

# Assignment 1

**Objective:** In this assignment, you will create a simple GitHub Actions workflow that builds a custom Docker image with a basic NGINX configuration, deploys it to an Amazon EC2 instance, and ensures that the container remains running even after an EC2 instance restart.

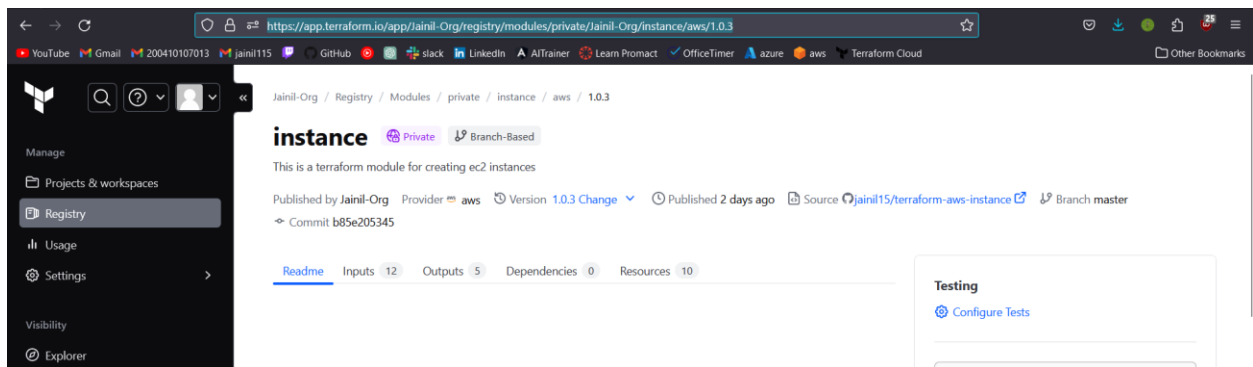
I have hosted terraform modules on github and on terraform cloud for better version control

Instance module: <https://github.com/jainil15/terraform-aws-instance>,

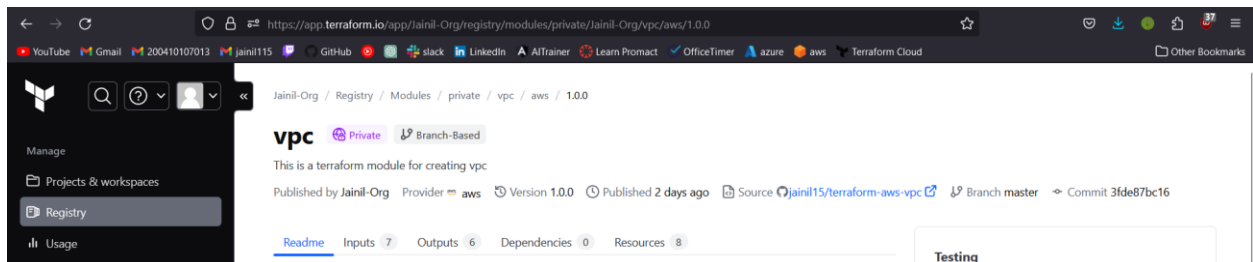
Aws module: <https://github.com/jainil15/terraform-aws-vpc>

I have also created private repository on the terraform cloud

Instance module:

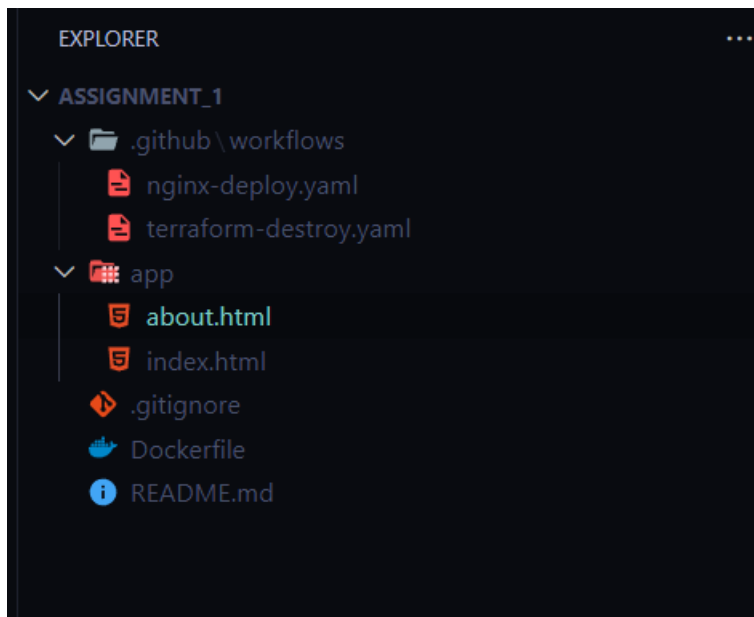


Vpc module:



Now lets create a simple application using html and docker:

1. I have already created two files index.html and about.html and placed them in app folder under root.



2. Then create a Dockerfile in root dir and enter the following code

```
FROM nginx:1.25-alpine
COPY ./app/* /usr/share/nginx/html
WORKDIR /app
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

Now lets create a new repository for storing infrastructure code.

Add Main.tf

main.tf:

```
terraform {
  backend "remote" {
    hostname      = "app.terraform.io"
    organization  = "Jainil-Org"
    workspaces {
      name = "ForAssignment"
    }
  }
}

required_providers {
  aws = {
    source  = "hashicorp/aws"
    version = "~> 5.0"
  }
}
```

```
}  
}  
  
provider "aws" {  
  region = var.aws_region  
}  
  
locals {  
  env = "dev"  
}  
  
data "aws_ami" "ubuntu" {  
  
  most_recent = true  
  
  filter {  
    name     = "name"  
    values   = ["ubuntu/images/hvm-ssd/ubuntu-jammy-22.04-amd64-server-*"]  
  }  
  
  filter {  
    name     = "virtualization-type"  
    values   = ["hvm"]  
  }  
  
  owners = ["amazon"]  
}  
data "http" "myip" {  
  url = "https://ipv4.icanhazip.com"  
}  
module "vpc" {  
  
  source              = "app.terraform.io/Jainil-0rg/vpc/aws"  
  version             = "1.0.0"  
  env                 = local.env  
  vpc_cidr_block      = "12.88.0.0/16"  
  azs                 = ["ap-south-1a"]  
  public_subnet_cidr_blocks = ["12.88.0.64/26"]  
  
}  
  
module "instance" {  
  source = "app.terraform.io/Jainil-0rg/instance/aws"  
  version = "1.0.3"  
  env     = local.env  
}
```

```

ami_id    = data.aws_ami.ubuntu.id
instance_type    = "t2.micro"
private_subnet_ids = module.vpc.private_subnet_ids
public_subnet_ids = module.vpc.public_subnet_ids
vpc_id        = module.vpc.vpc_id
public_sg_ingress_with_cidr_blocks = [
  {
    from_port    = 22
    to_port      = 22
    protocol     = "tcp"
    cidr_blocks  = ["${chomp(data.http.myip.response_body)}/32"]
  },
  {
    from_port    = 80
    to_port      = 80
    protocol     = "tcp"
    cidr_blocks  = ["0.0.0.0/0"]
    ipv6_cidr_blocks = [ "::/0"]
  },
  {
    from_port    = 443
    to_port      = 443
    protocol     = "tcp"
    cidr_blocks  = ["0.0.0.0/0"]
    ipv6_cidr_blocks = [ "::/0"]
  }
]
private_key = var.private_key
user_data   = file("./docker_installation.sh")
}

```

### 3. Create outputs.tf

```

output "public_instance_ip" {
  value = module.instance.public_instance_public_ipv4[0]
}

```

### 4. Create variables.tf

```

variable "private_key" {
  type      = string
  description = "Enter the key value pair for ssh"
}

```

```

    sensitive    = true
}

variable "aws_region" {
    type        = string
    description = "AWS region"
}

variable "my_ip" {
    type = string
    description = "my ip for ssh"
}

```

5. Create `docker_installation.sh`

```

#!/bin/bash

for pkg in docker.io docker-doc docker-compose docker-compose-v2
podman-docker containerd runc; do
    apt-get remove -y $pkg;
done

apt-get update
apt-get install -y ca-certificates curl

install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o
/etc/apt/keyrings/docker.asc
chmod a+r /etc/apt/keyrings/docker.asc

echo "deb [arch=$(dpkg --print-architecture) signed-
by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu $(. /etc/os-release && echo
"$VERSION_CODENAME") stable" | tee
/etc/apt/sources.list.d/docker.list > /dev/null
apt-get update

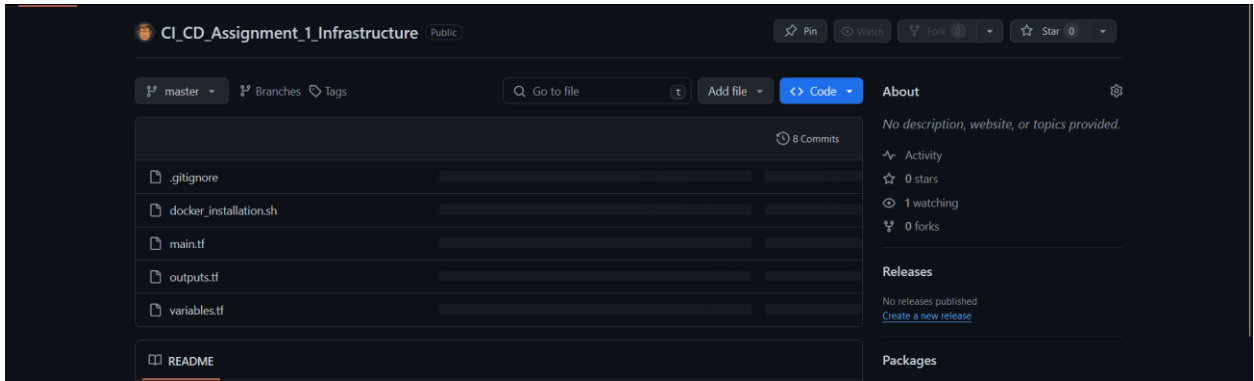
apt-get install -y docker-ce docker-ce-cli containerd.io docker-
buildx-plugin docker-compose-plugin

```

6. Generate ssh key using the following command:  
`ssh-keygen -t rsa -b 2048 -f mykeypair.pem`

```
> ssh-keygen -b 2048 -f mykeypair.pem -t rsa
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in mykeypair.pem
Your public key has been saved in mykeypair.pem.pub
The key fingerprint is:
```

7. Push these files to github repository (add private key it to .gitignore).



Now lets setup workflows for automatic deployment of nginx.

Create .github in root directory and inside that create workflow directory.

1. First lets setup nginx-deployment.yaml

```
name: "Nginx Deployment"
on:
  push:
    branches: ["dev"]
  pull_request:
    branches: ["dev"]
jobs:
  terraform:
    runs-on: ubuntu-latest
    steps:
      - name: Docker Checkout
        uses: actions/checkout@v4

      - name: Docker Login
        uses: docker/login-action@v3
        with:
          username: ${ secrets.DOCKER_USERNAME }
          password: ${ secrets.DOCKER_PASSWORD }

      - name: Docker Build
```

```

    run: /
      docker build -t ${{ secrets.DOCKER_USERNAME }}/nginx-
assignment:${{ github.sha }} .
      docker push ${{ secrets.DOCKER_USERNAME }}/nginx-
assignment:${{ github.sha }}

- name: Infra Checkout
  uses: actions/checkout@v4
  with:
    repository: "jainil15/${{ secrets.INFRA_REPO }}"

- name: Terraform setup
  uses: hashicorp/setup-terraform@v3
  with:
    cli_config_credentials_token: ${{ secrets.TF_API_TOKEN }}

- name: Configure AWS Credentials
  uses: aws-actions/configure-aws-credentials@v4.0.2
  with:
    aws-access-key-id: ${{ secrets.AWS_ACCESS_KEY_ID }}
    aws-secret-access-key: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
  }}

    aws-region: ${{ secrets.AWS_REGION }}

- name: Get MY IP
  id: get_my_ip
  run: echo "my_ip=$(curl https://ipv4.icanhazip.com/)" >>
$GITHUB_OUTPUT

- name: Terraform Init
  run: terraform init

- name: Terraform Plan
  env:
    TF_VAR_private_key: ${{ secrets.AWS_KEYPAIR }}
    TF_VAR_aws_region: ${{ secrets.AWS_REGION }}
    TF_VAR_my_ip: ${{ steps.get_my_ip.outputs.my_ip }}

  run: terraform plan -no-color

- name: Terraform Apply
  env:
    TF_VAR_private_key: ${{ secrets.AWS_KEYPAIR }}
    TF_VAR_aws_region: ${{ secrets.AWS_REGION }}

```

```

        TF_VAR_my_ip: ${ steps.get_my_ip.outputs.my_ip }}
    run: terraform apply -auto-approve

- name: Terraform output
  id: terraform_output
  run: /
    echo "public_instance_ip=$(terraform output
public_instance_ip | sed -e 's/^"//' -e 's/"$//')" >> $GITHUB_OUTPUT

- name: Wait for Port Open
  run: /
    while ! nc -zv ${ steps.terraform_output.outputs.public_instance_ip }} 22; do
      echo "Waiting for port 22 on host ${ steps.terraform_output.outputs.public_instance_ip }} to be open..."
      sleep 10
    done
    echo "Port 22 on host ${ steps.terraform_output.outputs.public_instance_ip }} is now open."

- name: SSH to AWS Instance and Pull Image
  uses: appleboy/ssh-action@master
  with:
    host: "${ steps.terraform_output.outputs.public_instance_ip }}"
    username: ${ secrets.AWS_USERNAME }}
    key: ${ secrets.AWS_KEYPAIR }}
    script: /
      while ! dpkg -l | grep -q docker; do
        echo "Docker is not installed yet. Waiting..."
        sleep 10
      done
      echo "Docker is installed"
      sleep 10
      sudo systemctl start docker
      sudo systemctl enable docker
      sudo docker stop nginx_container
      sudo docker rm nginx_container
      sudo docker run -d -p 80:80 --restart unless-stopped --
name nginx_container ${ secrets.DOCKER_USERNAME }}/nginx-
assignment:${ secrets.github.sha }}

```

2. Now lets setup terraform-destroy.yaml (for destroying)

```
name: "Terraform destroy"
```



```

on:
  workflow_dispatch:

jobs:
  terraform:
    runs-on: ubuntu-latest

    steps:
      - name: Infra Checkout
        uses: actions/checkout@v4
        with:
          repository: "jainil15/${{ secrets.INFRA_REPO }}"
      - name: Terraform setup
        uses: hashicorp/setup-terraform@v3
        with:
          cli_config_credentials_token: "${{ secrets.TF_API_TOKEN }}"

      - name: Configure AWS Credentials
        uses: aws-actions/configure-aws-credentials@v4.0.2
        with:
          aws-access-key-id: "${{ secrets.AWS_ACCESS_KEY_ID }}"
          aws-secret-access-key: "${{ secrets.AWS_SECRET_ACCESS_KEY }}"
          aws-region: "${{ secrets.AWS_REGION }}"

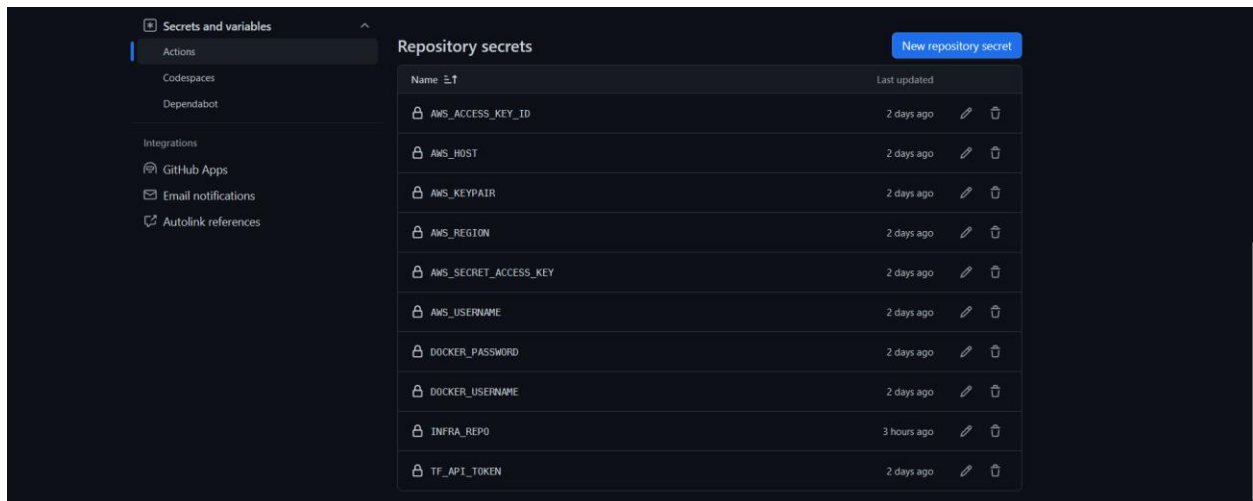
      - name: Get MY IP
        id: get_my_ip
        run: echo "my_ip=$(curl https://ipv4.icanhazip.com/)" >>
$GITHUB_OUTPUT

      - name: Terraform Init
        run: terraform init

      - name: Terraform Destroy
        env:
          TF_VAR_private_key: "${{ secrets.AWS_KEYPAIR }}"
          TF_VAR_aws_region: "${{ secrets.AWS_REGION }}"
          TF_VAR_my_ip: "${{ steps.get_my_ip.outputs.my_ip }}"
        run: terraform destroy -auto-approve

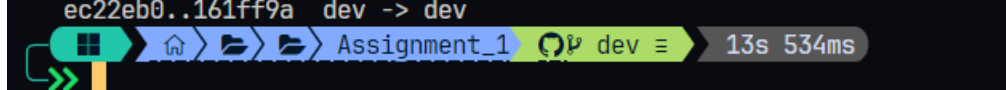
```

Now add the required secrets:



Now let create a new branch dev and push these changes.

```
> git commit -am "Final Commit"
[dev 161ff9a] Final Commit
 1 file changed, 1 insertion(+), 1 deletion(-)
> git push origin dev
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 295 bytes | 295.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/jainil15/CI_CD_Assignment_1.git
ec22eb0..161ff9a dev -> dev
```



After commit:

Summary

Jobs

Run details

Usage

Workflow file

terraform

Started 28s ago

Beta Give feedback Search logs

Terraform Init9s

30 Terraform has created a lock file .terraform.lock.hcl to record the provider selections it made above. Include this file in your version control repository so that Terraform can guarantee to make the same selections by default when you run "terraform init" in the future.

34 Terraform has been successfully initialized!

35

36 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.

39 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.

Terraform Plan2s

1 Run terraform plan --no-color

Terraform Apply

Terraform output

Wait for Port Open

SSH to AWS Instance and Pull Image

After complete:

Summary

Jobs

Run details

Usage

Workflow file

terraform

succeeded 1 minute ago in 2m 33s

Beta Give feedback Search logs

Post Infra Checkout0s

1 Post job cleanup.

2 /usr/bin/git version

3 git version 2.43.2

4 Temporarily overriding HOME='/home/runner/work/\_temp/442d8809-6fa2-4ca4-8f83-f2db3838b21b' before making global git config changes

5 Adding repository directory to the temporary git global config as a safe directory

6 /usr/bin/git config --global --add safe.directory /home/runner/work/CI\_CD\_Assignment\_1/CI\_CD\_Assignment\_1

7 /usr/bin/git config --local --name-only --get-regexp core.sshCommand

8 /usr/bin/git submodule foreach --recursive sh -c "git config --local --name-only --get-regexp 'core.sshCommand' && git config --local --unset-all 'core.sshCommand' || :"

9 /usr/bin/git config --local --name-only --get-regexp http.https://github.com/.extraheader

10 http.https://github.com/.extraheader

11 /usr/bin/git config --local --unset-all http.https://github.com/.extraheader

12 /usr/bin/git submodule foreach --recursive sh -c "git config --local --name-only --get-regexp 'http.https://github.com/.extraheader' && git config --local --unset-all 'http.https://github.com/.extraheader' || :"

Post Docker Login0s

1 Post job cleanup.

2 /usr/bin/docker logout

3 Removing login credentials for https://index.docker.io/v1/

Post Docker Checkout0s

Complete job0s

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

← Nginx Deployment

Final Commit #76

Re-run all jobs ...

Summary

Jobs

Run details

Usage

Workflow file

Triggered via push 4 minutes ago

Status

Total duration

Artifacts

jainil15 pushed • 161ffa dev

Success

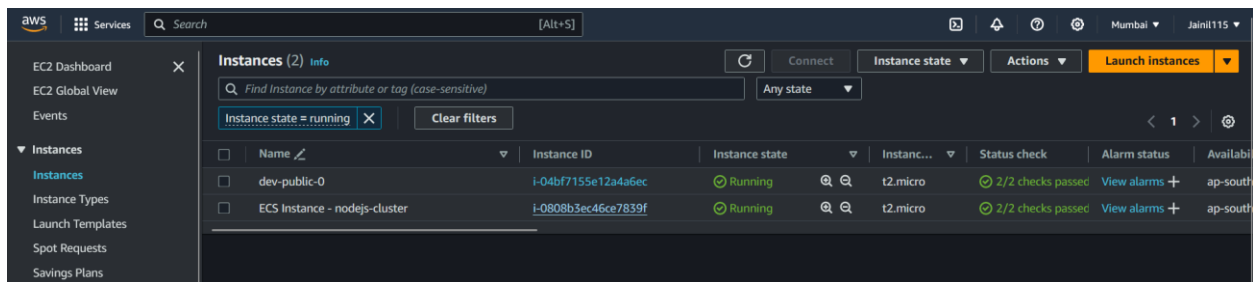
2m 40s

–

nginx-deploy.yaml

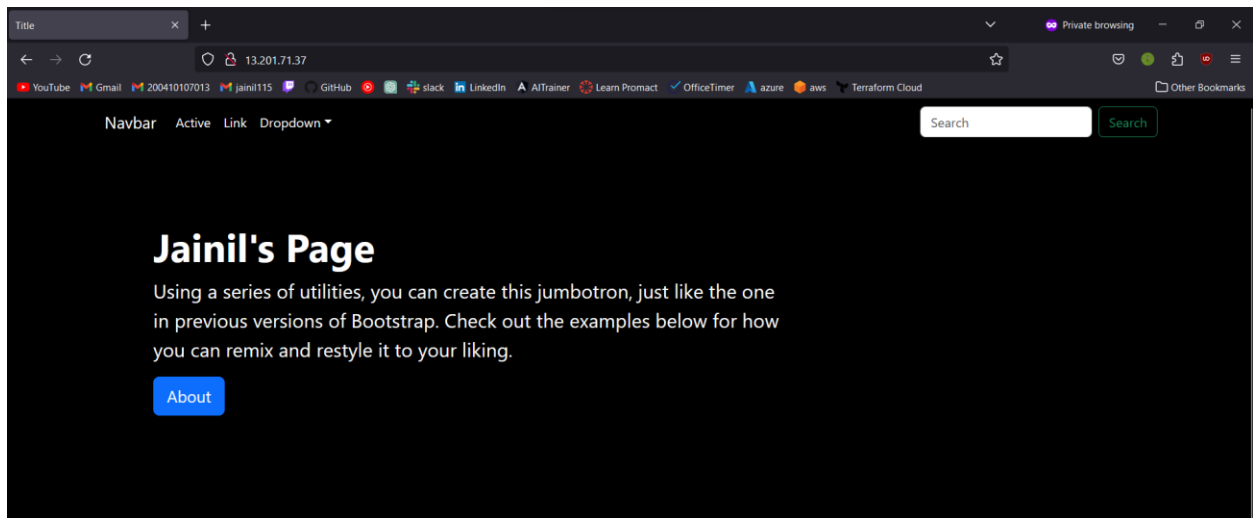
on: push

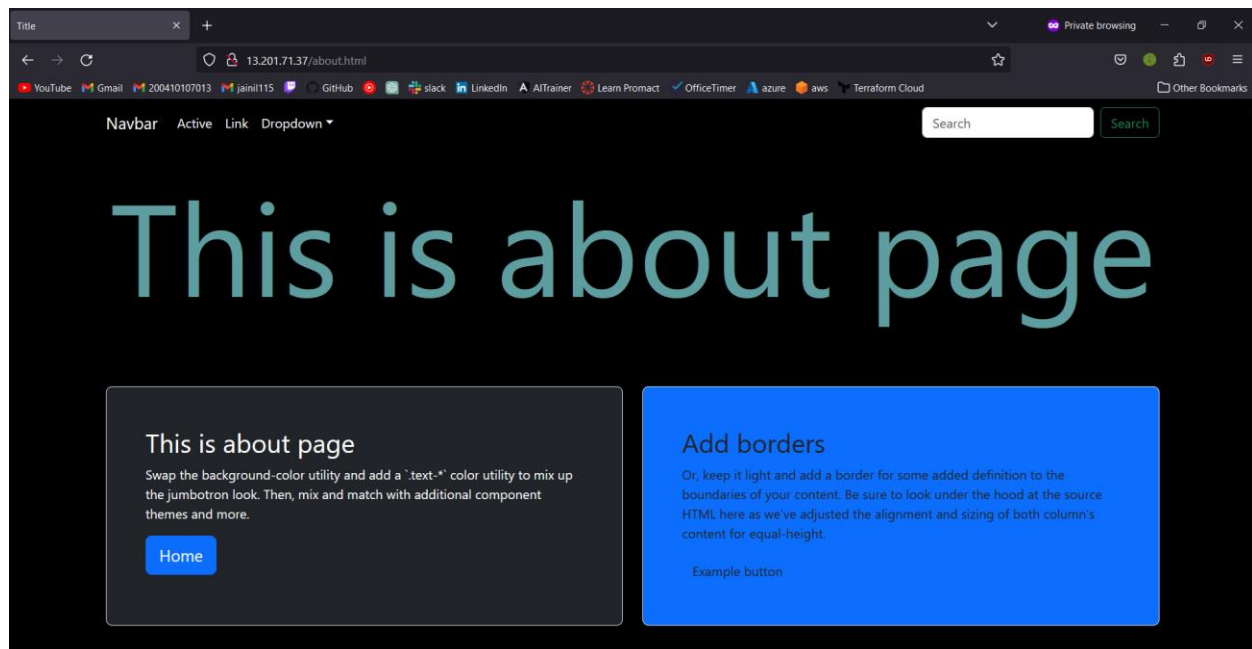
terraform2m 33s



Lets check that the website is working:

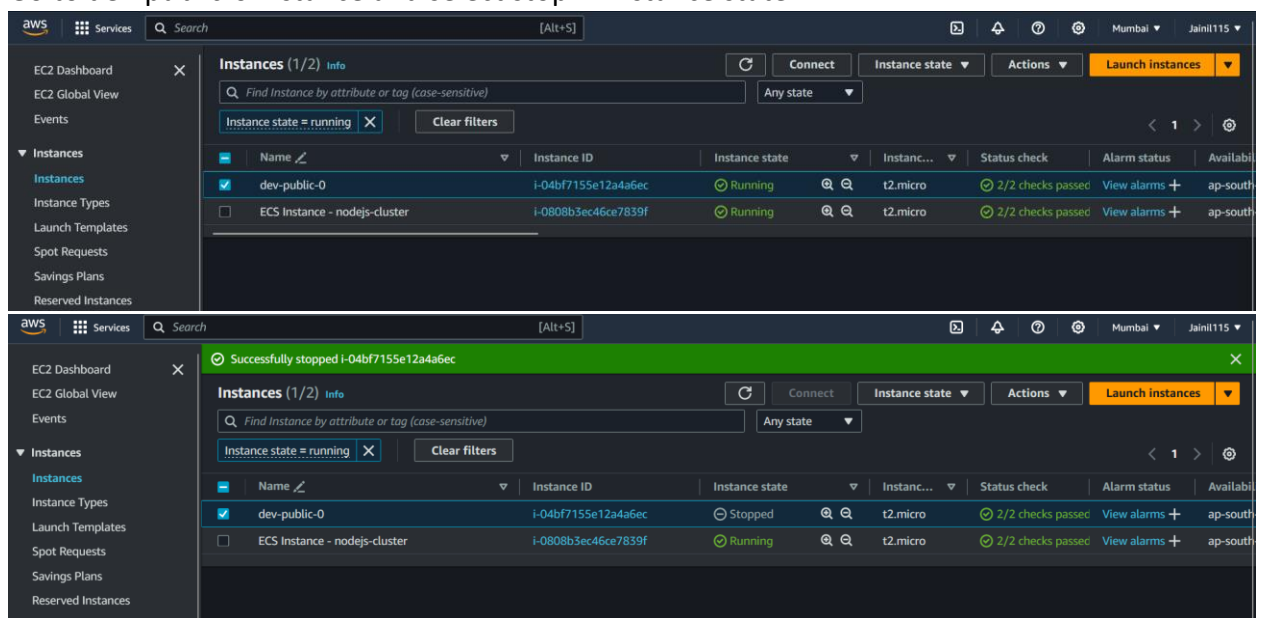
Enter the public ip in the browser, 13.201.71.37:



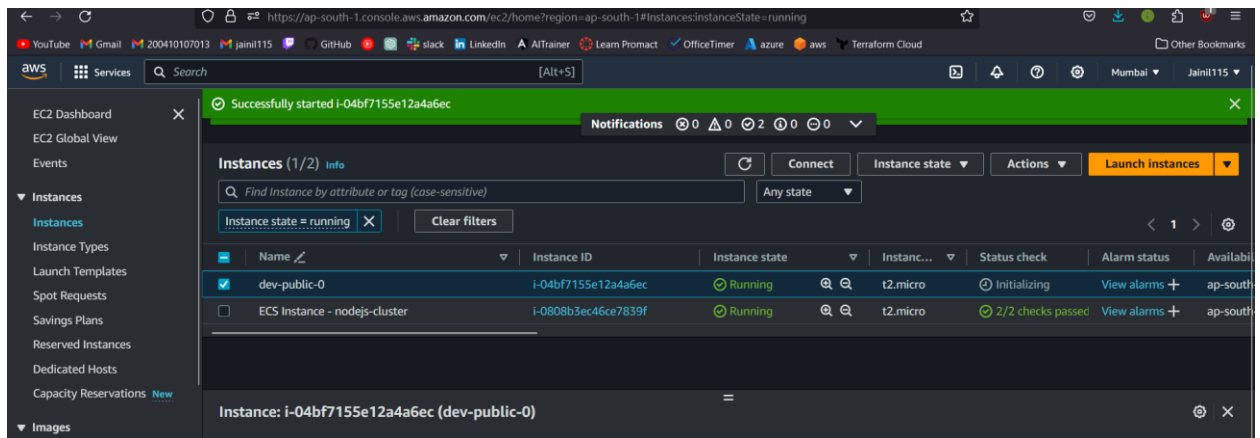


Now let's stop the page and test that after restarting does it work.

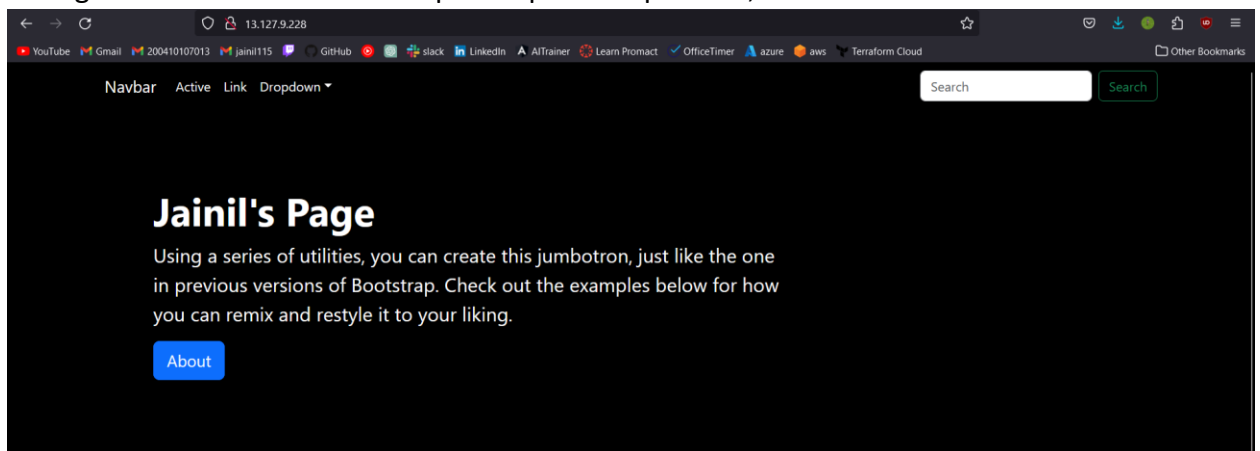
1. Go to dev-public-0 instance and select stop in instance state



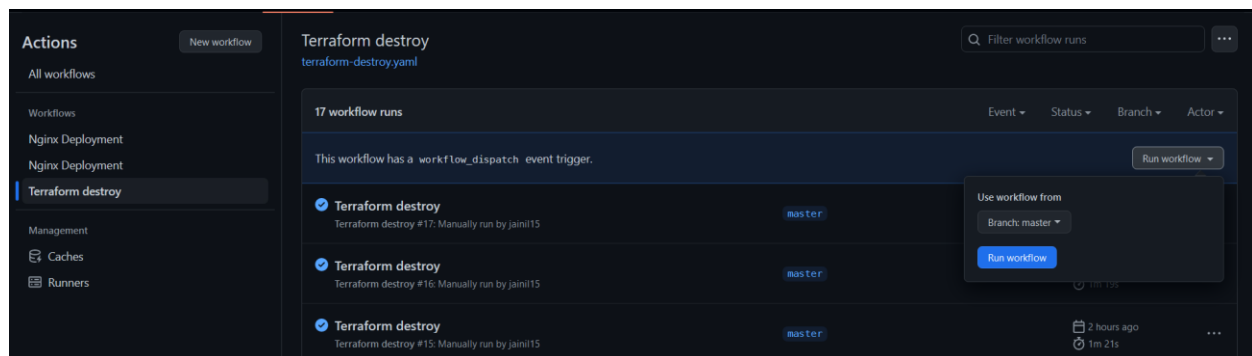
2. After it is stopped start the instance again by going to instance state and clicking on start



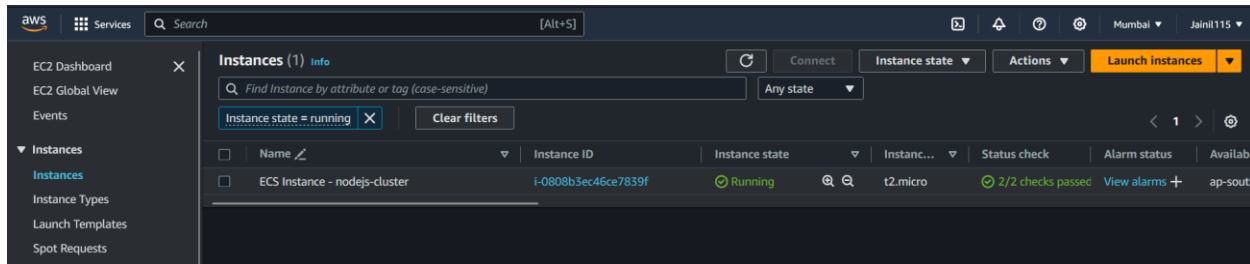
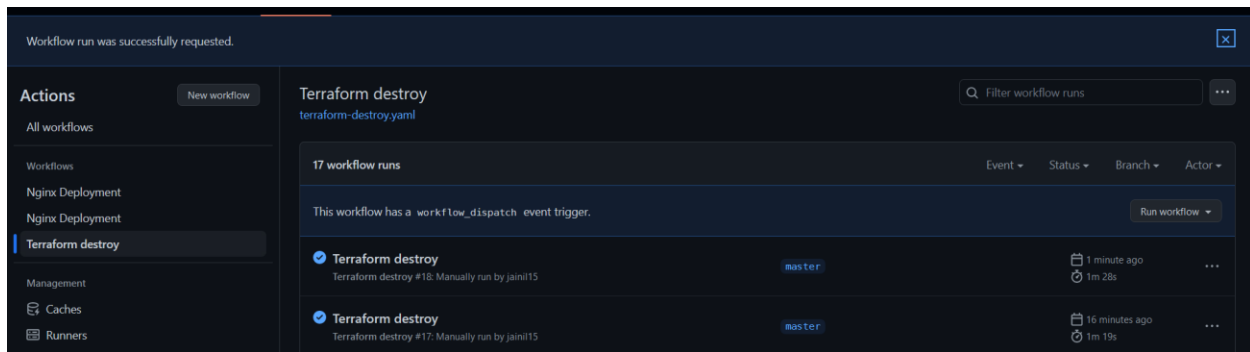
- Now go to browser and enter the public ip of dev-public-0, 13.127.9.228.



Now lets destroy these resources. Go to your github repo and go to actions and select terraform destroy action and click on run workflow:



After complete:



Now lets try pushing to a different branch and see if it is deploying.

Use the following command:

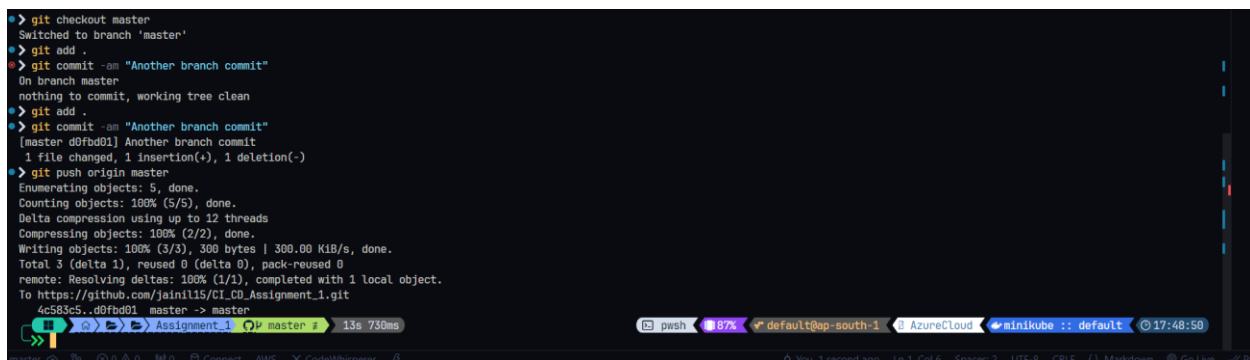
git branch master

git checkout master

git add .

git commit -m "Another branch commit"

git push origin master



jainil15 / CI\_CD\_Assignment\_1

Q Type [f] to search

+ -

n

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

CI\_CD\_Assignment\_1Public

PinUnwatch 1ForkStar 0

dev had recent pushes 16 minutes agoCompare & pull request

master2 Branches0 TagsGo to fileAdd fileCode

jainil15Another branch commitd0fb01 · 1 minute ago88 Commits

.github/workflows	Removed depracted outputs method	2 hours ago
app	Added commit hash as tag	16 hours ago
.gitignore	Initial Commit	2 days ago
Dockerfile	Added commit hash as tag	16 hours ago
README.md	Another branch commit	1 minute ago

About

No description, website, or topics provided.

ReadmeActivity0 stars1 watching0 forks

Releases

No releases published  
[Create a new release](#)

Packages