launch configurations
Launch templates
Actions
Create Auto Scaling group
Search your Auto Scaling groups
Name
Launch template/configuratio...
Instan...
Status
Desired capac...
Min
terraform-20241030094126584900000004
secondary-app-launch-template20241(
2
-
2
1
Auto Scaling group: terraform-20241030094126584900000004
Details
Activity
Automatic scaling
Instance management
Monitoring
Instance refresh
Group details
Edit
Auto Scaling group name
terraform-20241030094126584900000004
Desired capacity
2
Minimum capacity
1
Maximum capacity
5
Desired capacity type
Units (number of instances)
Status
-
Amazon Resource Name (ARN)
arn:aws:autoscaling:us-west-2:134575801911:autoScalingGroup:774d14fa-ec71-45e0-99ed-fd9c86ede3d7:autoScalingGroupName/terraform-20241030094126584900000004
Date created
Wed Oct 30 2024 03:41:29 GMT-0600 (Mountain Daylight Time)
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

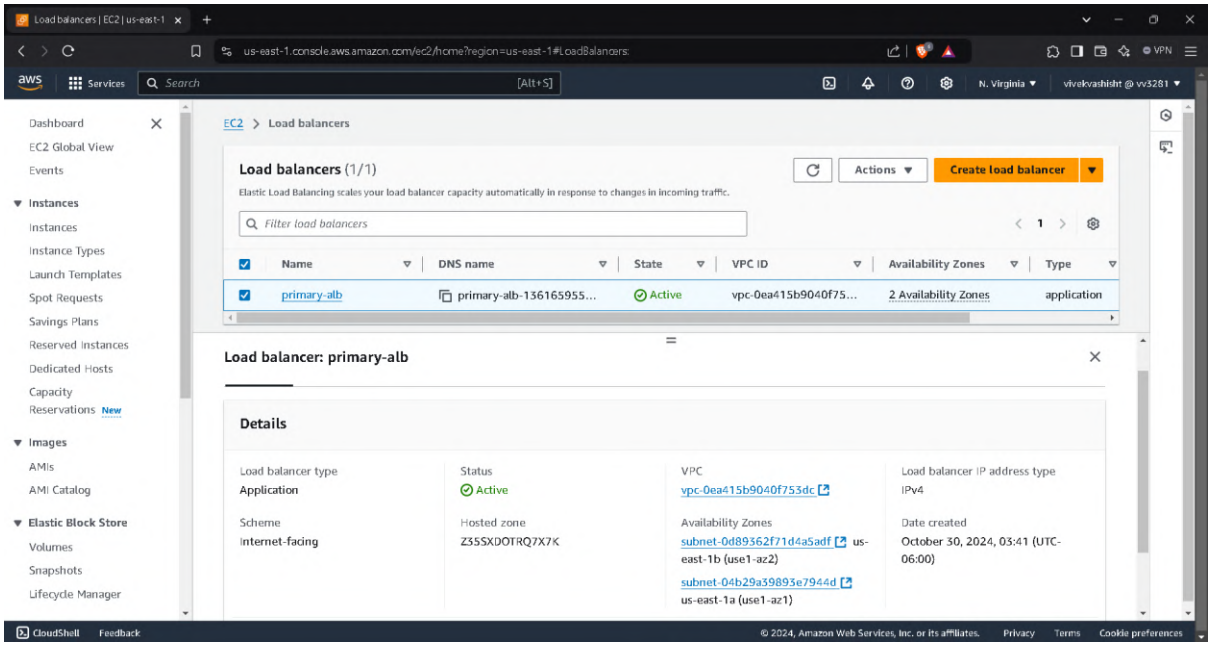

Load balancers | EC2 | us-west-
us-west-2.console.aws.amazon.com/ec2/home?region=us-west-2#LoadBalancers:v=3;$case=tags:false%5C,client:false;$regex=tag...
Services
Search
[Alt+S]
Oregon
vivekvashisht @ vv3281
Dashboard
EC2 Global View
Events
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations New
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
EC2 > Load balancers
Load balancers (1/1)
Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.
Actions
Create load balancer
Filter load balancers
Name
DNS name
State
VPC ID
Availability Zones
Type
secondary-alb
secondary-alb-18570568...
Active
vpc-0bc8cffa4878b0...
2 Availability Zones
application
Load balancer: secondary-alb
Details
Load balancer type
Application
Status
Active
VPC
vpc-0bc8cffa4878b083b
Load balancer IP address type
IPv4
Scheme
Internet-facing
Hosted zone
Z1H1FL5HABSF5
Availability Zones
subnet-0a1940534f9cb413c us-west-2a (usw2-az2)
subnet-05cf33fc2aa1e38da us-west-2b (usw2-az1)
Date created
October 30, 2024, 03:41 (UTC-06:00)
Load balancer ARN
DNS name Info
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

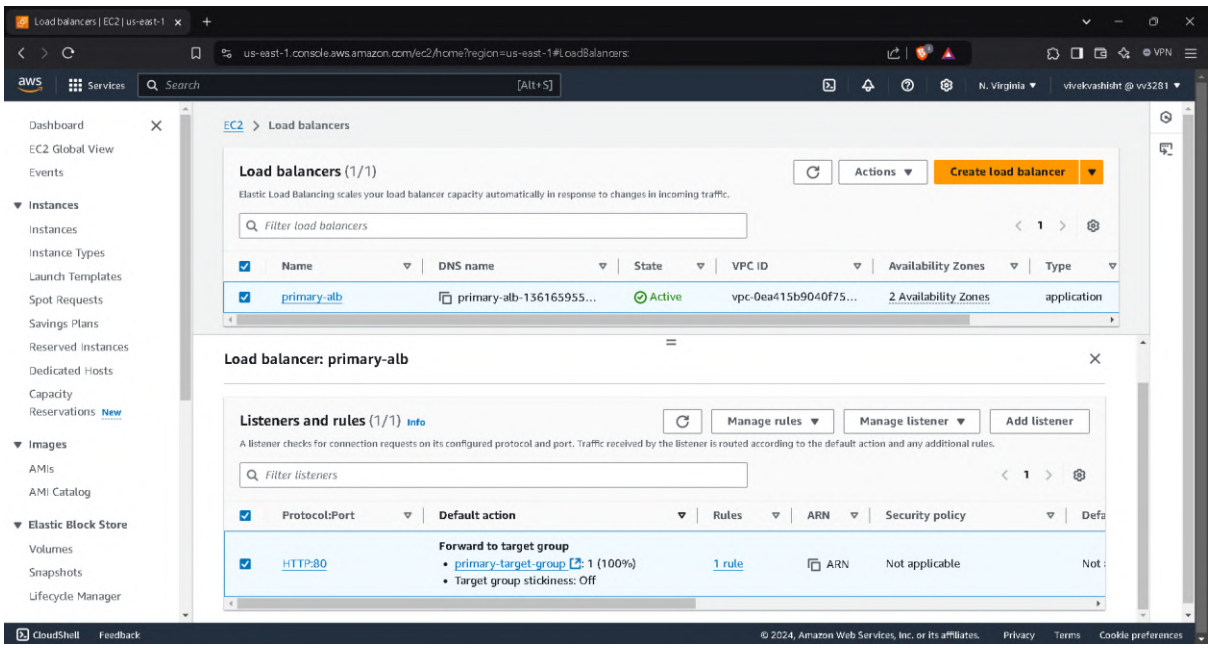

Load balancers | EC2 | us-west-
us-west-2.console.aws.amazon.com/ec2/home?region=us-west-2#LoadBalancers:v=3;$case=tags:false%5C,client:false;$regex=tag...
Services
Search
[Alt+S]
Oregon
vivekvashisht @ vv3281
Dashboard
EC2 Global View
Events
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations New
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
EC2 > Load balancers
Load balancers (1/1)
Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.
Actions
Create load balancer
Filter load balancers
Name
DNS name
State
VPC ID
Availability Zones
Type
secondary-alb
secondary-alb-18570568...
Active
vpc-0bc8cffa4878b0...
2 Availability Zones
application
Load balancer: secondary-alb
Listeners and rules (1/1) Info
Manage rules
Manage listener
Add listener
A listener checks for connection requests on its configured protocol and port. Traffic received by the listener is routed according to the default action and any additional rules.
Filter listeners
Protocol:Port
Default action
Rules
ARN
Security policy
HTTP:80
Forward to target group
secondary-target-group: 1 (100%)
Target group stickiness: Off
1 rule
ARN
Not applicable
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

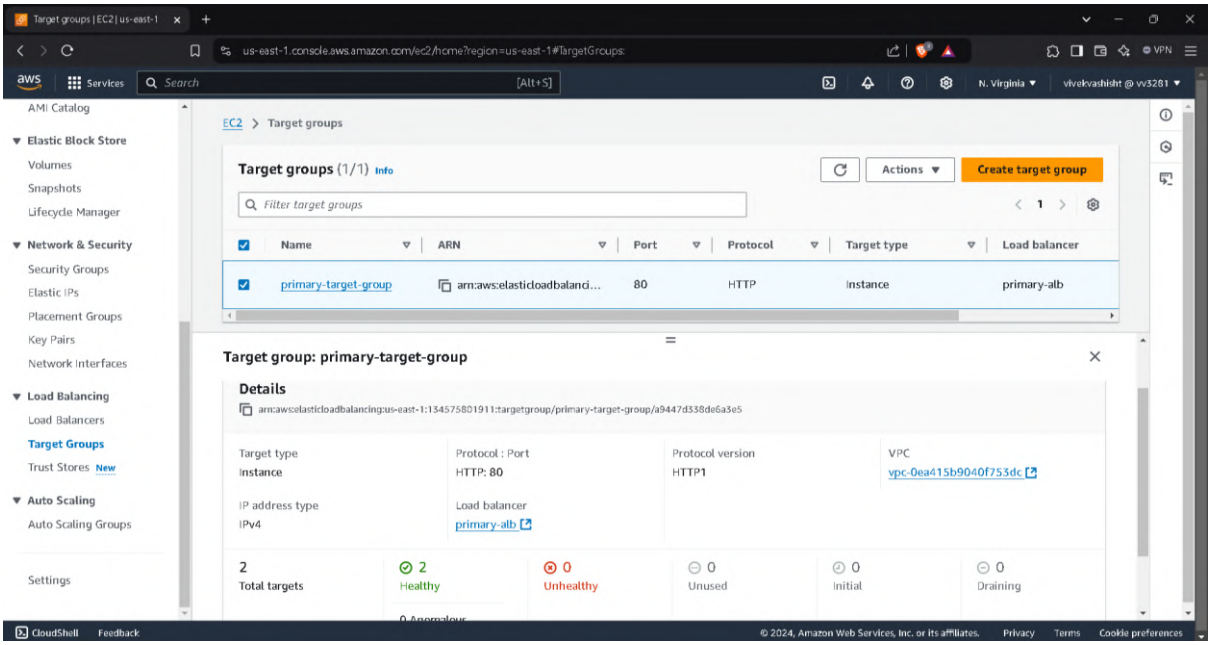

Target groups | EC2 | us-west-2
us-west-2.console.aws.amazon.com/ec2/home?region=us-west-2#TargetGroups:v=3;$case=tags:false%5C,client:false;$regex=tags:...
Services
Search
[Alt+S]
Oregon
vivekvashisht @ vv3281
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Network & Security
Security Groups
Elastic IPs
Placement Groups
Key Pairs
Network Interfaces
Load Balancing
Load Balancers
Target Groups
Trust Stores New
Auto Scaling
Auto Scaling Groups
Settings
EC2 > Target groups
Target groups (1/1) Info
Actions
Create target group
Filter target groups
1
Name
ARN
Port
Protocol
Target type
Load balancer
secondary-target-group
arn:aws:elasticloadbalanci...
80
HTTP
instance
secondary-alb
Target group: secondary-target-group
Details
arn:aws:elasticloadbalancing:us-west-2:134575801911:targetgroup/secondary-target-group/c46ebba4401c8967
Target type
Instance
Protocol : Port
HTTP: 80
Protocol version
HTTP1
VPC
vpc-0bc8cffa4878b083b
IP address type
IPv4
Load balancer
secondary-alb
2
Total targets
2
Healthy
0
Unhealthy
0
Unused
0
Initial
0
Draining
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

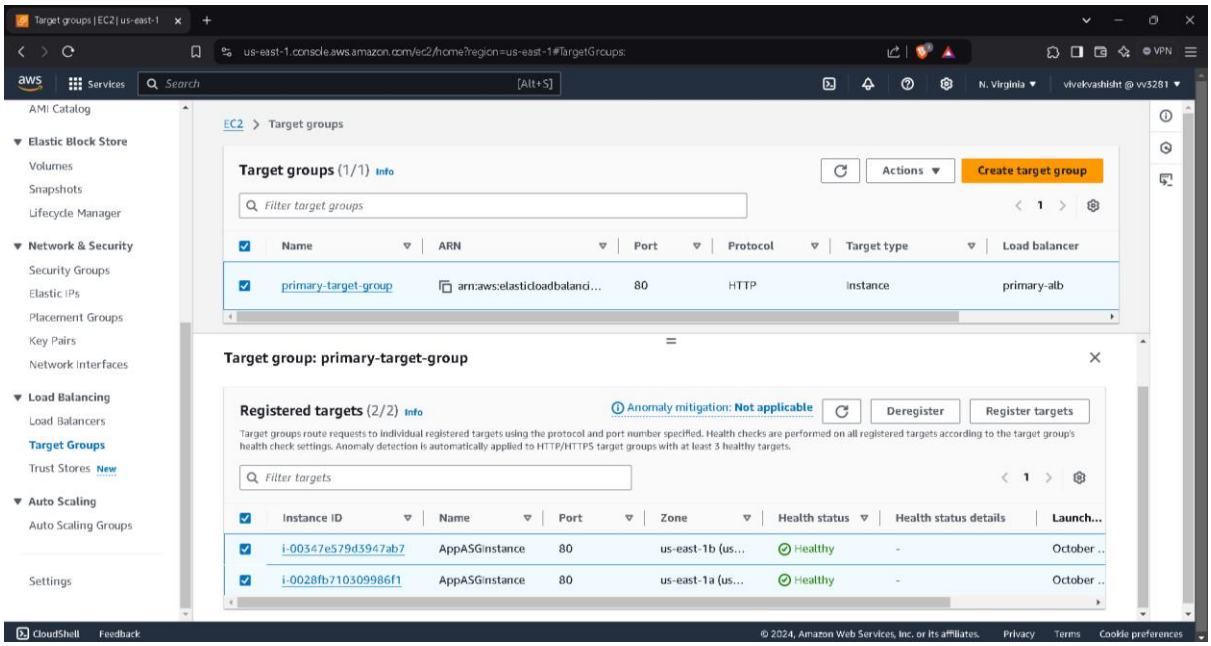

Target groups | EC2 | us-west-2
us-west-2.console.aws.amazon.com/ec2/home?region=us-west-2#TargetGroups:v=3;$case=tags:false%5C,client:false;$regex=tags:...
Services
Search
[Alt+S]
Oregon
vivekvashisht @ vv3281
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Network & Security
Security Groups
Elastic IPs
Placement Groups
Key Pairs
Network Interfaces
Load Balancing
Load Balancers
Target Groups
Trust Stores New
Auto Scaling
Auto Scaling Groups
Settings
EC2 > Target groups
Target groups (1/1) Info
Actions
Create target group
Filter target groups
1
Name
ARN
Port
Protocol
Target type
Load balancer
secondary-target-group
arn:aws:elasticloadbalanci...
80
HTTP
instance
secondary-alb
Target group: secondary-target-group
Registered targets (2/2) Info
Anomaly mitigation: Not applicable
Deregister
Register targets
Target groups route requests to individual registered targets using the protocol and port number specified. Health checks are performed on all registered targets according to the target group's health check settings. Anomaly detection is automatically applied to HTTP/HTTPS target groups with at least 3 healthy targets.
Filter targets
1
Instance ID
Name
Port
Zone
Health status
Health status details
i-03ebc0ea0e83ea805
SecondaryAppASGinstance
80
us-west-2a (usw2-az2)
Healthy
-
i-0eac73c68375c452e
SecondaryAppASGinstance
80
us-west-2b (usw2-az1)
Healthy
-
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

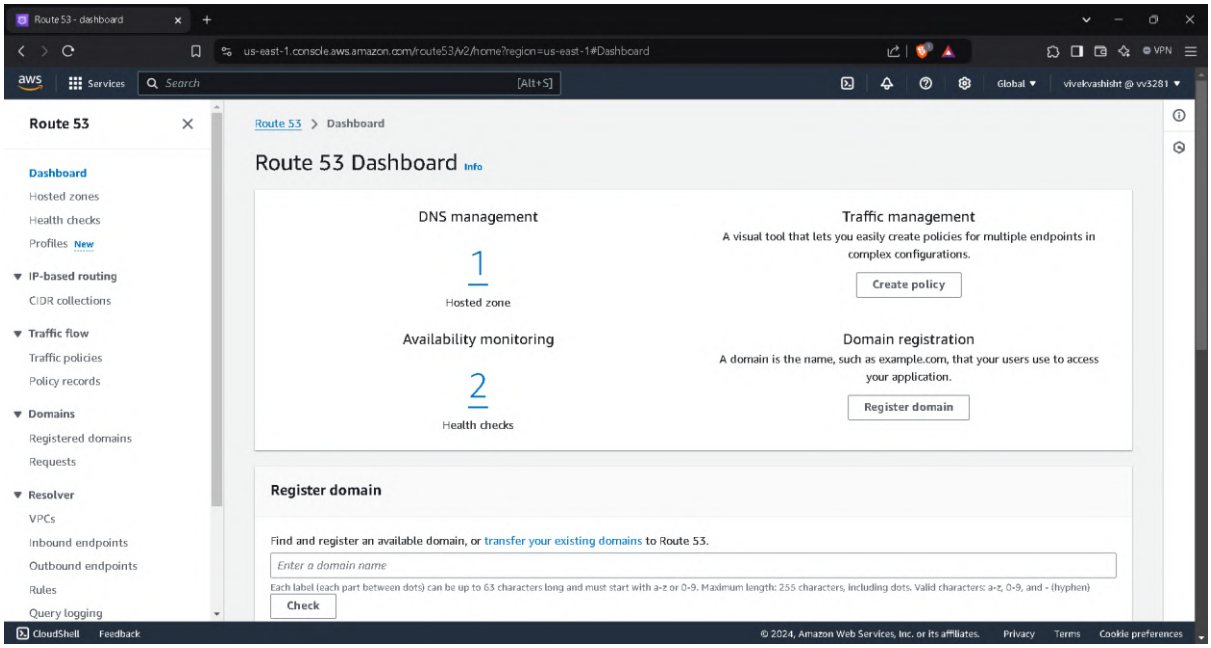

Target groups | EC2 | us-west-2
us-west-2.console.aws.amazon.com/ec2/home?region=us-west-2#TargetGroups:v=3;$case=tags:false%5C,client:false;$regex=tags:...
Services
Search
[Alt+S]
Oregon
vivekvashisht @ vv3281
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Network & Security
Security Groups
Elastic IPs
Placement Groups
Key Pairs
Network Interfaces
Load Balancing
Load Balancers
Target Groups
Trust Stores New
Auto Scaling
Auto Scaling Groups
Settings
EC2 > Target groups
Target groups (1/1) Info
Actions
Create target group
Filter target groups
1
Name
ARN
Port
Protocol
Target type
Load balancer
secondary-target-group
arn:aws:elasticloadbalanci...
80
HTTP
instance
secondary-alb
Target group: secondary-target-group
Details
Targets
Monitoring
Health checks
Attributes
Tags
Health check settings
Edit
Protocol
HTTP
Path
/
Port
Traffic port
Healthy threshold
2 consecutive health check successes
Unhealthy threshold
2 consecutive health check failures
Timeout
5 seconds
Interval
30 seconds
Success codes
200
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

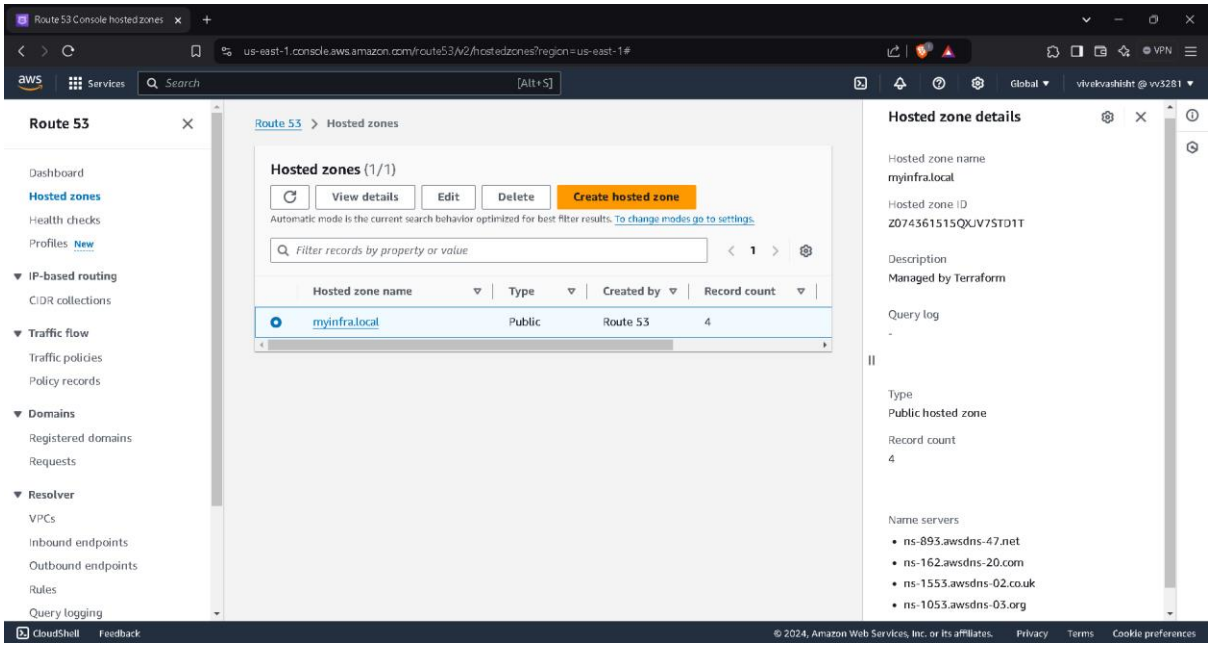

Load balancers | EC2 | us-east-1
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LoadBalancers:
Services
Search
[Alt+S]
N. Virginia
vivekvashisht @ vv3281
Dashboard
EC2 Global View
Events
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations New
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
EC2 > Load balancers
Load balancers (1/1)
Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.
Actions
Create load balancer
Filter load balancers
Name
DNS name
State
VPC ID
Availability Zones
Type
primary-alb
primary-alb-136165955...
Active
vpc-0ea415b9040f75...
2 Availability Zones
application
Load balancer: primary-alb
Details
Load balancer type
Application
Status
Active
VPC
vpc-0ea415b9040f753dc
Load balancer IP address type
IPv4
Scheme
Internet-facing
Hosted zone
Z35SXDOTRQ7X7K
Availability Zones
subnet-0d89362f71d4a5adf us-east-1b (use1-az2)
subnet-04b29a39893e7944d us-east-1a (use1-az1)
Date created
October 30, 2024, 03:41 (UTC-06:00)
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

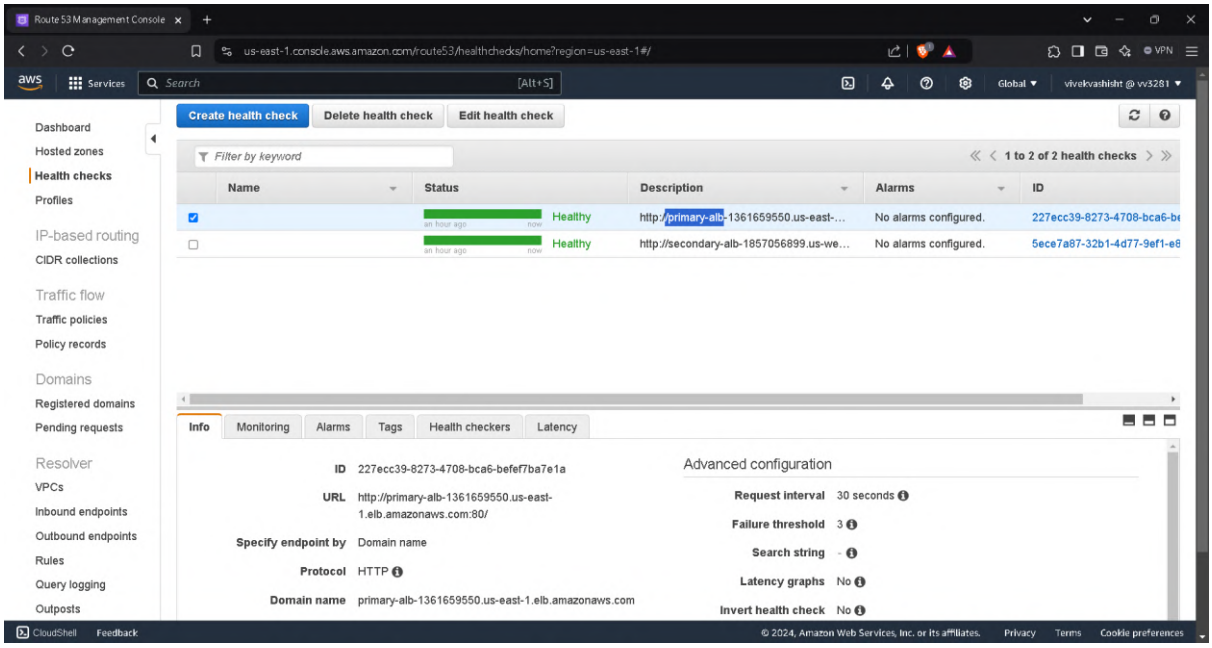

Load balancers | EC2 | us-east-1
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LoadBalancers:
Services
Search
[Alt+S]
N. Virginia
vivekvashisht @ vv3281
Dashboard
EC2 Global View
Events
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations New
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
EC2 > Load balancers
Load balancers (1/1)
Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.
Actions
Create load balancer
Filter load balancers
Name
DNS name
State
VPC ID
Availability Zones
Type
primary-alb
primary-alb-136165955...
Active
vpc-0ea415b9040f75...
2 Availability Zones
application
Load balancer: primary-alb
Listeners and rules (1/1) Info
Manage rules
Manage listener
Add listener
A listener checks for connection requests on its configured protocol and port. Traffic received by the listener is routed according to the default action and any additional rules.
Filter listeners
Protocol:Port
Default action
Rules
ARN
Security policy
HTTP:80
Forward to target group
primary-target-group: 1 (100%)
Target group stickiness: Off
1 rule
ARN
Not applicable
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

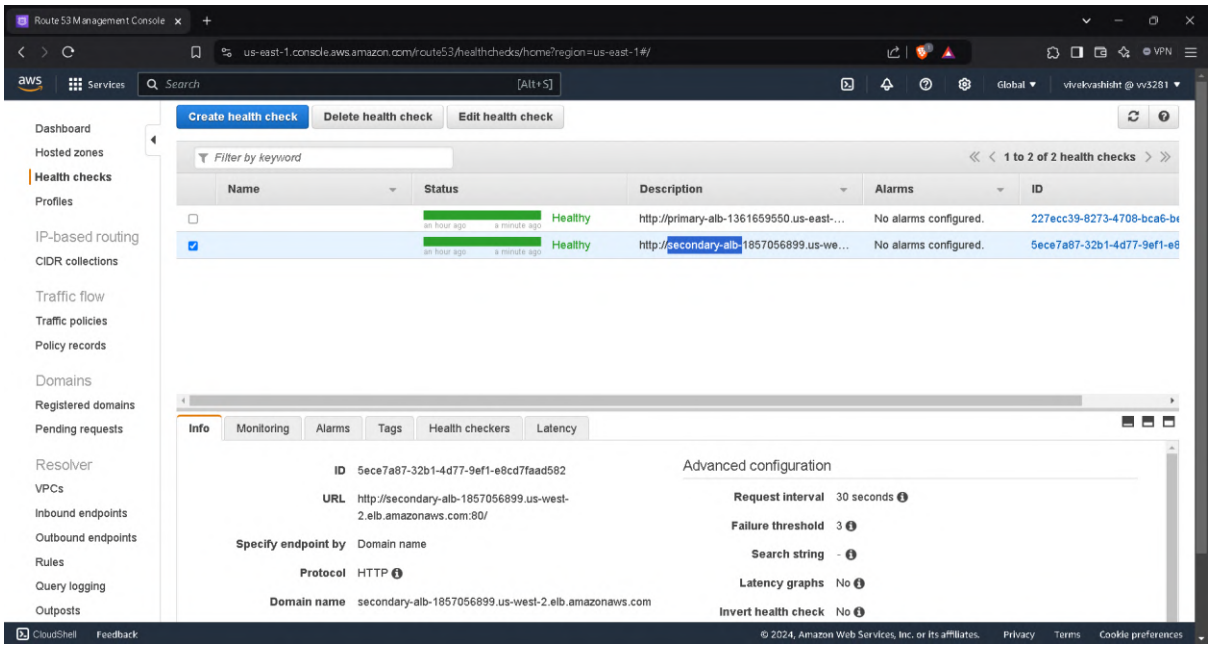

Target groups | EC2 | us-east-1
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#TargetGroups:
Services
Search
[Alt+S]
N. Virginia
vivekvashisht @ vv3281
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Network & Security
Security Groups
Elastic IPs
Placement Groups
Key Pairs
Network Interfaces
Load Balancing
Load Balancers
Target Groups
Trust Stores New
Auto Scaling
Auto Scaling Groups
Settings
EC2 > Target groups
Target groups (1/1) Info
Actions
Create target group
Filter target groups
Name
ARN
Port
Protocol
Target type
Load balancer
primary-target-group
arn:aws:elasticloadbalanci...
80
HTTP
instance
primary-alb
Target group: primary-target-group
Details
arn:aws:elasticloadbalancing:us-east-1:134575801911:targetgroup/primary-target-group/a9447d338de6a3e5
Target type
Instance
Protocol : Port
HTTP: 80
Protocol version
HTTP1
VPC
vpc-0ea415b9040f753dc
IP address type
IPv4
Load balancer
primary-alb
2
Total targets
2
Healthy
0
Unhealthy
0
Unused
0
Initial
0
Draining
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Target groups | EC2 | us-east-1
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#TargetGroups:
Services
Search
[Alt+S]
N. Virginia
vivekvashisht @ vv3281
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Network & Security
Security Groups
Elastic IPs
Placement Groups
Key Pairs
Network Interfaces
Load Balancing
Load Balancers
Target Groups
Trust Stores New
Auto Scaling
Auto Scaling Groups
Settings
EC2 > Target groups
Target groups (1/1) Info
Actions
Create target group
Filter target groups
Name
ARN
Port
Protocol
Target type
Load balancer
primary-target-group
arn:aws:elasticloadbalanci...
80
HTTP
instance
primary-alb
Target group: primary-target-group
Registered targets (2/2) Info
Anomaly mitigation: Not applicable
Deregister
Register targets
Target groups route requests to individual registered targets using the protocol and port number specified. Health checks are performed on all registered targets according to the target group's health check settings. Anomaly detection is automatically applied to HTTP/HTTPS target groups with at least 3 healthy targets.
Filter targets
Instance ID
Name
Port
Zone
Health status
Health status details
Launch...
i-00347e579d3947ab7
AppASGInstance
80
us-east-1b (us...
Healthy
October ..
i-0028fb710309986f1
AppASGInstance
80
us-east-1a (us...
Healthy
October ..
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Route 53 - dashboard
us-east-1.console.aws.amazon.com/route53/v2/home?region=us-east-1#Dashboard
Services
Search
[Alt+S]
Global
vivekvashisht @ vv3281
Route 53
Dashboard
Hosted zones
Health checks
Profiles New
IP-based routing
CIDR collections
Traffic flow
Traffic policies
Policy records
Domains
Registered domains
Requests
Resolver
VPCs
Inbound endpoints
Outbound endpoints
Rules
Query logging
Route 53 > Dashboard
Route 53 Dashboard Info
DNS management
1
Hosted zone
Traffic management
A visual tool that lets you easily create policies for multiple endpoints in complex configurations.
Create policy
Availability monitoring
2
Health checks
Domain registration
A domain is the name, such as example.com, that your users use to access your application.
Register domain
Register domain
Find and register an available domain, or transfer your existing domains to Route 53.
Enter a domain name
Each label (each part between dots) can be up to 63 characters long and must start with a-z or 0-9. Maximum length: 255 characters, including dots. Valid characters: a-z, 0-9, and - (hyphen)
Check
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Route 53 Console hosted zones
us-east-1.console.aws.amazon.com/route53/v2/hostedzones?region=us-east-1#
Services
Search
[Alt+S]
Global
vivekvashisht @ vv3281
Route 53
Dashboard
Hosted zones
Health checks
Profiles New
IP-based routing
CIDR collections
Traffic flow
Traffic policies
Policy records
Domains
Registered domains
Requests
Resolver
VPCs
Inbound endpoints
Outbound endpoints
Rules
Query logging
Route 53 > Hosted zones
Hosted zones (1/1)
View details
Edit
Delete
Create hosted zone
Automatic mode is the current search behavior optimized for best filter results. To change modes go to settings.
Filter records by property or value
Hosted zone name
Type
Created by
Record count
myinfra.local
Public
Route 53
4
Hosted zone details
Hosted zone name
myinfra.local
Hosted zone ID
Z074361515QXJV7STD1T
Description
Managed by Terraform
Query log
-
Type
Public hosted zone
Record count
4
Name servers
ns-893.awsdns-47.net
ns-162.awsdns-20.com
ns-1553.awsdns-02.co.uk
ns-1053.awsdns-03.org
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Route 53 Management Console
us-east-1.console.aws.amazon.com/route53/healthchecks/home?region=us-east-1#/
Services
Search
[Alt+S]
Global
vivekvashisht @ vv3281
Dashboard
Hosted zones
Health checks
Profiles
IP-based routing
CIDR collections
Traffic flow
Traffic policies
Policy records
Domains
Registered domains
Pending requests
Resolver
VPCs
Inbound endpoints
Outbound endpoints
Rules
Query logging
Outposts
Create health check
Delete health check
Edit health check
Filter by keyword
1 to 2 of 2 health checks
Name
Status
Description
Alarms
ID
Healthy
http://primary-alb-1361659550.us-east-...
No alarms configured.
227ecc39-8273-4708-bca6-be
Healthy
http://secondary-alb-1857056899.us-we...
No alarms configured.
5ece7a87-32b1-4d77-9ef1-e8
Info
Monitoring
Alarms
Tags
Health checkers
Latency
ID 227ecc39-8273-4708-bca6-befef7ba7e1a
URL http://primary-alb-1361659550.us-east-1.elb.amazonaws.com:80/
Specify endpoint by Domain name
Protocol HTTP
Domain name primary-alb-1361659550.us-east-1.elb.amazonaws.com
Advanced configuration
Request interval 30 seconds
Failure threshold 3
Search string -
Latency graphs No
Invert health check No
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Route 53 Management Console
us-east-1.console.aws.amazon.com/route53/healthchecks/home?region=us-east-1#/
Services
Search
[Alt+S]
Global
vivekvashisht @ vv3281
Dashboard
Hosted zones
Health checks
Profiles
IP-based routing
CIDR collections
Traffic flow
Traffic policies
Policy records
Domains
Registered domains
Pending requests
Resolver
VPCs
Inbound endpoints
Outbound endpoints
Rules
Query logging
Outposts
Create health check
Delete health check
Edit health check
Filter by keyword
1 to 2 of 2 health checks
Name
Status
Description
Alarms
ID
Healthy
http://primary-alb-1361659550.us-east-...
No alarms configured.
227ecc39-8273-4708-bca6-be
Healthy
http://secondary-alb-1857056899.us-we...
No alarms configured.
5ece7a87-32b1-4d77-9ef1-e8
Info
Monitoring
Alarms
Tags
Health checkers
Latency
ID 5ece7a87-32b1-4d77-9ef1-e8cd7faad582
URL http://secondary-alb-1857056899.us-west-2.elb.amazonaws.com:80/
Specify endpoint by Domain name
Protocol HTTP
Domain name secondary-alb-1857056899.us-west-2.elb.amazonaws.com
Advanced configuration
Request interval 30 seconds
Failure threshold 3
Search string -
Latency graphs No
Invert health check No
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Monitoring and Alerts with CloudWatch

a) Overview

In this phase, I set up monitoring and alerting for EC2 instances in both us-east-1 and us-west-2 to ensure visibility into resource usage and automated response to any high-utilization events. To enable this, I first updated the IAM role in iam_roles.tf to grant EC2 instances permission to run the CloudWatch Agent. Then, I configured the EC2 launch templates in us-east-1 and us-west-2 with user data scripts to pre-install both NGINX and the CloudWatch Agent, along with a configuration for tracking CPU, memory, and disk usage. For alerting, I created cloudwatch_alarms.tf, defining alarms for high CPU, memory, and disk utilization, each set to trigger notifications via SNS when metrics exceed thresholds. I also added an IAM policy to allow EC2 instances to publish to SNS topics in both regions. After deployment, I verified that CloudWatch alarms were correctly set up and linked to SNS, and I confirmed alert functionality by subscribing to SNS topics via email. This monitoring and alerting setup provides automated insights into resource utilization and prompt alerts for effective performance management across both regions.

b) Attaching CloudWatch Agent Permissions

To enable CloudWatch monitoring, I needed to attach the CloudWatch agent policy to the IAM role for the EC2 instances.

Code Block: IAM Role Policy Attachment

```
# Attachment of CloudWatch Agent Policy to the IAM role for the EC2
instances
resource "aws_iam_role_policy_attachment" "ec2_cloudwatch_agent" {
  role       = aws_iam_role.ec2_role.name
  policy_arn = "arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy"
}
```

After saving the updated `iam_roles.tf`, I ran `terraform plan` and `terraform apply -auto-approve` to implement the changes. After successful deployment, I verified in the AWS Console under IAM roles that the `CloudWatchAgentServerPolicy` was indeed attached to the `ec2-role`, confirming proper configuration.

c) Updating EC2 User Data for CloudWatch Agent Installation

To automate the installation of the CloudWatch agent on EC2 instances, I updated the user data in the Launch Template configuration. This ensures that each instance has both `nginx` and `amazon-cloudwatch-agent` pre-installed.

Code Block: Launch Template for us-east-1 with CloudWatch Agent Installation

```
# Launch Template for EC2 Instances for us-east-1
resource "aws_launch_template" "app" {
  name_prefix   = "app-launch-template"
```

```
  image_id      = "ami-06b21ccaeff8cd686"  # Amazon Linux AMI
  instance_type = var.instance_type

  # Attach IAM role to EC2 instances
  iam_instance_profile {
    name = aws_iam_instance_profile.ec2_instance_profile.name
  }

  network_interfaces {
    security_groups = [aws_security_group.app_sg.id]
  }

  user_data = base64encode(<<-EOF
              #!/bin/bash
              sudo dnf update -y
              sudo dnf install -y nginx amazon-cloudwatch-agent
              sudo systemctl start nginx
              sudo systemctl enable nginx

              # Configure the CloudWatch Agent
              sudo tee /opt/aws/amazon-cloudwatch-agent/etc/amazon-
cloudwatch-agent.json << 'EOF2'
              {
                "metrics": {
                  "append_dimensions": {
                    "AutoScalingGroupName": "AppASGInstance"
                  },
                  "metrics_collected": {
                    "mem": {
                      "measurement": [
                        "mem_used_percent"
                      ],
                      "metrics_collection_interval": 60
                    },
                    "disk": {
                      "measurement": [
                        "used_percent"
                      ],
                      "resources": [
                        "/"
                      ],
                      "metrics_collection_interval": 60
                    }
                  }
                }
              }
              EOF2

              # Start the CloudWatch Agent
              sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-
agent-ctl -a start -c file:/opt/aws/amazon-cloudwatch-agent/etc/amazon-
cloudwatch-agent.json -m ec2
            EOF
  )

  lifecycle {
    create_before_destroy = true
  }

  tag_specifications {
    resource_type = "instance"
```

```
    tags = {
      Name = "AppInstance"
    }
  }
}
```

d) Adding Instance Refresh for Auto Scaling Group (ASG) in us-east-1

To apply this updated configuration to all existing instances, I updated the Auto Scaling Group (ASG) in `us-east-1` to perform a rolling instance refresh. This ensures minimal downtime while applying the new settings.

Code Block: Instance Refresh Configuration

```
instance_refresh {
    strategy = "Rolling"
    preferences {
      min_healthy_percentage = 0
      instance_warmup        = 120
    }
  }
```

e) Updating the Launch Template and ASG for us-west-2

Similarly, I applied the same configuration updates to the Launch Template and ASG in `us-west-2` to ensure CloudWatch monitoring for instances in both regions.

Code Block: Launch Template for us-west-2 with CloudWatch Agent Installation

```
# Launch Template for EC2 Instances in us-west-2
resource "aws_launch_template" "secondary_app" {
  provider       = aws.west
  name_prefix    = "secondary-app-launch-template"
  image_id       = "ami-07c5ecd8498c59db5"  # Amazon Linux AMI
  instance_type  = var.instance_type

  # Attach IAM role to EC2 instances
  iam_instance_profile {
    name = aws_iam_instance_profile.ec2_instance_profile.name
  }

  network_interfaces {
    security_groups = [aws_security_group.west_app_sg.id]
  }

  user_data = base64encode(<<-EOF
              #!/bin/bash
              sudo dnf update -y
              sudo dnf install -y nginx amazon-cloudwatch-agent
              sudo systemctl start nginx
              sudo systemctl enable nginx

              # Configure the CloudWatch Agent
              sudo tee /opt/aws/amazon-cloudwatch-agent/etc/amazon-
cloudwatch-agent.json << 'EOF2'
              {
                "metrics": {
```

```
                  "append_dimensions": {
                    "AutoScalingGroupName": "SecondaryAppASGInstance"
                  },
                  "metrics_collected": {
                    "mem": {
                      "measurement": [
                        "mem_used_percent"
                      ],
                      "metrics_collection_interval": 60
                    },
                    "disk": {
                      "measurement": [
                        "used_percent"
                      ],
                      "resources": [
                        "/"
                      ],
                      "metrics_collection_interval": 60
                    }
                  }
                }
              }
              EOF2

              # Start the CloudWatch Agent
              sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-
agent-ctl -a start -c file:/opt/aws/amazon-cloudwatch-agent/etc/amazon-
cloudwatch-agent.json -m ec2
            EOF
  )

  lifecycle {
    create_before_destroy = true
  }

  tag_specifications {
    resource_type = "instance"
    tags = {
      Name = "SecondaryAppInstance"
    }
  }
}
```

Updating ASG for us-west-2 with Instance Refresh

```
instance_refresh {
    strategy = "Rolling"
    preferences {
      min_healthy_percentage = 0
      instance_warmup        = 120
    }
  }
```

f) Redeploying with Updated Configurations

To apply these configurations, I first destroyed the existing ASGs and launch templates to ensure fresh provisioning with the updated settings. I ran:

```
terraform destroy -target=aws_autoscaling_group.app_asg -
target=aws_autoscaling_group.secondary_asg -target=aws_launch_template.app
-target=aws_launch_template.secondary_app
```

After successfully destroying the resources, I re-ran:

```
terraform init
terraform plan
terraform apply -auto-approve
```

g) Verification

Once the resources were recreated, I verified the setup by connecting to instances in both `us-east-1` and `us-west-2` and running the following commands:

```
curl http://localhost
rpm -qa | grep amazon-cloudwatch-agent
```

These commands confirmed that both `nginx` and the `cloudwatch-agent` were correctly installed and active on all instances in both regions.

h) Creating CloudWatch Alarms and SNS Topics

To enable real-time alerts for CPU, memory, and disk utilization across both `us-east-1` and `us-west-2` regions, I created a new Terraform file named `cloudwatch_alarms.tf`. This file defines the CloudWatch alarms for each metric, as well as SNS topics for alert notifications.

Code Block: CloudWatch Alarms and SNS Topics Configuration

```
# us-east-1

# CloudWatch Alarm for CPU Utilization
resource "aws_cloudwatch_metric_alarm" "high_cpu_utilization" {
  alarm_name          = "HighCPUUtilizationAlarm"
  comparison_operator = "GreaterThanThreshold"
  evaluation_periods  = 2
  metric_name         = "CPUUtilization"
  namespace           = "AWS/EC2"
  period              = 300
  statistic           = "Average"
  threshold           = 80 # Threshold at 80% CPU Utilization
  alarm_description   = "Triggers if CPU utilization exceeds 80%."
  dimensions = {
    AutoScalingGroupName = aws_autoscaling_group.app_asg.name
  }
  actions_enabled = true
  alarm_actions   = [aws_sns_topic.alert_topic.arn]
}

# CloudWatch Alarm for Memory Utilization
resource "aws_cloudwatch_metric_alarm" "high_memory_utilization" {
  alarm_name          = "HighMemoryUtilizationAlarm"
  comparison_operator = "GreaterThanThreshold"
  evaluation_periods  = 2
  metric_name         = "mem_used_percent" # Custom metric from CloudWatch
Agent
```

```
  namespace           = "CWAgent"
  period              = 300
  statistic           = "Average"
  threshold           = 80 # Threshold at 80% Memory Utilization
  alarm_description   = "Triggers if memory utilization exceeds 80%."
  dimensions = {
    AutoScalingGroupName = aws_autoscaling_group.app_asg.name
  }
  actions_enabled = true
  alarm_actions   = [aws_sns_topic.alert_topic.arn]
}

# CloudWatch Alarm for Disk Utilization
resource "aws_cloudwatch_metric_alarm" "high_disk_utilization" {
  alarm_name          = "HighDiskUtilizationAlarm"
  comparison_operator = "GreaterThanThreshold"
  evaluation_periods  = 2
  metric_name         = "disk_used_percent" # Custom metric from CloudWatch
Agent
  namespace           = "CWAgent"
  period              = 300
  statistic           = "Average"
  threshold           = 80 # Threshold at 80% Disk Utilization
  alarm_description   = "Triggers if disk utilization exceeds 80%."
  dimensions = {
    AutoScalingGroupName = aws_autoscaling_group.app_asg.name
  }
  actions_enabled = true
  alarm_actions   = [aws_sns_topic.alert_topic.arn]
}

# SNS Topic for Alarm Notifications in us-east-1
resource "aws_sns_topic" "alert_topic" {
  name = "EC2AlertTopic"
}

resource "aws_sns_topic_subscription" "email_subscription" {
  topic_arn = aws_sns_topic.alert_topic.arn
  protocol  = "email"
  endpoint  = "vivekvash1507@gmail.com"
}
```

Similarly, I configured alarms and SNS topics for the `us-west-2` region, following the same structure.

Code Block: CloudWatch Alarms and SNS Topics Configuration for us-west-2

```
# us-west-2

# CloudWatch Alarm for CPU Utilization
resource "aws_cloudwatch_metric_alarm" "west_high_cpu_utilization" {
  provider              = aws.west
  alarm_name            = "WestHighCPUUtilizationAlarm"
  comparison_operator   = "GreaterThanThreshold"
  evaluation_periods    = 2
  metric_name           = "CPUUtilization"
  namespace             = "AWS/EC2"
  period                = 300
  statistic             = "Average"
  threshold             = 80
```

```
  alarm_description      = "Triggers if CPU utilization exceeds 80%."
  dimensions = {
    AutoScalingGroupName = aws_autoscaling_group.secondary_asg.name
  }
  actions_enabled = true
  alarm_actions   = [aws_sns_topic.west_alert_topic.arn]
}

# CloudWatch Alarm for Memory Utilization in us-west-2
resource "aws_cloudwatch_metric_alarm" "west_high_memory_utilization" {
  provider                = aws.west
  alarm_name              = "WestHighMemoryUtilizationAlarm"
  comparison_operator     = "GreaterThanThreshold"
  evaluation_periods      = 2
  metric_name             = "mem_used_percent"
  namespace               = "CWAgent"
  period                  = 300
  statistic               = "Average"
  threshold               = 80  # Set threshold based on your requirements
  alarm_description      = "This alarm triggers if memory utilization
exceeds 80%."
  dimensions = {
    AutoScalingGroupName = aws_autoscaling_group.secondary_asg.name
  }
  actions_enabled = true
  alarm_actions   = [aws_sns_topic.west_alert_topic.arn]
}

# CloudWatch Alarm for Disk Utilization in us-west-2
resource "aws_cloudwatch_metric_alarm" "west_high_disk_utilization" {
  provider                = aws.west
  alarm_name              = "WestHighDiskUtilizationAlarm"
  comparison_operator     = "GreaterThanThreshold"
  evaluation_periods      = 2
  metric_name             = "disk_used_percent"
  namespace               = "CWAgent"
  period                  = 300
  statistic               = "Average"
  threshold               = 80  # Set threshold based on your requirements
  alarm_description      = "This alarm triggers if disk utilization exceeds
80%."
  dimensions = {
    AutoScalingGroupName = aws_autoscaling_group.secondary_asg.name
  }
  actions_enabled = true
  alarm_actions   = [aws_sns_topic.west_alert_topic.arn]
}


# SNS Topic for Alarm Notifications in us-west-2
resource "aws_sns_topic" "west_alert_topic" {
  provider = aws.west
  name     = "WestEC2AlertTopic"
}

resource "aws_sns_topic_subscription" "west_email_subscription" {
  provider  = aws.west
  topic_arn = aws_sns_topic.west_alert_topic.arn
  protocol  = "email"
  endpoint  = "vivekvash1507@gmail.com"
}
```

i) Updating IAM Role Permissions for SNS Publishing

To allow EC2 instances to publish alerts to the SNS topics, I added an additional policy to the IAM role in `iam_roles.tf`.

Code Block: SNS Publish Policy Attachment

```
# Allow EC2 instances to publish to both SNS topics for alerts
resource "aws_iam_role_policy" "ec2_sns_publish_policy" {
  name = "ec2-sns-publish-policy"
  role = aws_iam_role.ec2_role.name

  policy = jsonencode({
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": "sns:Publish",
        "Resource": [
          "${aws_sns_topic.alert_topic.arn}",       # SNS topic in us-east-
1
          "${aws_sns_topic.west_alert_topic.arn}"   # SNS topic in us-west-
2
        ]
      }
    ]
  })
}
```

j) Deploying and Verifying CloudWatch Alarms and SNS Notifications

After saving the updated `iam_roles.tf`, I executed the following commands to apply the changes:

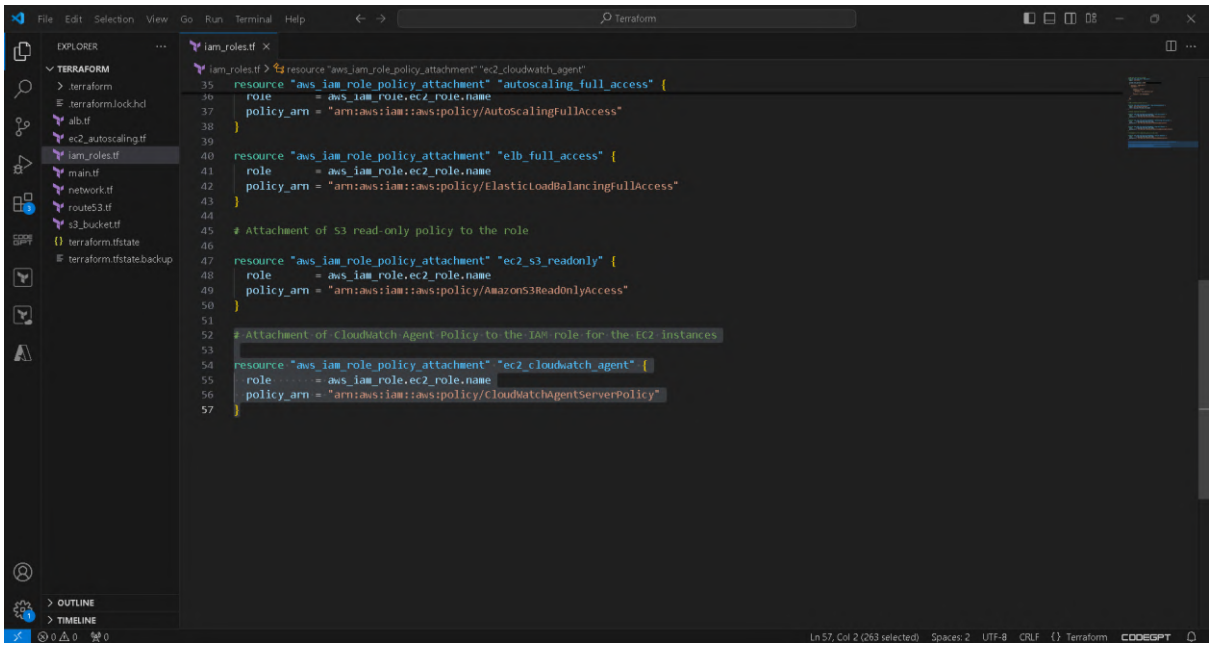
```
terraform init
terraform plan
terraform apply -auto-approve
```

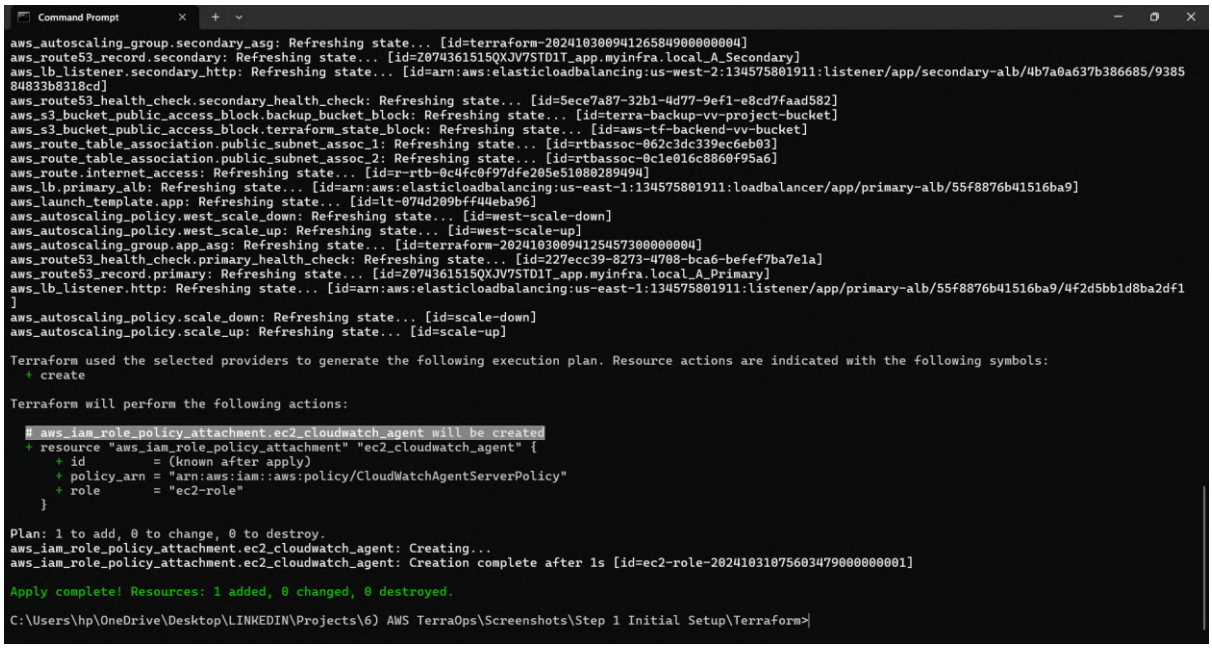
Following successful deployment, I verified each configuration:

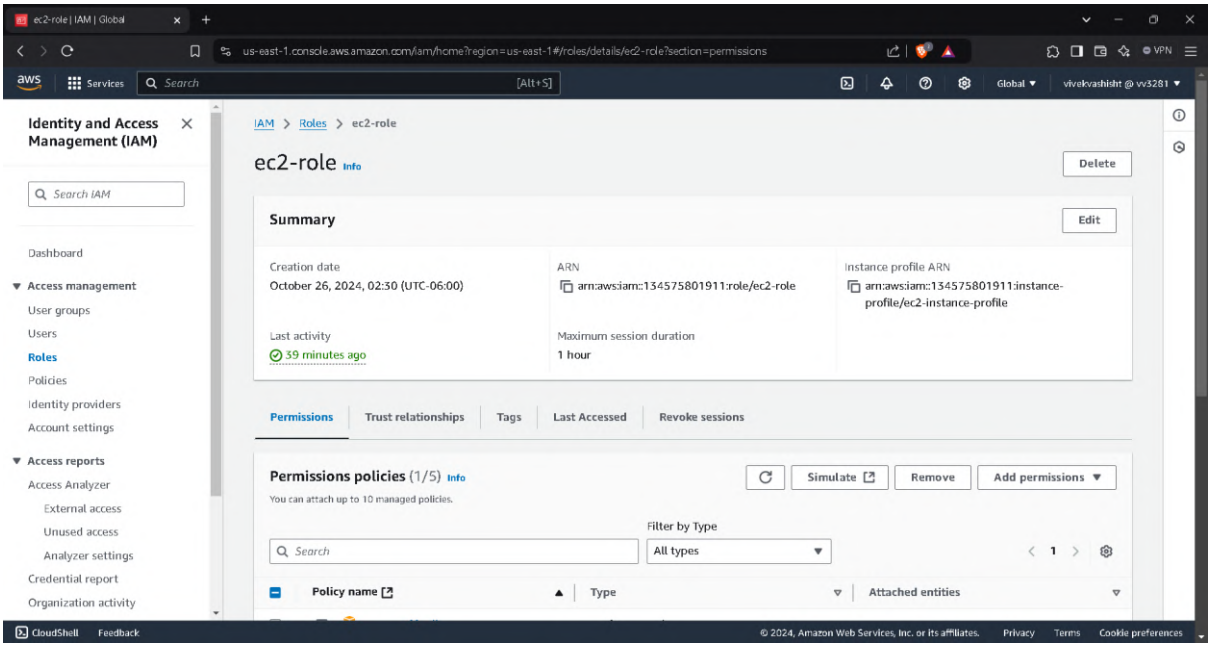
1. **IAM Role Verification**: In the AWS Console, I confirmed that the `ec2-sns-publish-policy` was attached to the `ec2-role`, granting SNS publish permissions.
2. **CloudWatch Alarm Verification**: In both `us-east-1` and `us-west-2`, I accessed CloudWatch and confirmed the presence of the alarms for CPU, memory, and disk utilization.
3. **Email Subscription Confirmation**: I received email confirmations from SNS for both `EC2AlertTopic` and `WestEC2AlertTopic`. I clicked the provided links to confirm each subscription.

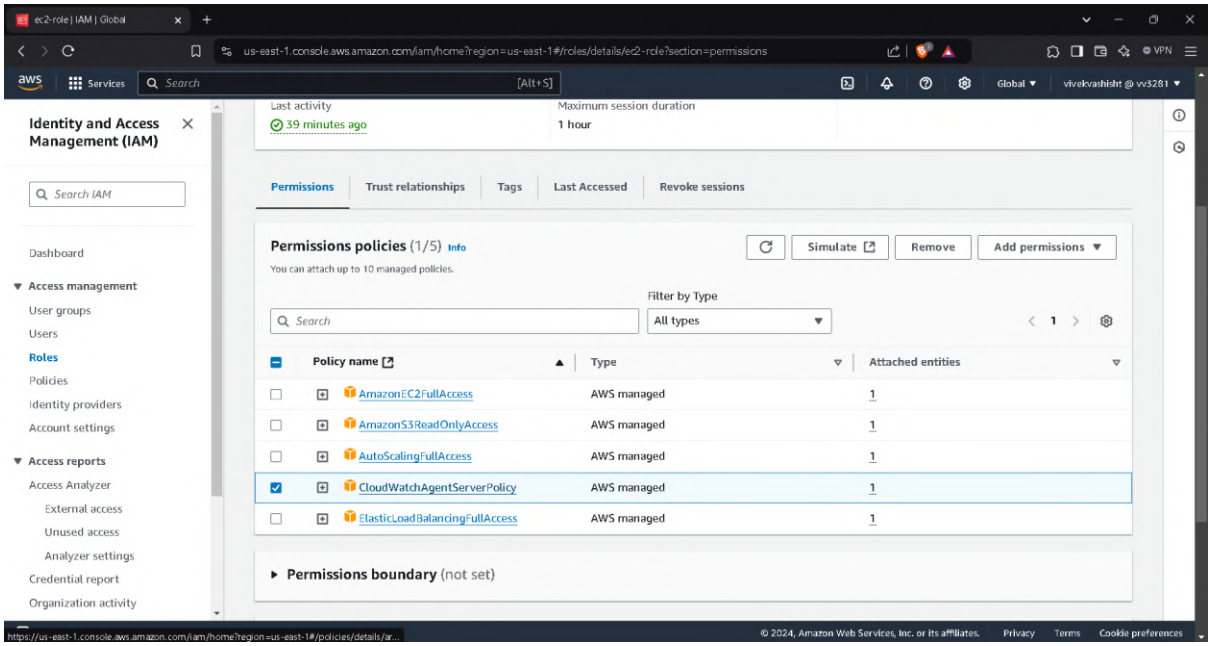
With these configurations, I successfully enabled CloudWatch monitoring and alerting for key utilization metrics across both regions. The EC2 instances are now capable of sending alerts to the configured SNS topics, ensuring timely notifications for any critical utilization thresholds. This marks the successful completion of the Monitoring and Alerts phase of the project.

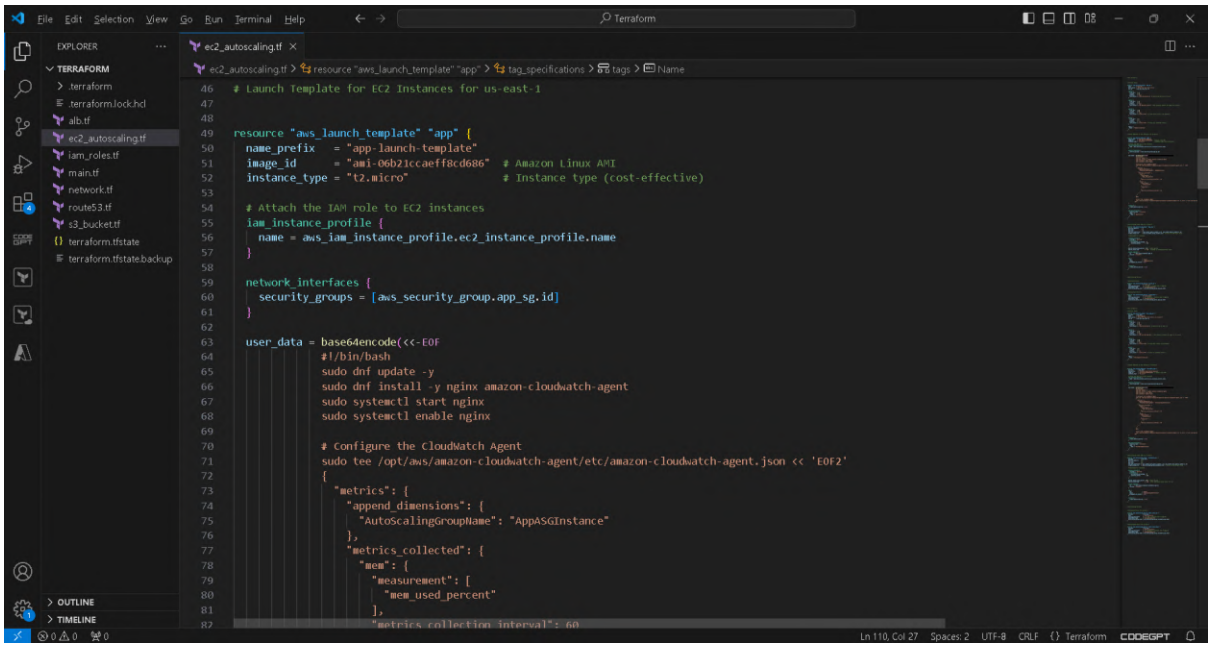
Screenshots

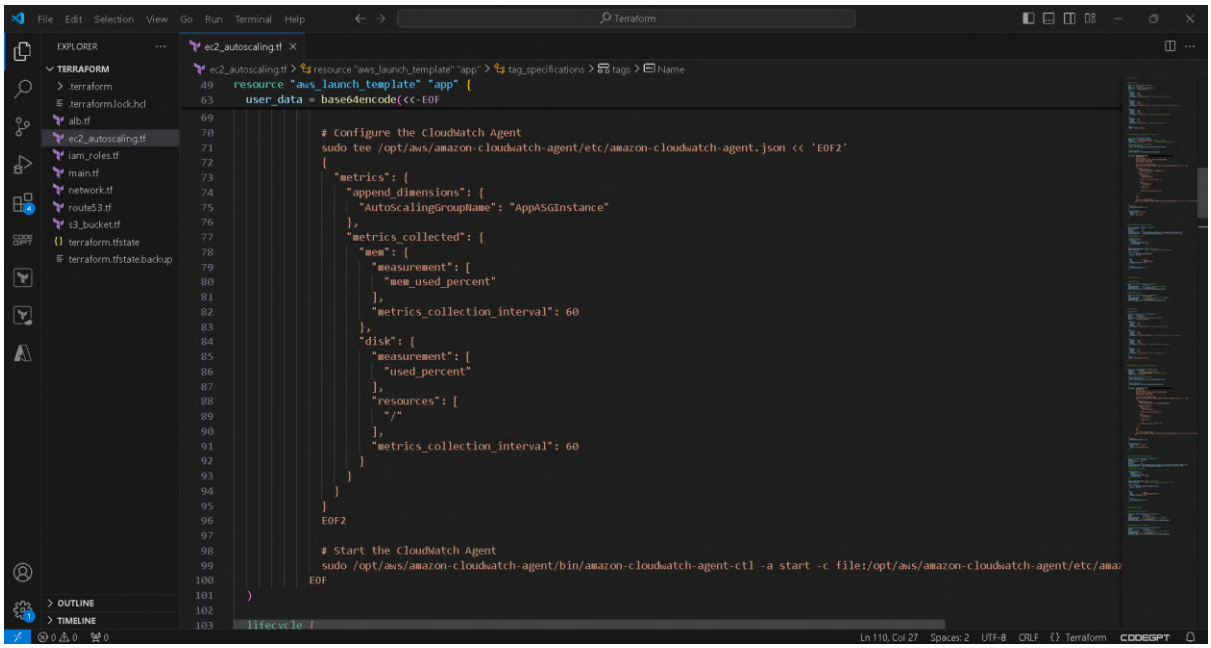




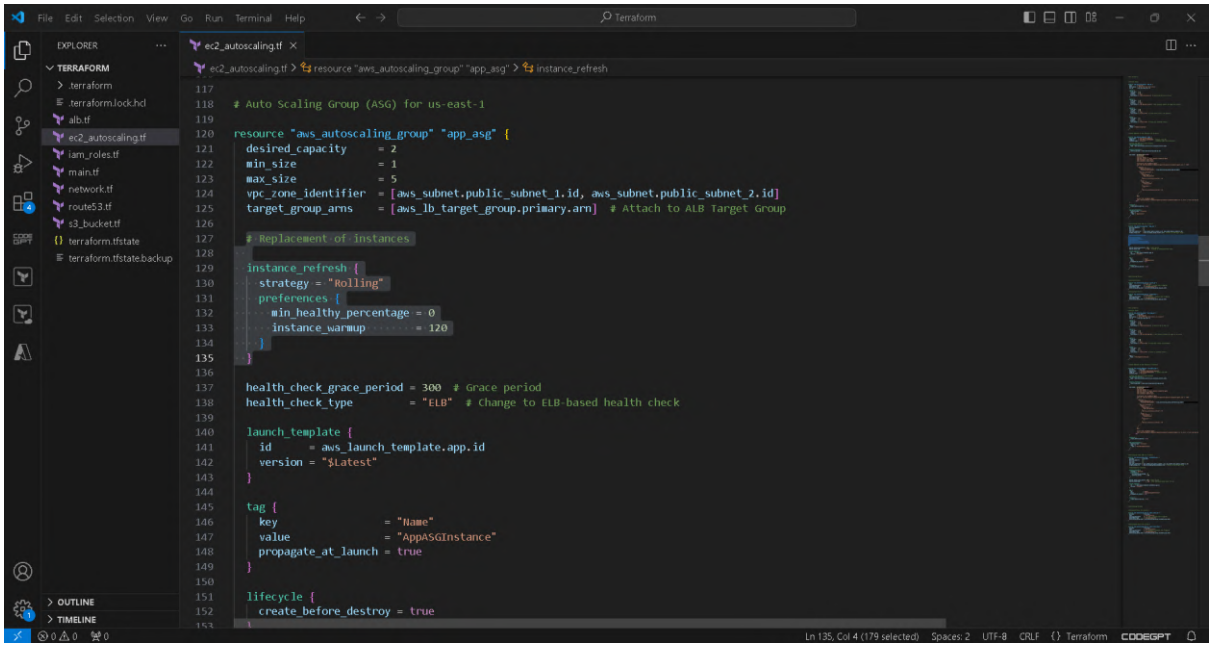

ec2-role | IAM | Global
us-east-1.console.aws.amazon.com/iam/home?region=us-east-1#/roles/details/ec2-role?section=permissions
Services
Search
[Alt+S]
Global
vivekvashisht @ vv3281
Identity and Access Management (IAM)
Search IAM
Dashboard
Access management
User groups
Users
Roles
Policies
Identity providers
Account settings
Access reports
Access Analyzer
External access
Unused access
Analyzer settings
Credential report
Organization activity
IAM > Roles > ec2-role
ec2-role Info
Delete
Summary
Edit
Creation date
October 26, 2024, 02:30 (UTC-06:00)
ARN
arn:aws:iam::134575801911:role/ec2-role
Instance profile ARN
arn:aws:iam::134575801911:instance-profile/ec2-instance-profile
Last activity
39 minutes ago
Maximum session duration
1 hour
Permissions
Trust relationships
Tags
Last Accessed
Revoke sessions
Permissions policies (1/5) Info
You can attach up to 10 managed policies.
Simulate
Remove
Add permissions
Filter by Type
Search
All types
Policy name
Type
Attached entities
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


ec2-role | IAM | Global
us-east-1.console.aws.amazon.com/iam/home?region=us-east-1#/roles/details/ec2-role?section=permissions
Identity and Access Management (IAM)
Search IAM
Dashboard
Access management
User groups
Users
Roles
Policies
Identity providers
Account settings
Access reports
Access Analyzer
External access
Unused access
Analyzer settings
Credential report
Organization activity
Last activity
39 minutes ago
Maximum session duration
1 hour
Permissions
Trust relationships
Tags
Last Accessed
Revoke sessions
Permissions policies (1/5) Info
You can attach up to 10 managed policies.
Simulate
Remove
Add permissions
Filter by Type
Search
All types
Policy name
Type
Attached entities
AmazonEC2FullAccess
AWS managed
1
AmazonS3ReadOnlyAccess
AWS managed
1
AutoScalingFullAccess
AWS managed
1
CloudWatchAgentServerPolicy
AWS managed
1
ElasticLoadBalancingFullAccess
AWS managed
1
Permissions boundary (not set)
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

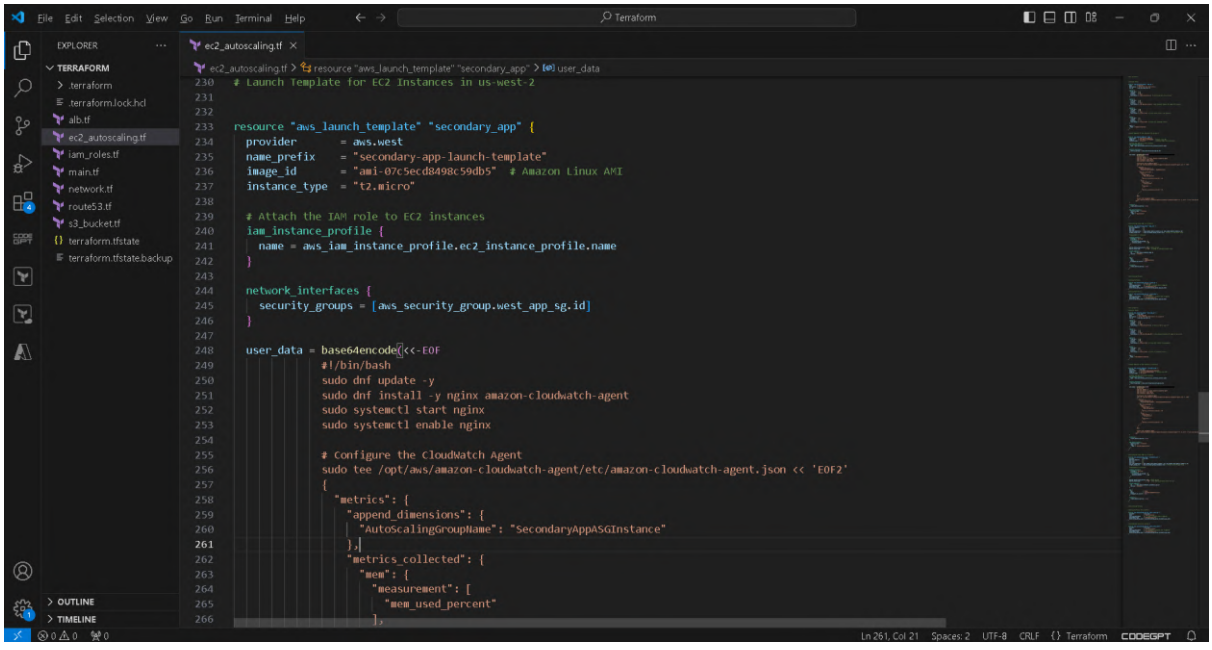

File Edit Selection View Go Run Terminal Help
Terraform
EXPLORER
TERRAFORM
.terraform
.terraform.lock.hcl
alb.tf
ec2_autoscaling.tf
iam_roles.tf
main.tf
network.tf
route53.tf
s3_bucket.tf
terraform.tfstate
terraform.tfstate.backup
OUTLINE
TIMELINE
ec2_autoscaling.tf
ec2_autoscaling.tf > resource "aws_launch_template" "app" > tag_specifications > tags > Name
# Launch Template for EC2 Instances for us-east-1
resource "aws_launch_template" "app" {
name_prefix = "app-launch-template"
image_id = "ami-06b21ccaeff8cd686" # Amazon Linux AMI
instance_type = "t2.micro" # Instance type (cost-effective)
# Attach the IAM role to EC2 instances
iam_instance_profile {
name = aws_iam_instance_profile.ec2_instance_profile.name
}
network_interfaces {
security_groups = [aws_security_group.app_sg.id]
}
user_data = base64encode(<<-EOF
#!/bin/bash
sudo dnf update -y
sudo dnf install -y nginx amazon-cloudwatch-agent
sudo systemctl start nginx
sudo systemctl enable nginx
# Configure the CloudWatch Agent
sudo tee /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json << 'EOF2'
{
"metrics": {
"append_dimensions": {
"AutoScalingGroupName": "AppASGInstance"
},
"metrics_collected": {
"mem": {
"measurement": [
"mem_used_percent"
],
"metrics_collection_interval": 60
Ln 110, Col 27
Spaces: 2
UTF-8
CRLF
Terraform
CODEGPT



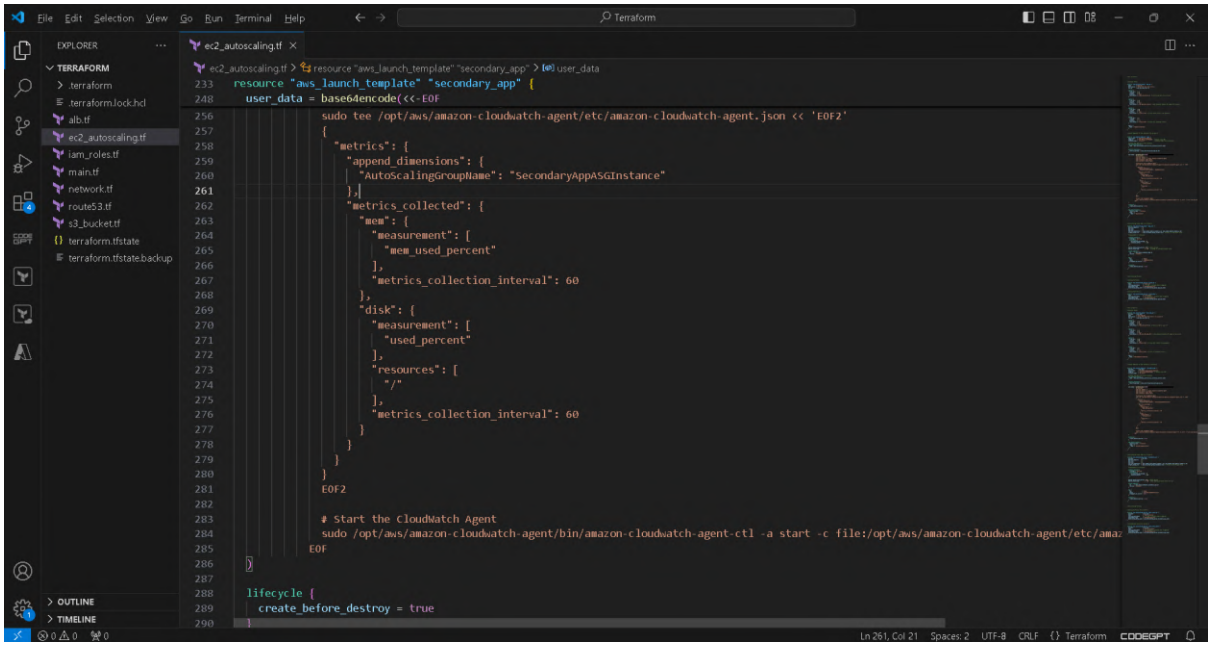
```
resource "aws_launch_template" "app" {
  user_data = base64encode(<<-EOF

              # Configure the CloudWatch Agent
              sudo tee /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json << 'EOF2'
              {
                "metrics": {
                  "append_dimensions": {
                    "AutoScalingGroupName": "AppASGInstance"
                  },
                  "metrics_collected": {
                    "mem": {
                      "measurement": [
                        "mem_used_percent"
                      ],
                      "metrics_collection_interval": 60
                    },
                    "disk": {
                      "measurement": [
                        "used_percent"
                      ],
                      "resources": [
                        "*/"
                      ],
                      "metrics_collection_interval": 60
                    }
                  }
                }
              }
              EOF2

              # Start the CloudWatch Agent
              sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a start -c file:/opt/aws/amazon-cloudwatch-agent/etc/amaz
            EOF
  )
  lifecycle {
```

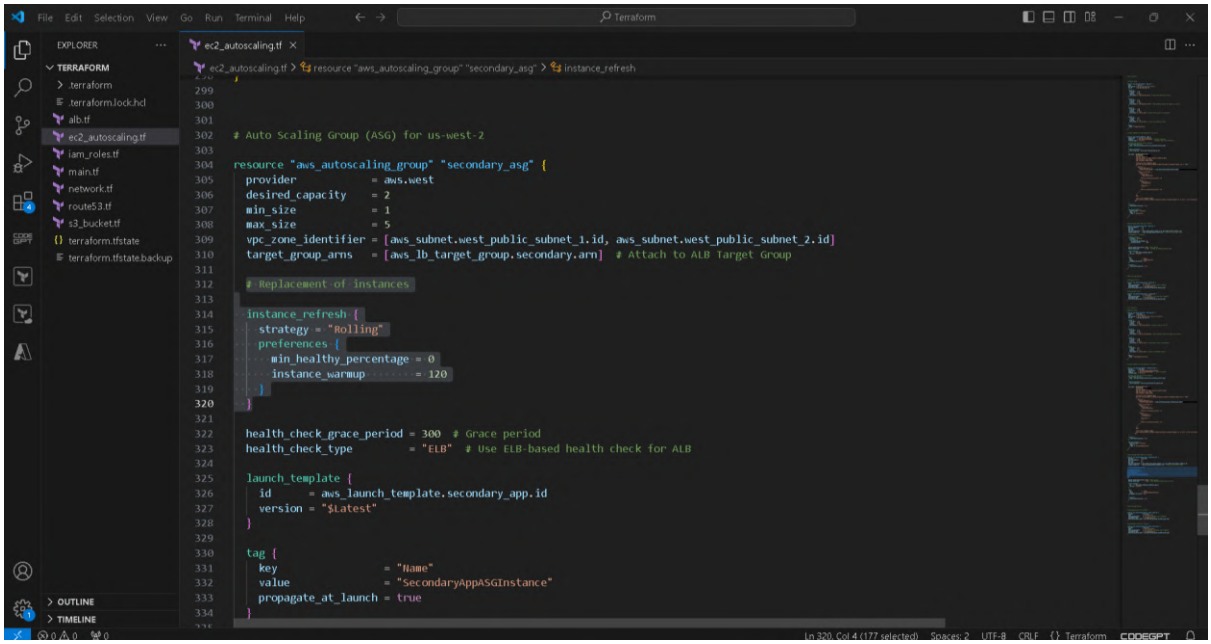


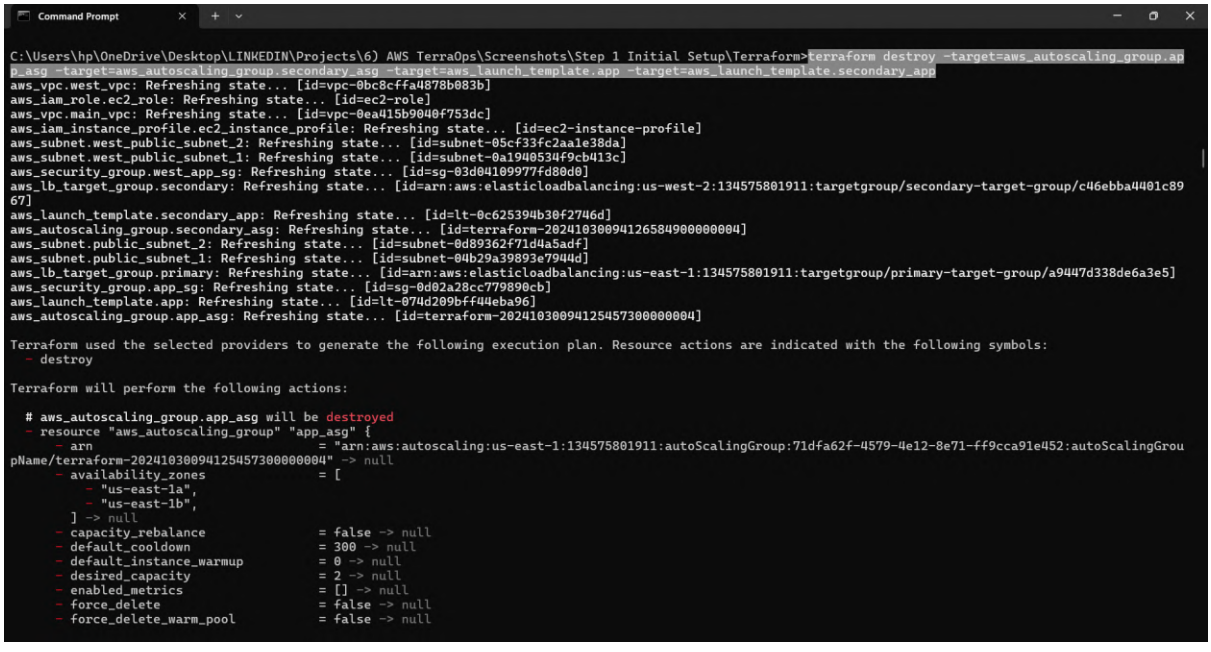
```
# Auto Scaling Group (ASG) for us-east-1

resource "aws_autoscaling_group" "app_asg" {
  desired_capacity    = 2
  min_size            = 1
  max_size            = 5
  vpc_zone_identifier = [aws_subnet.public_subnet_1.id, aws_subnet.public_subnet_2.id]
  target_group_arns   = [aws_lb_target_group.primary.arn]  # Attach to ALB Target Group

  # Replacement of instances
  instance_refresh {
    strategy = "Rolling"
    preferences {
      min_healthy_percentage = 0
      instance_warmup        = 120
    }
  }

  health_check_grace_period = 300  # Grace period
  health_check_type         = "ELB"  # Change to ELB-based health check

  launch_template {
    id      = aws_launch_template.app.id
    version = "$Latest"
  }

  tag {
    key                 = "Name"
    value               = "AppASGInstance"
    propagate_at_launch = true
  }

  lifecycle {
    create_before_destroy = true
```

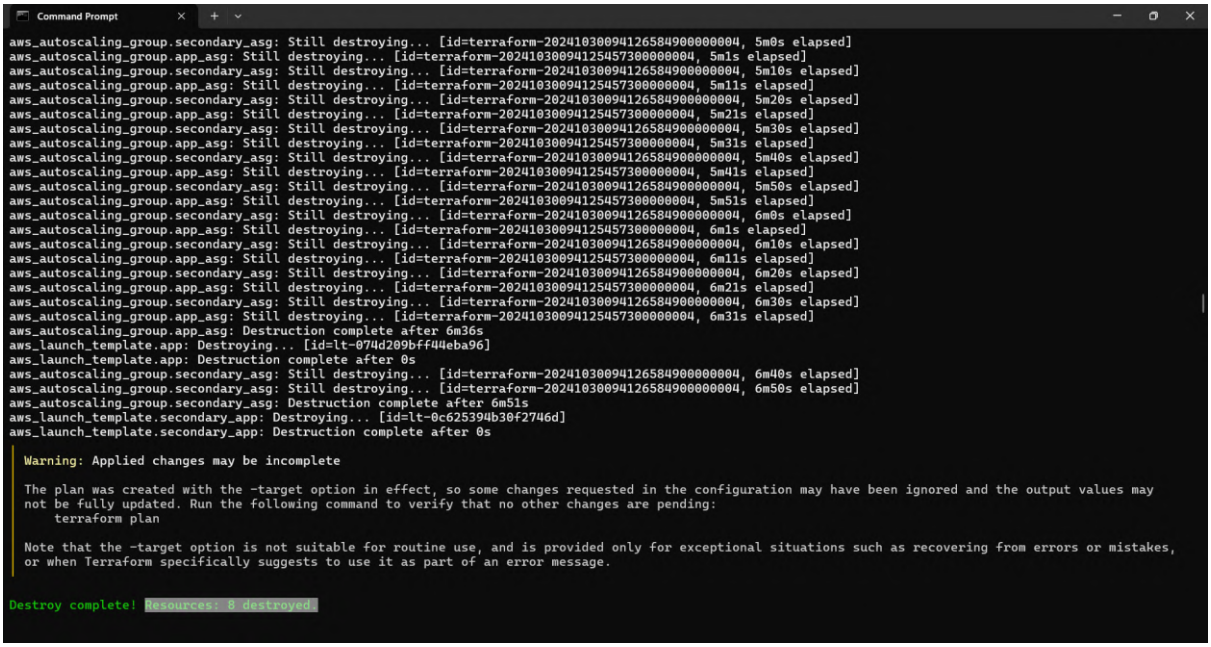


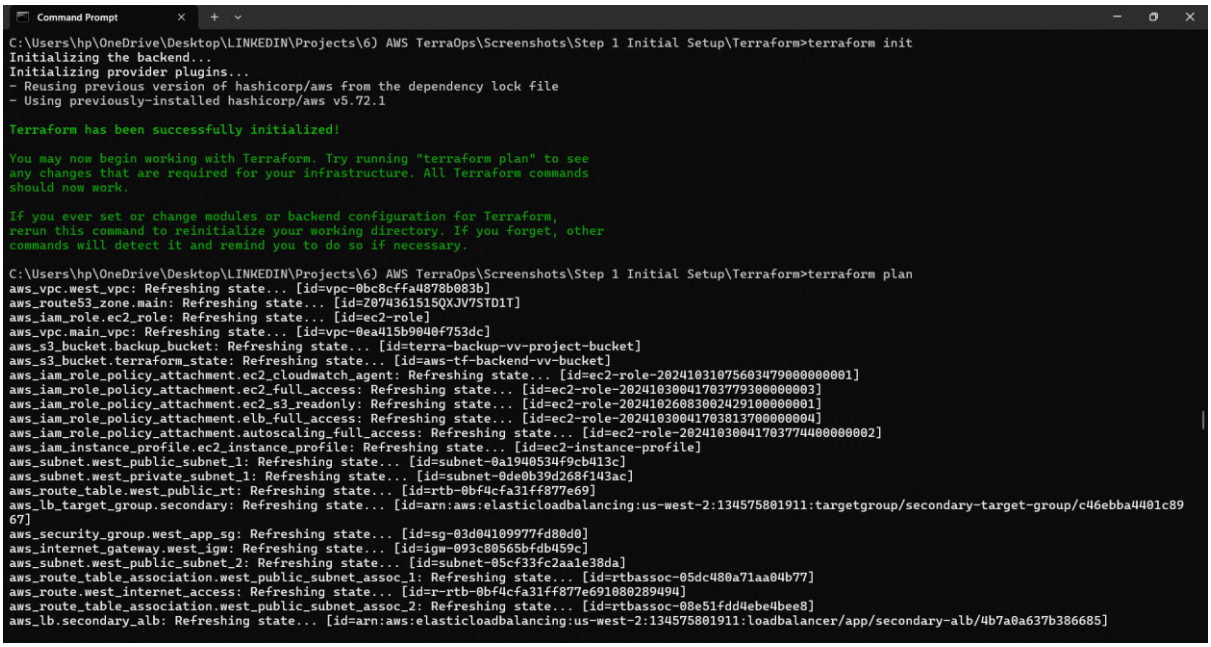
```
# Launch Template for EC2 Instances in us-west-2


resource "aws_launch_template" "secondary_app" {
  provider       = aws.west
  name_prefix    = "secondary-app-launch-template"
  image_id       = "ami-07c5ecd8498c59db5"  # Amazon Linux AMI
  instance_type  = "t2.micro"

  # Attach the IAM role to EC2 instances
  iam_instance_profile {
    name = aws_iam_instance_profile.ec2_instance_profile.name
  }

  network_interfaces {
    security_groups = [aws_security_group.west_app_sg.id]
  }

  user_data = base64encode(<<-EOF
              #!/bin/bash
              sudo dnf update -y
              sudo dnf install -y nginx amazon-cloudwatch-agent
              sudo systemctl start nginx
              sudo systemctl enable nginx

              # Configure the CloudWatch Agent
              sudo tee /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json << 'EOF2'
              {
                "metrics": {
                  "append_dimensions": {
                    "AutoScalingGroupName": "SecondaryAppASGInstance"
                  },
                  "metrics_collected": {
                    "mem": {
                      "measurement": [
                        "mem_used_percent"
                      ],
```

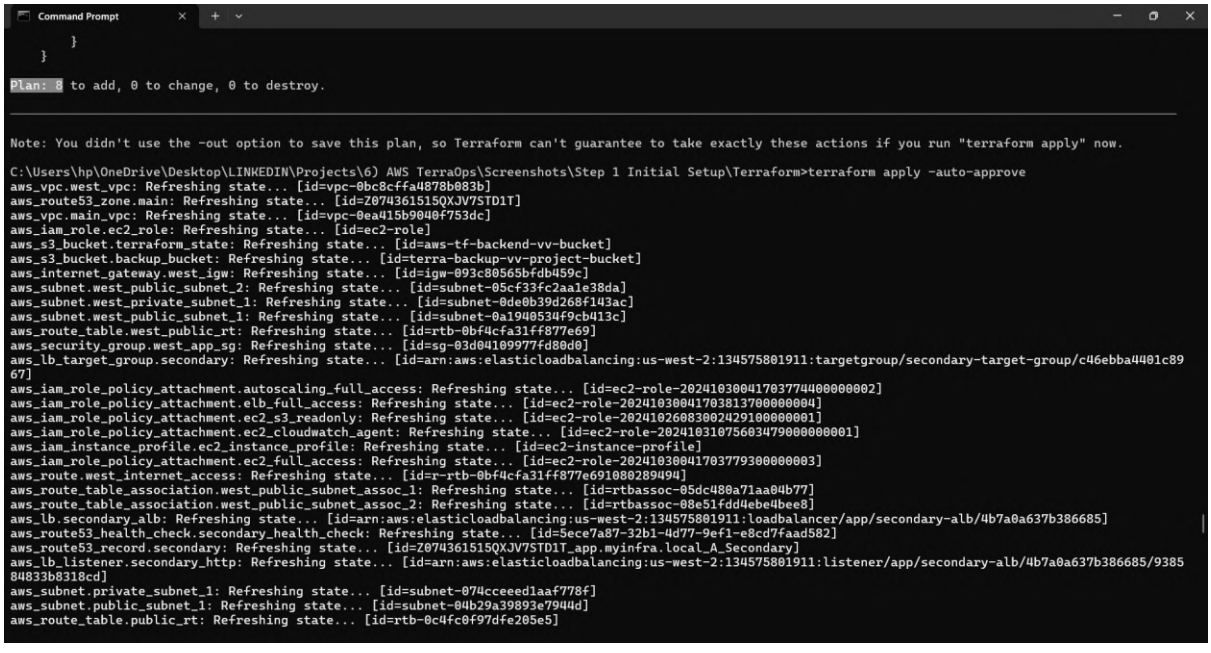

File Edit Selection View Go Run Terminal Help
Terraform
EXPLORER
TERRAFORM
.terraform
.terraform.lock.hcl
alb.tf
ec2_autoscaling.tf
iam_roles.tf
main.tf
network.tf
route53.tf
s3_bucket.tf
terraform.tfstate
terraform.tfstate.backup
OUTLINE
TIMELINE
ec2_autoscaling.tf
ec2_autoscaling.tf > resource "aws_launch_template" "secondary_app" > user_data
resource "aws_launch_template" "secondary_app" {
user_data = base64encode(<<-EOF
sudo tee /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json << 'EOF2'
{
"metrics": {
"append_dimensions": {
"AutoScalingGroupName": "SecondaryAppASGInstance"
},
"metrics_collected": {
"mem": {
"measurement": [
"mem_used_percent"
],
"metrics_collection_interval": 60
},
"disk": {
"measurement": [
"used_percent"
],
"resources": [
"/"
],
"metrics_collection_interval": 60
}
}
}
}
EOF2
# Start the CloudWatch Agent
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a start -c file:/opt/aws/amazon-cloudwatch-agent/etc/amaz
EOF
)
lifecycle {
create_before_destroy = true
}
Ln 261, Col 21 Spaces: 2 UTF-8 CRLF {} Terraform CODEGPT

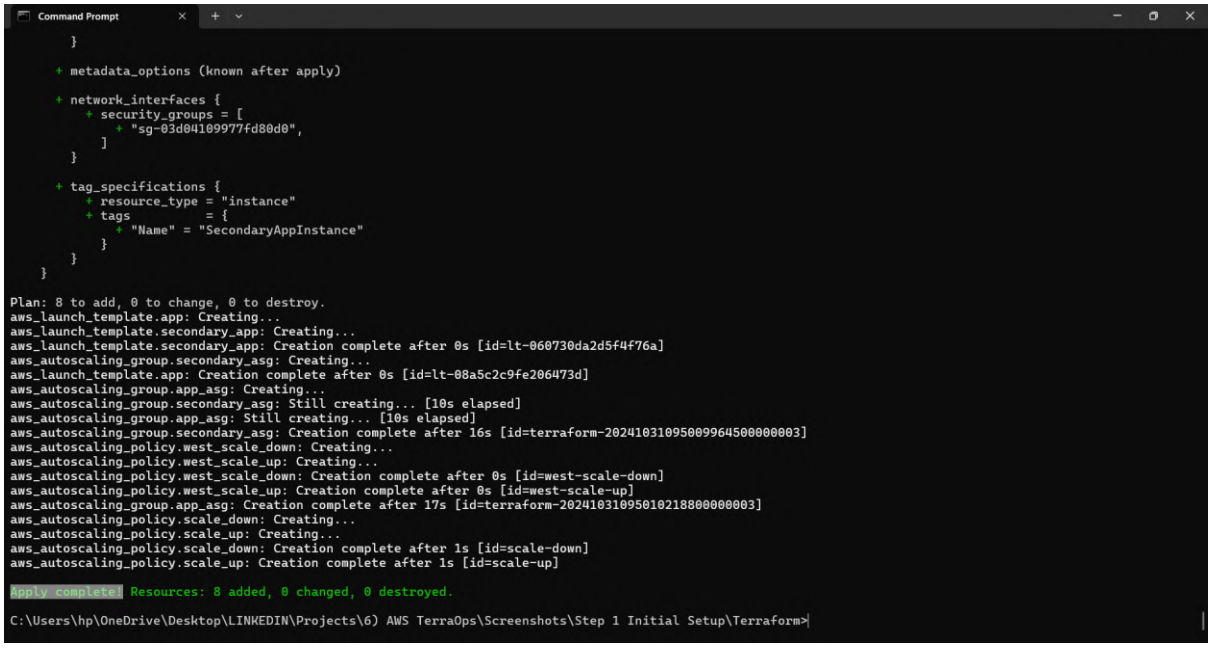

File Edit Selection View Go Run Terminal Help
Terraform
EXPLORER
TERRAFORM
.terraform
.terraform.lock.hcl
alb.tf
ec2_autoscaling.tf
iam_roles.tf
main.tf
network.tf
route53.tf
s3_bucket.tf
terraform.tfstate
terraform.tfstate.backup
OUTLINE
TIMELINE
ec2_autoscaling.tf
ec2_autoscaling.tf > resource "aws_autoscaling_group" "secondary_asg" > instance_refresh
# Auto Scaling Group (ASG) for us-west-2
resource "aws_autoscaling_group" "secondary_asg" {
provider = aws.west
desired_capacity = 2
min_size = 1
max_size = 5
vpc_zone_identifier = [aws_subnet.west_public_subnet_1.id, aws_subnet.west_public_subnet_2.id]
target_group_arns = [aws_lb_target_group.secondary.arn] # Attach to ALB Target Group
# Replacement of instances
instance_refresh {
strategy = "Rolling"
preferences {
min_healthy_percentage = 0
instance_warmup = 120
}
}
health_check_grace_period = 300 # Grace period
health_check_type = "ELB" # Use ELB-based health check for ALB
launch_template {
id = aws_launch_template.secondary_app.id
version = "$Latest"
}
tag {
key = "Name"
value = "SecondaryAppASGInstance"
propagate_at_launch = true
}
Ln 320, Col 4 (177 selected) Spaces: 2 UTF-8 CRLF {} Terraform CODEGPT

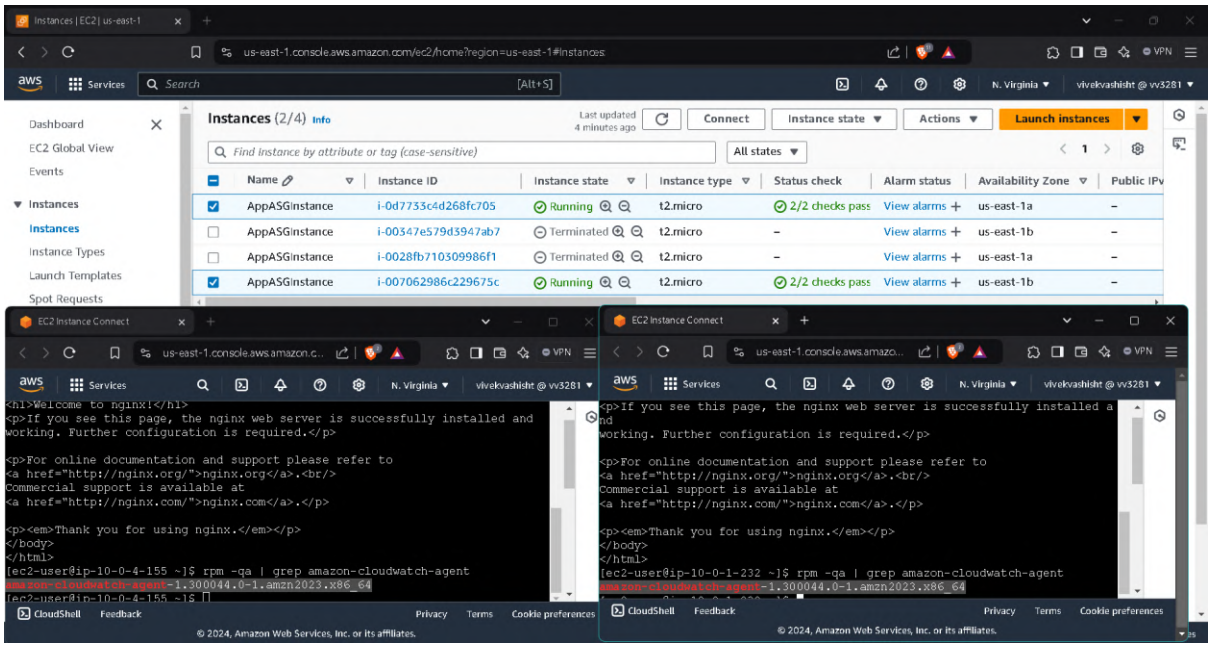

Command Prompt
C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>terraform destroy -target=aws_autoscaling_group.app_asg -target=aws_autoscaling_group.secondary_asg -target=aws_launch_template.app -target=aws_launch_template.secondary_app
aws_vpc.west_vpc: Refreshing state... [id=vpc-0bc8cffa4878b083b]
aws_iam_role.ec2_role: Refreshing state... [id=ec2-role]
aws_vpc.main_vpc: Refreshing state... [id=vpc-0ea415b9040f753dc]
aws_iam_instance_profile.ec2_instance_profile: Refreshing state... [id=ec2-instance-profile]
aws_subnet.west_public_subnet_2: Refreshing state... [id=subnet-05cf33fc2aa1e38da]
aws_subnet.west_public_subnet_1: Refreshing state... [id=subnet-0a1940534f9cb413c]
aws_security_group.west_app_sg: Refreshing state... [id=sg-03d04109977fd80d0]
aws_lb_target_group.secondary: Refreshing state... [id=arn:aws:elasticloadbalancing:us-west-2:134575801911:targetgroup/secondary-target-group/c46ebba4401c8967]
aws_launch_template.secondary_app: Refreshing state... [id=lt-0c625394b30f2746d]
aws_autoscaling_group.secondary_asg: Refreshing state... [id=terraform-20241030094126584900000004]
aws_subnet.public_subnet_2: Refreshing state... [id=subnet-0d89362f71d4a5adf]
aws_subnet.public_subnet_1: Refreshing state... [id=subnet-04b29a39893e7944d]
aws_lb_target_group.primary: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:134575801911:targetgroup/primary-target-group/a9447d338de6a3e5]
aws_security_group.app_sg: Refreshing state... [id=sg-0d02a28cc779890cb]
aws_launch_template.app: Refreshing state... [id=lt-074d209bff44eba96]
aws_autoscaling_group.app_asg: Refreshing state... [id=terraform-20241030094125457300000004]
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
- destroy
Terraform will perform the following actions:
# aws_autoscaling_group.app_asg will be destroyed
- resource "aws_autoscaling_group" "app_asg" {
- arn = "arn:aws:autoscaling:us-east-1:134575801911:autoScalingGroup:71dfa62f-4579-4e12-8e71-ff9cca91e452:autoScalingGroupName/terraform-20241030094125457300000004" -> null
- availability_zones = [
- "us-east-1a",
- "us-east-1b",
] -> null
- capacity_rebalance = false -> null
- default_cooldown = 300 -> null
- default_instance_warmup = 0 -> null
- desired_capacity = 2 -> null
- enabled_metrics = [] -> null
- force_delete = false -> null
- force_delete_warm_pool = false -> null

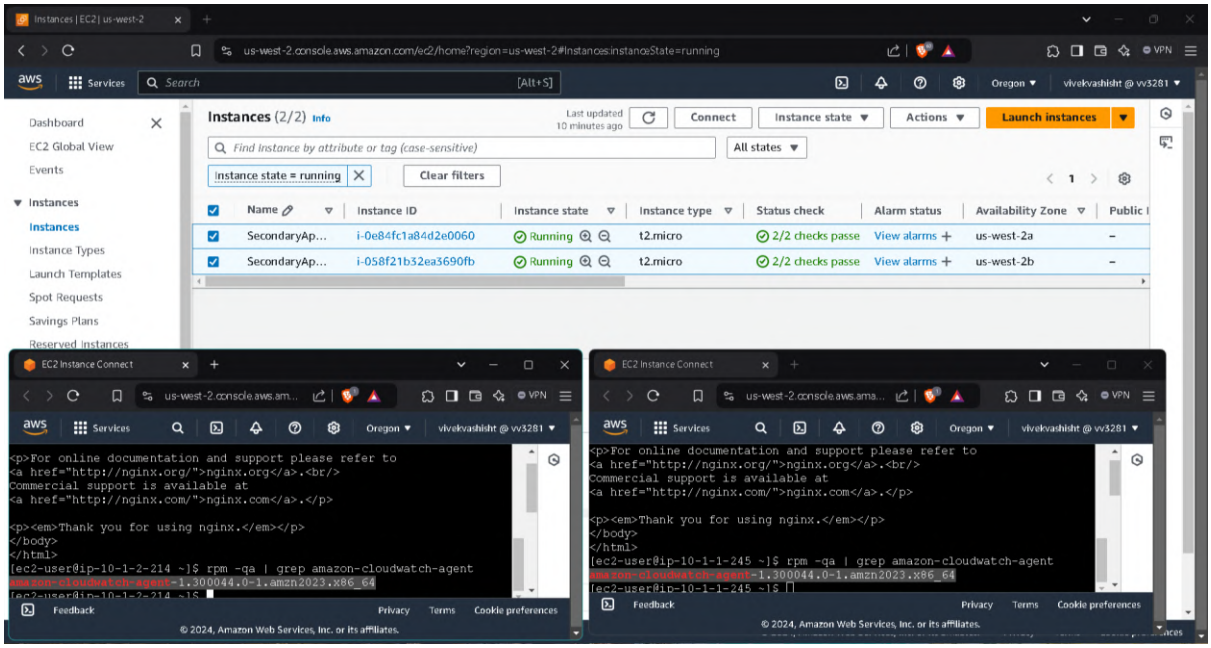

Command Prompt
aws_autoscaling_group.secondary_asg: Still destroying... [id=terraform-20241030094126584900000004, 5m0s elapsed]
aws_autoscaling_group.app_asg: Still destroying... [id=terraform-20241030094125457300000004, 5m1s elapsed]
aws_autoscaling_group.secondary_asg: Still destroying... [id=terraform-20241030094126584900000004, 5m10s elapsed]
aws_autoscaling_group.app_asg: Still destroying... [id=terraform-20241030094125457300000004, 5m11s elapsed]
aws_autoscaling_group.secondary_asg: Still destroying... [id=terraform-20241030094126584900000004, 5m20s elapsed]
aws_autoscaling_group.app_asg: Still destroying... [id=terraform-20241030094125457300000004, 5m21s elapsed]
aws_autoscaling_group.secondary_asg: Still destroying... [id=terraform-20241030094126584900000004, 5m30s elapsed]
aws_autoscaling_group.app_asg: Still destroying... [id=terraform-20241030094125457300000004, 5m31s elapsed]
aws_autoscaling_group.secondary_asg: Still destroying... [id=terraform-20241030094126584900000004, 5m40s elapsed]
aws_autoscaling_group.app_asg: Still destroying... [id=terraform-20241030094125457300000004, 5m41s elapsed]
aws_autoscaling_group.secondary_asg: Still destroying... [id=terraform-20241030094126584900000004, 5m50s elapsed]
aws_autoscaling_group.app_asg: Still destroying... [id=terraform-20241030094125457300000004, 5m51s elapsed]
aws_autoscaling_group.secondary_asg: Still destroying... [id=terraform-20241030094126584900000004, 6m0s elapsed]
aws_autoscaling_group.app_asg: Still destroying... [id=terraform-20241030094125457300000004, 6m1s elapsed]
aws_autoscaling_group.secondary_asg: Still destroying... [id=terraform-20241030094126584900000004, 6m10s elapsed]
aws_autoscaling_group.app_asg: Still destroying... [id=terraform-20241030094125457300000004, 6m11s elapsed]
aws_autoscaling_group.secondary_asg: Still destroying... [id=terraform-20241030094126584900000004, 6m20s elapsed]
aws_autoscaling_group.app_asg: Still destroying... [id=terraform-20241030094125457300000004, 6m21s elapsed]
aws_autoscaling_group.secondary_asg: Still destroying... [id=terraform-20241030094126584900000004, 6m30s elapsed]
aws_autoscaling_group.app_asg: Still destroying... [id=terraform-20241030094125457300000004, 6m31s elapsed]
aws_autoscaling_group.app_asg: Destruction complete after 6m36s
aws_launch_template.app: Destroying... [id=lt-074d209bff44eba96]
aws_launch_template.app: Destruction complete after 0s
aws_autoscaling_group.secondary_asg: Still destroying... [id=terraform-20241030094126584900000004, 6m40s elapsed]
aws_autoscaling_group.secondary_asg: Still destroying... [id=terraform-20241030094126584900000004, 6m50s elapsed]
aws_autoscaling_group.secondary_asg: Destruction complete after 6m51s
aws_launch_template.secondary_app: Destroying... [id=lt-0c625394b30f2746d]
aws_launch_template.secondary_app: Destruction complete after 0s
Warning: Applied changes may be incomplete
The plan was created with the -target option in effect, so some changes requested in the configuration may have been ignored and the output values may not be fully updated. Run the following command to verify that no other changes are pending:
terraform plan
Note that the -target option is not suitable for routine use, and is provided only for exceptional situations such as recovering from errors or mistakes, or when Terraform specifically suggests to use it as part of an error message.
Destroy complete! Resources: 8 destroyed.

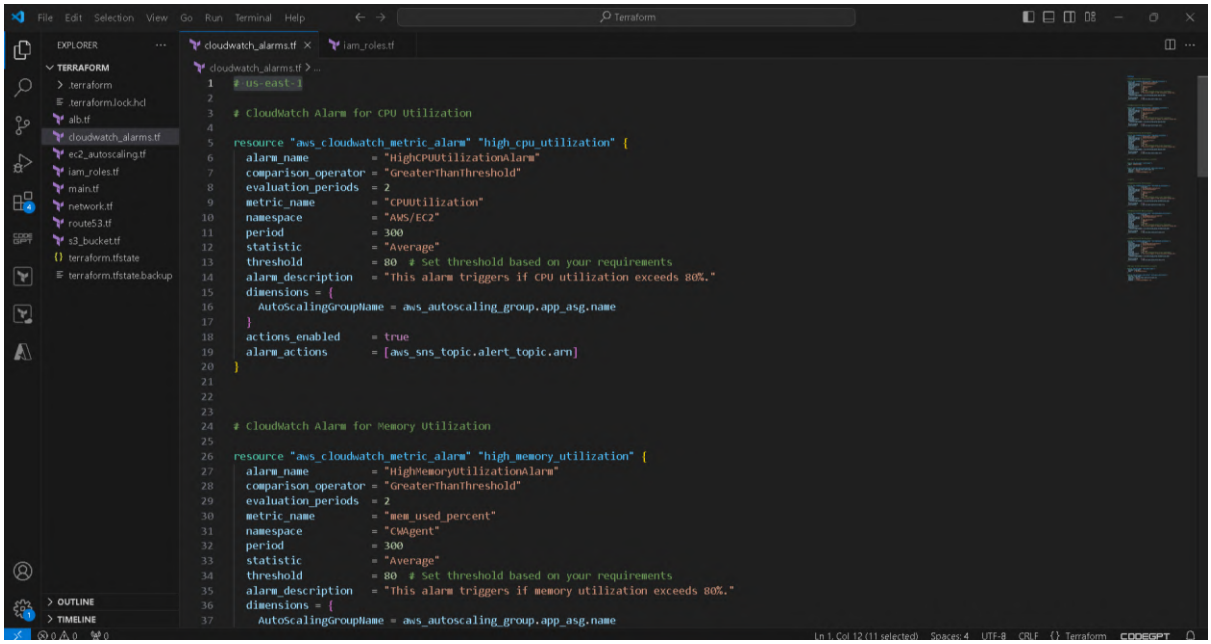

Command Prompt
C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.72.1
Terraform has been successfully initialized!
You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.
If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>terraform plan
aws_vpc.west_vpc: Refreshing state... [id=vpc-0bc8cffa4878b083b]
aws_route53_zone.main: Refreshing state... [id=Z0743615150XJV7STD1T]
aws_iam_role.ec2_role: Refreshing state... [id=ec2-role]
aws_vpc.main_vpc: Refreshing state... [id=vpc-0ea415b90400f753dc]
aws_s3_bucket.backup_bucket: Refreshing state... [id=terra-backup-vv-project-bucket]
aws_s3_bucket.terraform_state: Refreshing state... [id=aws-tf-backend-vv-bucket]
aws_iam_role_policy_attachment.ec2_cloudwatch_agent: Refreshing state... [id=ec2-role-20241031075603479000000001]
aws_iam_role_policy_attachment.ec2_full_access: Refreshing state... [id=ec2-role-20241030041703779300000003]
aws_iam_role_policy_attachment.ec2_s3_readonly: Refreshing state... [id=ec2-role-20241026083002429100000001]
aws_iam_role_policy_attachment.elb_full_access: Refreshing state... [id=ec2-role-20241030041703813700000004]
aws_iam_role_policy_attachment.autoscaling_full_access: Refreshing state... [id=ec2-role-20241030041703774400000002]
aws_iam_instance_profile.ec2_instance_profile: Refreshing state... [id=ec2-instance-profile]
aws_subnet.west_public_subnet_1: Refreshing state... [id=subnet-0a1940534f9cb413c]
aws_subnet.west_private_subnet_1: Refreshing state... [id=subnet-0de0b39d268f143ac]
aws_route_table.west_public_rt: Refreshing state... [id=rtb-0bf4cfa31ff877e69]
aws_lb_target_group.secondary: Refreshing state... [id=arn:aws:elasticloadbalancing:us-west-2:134575801911:targetgroup/secondary-target-group/c46ebba4401c8967]
aws_security_group.west_app_sg: Refreshing state... [id=sg-03d04109977fd80d0]
aws_internet_gateway.west_igw: Refreshing state... [id=igw-093c80565bfdb459c]
aws_subnet.west_public_subnet_2: Refreshing state... [id=subnet-05cf33fc2aa1e38da]
aws_route_table_association.west_public_subnet_assoc_1: Refreshing state... [id=rtbassoc-05dc480a71aa04b77]
aws_route.west_internet_access: Refreshing state... [id=r-rtb-0bf4cfa31ff877e691080289494]
aws_route_table_association.west_public_subnet_assoc_2: Refreshing state... [id=rtbassoc-08e51fdd4ebe4bee8]
aws_lb.secondary_alb: Refreshing state... [id=arn:aws:elasticloadbalancing:us-west-2:134575801911:loadbalancer/app/secondary-alb/4b7a0a637b386685]


Command Prompt
}
}
Plan: 8 to add, 0 to change, 0 to destroy.
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.
C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>terraform apply -auto-approve
aws_vpc.west_vpc: Refreshing state... [id=vpc-0bc8cffa4878b083b]
aws_route53_zone.main: Refreshing state... [id=Z0743615150XJV7STD1T]
aws_vpc.main_vpc: Refreshing state... [id=vpc-0ea415b90400f753dc]
aws_iam_role.ec2_role: Refreshing state... [id=ec2-role]
aws_s3_bucket.terraform_state: Refreshing state... [id=aws-tf-backend-vv-bucket]
aws_s3_bucket.backup_bucket: Refreshing state... [id=terra-backup-vv-project-bucket]
aws_internet_gateway.west_igw: Refreshing state... [id=igw-093c80565bfdb459c]
aws_subnet.west_public_subnet_2: Refreshing state... [id=subnet-05cf33fc2aa1e38da]
aws_subnet.west_private_subnet_1: Refreshing state... [id=subnet-0de0b39d268f143ac]
aws_subnet.west_public_subnet_1: Refreshing state... [id=subnet-0a1940534f9cb413c]
aws_route_table.west_public_rt: Refreshing state... [id=rtb-0bf4cfa31ff877e69]
aws_security_group.west_app_sg: Refreshing state... [id=sg-03d04109977fd80d0]
aws_lb_target_group.secondary: Refreshing state... [id=arn:aws:elasticloadbalancing:us-west-2:134575801911:targetgroup/secondary-target-group/c46ebba4401c8967]
aws_iam_role_policy_attachment.autoscaling_full_access: Refreshing state... [id=ec2-role-20241030041703774400000002]
aws_iam_role_policy_attachment.elb_full_access: Refreshing state... [id=ec2-role-20241030041703813700000004]
aws_iam_role_policy_attachment.ec2_s3_readonly: Refreshing state... [id=ec2-role-20241026083002429100000001]
aws_iam_role_policy_attachment.ec2_cloudwatch_agent: Refreshing state... [id=ec2-role-20241031075603479000000001]
aws_iam_instance_profile.ec2_instance_profile: Refreshing state... [id=ec2-instance-profile]
aws_iam_role_policy_attachment.ec2_full_access: Refreshing state... [id=ec2-role-20241030041703779300000003]
aws_route.west_internet_access: Refreshing state... [id=r-rtb-0bf4cfa31ff877e691080289494]
aws_route_table_association.west_public_subnet_assoc_1: Refreshing state... [id=rtbassoc-05dc480a71aa04b77]
aws_route_table_association.west_public_subnet_assoc_2: Refreshing state... [id=rtbassoc-08e51fdd4ebe4bee8]
aws_lb.secondary_alb: Refreshing state... [id=arn:aws:elasticloadbalancing:us-west-2:134575801911:loadbalancer/app/secondary-alb/4b7a0a637b386685]
aws_route53_health_check.secondary_health_check: Refreshing state... [id=5ece7a87-32b1-4d77-9ef1-e8cd7faad582]
aws_route53_record.secondary: Refreshing state... [id=Z0743615150XJV7STD1T_app.myinfra.local_A_Secondary]
aws_lb_listener.secondary_http: Refreshing state... [id=arn:aws:elasticloadbalancing:us-west-2:134575801911:listener/app/secondary-alb/4b7a0a637b386685/938584833b8318cd]
aws_subnet.private_subnet_1: Refreshing state... [id=subnet-074cceeed1aaf778f]
aws_subnet.public_subnet_1: Refreshing state... [id=subnet-04b29a39893e7944d]
aws_route_table.public_rt: Refreshing state... [id=rtb-0c4fc0f97dfe205e5]

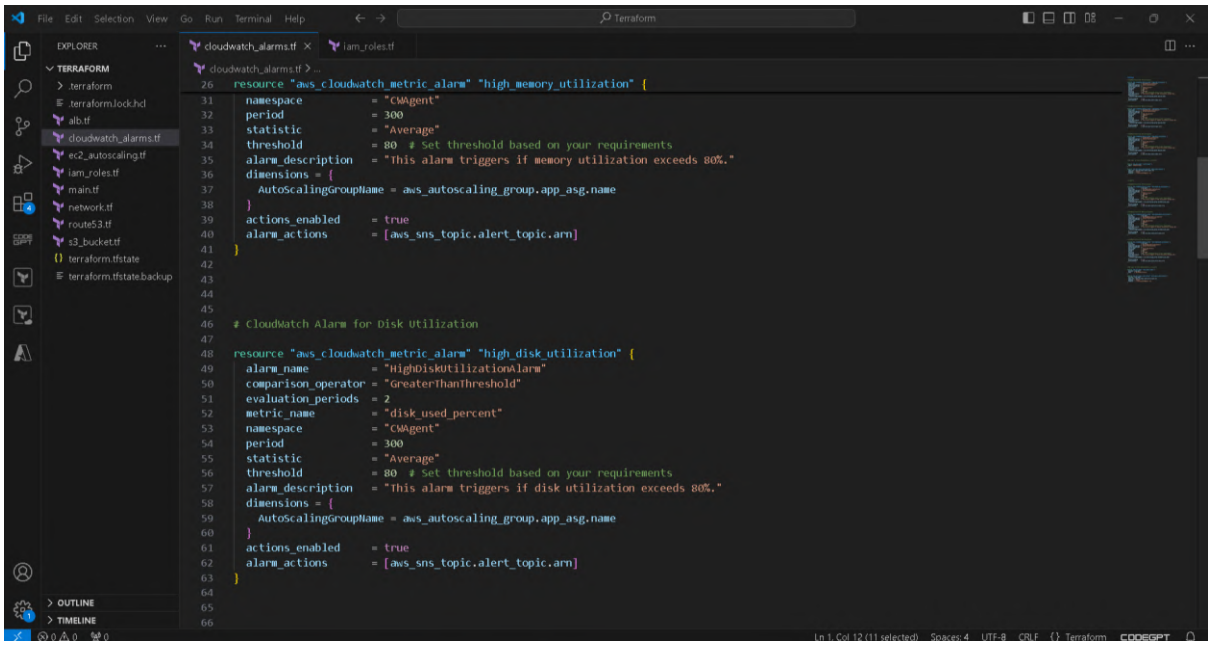

Command Prompt
}
+ metadata_options (known after apply)
+ network_interfaces {
+ security_groups = [
+ "sg-03d04109977fd80d0",
]
}
+ tag_specifications {
+ resource_type = "instance"
+ tags = {
+ "Name" = "SecondaryAppInstance"
}
}
}
Plan: 8 to add, 0 to change, 0 to destroy.
aws_launch_template.app: Creating...
aws_launch_template.secondary_app: Creating...
aws_launch_template.secondary_app: Creation complete after 0s [id=lt-060730da2d5f4f76a]
aws_autoscaling_group.secondary_asg: Creating...
aws_launch_template.app: Creation complete after 0s [id=lt-08a5c2c9fe206473d]
aws_autoscaling_group.app_asg: Creating...
aws_autoscaling_group.secondary_asg: Still creating... [10s elapsed]
aws_autoscaling_group.app_asg: Still creating... [10s elapsed]
aws_autoscaling_group.secondary_asg: Creation complete after 16s [id=terraform-20241031095009964500000003]
aws_autoscaling_policy.west_scale_down: Creating...
aws_autoscaling_policy.west_scale_up: Creating...
aws_autoscaling_policy.west_scale_down: Creation complete after 0s [id=west-scale-down]
aws_autoscaling_policy.west_scale_up: Creation complete after 0s [id=west-scale-up]
aws_autoscaling_group.app_asg: Creation complete after 17s [id=terraform-20241031095010218800000003]
aws_autoscaling_policy.scale_down: Creating...
aws_autoscaling_policy.scale_up: Creating...
aws_autoscaling_policy.scale_down: Creation complete after 1s [id=scale-down]
aws_autoscaling_policy.scale_up: Creation complete after 1s [id=scale-up]
Apply complete! Resources: 8 added, 0 changed, 0 destroyed.
C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>


Instances (2/4) Info
Launch instances
AppASGinstance
i-0d7733c4d268fc705
i-00347e579d3947ab7
i-0028fb710309986f1
i-007062986c229675c
[ec2-user@ip-10-0-4-155 ~]$ rpm -qa | grep amazon-cloudwatch-agent
amazon-cloudwatch-agent-1.300044.0-1.amzn2023.x86_64
[ec2-user@ip-10-0-1-232 ~]$ rpm -qa | grep amazon-cloudwatch-agent
amazon-cloudwatch-agent-1.300044.0-1.amzn2023.x86_64

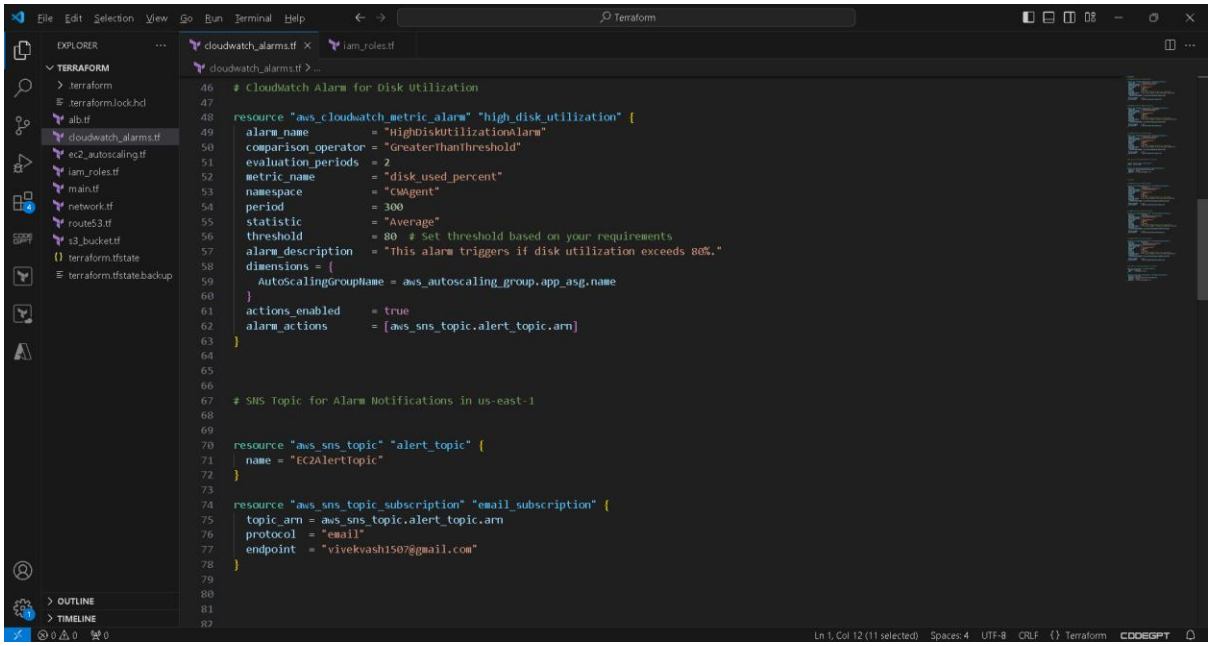

Instances (2/2) Info
Launch instances
SecondaryAp...
i-0e84fc1a84d2e0060
i-058f21b32ea3690fb
[ec2-user@ip-10-1-2-214 ~]$ rpm -qa | grep amazon-cloudwatch-agent
amazon-cloudwatch-agent-1.300044.0-1.amzn2023.x86_64
[ec2-user@ip-10-1-1-245 ~]$ rpm -qa | grep amazon-cloudwatch-agent
amazon-cloudwatch-agent-1.300044.0-1.amzn2023.x86_64



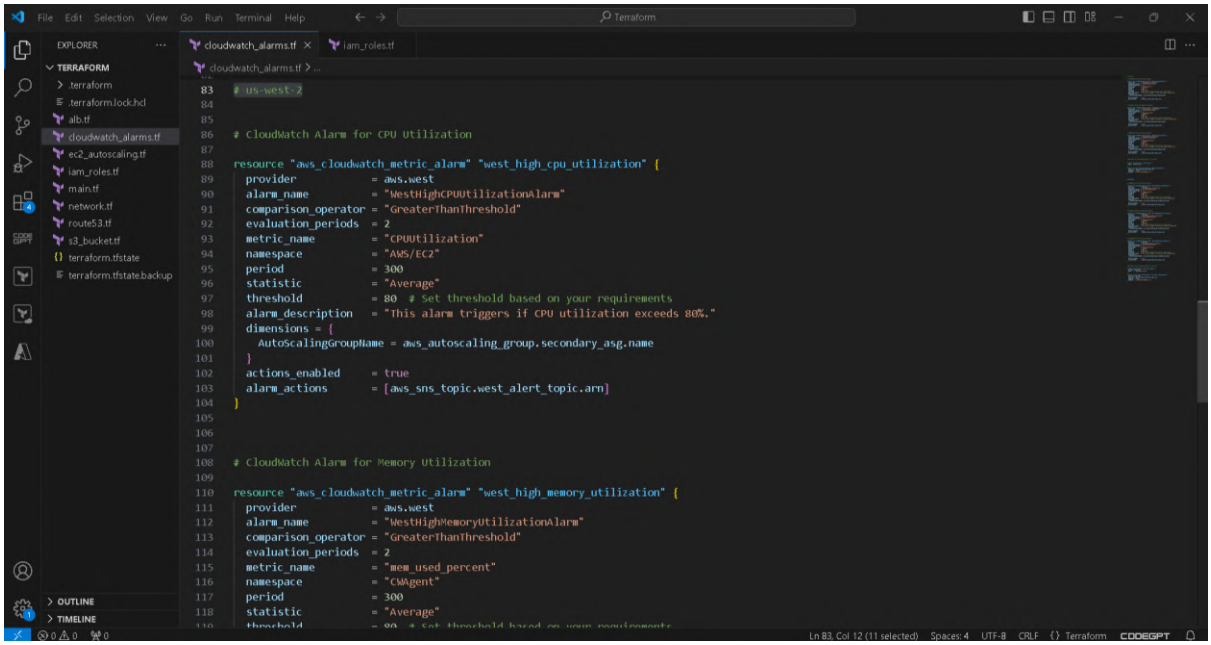
```
# us-east-1

# CloudWatch Alarm for CPU Utilization

resource "aws_cloudwatch_metric_alarm" "high_cpu_utilization" {
  alarm_name          = "HighCPUUtilizationAlarm"
  comparison_operator = "GreaterThanThreshold"
  evaluation_periods  = 2
  metric_name         = "CPUUtilization"
  namespace           = "AWS/EC2"
  period              = 300
  statistic           = "Average"
  threshold           = 80  # Set threshold based on your requirements
  alarm_description   = "This alarm triggers if CPU utilization exceeds 80%."
  dimensions = {
    AutoScalingGroupName = aws_autoscaling_group.app_asg.name
  }
  actions_enabled     = true
  alarm_actions       = [aws_sns_topic.alert_topic.arn]
}



# CloudWatch Alarm for Memory Utilization

resource "aws_cloudwatch_metric_alarm" "high_memory_utilization" {
  alarm_name          = "HighMemoryUtilizationAlarm"
  comparison_operator = "GreaterThanThreshold"
  evaluation_periods  = 2
  metric_name         = "mem_used_percent"
  namespace           = "CWAgent"
  period              = 300
  statistic           = "Average"
  threshold           = 80  # Set threshold based on your requirements
  alarm_description   = "This alarm triggers if memory utilization exceeds 80%."
  dimensions = {
    AutoScalingGroupName = aws_autoscaling_group.app_asg.name
```



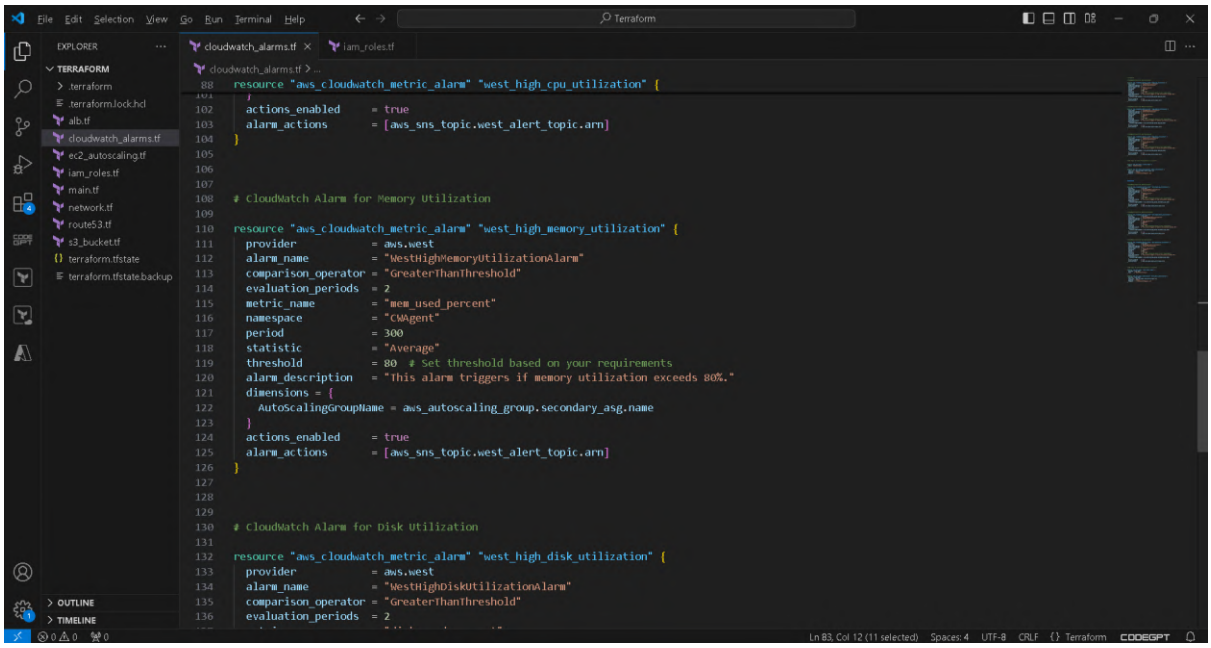
```
resource "aws_cloudwatch_metric_alarm" "high_memory_utilization" {
  namespace           = "CWAgent"
  period              = 300
  statistic           = "Average"
  threshold           = 80  # Set threshold based on your requirements
  alarm_description   = "This alarm triggers if memory utilization exceeds 80%."
  dimensions = {
    AutoScalingGroupName = aws_autoscaling_group.app_asg.name
  }
  actions_enabled     = true
  alarm_actions       = [aws_sns_topic.alert_topic.arn]
}



# CloudWatch Alarm for Disk Utilization

resource "aws_cloudwatch_metric_alarm" "high_disk_utilization" {
  alarm_name          = "HighDiskUtilizationAlarm"
  comparison_operator = "GreaterThanThreshold"
  evaluation_periods  = 2
  metric_name         = "disk_used_percent"
  namespace           = "CWAgent"
  period              = 300
  statistic           = "Average"
  threshold           = 80  # Set threshold based on your requirements
  alarm_description   = "This alarm triggers if disk utilization exceeds 80%."
  dimensions = {
    AutoScalingGroupName = aws_autoscaling_group.app_asg.name
  }
  actions_enabled     = true
  alarm_actions       = [aws_sns_topic.alert_topic.arn]
}
```



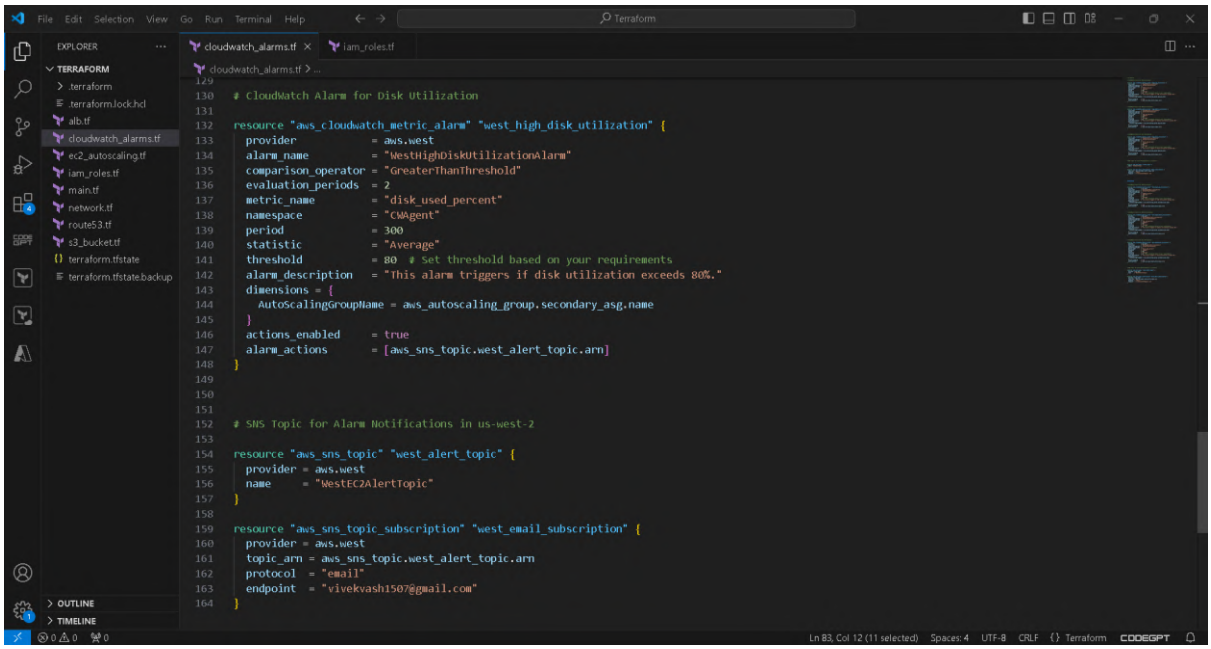
```
# CloudWatch Alarm for Disk Utilization

resource "aws_cloudwatch_metric_alarm" "high_disk_utilization" {
  alarm_name          = "HighDiskUtilizationAlarm"
  comparison_operator = "GreaterThanThreshold"
  evaluation_periods  = 2
  metric_name         = "disk_used_percent"
  namespace           = "CWAgent"
  period              = 300
  statistic           = "Average"
  threshold           = 80  # Set threshold based on your requirements
  alarm_description   = "This alarm triggers if disk utilization exceeds 80%."
  dimensions = {
    AutoScalingGroupName = aws_autoscaling_group.app_asg.name
  }
  actions_enabled     = true
  alarm_actions       = [aws_sns_topic.alert_topic.arn]
}



# SNS Topic for Alarm Notifications in us-east-1


resource "aws_sns_topic" "alert_topic" {
  name = "EC2AlertTopic"
}

resource "aws_sns_topic_subscription" "email_subscription" {
  topic_arn = aws_sns_topic.alert_topic.arn
  protocol  = "email"
  endpoint  = "vivekvash1507@gmail.com"
}
```



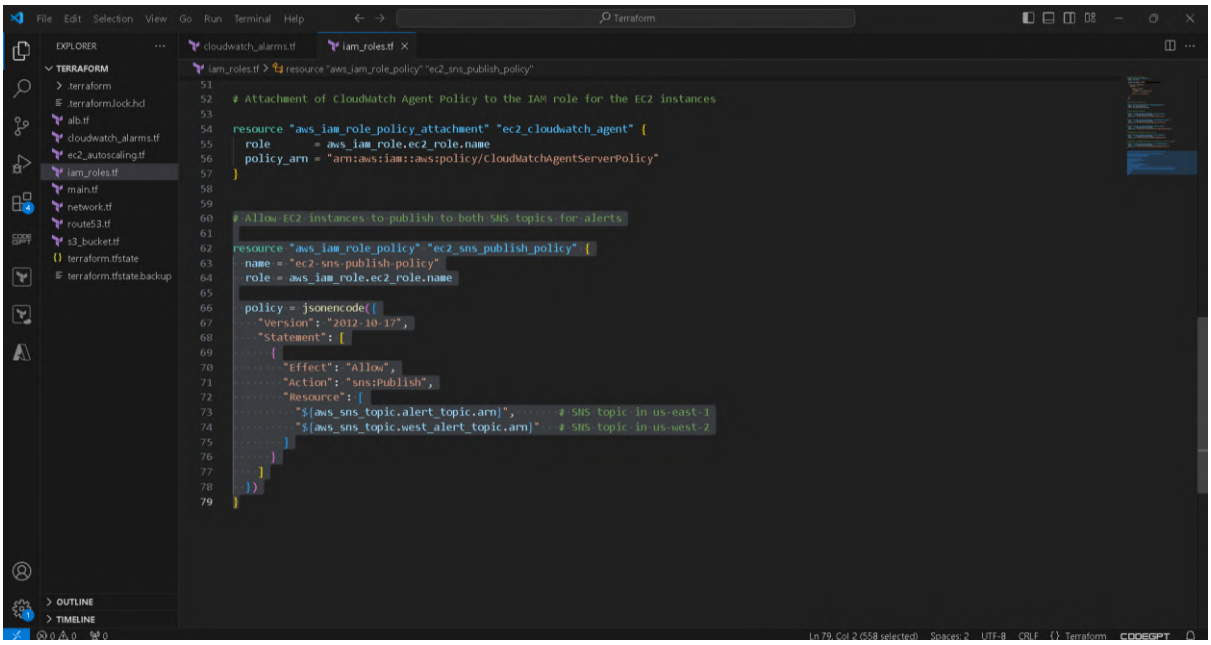
```
# us-west-2


# CloudWatch Alarm for CPU Utilization

resource "aws_cloudwatch_metric_alarm" "west_high_cpu_utilization" {
  provider            = aws.west
  alarm_name          = "WestHighCPUUtilizationAlarm"
  comparison_operator = "GreaterThanThreshold"
  evaluation_periods  = 2
  metric_name         = "CPUUtilization"
  namespace           = "AWS/EC2"
  period              = 300
  statistic           = "Average"
  threshold           = 80  # Set threshold based on your requirements
  alarm_description   = "This alarm triggers if CPU utilization exceeds 80%."
  dimensions = {
    AutoScalingGroupName = aws_autoscaling_group.secondary_asg.name
  }
  actions_enabled     = true
  alarm_actions       = [aws_sns_topic.west_alert_topic.arn]
}



# CloudWatch Alarm for Memory Utilization

resource "aws_cloudwatch_metric_alarm" "west_high_memory_utilization" {
  provider            = aws.west
  alarm_name          = "WestHighMemoryUtilizationAlarm"
  comparison_operator = "GreaterThanThreshold"
  evaluation_periods  = 2
  metric_name         = "mem_used_percent"
  namespace           = "CWAgent"
  period              = 300
  statistic           = "Average"
  threshold           = 80  # Set threshold based on your requirements
```



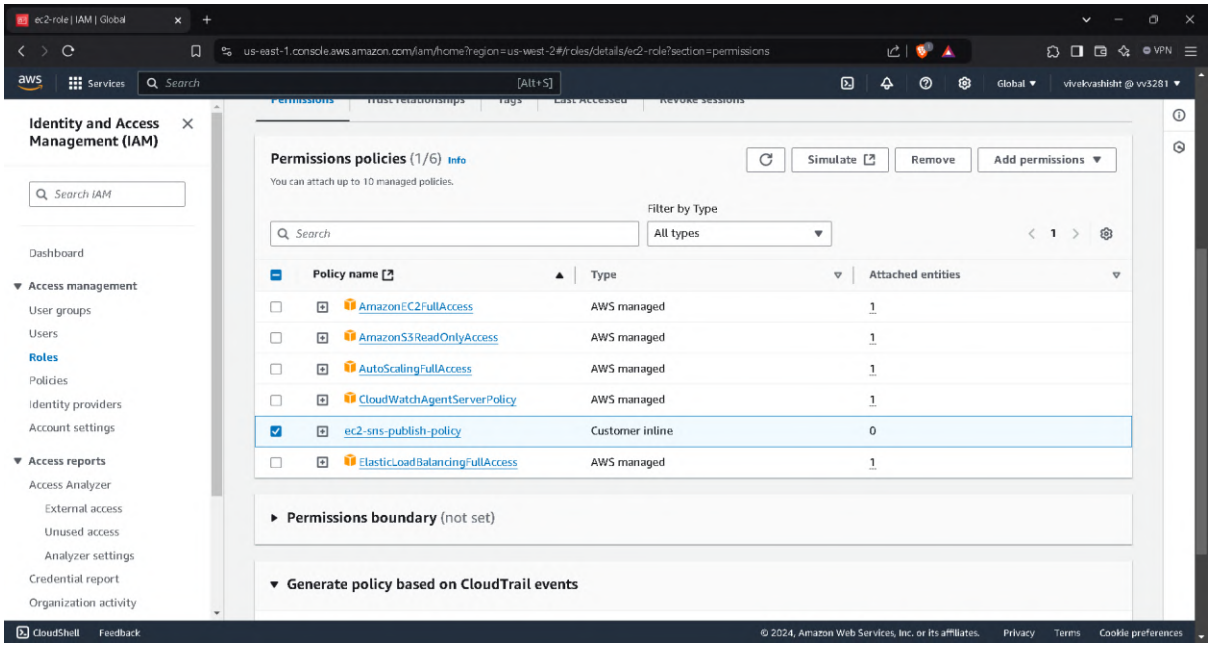
```
resource "aws_cloudwatch_metric_alarm" "west_high_cpu_utilization" {
  actions_enabled     = true
  alarm_actions       = [aws_sns_topic.west_alert_topic.arn]
}



# CloudWatch Alarm for Memory Utilization

resource "aws_cloudwatch_metric_alarm" "west_high_memory_utilization" {
  provider            = aws.west
  alarm_name          = "WestHighMemoryUtilizationAlarm"
  comparison_operator = "GreaterThanThreshold"
  evaluation_periods  = 2
  metric_name         = "mem_used_percent"
  namespace           = "CWAgent"
  period              = 300
  statistic           = "Average"
  threshold           = 80  # Set threshold based on your requirements
  alarm_description   = "This alarm triggers if memory utilization exceeds 80%."
  dimensions = {
    AutoScalingGroupName = aws_autoscaling_group.secondary_asg.name
  }
  actions_enabled     = true
  alarm_actions       = [aws_sns_topic.west_alert_topic.arn]
}



# CloudWatch Alarm for Disk Utilization

resource "aws_cloudwatch_metric_alarm" "west_high_disk_utilization" {
  provider            = aws.west
  alarm_name          = "WestHighDiskUtilizationAlarm"
  comparison_operator = "GreaterThanThreshold"
  evaluation_periods  = 2
```



```
# CloudWatch Alarm for Disk Utilization

resource "aws_cloudwatch_metric_alarm" "west_high_disk_utilization" {
  provider            = aws.west
  alarm_name          = "WestHighDiskUtilizationAlarm"
  comparison_operator = "GreaterThanThreshold"
  evaluation_periods  = 2
  metric_name         = "disk_used_percent"
  namespace           = "CWAgent"
  period              = 300
  statistic           = "Average"
  threshold           = 80  # Set threshold based on your requirements
  alarm_description   = "This alarm triggers if disk utilization exceeds 80%."
  dimensions = {
    AutoScalingGroupName = aws_autoscaling_group.secondary_asg.name
  }
  actions_enabled     = true
  alarm_actions       = [aws_sns_topic.west_alert_topic.arn]
}



# SNS Topic for Alarm Notifications in us-west-2

resource "aws_sns_topic" "west_alert_topic" {
  provider = aws.west
  name     = "WestEC2AlertTopic"
}

resource "aws_sns_topic_subscription" "west_email_subscription" {
  provider = aws.west
  topic_arn = aws_sns_topic.west_alert_topic.arn
  protocol  = "email"
  endpoint  = "vivekvash1507@gmail.com"
}
```



```
# Attachment of CloudWatch Agent Policy to the IAM role for the EC2 instances

resource "aws_iam_role_policy_attachment" "ec2_cloudwatch_agent" {
  role       = aws_iam_role.ec2_role.name
  policy_arn = "arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy"
}


# Allow EC2 instances to publish to both SNS topics for alerts

resource "aws_iam_role_policy" "ec2_sns_publish_policy" {
  name = "ec2-sns-publish-policy"
  role = aws_iam_role.ec2_role.name

  policy = jsonencode({
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": "sns:Publish",
        "Resource": [
          "${aws_sns_topic.alert_topic.arn}",       # SNS topic in us-east-1
          "${aws_sns_topic.west_alert_topic.arn}"   # SNS topic in us-west-2
        ]
      }
    ]
  })
}
```



Identity and Access Management (IAM)

Permissions policies (1/6) Info

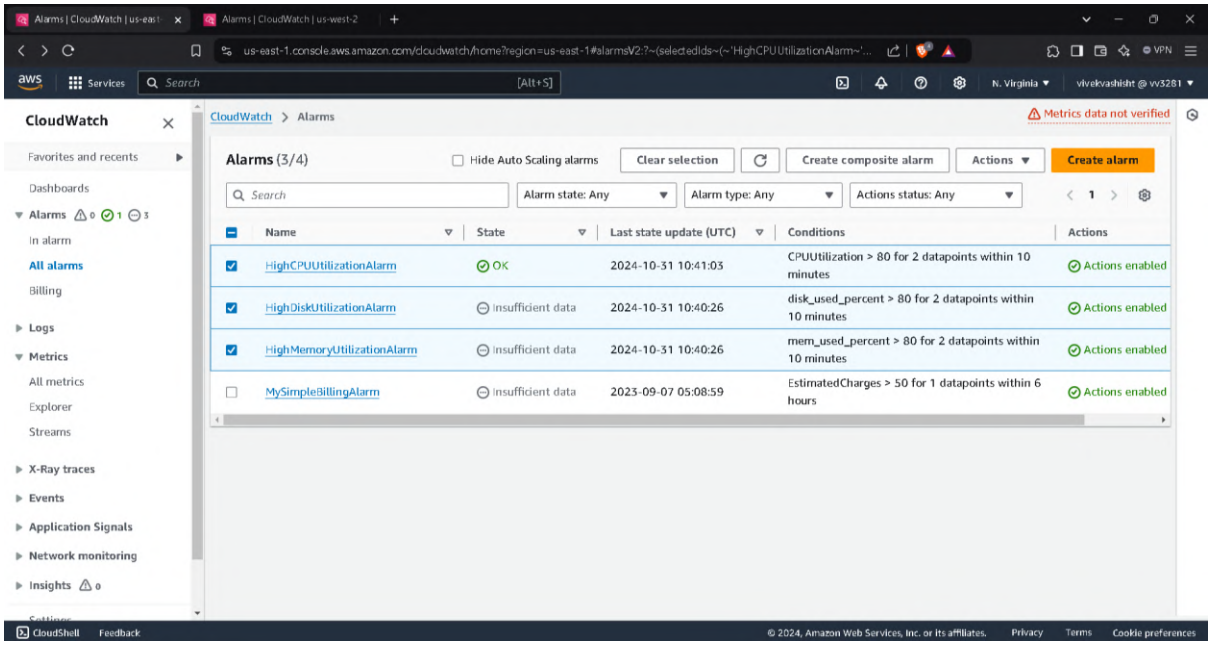
You can attach up to 10 managed policies.

Simulate | Remove | Add permissions

Policy name	Type	Attached entities
AmazonEC2FullAccess	AWS managed	1
AmazonS3ReadOnlyAccess	AWS managed	1
AutoScalingFullAccess	AWS managed	1
CloudWatchAgentServerPolicy	AWS managed	1
ec2-sns-publish-policy	Customer inline	0
ElasticLoadBalancingFullAccess	AWS managed	1

Permissions boundary (not set)

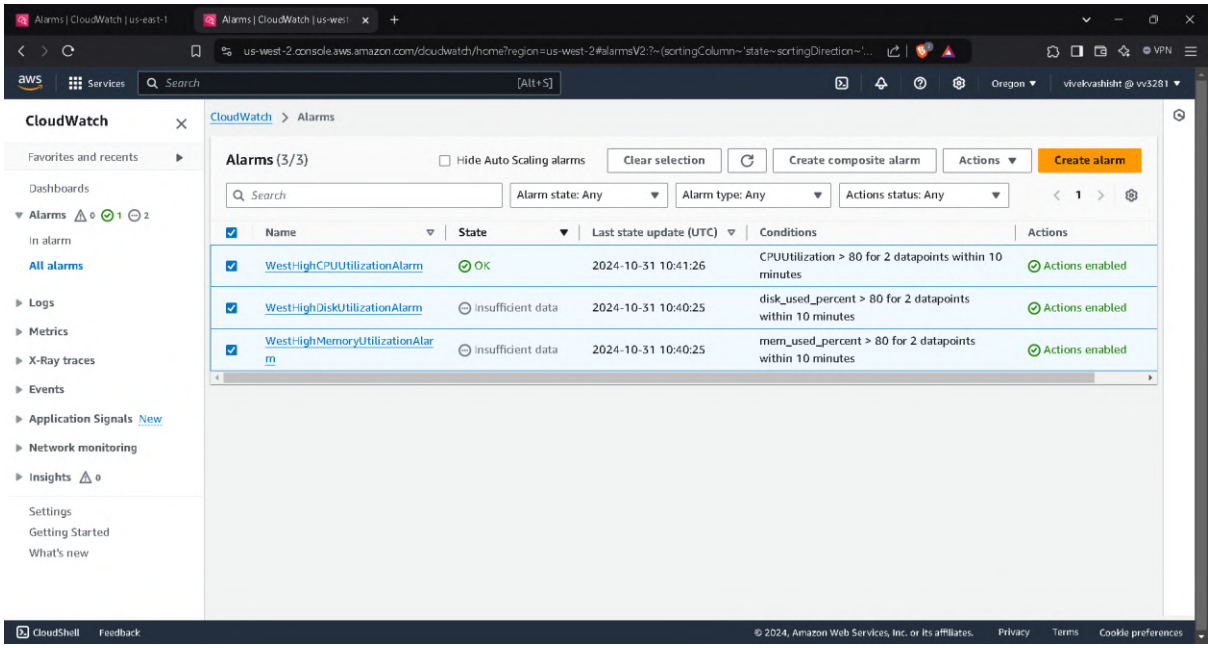
Generate policy based on CloudTrail events

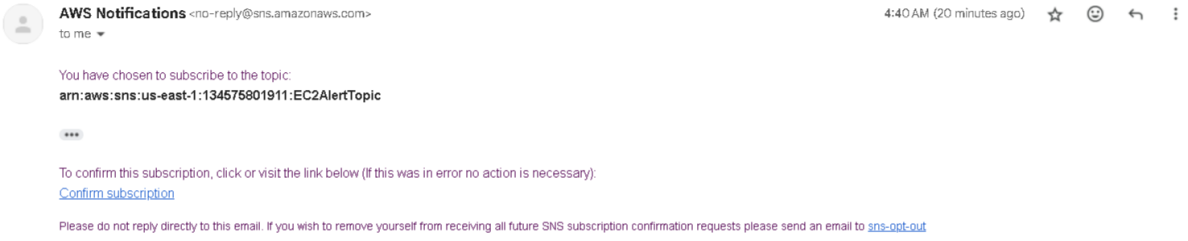


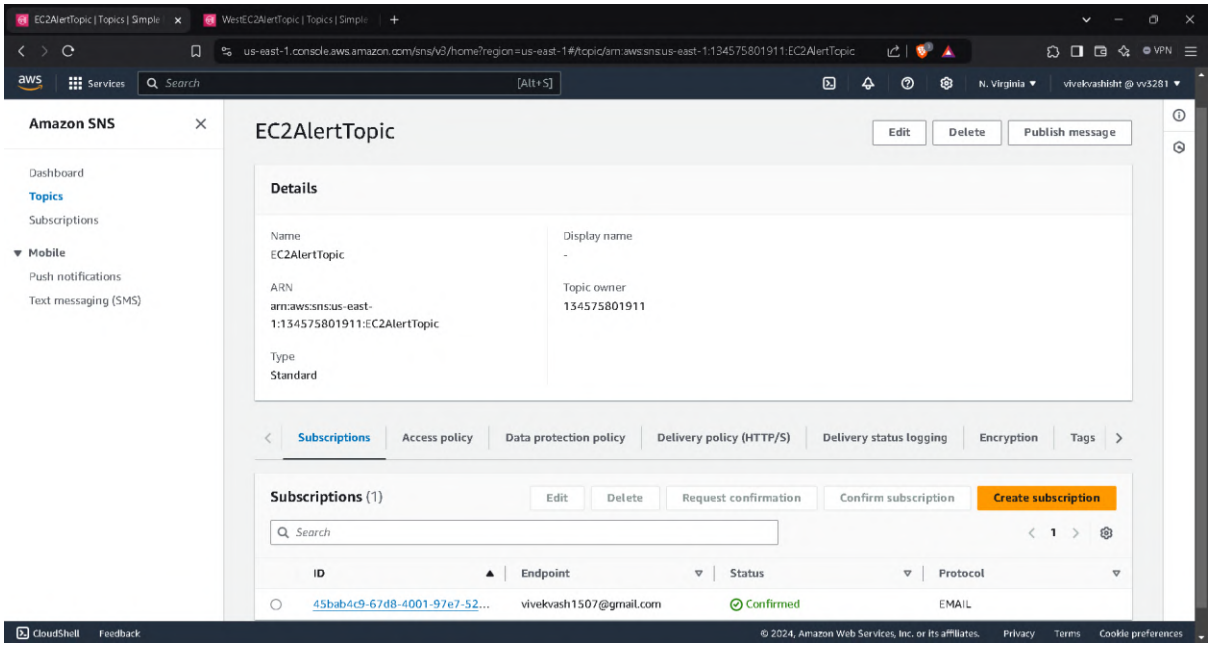
CloudWatch

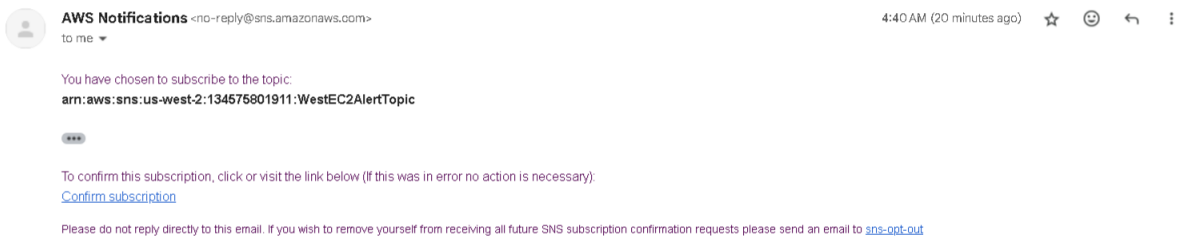
Alarms (3/4)

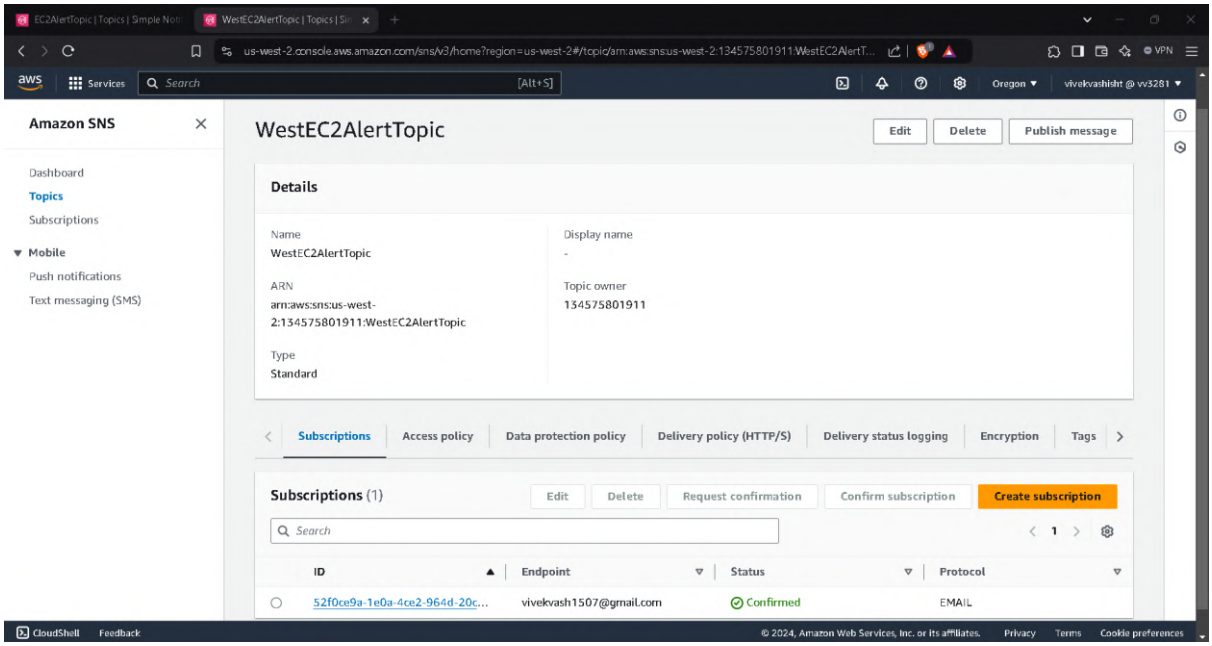
Name	State	Last state update (UTC)	Conditions	Actions
HighCPUUtilizationAlarm	OK	2024-10-31 10:41:03	CPUUtilization > 80 for 2 datapoints within 10 minutes	Actions enabled
HighDiskUtilizationAlarm	Insufficient data	2024-10-31 10:40:26	disk_used_percent > 80 for 2 datapoints within 10 minutes	Actions enabled
HighMemoryUtilizationAlarm	Insufficient data	2024-10-31 10:40:26	mem_used_percent > 80 for 2 datapoints within 10 minutes	Actions enabled
MySimpleBillingAlarm	Insufficient data	2023-09-07 05:08:59	EstimatedCharges > 50 for 1 datapoints within 6 hours	Actions enabled


CloudWatch
Alarms (3/3)
Hide Auto Scaling alarms
Clear selection
Create composite alarm
Actions
Create alarm
Name
State
Last state update (UTC)
Conditions
Actions
WestHighCPUUtilizationAlarm
OK
2024-10-31 10:41:26
CPUUtilization > 80 for 2 datapoints within 10 minutes
Actions enabled
WestHighDiskUtilizationAlarm
Insufficient data
2024-10-31 10:40:25
disk_used_percent > 80 for 2 datapoints within 10 minutes
Actions enabled
WestHighMemoryUtilizationAlarm
Insufficient data
2024-10-31 10:40:25
mem_used_percent > 80 for 2 datapoints within 10 minutes
Actions enabled


AWS Notifications <no-reply@sns.amazonaws.com>
to me
4:40 AM (20 minutes ago)
You have chosen to subscribe to the topic:
arn:aws:sns:us-east-1:134575801911:EC2AlertTopic
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-opt-out


EC2AlertTopic
Edit
Delete
Publish message
Details
Name
EC2AlertTopic
Display name
-
ARN
arn:aws:sns:us-east-1:134575801911:EC2AlertTopic
Topic owner
134575801911
Type
Standard
Subscriptions
Access policy
Data protection policy
Delivery policy (HTTP/S)
Delivery status logging
Encryption
Tags
Subscriptions (1)
Edit
Delete
Request confirmation
Confirm subscription
Create subscription
ID
Endpoint
Status
Protocol
45bab4c9-67d8-4001-97e7-52...
vivekvash1507@gmail.com
Confirmed
EMAIL





Parameterization and Validation

a) Overview

In this phase, I enhanced the flexibility and maintainability of the Terraform configurations by introducing variables and adding validation constraints. This adjustment allows easy modifications to configurations, making the infrastructure adaptable while ensuring inputs meet specific criteria. This centralized approach also enables consistent updates across configurations by managing values from a single source.

b) Defining Variables in variables.tf

To facilitate customization, I created a `variables.tf` file to define key variables that could be easily modified to adapt the infrastructure. This file includes primary and secondary AWS

regions, EC2 instance types, S3 bucket names for Terraform state and backups, Application Load Balancer (ALB) names, a Route 53 domain, and an email address for SNS notifications.

Additionally, I introduced validation rules for specific variables to enforce constraints:

- **Instance Type Validation**: Limits `instance_type` to `"t2.micro"`, `"t2.small"`, or `"t3.micro"`, ensuring only compatible options are used.
- **Email Format Validation**: Enforces a valid format for the `notification_email` variable.

The following is the complete structure of `variables.tf`:

```
# AWS Regions
variable "aws_region" {
  description = "Primary AWS region"
  type        = string
  default     = "us-east-1"
}

variable "aws_west_region" {
  description = "Secondary AWS region"
  type        = string
  default     = "us-west-2"
}

# Instance Type with Validation
variable "instance_type" {
  description = "EC2 instance type"
  type        = string
  default     = "t2.micro"
  validation {
    condition     = contains(["t2.micro", "t2.small", "t3.micro"],
var.instance_type)
    error_message = "The instance type must be t2.micro, t2.small, or
t3.micro."
  }
}

# S3 Bucket Names
variable "terraform_state_bucket" {
  description = "S3 bucket name for storing Terraform state"
  type        = string
  default     = "aws-tf-backend-vv-bucket"
}

variable "backup_bucket" {
  description = "S3 bucket name for backups"
  type        = string
  default     = "terra-backup-vv-project-bucket"
}

# ALB Names
variable "primary_alb_name" {
  description = "Name of the primary ALB"
  type        = string
  default     = "primary-alb"
}
```

```
variable "secondary_alb_name" {
  description = "Name of the secondary ALB"
  type        = string
  default     = "secondary-alb"
}

# Route 53 Zone Domain
variable "route53_zone_domain" {
  description = "Domain name for Route 53 hosted zone"
  type        = string
  default     = "myinfra.local"
}

# Notification Email for SNS with Validation
variable "notification_email" {
  description = "Email address for SNS notifications"
  type        = string
  default     = "vivekvash1507@gmail.com"
  validation {
    condition     = length(regexall("^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-
]+\\.[a-zA-Z]{2,}$", var.notification_email)) > 0
    error_message = "The email format is invalid."
  }
}
```

Each variable's `description` attribute provides clarity on its purpose, making it easier for anyone to understand its role in the configuration.

c) Modifying alb.tf to Use Variables

With `variables.tf` in place, I proceeded to modify the `alb.tf` file to utilize these variables. This step involved replacing hardcoded values with variable references for increased flexibility and ease of maintenance.

1. **Primary ALB Configuration (`us-east-1`)**
   The primary Application Load Balancer configuration now references `primary_alb_name` for the ALB name, `aws_region` for the primary AWS region, and integrates the security group and subnets using previously defined resources. This setup enhances clarity and adaptability.

   ```
   # Application Load Balancer (ALB) (us-east-1)
   resource "aws_lb" "primary_alb" {
     name               = var.primary_alb_name
     internal           = false
     load_balancer_type = "application"
     security_groups    = [aws_security_group.app_sg.id]
     subnets            = [aws_subnet.public_subnet_1.id,
   aws_subnet.public_subnet_2.id]
     enable_deletion_protection = false
     idle_timeout               = 400

     tags = {
       Name = "PrimaryALB"
     }
   }
   ```

```
# ALB Target Group for us-east-1
resource "aws_lb_target_group" "primary" {
  name     = "primary-target-group"
  port     = 80
  protocol = "HTTP"
  vpc_id   = aws_vpc.main_vpc.id

  health_check {
    path                = "/"
    interval            = 30
    timeout             = 5
    healthy_threshold   = 2
    unhealthy_threshold = 2
  }

  tags = {
    Name = "PrimaryTargetGroup"
  }
}

# ALB Listener for HTTP traffic
resource "aws_lb_listener" "http" {
  load_balancer_arn = aws_lb.primary_alb.arn
  port              = 80
  protocol          = "HTTP"

  default_action {
    type             = "forward"
    target_group_arn = aws_lb_target_group.primary.arn
  }

  tags = {
    Name = "PrimaryALBListener"
  }
}
```

2. **Secondary ALB Configuration (`us-west-2`)**
   The secondary ALB configuration, located in `us-west-2`, references `secondary_alb_name` for the ALB name and `aws_west_region` for the AWS region, thus centralizing region and name parameters.

```
# Application Load Balancer (ALB) for us-west-2
resource "aws_lb" "secondary_alb" {
  provider           = aws.west
  name               = var.secondary_alb_name
  internal           = false
  load_balancer_type = "application"
  security_groups    = [aws_security_group.west_app_sg.id]
  subnets            = [aws_subnet.west_public_subnet_1.id,
aws_subnet.west_public_subnet_2.id]
  enable_deletion_protection = false
  idle_timeout               = 400

  tags = {
    Name = "SecondaryALB"
  }
}

# ALB Target Group for us-west-2
resource "aws_lb_target_group" "secondary" {
```

```
  provider = aws.west
  name     = "secondary-target-group"
  port     = 80
  protocol = "HTTP"
  vpc_id   = aws_vpc.west_vpc.id

  health_check {
    path                = "/"
    interval            = 30
    timeout             = 5
    healthy_threshold   = 2
    unhealthy_threshold = 2
  }

  tags = {
    Name = "SecondaryTargetGroup"
  }
}

# ALB Listener for HTTP traffic in us-west-2
resource "aws_lb_listener" "secondary_http" {
  provider          = aws.west
  load_balancer_arn = aws_lb.secondary_alb.arn
  port              = 80
  protocol          = "HTTP"

  default_action {
    type             = "forward"
    target_group_arn = aws_lb_target_group.secondary.arn
  }

  tags = {
    Name = "SecondaryALBListener"
  }
}
```

d) Updated cloudwatch_alarms.tf to Use Variables

In this step, I updated the `cloudwatch_alarms.tf` file to use the variables defined in `variables.tf`, enhancing flexibility and consistency across the Terraform configuration. Below is the updated configuration file.

```
# us-east-1

# CloudWatch Alarm for CPU Utilization
resource "aws_cloudwatch_metric_alarm" "high_cpu_utilization" {
  alarm_name          = "HighCPUUtilizationAlarm"
  comparison_operator = "GreaterThanThreshold"
  evaluation_periods  = 2
  metric_name         = "CPUUtilization"
  namespace           = "AWS/EC2"
  period              = 300
  statistic           = "Average"
  threshold           = 80 # Set threshold to 80% CPU Utilization
  alarm_description   = "This alarm triggers if CPU utilization exceeds
80%."
```

```
  dimensions = {
    AutoScalingGroupName = aws_autoscaling_group.app_asg.name
  }
  actions_enabled = true
  alarm_actions   = [aws_sns_topic.alert_topic.arn]
}

# CloudWatch Alarm for Memory Utilization
resource "aws_cloudwatch_metric_alarm" "high_memory_utilization" {
  alarm_name          = "HighMemoryUtilizationAlarm"
  comparison_operator = "GreaterThanThreshold"
  evaluation_periods  = 2
  metric_name         = "mem_used_percent" # Assuming this is a custom
metric from CloudWatch Agent
  namespace           = "CWAgent"
  period              = 300
  statistic           = "Average"
  threshold           = 80 # Set threshold to 80% Memory Utilization
  alarm_description   = "This alarm triggers if memory utilization exceeds
80%."
  dimensions = {
    AutoScalingGroupName = aws_autoscaling_group.app_asg.name
  }
  actions_enabled = true
  alarm_actions   = [aws_sns_topic.alert_topic.arn]
}

# CloudWatch Alarm for Disk Utilization
resource "aws_cloudwatch_metric_alarm" "high_disk_utilization" {
  alarm_name          = "HighDiskUtilizationAlarm"
  comparison_operator = "GreaterThanThreshold"
  evaluation_periods  = 2
  metric_name         = "disk_used_percent" # Custom metric from CloudWatch
Agent
  namespace           = "CWAgent"
  period              = 300
  statistic           = "Average"
  threshold           = 80 # Set threshold based on your requirements
  alarm_description   = "This alarm triggers if disk utilization exceeds
80%."
  dimensions = {
    AutoScalingGroupName = aws_autoscaling_group.app_asg.name
  }
  actions_enabled = true
  alarm_actions   = [aws_sns_topic.alert_topic.arn]
}

# SNS Topic for Alarm Notifications in us-east-1
resource "aws_sns_topic" "alert_topic" {
  name = "EC2AlertTopic"
}

resource "aws_sns_topic_subscription" "email_subscription" {
  topic_arn = aws_sns_topic.alert_topic.arn
  protocol  = "email"
  endpoint  = var.notification_email
}

# us-west-2

# CloudWatch Alarm for CPU Utilization
```

```
resource "aws_cloudwatch_metric_alarm" "west_high_cpu_utilization" {
  provider                = aws.west
  alarm_name              = "WestHighCPUUtilizationAlarm"
  comparison_operator     = "GreaterThanThreshold"
  evaluation_periods      = 2
  metric_name             = "CPUUtilization"
  namespace               = "AWS/EC2"
  period                  = 300
  statistic               = "Average"
  threshold               = 80 # Set threshold based on your requirements
  alarm_description       = "This alarm triggers if CPU utilization exceeds
80%."
  dimensions = {
    AutoScalingGroupName = aws_autoscaling_group.secondary_asg.name
  }
  actions_enabled = true
  alarm_actions   = [aws_sns_topic.west_alert_topic.arn]
}

# CloudWatch Alarm for Memory Utilization
resource "aws_cloudwatch_metric_alarm" "west_high_memory_utilization" {
  provider                = aws.west
  alarm_name              = "WestHighMemoryUtilizationAlarm"
  comparison_operator     = "GreaterThanThreshold"
  evaluation_periods      = 2
  metric_name             = "mem_used_percent"
  namespace               = "CWAgent"
  period                  = 300
  statistic               = "Average"
  threshold               = 80 # Set threshold based on your requirements
  alarm_description       = "This alarm triggers if memory utilization
exceeds 80%."
  dimensions = {
    AutoScalingGroupName = aws_autoscaling_group.secondary_asg.name
  }
  actions_enabled = true
  alarm_actions   = [aws_sns_topic.west_alert_topic.arn]
}

# CloudWatch Alarm for Disk Utilization
resource "aws_cloudwatch_metric_alarm" "west_high_disk_utilization" {
  provider                = aws.west
  alarm_name              = "WestHighDiskUtilizationAlarm"
  comparison_operator     = "GreaterThanThreshold"
  evaluation_periods      = 2
  metric_name             = "disk_used_percent"
  namespace               = "CWAgent"
  period                  = 300
  statistic               = "Average"
  threshold               = 80 # Set threshold based on your requirements
  alarm_description       = "This alarm triggers if disk utilization exceeds
80%."
  dimensions = {
    AutoScalingGroupName = aws_autoscaling_group.secondary_asg.name
  }
  actions_enabled = true
  alarm_actions   = [aws_sns_topic.west_alert_topic.arn]
}

# SNS Topic for Alarm Notifications in us-west-2
resource "aws_sns_topic" "west_alert_topic" {
```

```
  provider = aws.west
  name     = "WestEC2AlertTopic"
}

resource "aws_sns_topic_subscription" "west_email_subscription" {
  provider  = aws.west
  topic_arn = aws_sns_topic.west_alert_topic.arn
  protocol  = "email"
  endpoint  = var.notification_email
}
```

e) Updated ec2_autoscaling.tf to Use Variables

In this step, I updated the `ec2_autoscaling.tf` file to use the variables defined in `variables.tf`, making the configuration more adaptable and easier to manage. Below is the updated Terraform configuration file.

```
# For us-east-1:

# Security Group
resource "aws_security_group" "app_sg" {
  name        = "app-sg"
  description = "Allow HTTP and SSH traffic"
  vpc_id      = aws_vpc.main_vpc.id

  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["142.59.28.49/32"]  # Restricted SSH to my IP only
  }

  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["18.206.107.24/29"]  # EC2 Instance Connect IP range for
us-east-1
  }

  ingress {
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]  # Allows HTTP traffic from anywhere
  }

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]  # Allows all outbound traffic
  }

  tags = {
    Name = "AppSecurityGroup"
```

```
  }
}

# Launch Template for EC2 Instances for us-east-1
resource "aws_launch_template" "app" {
  name_prefix   = "app-launch-template"
  image_id      = "ami-06b21ccaeff8cd686"  # Amazon Linux AMI
  instance_type = var.instance_type
  iam_instance_profile {
    name = aws_iam_instance_profile.ec2_instance_profile.name
  }

  network_interfaces {
    security_groups = [aws_security_group.app_sg.id]
  }

  user_data = base64encode(<<-EOF
              #!/bin/bash
              sudo dnf update -y
              sudo dnf install -y nginx amazon-cloudwatch-agent
              sudo systemctl start nginx
              sudo systemctl enable nginx

              # Configure the CloudWatch Agent
              sudo tee /opt/aws/amazon-cloudwatch-agent/etc/amazon-
cloudwatch-agent.json << 'EOF2'
              {
                "metrics": {
                  "append_dimensions": {
                    "AutoScalingGroupName": "AppASGInstance"
                  },
                  "metrics_collected": {
                    "mem": {
                      "measurement": [
                        "mem_used_percent"
                      ],
                      "metrics_collection_interval": 60
                    },
                    "disk": {
                      "measurement": [
                        "used_percent"
                      ],
                      "resources": [
                        "/"
                      ],
                      "metrics_collection_interval": 60
                    }
                  }
                }
              }
              EOF2

              # Start the CloudWatch Agent
              sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-
agent-ctl -a start -c file:/opt/aws/amazon-cloudwatch-agent/etc/amazon-
cloudwatch-agent.json -m ec2
            EOF
  )

  lifecycle {
    create_before_destroy = true
```

```
  }

  tag_specifications {
    resource_type = "instance"
    tags = {
      Name = "AppInstance"
    }
  }
}

# Auto Scaling Group (ASG) for us-east-1
resource "aws_autoscaling_group" "app_asg" {
  desired_capacity     = 2
  min_size             = 1
  max_size             = 5
  vpc_zone_identifier  = [aws_subnet.public_subnet_1.id,
aws_subnet.public_subnet_2.id]
  target_group_arns    = [aws_lb_target_group.primary.arn]

  instance_refresh {
    strategy = "Rolling"
    preferences {
      min_healthy_percentage = 0
      instance_warmup        = 120
    }
  }

  health_check_grace_period = 300
  health_check_type         = "ELB"

  launch_template {
    id      = aws_launch_template.app.id
    version = "$Latest"
  }

  tag {
    key                 = "Name"
    value               = "AppASGInstance"
    propagate_at_launch = true
  }

  lifecycle {
    create_before_destroy = true
  }
}

# Auto Scaling Policies
resource "aws_autoscaling_policy" "scale_up" {
  name                   = "scale-up"
  scaling_adjustment     = 1
  adjustment_type        = "ChangeInCapacity"
  autoscaling_group_name = aws_autoscaling_group.app_asg.name
}

resource "aws_autoscaling_policy" "scale_down" {
  name                   = "scale-down"
  scaling_adjustment     = -1
  adjustment_type        = "ChangeInCapacity"
  autoscaling_group_name = aws_autoscaling_group.app_asg.name
}
```

```
# For us-west-2:

# Security Group
resource "aws_security_group" "west_app_sg" {
  provider    = aws.west
  name        = "west-app-sg"
  description = "Allow HTTP and SSH traffic in us-west-2"
  vpc_id      = aws_vpc.west_vpc.id

  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["142.59.28.49/32"]  # Restrict SSH to your IP
  }

  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["18.237.140.160/29"]  # EC2 Instance Connect IP range
for us-west-2
  }

  ingress {
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }

  tags = {
    Name = "WestAppSecurityGroup"
  }
}

# Launch Template for EC2 Instances in us-west-2
resource "aws_launch_template" "secondary_app" {
  provider       = aws.west
  name_prefix    = "secondary-app-launch-template"
  image_id       = "ami-07c5ecd8498c59db5"
  instance_type  = var.instance_type

  iam_instance_profile {
    name = aws_iam_instance_profile.ec2_instance_profile.name
  }

  network_interfaces {
    security_groups = [aws_security_group.west_app_sg.id]
  }

  user_data = base64encode(<<-EOF
              #!/bin/bash
              sudo dnf update -y
```

```
              sudo dnf install -y nginx amazon-cloudwatch-agent
              sudo systemctl start nginx
              sudo systemctl enable nginx

              # Configure the CloudWatch Agent
              sudo tee /opt/aws/amazon-cloudwatch-agent/etc/amazon-
cloudwatch-agent.json << 'EOF2'
              {
                "metrics": {
                  "append_dimensions": {
                    "AutoScalingGroupName": "SecondaryAppASGInstance"
                  },
                  "metrics_collected": {
                    "mem": {
                      "measurement": [
                        "mem_used_percent"
                      ],
                      "metrics_collection_interval": 60
                    },
                    "disk": {
                      "measurement": [
                        "used_percent"
                      ],
                      "resources": [
                        "/"
                      ],
                      "metrics_collection_interval": 60
                    }
                  }
                }
              }
              EOF2

              # Start the CloudWatch Agent
              sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-
agent-ctl -a start -c file:/opt/aws/amazon-cloudwatch-agent/etc/amazon-
cloudwatch-agent.json -m ec2
            EOF
  )

  lifecycle {
    create_before_destroy = true
  }

  tag_specifications {
    resource_type = "instance"
    tags = {
      Name = "SecondaryAppInstance"
    }
  }
}

# Auto Scaling Group (ASG) for us-west-2
resource "aws_autoscaling_group" "secondary_asg" {
  provider            = aws.west
  desired_capacity    = 2
  min_size            = 1
  max_size            = 5
  vpc_zone_identifier = [aws_subnet.west_public_subnet_1.id,
aws_subnet.west_public_subnet_2.id]
  target_group_arns   = [aws_lb_target_group.secondary.arn]
```

```
  instance_refresh {
    strategy = "Rolling"
    preferences {
      min_healthy_percentage = 0
      instance_warmup        = 120
    }
  }

  health_check_grace_period = 300
  health_check_type         = "ELB"

  launch_template {
    id      = aws_launch_template.secondary_app.id
    version = "$Latest"
  }

  tag {
    key                 = "Name"
    value               = "SecondaryAppASGInstance"
    propagate_at_launch = true
  }

  lifecycle {
    create_before_destroy = true
  }
}

# Auto Scaling Policies
resource "aws_autoscaling_policy" "west_scale_up" {
  provider               = aws.west
  name                   = "west-scale-up"
  scaling_adjustment     = 1
  adjustment_type        = "ChangeInCapacity"
  autoscaling_group_name = aws_autoscaling_group.secondary_asg.name
}

resource "aws_autoscaling_policy" "west_scale_down" {
  provider               = aws.west
  name                   = "west-scale-down"
  scaling_adjustment     = -1
  adjustment_type        = "ChangeInCapacity"
  autoscaling_group_name = aws_autoscaling_group.secondary_asg.name
}
```

f) Updated network.tf to Use Variables

I updated the `network.tf` file to utilize variables defined in `variables.tf`, enhancing flexibility and allowing consistent management across configurations. Below is the updated Terraform configuration file:

```
# VPC for us-east-1
resource "aws_vpc" "main_vpc" {
  cidr_block = "10.0.0.0/16"  # VPC with a /16 subnet range
  tags = {
    Name = "MainVPC"
  }
```

```
}

# Public Subnet in us-east-1a
resource "aws_subnet" "public_subnet_1" {
  vpc_id                  = aws_vpc.main_vpc.id
  cidr_block              = "10.0.1.0/24"  # Public subnet's CIDR block
  availability_zone       = "${var.aws_region}a"
  map_public_ip_on_launch = true
  tags = {
    Name = "PublicSubnet1"
  }
}

# Public Subnet in us-east-1b
resource "aws_subnet" "public_subnet_2" {
  vpc_id                  = aws_vpc.main_vpc.id
  cidr_block              = "10.0.4.0/24"
  availability_zone       = "${var.aws_region}b"
  map_public_ip_on_launch = true
  tags = {
    Name = "PublicSubnet2"
  }
}

# Private Subnet in us-east-1a
resource "aws_subnet" "private_subnet_1" {
  vpc_id                  = aws_vpc.main_vpc.id
  cidr_block              = "10.0.3.0/24"  # Private subnet's CIDR block
  availability_zone       = "${var.aws_region}a"
  map_public_ip_on_launch = false
  tags = {
    Name = "PrivateSubnet1"
  }
}

# VPC for us-west-2
resource "aws_vpc" "west_vpc" {
  provider   = aws.west
  cidr_block = "10.1.0.0/16"
  tags = {
    Name = "WestVPC"
  }
}

# Public Subnet in us-west-2a
resource "aws_subnet" "west_public_subnet_1" {
  provider                = aws.west
  vpc_id                  = aws_vpc.west_vpc.id
  cidr_block              = "10.1.1.0/24"
  availability_zone       = "${var.aws_west_region}a"
  map_public_ip_on_launch = true
  tags = {
    Name = "WestPublicSubnet1"
  }
}

# Public Subnet in us-west-2b
resource "aws_subnet" "west_public_subnet_2" {
  provider                = aws.west
  vpc_id                  = aws_vpc.west_vpc.id
  cidr_block              = "10.1.2.0/24"
```

```
  availability_zone      = "${var.aws_west_region}b"
  map_public_ip_on_launch = true
  tags = {
    Name = "WestPublicSubnet2"
  }
}

# Private Subnet in us-west-2a
resource "aws_subnet" "west_private_subnet_1" {
  provider          = aws.west
  vpc_id            = aws_vpc.west_vpc.id
  cidr_block        = "10.1.3.0/24"
  availability_zone = "${var.aws_west_region}a"
  tags = {
    Name = "WestPrivateSubnet1"
  }
}

# Internet Gateway (IGW) for us-east-1
resource "aws_internet_gateway" "igw" {
  vpc_id = aws_vpc.main_vpc.id
  tags = {
    Name = "MainIGW"
  }
}

# Public Route Table and Route for us-east-1
resource "aws_route_table" "public_rt" {
  vpc_id = aws_vpc.main_vpc.id
  tags = {
    Name = "PublicRouteTable"
  }
}

resource "aws_route" "internet_access" {
  route_table_id         = aws_route_table.public_rt.id
  destination_cidr_block = "0.0.0.0/0"
  gateway_id             = aws_internet_gateway.igw.id
}

resource "aws_route_table_association" "public_subnet_assoc_1" {
  subnet_id      = aws_subnet.public_subnet_1.id
  route_table_id = aws_route_table.public_rt.id
}

resource "aws_route_table_association" "public_subnet_assoc_2" {
  subnet_id      = aws_subnet.public_subnet_2.id
  route_table_id = aws_route_table.public_rt.id
}

# Internet Gateway (IGW) for us-west-2
resource "aws_internet_gateway" "west_igw" {
  provider = aws.west
  vpc_id   = aws_vpc.west_vpc.id
  tags = {
    Name = "WestIGW"
  }
}

# Public Route Table and Route for us-west-2
resource "aws_route_table" "west_public_rt" {
```

```
  provider = aws.west
  vpc_id   = aws_vpc.west_vpc.id
  tags = {
    Name = "WestPublicRouteTable"
  }
}

resource "aws_route" "west_internet_access" {
  provider               = aws.west
  route_table_id         = aws_route_table.west_public_rt.id
  destination_cidr_block = "0.0.0.0/0"
  gateway_id             = aws_internet_gateway.west_igw.id
}

resource "aws_route_table_association" "west_public_subnet_assoc_1" {
  provider       = aws.west
  subnet_id      = aws_subnet.west_public_subnet_1.id
  route_table_id = aws_route_table.west_public_rt.id
}

resource "aws_route_table_association" "west_public_subnet_assoc_2" {
  provider       = aws.west
  subnet_id      = aws_subnet.west_public_subnet_2.id
  route_table_id = aws_route_table.west_public_rt.id
}
```

g) Updated route53.tf to Use Variables

I updated the `route53.tf` file to utilize variables from `variables.tf`, making the configuration adaptable for different environments. Below is the updated Terraform configuration file:

```
# Route 53 Hosted Zone
resource "aws_route53_zone" "main" {
  name = var.route53_zone_domain
}

# Primary Route 53 Record (us-east-1)
resource "aws_route53_record" "primary" {
  zone_id = aws_route53_zone.main.zone_id
  name    = "app.${var.route53_zone_domain}"
  type    = "A"

  alias {
    name                   = aws_lb.primary_alb.dns_name
    zone_id                = aws_lb.primary_alb.zone_id
    evaluate_target_health = true
  }

  set_identifier           = "Primary"
  failover_routing_policy {
    type = "PRIMARY"
  }
}
```

```
# Secondary Route 53 Record (us-west-2)
resource "aws_route53_record" "secondary" {
  zone_id = aws_route53_zone.main.zone_id
  name    = "app.${var.route53_zone_domain}"
  type    = "A"

  alias {
    name                   = aws_lb.secondary_alb.dns_name
    zone_id                = aws_lb.secondary_alb.zone_id
    evaluate_target_health = true
  }

  set_identifier           = "Secondary"
  failover_routing_policy {
    type = "SECONDARY"
  }
}

# Route 53 Health Checks

# Primary Health Check (us-east-1)
resource "aws_route53_health_check" "primary_health_check" {
  fqdn              = aws_lb.primary_alb.dns_name  # Use ALB DNS Name
  type              = "HTTP"
  port              = 80
  request_interval  = 30
  failure_threshold = 3
}

# Secondary Health Check (us-west-2)
resource "aws_route53_health_check" "secondary_health_check" {
  fqdn              = aws_lb.secondary_alb.dns_name  # Use ALB DNS Name
  type              = "HTTP"
  port              = 80
  request_interval  = 30
  failure_threshold = 3
}
```

h) Updated s3_bucket.tf to Use Variables

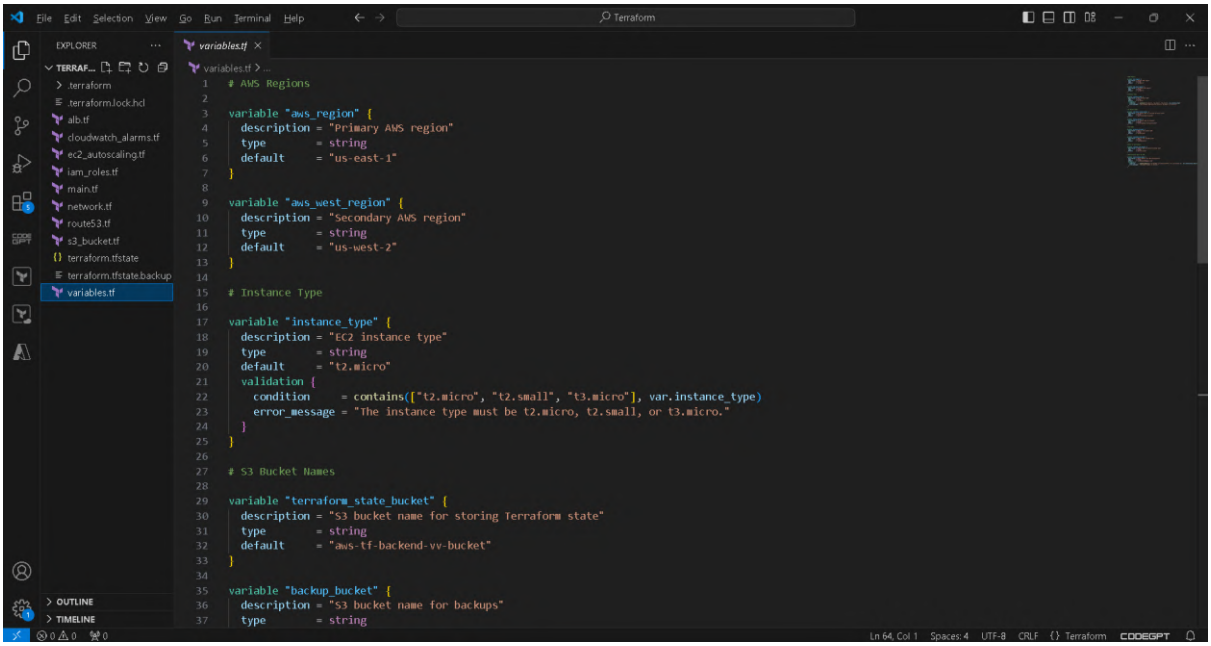
I updated the `s3_bucket.tf` file to reference variables defined in `variables.tf`, allowing for adaptable S3 bucket names and configurations. Below is the updated configuration:

```
# S3 bucket for storing Terraform state
resource "aws_s3_bucket" "terraform_state" {
  bucket = var.terraform_state_bucket  # Use variable for Terraform state
bucket name

  tags = {
    Name        = "TerraformStateBucket"
    Environment = "Dev"
  }
}

# Public access block for the S3 bucket
resource "aws_s3_bucket_public_access_block" "terraform_state_block" {
```

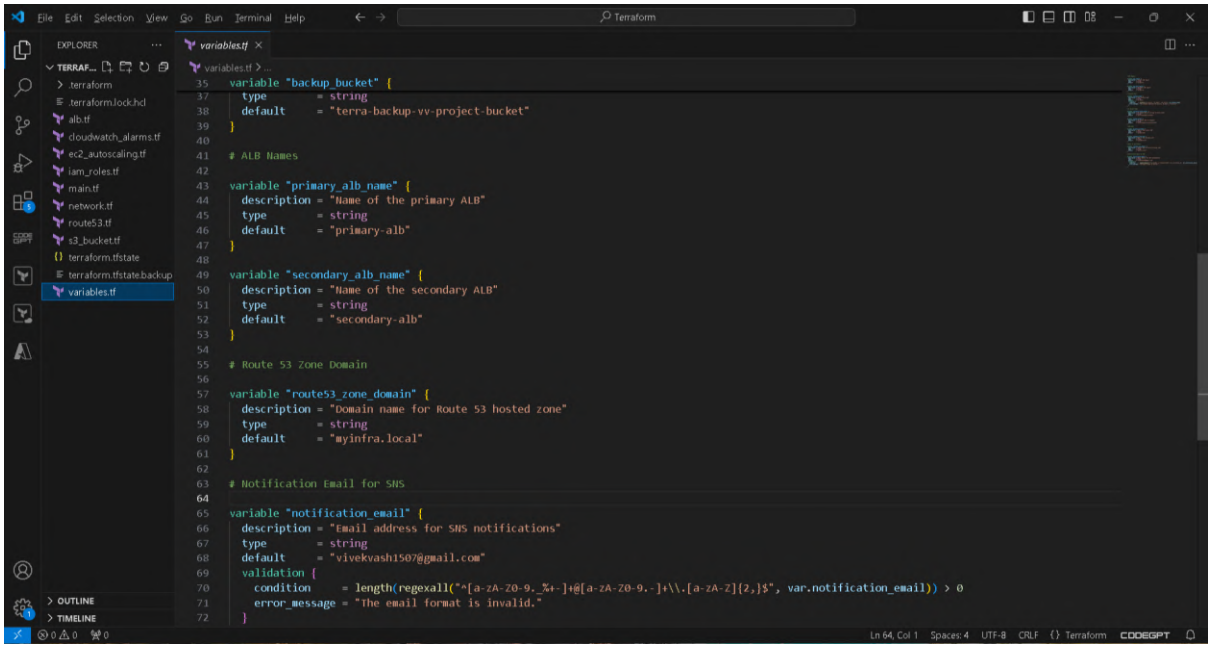
```
  bucket = aws_s3_bucket.terraform_state.id

  block_public_acls       = true
  block_public_policy      = true
  restrict_public_buckets  = true
  ignore_public_acls       = true
}

# S3 bucket for backups
resource "aws_s3_bucket" "backup_bucket" {
  bucket = var.backup_bucket  # Use variable for backup bucket name

  tags = {
    Name        = "BackupBucket"
    Environment = "Dev"
  }
}

# Public access block for the backup S3 bucket
resource "aws_s3_bucket_public_access_block" "backup_bucket_block" {
  bucket = aws_s3_bucket.backup_bucket.id

  block_public_acls       = true
  block_public_policy      = true
  restrict_public_buckets  = true
  ignore_public_acls       = true
}
```

I executed terraform init to initialize the working directory and ensure that all required Terraform modules and provider plugins were properly set up. Following that, I ran terraform plan to generate and review the execution plan, which outlined any anticipated infrastructure changes based on the updated configurations across all Terraform configuration files. The plan output confirmed that no modifications were needed, indicating that the configurations using variables matched the current state. Finally, I executed terraform apply, which applied the plan without making any changes to the infrastructure, further verifying that all configurations were in sync with the existing resources. This process confirmed that the updates were successful, marking the completion of this phase.

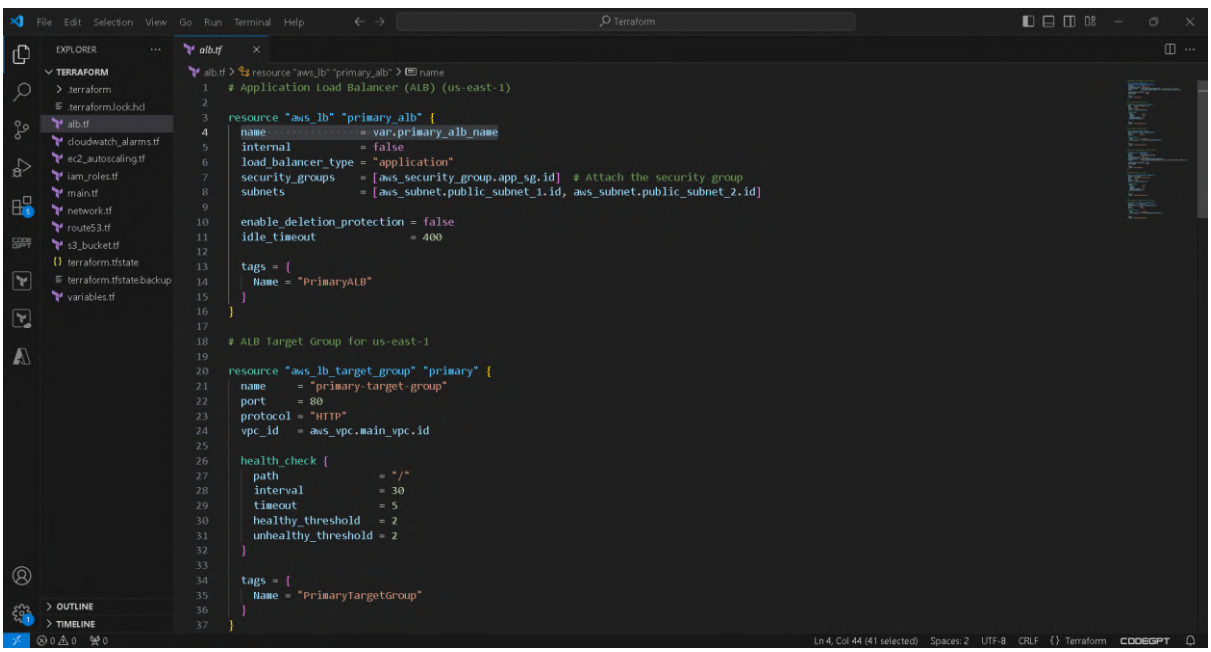
Screenshots



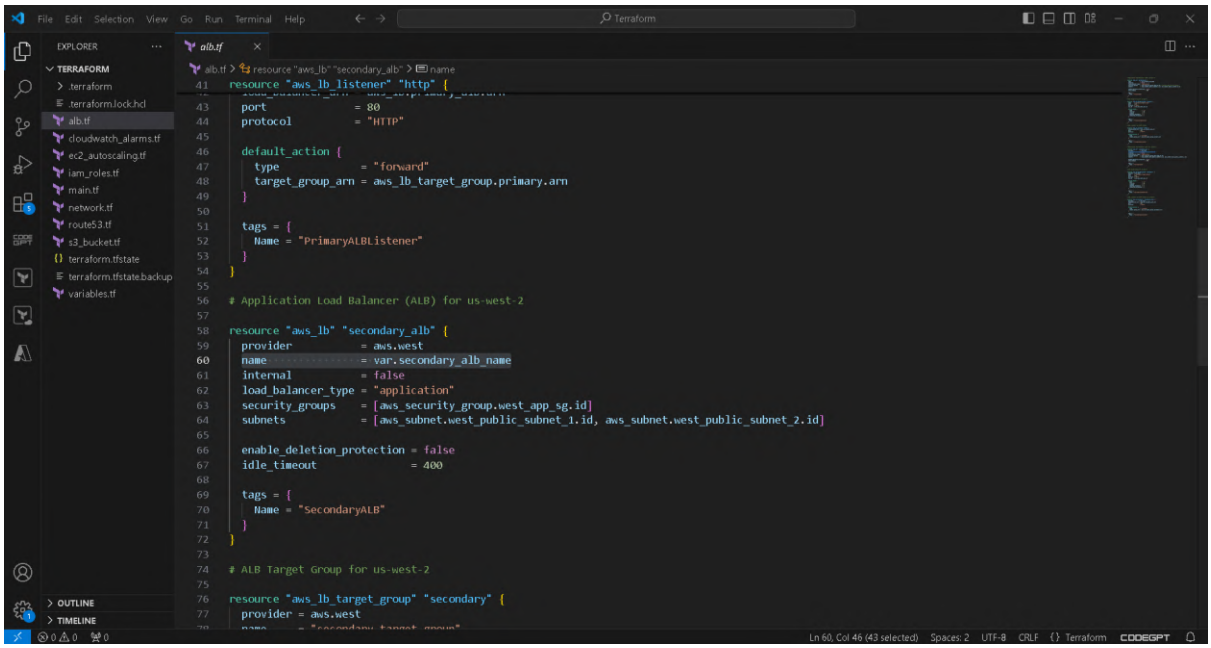
```
# AWS Regions

variable "aws_region" {
  description = "Primary AWS region"
  type        = string
  default     = "us-east-1"
}

variable "aws_west_region" {
  description = "Secondary AWS region"
  type        = string
  default     = "us-west-2"
}

# Instance Type

variable "instance_type" {
  description = "EC2 instance type"
  type        = string
  default     = "t2.micro"
  validation {
    condition     = contains(["t2.micro", "t2.small", "t3.micro"], var.instance_type)
    error_message = "The instance type must be t2.micro, t2.small, or t3.micro."
  }
}

# S3 Bucket Names

variable "terraform_state_bucket" {
  description = "S3 bucket name for storing Terraform state"
  type        = string
  default     = "aws-tf-backend-vv-bucket"
}

variable "backup_bucket" {
  description = "S3 bucket name for backups"
  type        = string
```



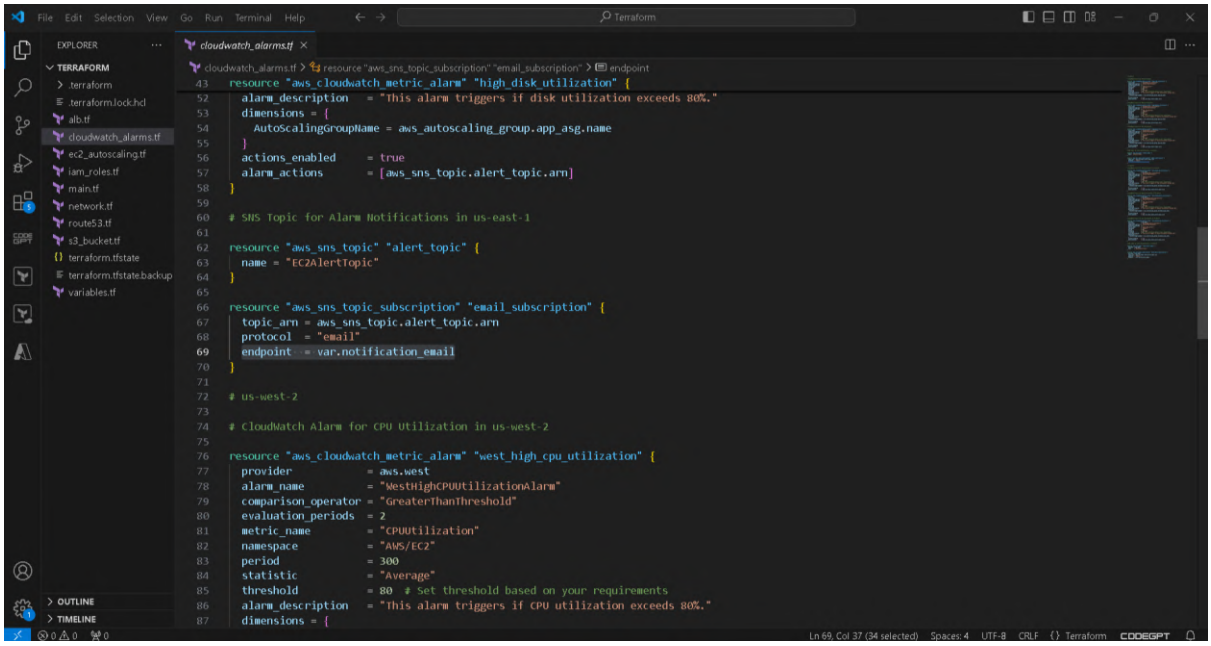
```
variable "backup_bucket" {
  type        = string
  default     = "terra-backup-vv-project-bucket"
}

# ALB Names

variable "primary_alb_name" {
  description = "Name of the primary ALB"
  type        = string
  default     = "primary-alb"
}

variable "secondary_alb_name" {
  description = "Name of the secondary ALB"
  type        = string
  default     = "secondary-alb"
}

# Route 53 Zone Domain

variable "route53_zone_domain" {
  description = "Domain name for Route 53 hosted zone"
  type        = string
  default     = "myinfra.local"
}

# Notification Email for SNS

variable "notification_email" {
  description = "Email address for SNS notifications"
  type        = string
  default     = "vivekvash1507@gmail.com"
  validation {
    condition     = length(regexall("^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}$", var.notification_email)) > 0
    error_message = "The email format is invalid."
  }
```



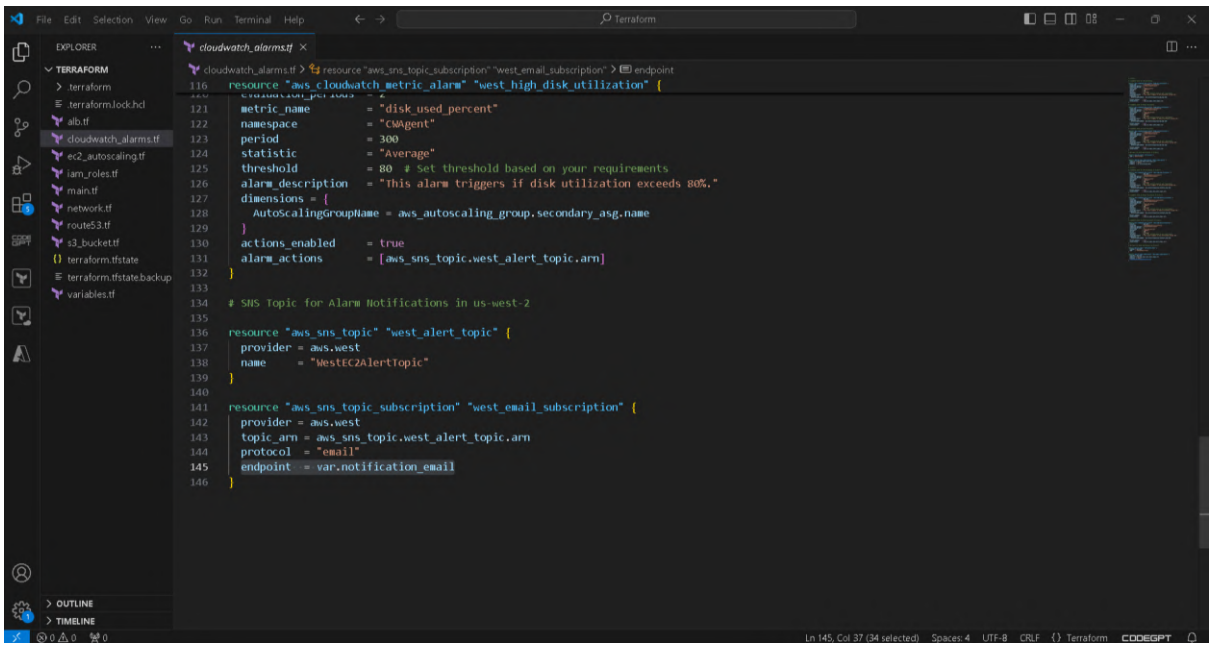
```
# Application Load Balancer (ALB) (us-east-1)

resource "aws_lb" "primary_alb" {
  name               = var.primary_alb_name
  internal           = false
  load_balancer_type = "application"
  security_groups    = [aws_security_group.app_sg.id]  # Attach the security group
  subnets            = [aws_subnet.public_subnet_1.id, aws_subnet.public_subnet_2.id]

  enable_deletion_protection = false
  idle_timeout               = 400

  tags = {
    Name = "PrimaryALB"
  }
}

# ALB Target Group for us-east-1

resource "aws_lb_target_group" "primary" {
  name     = "primary-target-group"
  port     = 80
  protocol = "HTTP"
  vpc_id   = aws_vpc.main_vpc.id

  health_check {
    path                = "/"
    interval            = 30
    timeout             = 5
    healthy_threshold   = 2
    unhealthy_threshold = 2
  }

  tags = {
    Name = "PrimaryTargetGroup"
  }
}
```



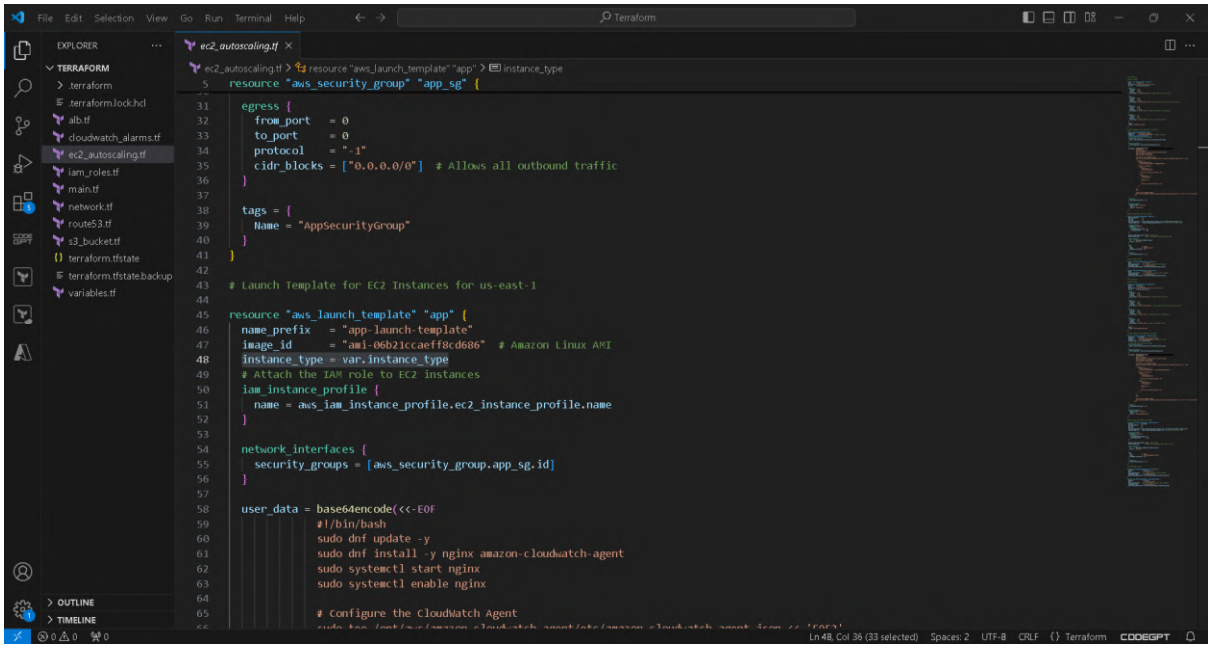
```
alb.tf > resource "aws_lb" "secondary_alb" > name
41  resource "aws_lb_listener" "http" {
43    port              = 80
44    protocol          = "HTTP"
45
46    default_action {
47      type             = "forward"
48      target_group_arn = aws_lb_target_group.primary.arn
49    }
50
51    tags = {
52      Name = "PrimaryALBListener"
53    }
54  }
55
56  # Application Load Balancer (ALB) for us-west-2
57
58  resource "aws_lb" "secondary_alb" {
59    provider           = aws.west
60    name               = var.secondary_alb_name
61    internal           = false
62    load_balancer_type = "application"
63    security_groups    = [aws_security_group.west_app_sg.id]
64    subnets            = [aws_subnet.west_public_subnet_1.id, aws_subnet.west_public_subnet_2.id]
65
66    enable_deletion_protection = false
67    idle_timeout               = 400
68
69    tags = {
70      Name = "SecondaryALB"
71    }
72  }
73
74  # ALB Target Group for us-west-2
75
76  resource "aws_lb_target_group" "secondary" {
77    provider = aws.west
```



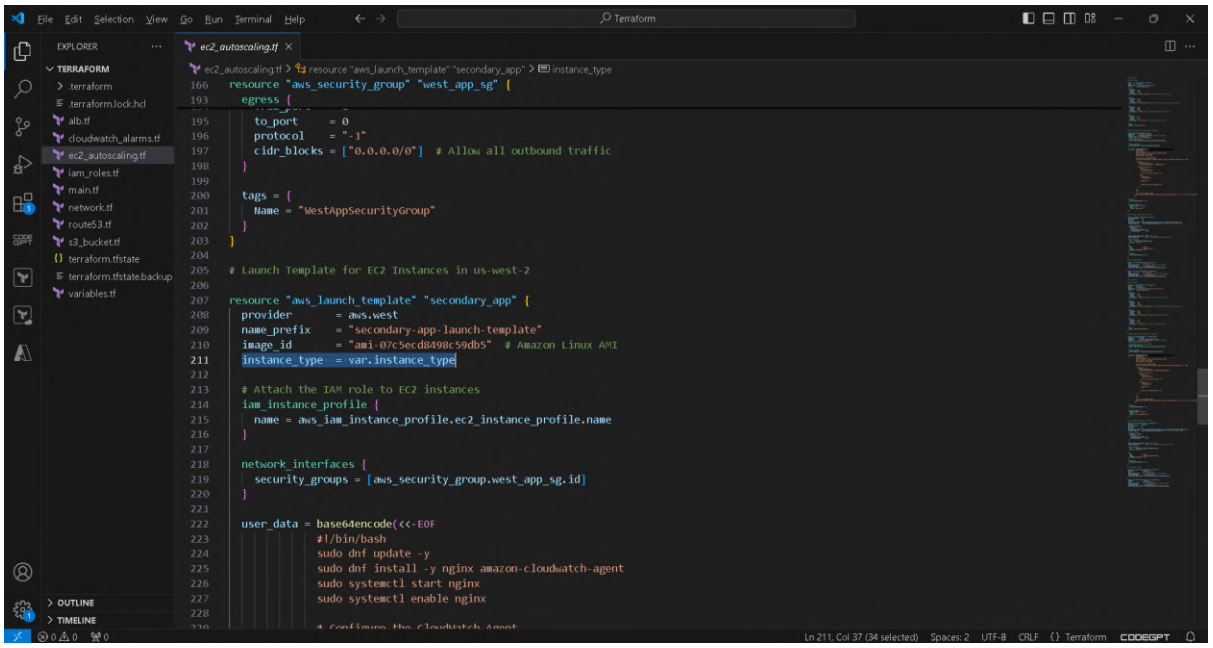
```
cloudwatch_alarms.tf > resource "aws_sns_topic_subscription" "email_subscription" > endpoint
43  resource "aws_cloudwatch_metric_alarm" "high_disk_utilization" {
52    alarm_description   = "This alarm triggers if disk utilization exceeds 80%."
53    dimensions = {
54      AutoScalingGroupName = aws_autoscaling_group.app_asg.name
55    }
56    actions_enabled     = true
57    alarm_actions       = [aws_sns_topic.alert_topic.arn]
58  }
59
60  # SNS Topic for Alarm Notifications in us-east-1
61
62  resource "aws_sns_topic" "alert_topic" {
63    name = "EC2AlertTopic"
64  }
65
66  resource "aws_sns_topic_subscription" "email_subscription" {
67    topic_arn = aws_sns_topic.alert_topic.arn
68    protocol  = "email"
69    endpoint  = var.notification_email
70  }
71
72  # us-west-2
73
74  # CloudWatch Alarm for CPU Utilization in us-west-2
75
76  resource "aws_cloudwatch_metric_alarm" "west_high_cpu_utilization" {
77    provider            = aws.west
78    alarm_name          = "WestHighCPUUtilizationAlarm"
79    comparison_operator = "GreaterThanThreshold"
80    evaluation_periods  = 2
81    metric_name         = "CPUUtilization"
82    namespace           = "AWS/EC2"
83    period              = 300
84    statistic           = "Average"
85    threshold           = 80  # Set threshold based on your requirements
86    alarm_description   = "This alarm triggers if CPU utilization exceeds 80%."
87    dimensions = {
```



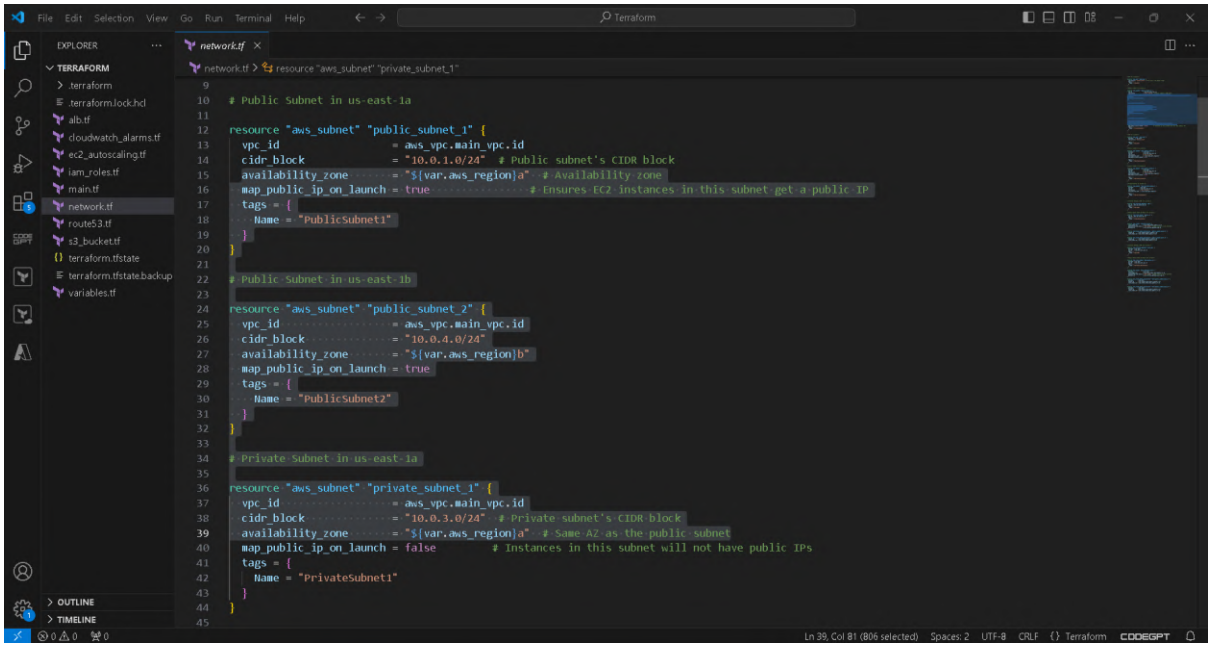
```
cloudwatch_alarms.tf > resource "aws_sns_topic_subscription" "west_email_subscription" > endpoint
116 resource "aws_cloudwatch_metric_alarm" "west_high_disk_utilization" {
121   metric_name         = "disk_used_percent"
122   namespace           = "CWAgent"
123   period              = 300
124   statistic           = "Average"
125   threshold           = 80  # Set threshold based on your requirements
126   alarm_description   = "This alarm triggers if disk utilization exceeds 80%."
127   dimensions = {
128     AutoScalingGroupName = aws_autoscaling_group.secondary_asg.name
129   }
130   actions_enabled     = true
131   alarm_actions       = [aws_sns_topic.west_alert_topic.arn]
132 }
133
134 # SNS Topic for Alarm Notifications in us-west-2
135
136 resource "aws_sns_topic" "west_alert_topic" {
137   provider = aws.west
138   name     = "WestEC2AlertTopic"
139 }
140
141 resource "aws_sns_topic_subscription" "west_email_subscription" {
142   provider = aws.west
143   topic_arn = aws_sns_topic.west_alert_topic.arn
144   protocol  = "email"
145   endpoint  = var.notification_email
146 }
```



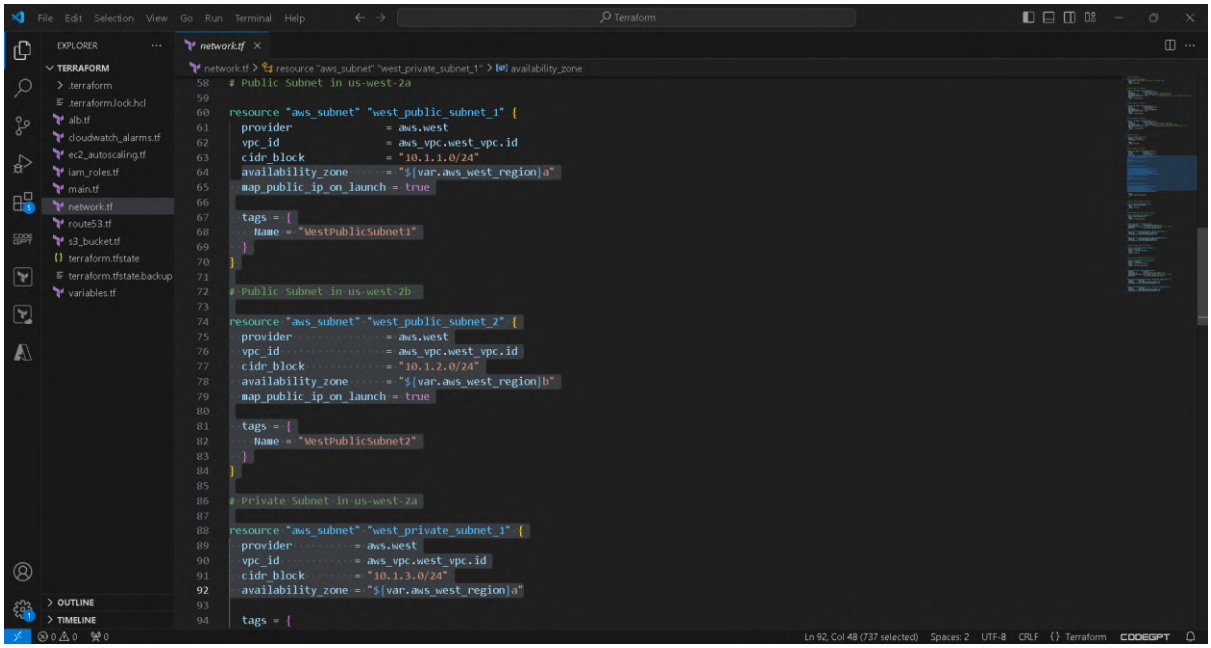
```
resource "aws_security_group" "app_sg" {
  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]  # Allows all outbound traffic
  }

  tags = {
    Name = "AppSecurityGroup"
  }
}

# Launch Template for EC2 Instances for us-east-1

resource "aws_launch_template" "app" {
  name_prefix   = "app-launch-template"
  image_id      = "ami-06b21ccaeff8cd686"  # Amazon Linux AMI
  instance_type = var.instance_type
  # Attach the IAM role to EC2 instances
  iam_instance_profile {
    name = aws_iam_instance_profile.ec2_instance_profile.name
  }

  network_interfaces {
    security_groups = [aws_security_group.app_sg.id]
  }

  user_data = base64encode(<<-EOF
              #!/bin/bash
              sudo dnf update -y
              sudo dnf install -y nginx amazon-cloudwatch-agent
              sudo systemctl start nginx
              sudo systemctl enable nginx

              # Configure the CloudWatch Agent
```



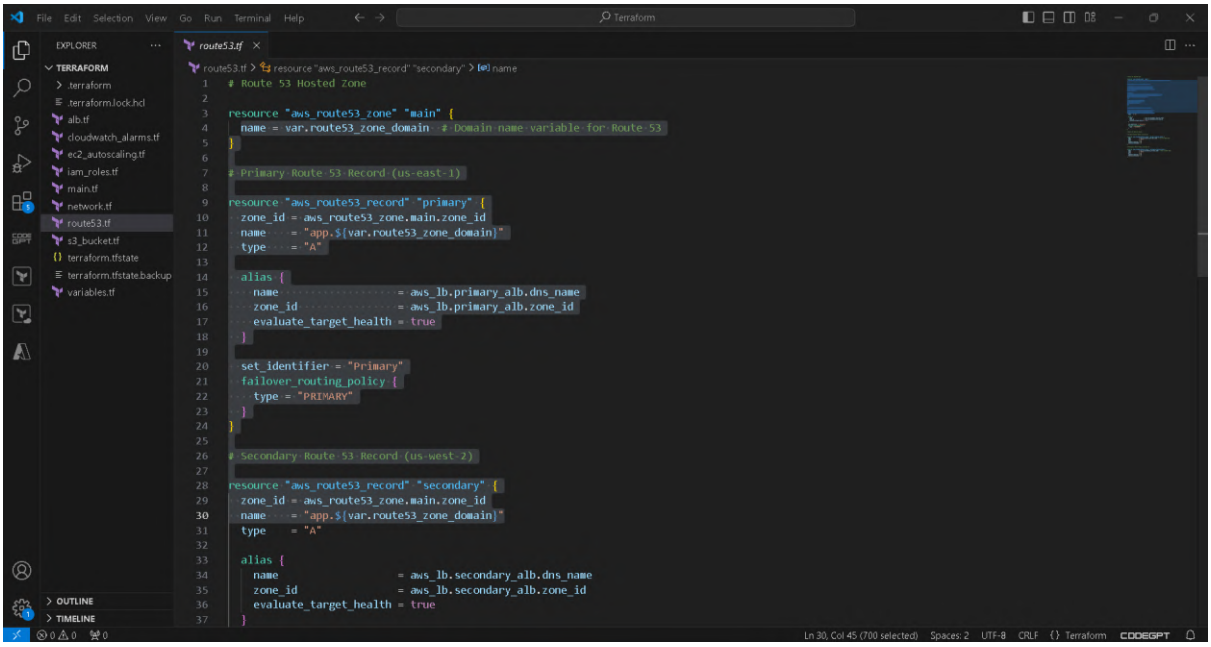
```
resource "aws_security_group" "west_app_sg" {
  egress {
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]  # Allow all outbound traffic
  }

  tags = {
    Name = "WestAppSecurityGroup"
  }
}

# Launch Template for EC2 Instances in us-west-2

resource "aws_launch_template" "secondary_app" {
  provider      = aws.west
  name_prefix   = "secondary-app-launch-template"
  image_id      = "ami-07c5ecd8498c59db5"  # Amazon Linux AMI
  instance_type = var.instance_type

  # Attach the IAM role to EC2 instances
  iam_instance_profile {
    name = aws_iam_instance_profile.ec2_instance_profile.name
  }

  network_interfaces {
    security_groups = [aws_security_group.west_app_sg.id]
  }

  user_data = base64encode(<<-EOF
              #!/bin/bash
              sudo dnf update -y
              sudo dnf install -y nginx amazon-cloudwatch-agent
              sudo systemctl start nginx
              sudo systemctl enable nginx
```



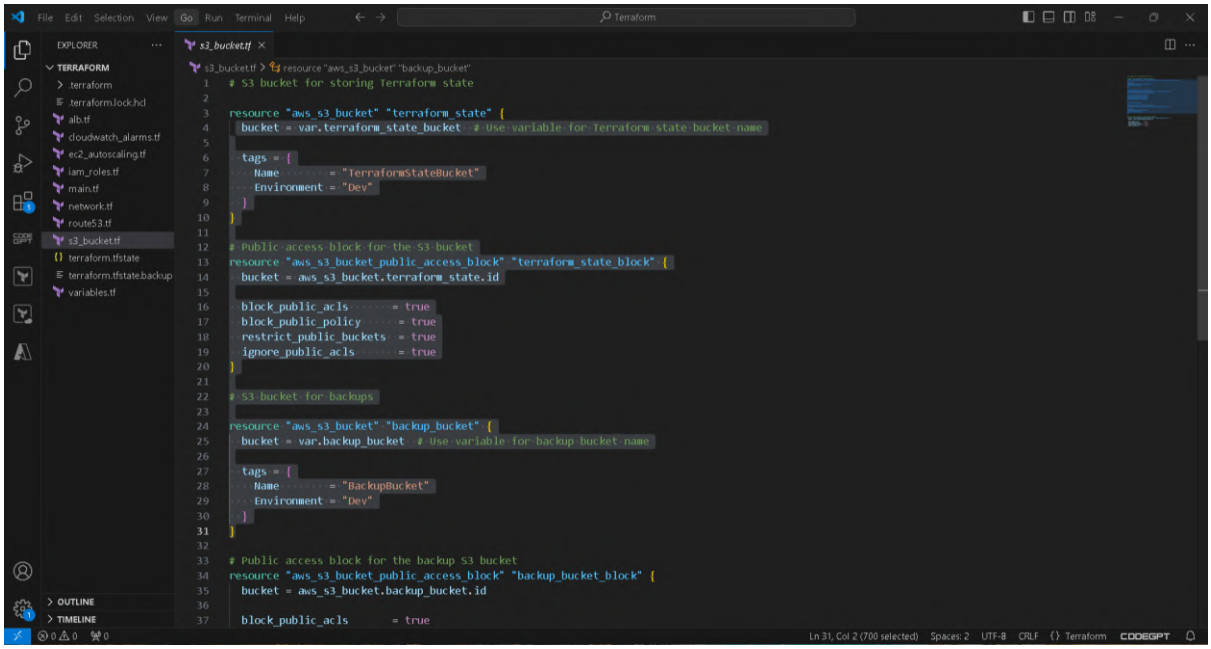
```
# Public Subnet in us-east-1a

resource "aws_subnet" "public_subnet_1" {
  vpc_id                  = aws_vpc.main_vpc.id
  cidr_block              = "10.0.1.0/24"  # Public subnet's CIDR block
  availability_zone       = "${var.aws_region}a"  # Availability zone
  map_public_ip_on_launch = true                  # Ensures EC2 instances in this subnet get a public IP
  tags = {
    Name = "PublicSubnet1"
  }
}

# Public Subnet in us-east-1b

resource "aws_subnet" "public_subnet_2" {
  vpc_id                  = aws_vpc.main_vpc.id
  cidr_block              = "10.0.4.0/24"
  availability_zone       = "${var.aws_region}b"
  map_public_ip_on_launch = true
  tags = {
    Name = "PublicSubnet2"
  }
}

# Private Subnet in us-east-1a

resource "aws_subnet" "private_subnet_1" {
  vpc_id                  = aws_vpc.main_vpc.id
  cidr_block              = "10.0.3.0/24"  # Private subnet's CIDR block
  availability_zone       = "${var.aws_region}a"  # Same AZ as the public subnet
  map_public_ip_on_launch = false         # Instances in this subnet will not have public IPs
  tags = {
    Name = "PrivateSubnet1"
  }
}
```



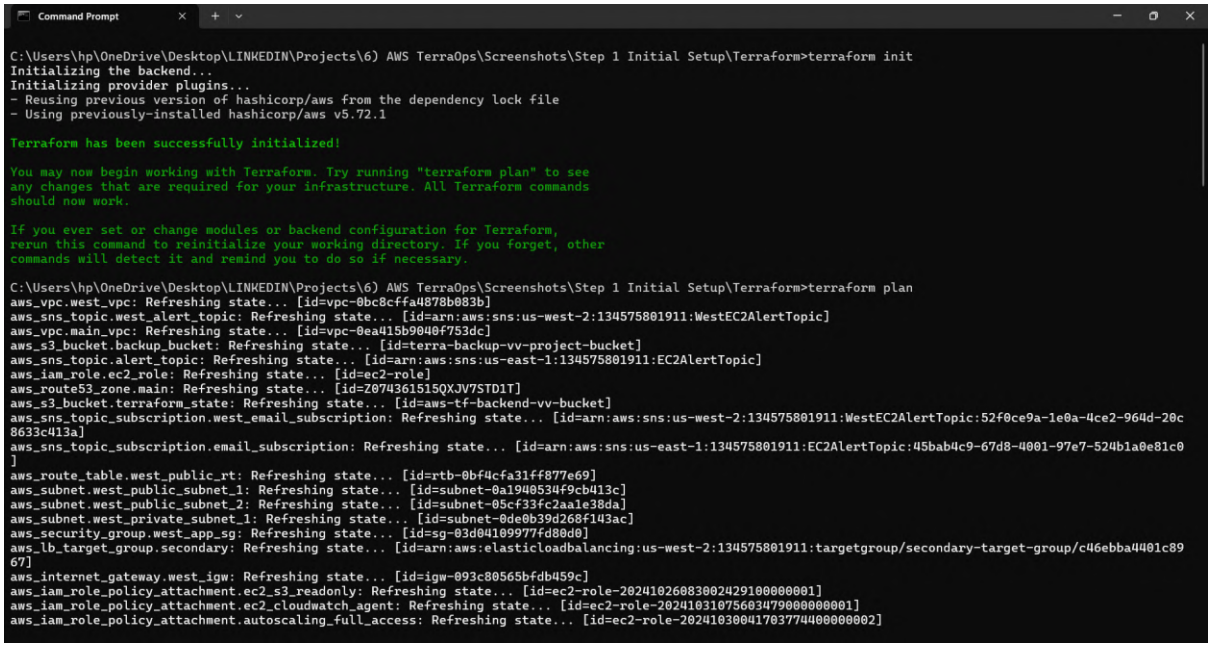
```
# Public Subnet in us-west-2a

resource "aws_subnet" "west_public_subnet_1" {
  provider                = aws.west
  vpc_id                  = aws_vpc.west_vpc.id
  cidr_block              = "10.1.1.0/24"
  availability_zone       = "${var.aws_west_region}a"
  map_public_ip_on_launch = true

  tags = {
    Name = "WestPublicSubnet1"
  }
}

# Public Subnet in us-west-2b

resource "aws_subnet" "west_public_subnet_2" {
  provider                = aws.west
  vpc_id                  = aws_vpc.west_vpc.id
  cidr_block              = "10.1.2.0/24"
  availability_zone       = "${var.aws_west_region}b"
  map_public_ip_on_launch = true

  tags = {
    Name = "WestPublicSubnet2"
  }
}

# Private Subnet in us-west-2a

resource "aws_subnet" "west_private_subnet_1" {
  provider          = aws.west
  vpc_id            = aws_vpc.west_vpc.id
  cidr_block        = "10.1.3.0/24"
  availability_zone = "${var.aws_west_region}a"

  tags = {
```



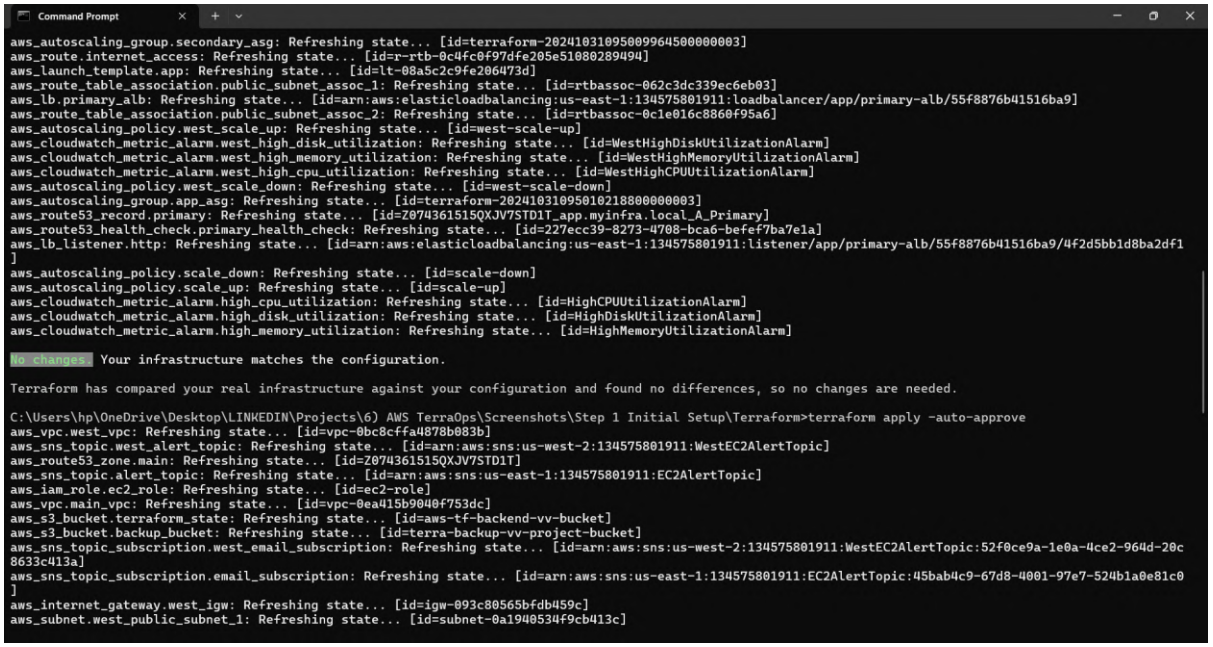
```
# Route 53 Hosted Zone

resource "aws_route53_zone" "main" {
  name = var.route53_zone_domain  # Domain name variable for Route 53
}

# Primary Route 53 Record (us-east-1)

resource "aws_route53_record" "primary" {
  zone_id = aws_route53_zone.main.zone_id
  name    = "app.${var.route53_zone_domain}"
  type    = "A"

  alias {
    name                   = aws_lb.primary_alb.dns_name
    zone_id                = aws_lb.primary_alb.zone_id
    evaluate_target_health = true
  }

  set_identifier = "Primary"
  failover_routing_policy {
    type = "PRIMARY"
  }
}

# Secondary Route 53 Record (us-west-2)

resource "aws_route53_record" "secondary" {
  zone_id = aws_route53_zone.main.zone_id
  name    = "app.${var.route53_zone_domain}"
  type    = "A"

  alias {
    name                   = aws_lb.secondary_alb.dns_name
    zone_id                = aws_lb.secondary_alb.zone_id
    evaluate_target_health = true
  }
```



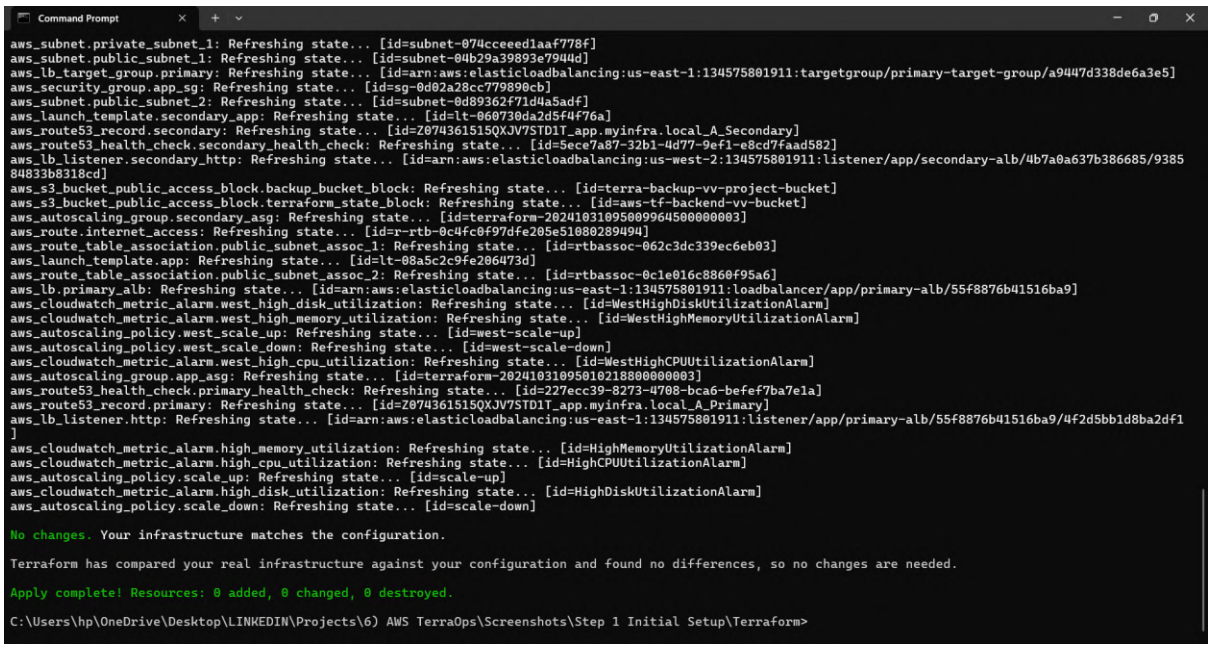
```
# S3 bucket for storing Terraform state

resource "aws_s3_bucket" "terraform_state" {
  bucket = var.terraform_state_bucket  # Use variable for Terraform state bucket name

  tags = {
    Name        = "TerraformStateBucket"
    Environment = "Dev"
  }
}

# Public access block for the S3 bucket
resource "aws_s3_bucket_public_access_block" "terraform_state_block" {
  bucket = aws_s3_bucket.terraform_state.id

  block_public_acls       = true
  block_public_policy     = true
  restrict_public_buckets = true
  ignore_public_acls      = true
}

# S3 bucket for backups

resource "aws_s3_bucket" "backup_bucket" {
  bucket = var.backup_bucket  # Use variable for backup bucket name

  tags = {
    Name        = "BackupBucket"
    Environment = "Dev"
  }
}

# Public access block for the backup S3 bucket
resource "aws_s3_bucket_public_access_block" "backup_bucket_block" {
  bucket = aws_s3_bucket.backup_bucket.id

  block_public_acls       = true
```



```
Command Prompt

C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.72.1

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.

C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>terraform plan
aws_vpc.west_vpc: Refreshing state... [id=vpc-0bc8cffa4878b083b]
aws_sns_topic.west_alert_topic: Refreshing state... [id=arn:aws:sns:us-west-2:134575801911:WestEC2AlertTopic]
aws_vpc.main_vpc: Refreshing state... [id=vpc-0ea415b9040f753dc]
aws_s3_bucket.backup_bucket: Refreshing state... [id=terra-backup-vv-project-bucket]
aws_sns_topic.alert_topic: Refreshing state... [id=arn:aws:sns:us-east-1:134575801911:EC2AlertTopic]
aws_iam_role.ec2_role: Refreshing state... [id=ec2-role]
aws_route53_zone.main: Refreshing state... [id=Z074361515QXJV7STD1T]
aws_s3_bucket.terraform_state: Refreshing state... [id=aws-tf-backend-vv-bucket]
aws_sns_topic_subscription.west_email_subscription: Refreshing state... [id=arn:aws:sns:us-west-2:134575801911:WestEC2AlertTopic:52f0ce9a-1e0a-4ce2-964d-20c
8633c413a]
aws_sns_topic_subscription.email_subscription: Refreshing state... [id=arn:aws:sns:us-east-1:134575801911:EC2AlertTopic:45bab4c9-67d8-4001-97e7-524b1a0e81c0
]
aws_route_table.west_public_rt: Refreshing state... [id=rtb-0bf4cfa31ff877e69]
aws_subnet.west_public_subnet_1: Refreshing state... [id=subnet-0a1940534f9cb413c]
aws_subnet.west_public_subnet_2: Refreshing state... [id=subnet-05cf33fc2aa1e38da]
aws_subnet.west_private_subnet_1: Refreshing state... [id=subnet-0de0b39d268f143ac]
aws_security_group.west_app_sg: Refreshing state... [id=sg-03d04109977fd80d0]
aws_lb_target_group.secondary: Refreshing state... [id=arn:aws:elasticloadbalancing:us-west-2:134575801911:targetgroup/secondary-target-group/c46ebba4401c89
67]
aws_internet_gateway.west_igw: Refreshing state... [id=igw-093c80565bfdb459c]
aws_iam_role_policy_attachment.ec2_s3_readonly: Refreshing state... [id=ec2-role-20241026083002429100000001]
aws_iam_role_policy_attachment.ec2_cloudwatch_agent: Refreshing state... [id=ec2-role-20241031075034790000000001]
aws_iam_role_policy_attachment.autoscaling_full_access: Refreshing state... [id=ec2-role-20241030041703774400000002]
```



```
Command Prompt

aws_autoscaling_group.secondary_asg: Refreshing state... [id=terraform-20241031095009964500000003]
aws_route.internet_access: Refreshing state... [id=r-rtb-0c4fc0f97dfe205e51080289494]
aws_launch_template.app: Refreshing state... [id=lt-08a5c2c9fe206473d]
aws_route_table_association.public_subnet_assoc_1: Refreshing state... [id=rtbassoc-062c3dc339ec6eb03]
aws_lb.primary_alb: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:134575801911:loadbalancer/app/primary-alb/55f8876b41516ba9]
aws_route_table_association.public_subnet_assoc_2: Refreshing state... [id=rtbassoc-0c1e016c8860f95a6]
aws_autoscaling_policy.west_scale_up: Refreshing state... [id=west-scale-up]
aws_cloudwatch_metric_alarm.west_high_disk_utilization: Refreshing state... [id=WestHighDiskUtilizationAlarm]
aws_cloudwatch_metric_alarm.west_high_memory_utilization: Refreshing state... [id=WestHighMemoryUtilizationAlarm]
aws_cloudwatch_metric_alarm.west_high_cpu_utilization: Refreshing state... [id=WestHighCPUUtilizationAlarm]
aws_autoscaling_policy.west_scale_down: Refreshing state... [id=west-scale-down]
aws_autoscaling_group.app_asg: Refreshing state... [id=terraform-20241031095010218800000003]
aws_route53_record.primary: Refreshing state... [id=Z074361515QXJV7STD1T_app.myinfra.local_A_Primary]
aws_route53_health_check.primary_health_check: Refreshing state... [id=227ecc39-8273-4708-bca6-befef7ba7e1a]
aws_lb_listener.http: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:134575801911:listener/app/primary-alb/55f8876b41516ba9/4f2d5bb1d8ba2df1
]
aws_autoscaling_policy.scale_down: Refreshing state... [id=scale-down]
aws_autoscaling_policy.scale_up: Refreshing state... [id=scale-up]
aws_cloudwatch_metric_alarm.high_cpu_utilization: Refreshing state... [id=HighCPUUtilizationAlarm]
aws_cloudwatch_metric_alarm.high_disk_utilization: Refreshing state... [id=HighDiskUtilizationAlarm]
aws_cloudwatch_metric_alarm.high_memory_utilization: Refreshing state... [id=HighMemoryUtilizationAlarm]

No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and found no differences, so no changes are needed.

C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>terraform apply -auto-approve
aws_vpc.west_vpc: Refreshing state... [id=vpc-0bc8cffa4878b083b]
aws_sns_topic.west_alert_topic: Refreshing state... [id=arn:aws:sns:us-west-2:134575801911:WestEC2AlertTopic]
aws_route53_zone.main: Refreshing state... [id=Z074361515QXJV7STD1T]
aws_sns_topic.alert_topic: Refreshing state... [id=arn:aws:sns:us-east-1:134575801911:EC2AlertTopic]
aws_iam_role.ec2_role: Refreshing state... [id=ec2-role]
aws_vpc.main_vpc: Refreshing state... [id=vpc-0ea415b9040f753dc]
aws_s3_bucket.terraform_state: Refreshing state... [id=aws-tf-backend-vv-bucket]
aws_s3_bucket.backup_bucket: Refreshing state... [id=terra-backup-vv-project-bucket]
aws_sns_topic_subscription.west_email_subscription: Refreshing state... [id=arn:aws:sns:us-west-2:134575801911:WestEC2AlertTopic:52f0ce9a-1e0a-4ce2-964d-20c
8633c413a]
aws_sns_topic_subscription.email_subscription: Refreshing state... [id=arn:aws:sns:us-east-1:134575801911:EC2AlertTopic:45bab4c9-67d8-4001-97e7-524b1a0e81c0
]
aws_internet_gateway.west_igw: Refreshing state... [id=igw-093c80565bfdb459c]
aws_subnet.west_public_subnet_1: Refreshing state... [id=subnet-0a1940534f9cb413c]
```



```
Command Prompt

aws_subnet.private_subnet_1: Refreshing state... [id=subnet-074cceeed1aaf778f]
aws_subnet.public_subnet_1: Refreshing state... [id=subnet-04b29a39893e7944d]
aws_lb_target_group.primary: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:134575801911:targetgroup/primary-target-group/a9447d338de6a3e5]
aws_security_group.app_sg: Refreshing state... [id=sg-0d02a28cc779890cb]
aws_subnet.public_subnet_2: Refreshing state... [id=subnet-0d89362f71d4a5adf]
aws_launch_template.secondary_app: Refreshing state... [id=lt-060730da2d5f4f76a]
aws_route53_record.secondary: Refreshing state... [id=Z074361515QXJV7STD1T_app.myinfra.local_A_Secondary]
aws_route53_health_check.secondary_health_check: Refreshing state... [id=5ece7a87-32b1-4d77-9ef1-e8cd7faad582]
aws_lb_listener.secondary_http: Refreshing state... [id=arn:aws:elasticloadbalancing:us-west-2:134575801911:listener/app/secondary-alb/4b7a0a637b386685/9385
84833b8318cd]
aws_s3_bucket_public_access_block.backup_bucket_block: Refreshing state... [id=terra-backup-vv-project-bucket]
aws_s3_bucket_public_access_block.terraform_state_block: Refreshing state... [id=aws-tf-backend-vv-bucket]
aws_autoscaling_group.secondary_asg: Refreshing state... [id=terraform-20241031095009964500000003]
aws_route.internet_access: Refreshing state... [id=r-rtb-0c4fc0f97dfe205e51080289494]
aws_route_table_association.public_subnet_assoc_1: Refreshing state... [id=rtbassoc-062c3dc339ec6eb03]
aws_launch_template.app: Refreshing state... [id=lt-08a5c2c9fe206473d]
aws_route_table_association.public_subnet_assoc_2: Refreshing state... [id=rtbassoc-0c1e016c8860f95a6]
aws_lb.primary_alb: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:134575801911:loadbalancer/app/primary-alb/55f8876b41516ba9]
aws_cloudwatch_metric_alarm.west_high_disk_utilization: Refreshing state... [id=WestHighDiskUtilizationAlarm]
aws_cloudwatch_metric_alarm.west_high_memory_utilization: Refreshing state... [id=WestHighMemoryUtilizationAlarm]
aws_autoscaling_policy.west_scale_up: Refreshing state... [id=west-scale-up]
aws_autoscaling_policy.west_scale_down: Refreshing state... [id=west-scale-down]
aws_cloudwatch_metric_alarm.west_high_cpu_utilization: Refreshing state... [id=WestHighCPUUtilizationAlarm]
aws_autoscaling_group.app_asg: Refreshing state... [id=terraform-20241031095010218800000003]
aws_route53_health_check.primary_health_check: Refreshing state... [id=227ecc39-8273-4708-bca6-befef7ba7e1a]
aws_route53_record.primary: Refreshing state... [id=Z074361515QXJV7STD1T_app.myinfra.local_A_Primary]
aws_lb_listener.http: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:134575801911:listener/app/primary-alb/55f8876b41516ba9/4f2d5bb1d8ba2df1
]
aws_cloudwatch_metric_alarm.high_memory_utilization: Refreshing state... [id=HighMemoryUtilizationAlarm]
aws_cloudwatch_metric_alarm.high_cpu_utilization: Refreshing state... [id=HighCPUUtilizationAlarm]
aws_autoscaling_policy.scale_up: Refreshing state... [id=scale-up]
aws_cloudwatch_metric_alarm.high_disk_utilization: Refreshing state... [id=HighDiskUtilizationAlarm]
aws_autoscaling_policy.scale_down: Refreshing state... [id=scale-down]

No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and found no differences, so no changes are needed.

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

C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>
```

Testing

1) Application Load Balancer (ALB) Testing

a) Overview

In this phase, I focused on testing the functionality of the Application Load Balancers (ALBs) deployed in both the primary (us-east-1) and secondary (us-west-2) regions. The primary goal was to confirm that the ALBs distribute incoming traffic evenly across all EC2 instances within each region, ensuring load balancing is functioning as expected.

b) Setting Up Test Pages on EC2 Instances

To identify which instance was serving the traffic during testing, I customized the default NGINX web page on each instance in both regions with unique identifiers.

- **us-east-1 Instances:**
  - **Instance 1:**

```
echo "Welcome to nginx! Served by instance ID: i-
0aad6808354626dab (us-east-1) INSTANCE 1" | sudo tee
/usr/share/nginx/html/index.html
curl http://localhost
curl http://44.201.4.236
```

  - **Instance 2:**

```
echo "Welcome to nginx! Served by instance ID: i-
0003f08c9125643ef (us-east-1) INSTANCE 2" | sudo tee
/usr/share/nginx/html/index.html
curl http://localhost
curl http://18.207.173.25
```

- **us-west-2 Instances:**
  - **Instance 1:**

```
echo "Welcome to nginx! Served by instance ID: i-
0848e26fb476e924b (us-west-2) INSTANCE 1" | sudo tee
/usr/share/nginx/html/index.html
curl http://localhost
curl http://18.236.133.124
```

  - **Instance 2:**

```
echo "Welcome to nginx! Served by instance ID: i-
070b3c508dbbb30d5 (us-west-2) INSTANCE 2" | sudo tee
/usr/share/nginx/html/index.html
curl http://localhost
```

```
curl http://34.213.180.233
```

These commands modified the index page on each instance to reflect its unique instance ID, allowing for clear tracking of traffic distribution during testing.

c) Retrieving ALB DNS Names

After setting up the EC2 instances, I obtained the DNS names of both ALBs from the EC2 Console:

- **Primary ALB (us-east-1):** `primary-alb-1361659550.us-east-1.elb.amazonaws.com`
- **Secondary ALB (us-west-2):** `secondary-alb-1857056899.us-west-2.elb.amazonaws.com`

d) Testing ALB Load Balancing via Command Line

Using the ALB DNS names, I tested traffic distribution by repeatedly sending requests from the command line:

- **Primary ALB Test (us-east-1):**

  ```
  curl http://primary-alb-1361659550.us-east-1.elb.amazonaws.com
  ```

- **Secondary ALB Test (us-west-2):**

  ```
  curl http://secondary-alb-1857056899.us-west-2.elb.amazonaws.com
  ```

By running these commands multiple times, I confirmed that traffic was equally balanced across the two instances in each region, as each instance took turns responding to the requests.

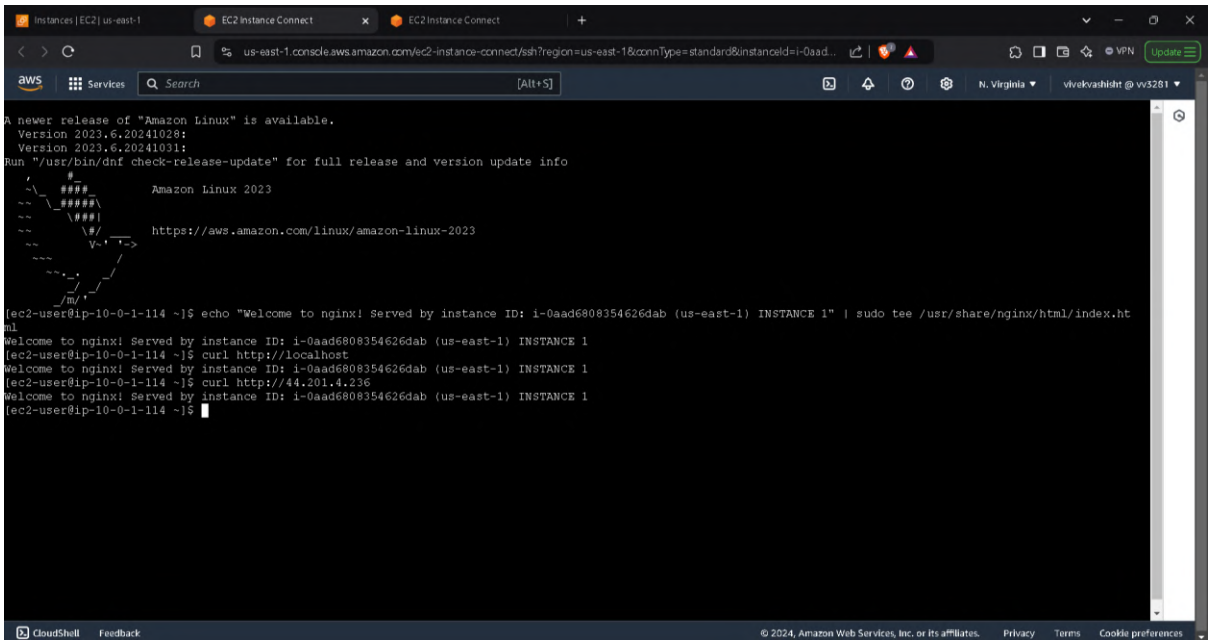
e) Testing ALB Load Balancing via Browser

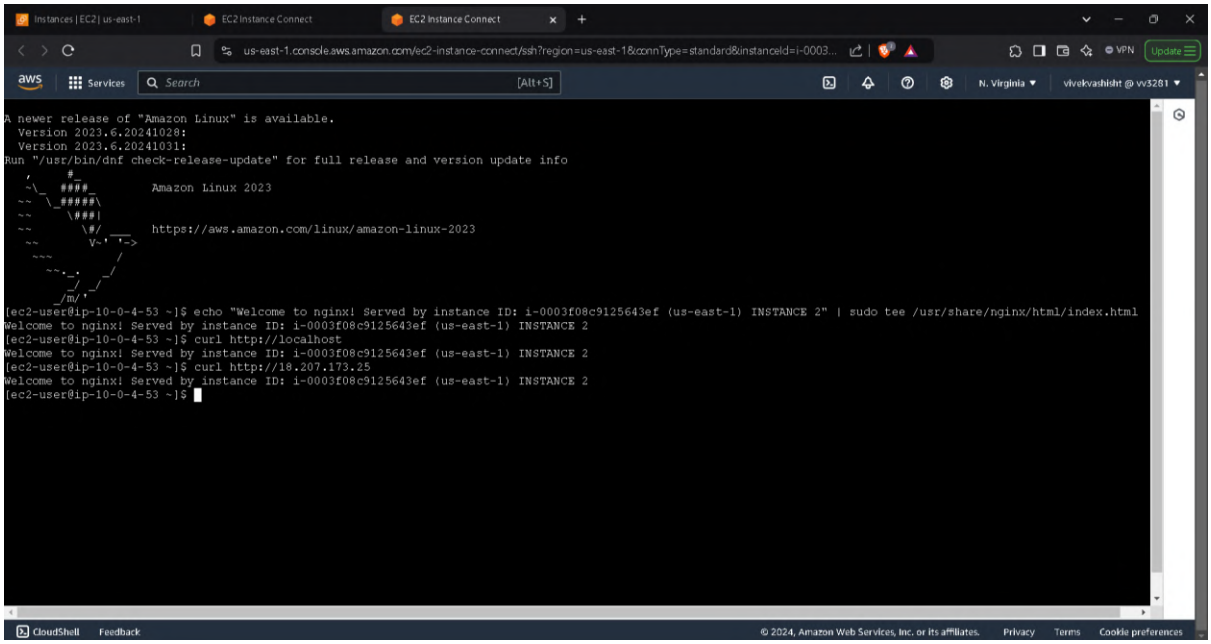
To further verify load balancing, I accessed each ALB's DNS URL in my local browser:

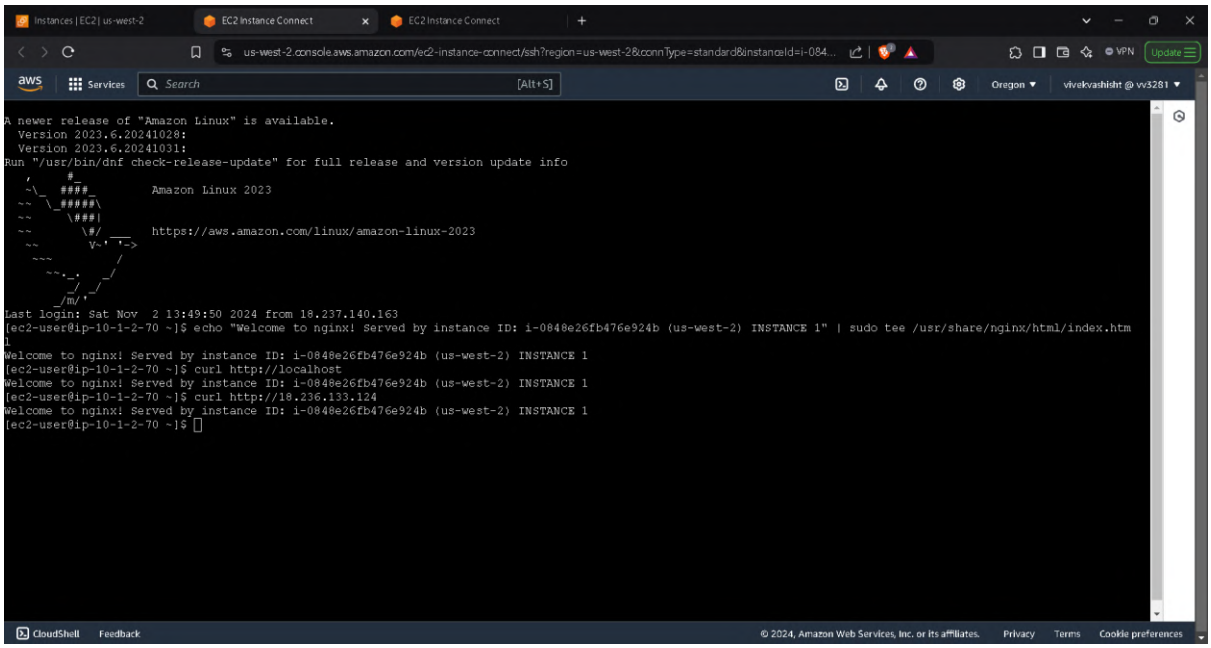
- **Primary ALB (us-east-1):** http://primary-alb-1361659550.us-east-1.elb.amazonaws.com
- **Secondary ALB (us-west-2):** http://secondary-alb-1857056899.us-west-2.elb.amazonaws.com

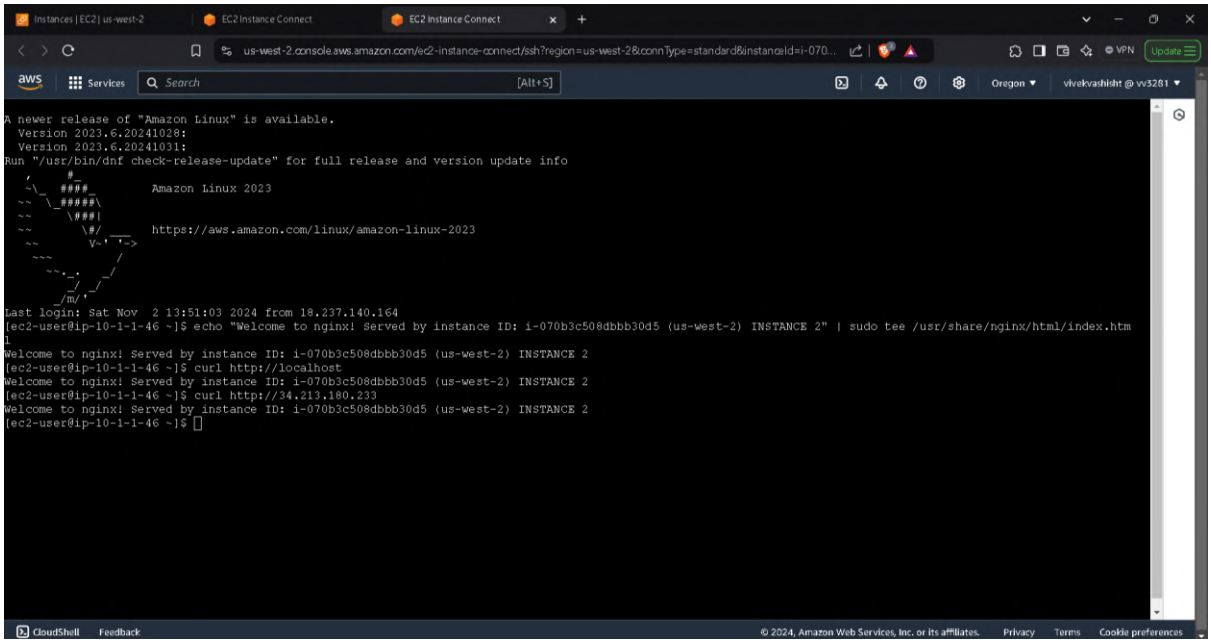
By refreshing the pages multiple times, I observed the instances taking turns to serve the web page, further confirming that the load balancer was distributing the traffic evenly.The ALB testing was successfully completed, with both the primary and secondary ALBs functioning as expected, distributing traffic equally across their respective EC2 instances in each region. This validated the load balancer setup and ensured high availability across the deployment.

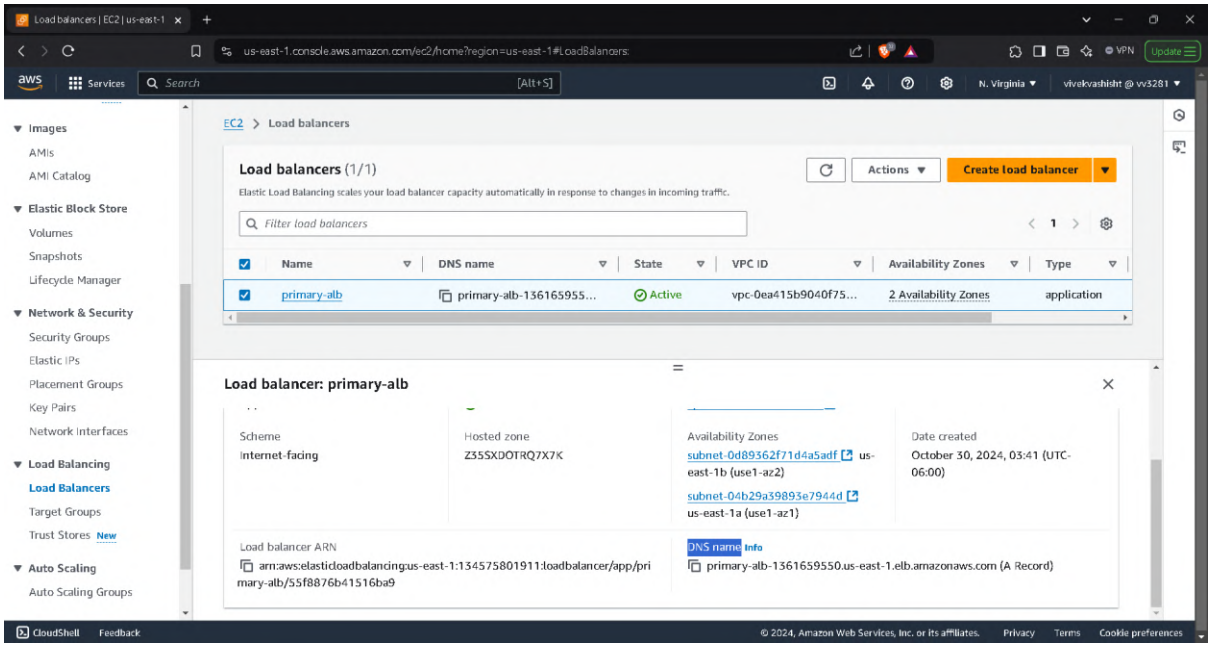
Screenshots

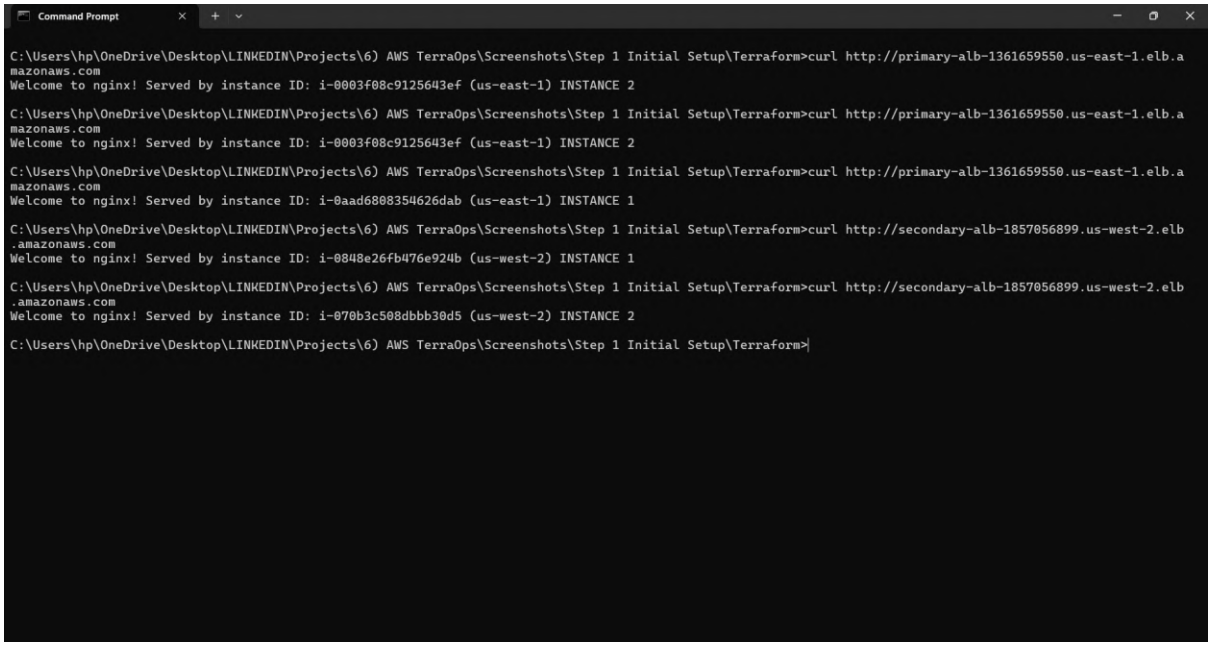


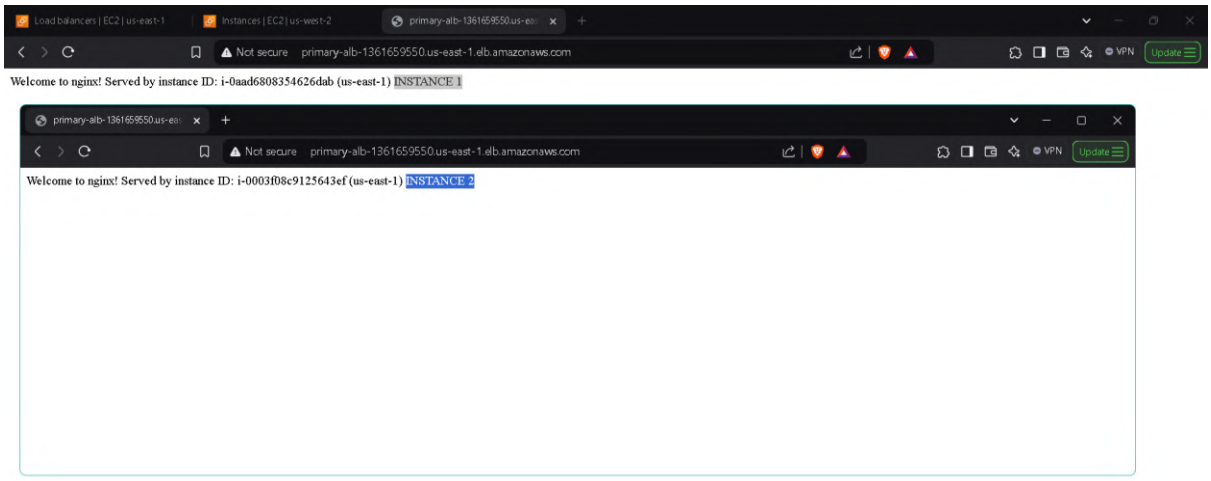


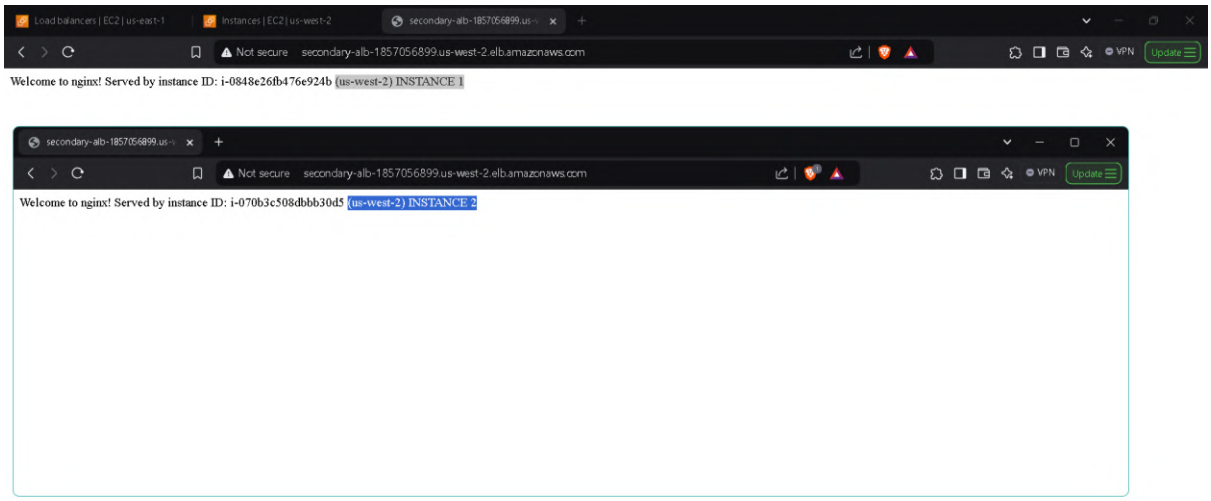

Instances | EC2 | us-west-2
EC2 Instance Connect
us-west-2.console.aws.amazon.com/ec2-instance-connect/ssh?region=us-west-2&connType=standard&instanceId=i-084...
Services
Search
[Alt+S]
Oregon
vivekvashisht @ vv3281
A newer release of "Amazon Linux" is available.
Version 2023.6.20241028:
Version 2023.6.20241031:
Run "/usr/bin/dnf check-release-update" for full release and version update info
Amazon Linux 2023
https://aws.amazon.com/linux/amazon-linux-2023
Last login: Sat Nov 2 13:49:50 2024 from 18.237.140.163
[ec2-user@ip-10-1-2-70 ~]$ echo "Welcome to nginx! Served by instance ID: i-0848e26fb476e924b (us-west-2) INSTANCE 1" | sudo tee /usr/share/nginx/html/index.htm
l
Welcome to nginx! Served by instance ID: i-0848e26fb476e924b (us-west-2) INSTANCE 1
[ec2-user@ip-10-1-2-70 ~]$ curl http://localhost
Welcome to nginx! Served by instance ID: i-0848e26fb476e924b (us-west-2) INSTANCE 1
[ec2-user@ip-10-1-2-70 ~]$ curl http://18.236.133.124
Welcome to nginx! Served by instance ID: i-0848e26fb476e924b (us-west-2) INSTANCE 1
[ec2-user@ip-10-1-2-70 ~]$
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Instances | EC2 | us-west-2
EC2 Instance Connect
us-west-2.console.aws.amazon.com/ec2-instance-connect/ssh?region=us-west-2&connType=standard&instanceId=i-070...
Services
Search
[Alt+S]
Oregon
vivekvashisht @ vv3281
A newer release of "Amazon Linux" is available.
Version 2023.6.20241028:
Version 2023.6.20241031:
Run "/usr/bin/dnf check-release-update" for full release and version update info
Amazon Linux 2023
https://aws.amazon.com/linux/amazon-linux-2023
Last login: Sat Nov 2 13:51:03 2024 from 18.237.140.164
[ec2-user@ip-10-1-1-46 ~]$ echo "Welcome to nginx! Served by instance ID: i-070b3c508dbbb30d5 (us-west-2) INSTANCE 2" | sudo tee /usr/share/nginx/html/index.htm
l
Welcome to nginx! Served by instance ID: i-070b3c508dbbb30d5 (us-west-2) INSTANCE 2
[ec2-user@ip-10-1-1-46 ~]$ curl http://localhost
Welcome to nginx! Served by instance ID: i-070b3c508dbbb30d5 (us-west-2) INSTANCE 2
[ec2-user@ip-10-1-1-46 ~]$ curl http://34.213.180.233
Welcome to nginx! Served by instance ID: i-070b3c508dbbb30d5 (us-west-2) INSTANCE 2
[ec2-user@ip-10-1-1-46 ~]$
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Load balancers | EC2 | us-east-1
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LoadBalancers:
Services
Search
[Alt+S]
N. Virginia
vivekvashisht @ vv3281
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Network & Security
Security Groups
Elastic IPs
Placement Groups
Key Pairs
Network Interfaces
Load Balancing
Load Balancers
Target Groups
Trust Stores New
Auto Scaling
Auto Scaling Groups
EC2 > Load balancers
Load balancers (1/1)
Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.
Actions
Create load balancer
Filter load balancers
Name
DNS name
State
VPC ID
Availability Zones
Type
primary-alb
primary-alb-136165955...
Active
vpc-0ea415b9040f75...
2 Availability Zones
application
Load balancer: primary-alb
Scheme
internet-facing
Hosted zone
Z35SXDOTRQ7X7K
Availability Zones
subnet-0d89362f71d4a5adf us-east-1b (use1-az2)
subnet-04b29a39893e7944d us-east-1a (use1-az1)
Date created
October 30, 2024, 03:41 (UTC-06:00)
Load balancer ARN
arn:aws:elasticloadbalancing:us-east-1:134575801911:loadbalancer/app/primary-alb/55f8876b41516ba9
DNS name Info
primary-alb-1361659550.us-east-1.elb.amazonaws.com (A Record)
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Command Prompt
C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>curl http://primary-alb-1361659550.us-east-1.elb.amazonaws.com
Welcome to nginx! Served by instance ID: i-0003f08c9125643ef (us-east-1) INSTANCE 2
C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>curl http://primary-alb-1361659550.us-east-1.elb.amazonaws.com
Welcome to nginx! Served by instance ID: i-0003f08c9125643ef (us-east-1) INSTANCE 2
C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>curl http://primary-alb-1361659550.us-east-1.elb.amazonaws.com
Welcome to nginx! Served by instance ID: i-0aad6808354626dab (us-east-1) INSTANCE 1
C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>curl http://secondary-alb-1857056899.us-west-2.elb.amazonaws.com
Welcome to nginx! Served by instance ID: i-0848e26fb476e924b (us-west-2) INSTANCE 1
C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>curl http://secondary-alb-1857056899.us-west-2.elb.amazonaws.com
Welcome to nginx! Served by instance ID: i-070b3c508dbbb30d5 (us-west-2) INSTANCE 2
C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>


Not secure primary-alb-1361659550.us-east-1.elb.amazonaws.com
Welcome to nginx! Served by instance ID: i-0aad6808354626dab (us-east-1) INSTANCE 1
Not secure primary-alb-1361659550.us-east-1.elb.amazonaws.com
Welcome to nginx! Served by instance ID: i-0003f08c9125643ef (us-east-1) INSTANCE 2


Not secure secondary-alb-1857056899.us-west-2.elb.amazonaws.com
Welcome to nginx! Served by instance ID: i-0848e26fb476e924b (us-west-2) INSTANCE 1
Not secure secondary-alb-1857056899.us-west-2.elb.amazonaws.com
Welcome to nginx! Served by instance ID: i-070b3c508dbbb30d5 (us-west-2) INSTANCE 2

2) Route 53 Failover Testing

a) Overview

This phase focused on testing the Route 53 failover configuration to ensure that traffic seamlessly switches to the secondary region (us-west-2) if instances in the primary region (us-east-1) become unavailable. Since this was for testing purposes, I updated `route53.tf` to create a private Route 53 hosted zone. This allowed testing failover without requiring a public domain.

b) Updating route53.tf for Private Hosted Zone

To work with a private zone in both VPCs across regions, I updated `route53.tf` to associate the `myinfra.local` private hosted zone with `MainVPC` in us-east-1 and `WestVPC` in us-west-2. This configuration enabled VPC connectivity in each region to handle the failover.

```
# Route 53 Hosted Zone
resource "aws_route53_zone" "main" {
  name = var.route53_zone_domain

  # Associate with the VPC in us-east-1
  vpc {
    vpc_id = aws_vpc.main_vpc.id  # us-east-1 VPC
  }

  # Associate with the VPC in us-west-2
  vpc {
    vpc_id     = aws_vpc.west_vpc.id  # us-west-2 VPC
    vpc_region = "us-west-2"
  }
}

# Primary Route 53 Record (us-east-1)
resource "aws_route53_record" "primary" {
  zone_id = aws_route53_zone.main.zone_id
  name    = "app.${var.route53_zone_domain}"
  type    = "A"

  alias {
    name                   = aws_lb.primary_alb.dns_name
    zone_id                = aws_lb.primary_alb.zone_id
    evaluate_target_health = true
  }

  set_identifier = "Primary"
  failover_routing_policy {
    type = "PRIMARY"
  }
}

# Secondary Route 53 Record (us-west-2)
resource "aws_route53_record" "secondary" {
  zone_id = aws_route53_zone.main.zone_id
  name    = "app.${var.route53_zone_domain}"
```

```
  type    = "A"

  alias {
    name                   = aws_lb.secondary_alb.dns_name
    zone_id                = aws_lb.secondary_alb.zone_id
    evaluate_target_health = true
  }

  set_identifier = "Secondary"
  failover_routing_policy {
    type = "SECONDARY"
  }
}

# Route 53 Health Checks

# Primary Health Check (us-east-1)
resource "aws_route53_health_check" "primary_health_check" {
  fqdn              = aws_lb.primary_alb.dns_name  # Use ALB DNS Name
  type              = "HTTP"
  port              = 80
  request_interval  = 30
  failure_threshold = 3
}

# Secondary Health Check (us-west-2)
resource "aws_route53_health_check" "secondary_health_check" {
  fqdn              = aws_lb.secondary_alb.dns_name  # Use ALB DNS Name
  type              = "HTTP"
  port              = 80
  request_interval  = 30
  failure_threshold = 3
}
```

c) Applying Updated route53.tf Configuration

After saving the updated `route53.tf`, I ran the following commands in the terminal to destroy the existing resources and apply the new configuration:

```
terraform destroy -target=aws_route53_zone.main -
target=aws_route53_health_check.primary_health_check -
target=aws_route53_health_check.secondary_health_check
terraform apply -auto-approve
```

After the successful application, the private hosted zone and health checks were redeployed.

d) Verifying Route 53 Setup

In the AWS Console:

1. Under Route 53, I confirmed that the `myinfra.local` private hosted zone was created with A records for Primary and Secondary ALBs.
2. I verified that `MainVPC` in us-east-1 and `WestVPC` in us-west-2 were associated with the hosted zone.
3. Both health checks for the primary and secondary ALBs were in a healthy state.

e) Enabling DNS Hostnames for VPCs

To enable instance-level DNS resolution, I edited the DNS settings for both `MainVPC` and `WestVPC`, enabling DNS hostnames.

f) Initial Testing with Failover Disabled

To validate the initial setup, I connected to an EC2 instance in us-west-2 and issued the following command multiple times:

```
curl http://app.myinfra.local
```

The output showed responses from the primary region instances in us-east-1:

```
Welcome to nginx! Served by instance ID: i-0aad6808354626dab (us-east-1)
INSTANCE 1
Welcome to nginx! Served by instance ID: i-0003f08c9125643ef (us-east-1)
INSTANCE 2
```

g) Simulating Primary Region Outage

To test the failover:

1. I disabled both scale-up and scale-down policies in `AppASGInstance` in us-east-1.
2. Detached both instances in `AppASGInstance`, set the desired capacity to 0, and stopped the instances from the AWS Console.

h) Observing Failover Behavior

With primary instances offline, I checked the health checks in Route 53 and saw `primary-alb` transitioning to an unhealthy state. I then issued the `curl` command again from the us-west-2 instance:

```
curl http://app.myinfra.local
```

This time, the output showed responses from the secondary region instances in us-west-2:

```
Welcome to nginx! Served by instance ID: i-0848e26fb476e924b (us-west-2)
INSTANCE 1
Welcome to nginx! Served by instance ID: i-070b3c508dbbb30d5 (us-west-2)
INSTANCE 2
```

i) Testing Failback to Primary Region

To confirm the failback process:

1. I re-enabled the Auto Scaling policies in `AppASGInstance` and restarted the stopped instances in us-east-1.
2. Once the `primary-alb` health check returned to a healthy state, I ran the `curl` command from the us-west-2 instance:

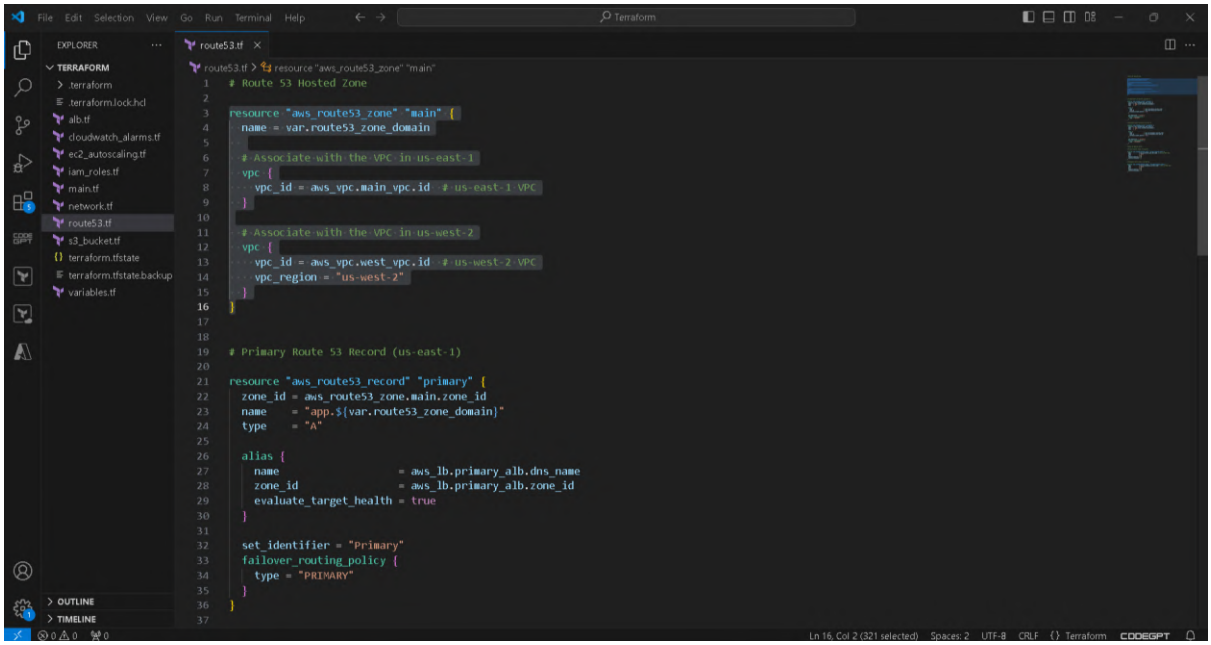
```
curl http://app.myinfra.local
```

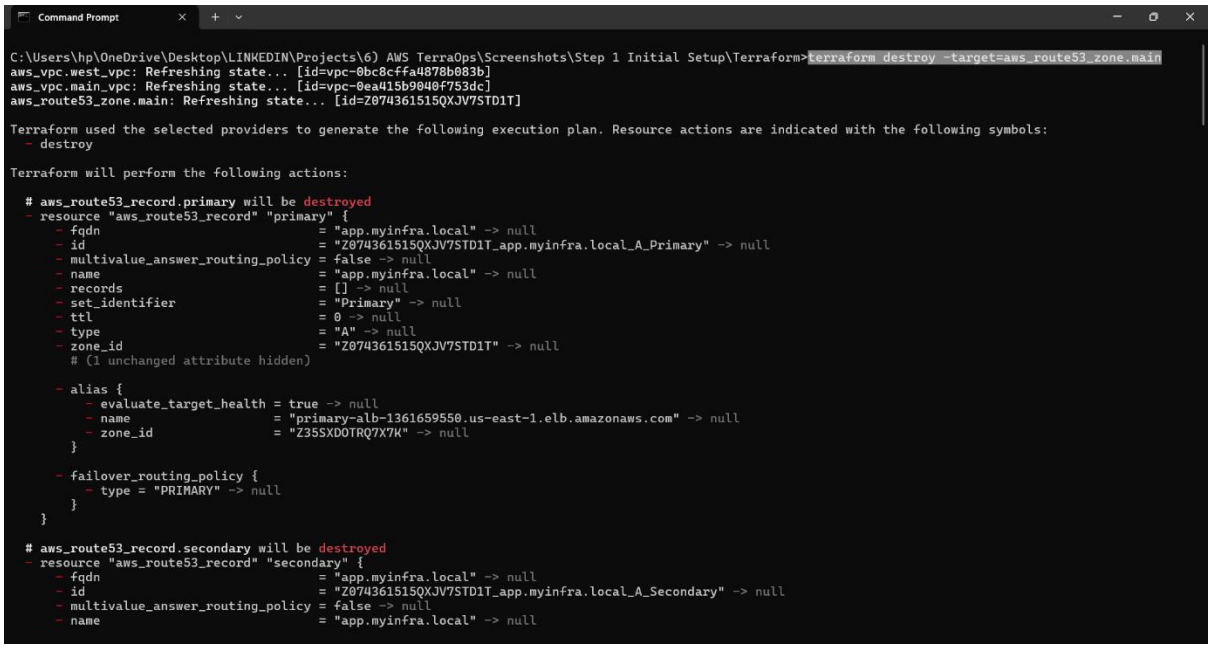
The response returned to instances in us-east-1, confirming failback:

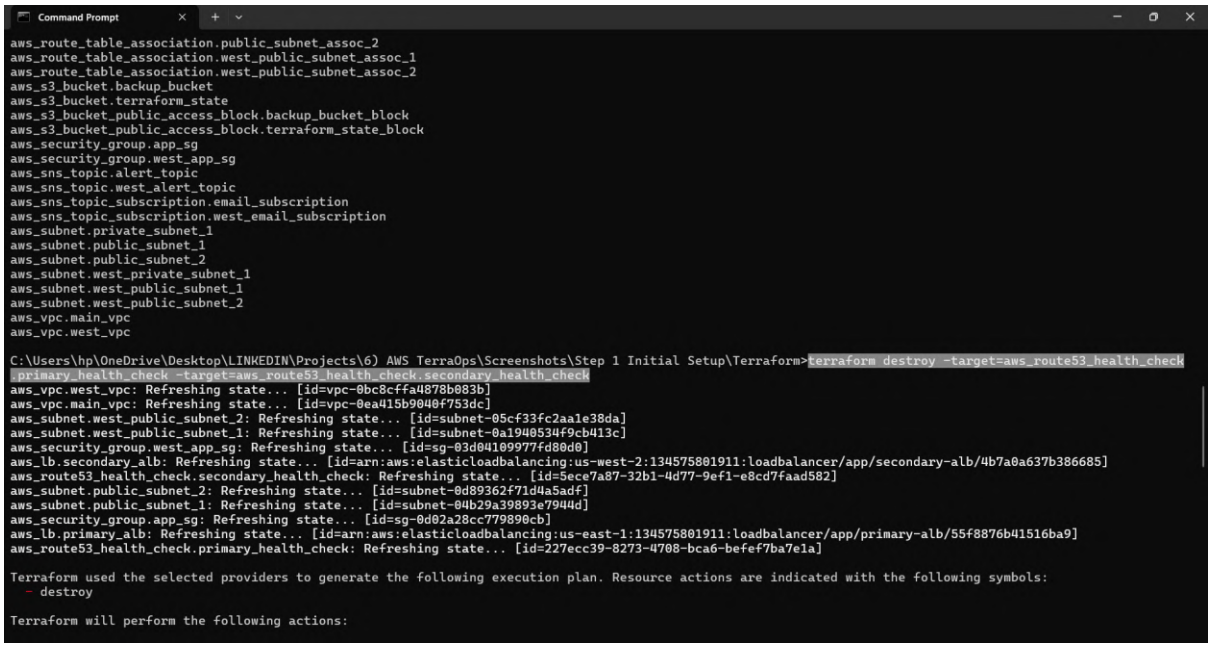
```
Welcome to nginx! Served by instance ID: i-0aad6808354626dab (us-east-1)
INSTANCE 1
Welcome to nginx! Served by instance ID: i-0003f08c9125643ef (us-east-1)
INSTANCE 2
```

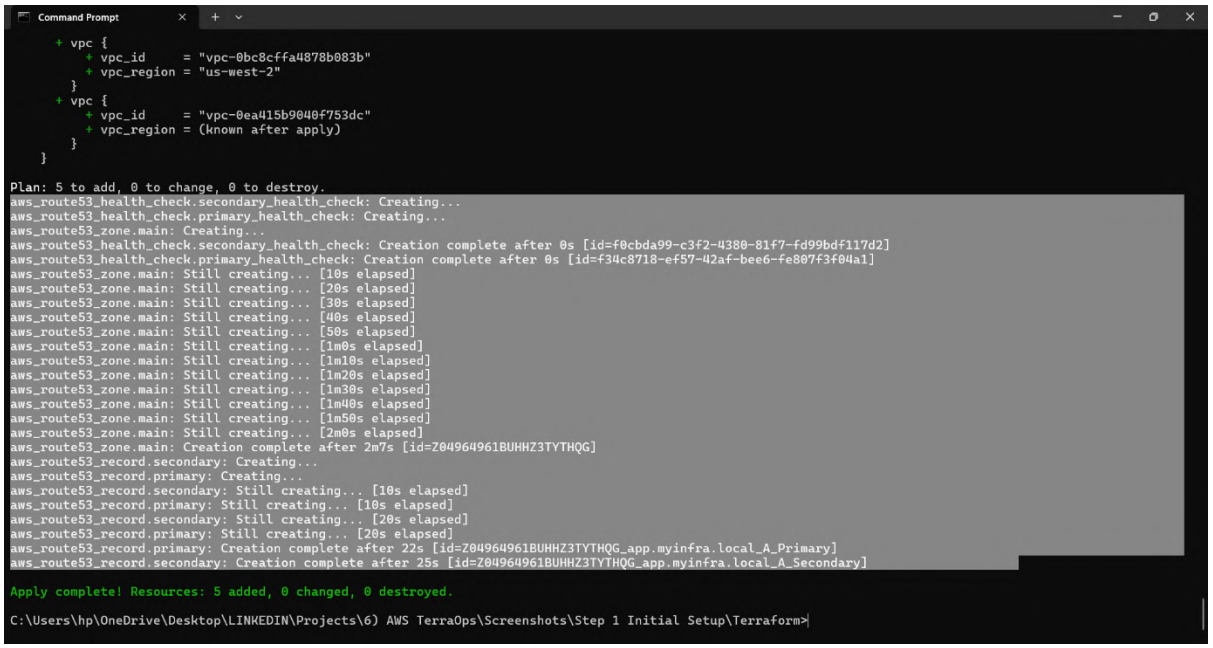
The Route 53 failover testing was successful, with traffic automatically redirecting to us-west-2 when us-east-1 was down and reverting once us-east-1 was available again. This testing confirmed the setup's robustness and reliability in supporting failover for high availability.

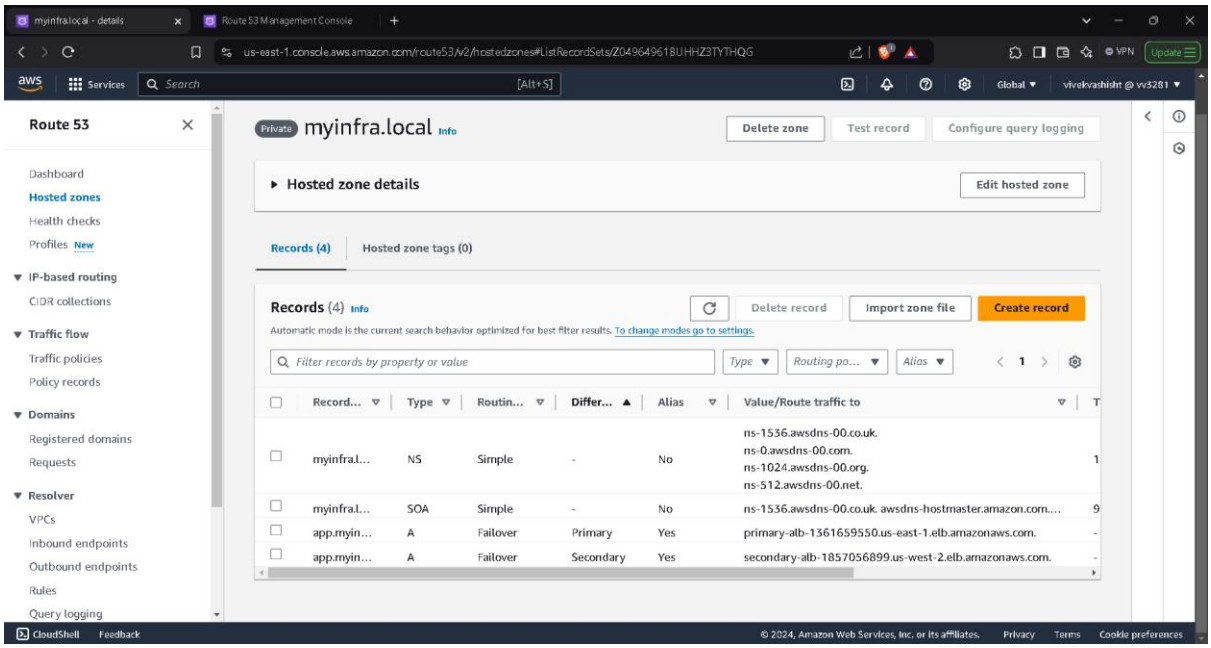
Screenshots

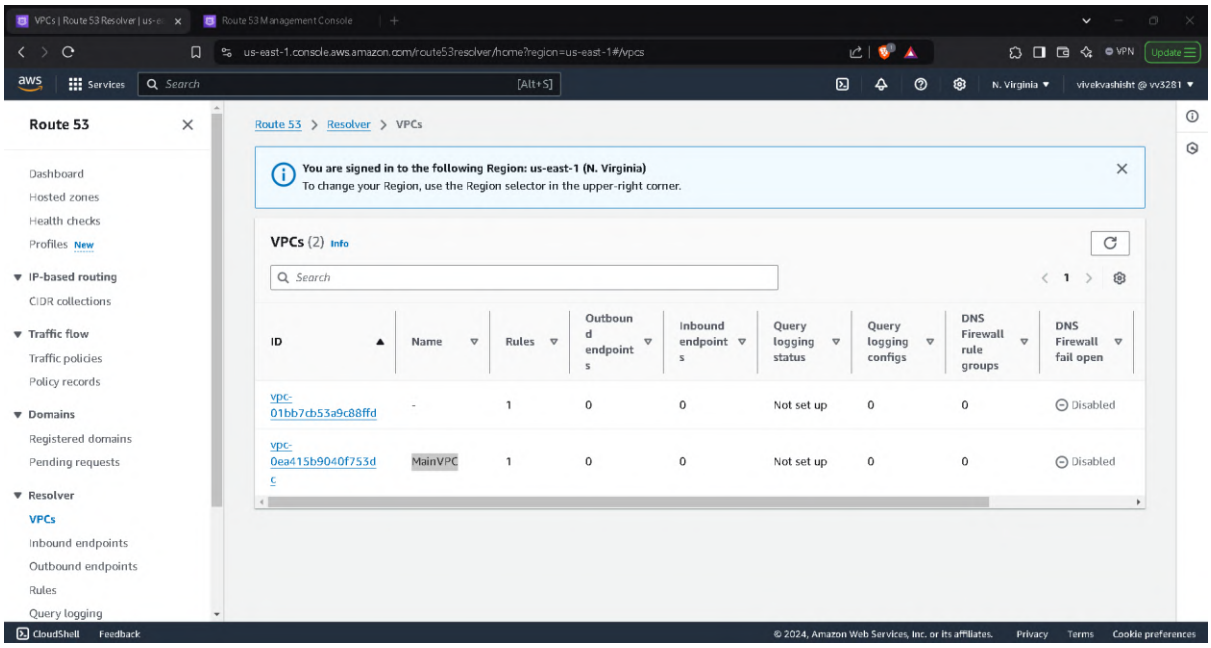


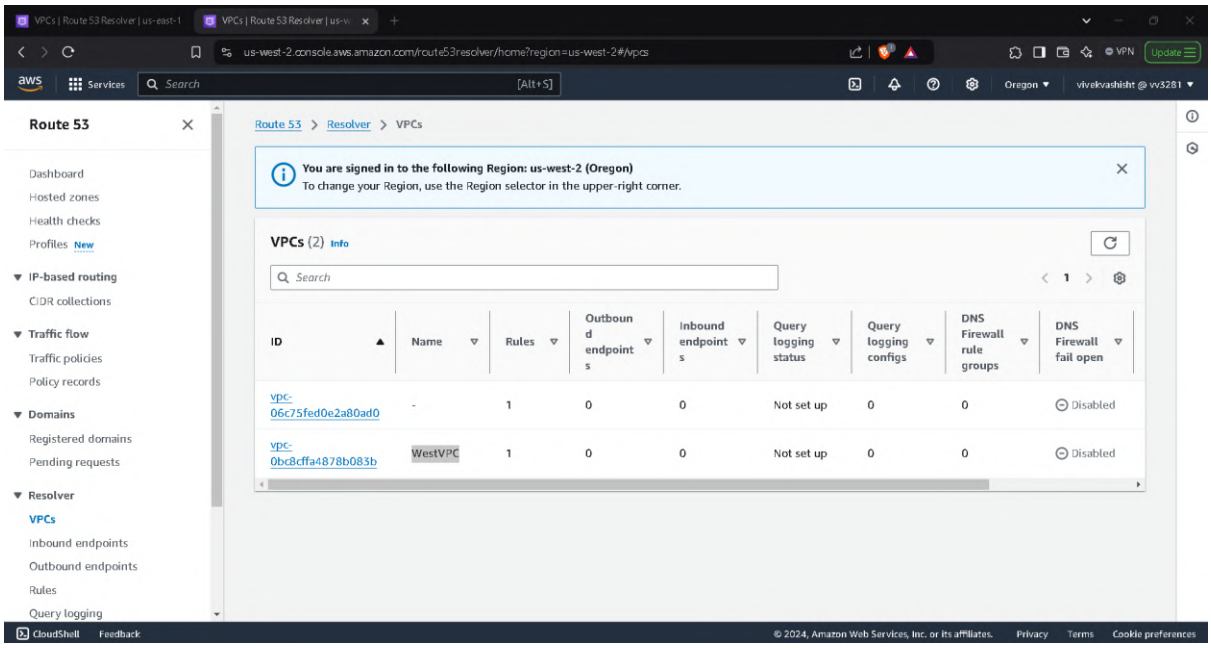

Command Prompt
C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>terraform destroy -target=aws_route53_zone.main
aws_vpc.west_vpc: Refreshing state... [id=vpc-0bc8cffa4878b083b]
aws_vpc.main_vpc: Refreshing state... [id=vpc-0ea415b9040f753dc]
aws_route53_zone.main: Refreshing state... [id=Z074361515QXJV7STD1T]
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
- destroy
Terraform will perform the following actions:
# aws_route53_record.primary will be destroyed
- resource "aws_route53_record" "primary" {
- fqdn = "app.myinfra.local" -> null
- id = "Z074361515QXJV7STD1T_app.myinfra.local_A_Primary" -> null
- multivalue_answer_routing_policy = false -> null
- name = "app.myinfra.local" -> null
- records = [] -> null
- set_identifier = "Primary" -> null
- ttl = 0 -> null
- type = "A" -> null
- zone_id = "Z074361515QXJV7STD1T" -> null
# (1 unchanged attribute hidden)
- alias {
- evaluate_target_health = true -> null
- name = "primary-alb-1361659550.us-east-1.elb.amazonaws.com" -> null
- zone_id = "Z35SXDOTRQ7X7K" -> null
}
- failover_routing_policy {
- type = "PRIMARY" -> null
}
}
# aws_route53_record.secondary will be destroyed
- resource "aws_route53_record" "secondary" {
- fqdn = "app.myinfra.local" -> null
- id = "Z074361515QXJV7STD1T_app.myinfra.local_A_Secondary" -> null
- multivalue_answer_routing_policy = false -> null
- name = "app.myinfra.local" -> null

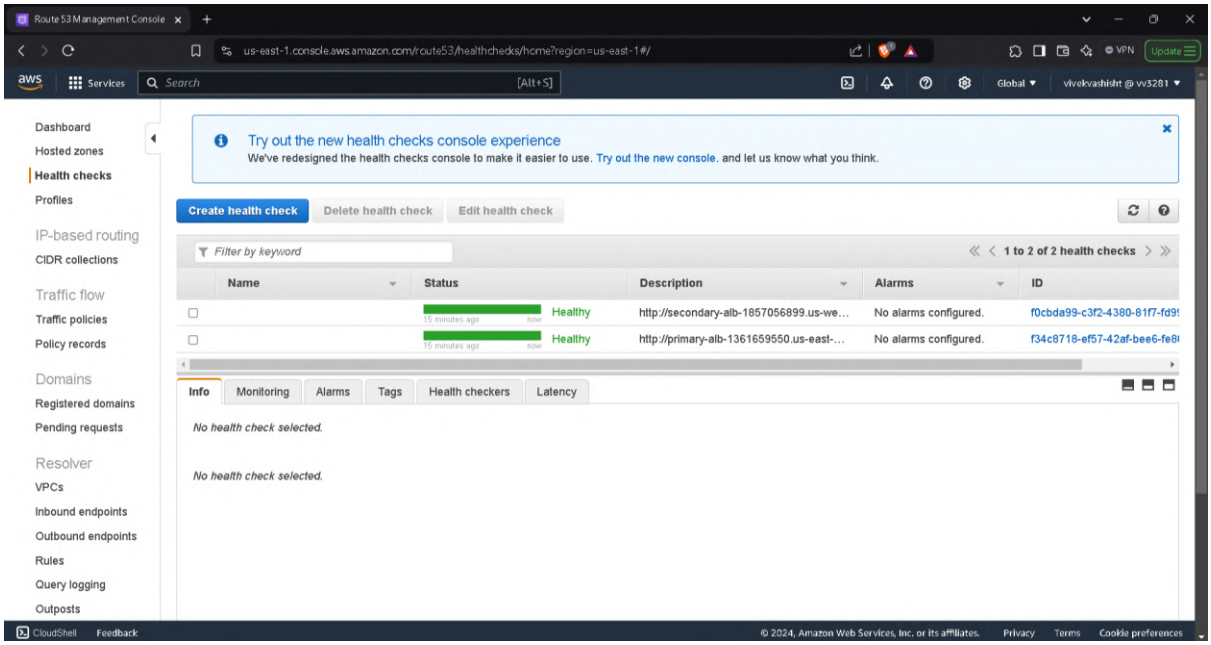

Command Prompt
aws_route_table_association.public_subnet_assoc_2
aws_route_table_association.west_public_subnet_assoc_1
aws_route_table_association.west_public_subnet_assoc_2
aws_s3_bucket.backup_bucket
aws_s3_bucket.terraform_state
aws_s3_bucket_public_access_block.backup_bucket_block
aws_s3_bucket_public_access_block.terraform_state_block
aws_security_group.app_sg
aws_security_group.west_app_sg
aws_sns_topic.alert_topic
aws_sns_topic.west_alert_topic
aws_sns_topic_subscription.email_subscription
aws_sns_topic_subscription.west_email_subscription
aws_subnet.private_subnet_1
aws_subnet.public_subnet_1
aws_subnet.public_subnet_2
aws_subnet.west_private_subnet_1
aws_subnet.west_public_subnet_1
aws_subnet.west_public_subnet_2
aws_vpc.main_vpc
aws_vpc.west_vpc
C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>terraform destroy -target=aws_route53_health_check
.primary_health_check -target=aws_route53_health_check.secondary_health_check
aws_vpc.west_vpc: Refreshing state... [id=vpc-0bc8cffa4878b083b]
aws_vpc.main_vpc: Refreshing state... [id=vpc-0ea415b9040f753dc]
aws_subnet.west_public_subnet_2: Refreshing state... [id=subnet-05cf33fc2aa1e38da]
aws_subnet.west_public_subnet_1: Refreshing state... [id=subnet-0a1940534f9cb413c]
aws_security_group.west_app_sg: Refreshing state... [id=sg-03d04109977fd80d0]
aws_lb.secondary_alb: Refreshing state... [id=arn:aws:elasticloadbalancing:us-west-2:134575801911:loadbalancer/app/secondary-alb/4b7a0a637b386685]
aws_route53_health_check.secondary_health_check: Refreshing state... [id=5ece7a87-32b1-4d77-9ef1-e8cd7faad582]
aws_subnet.public_subnet_2: Refreshing state... [id=subnet-0d89362f71d4a5adf]
aws_subnet.public_subnet_1: Refreshing state... [id=subnet-04b29a39893e7944d]
aws_security_group.app_sg: Refreshing state... [id=sg-0d02a28cc779890cb]
aws_lb.primary_alb: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:134575801911:loadbalancer/app/primary-alb/55f8876b41516ba9]
aws_route53_health_check.primary_health_check: Refreshing state... [id=227ecc39-8273-4708-bca6-befef7ba7e1a]
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
- destroy
Terraform will perform the following actions:

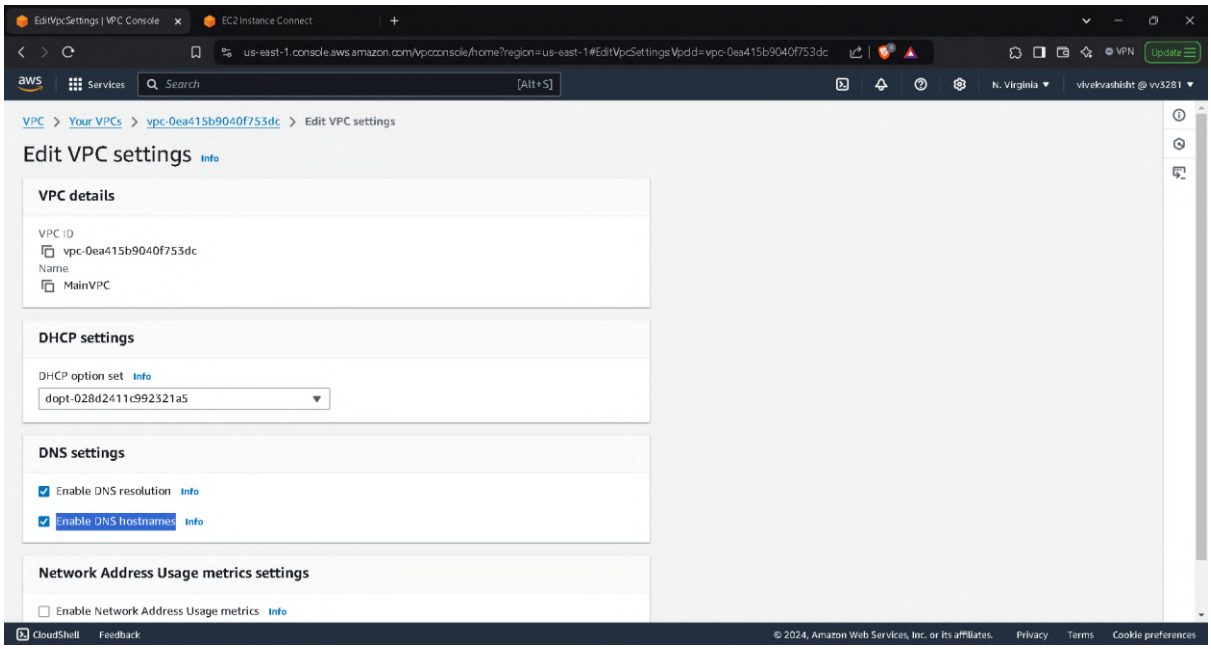

Command Prompt
+ vpc {
+ vpc_id = "vpc-0bc8cffa4878b083b"
+ vpc_region = "us-west-2"
}
+ vpc {
+ vpc_id = "vpc-0ea415b9040f753dc"
+ vpc_region = (known after apply)
}
}
Plan: 5 to add, 0 to change, 0 to destroy.
aws_route53_health_check.secondary_health_check: Creating...
aws_route53_health_check.primary_health_check: Creating...
aws_route53_zone.main: Creating...
aws_route53_health_check.secondary_health_check: Creation complete after 0s [id=f0cbda99-c3f2-4380-81f7-fd99bdf117d2]
aws_route53_health_check.primary_health_check: Creation complete after 0s [id=f34c8718-ef57-42af-bee6-fe807f3f04a1]
aws_route53_zone.main: Still creating... [10s elapsed]
aws_route53_zone.main: Still creating... [20s elapsed]
aws_route53_zone.main: Still creating... [30s elapsed]
aws_route53_zone.main: Still creating... [40s elapsed]
aws_route53_zone.main: Still creating... [50s elapsed]
aws_route53_zone.main: Still creating... [1m0s elapsed]
aws_route53_zone.main: Still creating... [1m10s elapsed]
aws_route53_zone.main: Still creating... [1m20s elapsed]
aws_route53_zone.main: Still creating... [1m30s elapsed]
aws_route53_zone.main: Still creating... [1m40s elapsed]
aws_route53_zone.main: Still creating... [1m50s elapsed]
aws_route53_zone.main: Still creating... [2m0s elapsed]
aws_route53_zone.main: Creation complete after 2m7s [id=Z04964961BUHHZ3TYTHQG]
aws_route53_record.secondary: Creating...
aws_route53_record.primary: Creating...
aws_route53_record.secondary: Still creating... [10s elapsed]
aws_route53_record.primary: Still creating... [10s elapsed]
aws_route53_record.secondary: Still creating... [20s elapsed]
aws_route53_record.primary: Still creating... [20s elapsed]
aws_route53_record.primary: Creation complete after 22s [id=Z04964961BUHHZ3TYTHQG_app.myinfra.local_A_Primary]
aws_route53_record.secondary: Creation complete after 25s [id=Z04964961BUHHZ3TYTHQG_app.myinfra.local_A_Secondary]
Apply complete! Resources: 5 added, 0 changed, 0 destroyed.
C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>

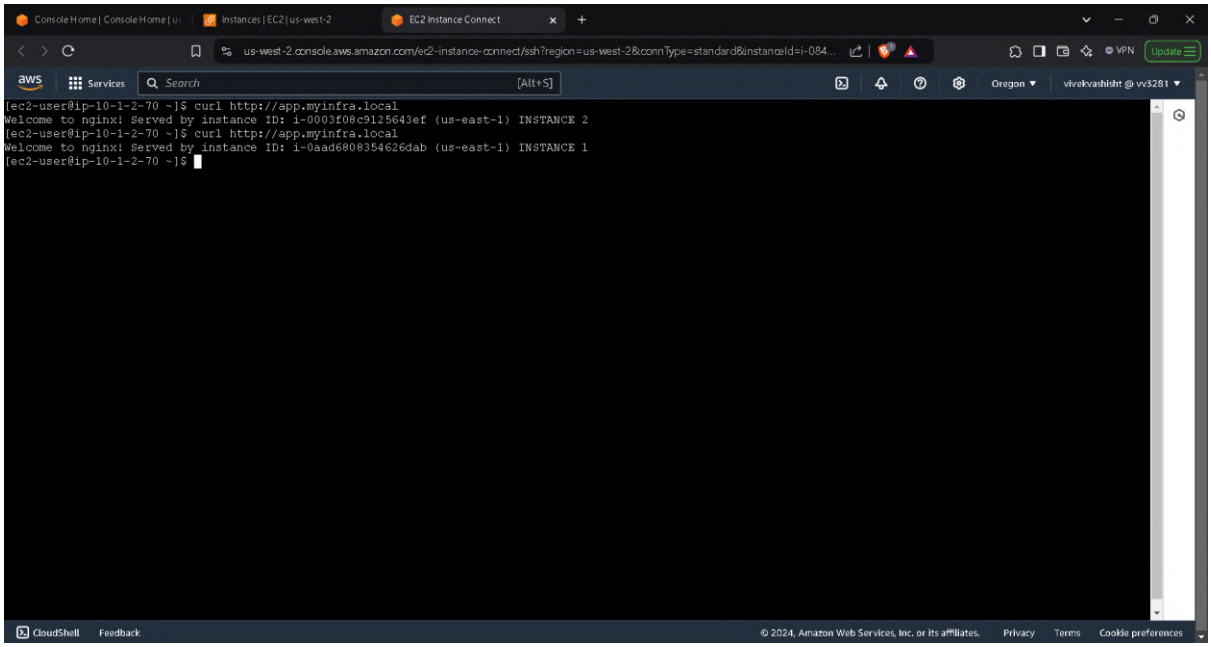

myinfra.local - details
Route 53 Management Console
us-east-1.console.aws.amazon.com/route53/v2/hostedzones#ListRecordSets/Z049649618UHHZ3TYTHQG
Route 53
Dashboard
Hosted zones
Health checks
Profiles New
IP-based routing
CIDR collections
Traffic flow
Traffic policies
Policy records
Domains
Registered domains
Requests
Resolver
VPCs
Inbound endpoints
Outbound endpoints
Rules
Query logging
Private myinfra.local Info
Delete zone
Test record
Configure query logging
Hosted zone details
Edit hosted zone
Records (4)
Hosted zone tags (0)
Records (4) Info
Delete record
Import zone file
Create record
Automatic mode is the current search behavior optimized for best filter results. To change modes go to settings.
Filter records by property or value
Record... Type Routin... Differ... Alias Value/Route traffic to
myinfra.l... NS Simple - No ns-1536.awsdns-00.co.uk. ns-0.awsdns-00.com. ns-1024.awsdns-00.org. ns-512.awsdns-00.net.
myinfra.l... SOA Simple - No ns-1536.awsdns-00.co.uk. awsdns-hostmaster.amazon.com....
app.myin... A Failover Primary Yes primary-alb-1361659550.us-east-1.elb.amazonaws.com.
app.myin... A Failover Secondary Yes secondary-alb-1857056899.us-west-2.elb.amazonaws.com.
CloudShell Feedback
© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

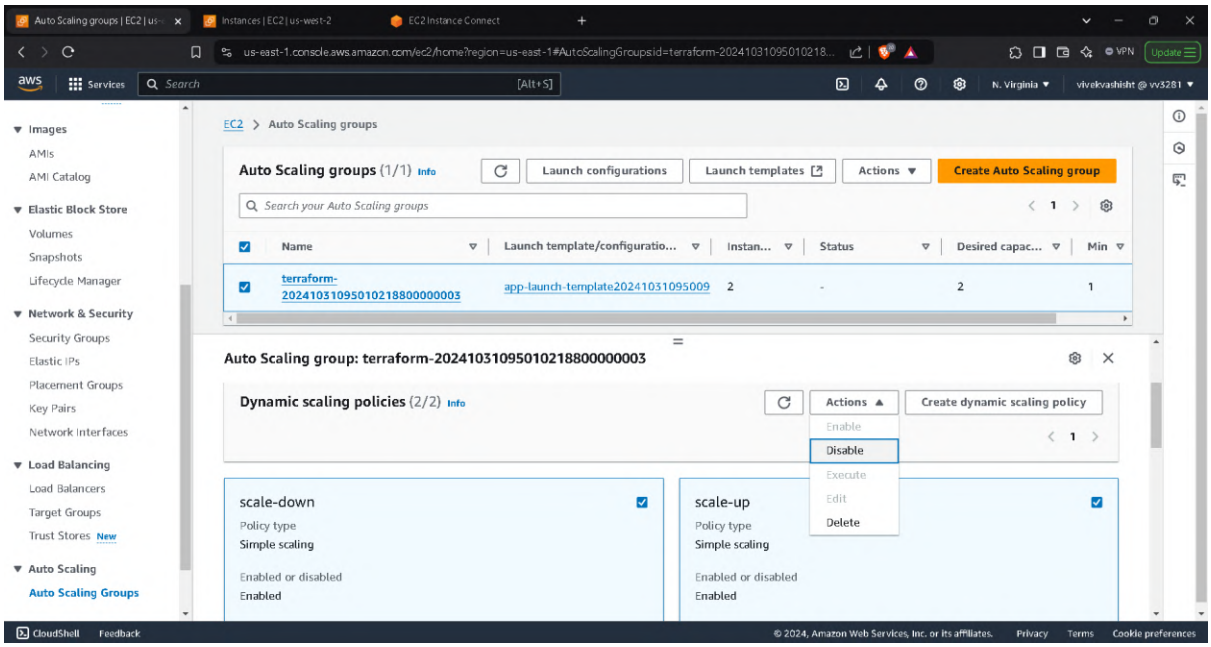

us-east-1.console.aws.amazon.com/route53resolver/home?region=us-east-1#/vpcs
Route 53
Dashboard
Hosted zones
Health checks
Profiles New
IP-based routing
CIDR collections
Traffic flow
Traffic policies
Policy records
Domains
Registered domains
Pending requests
Resolver
VPCs
Inbound endpoints
Outbound endpoints
Rules
Query logging
Route 53 > Resolver > VPCs
You are signed in to the following Region: us-east-1 (N. Virginia)
To change your Region, use the Region selector in the upper-right corner.
VPCs (2) Info
ID Name Rules Outbound endpoints Inbound endpoints Query logging status Query logging configs DNS Firewall rule groups DNS Firewall fail open
vpc-01bb7cb53a9c88ffd - 1 0 0 Not set up 0 0 Disabled
vpc-0ea415b9040f753dc MainVPC 1 0 0 Not set up 0 0 Disabled
CloudShell Feedback
© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

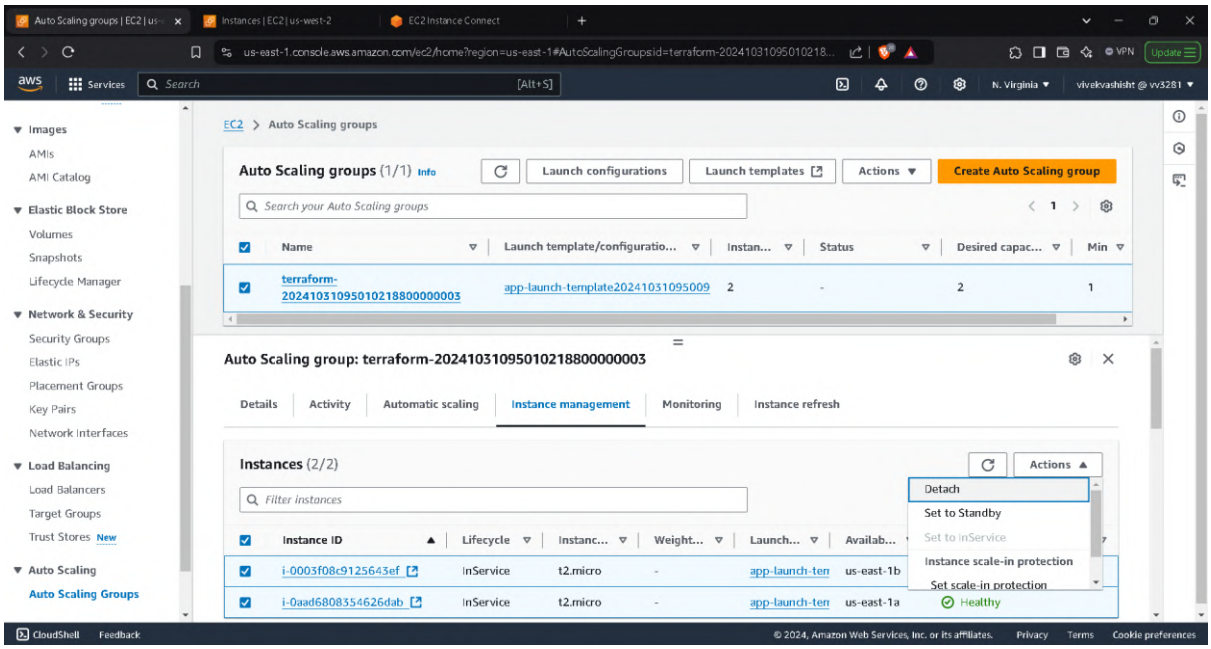

us-west-2.console.aws.amazon.com/route53resolver/home?region=us-west-2#/vpcs
Route 53
Dashboard
Hosted zones
Health checks
Profiles New
IP-based routing
CIDR collections
Traffic flow
Traffic policies
Policy records
Domains
Registered domains
Pending requests
Resolver
VPCs
Inbound endpoints
Outbound endpoints
Rules
Query logging
Route 53 > Resolver > VPCs
You are signed in to the following Region: us-west-2 (Oregon)
To change your Region, use the Region selector in the upper-right corner.
VPCs (2) Info
ID Name Rules Outbound endpoints Inbound endpoints Query logging status Query logging configs DNS Firewall rule groups DNS Firewall fail open
vpc-06c75fed0e2a80ad0 - 1 0 0 Not set up 0 0 Disabled
vpc-0bc8cffa4878b083b WestVPC 1 0 0 Not set up 0 0 Disabled
CloudShell Feedback
© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

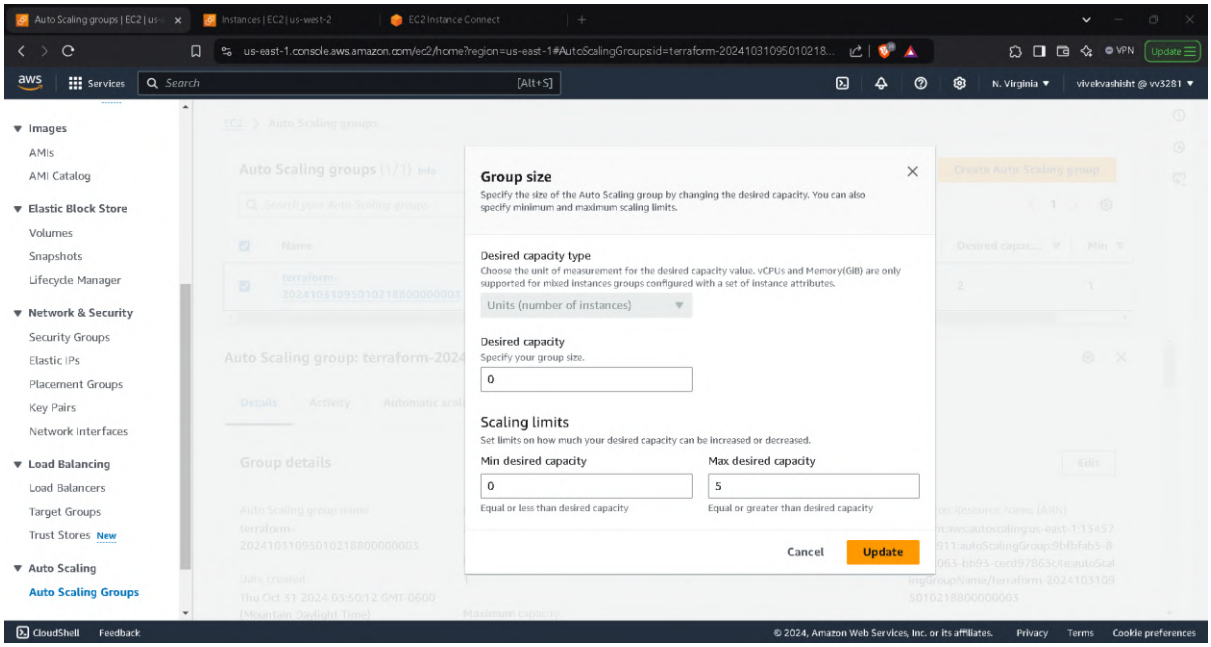

Route 53 Management Console
us-east-1.console.aws.amazon.com/route53/healthchecks/home?region=us-east-1#/
Dashboard
Hosted zones
Health checks
Profiles
IP-based routing
CIDR collections
Traffic flow
Traffic policies
Policy records
Domains
Registered domains
Pending requests
Resolver
VPCs
Inbound endpoints
Outbound endpoints
Rules
Query logging
Outposts
Try out the new health checks console experience
We've redesigned the health checks console to make it easier to use. Try out the new console, and let us know what you think.
Create health check
Delete health check
Edit health check
Filter by keyword
1 to 2 of 2 health checks
Name
Status
Description
Alarms
ID
Healthy
http://secondary-alb-1857056899.us-we...
No alarms configured.
Healthy
http://primary-alb-1361659550.us-east-...
No alarms configured.
Info
Monitoring
Alarms
Tags
Health checkers
Latency
No health check selected.
No health check selected.
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

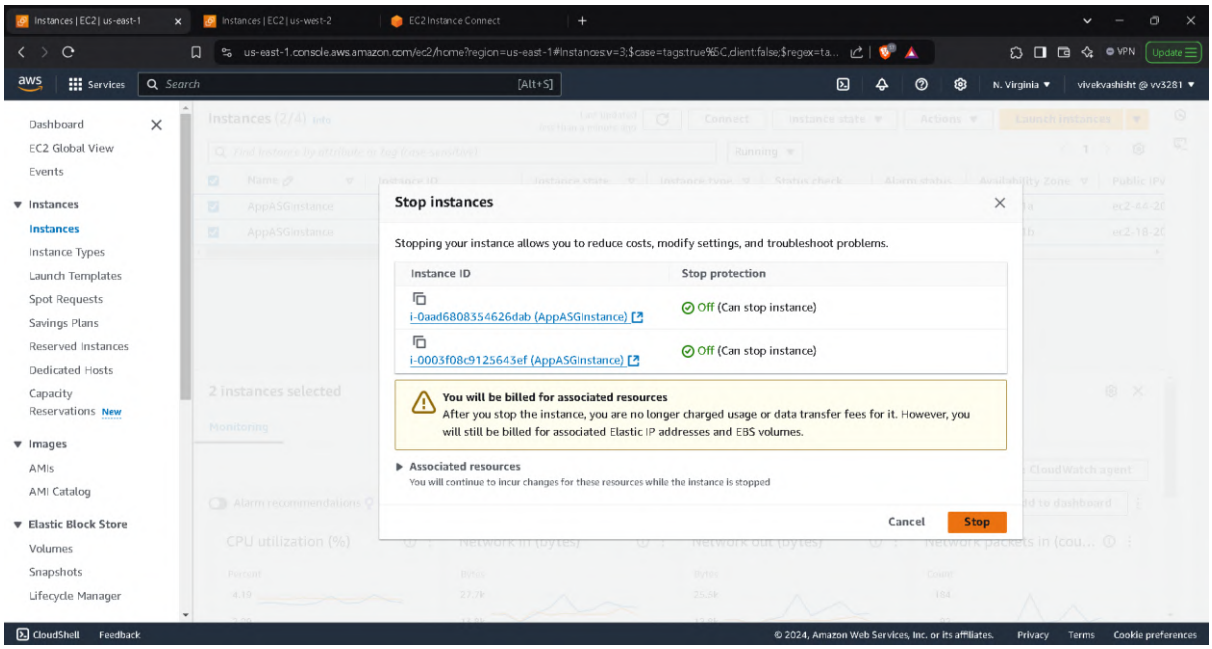

EditVpcSettings | VPC Console
EC2 Instance Connect
us-east-1.console.aws.amazon.com/vpcconsole/home?region=us-east-1#EditVpcSettings:VpcId=vpc-0ea415b9040f753dc
VPC > Your VPCs > vpc-0ea415b9040f753dc > Edit VPC settings
Edit VPC settings Info
VPC details
VPC ID
vpc-0ea415b9040f753dc
Name
MainVPC
DHCP settings
DHCP option set Info
dopt-028d2411c992321a5
DNS settings
Enable DNS resolution Info
Enable DNS hostnames Info
Network Address Usage metrics settings
Enable Network Address Usage metrics Info
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

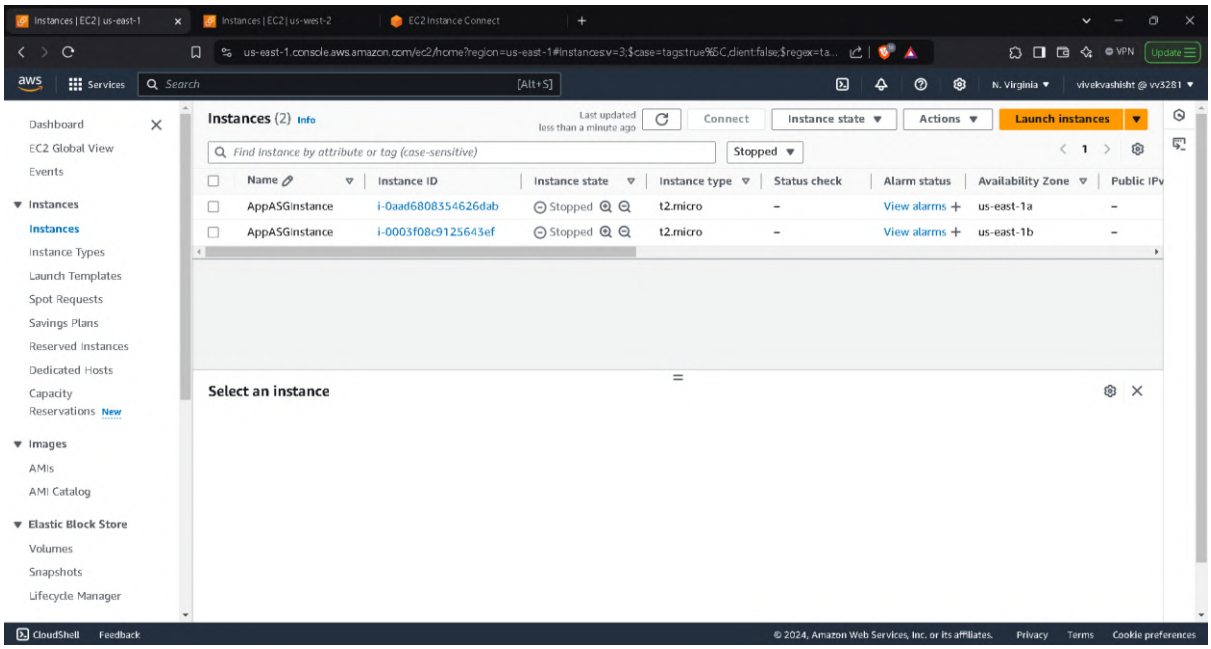

Console Home | Console Home | us
Instances | EC2 | us-west-2
EC2 Instance Connect
us-west-2.console.aws.amazon.com/ec2-instance-connect/ssh?region=us-west-2&connType=standard&instanceId=i-084...
[ec2-user@ip-10-1-2-70 ~]$ curl http://app.myinfra.local
Welcome to nginx! Served by instance ID: i-0003f08c9125643ef (us-east-1) INSTANCE 2
[ec2-user@ip-10-1-2-70 ~]$ curl http://app.myinfra.local
Welcome to nginx! Served by instance ID: i-0aad6808354626dab (us-east-1) INSTANCE 1
[ec2-user@ip-10-1-2-70 ~]$
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

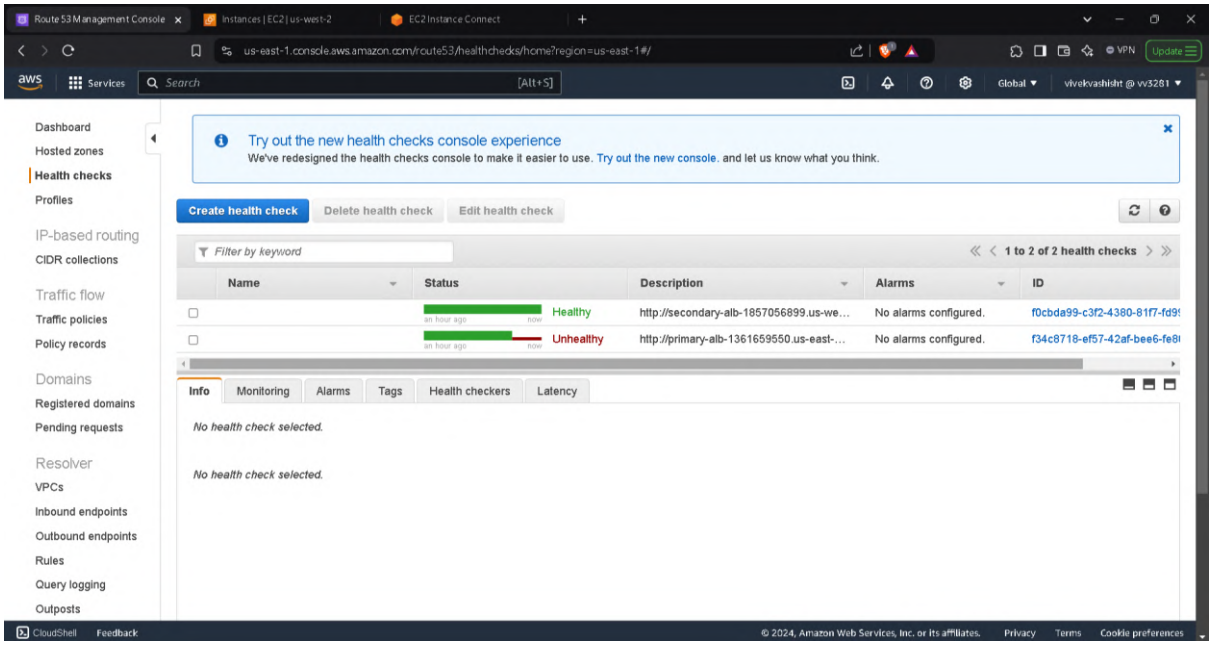

Auto Scaling groups | EC2 | us-
Instances | EC2 | us-west-2
EC2 Instance Connect
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#AutoScalingGroups:id=terraform-20241031095010218...
Update
aws
Services
Search
[Alt+S]
N. Virginia
vivekvashisht @ vv3281
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Network & Security
Security Groups
Elastic IPs
Placement Groups
Key Pairs
Network Interfaces
Load Balancing
Load Balancers
Target Groups
Trust Stores New
Auto Scaling
Auto Scaling Groups
EC2 > Auto Scaling groups
Auto Scaling groups (1/1) Info
Launch configurations
Launch templates
Actions
Create Auto Scaling group
Search your Auto Scaling groups
Name
Launch template/configuratio...
Instan...
Status
Desired capac...
Min
terraform-20241031095010218800000003
app-launch-template20241031095009
2
2
1
Auto Scaling group: terraform-20241031095010218800000003
Dynamic scaling policies (2/2) Info
Actions
Create dynamic scaling policy
Enable
Disable
Execute
Edit
Delete
scale-down
Policy type
Simple scaling
Enabled or disabled
Enabled
scale-up
Policy type
Simple scaling
Enabled or disabled
Enabled
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

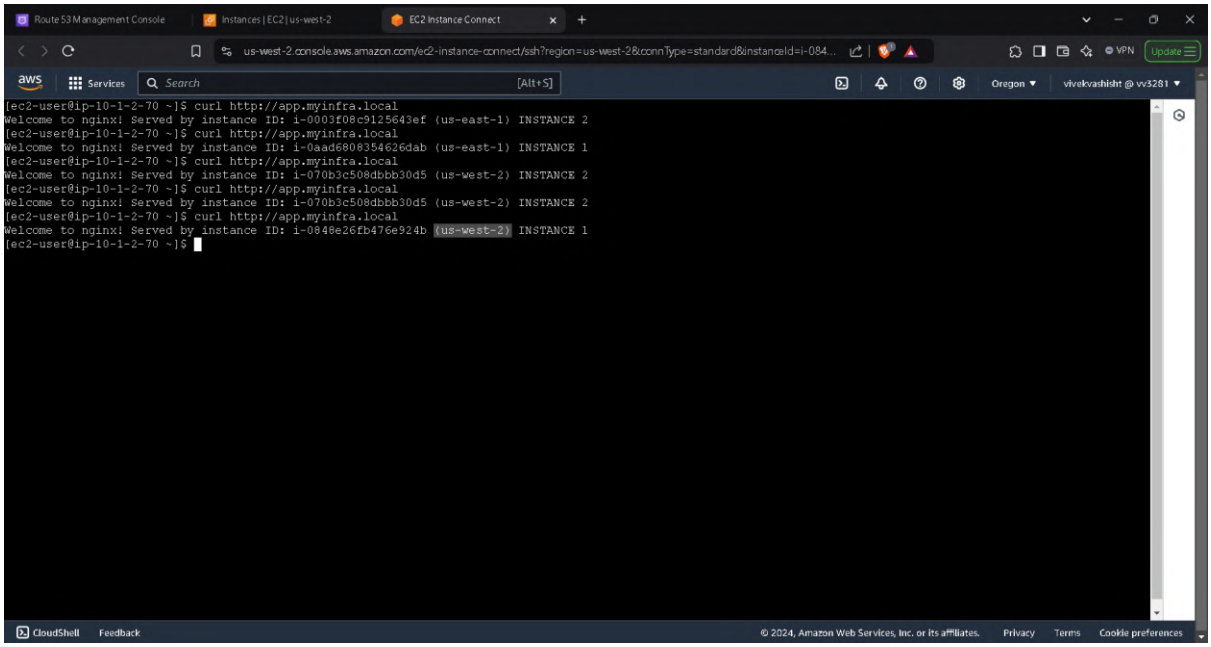

Auto Scaling group: terraform-20241031095010218800000003
Details
Activity
Automatic scaling
Instance management
Monitoring
Instance refresh
Instances (2/2)
Actions
Detach
Set to Standby
Set to InService
Instance scale-in protection
Set scale-in protection
Filter instances
Instance ID
Lifecycle
Instanc...
Weight...
Launch...
Availab...
i-0003f08c9125643ef
InService
t2.micro
app-launch-tem
us-east-1b
i-0aad6808354626dab
InService
t2.micro
app-launch-tem
us-east-1a
Healthy

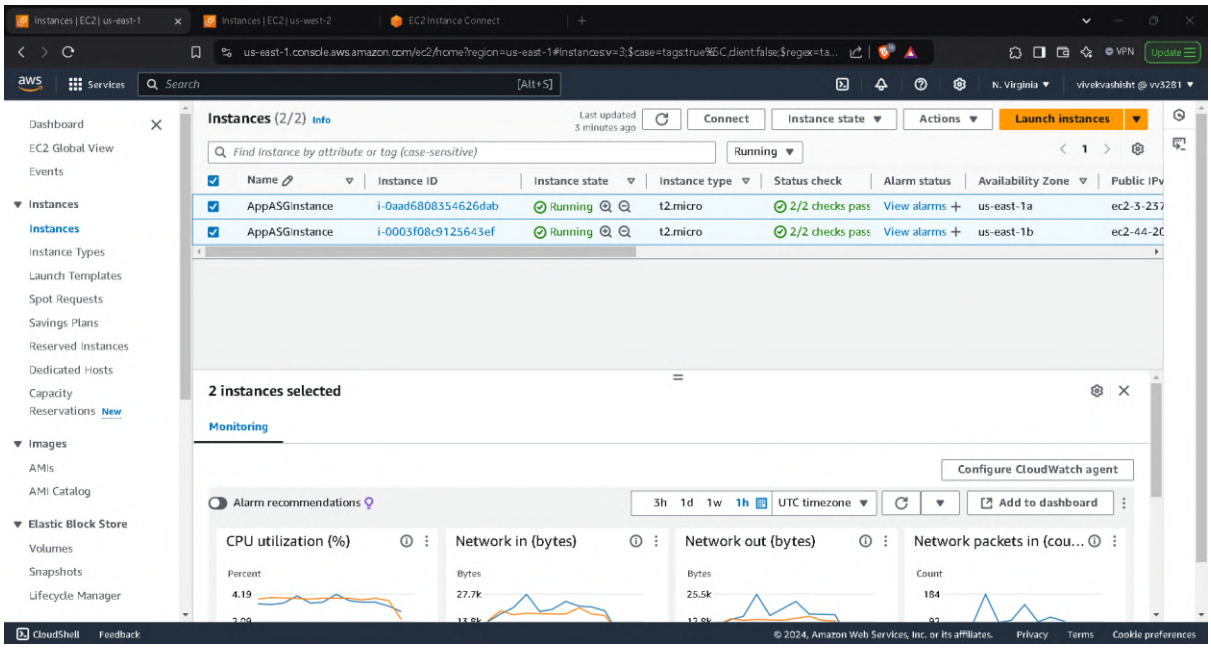

Group size
Specify the size of the Auto Scaling group by changing the desired capacity. You can also specify minimum and maximum scaling limits.
Desired capacity type
Choose the unit of measurement for the desired capacity value. vCPUs and Memory(GiB) are only supported for mixed instances groups configured with a set of instance attributes.
Units (number of instances)
Desired capacity
Specify your group size.
0
Scaling limits
Set limits on how much your desired capacity can be increased or decreased.
Min desired capacity
0
Equal or less than desired capacity
Max desired capacity
5
Equal or greater than desired capacity
Cancel
Update

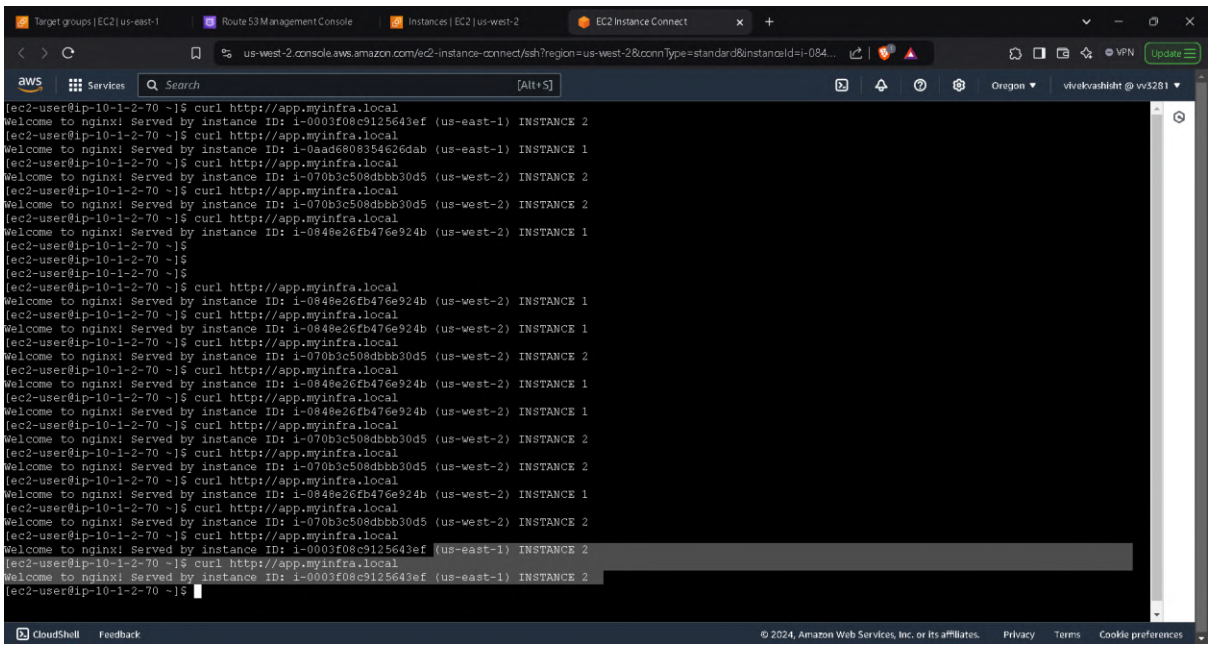

Stop instances
Stopping your instance allows you to reduce costs, modify settings, and troubleshoot problems.
Instance ID
Stop protection
i-0aad6808354626dab (AppASGinstance)
Off (Can stop instance)
i-0003f08c9125643ef (AppASGinstance)
Off (Can stop instance)
You will be billed for associated resources
After you stop the instance, you are no longer charged usage or data transfer fees for it. However, you will still be billed for associated Elastic IP addresses and EBS volumes.
Associated resources
You will continue to incur charges for these resources while the instance is stopped
Cancel
Stop


Instances (2)
Select an instance


Try out the new health checks console experience
Create health check
Delete health check
Edit health check
Name
Status
Description
Alarms
ID
Healthy
Unhealthy
Info
Monitoring
Alarms
Tags
Health checkers
Latency
No health check selected.
No health check selected.


Route 53 Management Console
Instances | EC2 | us-west-2
EC2 Instance Connect
us-west-2.console.aws.amazon.com/ec2-instance-connect/ssh?region=us-west-2&connType=standard&instanceId=i-084...
Services
Search
[Alt+S]
Oregon
vivekvashisht @ vv3281
[ec2-user@ip-10-1-2-70 ~]$ curl http://app.myinfra.local
Welcome to nginx! Served by instance ID: i-0003f08c9125643ef (us-east-1) INSTANCE 2
[ec2-user@ip-10-1-2-70 ~]$ curl http://app.myinfra.local
Welcome to nginx! Served by instance ID: i-0aad6808354626dab (us-east-1) INSTANCE 1
[ec2-user@ip-10-1-2-70 ~]$ curl http://app.myinfra.local
Welcome to nginx! Served by instance ID: i-070b3c508dbbb30d5 (us-west-2) INSTANCE 2
[ec2-user@ip-10-1-2-70 ~]$ curl http://app.myinfra.local
Welcome to nginx! Served by instance ID: i-070b3c508dbbb30d5 (us-west-2) INSTANCE 2
[ec2-user@ip-10-1-2-70 ~]$ curl http://app.myinfra.local
Welcome to nginx! Served by instance ID: i-0848e26fb476e924b (us-west-2) INSTANCE 1
[ec2-user@ip-10-1-2-70 ~]$
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Instances | EC2 | us-east-1
Instances | EC2 | us-west-2
EC2 Instance Connect
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#Instances:v=3;$case=tags:true%5C,client:false;$regex=ta...
N. Virginia
vivekvashisht @ vv3281
Dashboard
EC2 Global View
Events
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations New
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Instances (2/2) Info
Last updated 3 minutes ago
Connect
Instance state
Actions
Launch instances
Find instance by attribute or tag (case-sensitive)
Running
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
Availability Zone
Public IPv
AppASGinstance
i-0aad6808354626dab
Running
t2.micro
2/2 checks pass
View alarms
us-east-1a
AppASGinstance
i-0003f08c9125643ef
Running
t2.micro
2/2 checks pass
View alarms
us-east-1b
2 instances selected
Monitoring
Configure CloudWatch agent
Alarm recommendations
3h 1d 1w 1h UTC timezone
Add to dashboard
CPU utilization (%)
Network in (bytes)
Network out (bytes)
Network packets in (cou...
Percent
4.19
Bytes
27.7k
Bytes
25.5k
Count
184
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Target groups | EC2 | us-east-1
Route 53 Management Console
Instances | EC2 | us-west-2
EC2 Instance Connect
us-west-2.console.aws.amazon.com/ec2-instance-connect/ssh?region=us-west-2&connType=standard&instanceId=i-084...
Services
Search
[Alt+S]
Oregon
vivekvashisht @ vv3281
[ec2-user@ip-10-1-2-70 ~]$ curl http://app.myinfra.local
Welcome to nginx! Served by instance ID: i-0003f08c9125643ef (us-east-1) INSTANCE 2
[ec2-user@ip-10-1-2-70 ~]$ curl http://app.myinfra.local
Welcome to nginx! Served by instance ID: i-0aad6808354626dab (us-east-1) INSTANCE 1
[ec2-user@ip-10-1-2-70 ~]$ curl http://app.myinfra.local
Welcome to nginx! Served by instance ID: i-070b3c508dbbb30d5 (us-west-2) INSTANCE 2
[ec2-user@ip-10-1-2-70 ~]$ curl http://app.myinfra.local
Welcome to nginx! Served by instance ID: i-070b3c508dbbb30d5 (us-west-2) INSTANCE 2
[ec2-user@ip-10-1-2-70 ~]$ curl http://app.myinfra.local
Welcome to nginx! Served by instance ID: i-0848e26fb476e924b (us-west-2) INSTANCE 1
[ec2-user@ip-10-1-2-70 ~]$
[ec2-user@ip-10-1-2-70 ~]$
[ec2-user@ip-10-1-2-70 ~]$
[ec2-user@ip-10-1-2-70 ~]$ curl http://app.myinfra.local
Welcome to nginx! Served by instance ID: i-0848e26fb476e924b (us-west-2) INSTANCE 1
[ec2-user@ip-10-1-2-70 ~]$ curl http://app.myinfra.local
Welcome to nginx! Served by instance ID: i-0848e26fb476e924b (us-west-2) INSTANCE 1
[ec2-user@ip-10-1-2-70 ~]$ curl http://app.myinfra.local
Welcome to nginx! Served by instance ID: i-070b3c508dbbb30d5 (us-west-2) INSTANCE 2
[ec2-user@ip-10-1-2-70 ~]$ curl http://app.myinfra.local
Welcome to nginx! Served by instance ID: i-0848e26fb476e924b (us-west-2) INSTANCE 1
[ec2-user@ip-10-1-2-70 ~]$ curl http://app.myinfra.local
Welcome to nginx! Served by instance ID: i-0848e26fb476e924b (us-west-2) INSTANCE 1
[ec2-user@ip-10-1-2-70 ~]$ curl http://app.myinfra.local
Welcome to nginx! Served by instance ID: i-070b3c508dbbb30d5 (us-west-2) INSTANCE 2
[ec2-user@ip-10-1-2-70 ~]$ curl http://app.myinfra.local
Welcome to nginx! Served by instance ID: i-070b3c508dbbb30d5 (us-west-2) INSTANCE 2
[ec2-user@ip-10-1-2-70 ~]$ curl http://app.myinfra.local
Welcome to nginx! Served by instance ID: i-0848e26fb476e924b (us-west-2) INSTANCE 1
[ec2-user@ip-10-1-2-70 ~]$ curl http://app.myinfra.local
Welcome to nginx! Served by instance ID: i-070b3c508dbbb30d5 (us-west-2) INSTANCE 2
[ec2-user@ip-10-1-2-70 ~]$ curl http://app.myinfra.local
Welcome to nginx! Served by instance ID: i-0003f08c9125643ef (us-east-1) INSTANCE 2
[ec2-user@ip-10-1-2-70 ~]$ curl http://app.myinfra.local
Welcome to nginx! Served by instance ID: i-0003f08c9125643ef (us-east-1) INSTANCE 2
[ec2-user@ip-10-1-2-70 ~]$
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

3) Auto Scaling Group (ASG) Validation

a) Overview

In this phase, the focus was on testing and validating the Auto Scaling Group (ASG) configurations by simulating conditions to trigger scaling actions. To achieve this, I first modified specific CloudWatch alarms in the `cloudwatch_alarms.tf` file to define scale-up and scale-down conditions for CPU utilization. This testing phase ensured that the ASG could automatically increase or decrease the number of instances based on the load, simulating real-world scenarios.

b) CloudWatch Alarm Adjustments

To set up alarms that would trigger scaling actions, I first removed the existing general CPU utilization alarms from `cloudwatch_alarms.tf`, which included:

```
# Removed - CPU Utilization Alarm in us-east-1
resource "aws_cloudwatch_metric_alarm" "high_cpu_utilization" {
  alarm_name          = "HighCPUUtilizationAlarm"
  comparison_operator = "GreaterThanThreshold"
  evaluation_periods  = 2
  metric_name         = "CPUUtilization"
  namespace           = "AWS/EC2"
  period              = 300
  statistic           = "Average"
  threshold           = 80  # Threshold for alarm
  alarm_description   = "Triggers if CPU utilization exceeds 80%."
  dimensions = {
    AutoScalingGroupName = aws_autoscaling_group.app_asg.name
  }
  actions_enabled = true
  alarm_actions   = [aws_sns_topic.alert_topic.arn]
}

# Removed - CPU Utilization Alarm in us-west-2
resource "aws_cloudwatch_metric_alarm" "west_high_cpu_utilization" {
  provider              = aws.west
  alarm_name            = "WestHighCPUUtilizationAlarm"
  comparison_operator   = "GreaterThanThreshold"
  evaluation_periods    = 2
  metric_name           = "CPUUtilization"
  namespace             = "AWS/EC2"
  period                = 300
  statistic             = "Average"
  threshold             = 80
  alarm_description     = "Triggers if CPU utilization exceeds 80%."
  dimensions = {
    AutoScalingGroupName = aws_autoscaling_group.secondary_asg.name
  }
  actions_enabled = true
  alarm_actions   = [aws_sns_topic.west_alert_topic.arn]
}
```

c) New Alarms for Scale-Up and Scale-Down Triggers

After removing the original alarms, I added new CPU utilization alarms to enable automatic scaling. These alarms trigger based on predefined CPU thresholds.

For us-east-1 Region (Scale Up and Scale Down):

```
# Scale-Up Alarm in us-east-1
resource "aws_cloudwatch_metric_alarm" "scale_up_cpu_utilization" {
  alarm_name          = "ScaleUpCPUUtilizationAlarm"
  comparison_operator = "GreaterThanThreshold"
  evaluation_periods  = 2
  metric_name         = "CPUUtilization"
  namespace           = "AWS/EC2"
  period              = 300
  statistic           = "Average"
  threshold           = 80
  alarm_description   = "Triggers if CPU utilization exceeds 80% to scale
up."
  dimensions = {
    AutoScalingGroupName = aws_autoscaling_group.app_asg.name
  }
  actions_enabled = true
  alarm_actions   = [aws_autoscaling_policy.scale_up.arn,
aws_sns_topic.alert_topic.arn]
}

# Scale-Down Alarm in us-east-1
resource "aws_cloudwatch_metric_alarm" "scale_down_cpu_utilization" {
  alarm_name          = "ScaleDownCPUUtilizationAlarm"
  comparison_operator = "LessThanThreshold"
  evaluation_periods  = 2
  metric_name         = "CPUUtilization"
  namespace           = "AWS/EC2"
  period              = 300
  statistic           = "Average"
  threshold           = 30
  alarm_description   = "Triggers if CPU utilization drops below 30% to
scale down."
  dimensions = {
    AutoScalingGroupName = aws_autoscaling_group.app_asg.name
  }
  actions_enabled = true
  alarm_actions   = [aws_autoscaling_policy.scale_down.arn,
aws_sns_topic.alert_topic.arn]
}
```

For us-west-2 Region (Scale Up and Scale Down):

```
# Scale-Up Alarm in us-west-2
resource "aws_cloudwatch_metric_alarm" "west_scale_up_cpu_utilization" {
  provider              = aws.west
  alarm_name            = "WestScaleUpCPUUtilizationAlarm"
  comparison_operator   = "GreaterThanThreshold"
  evaluation_periods    = 2
  metric_name           = "CPUUtilization"
  namespace             = "AWS/EC2"
  period                = 300
  statistic             = "Average"
```

```
  threshold             = 80
  alarm_description     = "Triggers if CPU utilization exceeds 80% to
scale up."
  dimensions = {
    AutoScalingGroupName = aws_autoscaling_group.secondary_asg.name
  }
  actions_enabled = true
  alarm_actions   = [aws_autoscaling_policy.west_scale_up.arn,
aws_sns_topic.west_alert_topic.arn]
}

# Scale-Down Alarm in us-west-2
resource "aws_cloudwatch_metric_alarm" "west_scale_down_cpu_utilization" {
  provider              = aws.west
  alarm_name            = "WestScaleDownCPUUtilizationAlarm"
  comparison_operator   = "LessThanThreshold"
  evaluation_periods    = 2
  metric_name           = "CPUUtilization"
  namespace             = "AWS/EC2"
  period                = 300
  statistic             = "Average"
  threshold             = 30
  alarm_description     = "Triggers if CPU utilization drops below 30% to
scale down."
  dimensions = {
    AutoScalingGroupName = aws_autoscaling_group.secondary_asg.name
  }
  actions_enabled = true
  alarm_actions   = [aws_autoscaling_policy.west_scale_down.arn,
aws_sns_topic.west_alert_topic.arn]
}
```

d) Terraform Commands and Resource Verification

After modifying `cloudwatch_alarms.tf`, I ran the following commands:

```
terraform init
terraform plan
terraform apply -auto-approve
```

The apply command successfully executed, creating 4 new resources and removing 2 resources (the initial CPU utilization alarms).

e) Auto Scaling Group Update

In `ec2_autoscaling.tf`, I updated the `desired_capacity` and `min_size` of the ASG to 1 to align with testing requirements.

f) Verification in AWS Console

I verified the creation of the new CloudWatch alarms under the “Alarms” section of CloudWatch in both the us-east-1 and us-west-2 regions. The scale-up and scale-down alarms were visible as expected.

g) Testing the Scale-Up Alarm

To simulate high CPU usage and test the scale-up alarm, I connected to an EC2 instance and installed the Apache HTTP server benchmarking tool:

```
sudo yum install -y httpd-tools
```

Then, I initiated a load test:

```
ab -n 5000000 -c 500 http://primary-alb-1361659550.us-east-1.elb.amazonaws.com/
```

As the CPU utilization rose above 80%, the ScaleUpCPUUtilizationAlarm triggered, and the ASG increased its capacity from 1 to 2. This was confirmed by reviewing the ASG event logs:

```
Status: Success
Description: Launching a new EC2 instance: i-058967e362272ea8. Triggered by ScaleUpCPUUtilizationAlarm in state ALARM.
```

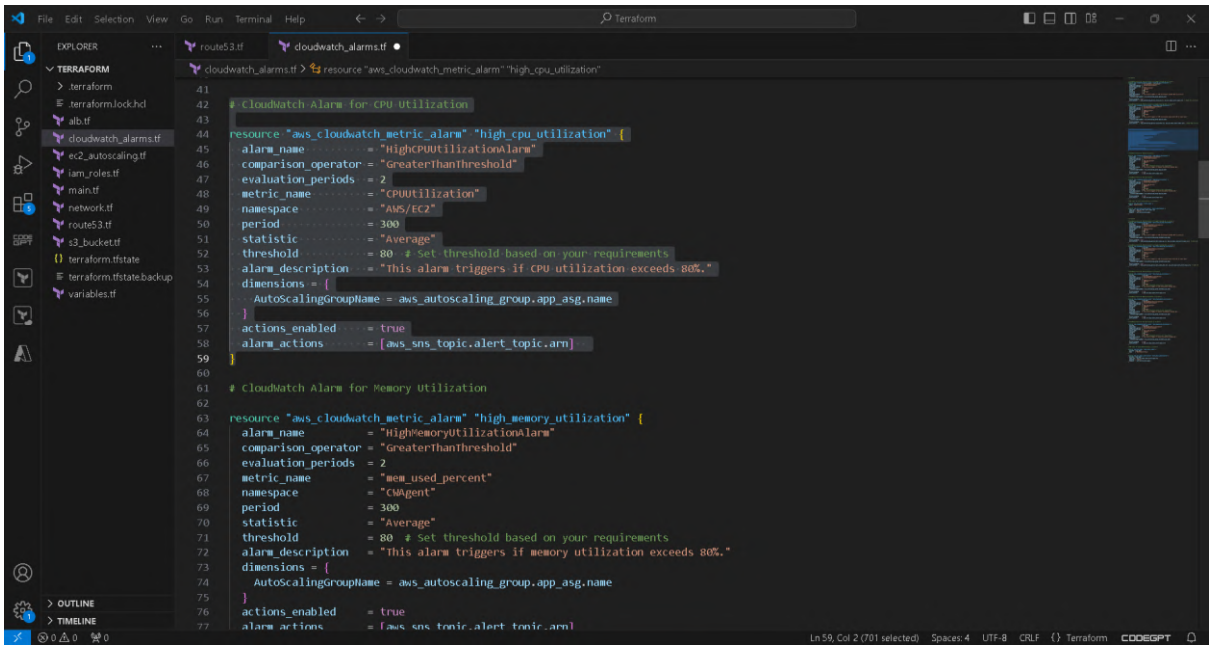
h) Testing the Scale-Down Alarm

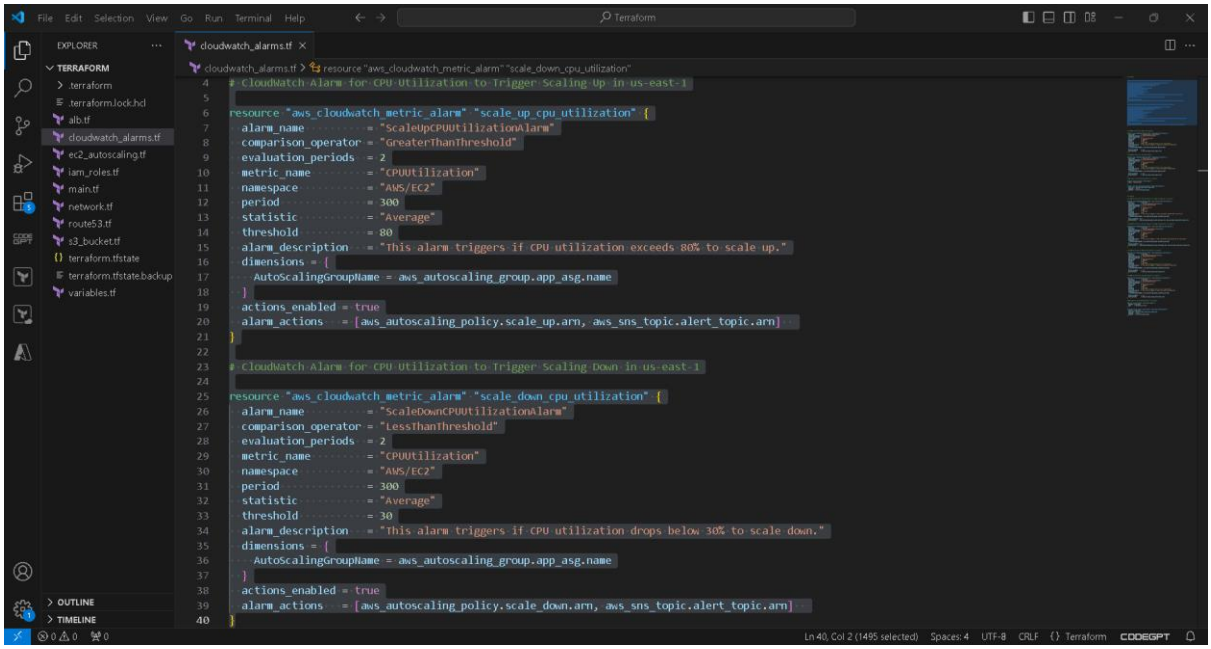
To test the scale-down alarm, I stopped the load test by pressing `Ctrl + C`. As CPU utilization dropped below 30%, the ScaleDownCPUUtilizationAlarm triggered, reducing the ASG capacity from 2 back to 1:

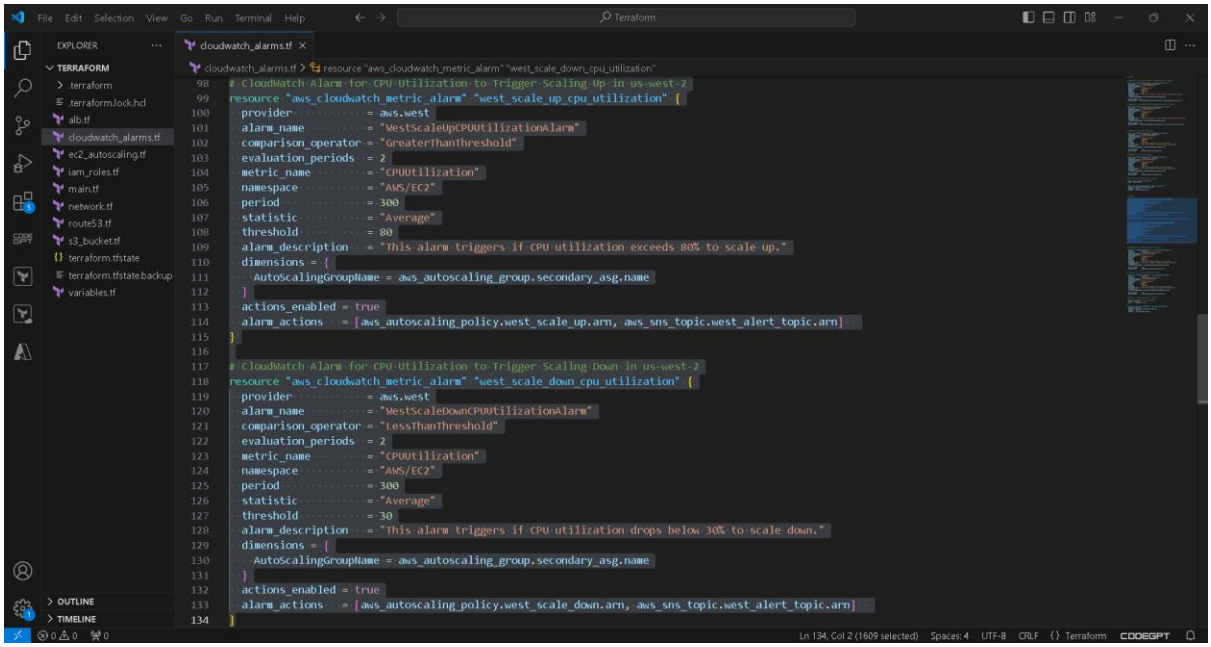
```
Status: Success
Description: Terminating EC2 instance: i-058967362272ea8 due to ScaleDownCPUUtilizationAlarm.
```

This phase confirmed that the ASG dynamically adjusts based on load conditions, successfully scaling up and down according to the set CloudWatch alarms.

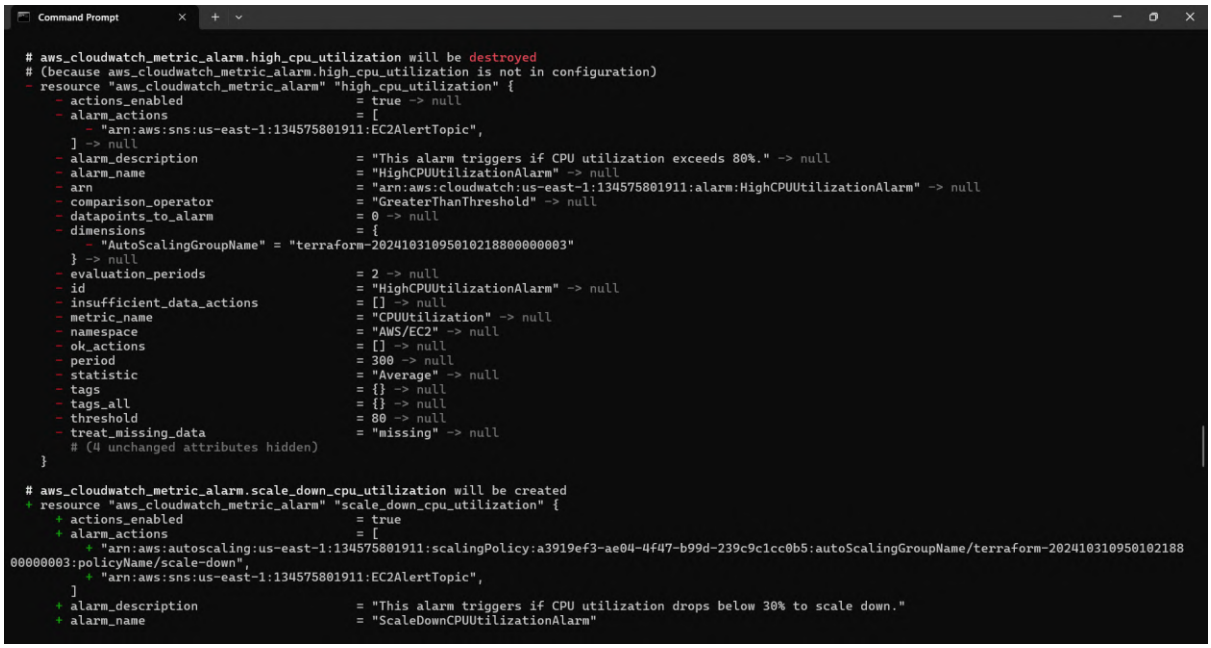
Screenshots

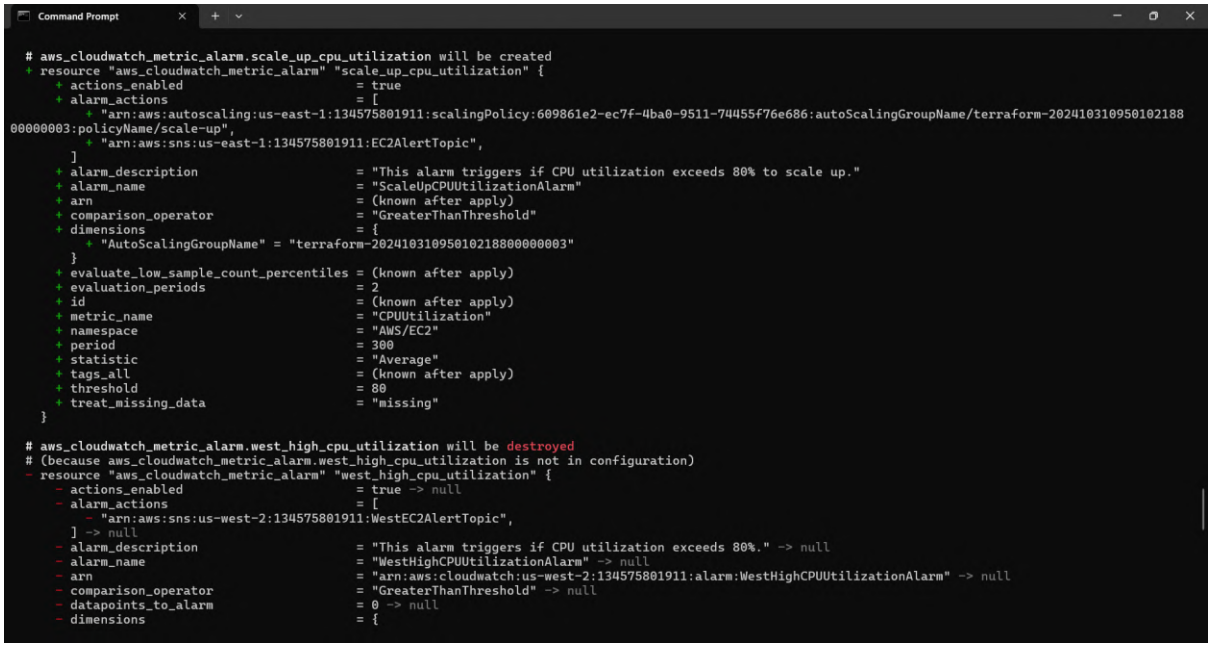


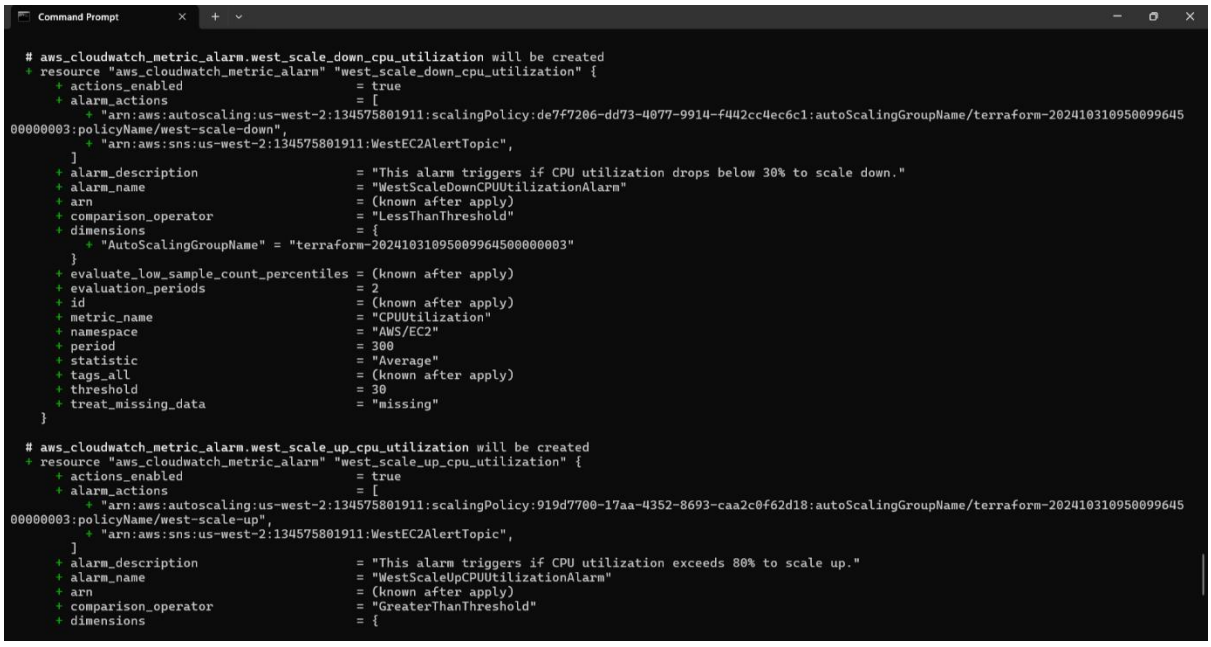


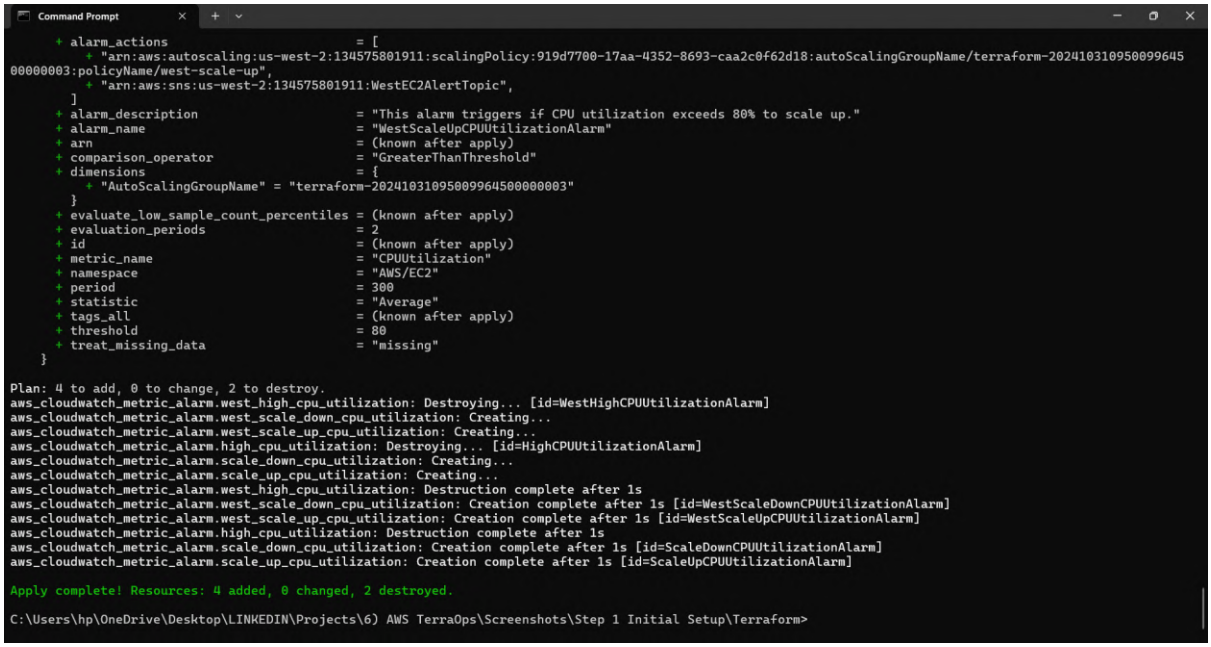



```
Command Prompt

C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.72.1

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.

C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>terraform plan
aws_vpc.west_vpc: Refreshing state... [id=vpc-0bc8cffa4878b083b]
aws_sns_topic.west_alert_topic: Refreshing state... [id=arn:aws:sns:us-west-2:134575801911:WestEC2AlertTopic]
aws_cloudwatch_metric_alarm.west_high_cpu_utilization: Refreshing state... [id=WestHighCPUUtilizationAlarm]
aws_sns_topic.alert_topic: Refreshing state... [id=arn:aws:sns:us-east-1:134575801911:EC2AlertTopic]
aws_iam_role.ec2_role: Refreshing state... [id=ec2-role]
aws_vpc.main_vpc: Refreshing state... [id=vpc-0ea415b90040f753dc]
aws_s3_bucket.terraform_state: Refreshing state... [id=aws-tf-backend-vv-bucket]
aws_cloudwatch_metric_alarm.high_cpu_utilization: Refreshing state... [id=HighCPUUtilizationAlarm]
aws_s3_bucket.backup_bucket: Refreshing state... [id=terra-backup-vv-project-bucket]
aws_sns_topic_subscription.west_email_subscription: Refreshing state... [id=arn:aws:sns:us-west-2:134575801911:WestEC2AlertTopic:52f0ce9a-1e0a-4ce2-964d-20c
8633c413a]
aws_sns_topic_subscription.email_subscription: Refreshing state... [id=arn:aws:sns:us-east-1:134575801911:EC2AlertTopic:45bab4c9-67d8-4001-97e7-524b1a0e81c0
]
aws_iam_role_policy_attachment.elb_full_access: Refreshing state... [id=ec2-role-20241030041703813700000004]
aws_iam_instance_profile.ec2_instance_profile: Refreshing state... [id=ec2-instance-profile]
aws_iam_role_policy_attachment.ec2_s3_readonly: Refreshing state... [id=ec2-role-20241026083002429100000001]
aws_iam_role_policy.ec2_sns_publish_policy: Refreshing state... [id=ec2-role:ec2-sns-publish-policy]
aws_iam_role_policy_attachment.ec2_cloudwatch_agent: Refreshing state... [id=ec2-role-20241031075603479000000001]
aws_iam_role_policy_attachment.autoscaling_full_access: Refreshing state... [id=ec2-role-20241030041703774400000002]
aws_iam_role_policy_attachment.ec2_full_access: Refreshing state... [id=ec2-role-20241030041703779300000003]
aws_route_table.west_public_rt: Refreshing state... [id=rtb-0bf4cfa31ff877e69]
aws_subnet.west_private_subnet_1: Refreshing state... [id=subnet-0de0b39d268f143ac]
aws_internet_gateway.west_igw: Refreshing state... [id=igw-093c80565bfdb459c]
```

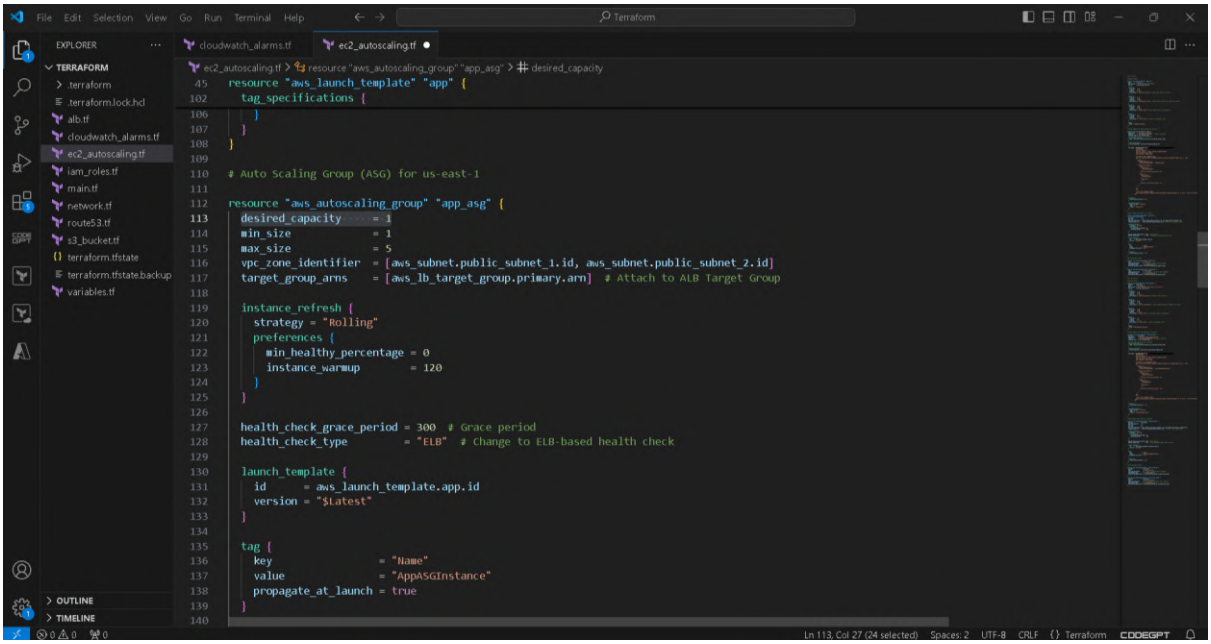

Command Prompt
# aws_cloudwatch_metric_alarm.high_cpu_utilization will be destroyed
# (because aws_cloudwatch_metric_alarm.high_cpu_utilization is not in configuration)
- resource "aws_cloudwatch_metric_alarm" "high_cpu_utilization" {
- actions_enabled = true -> null
- alarm_actions = [
- "arn:aws:sns:us-east-1:134575801911:EC2AlertTopic",
] -> null
- alarm_description = "This alarm triggers if CPU utilization exceeds 80%." -> null
- alarm_name = "HighCPUUtilizationAlarm" -> null
- arn = "arn:aws:cloudwatch:us-east-1:134575801911:alarm:HighCPUUtilizationAlarm" -> null
- comparison_operator = "GreaterThanThreshold" -> null
- datapoints_to_alarm = 0 -> null
- dimensions = {
- "AutoScalingGroupName" = "terraform-20241031095010218800000003"
} -> null
- evaluation_periods = 2 -> null
- id = "HighCPUUtilizationAlarm" -> null
- insufficient_data_actions = [] -> null
- metric_name = "CPUUtilization" -> null
- namespace = "AWS/EC2" -> null
- ok_actions = [] -> null
- period = 300 -> null
- statistic = "Average" -> null
- tags = {} -> null
- tags_all = {} -> null
- threshold = 80 -> null
- treat_missing_data = "missing" -> null
# (4 unchanged attributes hidden)
}
# aws_cloudwatch_metric_alarm.scale_down_cpu_utilization will be created
+ resource "aws_cloudwatch_metric_alarm" "scale_down_cpu_utilization" {
+ actions_enabled = true
+ alarm_actions = [
+ "arn:aws:autoscaling:us-east-1:134575801911:scalingPolicy:a3919ef3-ae04-4f47-b99d-239c9c1cc0b5:autoScalingGroupName/terraform-20241031095010218800000003:policyName/scale-down",
+ "arn:aws:sns:us-east-1:134575801911:EC2AlertTopic",
]
+ alarm_description = "This alarm triggers if CPU utilization drops below 30% to scale down."
+ alarm_name = "ScaleDownCPUUtilizationAlarm"


Command Prompt
# aws_cloudwatch_metric_alarm.scale_up_cpu_utilization will be created
+ resource "aws_cloudwatch_metric_alarm" "scale_up_cpu_utilization" {
+ actions_enabled = true
+ alarm_actions = [
+ "arn:aws:autoscaling:us-east-1:134575801911:scalingPolicy:609861e2-ec7f-4ba0-9511-74455f76e686:autoScalingGroupName/terraform-20241031095010218800000003:policyName/scale-up",
+ "arn:aws:sns:us-east-1:134575801911:EC2AlertTopic",
]
+ alarm_description = "This alarm triggers if CPU utilization exceeds 80% to scale up."
+ alarm_name = "ScaleUpCPUUtilizationAlarm"
+ arn = (known after apply)
+ comparison_operator = "GreaterThanThreshold"
+ dimensions = {
+ "AutoScalingGroupName" = "terraform-20241031095010218800000003"
}
+ evaluate_low_sample_count_percentiles = (known after apply)
+ evaluation_periods = 2
+ id = (known after apply)
+ metric_name = "CPUUtilization"
+ namespace = "AWS/EC2"
+ period = 300
+ statistic = "Average"
+ tags_all = (known after apply)
+ threshold = 80
+ treat_missing_data = "missing"
}
# aws_cloudwatch_metric_alarm.west_high_cpu_utilization will be destroyed
# (because aws_cloudwatch_metric_alarm.west_high_cpu_utilization is not in configuration)
- resource "aws_cloudwatch_metric_alarm" "west_high_cpu_utilization" {
- actions_enabled = true -> null
- alarm_actions = [
- "arn:aws:sns:us-west-2:134575801911:WestEC2AlertTopic",
] -> null
- alarm_description = "This alarm triggers if CPU utilization exceeds 80%." -> null
- alarm_name = "WestHighCPUUtilizationAlarm" -> null
- arn = "arn:aws:cloudwatch:us-west-2:134575801911:alarm:WestHighCPUUtilizationAlarm" -> null
- comparison_operator = "GreaterThanThreshold" -> null
- datapoints_to_alarm = 0 -> null
- dimensions = {

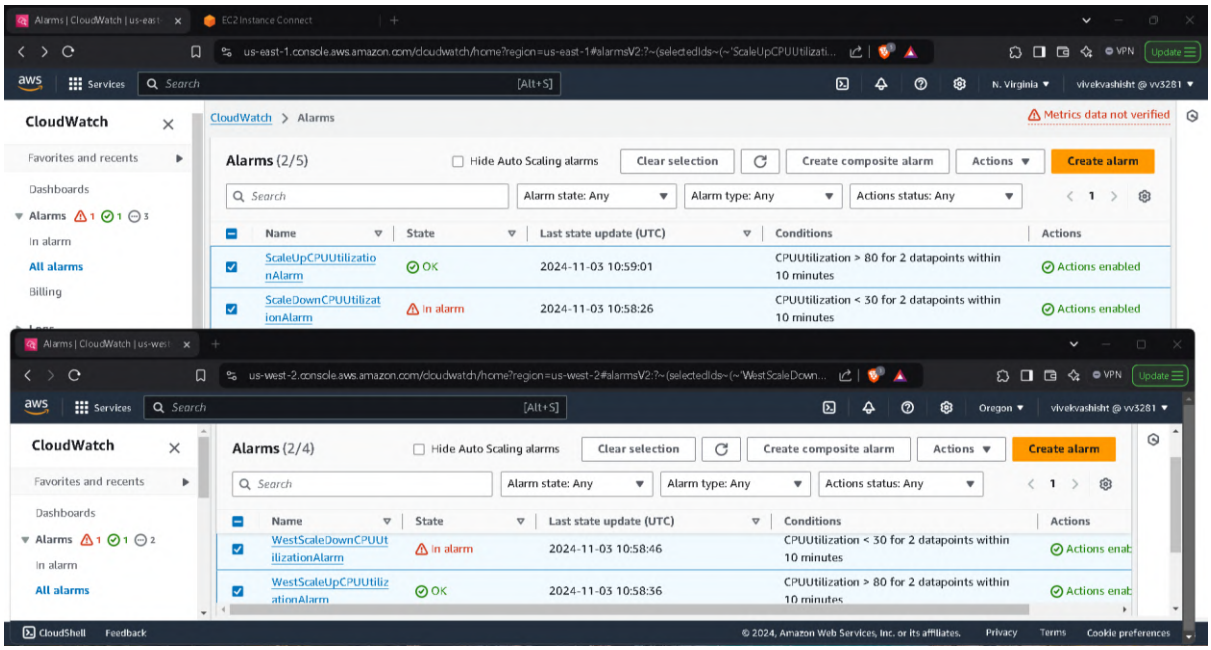

Command Prompt
# aws_cloudwatch_metric_alarm.west_scale_down_cpu_utilization will be created
+ resource "aws_cloudwatch_metric_alarm" "west_scale_down_cpu_utilization" {
+ actions_enabled = true
+ alarm_actions = [
+ "arn:aws:autoscaling:us-west-2:134575801911:scalingPolicy:de7f7206-dd73-4077-9914-f442cc4ec6c1:autoScalingGroupName/terraform-20241031095009964500000003:policyName/west-scale-down",
+ "arn:aws:sns:us-west-2:134575801911:WestEC2AlertTopic",
]
+ alarm_description = "This alarm triggers if CPU utilization drops below 30% to scale down."
+ alarm_name = "WestScaleDownCPUUtilizationAlarm"
+ arn = (known after apply)
+ comparison_operator = "LessThanThreshold"
+ dimensions = {
+ "AutoScalingGroupName" = "terraform-20241031095009964500000003"
}
+ evaluate_low_sample_count_percentiles = (known after apply)
+ evaluation_periods = 2
+ id = (known after apply)
+ metric_name = "CPUUtilization"
+ namespace = "AWS/EC2"
+ period = 300
+ statistic = "Average"
+ tags_all = (known after apply)
+ threshold = 30
+ treat_missing_data = "missing"
}
# aws_cloudwatch_metric_alarm.west_scale_up_cpu_utilization will be created
+ resource "aws_cloudwatch_metric_alarm" "west_scale_up_cpu_utilization" {
+ actions_enabled = true
+ alarm_actions = [
+ "arn:aws:autoscaling:us-west-2:134575801911:scalingPolicy:919d7700-17aa-4352-8693-caa2c0f62d18:autoScalingGroupName/terraform-20241031095009964500000003:policyName/west-scale-up",
+ "arn:aws:sns:us-west-2:134575801911:WestEC2AlertTopic",
]
+ alarm_description = "This alarm triggers if CPU utilization exceeds 80% to scale up."
+ alarm_name = "WestScaleUpCPUUtilizationAlarm"
+ arn = (known after apply)
+ comparison_operator = "GreaterThanThreshold"
+ dimensions = {

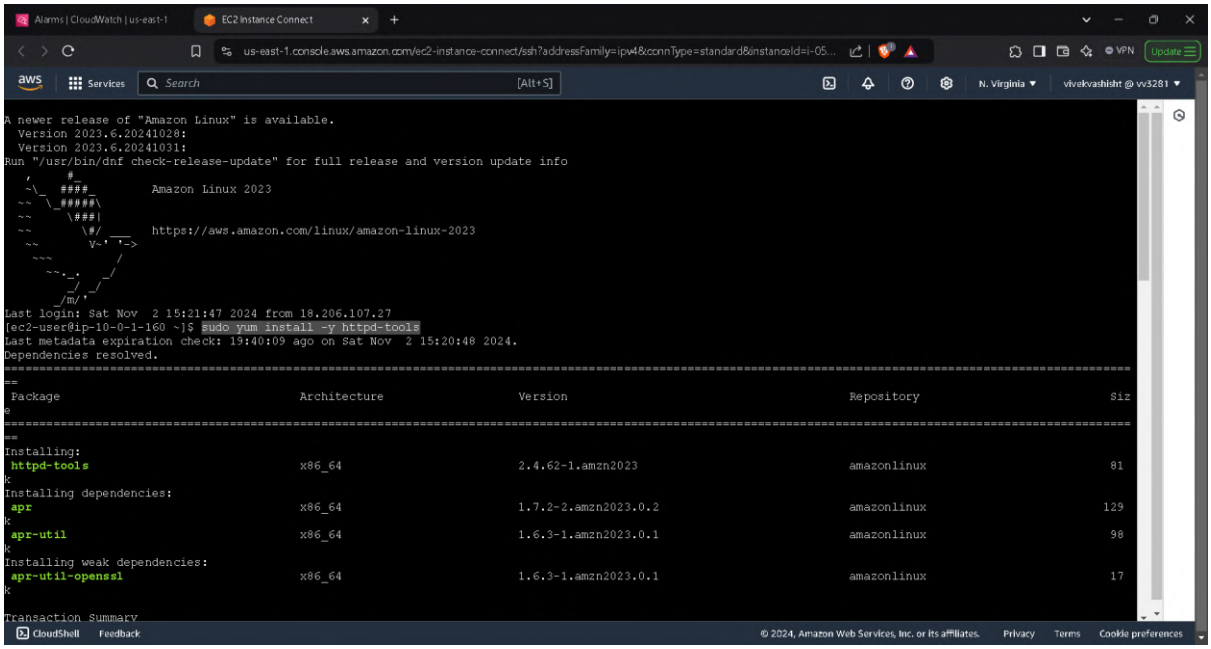


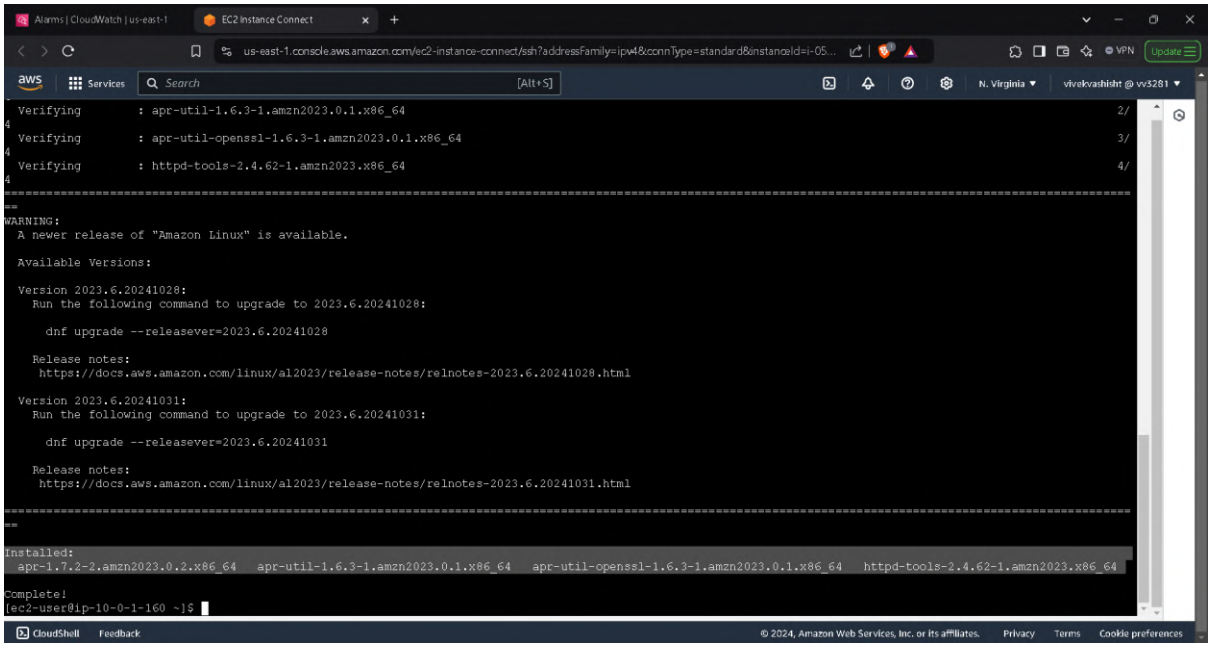
```
Command Prompt

      + alarm_actions                       = [
          + "arn:aws:autoscaling:us-west-2:134575801911:scalingPolicy:919d7700-17aa-4352-8693-caa2c0f62d18:autoScalingGroupName/terraform-20241031095009964500000003:policyName/west-scale-up",
          + "arn:aws:sns:us-west-2:134575801911:WestEC2AlertTopic",
        ]
      + alarm_description                   = "This alarm triggers if CPU utilization exceeds 80% to scale up."
      + alarm_name                          = "WestScaleUpCPUUtilizationAlarm"
      + arn                                 = (known after apply)
      + comparison_operator                 = "GreaterThanThreshold"
      + dimensions                          = {
          + "AutoScalingGroupName" = "terraform-20241031095009964500000003"
        }
      + evaluate_low_sample_count_percentiles = (known after apply)
      + evaluation_periods                  = 2
      + id                                  = (known after apply)
      + metric_name                         = "CPUUtilization"
      + namespace                           = "AWS/EC2"
      + period                              = 300
      + statistic                           = "Average"
      + tags_all                            = (known after apply)
      + threshold                           = 80
      + treat_missing_data                  = "missing"
    }

Plan: 4 to add, 0 to change, 2 to destroy.
aws_cloudwatch_metric_alarm.west_high_cpu_utilization: Destroying... [id=WestHighCPUUtilizationAlarm]
aws_cloudwatch_metric_alarm.west_scale_down_cpu_utilization: Creating...
aws_cloudwatch_metric_alarm.west_scale_up_cpu_utilization: Creating...
aws_cloudwatch_metric_alarm.high_cpu_utilization: Destroying... [id=HighCPUUtilizationAlarm]
aws_cloudwatch_metric_alarm.scale_down_cpu_utilization: Creating...
aws_cloudwatch_metric_alarm.scale_up_cpu_utilization: Creating...
aws_cloudwatch_metric_alarm.west_high_cpu_utilization: Destruction complete after 1s
aws_cloudwatch_metric_alarm.west_scale_down_cpu_utilization: Creation complete after 1s [id=WestScaleDownCPUUtilizationAlarm]
aws_cloudwatch_metric_alarm.west_scale_up_cpu_utilization: Creation complete after 1s [id=WestScaleUpCPUUtilizationAlarm]
aws_cloudwatch_metric_alarm.high_cpu_utilization: Destruction complete after 1s
aws_cloudwatch_metric_alarm.scale_down_cpu_utilization: Creation complete after 1s [id=ScaleDownCPUUtilizationAlarm]
aws_cloudwatch_metric_alarm.scale_up_cpu_utilization: Creation complete after 1s [id=ScaleUpCPUUtilizationAlarm]

Apply complete! Resources: 4 added, 0 changed, 2 destroyed.

C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>
```

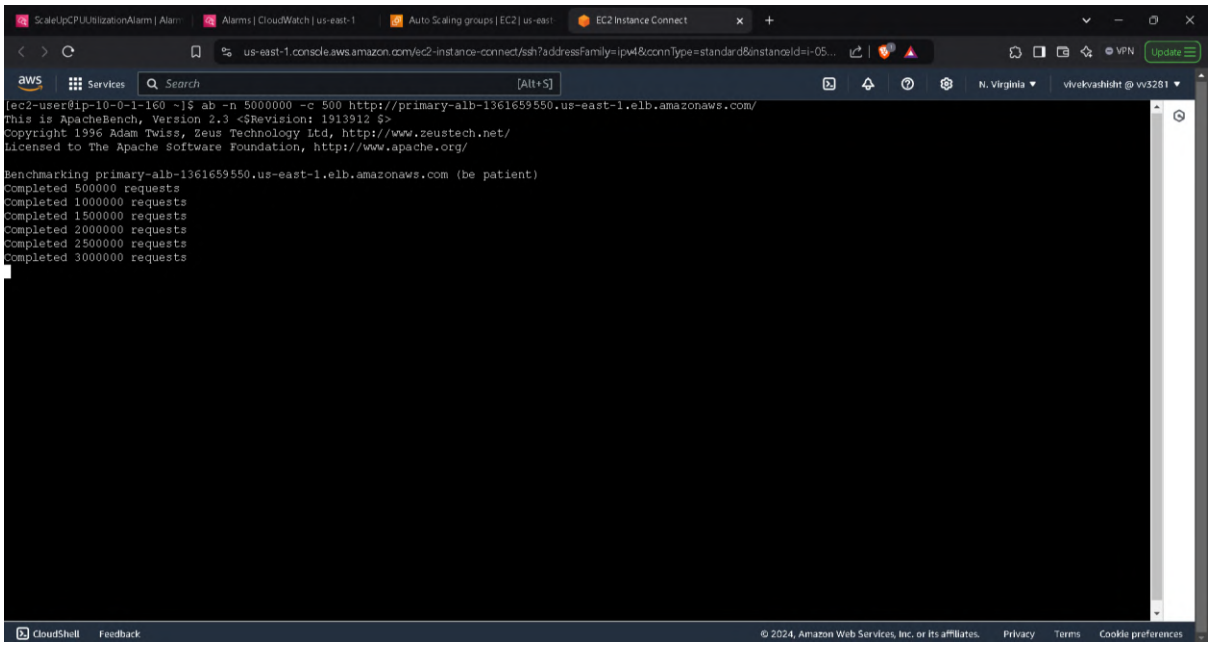


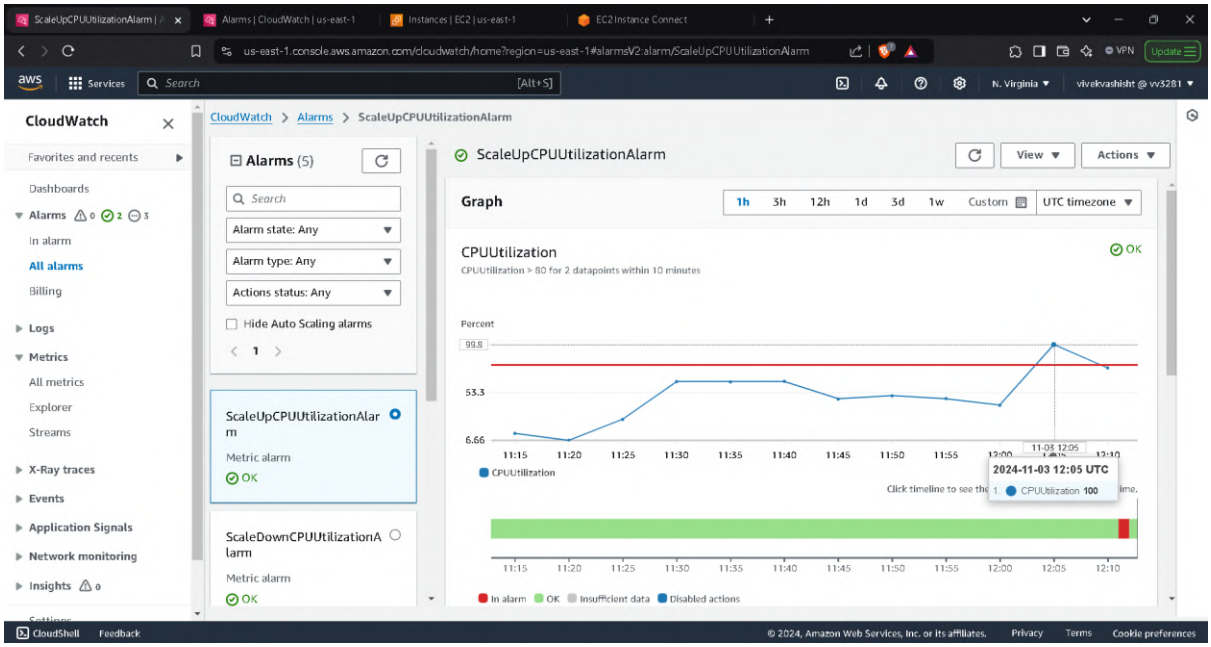
```
resource "aws_launch_template" "app" {
  tag_specifications {
  }
}

# Auto Scaling Group (ASG) for us-east-1

resource "aws_autoscaling_group" "app_asg" {
  desired_capacity    = 1
  min_size            = 1
  max_size            = 5
  vpc_zone_identifier = [aws_subnet.public_subnet_1.id, aws_subnet.public_subnet_2.id]
  target_group_arns   = [aws_lb_target_group.primary.arn]  # Attach to ALB Target Group

  instance_refresh {
    strategy = "Rolling"
    preferences {
      min_healthy_percentage = 0
      instance_warmup        = 120
    }
  }

  health_check_grace_period = 300  # Grace period
  health_check_type         = "ELB"  # Change to ELB-based health check

  launch_template {
    id      = aws_launch_template.app.id
    version = "$Latest"
  }

  tag {
    key                 = "Name"
    value               = "AppASGInstance"
    propagate_at_launch = true
  }
```

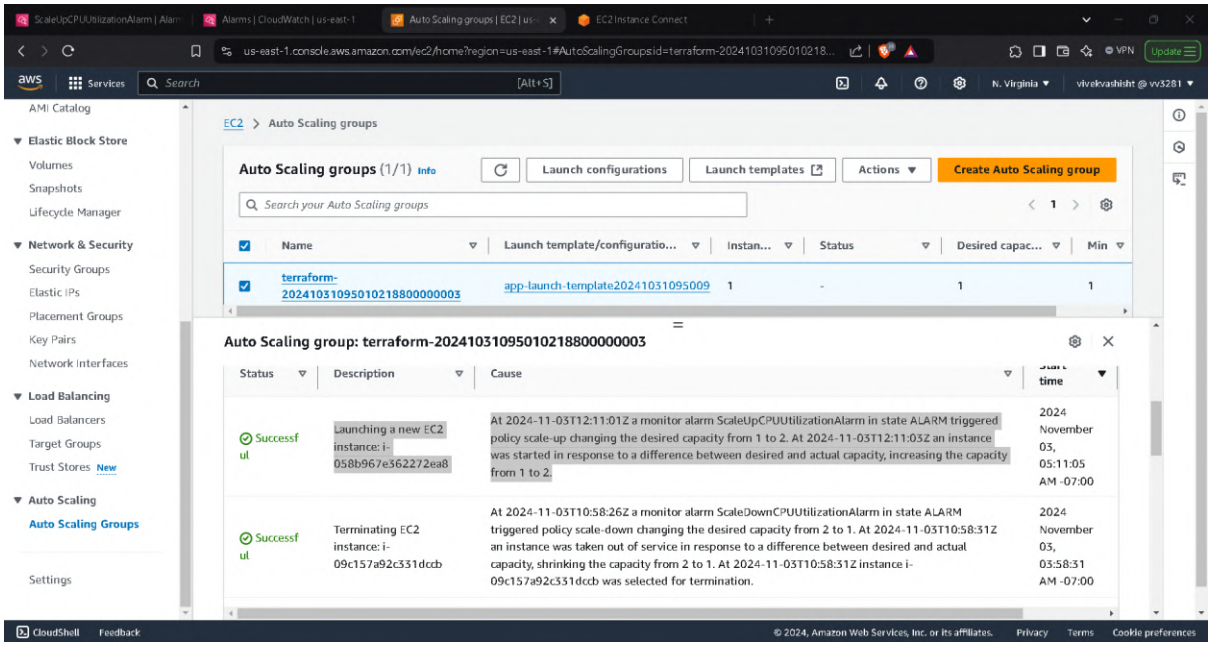


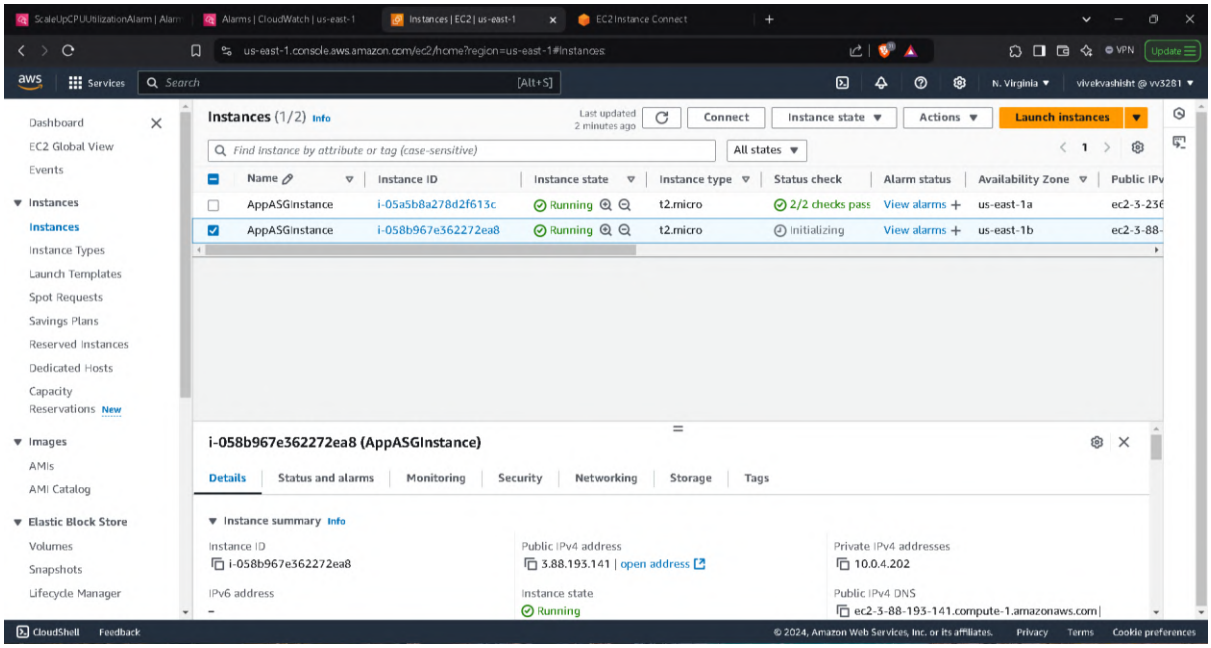

Alarms | CloudWatch | us-east-1
EC2 Instance Connect
us-east-1.console.aws.amazon.com/ec2-instance-connect/ssh?addressFamily=ipv4&connType=standard&instanceId=i-05...
Services
Search
[Alt+S]
N. Virginia
vivekvashisht @ vv3281
A newer release of "Amazon Linux" is available.
Version 2023.6.20241028:
Version 2023.6.20241031:
Run "/usr/bin/dnf check-release-update" for full release and version update info
Amazon Linux 2023
https://aws.amazon.com/linux/amazon-linux-2023
Last login: Sat Nov 2 15:21:47 2024 from 18.206.107.27
[ec2-user@ip-10-0-1-160 ~]$ sudo yum install -y httpd-tools
Last metadata expiration check: 19:40:09 ago on Sat Nov 2 15:20:48 2024.
Dependencies resolved.
Package Architecture Version Repository Siz
Installing:
httpd-tools x86_64 2.4.62-1.amzn2023 amazonlinux 81
Installing dependencies:
apr x86_64 1.7.2-2.amzn2023.0.2 amazonlinux 129
apr-util x86_64 1.6.3-1.amzn2023.0.1 amazonlinux 98
Installing weak dependencies:
apr-util-openssl x86_64 1.6.3-1.amzn2023.0.1 amazonlinux 17
Transaction Summary
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

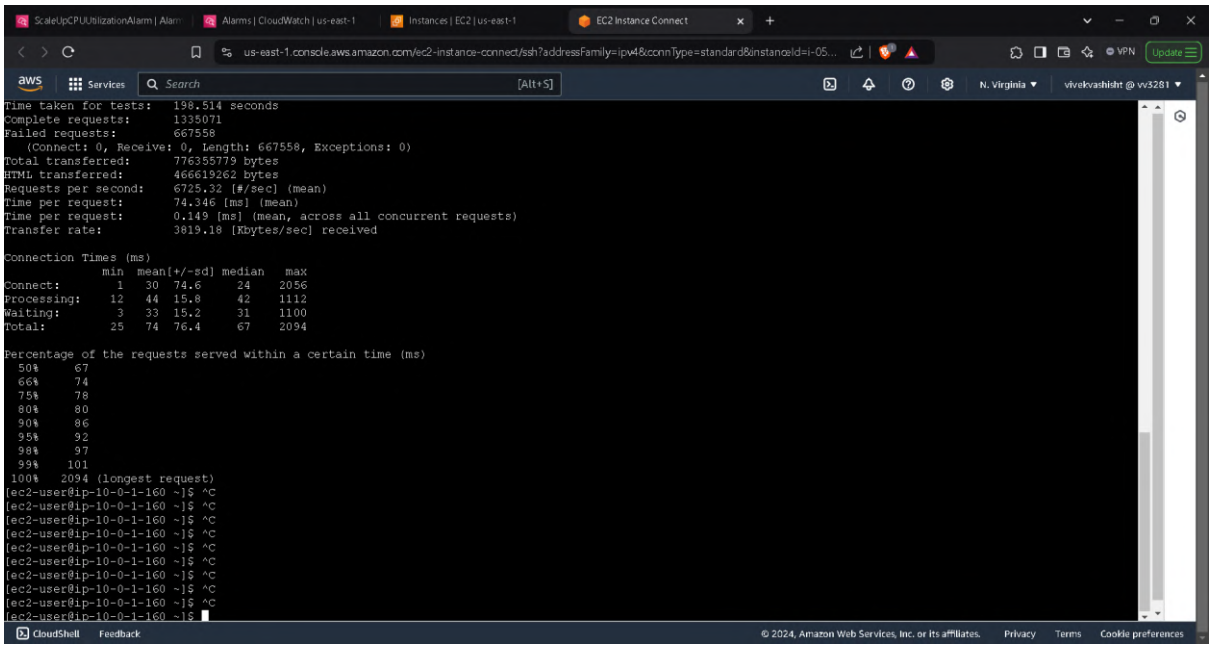

Verifying : apr-util-1.6.3-1.amzn2023.0.1.x86_64
Verifying : apr-util-openssl-1.6.3-1.amzn2023.0.1.x86_64
Verifying : httpd-tools-2.4.62-1.amzn2023.x86_64
WARNING:
A newer release of "Amazon Linux" is available.
Available Versions:
Version 2023.6.20241028:
Run the following command to upgrade to 2023.6.20241028:
dnf upgrade --releasever=2023.6.20241028
Release notes:
https://docs.aws.amazon.com/linux/al2023/release-notes/relnotes-2023.6.20241028.html
Version 2023.6.20241031:
Run the following command to upgrade to 2023.6.20241031:
dnf upgrade --releasever=2023.6.20241031
Release notes:
https://docs.aws.amazon.com/linux/al2023/release-notes/relnotes-2023.6.20241031.html
Installed:
apr-1.7.2-2.amzn2023.0.2.x86_64 apr-util-1.6.3-1.amzn2023.0.1.x86_64 apr-util-openssl-1.6.3-1.amzn2023.0.1.x86_64 httpd-tools-2.4.62-1.amzn2023.x86_64
Complete!
[ec2-user@ip-10-0-1-160 ~]$
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

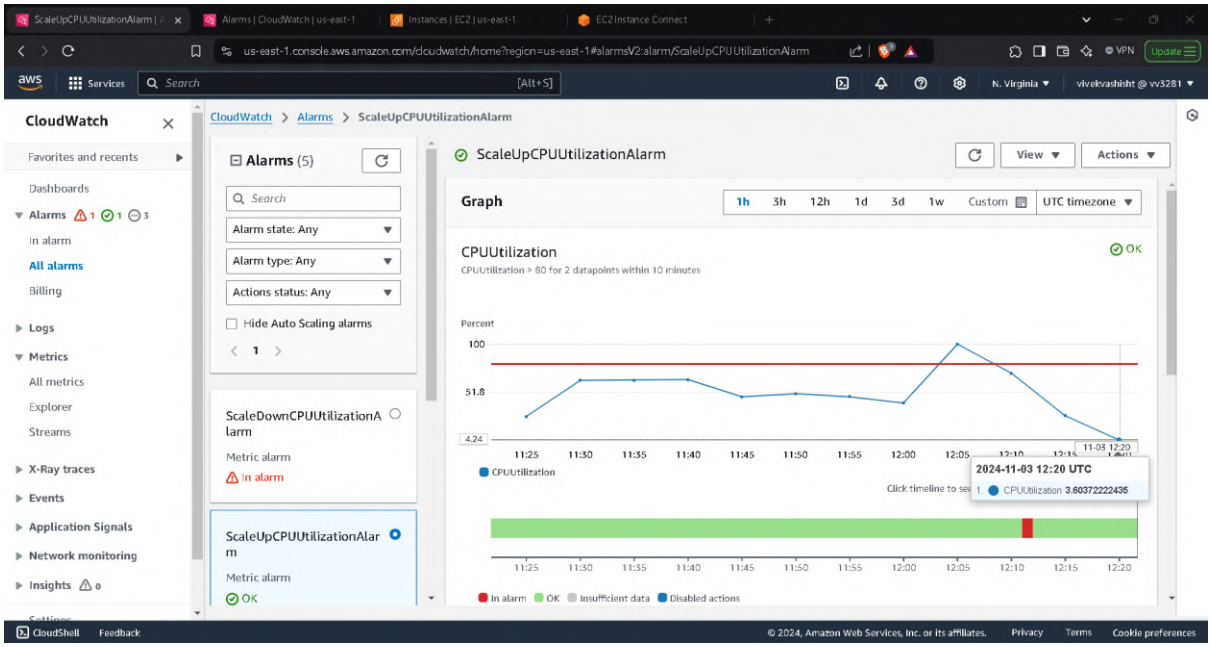

ScaleUpCPUUtilizationAlarm | Alarm
Alarms | CloudWatch | us-east-1
Auto Scaling groups | EC2 | us-east-1
EC2 Instance Connect
us-east-1.console.aws.amazon.com/ec2-instance-connect/ssh?addressFamily=ipv4&connType=standard&instanceId=i-05...
Services
Search
[Alt+S]
N. Virginia
vivekvashisht @ vv3281
[ec2-user@ip-10-0-1-160 ~]$ ab -n 5000000 -c 500 http://primary-alb-1361659550.us-east-1.elb.amazonaws.com/
This is ApacheBench, Version 2.3 <$Revision: 1913912 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/
Benchmarking primary-alb-1361659550.us-east-1.elb.amazonaws.com (be patient)
Completed 500000 requests
Completed 1000000 requests
Completed 1500000 requests
Completed 2000000 requests
Completed 2500000 requests
Completed 3000000 requests
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

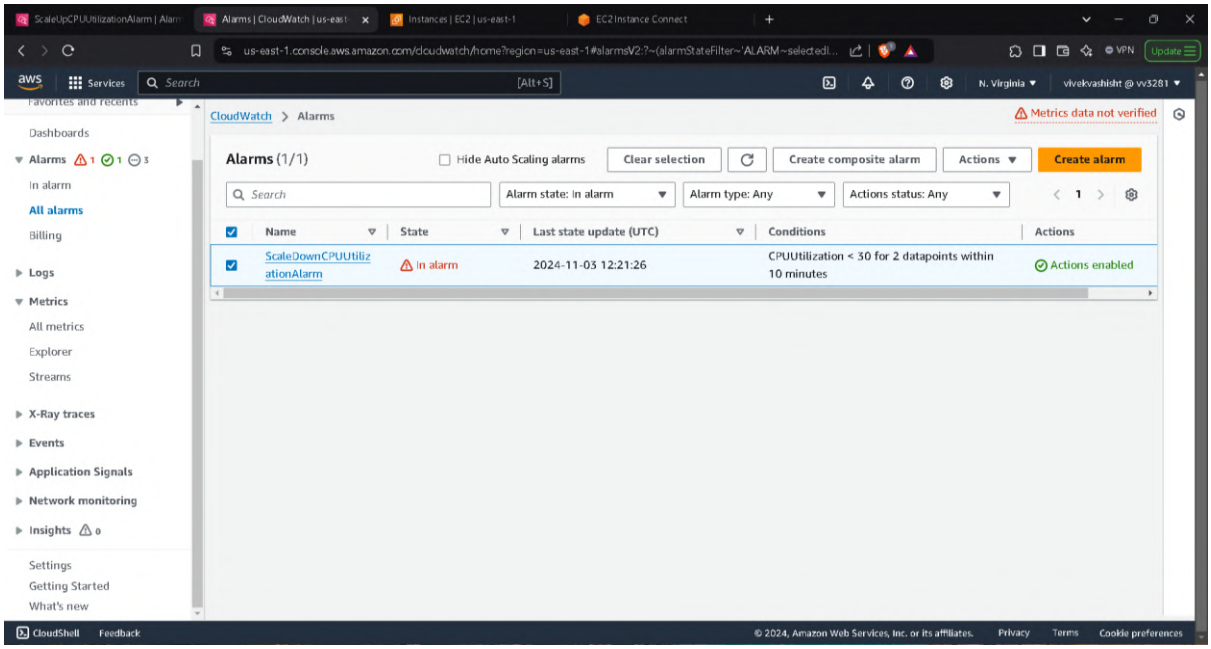

CloudWatch
Alarms (5)
ScaleUpCPUUtilizationAlarm
Metric alarm
OK
ScaleDownCPUUtilizationAlarm
Metric alarm
OK
ScaleUpCPUUtilizationAlarm
View
Actions
Graph
1h 3h 12h 1d 3d 1w Custom UTC timezone
CPUUtilization
CPUUtilization > 80 for 2 datapoints within 10 minutes
OK
Percent
99.8
53.3
6.66
11:15 11:20 11:25 11:30 11:35 11:40 11:45 11:50 11:55 12:00 12:05 12:10
2024-11-03 12:05 UTC
CPUUtilization 100
In alarm OK Insufficient data Disabled actions

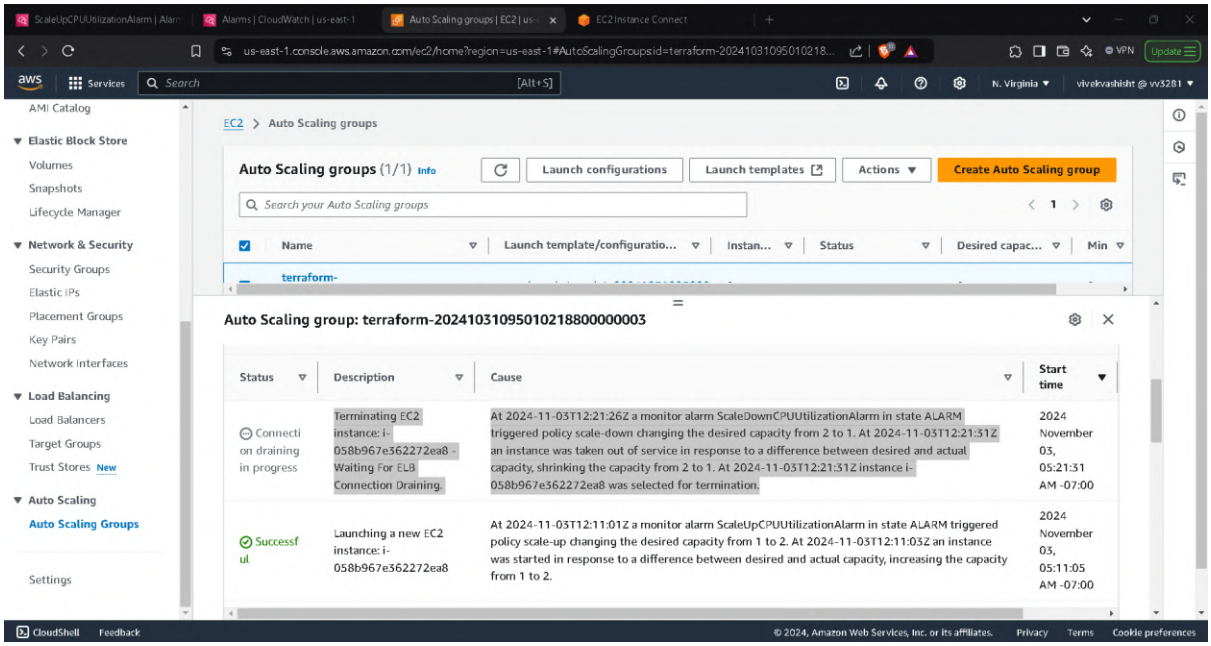

Auto Scaling groups (1/1) Info
Launch configurations
Launch templates
Actions
Create Auto Scaling group
Search your Auto Scaling groups
Name
Launch template/configuratio...
Instan...
Status
Desired capac...
Min
terraform-20241031095010218800000003
app-launch-template20241031095009
Auto Scaling group: terraform-20241031095010218800000003
Status
Description
Cause
Start time
Successful
Launching a new EC2 instance: i-058b967e362272ea8
At 2024-11-03T12:11:01Z a monitor alarm ScaleUpCPUUtilizationAlarm in state ALARM triggered policy scale-up changing the desired capacity from 1 to 2. At 2024-11-03T12:11:03Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 1 to 2.
2024 November 03, 05:11:05 AM -07:00
Successful
Terminating EC2 instance: i-09c157a92c331dccb
At 2024-11-03T10:58:26Z a monitor alarm ScaleDownCPUUtilizationAlarm in state ALARM triggered policy scale-down changing the desired capacity from 2 to 1. At 2024-11-03T10:58:31Z an instance was taken out of service in response to a difference between desired and actual capacity, shrinking the capacity from 2 to 1. At 2024-11-03T10:58:31Z instance i-09c157a92c331dccb was selected for termination.
2024 November 03, 03:58:31 AM -07:00

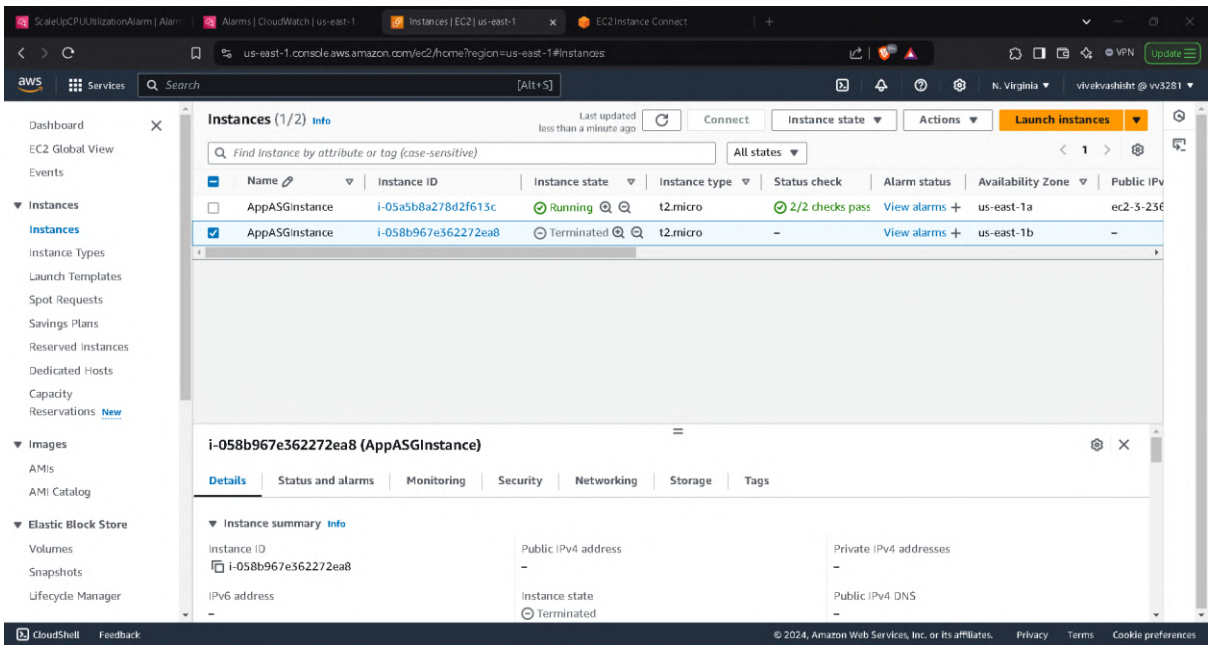

Instances (1/2) Info
Connect
Instance state
Actions
Launch instances
Find instance by attribute or tag (case-sensitive)
All states
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
Availability Zone
Public IPv
AppASGInstance
i-05a5b8a278d2f613c
Running
t2.micro
2/2 checks pass
View alarms
us-east-1a
AppASGInstance
i-058b967e362272ea8
Running
t2.micro
Initializing
View alarms
us-east-1b
i-058b967e362272ea8 (AppASGInstance)
Details
Status and alarms
Monitoring
Security
Networking
Storage
Tags
Instance summary Info
Instance ID
i-058b967e362272ea8
Public IPv4 address
3.88.193.141 | open address
Private IPv4 addresses
10.0.4.202
IPv6 address
Instance state
Running
Public IPv4 DNS
ec2-3-88-193-141.compute-1.amazonaws.com


Time taken for tests: 198.514 seconds
Complete requests: 1335071
Failed requests: 667558
(Connect: 0, Receive: 0, Length: 667558, Exceptions: 0)
Total transferred: 776355779 bytes
HTML transferred: 466619262 bytes
Requests per second: 6725.32 [#/sec] (mean)
Time per request: 74.346 [ms] (mean)
Time per request: 0.149 [ms] (mean, across all concurrent requests)
Transfer rate: 3819.18 [Kbytes/sec] received
Connection Times (ms)
min mean[+/-sd] median max
Connect: 1 30 74.6 24 2056
Processing: 12 44 15.8 42 1112
Waiting: 3 33 15.2 31 1100
Total: 25 74 76.4 67 2094
Percentage of the requests served within a certain time (ms)
50% 67
66% 74
75% 78
80% 80
90% 86
95% 92
98% 97
99% 101
100% 2094 (longest request)
[ec2-user@ip-10-0-1-160 ~]$ ^C
[ec2-user@ip-10-0-1-160 ~]$ ^C
[ec2-user@ip-10-0-1-160 ~]$ ^C
[ec2-user@ip-10-0-1-160 ~]$ ^C
[ec2-user@ip-10-0-1-160 ~]$ ^C
[ec2-user@ip-10-0-1-160 ~]$ ^C
[ec2-user@ip-10-0-1-160 ~]$ ^C
[ec2-user@ip-10-0-1-160 ~]$ ^C
[ec2-user@ip-10-0-1-160 ~]$


CloudWatch
Alarms (5)
ScaleUpCPUUtilizationAlarm
Graph
CPUUtilization
ScaleDownCPUUtilizationAlarm
Metric alarm
In alarm
ScaleUpCPUUtilizationAlarm
Metric alarm
OK


Alarms (1/1)
Name
State
Last state update (UTC)
Conditions
Actions
ScaleDownCPUUtilizationAlarm
In alarm
2024-11-03 12:21:26
CPUUtilization < 30 for 2 datapoints within 10 minutes
Actions enabled


EC2 > Auto Scaling groups
Auto Scaling groups (1/1) Info
Launch configurations
Launch templates
Actions
Create Auto Scaling group
Search your Auto Scaling groups
Name
Launch template/configuratio...
Instan...
Status
Desired capac...
Min
terraform-
Auto Scaling group: terraform-20241031095010218800000003
Status
Description
Cause
Start time
Connecti on draining in progress
Terminating EC2 instance: i-058b967e362272ea8 - Waiting For ELB Connection Draining.
At 2024-11-03T12:21:26Z a monitor alarm ScaleDownCPUUtilizationAlarm in state ALARM triggered policy scale-down changing the desired capacity from 2 to 1. At 2024-11-03T12:21:31Z an instance was taken out of service in response to a difference between desired and actual capacity, shrinking the capacity from 2 to 1. At 2024-11-03T12:21:31Z instance i-058b967e362272ea8 was selected for termination.
2024 November 03, 05:21:31 AM -07:00
Successf ul
Launching a new EC2 instance: i-058b967e362272ea8
At 2024-11-03T12:11:01Z a monitor alarm ScaleUpCPUUtilizationAlarm in state ALARM triggered policy scale-up changing the desired capacity from 1 to 2. At 2024-11-03T12:11:03Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 1 to 2.
2024 November 03, 05:11:05 AM -07:00
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Instances (1/2) Info
Last updated less than a minute ago
Connect
Instance state
Actions
Launch instances
Find instance by attribute or tag (case-sensitive)
All states
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
Availability Zone
Public IPv
AppASGInstance
i-05a5b8a278d2f613c
Running
t2.micro
2/2 checks pass
View alarms
us-east-1a
ec2-3-236
AppASGInstance
i-058b967e362272ea8
Terminated
t2.micro
–
View alarms
us-east-1b
–
i-058b967e362272ea8 (AppASGInstance)
Details
Status and alarms
Monitoring
Security
Networking
Storage
Tags
Instance summary Info
Instance ID
i-058b967e362272ea8
Public IPv4 address
–
Private IPv4 addresses
–
IPv6 address
–
Instance state
Terminated
Public IPv4 DNS
–
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

4) CloudWatch Alarms and Notifications Testing

a) Overview

This phase of the project focused on verifying the functionality of the CloudWatch alarms and ensuring that notifications were properly sent via email when specific thresholds were crossed. For this test, I chose to trigger the `ScaleUpCPUUtilizationAlarm` in the us-east-1 (N. Virginia) region, as it provided a quick way to confirm that the configured Simple Notification Service (SNS) would send an alert email when the alarm was triggered.

b) Preparation: Installing Stress Tool for Load Testing

To simulate high CPU usage on the EC2 instance and trigger the alarm, I connected to one of the EC2 instances in us-east-1 and installed the `stress` tool, which is commonly used for generating CPU load.

```
sudo yum install stress -y
```

This tool installation completed successfully, enabling me to proceed with the testing.

c) Triggering the Scale-Up Alarm

After installing the tool, I ran the following command to place CPU load on the instance:

```
stress --cpu 2
```

With this command running, the instance CPU utilization increased significantly. I accessed the `ScaleUpCPUUtilizationAlarm` in the CloudWatch Console in us-east-1 to monitor the alarm's status. The CPU utilization showed a spike to 99.1%, which met the alarm’s threshold conditions (greater than 80% for two consecutive data points within 10 minutes).

d) Verifying the Notification

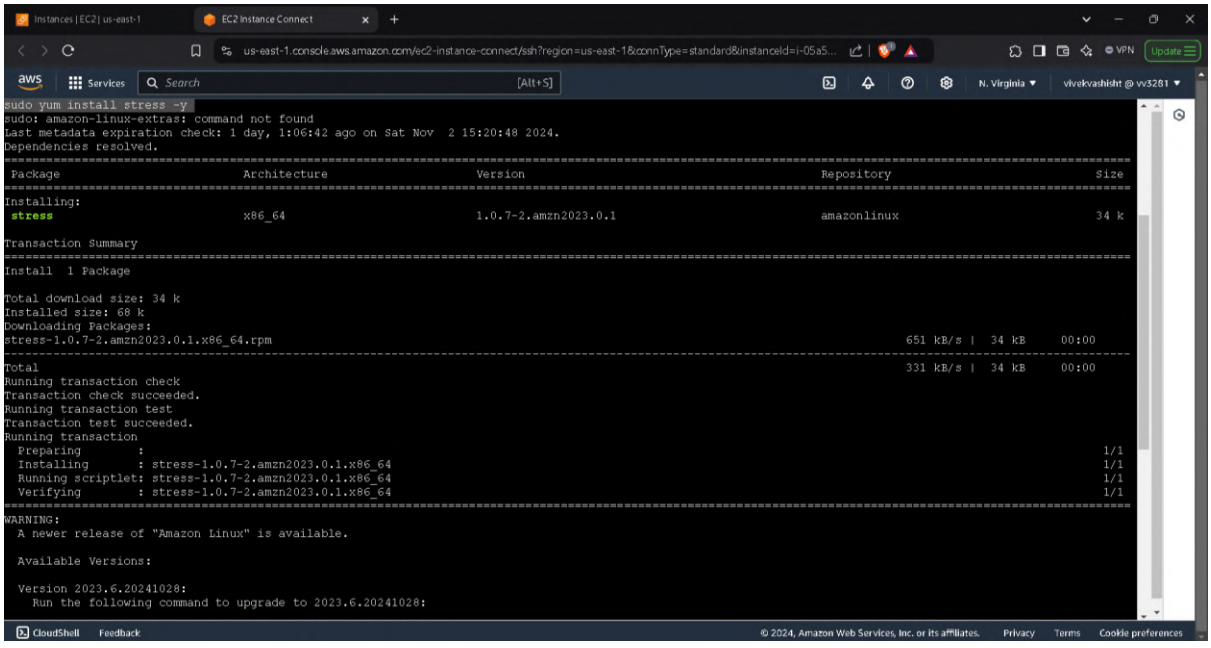
As expected, the `ScaleUpCPUUtilizationAlarm` entered the "ALARM" state due to the high CPU load. I received an email notification from SNS to the configured email address. The notification email included key details, confirming that the alarm was working as configured. Here’s an excerpt from the email notification:

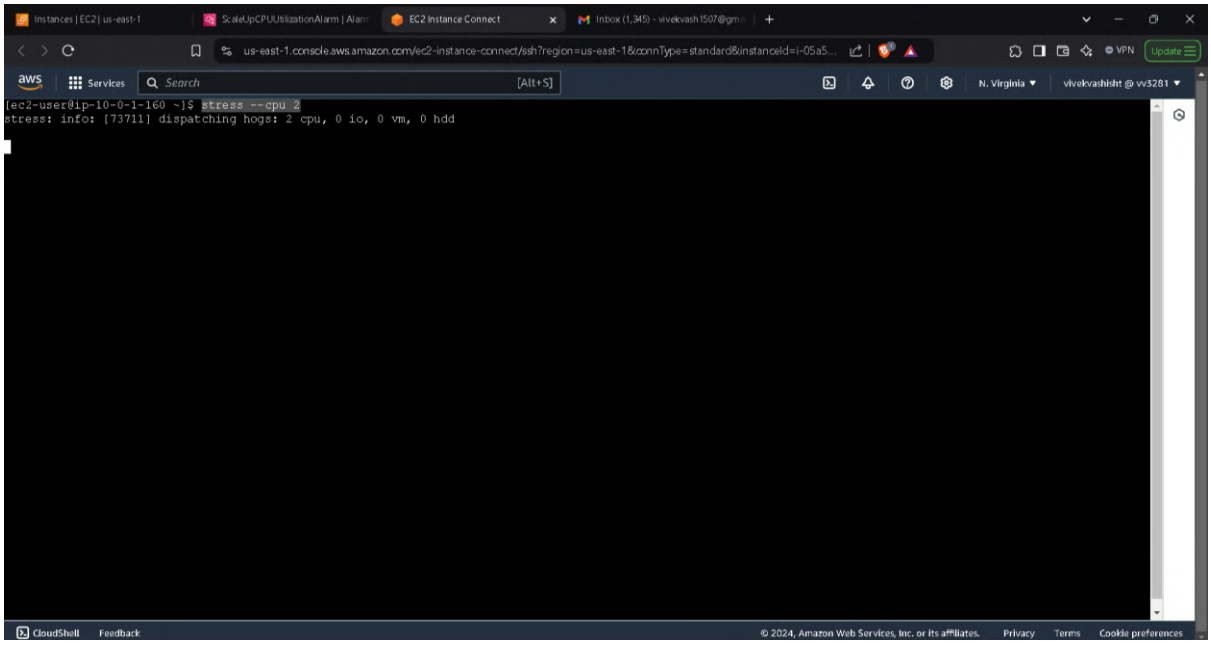
Email Notification:

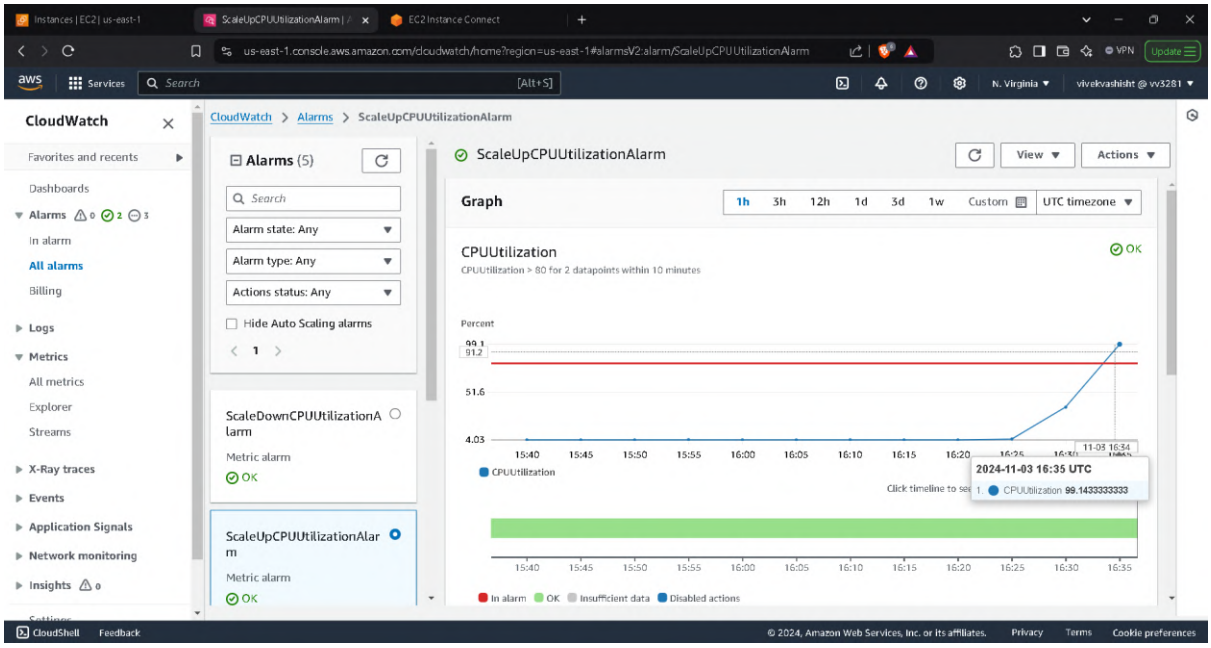
```
AWS Notifications <no-reply@sns.amazonaws.com>
9:41 AM (4 minutes ago)
You are receiving this email because your Amazon CloudWatch Alarm
"ScaleUpCPUUtilizationAlarm" in the US East (N. Virginia) region has
entered the ALARM state, because "Threshold Crossed: 2 datapoints
[99.20501324977917 (03/11/24 16:36:00), 98.81166107758926 (03/11/24
16:31:00)] were greater than the threshold (80.0)." at "Sunday 03 November,
2024 16:41:01 UTC".

View this alarm in the AWS Management Console:
https://us-east-1.console.aws.amazon.com/cloudwatch/deeplink.js?region=us-
east-1/alarm/ScaleUpCPUUtilizationAlarm

**Alarm Details:**
- **Name:** ScaleUpCPUUtilizationAlarm
- **Description:** This alarm triggers if CPU utilization exceeds 80% to
scale up.
- **State Change:** OK -> ALARM
- **Timestamp:** Sunday 03 November, 2024 16:41:01 UTC
- **Reason for State Change:** Threshold Crossed: 2 datapoints
[99.20501324977917 (03/11/24 16:36:00), 98.81166107758926 (03/11/24
16:31:00)] were greater than the threshold (80.0).
- **AWS Account:** 134575801911
- **Alarm Arn:** arn:aws:cloudwatch:us-east-
1:134575801911:alarm:ScaleUpCPUUtilizationAlarm

**Threshold:**
The alarm is in the ALARM state when the metric is Greater Than Threshold
for at least 2 of the last 2 period(s) of 300 seconds.
```

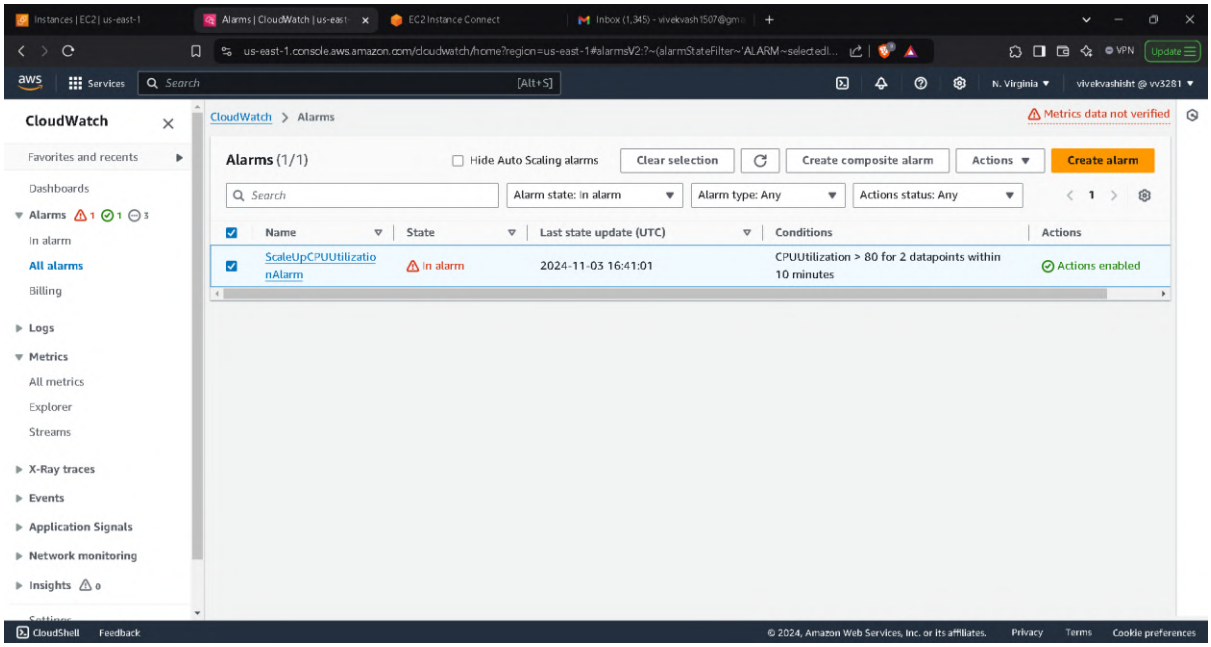
The testing was successful, confirming that the CloudWatch alarm triggers as expected under high CPU load and sends an email notification through SNS to the designated email. This test validated both the alarm configuration and the email notification setup.

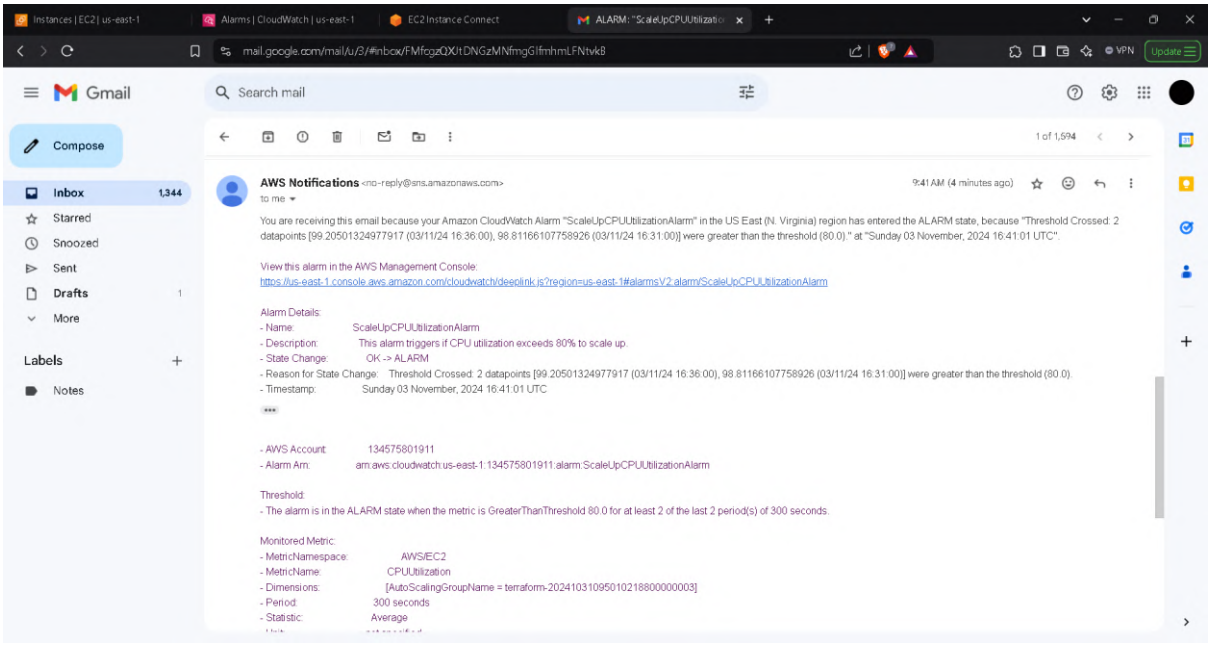
Screenshots




[ec2-user@ip-10-0-1-160 ~]$ stress --cpu 2
stress: info: [73711] dispatching hogs: 2 cpu, 0 io, 0 vm, 0 hdd


CloudWatch
Alarms (5)
ScaleUpCPUUtilizationAlarm
Graph
CPUUtilization
ScaleDownCPUUtilizationAlarm
Metric alarm
OK
ScaleUpCPUUtilizationAlarm
Metric alarm
OK
2024-11-03 16:35 UTC
CPUUtilization 99.1433333333


CloudWatch
Alarms (1/1)
Metrics data not verified
ScaleUpCPUUtilizationAlarm
In alarm
2024-11-03 16:41:01
CPUUtilization > 80 for 2 datapoints within 10 minutes
Actions enabled



Cleanup

a) Overview

In this final phase, the objective was to clean up and delete all AWS resources created during the project. Running `terraform destroy` ensures that all infrastructure is decommissioned, preventing unnecessary costs by terminating any active resources across all regions. This included resources like EC2 instances, S3 buckets, Route 53 records, and more.

b) Initial Destroy Plan

To start, I opened the command line interface (CLI) and ran the following command to view a detailed plan of the resources scheduled for destruction:

```
terraform plan -destroy
```

The output displayed a total of **63 resources** that Terraform would destroy. Reviewing this plan allowed me to verify that all resources were identified and included in the cleanup process.

c) Executing the Destroy Command

Once I confirmed the destroy plan, I executed the following command to initiate the cleanup:

```
terraform destroy -auto-approve
```

This command started the process of tearing down the infrastructure. However, an issue arose during the destruction process when Terraform encountered an error while attempting to

delete the S3 bucket named `aws-tf-backend-vv-bucket`. This error occurred because the bucket was not empty; it contained the `terraform.tfstate` file, which needed to be removed before the bucket could be deleted.

d) Resolving the S3 Bucket Issue

To resolve the issue, I navigated to the AWS Management Console, accessed the `aws-tf-backend-vv-bucket` in the S3 service, and manually deleted its contents. Once the bucket was empty, it was ready for deletion.

e) Re-running the Destroy Command

After clearing the S3 bucket, I ran the `terraform destroy` command once more:

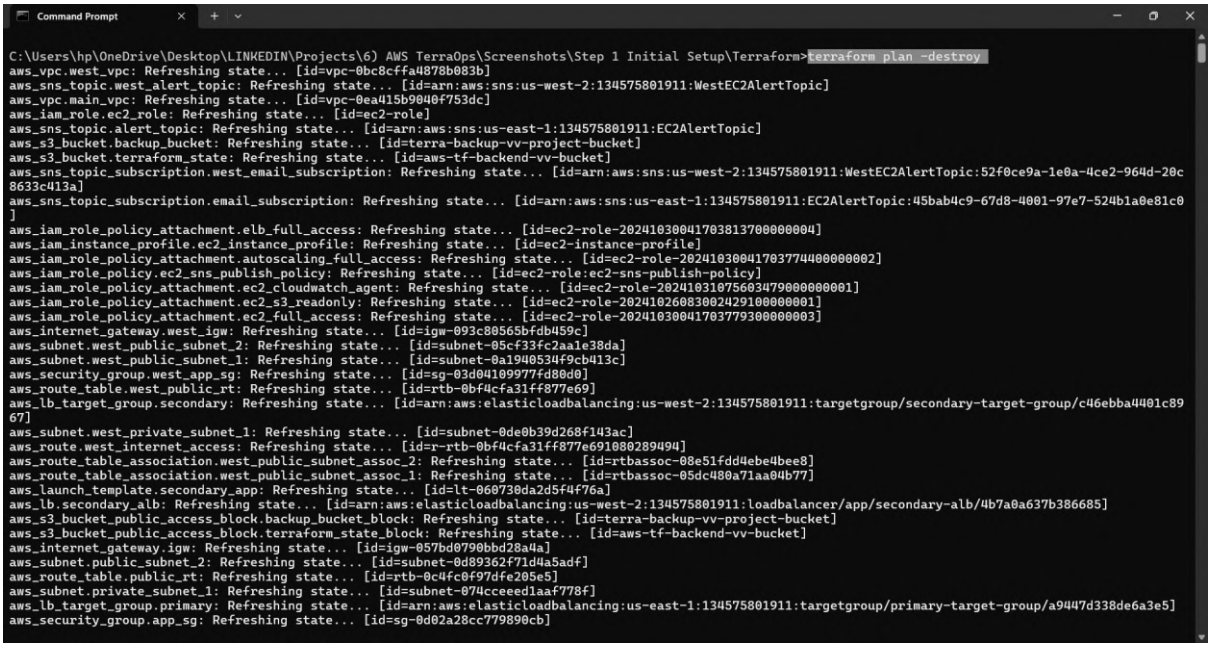
```
terraform destroy -auto-approve
```

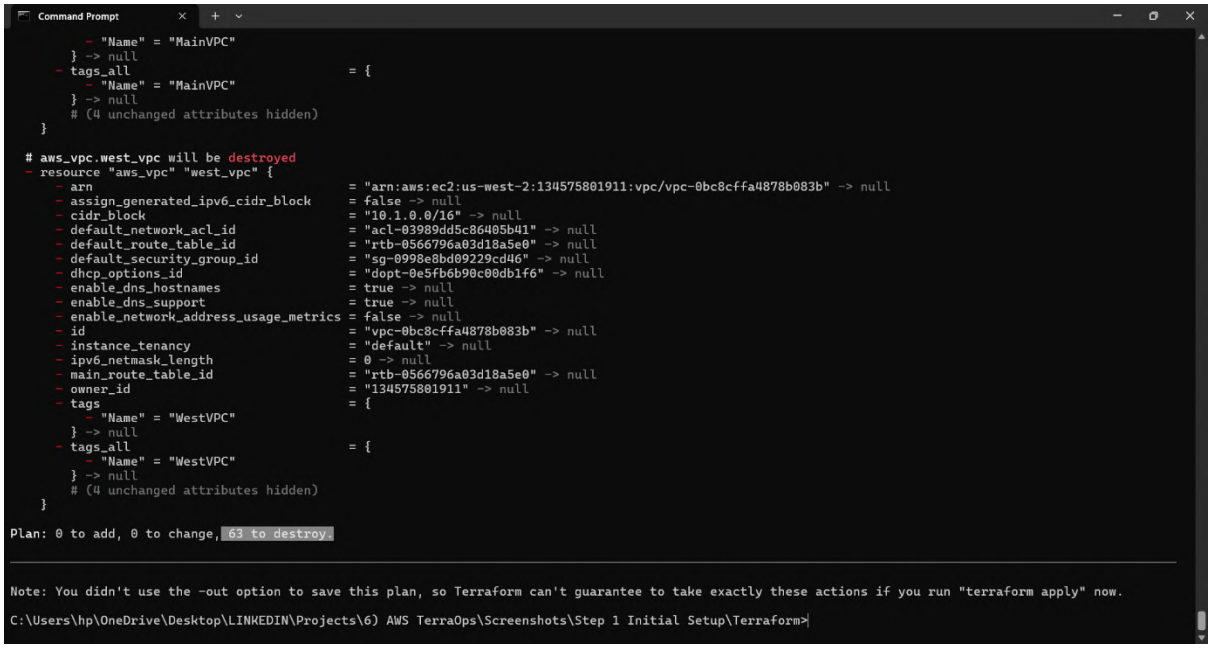
This time, the destruction process completed successfully. The command output confirmed that all resources were deleted, displaying the message:

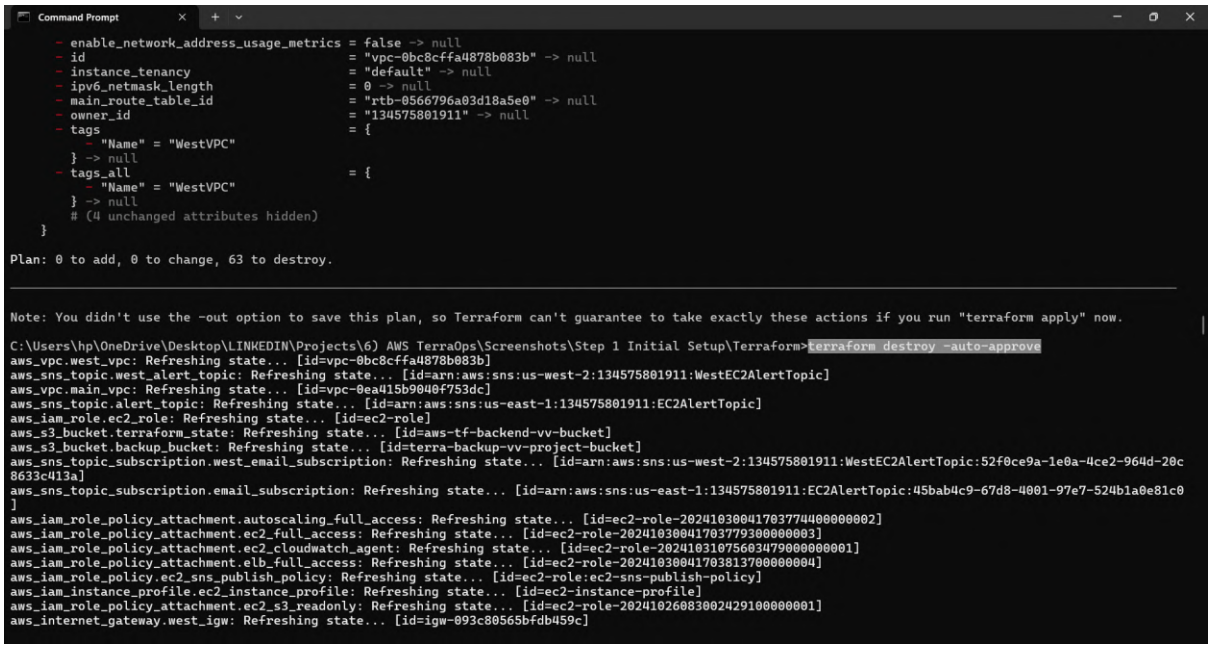
```
No changes. No objects need to be destroyed.
```

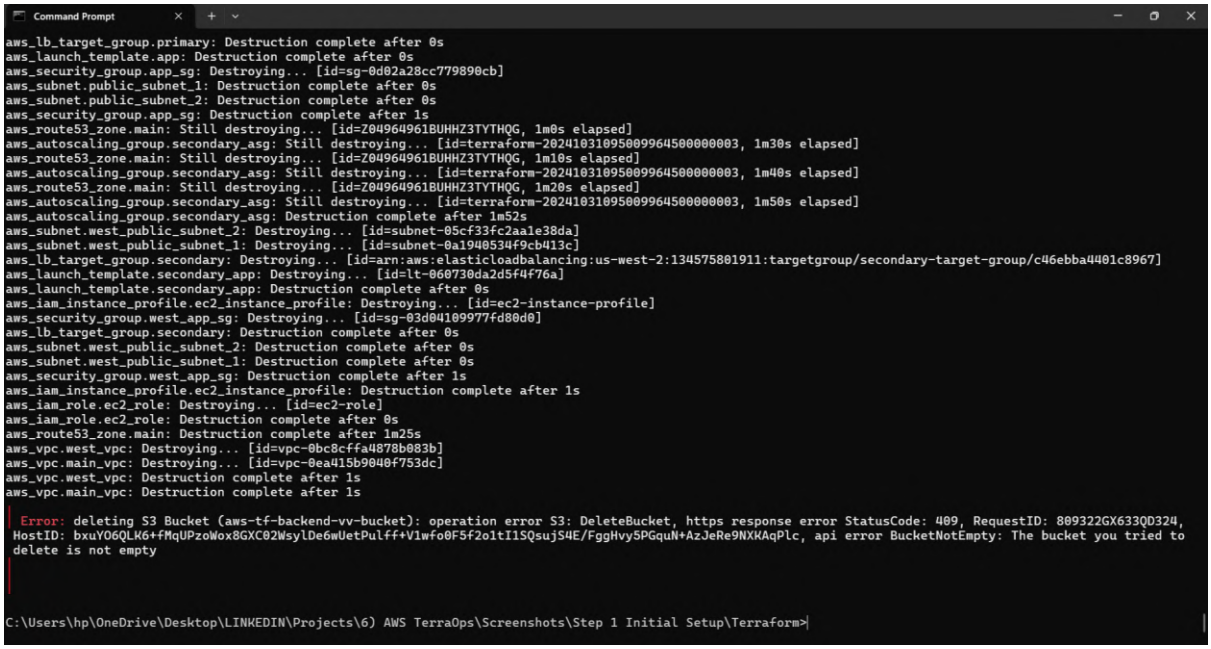
With the successful execution of `terraform destroy`, all resources associated with the project were fully decommissioned. This ensured a complete cleanup of the project infrastructure, marking the final and successful completion of the project.

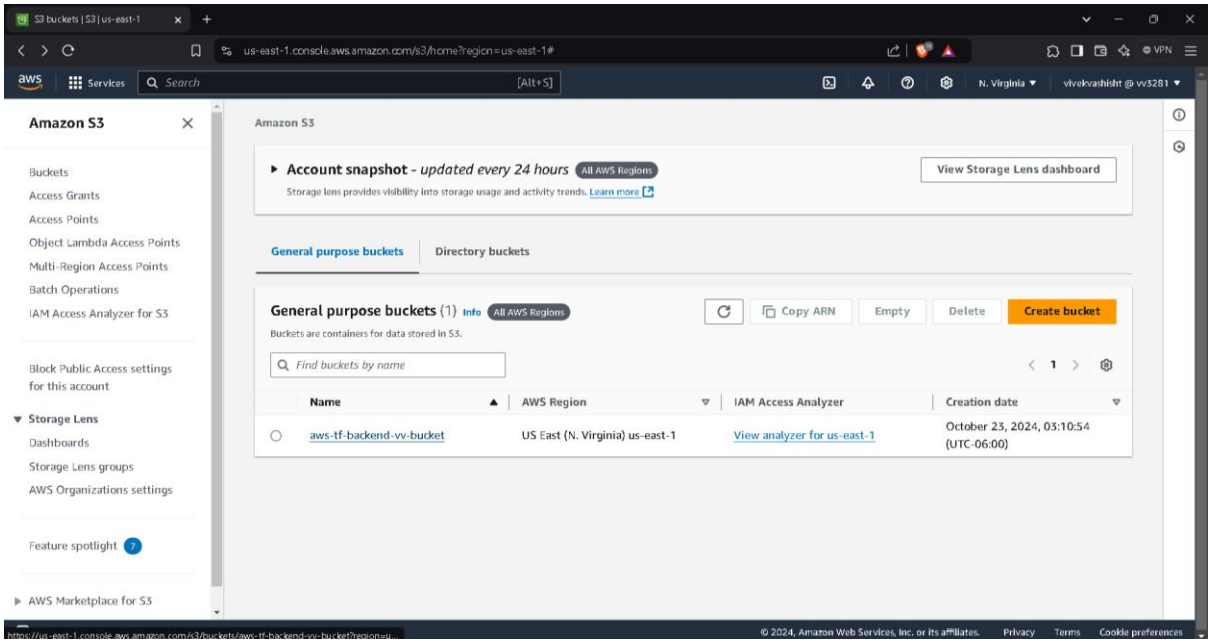
Screenshots

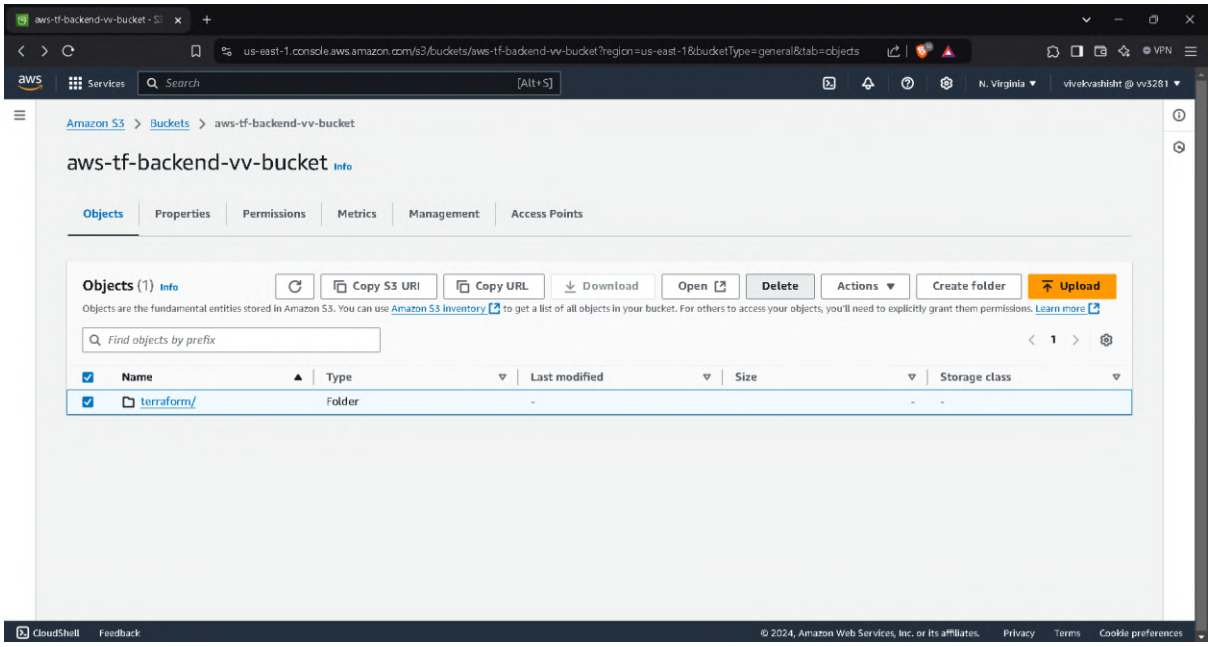


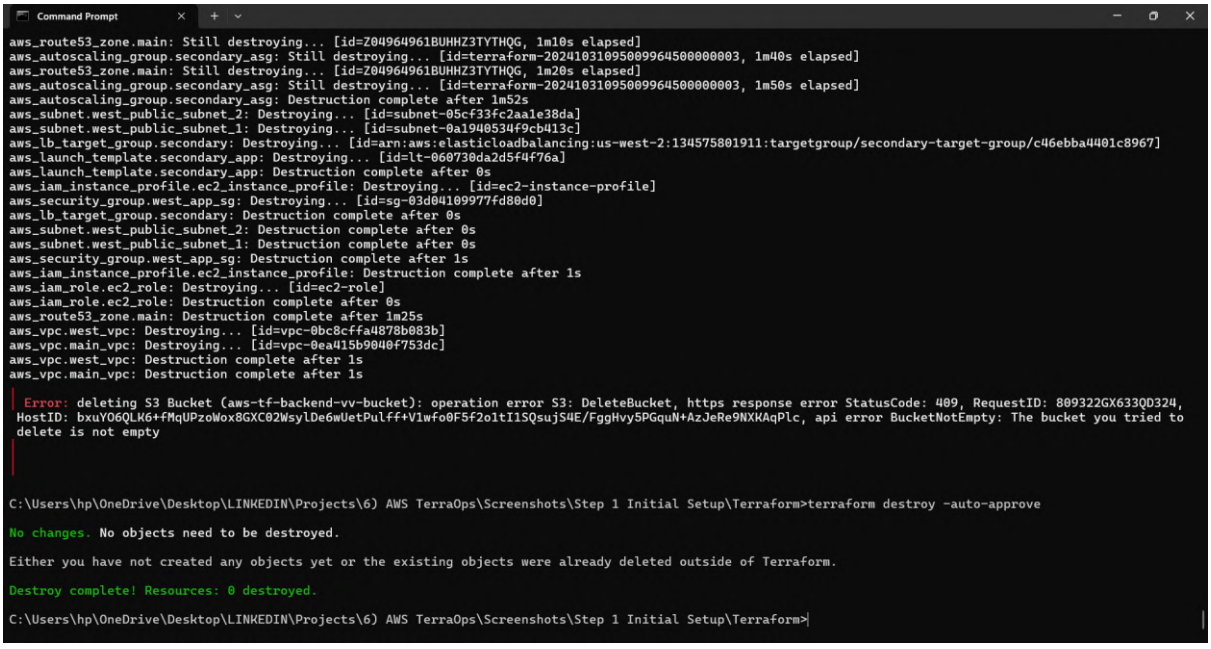

Command Prompt
- "Name" = "MainVPC"
} -> null
- tags_all = {
- "Name" = "MainVPC"
} -> null
# (4 unchanged attributes hidden)
}
# aws_vpc.west_vpc will be destroyed
- resource "aws_vpc" "west_vpc" {
- arn = "arn:aws:ec2:us-west-2:134575801911:vpc/vpc-0bc8cffa4878b083b" -> null
- assign_generated_ipv6_cidr_block = false -> null
- cidr_block = "10.1.0.0/16" -> null
- default_network_acl_id = "acl-03989dd5c86405b41" -> null
- default_route_table_id = "rtb-0566796a03d18a5e0" -> null
- default_security_group_id = "sg-0998e8bd09229cd46" -> null
- dhcp_options_id = "dopt-0e5fb6b90c00db1f6" -> null
- enable_dns_hostnames = true -> null
- enable_dns_support = true -> null
- enable_network_address_usage_metrics = false -> null
- id = "vpc-0bc8cffa4878b083b" -> null
- instance_tenancy = "default" -> null
- ipv6_netmask_length = 0 -> null
- main_route_table_id = "rtb-0566796a03d18a5e0" -> null
- owner_id = "134575801911" -> null
- tags = {
- "Name" = "WestVPC"
} -> null
- tags_all = {
- "Name" = "WestVPC"
} -> null
# (4 unchanged attributes hidden)
}
Plan: 0 to add, 0 to change, 63 to destroy.
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.
C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>


Command Prompt
- enable_network_address_usage_metrics = false -> null
- id = "vpc-0bc8cffa4878b083b" -> null
- instance_tenancy = "default" -> null
- ipv6_netmask_length = 0 -> null
- main_route_table_id = "rtb-0566796a03d18a5e0" -> null
- owner_id = "134575801911" -> null
- tags = {
- "Name" = "WestVPC"
} -> null
- tags_all = {
- "Name" = "WestVPC"
} -> null
# (4 unchanged attributes hidden)
}
Plan: 0 to add, 0 to change, 63 to destroy.
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.
C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>terraform destroy -auto-approve
aws_vpc.west_vpc: Refreshing state... [id=vpc-0bc8cffa4878b083b]
aws_sns_topic.west_alert_topic: Refreshing state... [id=arn:aws:sns:us-west-2:134575801911:WestEC2AlertTopic]
aws_vpc.main_vpc: Refreshing state... [id=vpc-0ea415b9040f753dc]
aws_sns_topic.alert_topic: Refreshing state... [id=arn:aws:sns:us-east-1:134575801911:EC2AlertTopic]
aws_iam_role.ec2_role: Refreshing state... [id=ec2-role]
aws_s3_bucket.terraform_state: Refreshing state... [id=aws-tf-backend-vv-bucket]
aws_s3_bucket.backup_bucket: Refreshing state... [id=terra-backup-vv-project-bucket]
aws_sns_topic_subscription.west_email_subscription: Refreshing state... [id=arn:aws:sns:us-west-2:134575801911:WestEC2AlertTopic:52f0ce9a-1e0a-4ce2-964d-20c8633c413a]
aws_sns_topic_subscription.email_subscription: Refreshing state... [id=arn:aws:sns:us-east-1:134575801911:EC2AlertTopic:45bab4c9-67d8-4001-97e7-524b1a0e81c0]
aws_iam_role_policy_attachment.autoscaling_full_access: Refreshing state... [id=ec2-role-2024103004170377440000002]
aws_iam_role_policy_attachment.ec2_full_access: Refreshing state... [id=ec2-role-2024103004170377930000003]
aws_iam_role_policy_attachment.ec2_cloudwatch_agent: Refreshing state... [id=ec2-role-2024103107560347900000001]
aws_iam_role_policy_attachment.elb_full_access: Refreshing state... [id=ec2-role-2024103004170381370000004]
aws_iam_role_policy.ec2_sns_publish_policy: Refreshing state... [id=ec2-role:ec2-sns-publish-policy]
aws_iam_instance_profile.ec2_instance_profile: Refreshing state... [id=ec2-instance-profile]
aws_iam_role_policy_attachment.ec2_s3_readonly: Refreshing state... [id=ec2-role-2024102608300242910000001]
aws_internet_gateway.west_igw: Refreshing state... [id=igw-093c80565bfdb459c]


Command Prompt
aws_lb_target_group.primary: Destruction complete after 0s
aws_launch_template.app: Destruction complete after 0s
aws_security_group.app_sg: Destroying... [id=sg-0d02a28cc779890cb]
aws_subnet.public_subnet_1: Destruction complete after 0s
aws_subnet.public_subnet_2: Destruction complete after 0s
aws_security_group.app_sg: Destruction complete after 1s
aws_route53_zone.main: Still destroying... [id=Z04964961BUHHZ3TYTHQG, 1m0s elapsed]
aws_autoscaling_group.secondary_asg: Still destroying... [id=terraform-20241031095009964500000003, 1m30s elapsed]
aws_route53_zone.main: Still destroying... [id=Z04964961BUHHZ3TYTHQG, 1m10s elapsed]
aws_autoscaling_group.secondary_asg: Still destroying... [id=terraform-20241031095009964500000003, 1m40s elapsed]
aws_route53_zone.main: Still destroying... [id=Z04964961BUHHZ3TYTHQG, 1m20s elapsed]
aws_autoscaling_group.secondary_asg: Still destroying... [id=terraform-20241031095009964500000003, 1m50s elapsed]
aws_autoscaling_group.secondary_asg: Destruction complete after 1m52s
aws_subnet.west_public_subnet_2: Destroying... [id=subnet-05cf33fc2aa1e38da]
aws_subnet.west_public_subnet_1: Destroying... [id=subnet-0a1940534f9cb413c]
aws_lb_target_group.secondary: Destroying... [id=arn:aws:elasticloadbalancing:us-west-2:134575801911:targetgroup/secondary-target-group/c46ebba4401c8967]
aws_launch_template.secondary_app: Destroying... [id=lt-060730da2d5f4f76a]
aws_launch_template.secondary_app: Destruction complete after 0s
aws_iam_instance_profile.ec2_instance_profile: Destroying... [id=ec2-instance-profile]
aws_security_group.west_app_sg: Destroying... [id=sg-03d04109977fd80d0]
aws_lb_target_group.secondary: Destruction complete after 0s
aws_subnet.west_public_subnet_2: Destruction complete after 0s
aws_subnet.west_public_subnet_1: Destruction complete after 0s
aws_security_group.west_app_sg: Destruction complete after 1s
aws_iam_instance_profile.ec2_instance_profile: Destruction complete after 1s
aws_iam_role.ec2_role: Destroying... [id=ec2-role]
aws_iam_role.ec2_role: Destruction complete after 0s
aws_route53_zone.main: Destruction complete after 1m25s
aws_vpc.west_vpc: Destroying... [id=vpc-0bc8cffa4878b083b]
aws_vpc.main_vpc: Destroying... [id=vpc-0ea415b9040f753dc]
aws_vpc.west_vpc: Destruction complete after 1s
aws_vpc.main_vpc: Destruction complete after 1s
Error: deleting S3 Bucket (aws-tf-backend-vv-bucket): operation error S3: DeleteBucket, https response error StatusCode: 409, RequestID: 809322GX633QD324, HostID: bxuYO6QLK6+fMqUPzoWox8GXC02WsylDe6wUetPulff+V1wfo0F5f2o1tI1SQsujS4E/FggHvy5PGquN+AzJeRe9NXKAqPlc, api error BucketNotEmpty: The bucket you tried to delete is not empty
C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>


S3 buckets | S3 | us-east-1
us-east-1.console.aws.amazon.com/s3/home?region=us-east-1#
Services
Search
[Alt+S]
N. Virginia
vivekvashisht @ vv3281
Amazon S3
Buckets
Access Grants
Access Points
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
IAM Access Analyzer for S3
Block Public Access settings for this account
Storage Lens
Dashboards
Storage Lens groups
AWS Organizations settings
Feature spotlight
AWS Marketplace for S3
Amazon S3
Account snapshot - updated every 24 hours
All AWS Regions
Storage lens provides visibility into storage usage and activity trends. Learn more
View Storage Lens dashboard
General purpose buckets
Directory buckets
General purpose buckets (1) Info
All AWS Regions
Buckets are containers for data stored in S3.
Copy ARN
Empty
Delete
Create bucket
Find buckets by name
Name
AWS Region
IAM Access Analyzer
Creation date
aws-tf-backend-vv-bucket
US East (N. Virginia) us-east-1
View analyzer for us-east-1
October 23, 2024, 03:10:54 (UTC-06:00)
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


aws-tf-backend-vv-bucket - S3
us-east-1.console.aws.amazon.com/s3/buckets/aws-tf-backend-vv-bucket?region=us-east-1&bucketType=general&tab=objects
Services
Search
[Alt+S]
N. Virginia
vivekvashisht @ vv3281
Amazon S3 > Buckets > aws-tf-backend-vv-bucket
aws-tf-backend-vv-bucket Info
Objects
Properties
Permissions
Metrics
Management
Access Points
Objects (1) Info
Copy S3 URI
Copy URL
Download
Open
Delete
Actions
Create folder
Upload
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more
Find objects by prefix
Name
Type
Last modified
Size
Storage class
terraform/
Folder
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Command Prompt
aws_route53_zone.main: Still destroying... [id=Z04964961BUHHZ3TYTHQG, 1m10s elapsed]
aws_autoscaling_group.secondary_asg: Still destroying... [id=terraform-20241031095009964500000003, 1m40s elapsed]
aws_route53_zone.main: Still destroying... [id=Z04964961BUHHZ3TYTHQG, 1m20s elapsed]
aws_autoscaling_group.secondary_asg: Still destroying... [id=terraform-20241031095009964500000003, 1m50s elapsed]
aws_autoscaling_group.secondary_asg: Destruction complete after 1m52s
aws_subnet.west_public_subnet_2: Destroying... [id=subnet-05cf33fc2aa1e38da]
aws_subnet.west_public_subnet_1: Destroying... [id=subnet-0a1940534f9cb413c]
aws_lb_target_group.secondary: Destroying... [id=arn:aws:elasticloadbalancing:us-west-2:134575801911:targetgroup/secondary-target-group/c46ebba4401c8967]
aws_launch_template.secondary_app: Destroying... [id=lt-060730da2d5f4f76a]
aws_launch_template.secondary_app: Destruction complete after 0s
aws_iam_instance_profile.ec2_instance_profile: Destroying... [id=ec2-instance-profile]
aws_security_group.west_app_sg: Destroying... [id=sg-03d04109977fd80d0]
aws_lb_target_group.secondary: Destruction complete after 0s
aws_subnet.west_public_subnet_2: Destruction complete after 0s
aws_subnet.west_public_subnet_1: Destruction complete after 0s
aws_security_group.west_app_sg: Destruction complete after 1s
aws_iam_instance_profile.ec2_instance_profile: Destruction complete after 1s
aws_iam_role.ec2_role: Destroying... [id=ec2-role]
aws_iam_role.ec2_role: Destruction complete after 0s
aws_route53_zone.main: Destruction complete after 1m25s
aws_vpc.west_vpc: Destroying... [id=vpc-0bc8cffa4878b083b]
aws_vpc.main_vpc: Destroying... [id=vpc-0ea415b9040f753dc]
aws_vpc.west_vpc: Destruction complete after 1s
aws_vpc.main_vpc: Destruction complete after 1s
Error: deleting S3 Bucket (aws-tf-backend-vv-bucket): operation error S3: DeleteBucket, https response error StatusCode: 409, RequestID: 809322GX633QD324, HostID: bxuYO6QLK6+fMqUPzoWox8GXC02WsylDe6wUetPulff+V1wfo0F5f2o1tI1SQsujS4E/FggHvy5PGquN+AzJeRe9NXKAqPlc, api error BucketNotEmpty: The bucket you tried to delete is not empty
C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>terraform destroy -auto-approve
No changes. No objects need to be destroyed.
Either you have not created any objects yet or the existing objects were already deleted outside of Terraform.
Destroy complete! Resources: 0 destroyed.
C:\Users\hp\OneDrive\Desktop\LINKEDIN\Projects\6) AWS TerraOps\Screenshots\Step 1 Initial Setup\Terraform>

Conclusion

The AWS TerraOps project demonstrated the deployment of a highly available, scalable, and automated cloud infrastructure using AWS services and Terraform. This initiative was designed to validate practical skills for AWS Cloud Practitioner and HashiCorp Terraform Associate certifications, ensuring a robust understanding of infrastructure automation and cloud scaling. Below is a concise summary of each major step:

Summary of Steps:

1. **Initial Setup**: Installed Terraform and AWS CLI, configured AWS credentials, and set up version control with Git. Established an S3 bucket for storing Terraform state and configured remote backend storage.
2. **VPC and Networking Configuration**: Created a Virtual Private Cloud (VPC) with public and private subnets, set up an Internet Gateway, and associated public route tables to enable communication with the internet.
3. **EC2 and Auto Scaling Setup**: Defined a security group for EC2 instances, set up a launch template with a user data script to install Nginx, and configured an Auto Scaling Group (ASG) with policies for scaling instances based on demand.
4. **IAM and S3 Integration**: Configured IAM roles and policies to allow EC2 instances access to S3, then set up an additional S3 bucket for storing backups.
5. **Multi-Region Deployment and Failover**: Established resources in two AWS regions, created Application Load Balancers (ALBs) in each region, and implemented Route 53 for DNS-based failover. This configuration ensures high availability and resiliency by redirecting traffic to the secondary region in case of primary region failures.
6. **Monitoring and Alerts**: Installed the CloudWatch agent on EC2 instances to monitor CPU, memory, and disk utilization. Configured CloudWatch Alarms and set up SNS notifications to alert based on resource usage.
7. **Parameterization and Validation**: Introduced variables for reusability and validation checks to enforce correct inputs, enhancing flexibility and scalability.
8. **Testing and Validation**:
    - **ALB Testing**: Verified load balancing across multiple instances within each region.
    - **Route 53 Failover Testing**: Simulated a failure in the primary region to confirm automatic failover to the secondary region.
    - **ASG Testing**: Validated scaling policies by monitoring CPU utilization, triggering scale-up and scale-down actions based on load.
    - **CloudWatch Alarms and Notifications Testing:** Verified alarm configurations by simulating metric thresholds, ensuring alerts are triggered accurately and notifications are sent to designated channels

This project successfully validated the core competencies of deploying and managing AWS infrastructure with Terraform. The automated deployment process and multi-region failover setup illustrated a strong foundation in cloud infrastructure automation, enhancing availability, scalability, and resiliency. This documentation and practical experience add credibility to the theoretical knowledge acquired through AWS and Terraform certifications, providing a complete end-to-end infrastructure solution ideal for real-world applications.

Lessons Learned

• **Significance of Proactive Monitoring and Alerts**: Understood the critical role of proactive monitoring and alert configurations for AWS resources, ensuring any abnormal events are quickly identified and addressed to maintain system reliability and prevent downtime.

• **AWS Security Best Practices**: Gained a deeper understanding of AWS security best practices, including the effective use of IAM policies, Security Groups, encryption, and Multi-Factor Authentication (MFA) to secure cloud infrastructure.

• **Scalability and Fault Tolerance**: Learned how to implement Auto Scaling and failover mechanisms to ensure high availability and resilience, enabling resources to handle varying loads and maintain performance during peak demand.

• **Automation with Terraform**: Acquired hands-on experience in using Terraform for infrastructure automation, streamlining the process of deploying and managing cloud resources while maintaining an organized and efficient environment.

• **Enhanced Understanding of Infrastructure Automation**: Gained in-depth knowledge of how to use Terraform to create, modify, and version control infrastructure efficiently, promoting repeatable and consistent deployments across environments.

• **Efficient Resource Scaling and Load Management**: Realized the importance of automated scaling policies to handle traffic fluctuations without manual intervention, ensuring cost-effective and efficient resource utilization.

• **Cross-Region Failover and Disaster Preparedness**: Learned to plan for and implement cross-region failover strategies with Route 53, enhancing fault tolerance and ensuring high availability even in case of regional disruptions.

Skills Demonstrated

Through the AWS TerraOps Project, the following skills were showcased:

1. **Infrastructure as Code (IaC) Implementation**:
    - Designed and deployed AWS infrastructure using Terraform.
    - Created modular and reusable Terraform code to define cloud resources effectively.
2. **Scalable and Resilient Architecture Design**:
    - Configured Auto Scaling Groups to manage dynamic workloads.
    - Implemented Load Balancing with Application Load Balancer (ALB) for optimized traffic distribution.
3. **Monitoring and Alert Configuration**:
    - Set up monitoring using CloudWatch, capturing metrics for resource health and performance.
    - Configured custom CloudWatch alarms to trigger notifications based on specific metrics and conditions.

4. **Security Hardening**:
    - Applied IAM roles, policies, and security group configurations for secure access control.
    - Enabled encryption at rest and in transit, aligning with best practices for data protection.
5. **Automated Infrastructure Management**:
    - Developed infrastructure automation scripts with Terraform, improving deployment speed and accuracy.
    - Leveraged CloudFormation as needed for additional automation, enhancing deployment flexibility.
6. **Network Architecture and Troubleshooting**:
    - Designed VPC architecture with subnets and route tables to isolate and manage resources effectively.
    - Utilized VPC Flow Logs for network analysis and troubleshooting.
7. **Backup and Recovery Setup**:
    - Configured AWS Backup for critical resources, ensuring a secure recovery strategy.
    - Tested backup and restoration processes to validate disaster recovery readiness.
8. **Cost Management and Resource Optimization**:
    - Practiced cost management by cleaning up resources after testing to prevent unnecessary charges.
    - Configured AWS Cost Explorer and Budgets for ongoing cost tracking and alerting.
9. **Performance and Availability Testing**:
    - Conducted failover testing by simulating region outages, ensuring Route 53 successfully redirected traffic.
    - Monitored EC2 instance health and load balancer performance, identifying and resolving bottlenecks.

And Strong Hands-on Capabilities in

(AWS Infrastructure Management and Automation with Terraform)

Scope of AWS Certified Cloud Practitioner (CLF-C02) and HashiCorp Certified: Terraform Associate (HCTA0-003)

HashiCorp Terraform