**Terraform Task-2**

**Task Description:**

Create 2 EC2 instances on 2 different regions and install nginx using terraform script.

**Techstacks needs to be used :**

* AWS EC2
* Terraform
* AWS CLI

**How do I submit my work?**

* Push all your work files to GitHub (Code & O/P screenshot images must).
* Submit your URLs in the portal.

**Terms and Conditions?**

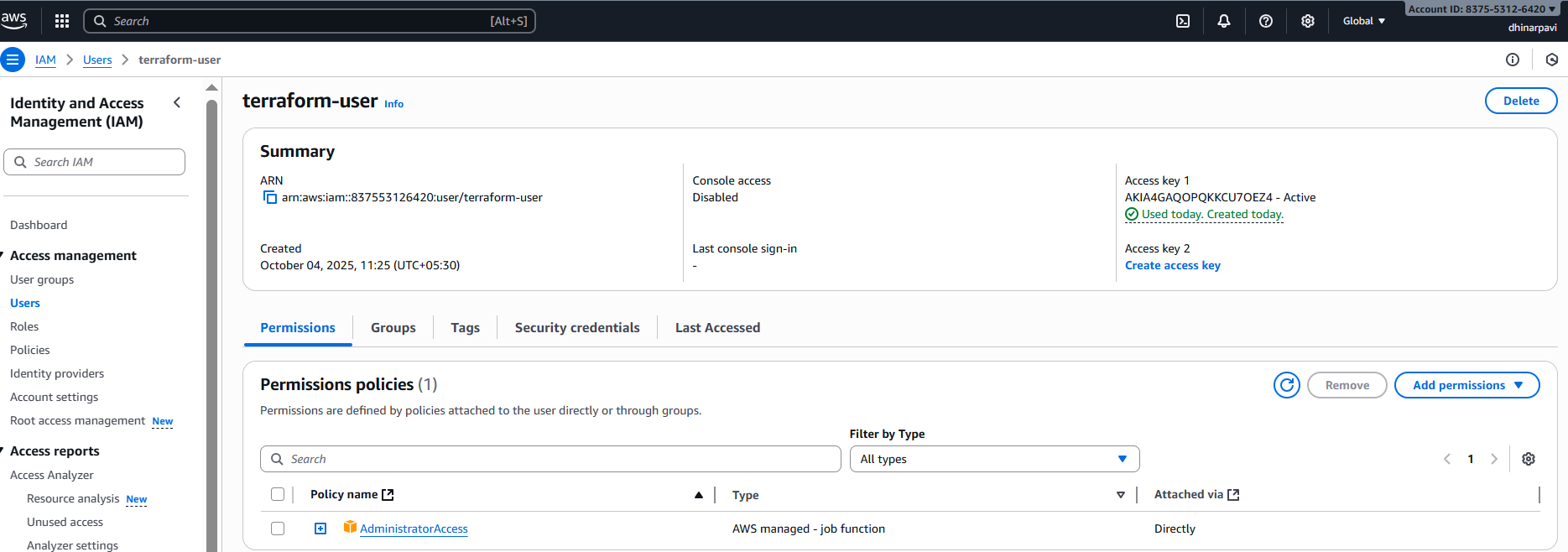
* You agree to not share this confidential document with anyone.
* You agree to open-source your code (it may even look good on your profile!). Do not mention our company’s name anywhere in the code.
* We will never use your source code under any circumstances for any commercial purposes; this is just a basic assessment task.

**NOTE:** Any violation of Terms and conditions is strictly prohibited. You are bound to adhere to it.

**Solution:**

**AWS IAM Access Key Creation**

Created a dedicated IAM user named terraform-user in AWS IAM and generated an Access Key and Secret Access Key for Terraform authentication.

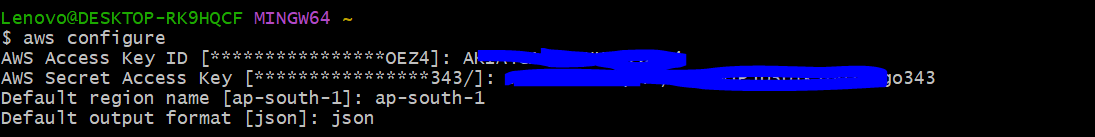


**AWS CLI Configuration**

Configured AWS CLI using the generated access keys to enable Terraform authentication with my AWS account.

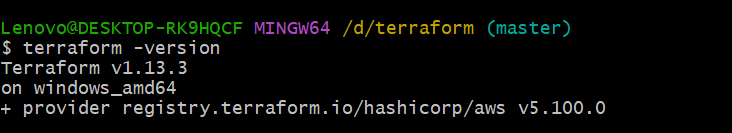
Code/Command:

aws configure



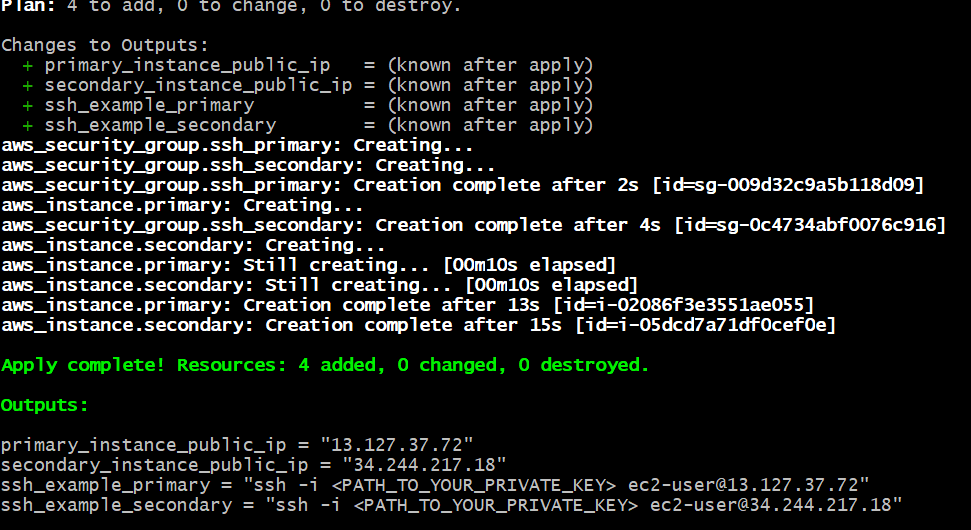
**Terraform Initialization**

Initialized Terraform in the project directory to download the AWS provider and prepare the working directory.



**Terraform Apply (Resource Creation)**

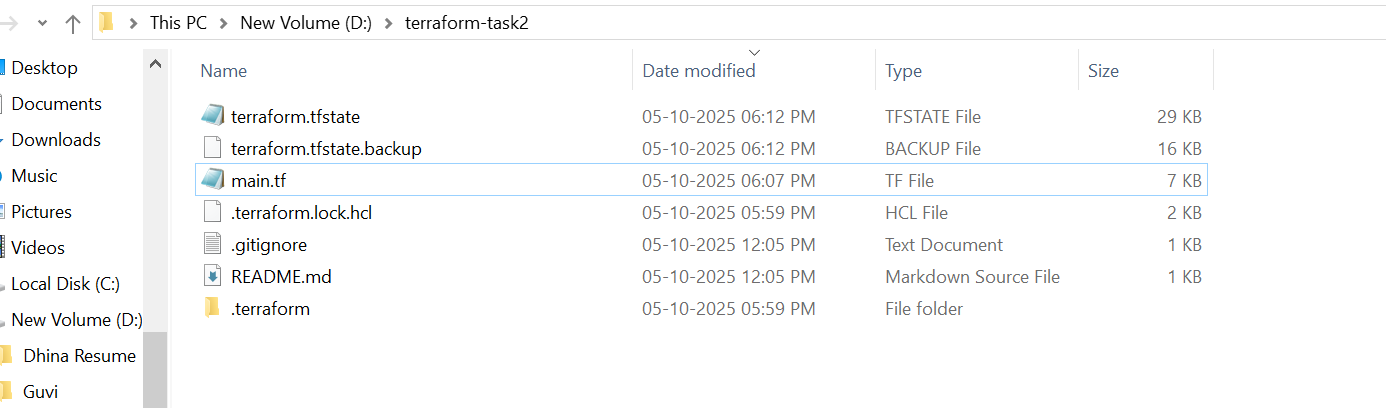
Executed terraform apply to provision two Amazon Linux EC2 instances in two different AWS regions (ap-south-1 and us-east-1).



**AWS Console – EC2 Instances**

Verified both EC2 instances running successfully in the AWS Management Console under EC2 service.

Main.tf



# Terraform Task-2: Two EC2s in Two Regions + NGINX

# v2: regional key pairs + t3.micro default

terraform {

required\_version = ">= 1.4.0"

required\_providers {

aws = {

source = "hashicorp/aws"

version = "~> 5.0"

}

}

}

# VARIABLES

variable "primary\_region" {

description = "Primary AWS region (e.g., ap-south-1)."

type = string

default = "ap-south-1"

}

variable "secondary\_region" {

description = "Secondary AWS region (e.g., us-east-1)."

type = string

default = "us-east-1"

}

variable "instance\_type" {

description = "EC2 instance type."

type = string

default = "t3.micro"

}

variable "ssh\_ingress\_cidr" {

description = "CIDR allowed for SSH (port 22). Prefer YOUR.IP/32."

type = string

default = "0.0.0.0/0"

}

variable "http\_ingress\_cidr" {

description = "CIDR allowed for HTTP (port 80)."

type = string

default = "0.0.0.0/0"

}

variable "public\_key" {

description = "Your SSH PUBLIC key content to create AWS key pairs (leave empty to skip)."

type = string

default = ""

}

# PROVIDERS

provider "aws" {

region = var.primary\_region

}

provider "aws" {

alias = "secondary"

region = var.secondary\_region

}

# LOCALS

locals {

common\_tags = {

Project = "Guvi\_Terraform\_Task-2"

ManagedBy = "Terraform"

Environment = "dev"

}

# User-data to install and start NGINX (Amazon Linux 2023 uses dnf)

user\_data\_primary = <<-EOP

#!/bin/bash

set -e

dnf update -y

dnf install -y nginx

echo "<h1>NGINX up in ${var.primary\_region}</h1><p>Host: $(hostname)</p>" > /usr/share/nginx/html/index.html

systemctl enable nginx

systemctl start nginx

EOP

user\_data\_secondary = <<-EOS

#!/bin/bash

set -e

dnf update -y

dnf install -y nginx

echo "<h1>NGINX up in ${var.secondary\_region}</h1><p>Host: $(hostname)</p>" > /usr/share/nginx/html/index.html

systemctl enable nginx

systemctl start nginx

EOS

}

# AMI LOOKUPS

data "aws\_ami" "al2023\_primary" {

owners = ["amazon"]

most\_recent = true

filter {

name = "name"

values = ["al2023-ami-\*-x86\_64"]

}

}

data "aws\_ami" "al2023\_secondary" {

provider = aws.secondary

owners = ["amazon"]

most\_recent = true

filter {

name = "name"

values = ["al2023-ami-\*-x86\_64"]

}

}

# DEFAULT VPCs

data "aws\_vpc" "primary\_default" {

default = true

}

data "aws\_vpc" "secondary\_default" {

provider = aws.secondary

default = true

}

# SECURITY GROUPS

resource "aws\_security\_group" "web\_primary" {

name = "tf2-web-primary"

description = "Allow SSH (22) + HTTP (80)"

vpc\_id = data.aws\_vpc.primary\_default.id

ingress {

description = "SSH"

from\_port = 22

to\_port = 22

protocol = "tcp"

cidr\_blocks = [var.ssh\_ingress\_cidr]

}

ingress {

description = "HTTP"

from\_port = 80

to\_port = 80

protocol = "tcp"

cidr\_blocks = [var.http\_ingress\_cidr]

}

egress {

from\_port = 0

to\_port = 0

protocol = "-1"

cidr\_blocks = ["0.0.0.0/0"]

}

tags = merge(local.common\_tags, { Name = "tf2-web-primary" })

}

resource "aws\_security\_group" "web\_secondary" {

provider = aws.secondary

name = "tf2-web-secondary"

description = "Allow SSH (22) + HTTP (80)"

vpc\_id = data.aws\_vpc.secondary\_default.id

ingress {

description = "SSH"

from\_port = 22

to\_port = 22

protocol = "tcp"

cidr\_blocks = [var.ssh\_ingress\_cidr]

}

ingress {

description = "HTTP"

from\_port = 80

to\_port = 80

protocol = "tcp"

cidr\_blocks = [var.http\_ingress\_cidr]

}

egress {

from\_port = 0

to\_port = 0

protocol = "-1"

cidr\_blocks = ["0.0.0.0/0"]

}

tags = merge(local.common\_tags, { Name = "tf2-web-secondary" })

}

# REGIONAL KEY PAIRS (optional)

# Create in PRIMARY region

resource "aws\_key\_pair" "primary" {

count = var.public\_key != "" ? 1 : 0

key\_name = "tf2-key-primary"

public\_key = var.public\_key

tags = merge(local.common\_tags, { Name = "tf2-key-primary" })

}

# Create in SECONDARY region

resource "aws\_key\_pair" "secondary" {

provider = aws.secondary

count = var.public\_key != "" ? 1 : 0

key\_name = "tf2-key-secondary"

public\_key = var.public\_key

tags = merge(local.common\_tags, { Name = "tf2-key-secondary" })

}

# INSTANCES

resource "aws\_instance" "primary" {

ami = data.aws\_ami.al2023\_primary.id

instance\_type = var.instance\_type

vpc\_security\_group\_ids = [aws\_security\_group.web\_primary.id]

key\_name = length(aws\_key\_pair.primary) > 0 ? aws\_key\_pair.primary[0].key\_name : null

user\_data = local.user\_data\_primary

tags = merge(local.common\_tags, {

Name = "tf2-ec2-primary"

Region = var.primary\_region

Role = "nginx-web"

})

}

resource "aws\_instance" "secondary" {

provider = aws.secondary

ami = data.aws\_ami.al2023\_secondary.id

instance\_type = var.instance\_type

vpc\_security\_group\_ids = [aws\_security\_group.web\_secondary.id]

key\_name = length(aws\_key\_pair.secondary) > 0 ? aws\_key\_pair.secondary[0].key\_name : null

user\_data = local.user\_data\_secondary

tags = merge(local.common\_tags, {

Name = "tf2-ec2-secondary"

Region = var.secondary\_region

Role = "nginx-web"

})

}

# OUTPUTS

output "primary\_public\_ip" {

value = aws\_instance.primary.public\_ip

}

output "secondary\_public\_ip" {

value = aws\_instance.secondary.public\_ip

}

output "primary\_http\_url" {

value = "http://${aws\_instance.primary.public\_ip}"

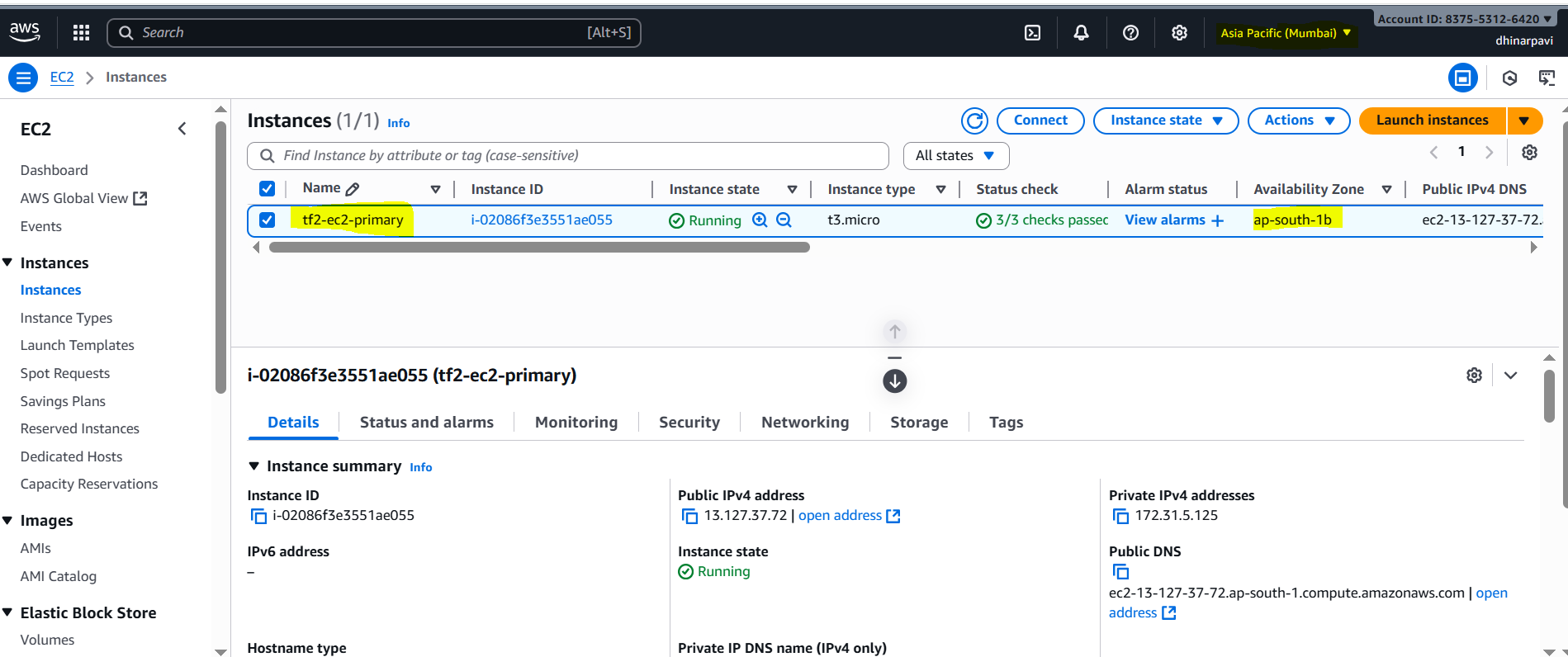
}

output "secondary\_http\_url" {

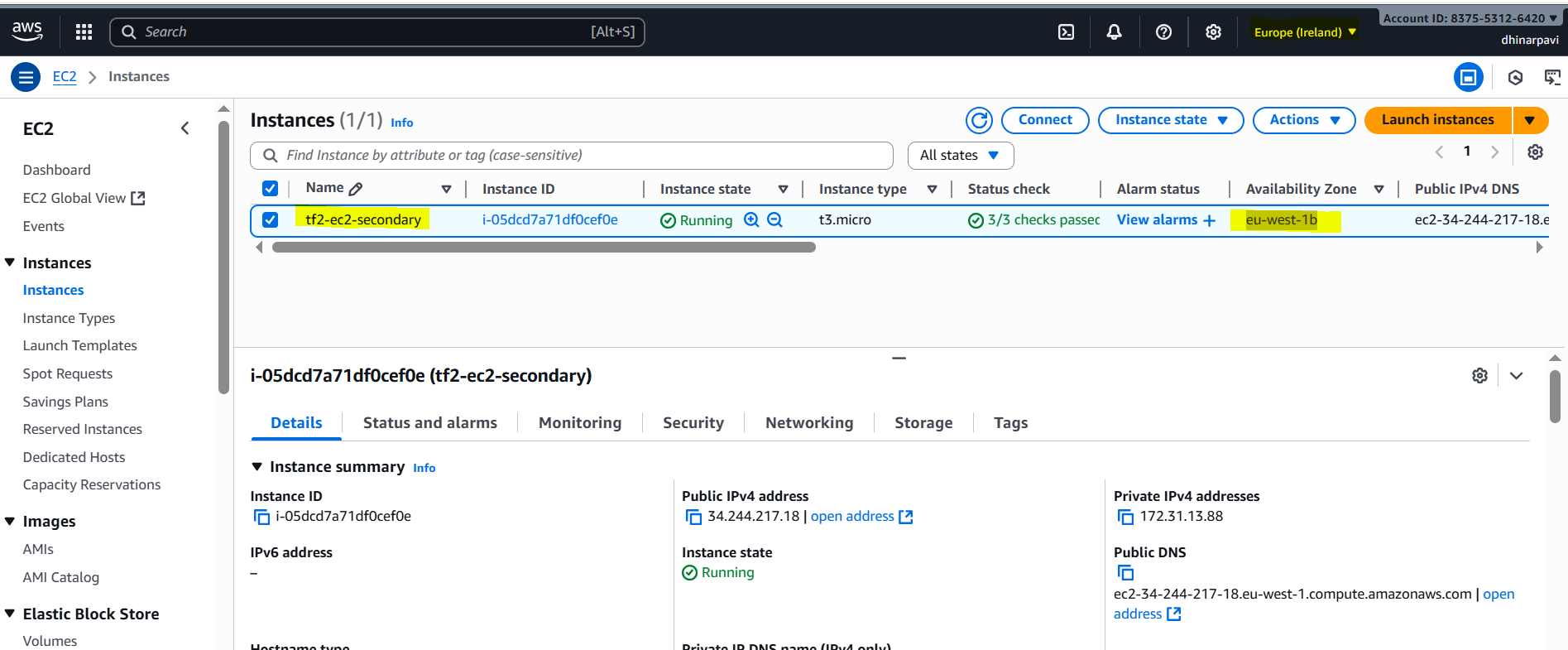
value = "http://${aws\_instance.secondary.public\_ip}"

}

**Region 1:** EC2 Dashboard → ap-south-1 (Mumbai)



**Region 2:** Switch region → eu-west-1 (Europe)

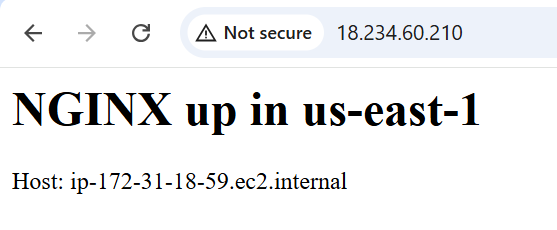


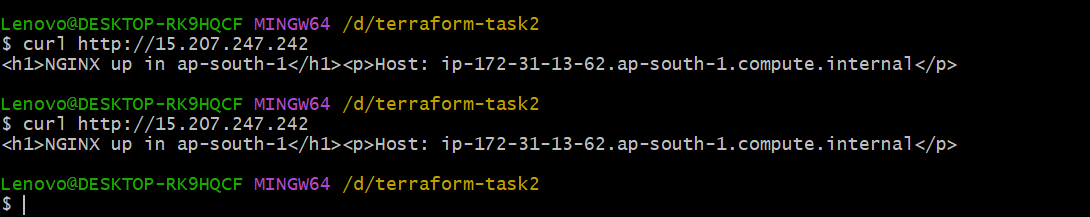
**SSH Login**

Successfully connected to the EC2 instance via SSH using the key pair created locally.

Console output







**Terraform repo in local**

