Terrform handson

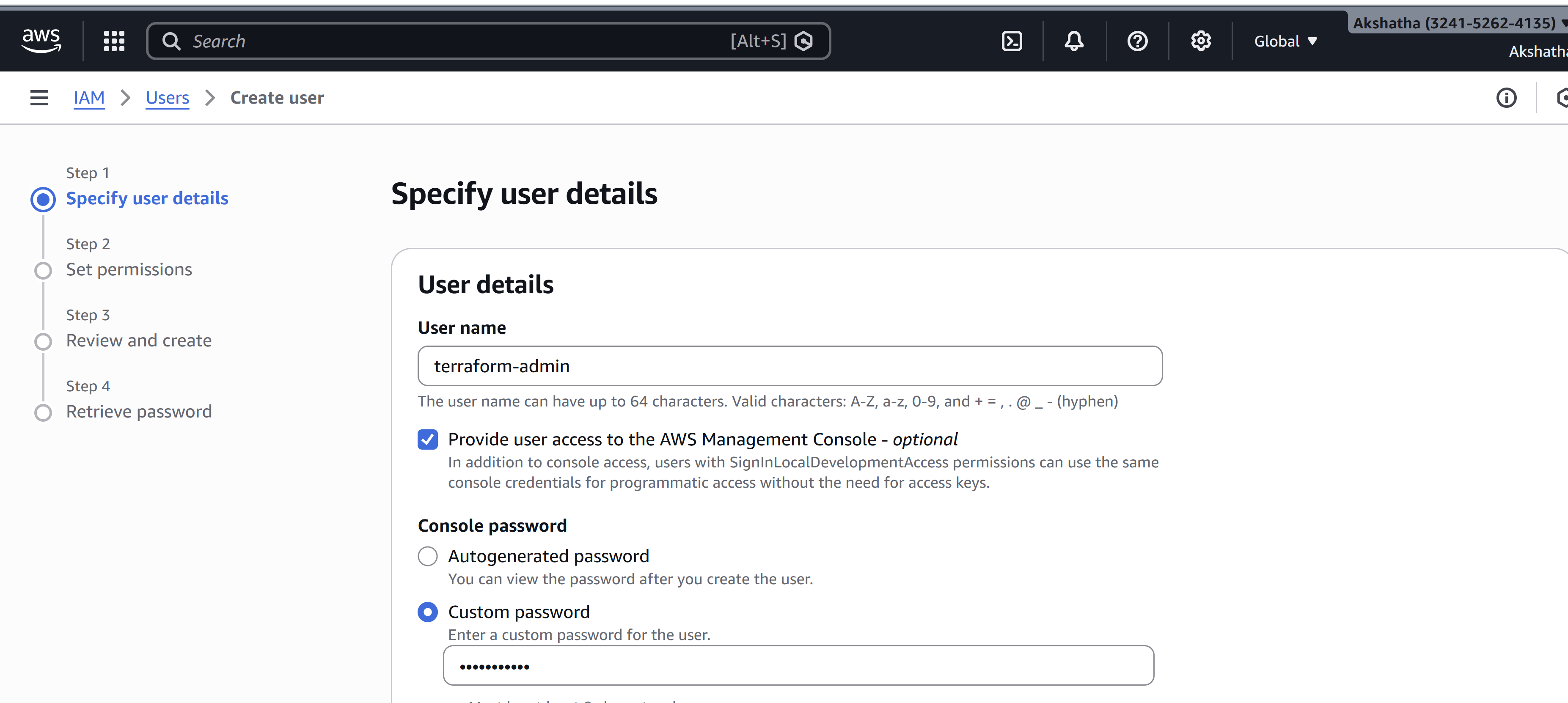
1. Create Terraform Infra to deploy Jenkins Server on AWS. Terraform Code must be modular and follow all best practices. The instance must be assigned a elastic IP and have scheduled ebs snapshots once per day.

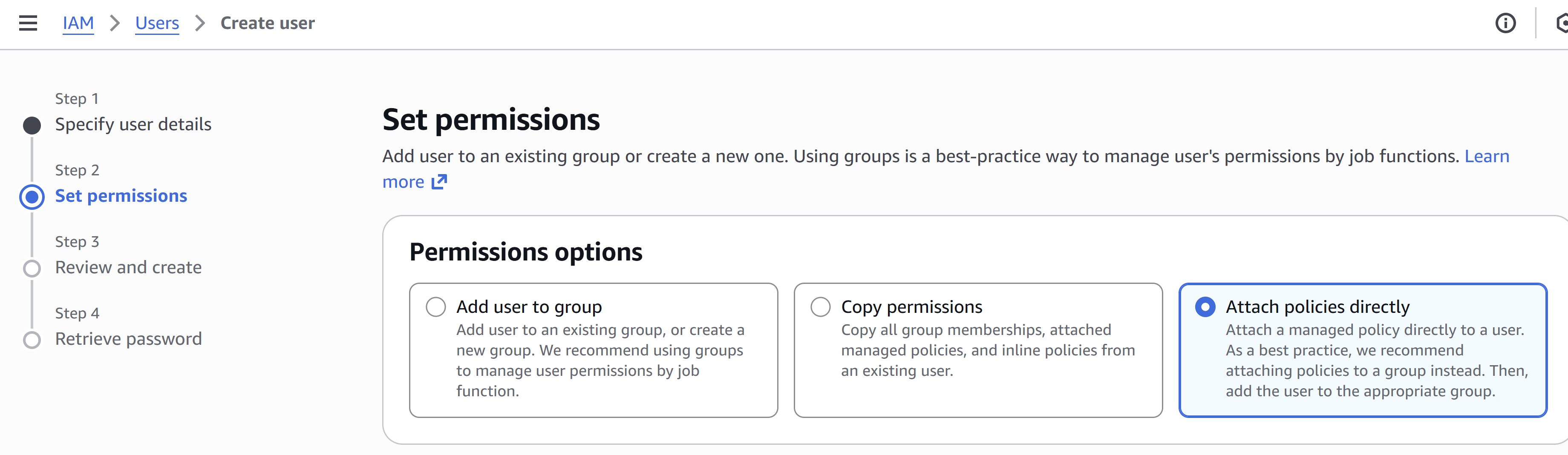
Jenkins must be deployed using docker compose, ensure data persistence for jenkins.

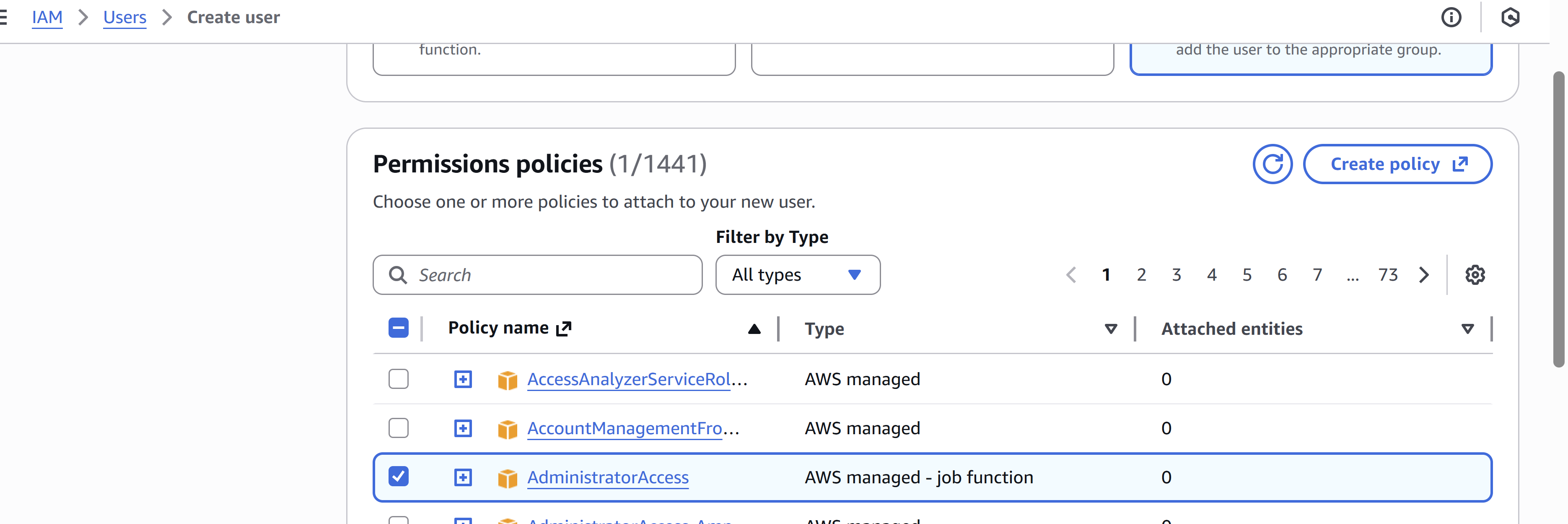
Maintain the docker compose in github repo and clone docker compose in the instance to launch Jenkins Server

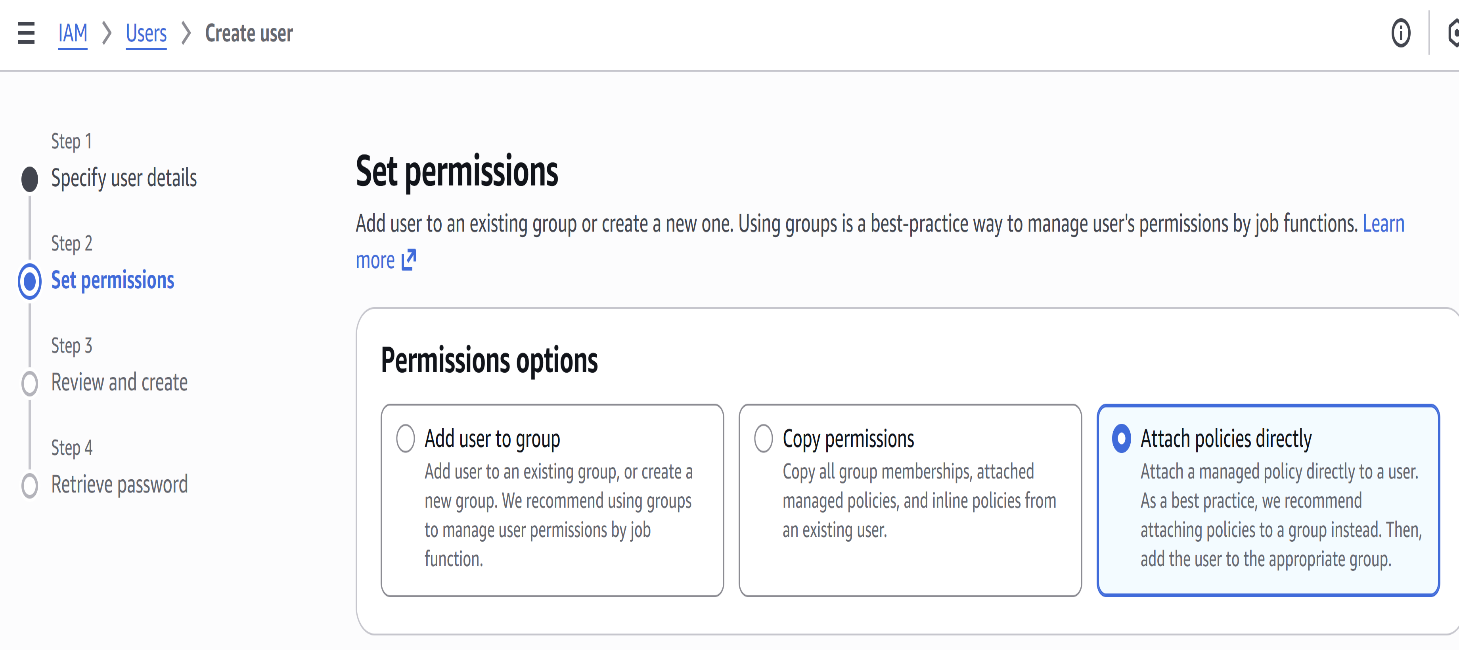
IAM creating in aws

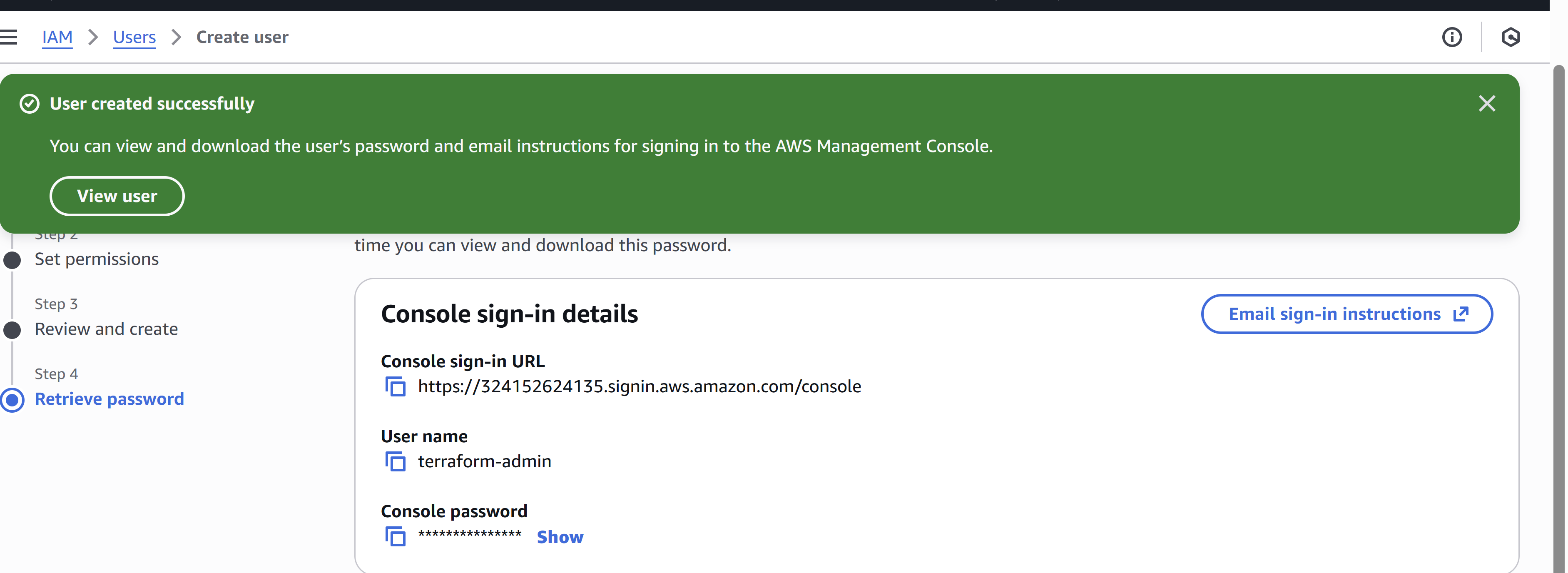
Steps 1 :





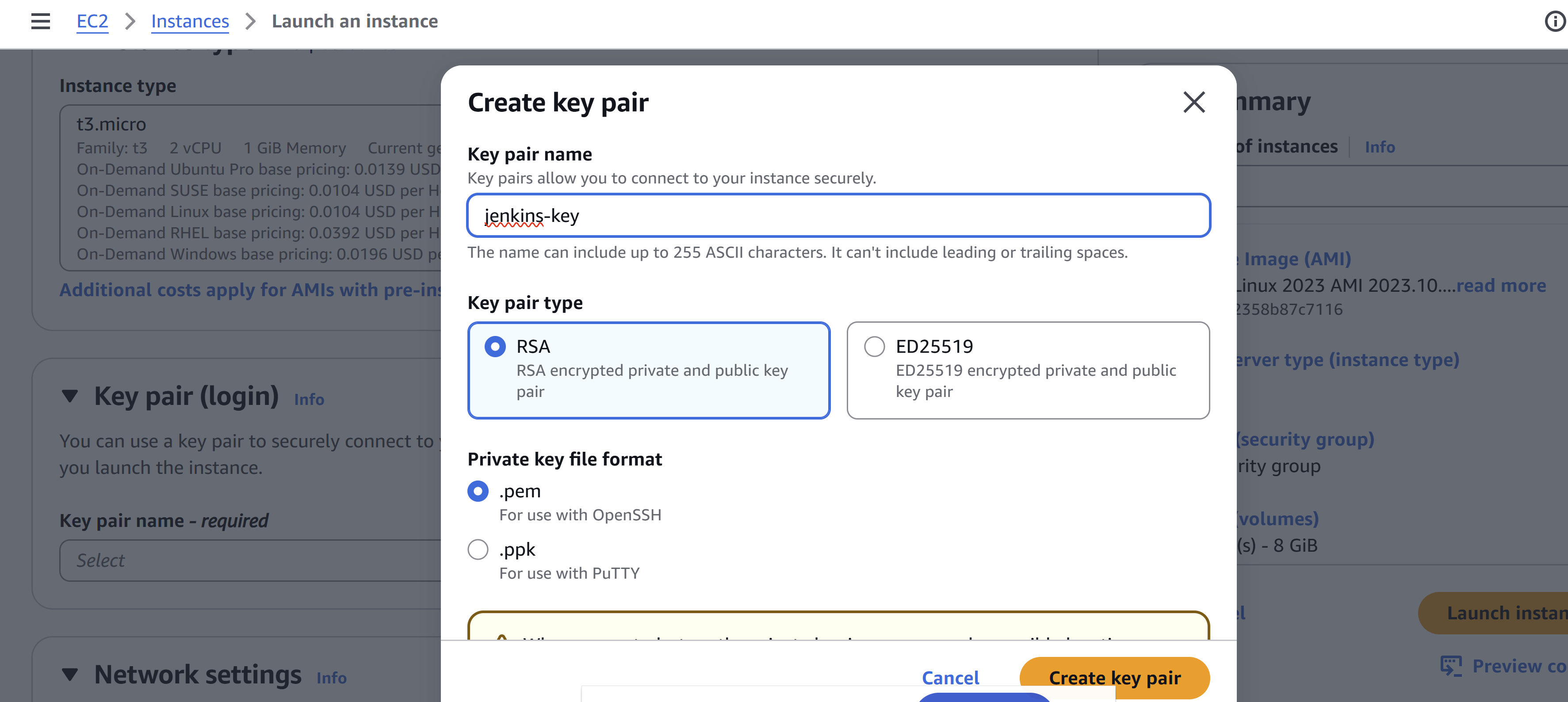


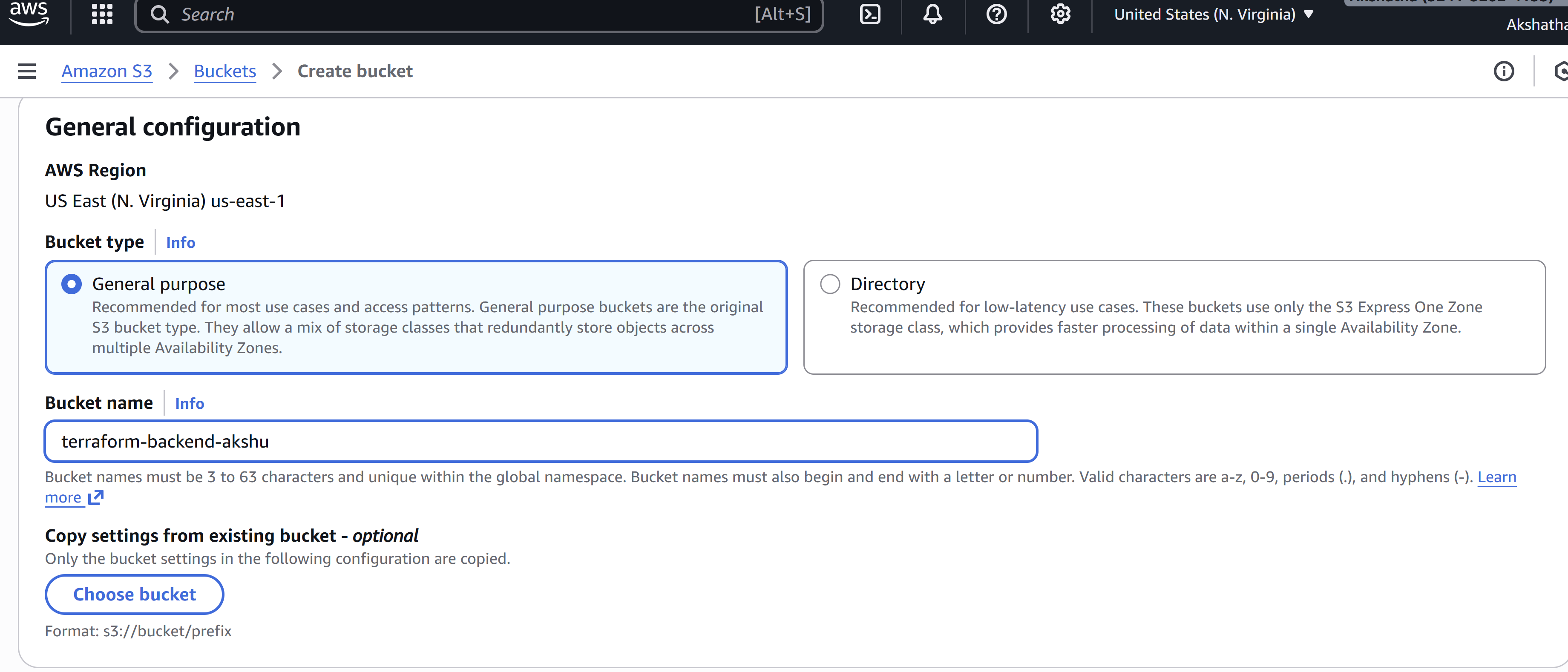


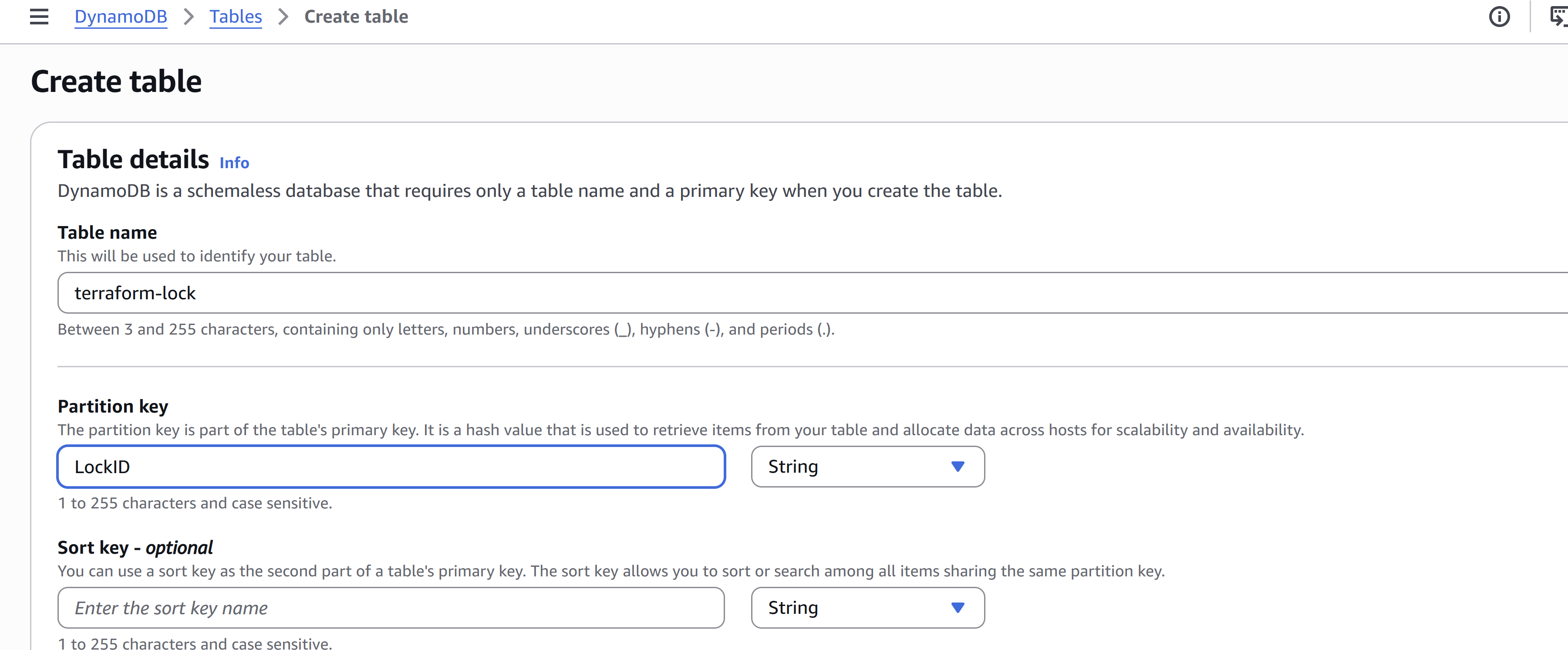




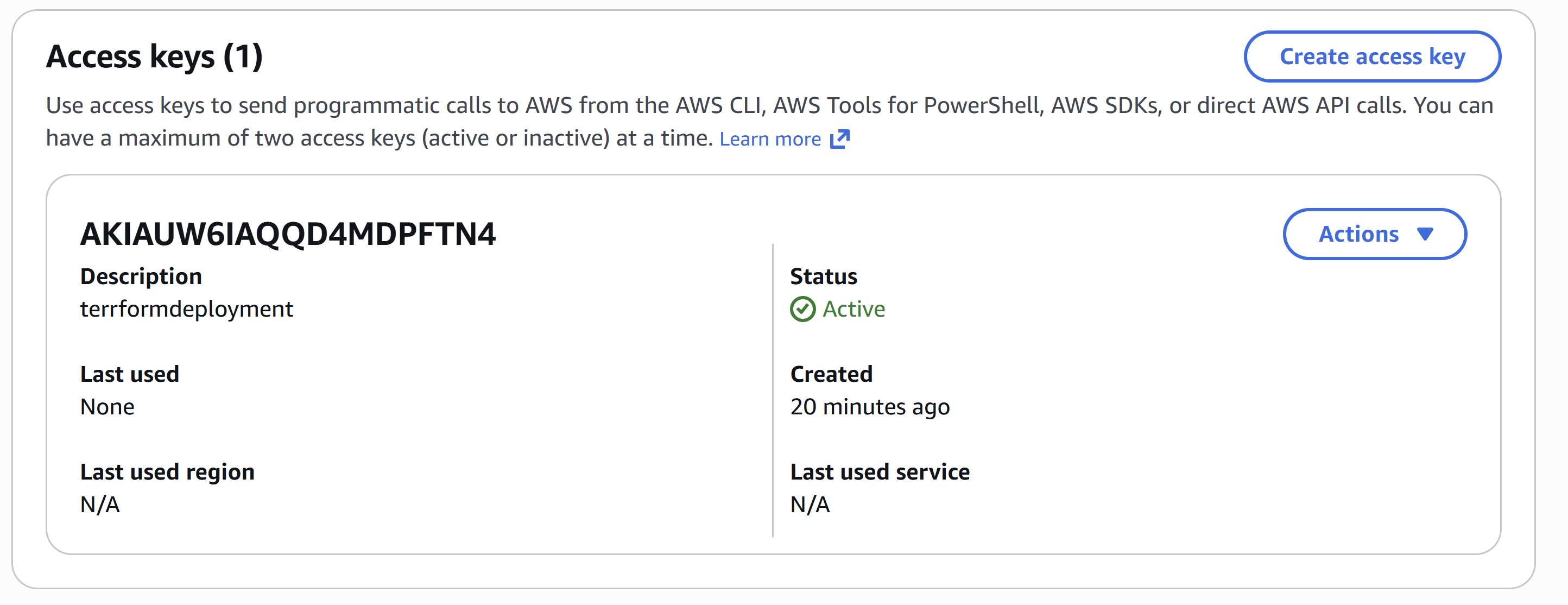
EC2 instance

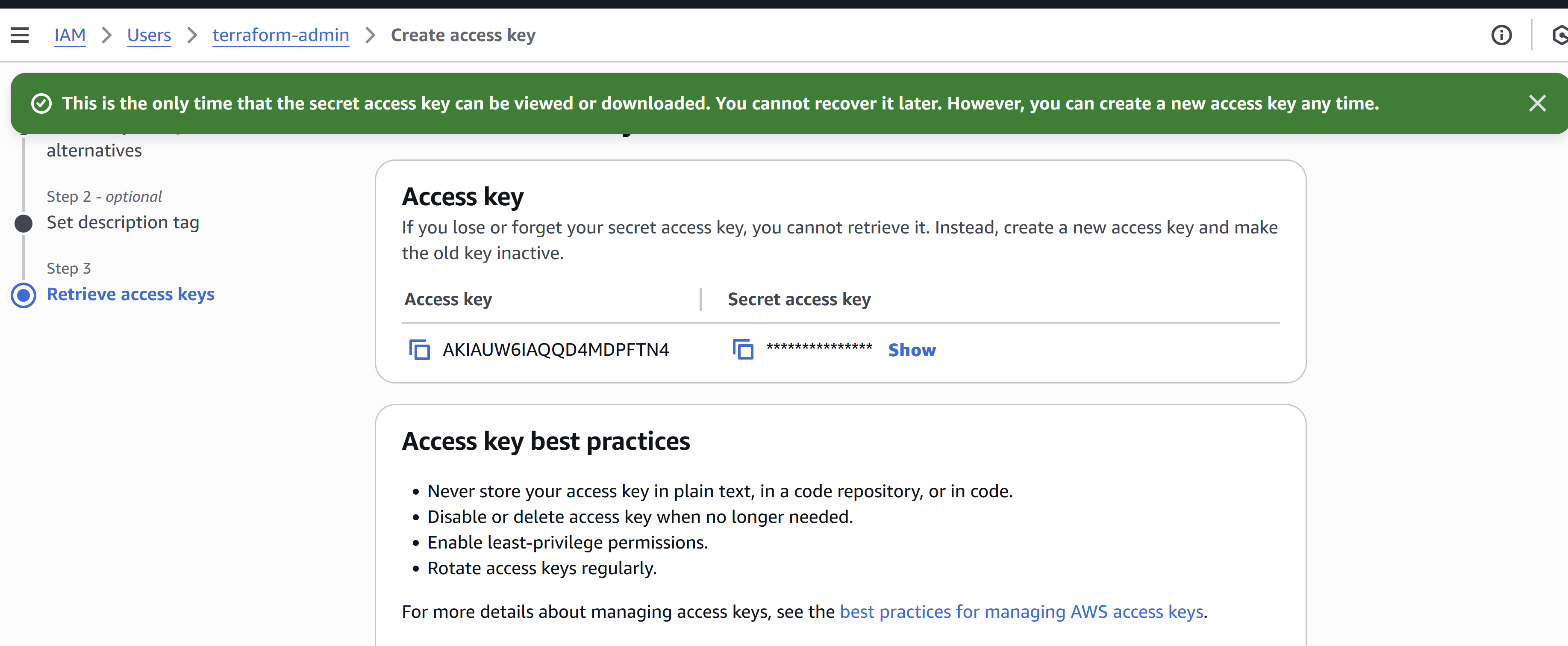


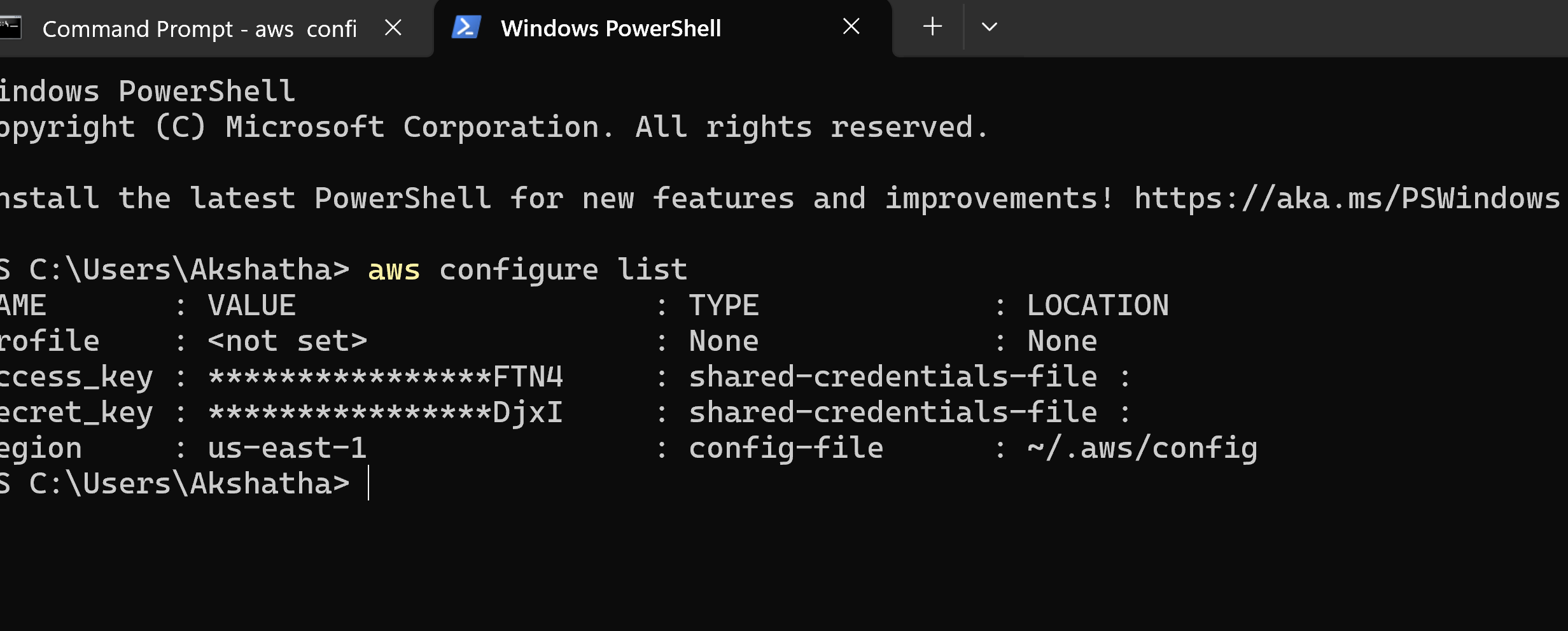


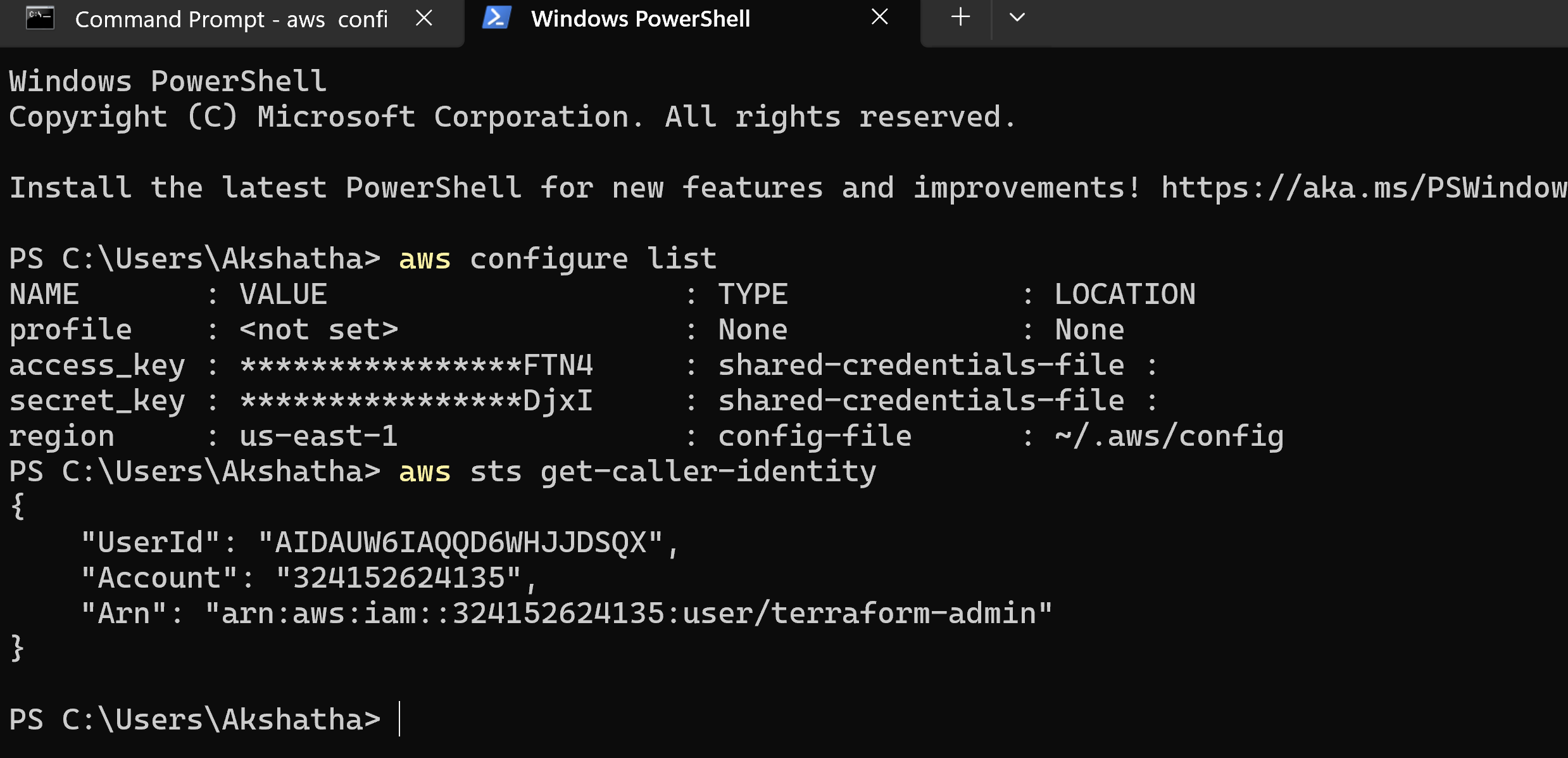


IAM creating access key



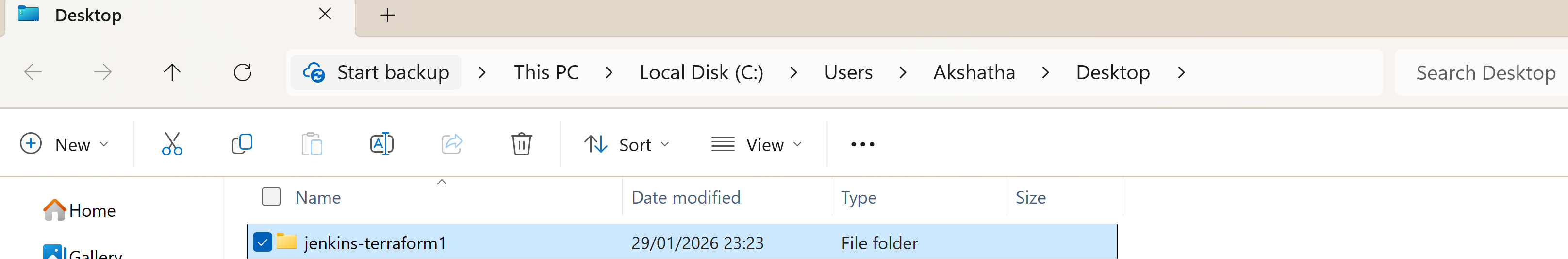


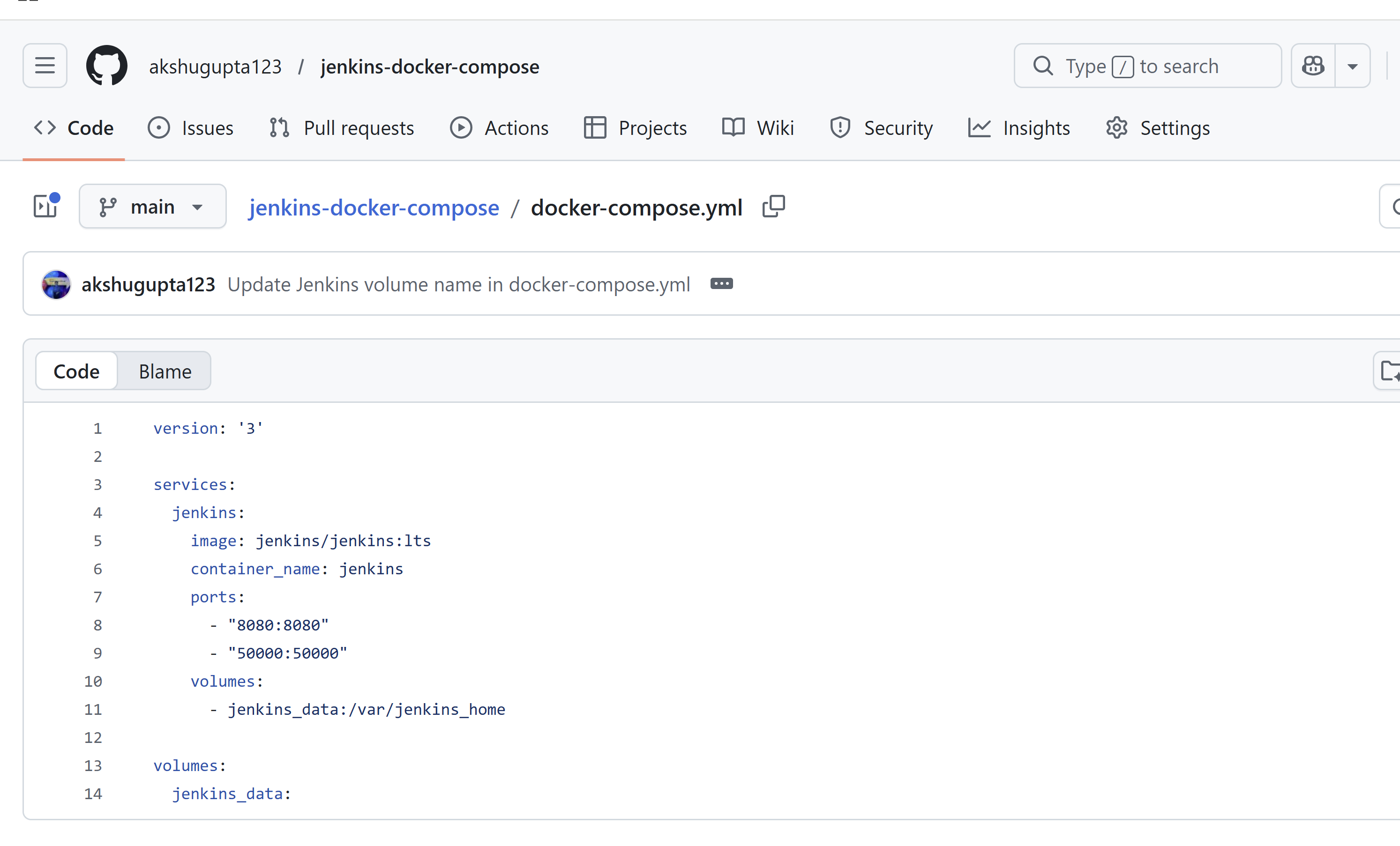




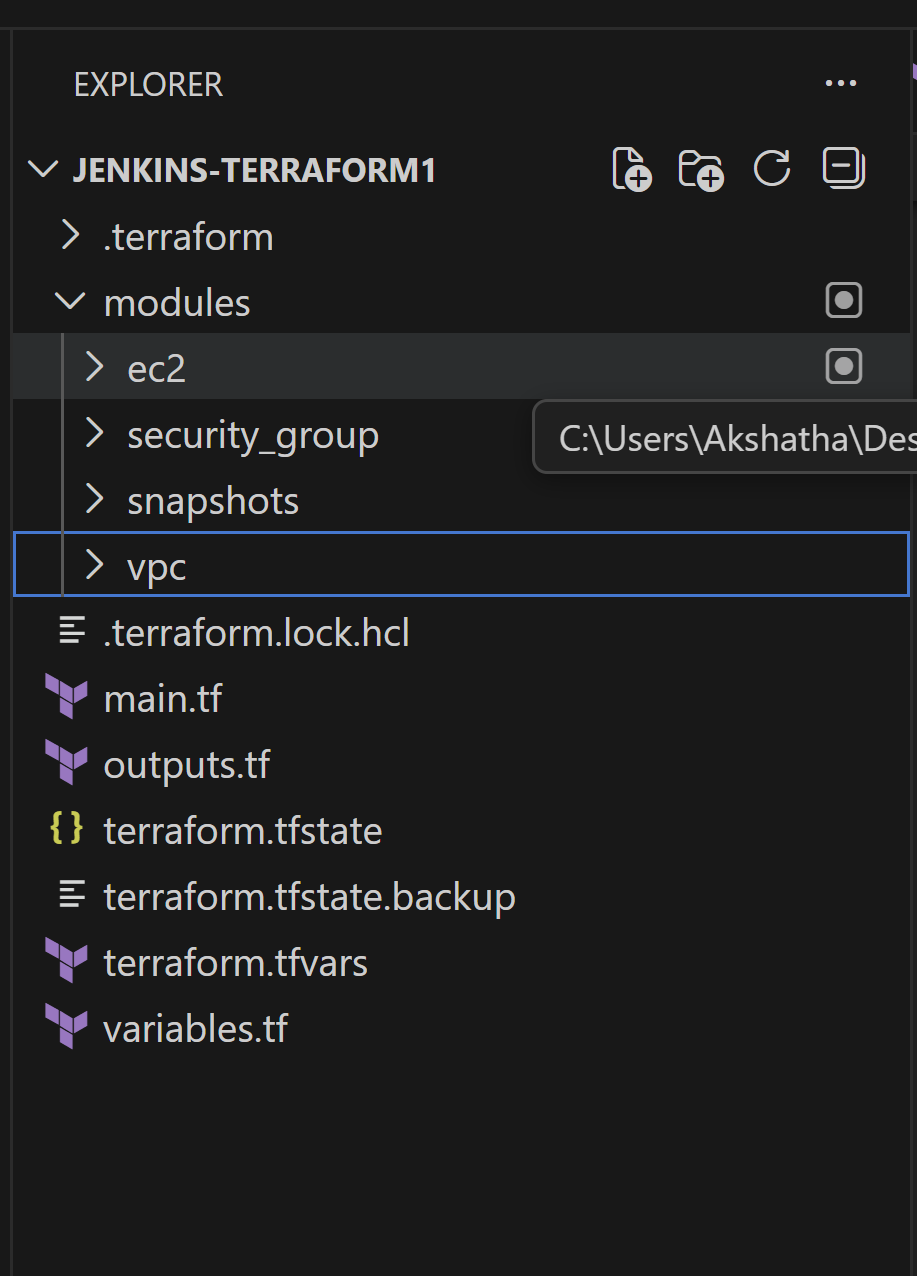
CREATE TERRAFORM PROJECT

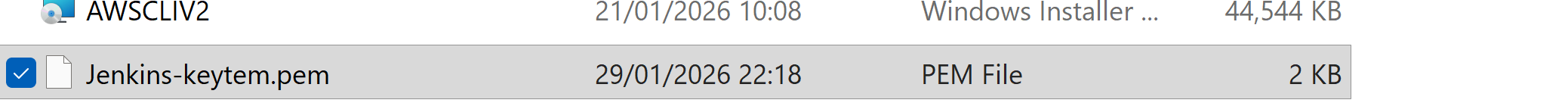
1)create folder local drive jenkins-terraform1

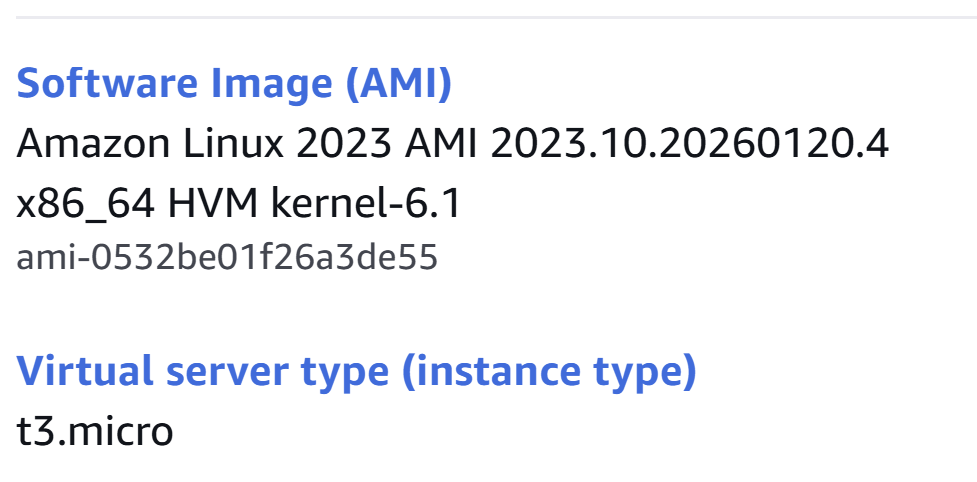


2) create github repo

3)open vs code create this structure



Create keypair : 

Don’t lanuch instance . 

Root code main.tf

provider "aws" {

  region = var.region

}

module "vpc" {

  source = "./modules/vpc"

}

module "sg" {

  source = "./modules/security\_group"

  vpc\_id = module.vpc.vpc\_id

}

module "ec2" {

  source      = "./modules/ec2"

  subnet\_id   = module.vpc.subnet\_id

  sg\_id       = module.sg.sg\_id

  key\_name    = var.key\_name

  github\_repo = var.github\_repo

}

module "snapshots" {

  source      = "./modules/snapshots"

  instance\_id = module.ec2.instance\_id

}

Outputs.tf

output "jenkins\_url" {

  value = "http://${module.ec2.eip}:8080"

}

terraform.tfvars

key\_name    = "Jenkins-keytem"

github\_repo = "https://github.com/akshugupta123/jenkins-docker-compose.git"

variables.tf

variable "region" {

  default = "us-east-1"

}

variable "key\_name" {}

variable "github\_repo" {}

modules/ec2/main.tf

variable "subnet\_id" {}

variable "sg\_id" {}

variable "key\_name" {}

variable "github\_repo" {}

data "aws\_ami" "amazon\_linux" {

  most\_recent = true

  owners      = ["amazon"]

  filter {

    name   = "name"

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

  }

}

resource "aws\_eip" "this" {

  domain = "vpc"

}

resource "aws\_instance" "jenkins" {

  ami                         = data.aws\_ami.amazon\_linux.id

  instance\_type               = "t3.micro"

  key\_name                    = var.key\_name

  subnet\_id                   = var.subnet\_id

  vpc\_security\_group\_ids      = [var.sg\_id]

  associate\_public\_ip\_address = true

  # 🔥 Important — bigger disk for Docker/Jenkins

  root\_block\_device {

    volume\_size = 20

    volume\_type = "gp3"

  }

  user\_data = replace(file("${path.module}/userdata.sh"), "REPO\_PLACEHOLDER", var.github\_repo)

  tags = {

    Name = "jenkins-server"

  }

}

resource "aws\_eip\_association" "assoc" {

  instance\_id   = aws\_instance.jenkins.id

  allocation\_id = aws\_eip.this.id

}

output "instance\_id" {

  value = aws\_instance.jenkins.id

}

output "eip" {

  value = aws\_eip.this.public\_ip

}

Userdata.sh

#!/bin/bash

set -eux

yum update -y

yum install -y docker git

systemctl enable docker

systemctl start docker

# Install Docker Compose

curl -L https://github.com/docker/compose/releases/latest/download/docker-compose-linux-x86\_64 -o /usr/local/bin/docker-compose

chmod +x /usr/local/bin/docker-compose

usermod -aG docker ec2-user

# Clone repo as ec2-user

sudo -u ec2-user git clone "REPO\_PLACEHOLDER" /home/ec2-user/jenkins-docker-compose

# Give Docker some time

sleep 40

# Start Jenkins using full path (no group issue)

cd /home/ec2-user/jenkins-docker-compose

/usr/local/bin/docker-compose up -d

Security\_group/main.tf

variable "vpc\_id" {}

resource "aws\_security\_group" "jenkins\_sg" {

  vpc\_id = var.vpc\_id

  ingress {

    from\_port   = 22

    to\_port     = 22

    protocol    = "tcp"

    cidr\_blocks = ["0.0.0.0/0"]

  }

  ingress {

    from\_port   = 8080

    to\_port     = 8080

    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"]

  }

}

Outputs.tf

output "sg\_id" { value = aws\_security\_group.jenkins\_sg.id }

snapshots/main.tf

variable "instance\_id" {}

resource "aws\_iam\_role" "dlm\_role" {

  name = "dlm-lifecycle-role"

  assume\_role\_policy = jsonencode({

    Version = "2012-10-17",

    Statement = [{

      Action = "sts:AssumeRole",

      Effect = "Allow",

      Principal = { Service = "dlm.amazonaws.com" }

    }]

  })

}

resource "aws\_iam\_role\_policy\_attachment" "attach" {

  role       = aws\_iam\_role.dlm\_role.name

  policy\_arn = "arn:aws:iam::aws:policy/service-role/AWSDataLifecycleManagerServiceRole"

}

resource "aws\_dlm\_lifecycle\_policy" "daily" {

  description        = "Daily Jenkins EBS snapshots"

  execution\_role\_arn = aws\_iam\_role.dlm\_role.arn

  state              = "ENABLED"

  policy\_details {

    resource\_types = ["VOLUME"]

    schedule {

      name = "daily"

      create\_rule {

        interval      = 24

        interval\_unit = "HOURS"

      }

      retain\_rule {

        count = 7

      }

    }

    target\_tags = {

      Name = "jenkins-server"

    }

  }

}

vpc/main.tf

resource "aws\_vpc" "this" {

  cidr\_block = "10.0.0.0/16"

  tags = { Name = "jenkins-vpc" }

}

resource "aws\_subnet" "public" {

  vpc\_id                  = aws\_vpc.this.id

  cidr\_block              = "10.0.1.0/24"

  map\_public\_ip\_on\_launch = true

  availability\_zone       = "us-east-1a"

  tags = { Name = "jenkins-public-subnet" }

}

resource "aws\_internet\_gateway" "igw" {

  vpc\_id = aws\_vpc.this.id

}

resource "aws\_route\_table" "rt" {

  vpc\_id = aws\_vpc.this.id

}

resource "aws\_route" "default" {

  route\_table\_id         = aws\_route\_table.rt.id

  destination\_cidr\_block = "0.0.0.0/0"

  gateway\_id             = aws\_internet\_gateway.igw.id

}

resource "aws\_route\_table\_association" "rta" {

  subnet\_id      = aws\_subnet.public.id

  route\_table\_id = aws\_route\_table.rt.id

}

Outputs.tf

output "vpc\_id"     { value = aws\_vpc.this.id }

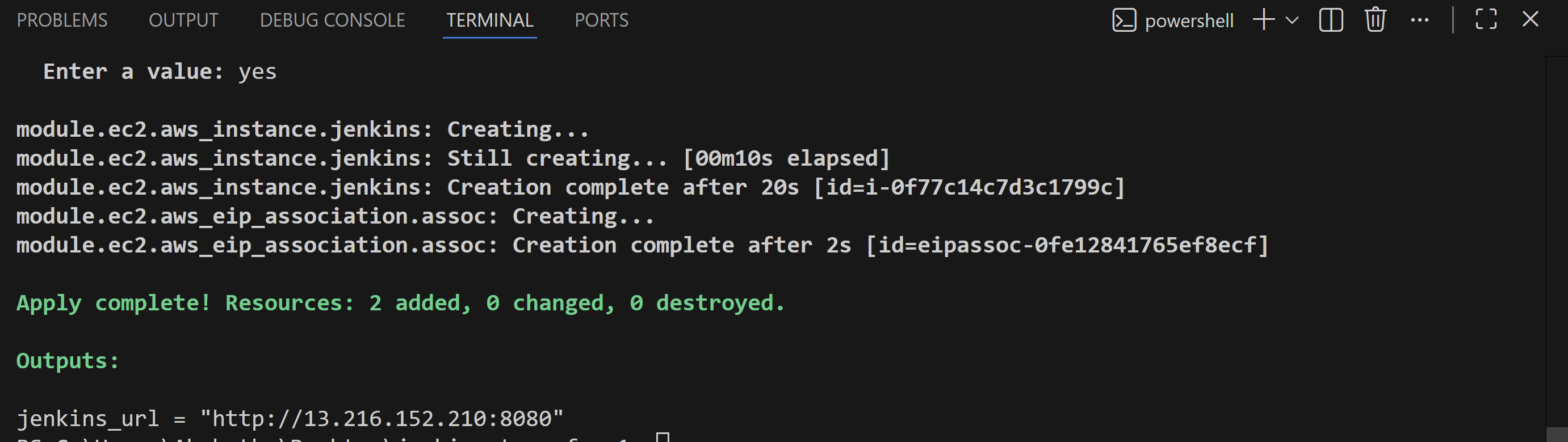
output "subnet\_id"  { value = aws\_subnet.public.id }

commands:

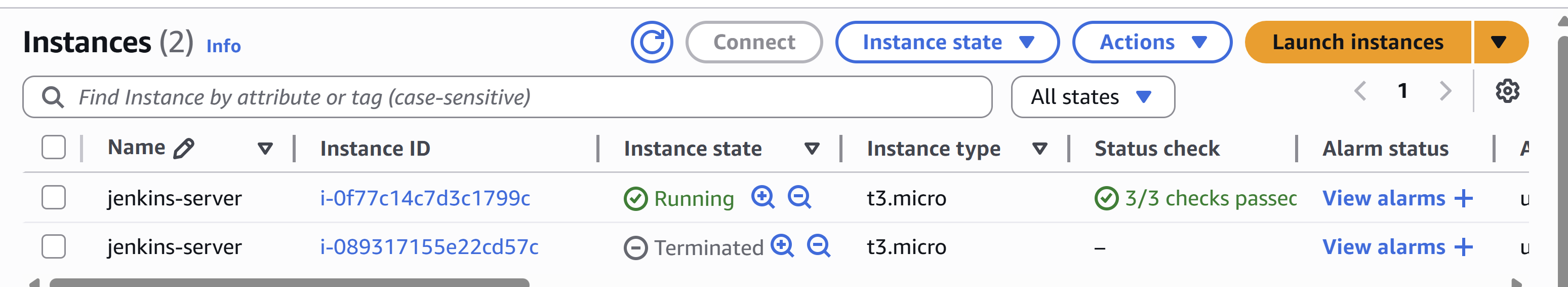
terraform init -reconfigure

terraform plan

terraform apply

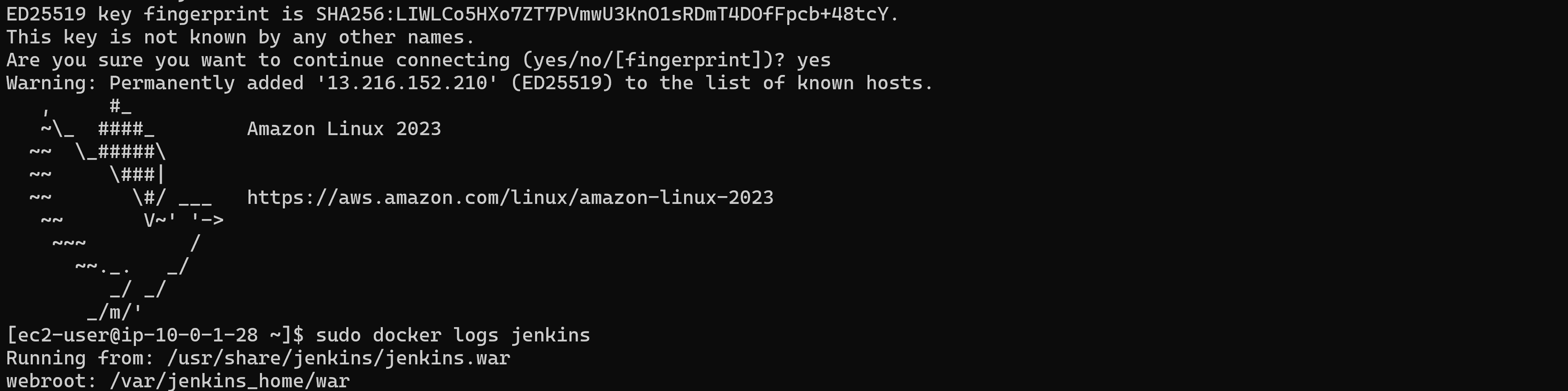


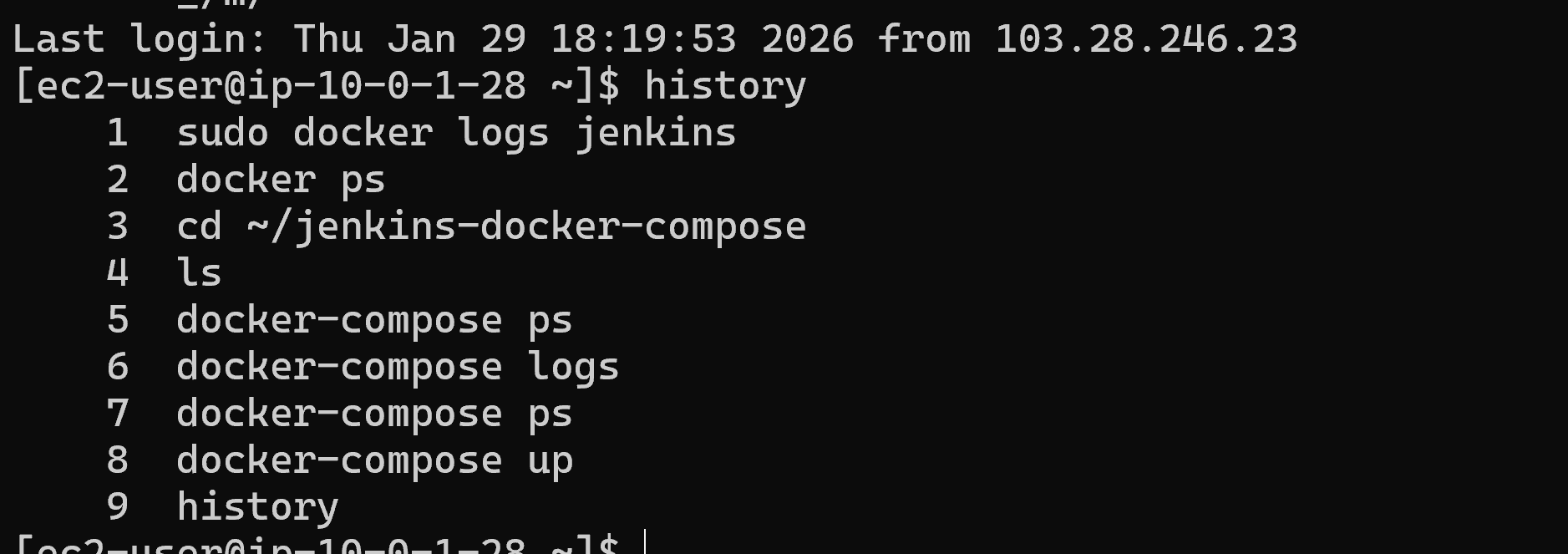
After check with aws instance create or not

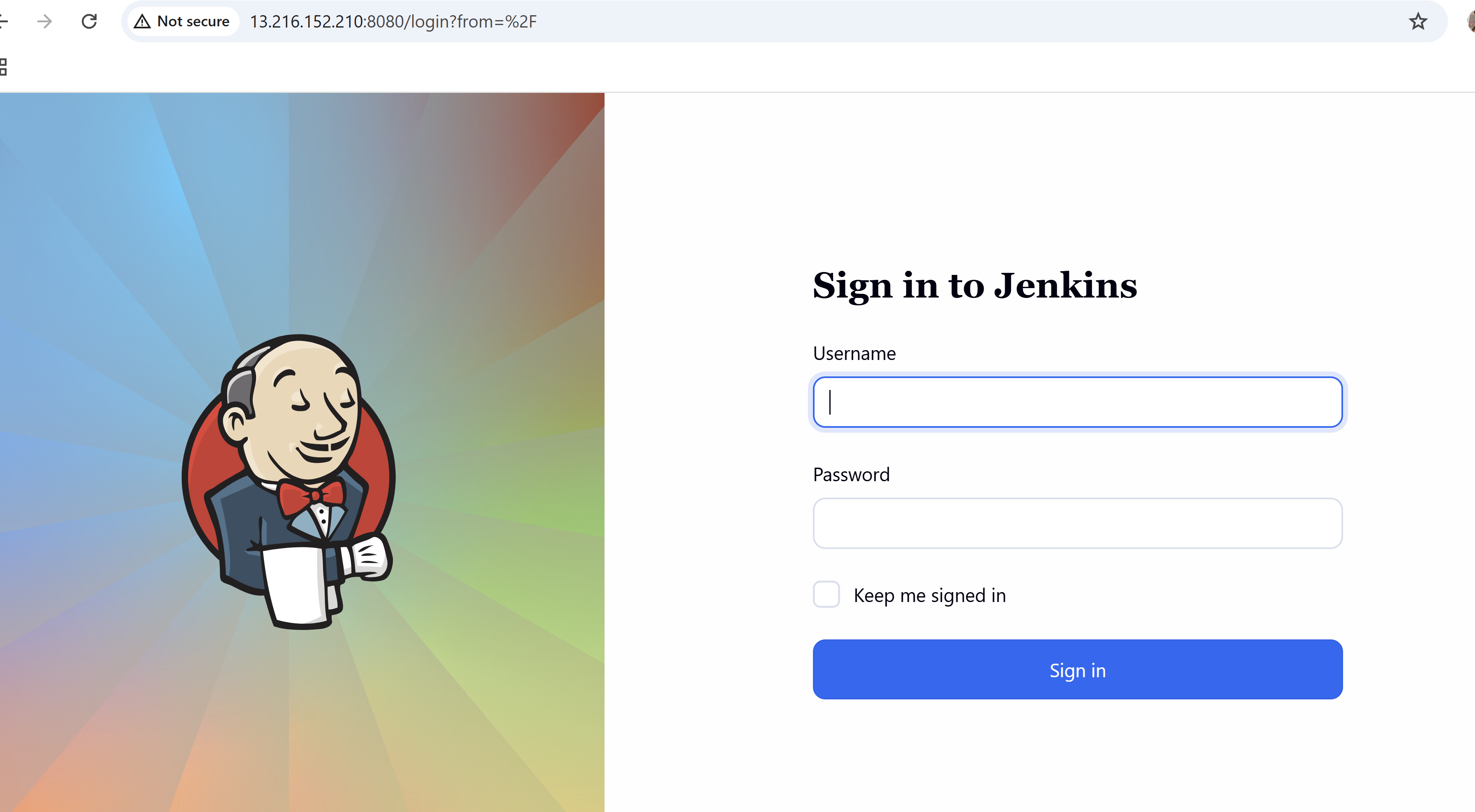


ssh-keygen -R 13.216.152.210

ssh -i "C:\Users\Akshatha\Desktop\Jenkins-keytem.pem" [ec2-user@13.216.152.210](mailto:ec2-user@13.216.152.210)







**🚀 Project: Jenkins Server on AWS using Terraform, Docker Compose & GitHub**

**🎯 Goal**

Provision a fully automated Jenkins server on AWS using **modular Terraform**, deploy Jenkins using **Docker Compose from a GitHub repo**, assign an **Elastic IP**, and enable **daily EBS snapshots** with **data persistence**.

**🧱 Infrastructure Provisioned (Terraform – Modular)**

You created reusable modules for:

* **VPC** (subnet, route table, IGW)
* **Security Group** (22, 8080 open)
* **EC2 Instance**
  + Amazon Linux 2023
  + 20 GB EBS root volume (gp3)
  + Elastic IP attached
  + Userdata for automated setup
* **IAM + AWS DLM**
  + Automated daily EBS snapshots
  + Retention policy

All wired through a **root main.tf** using variables and tfvars.

**🐳 Jenkins Deployment (Docker Compose from GitHub)**

Instead of installing Jenkins manually:

* Created a GitHub repo with docker-compose.yml
* EC2 userdata:
  + Installs Docker
  + Installs Docker Compose
  + Clones GitHub repo
  + Runs docker-compose up -d

This means Jenkins can be recreated anytime from code.

**💾 Data Persistence (Critical Requirement)**

Verified using:

docker volume ls

docker volume inspect jenkins-docker-compose\_jenkins\_data

Confirmed Jenkins data stored in:

/var/lib/docker/volumes/.../\_data

Contains:

jobs, plugins, users, secrets, config.xml

Container can be removed → data still safe.

**🌍 Access**

Jenkins accessible via:

http://Elastic-IP:8080

**🔁 EBS Snapshots (Disaster Recovery)**

Using AWS Data Lifecycle Manager via Terraform:

* Snapshot every 24 hours
* Retain last 7 snapshots
* Tagged to Jenkins volume

**🧠 Key DevOps Learnings You Hit (real-world issues)**

You solved real production problems:

* Terraform modules not wired
* Region mismatch confusion
* Docker compose missing on AL2023
* Userdata timing issues (docker group)
* Disk full error (default 2GB EBS)
* Forcing EC2 recreation with terraform taint
* Templatefile vs replace() issue

This is **real DevOps troubleshooting**, not tutorial work.

**🗣️ How you explain this in interview (short)**

“I provisioned a modular AWS infrastructure using Terraform to host Jenkins on EC2 with an Elastic IP. Jenkins is deployed via Docker Compose pulled from GitHub with persistent Docker volumes. I also implemented automated daily EBS snapshots using AWS DLM for backup and recovery.”