Name: Kaushal Kotkar Department: Cyber Security

Div: BE-15 Roll No: 54

Subject: DSO

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Experiment No. – 10** | | | | |
| **Date of Performance:** | 16/09/2024 | | | |
| **Date of Submission:** | 07/10/2024 | | | |
| Program Execution/ formation/ correction/  ethical practices (06) | Timely Submission  (01) | Viva (03) | Experiment Total (10) | Sign with Date |
|  |  |  |  |  |

**Experiment No. 10**

**Aim:** To implement terraform script for deploying compute/Storage/network infrastructure on the public cloud platform (GCP / AWS / Azure).

**Lab Outcome:** Implement Terraform scripts to manage VMs on a cloud.

**Theory:**

**Terraform:** Terraform is an open-source infrastructure as code (IAC) tool by HashiCorp that allows you to define and provision infrastructure using a declarative configuration language. It provides a consistent way to create, modify, and manage infrastructure across various cloud providers and on-premises environments.

**Why Use Terraform:**

* **Infrastructure as Code:** Terraform allows you to define your infrastructure as code, making it version able, maintainable, and reproducible.
* **Multi-Cloud Support:** It supports multiple cloud providers, enabling you to manage infrastructure consistently across AWS, Azure, Google Cloud, and others.
* **Resource Management:** Terraform can manage a wide range of resources, including virtual machines, networks, storage, and more.
* **Modularity:** It encourages modular code and reusability, making it easier to manage complex infrastructure.

**Why Terraform with Cloud Platforms:**

* Using Terraform with cloud platforms like AWS provides a structured and automated approach to provisioning and managing resources.
* It helps ensure infrastructure consistency and scalability.
* Terraform's modular approach allows you to manage complex cloud environments efficiently.

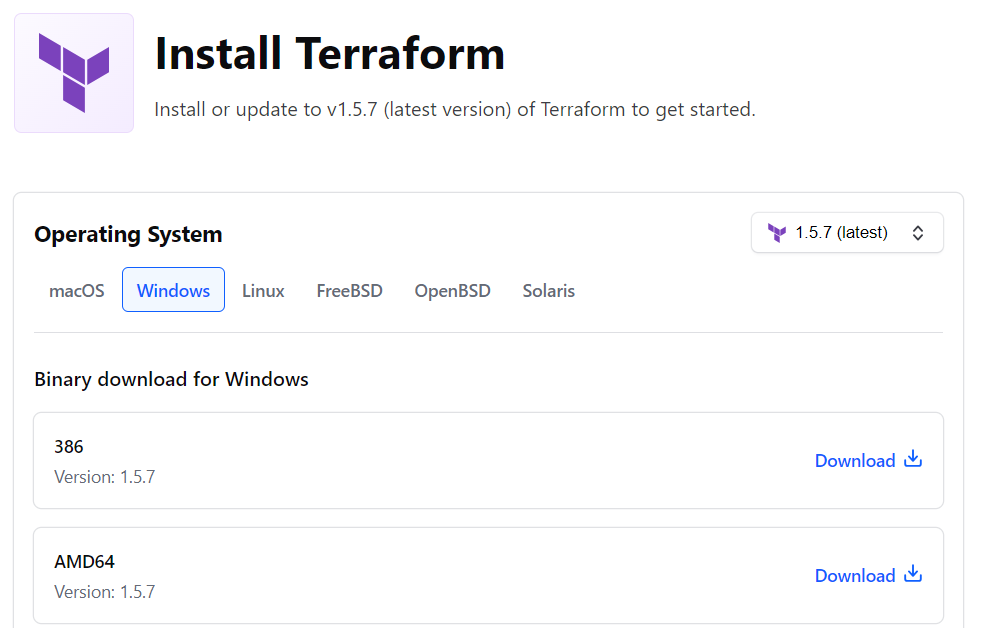
**AWS (Amazon Web Services):** AWS is a leading cloud services provider offering a wide range of cloud computing services, including computing power, storage, databases, machine learning, analytics, and more. It's widely used for hosting applications, websites, and managing various cloud resources.

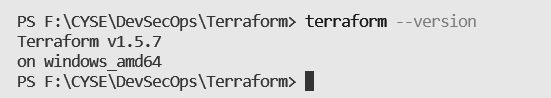
**Steps Followed:**

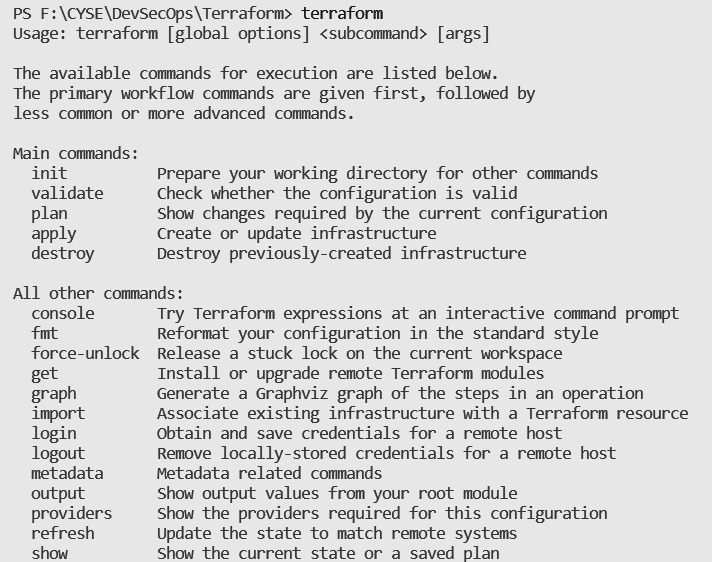
1. **Installation of Terraform:**
   * Download Terraform from the HashiCorp website.
   * Add the Terraform binary to your system's PATH.
   * Verify the installation using **terraform --version**.
2. **AWS Account Setup:**
   * Sign up for an AWS account as the root user.
   * Provide personal and payment details to complete the registration.
3. **Add IAM User:**
   * Log in to the AWS console as the root user.
   * Go to the Identity and Access Management (IAM) service and create a new IAM user.
   * Configure permissions
   * Login as IAM user
   * Create an access key for the IAM user.
   * Save the access key ID and secret access key for later use.
4. **Create Instance Using Terraform Script:**
   * Install Visual Studio Code or any other code editor.
   * Create a folder for your Terraform project.
   * Create a **main.tf** file and write a Terraform script to define your infrastructure resources (e.g., EC2 instance, VPC, security groups).
   * Run the following Terraform commands:
     + **terraform init**: Initializes the project and downloads necessary providers.
     + **terraform plan**: Shows the execution plan without making changes.
     + **terraform apply**: Creates the AWS resources based on your Terraform configuration.
     + **terraform destroy**: Destroys the created resources when no longer needed.

**Output:**

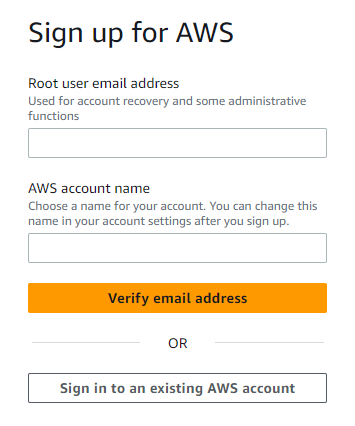
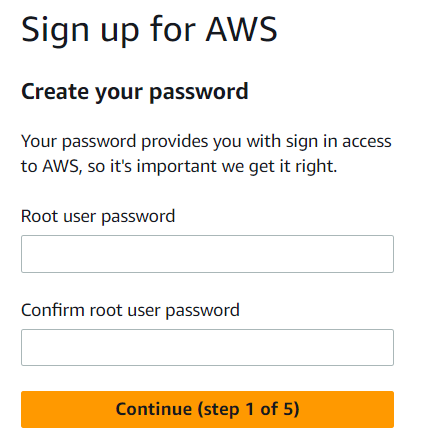
Installation of Terraform:

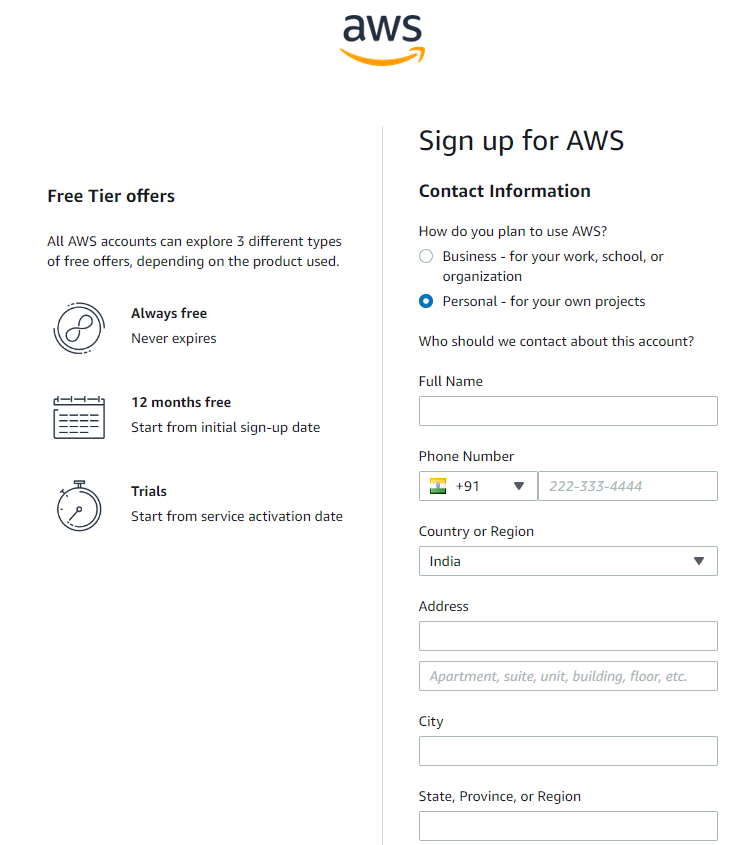


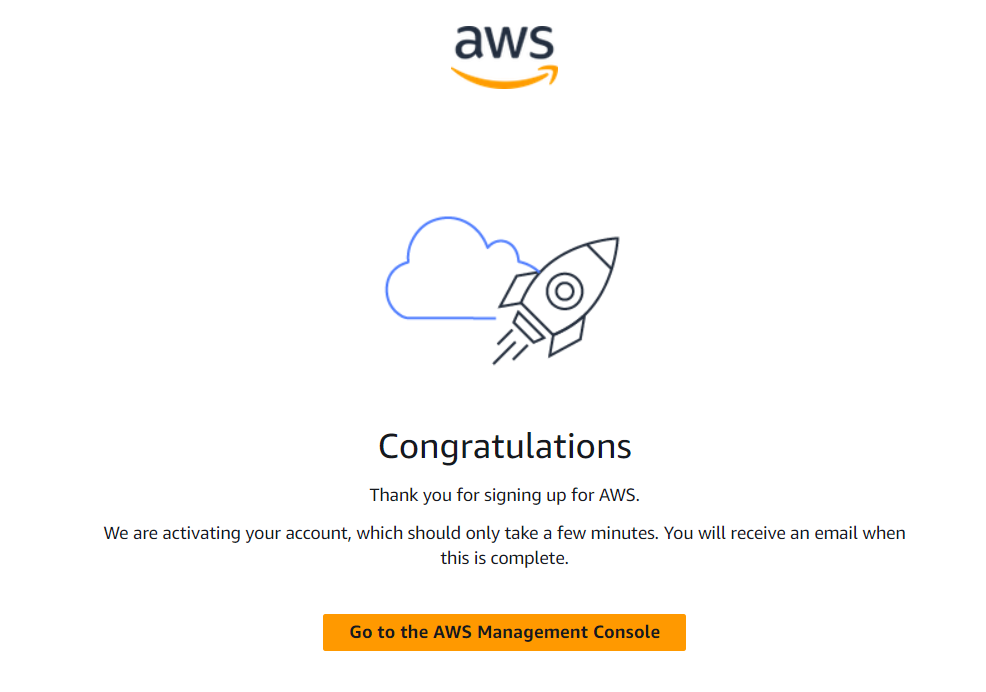


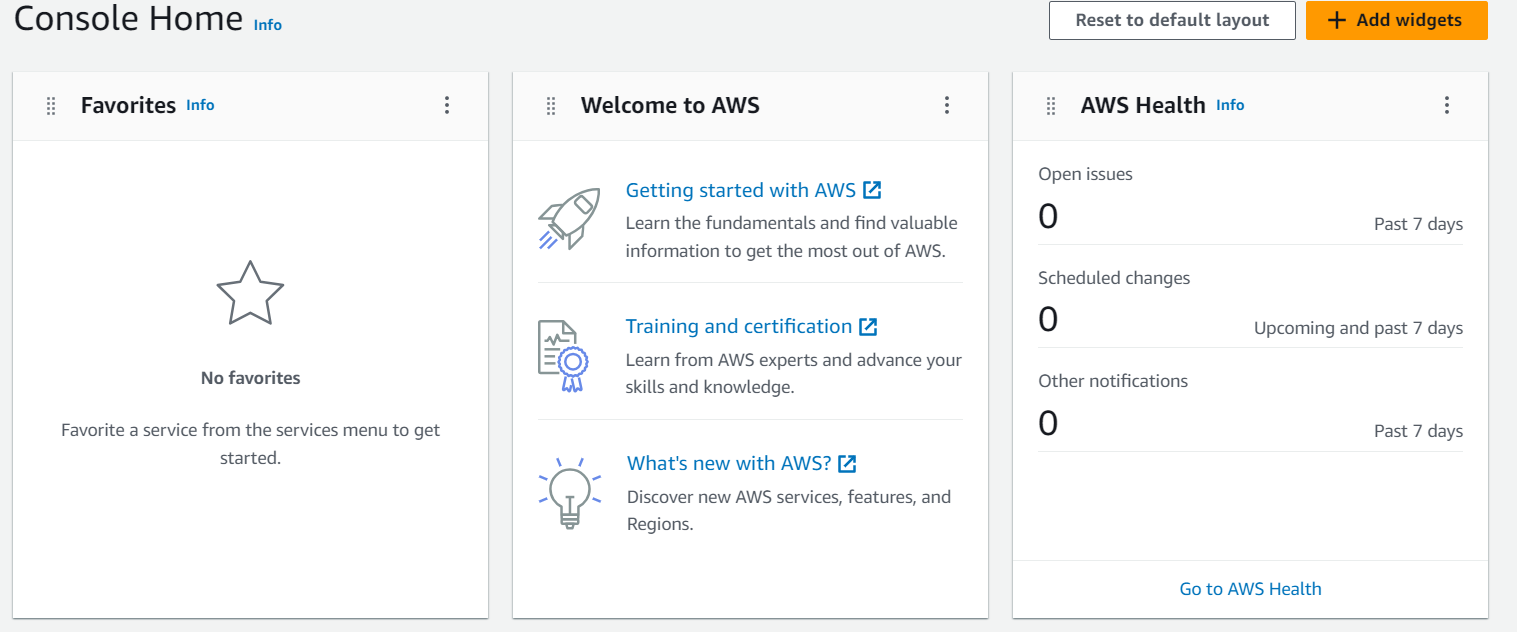


AWS Account Setup:

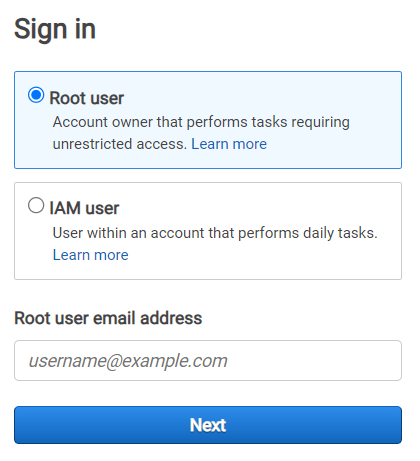
 

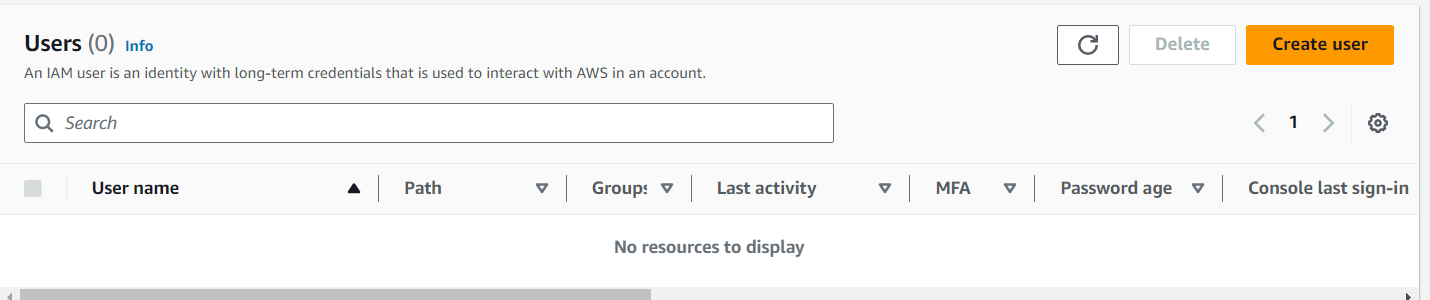


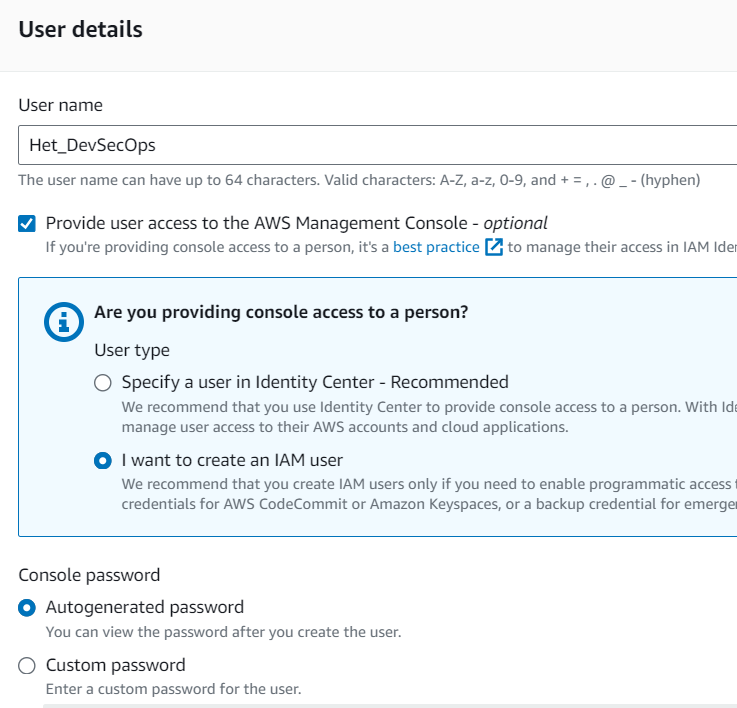


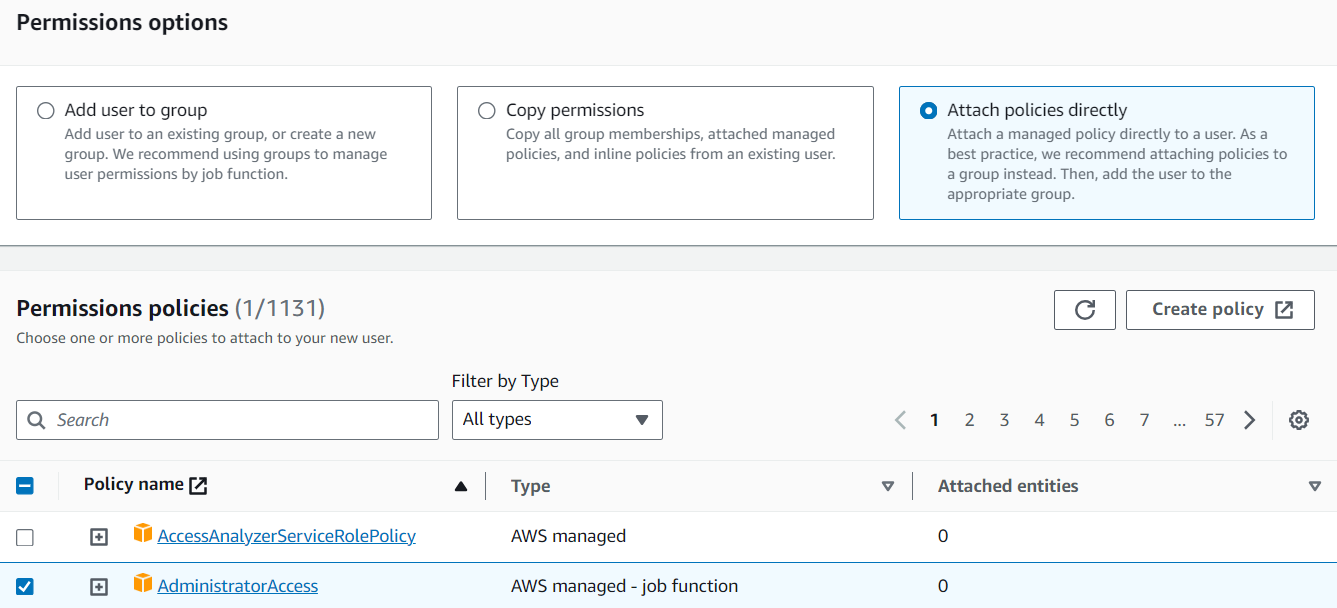


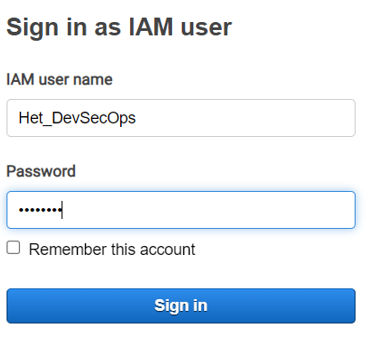
Add IAM User:

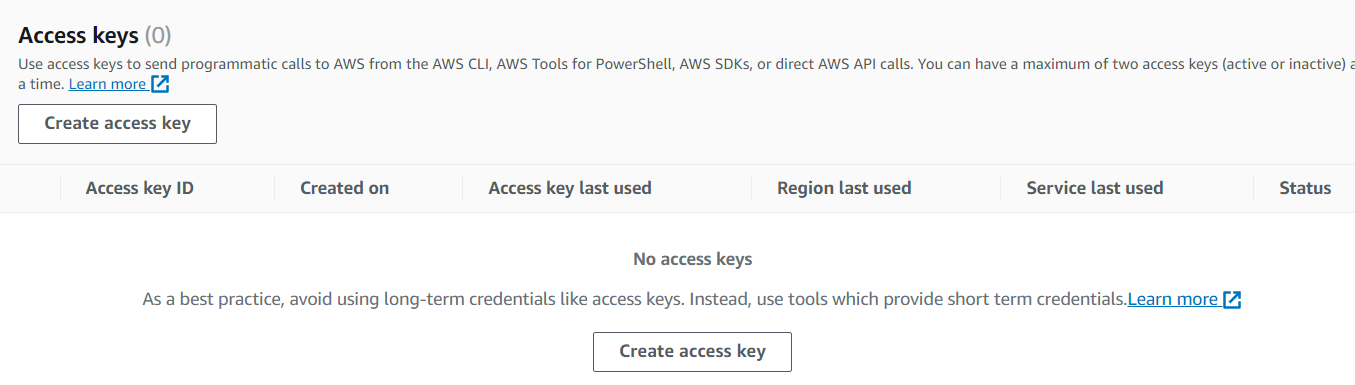




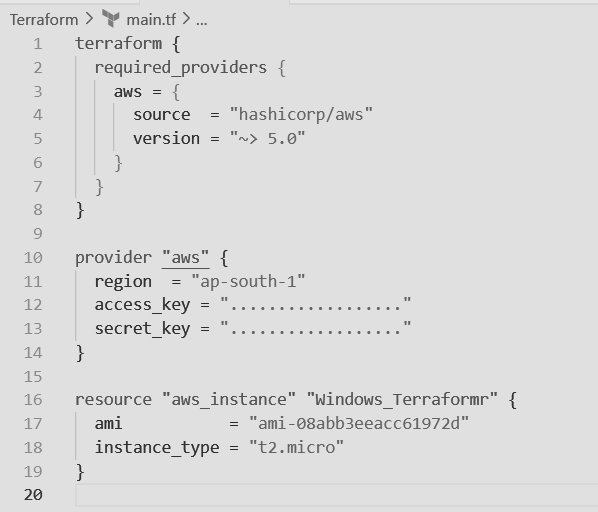




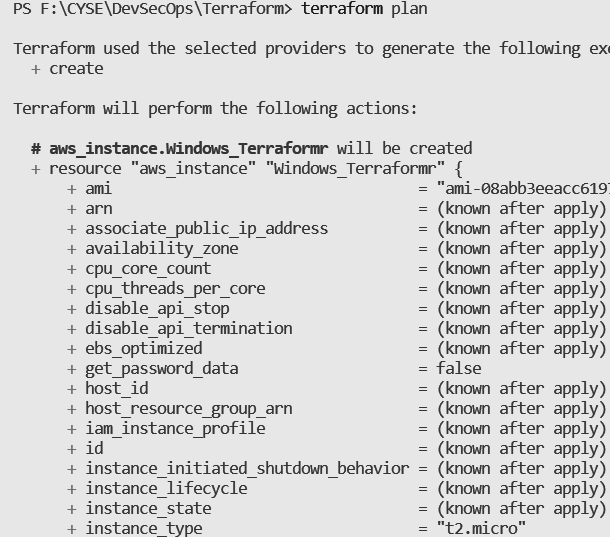




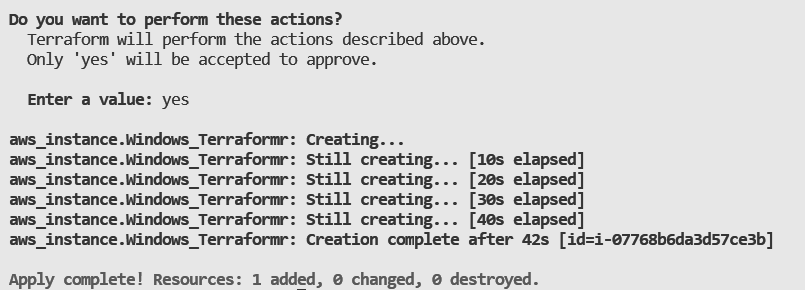
Create Instance Using Terraform Script:

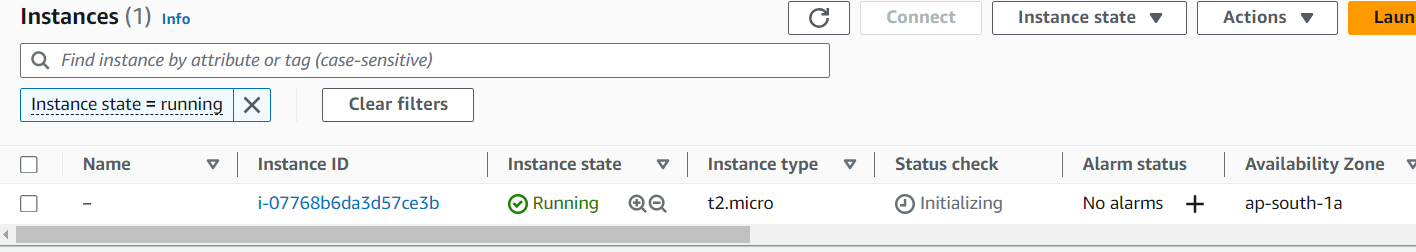


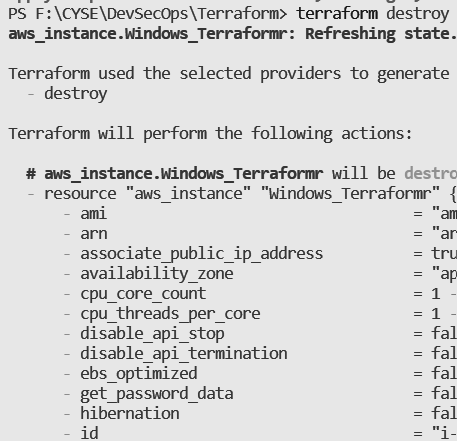


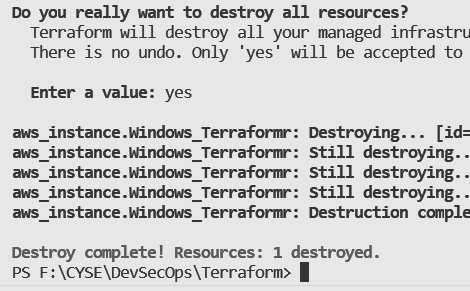


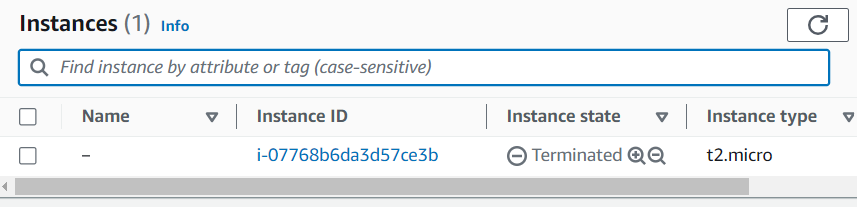












**Conclusion:**

In this experiment, we successfully utilized Terraform to deploy an AWS EC2 instance using infrastructure-as-code (IAC) scripts. Terraform's declarative approach allowed us to define and provision cloud resources with precision and repeatability. This streamlined method enhances the management of cloud infrastructure, making it easier to scale, version, and maintain AWS resources.