Terraform

*(@theshivanshvasu)*

**What is Terraform?**

Terraform is an open-source Infrastructure as Code (IaC) tool created by HashiCorp.

It allows you to define and provision infrastructure using a high-level configuration language.

**Key Features of Terraform:**

Declarative Language: Users write configurations in *HashiCorp Configuration Language (HCL)* or JSON.

Execution Plans: Terraform generates an execution plan showing what it will do before making changes.

Resource Graph: It builds a dependency graph and determines the most efficient sequence to create, update, or destroy resources.

Change Automation: Manages changes and dependencies across infrastructure.

**Installing Terraform**

Download and Install:

Go to the Terraform website.

Download the appropriate package for your operating system.

Unzip the package and place the binary in a directory included in your system's PATH.

**Verify Installation:**

**Run terraform --version** in your terminal to ensure Terraform is installed correctly.

**Basic Concepts**

1. *Providers*:

Providers are responsible for understanding API interactions and exposing resources.

Examples: AWS, Azure, Google Cloud, Kubernetes, etc.

1. *Resources*:

Resources are the most important element in the Terraform language.

Each resource block describes one or more infrastructure objects.

1. *Variables:*

Variables allow you to parameterize your Terraform configurations.

They can be defined in the .tf files or via environment variables.

1. *State:*

Terraform maintains a state file to map your configuration to real-world resources.

The state file is used to track the current state of the infrastructure.

**Writing Your First Terraform Configuration**

*Create a Directory:*

Create a new directory for your Terraform configuration files.

Define a Provider:

| provider "aws" {  region = "us-west-2" }   resource "aws\_instance" "example" {  ami = "ami-0c55b159cbfafe1f0"  instance\_type = "t2.micro" } |
| --- |

Initialize Terraform:

1. Run terraform init to initialize the working directory and download necessary providers.
2. Create an Execution Plan:
3. Run terraform plan to create an execution plan.
4. This shows what actions Terraform will take to achieve the desired state.
5. Apply the Changes:
6. Run terraform apply to execute the plan and create the resources.

Inspect the State:

Run terraform show to display the current state of your infrastructure.

*Terraform is an open-source infrastructure as code software tool created by HashiCorp. Users define and provide data center infrastructure using a declarative configuration language known as HashiCorp Configuration Language, or optionally JSON.*

Resources:

* <https://www.youtube.com/watch?v=HmxkYNv1ksg>
* <https://www.youtube.com/watch?v=YcJ9IeukJL8&t=2889s>
* <https://www.youtube.com/watch?v=iRaai1IBlB0>
* <https://www.youtube.com/watch?v=SLB_c_ayRMo>
* <https://www.youtube.com/c/AntonPutra>

Links:

* <https://www.terraform.io/>
* <https://learn.hashicorp.com/terraform?utm_source=terraform_io>
* <https://learn.hashicorp.com/collections/terraform/aws-get-started>
* <https://learn.hashicorp.com/collections/terraform/cloud-get-started>

Courses:

* <https://www.udemy.com/course/terraform-beginner-to-advanced/>
* Check out the terraform learning website, it has all you need to know about terraform

Getting Started:

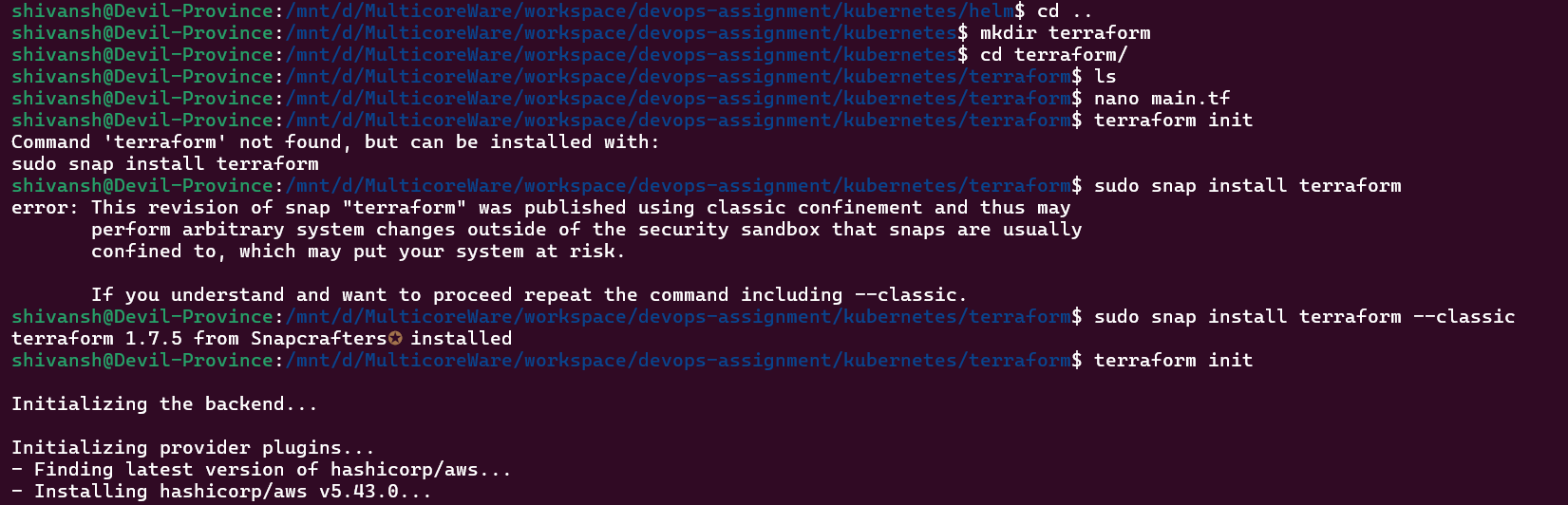
* [Install terraform CLI](https://learn.hashicorp.com/tutorials/terraform/install-cli)
* [Terraform Concepts](https://blog.knoldus.com/terraform-basic-concepts/#:~:text=Terraform%20creates%20an%20execution%20plan,plans%20that%20can%20be%20implemented.)
* [Terraform Modules](https://learn.hashicorp.com/collections/terraform/modules)
* [Terraform State](https://learn.hashicorp.com/collections/terraform/state)
* [Terraform with kubernetes](https://learn.hashicorp.com/collections/terraform/kubernetes)

Certifications:

* <https://www.hashicorp.com/certification/terraform-associate>

**To create a basic Kubernetes cluster and deploy a simple Docker container on AWS using Terraform**

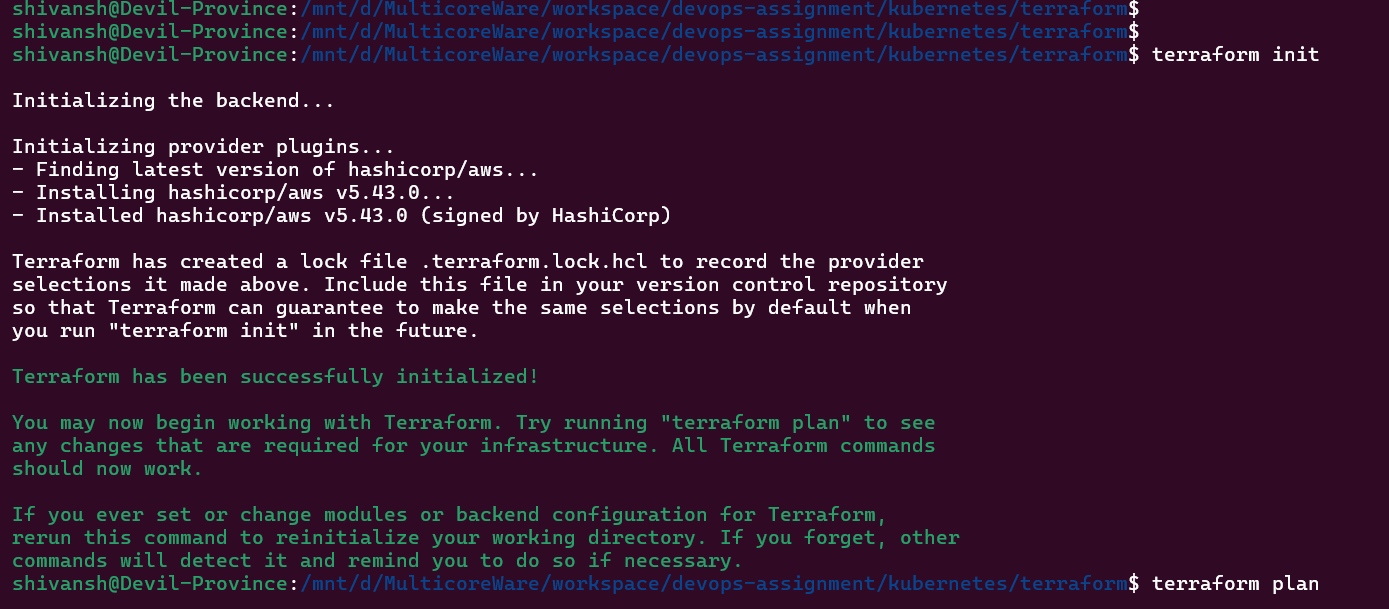
Installing Terraform:



Main.tf File:

| # Define the provider (AWS) provider "aws" {  region = "us-west-2" # Specify your desired AWS region } # Create an EC2 instance for the Kubernetes cluster resource "aws\_instance" "kubernetes\_cluster" {  ami = "ami-0c55b159cbfafe1f0" # Specify a suitable AMI for your EC2 instance  instance\_type = "t2.micro" # Specify the instance type (adjust as needed)  key\_name = "your\_key\_pair" # Specify your SSH key pair } # Create a security group for the EC2 instance (open ports for Kubernetes and Docker) resource "aws\_security\_group" "kubernetes\_security\_group" {  name\_prefix = "kubernetes-"    ingress {  from\_port = 22 # SSH  to\_port = 22  protocol = "tcp"  cidr\_blocks = ["0.0.0.0/0"]  }   ingress {  from\_port = 6443 # Kubernetes API server  to\_port = 6443  protocol = "tcp"  cidr\_blocks = ["0.0.0.0/0"]  }   ingress {  from\_port = 80 # HTTP (Docker container)  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 = "kubernetes-security-group"  } } |
| --- |

**Terraform Init:**



**Terraform Plan.**

**Terraform Apply.**

* Create an Nginx deployment with 4 replicas using kubectl.

| # main.tf  # Provider configuration for Kubernetes provider "kubernetes" {  config\_context\_cluster = "YOUR\_CLUSTER\_NAME" # Replace with your cluster name }  # Define the Nginx deployment resource "kubernetes\_deployment" "nginx\_deployment" {  metadata {  name = "nginx-deployment"  }   spec {  replicas = 4   selector {  match\_labels = {  app = "nginx"  }  }   template {  metadata {  labels = {  app = "nginx"  }  }   spec {  container {  image = "nginx:latest"  name = "nginx"  ports {  container\_port = 80  }  }  }  }  } } |
| --- |

