Terraform

What is Terraform?

Terraform is an open-source Infrastructure as Code (IaC) tool developed by HashiCorp that allows users to define, provision, and manage cloud resources in a consistent and repeatable manner.

It provides a declarative configuration language, HashiCorp Configuration Language (HCL), enabling users to define infrastructure in human-readable and machine-parsable code.

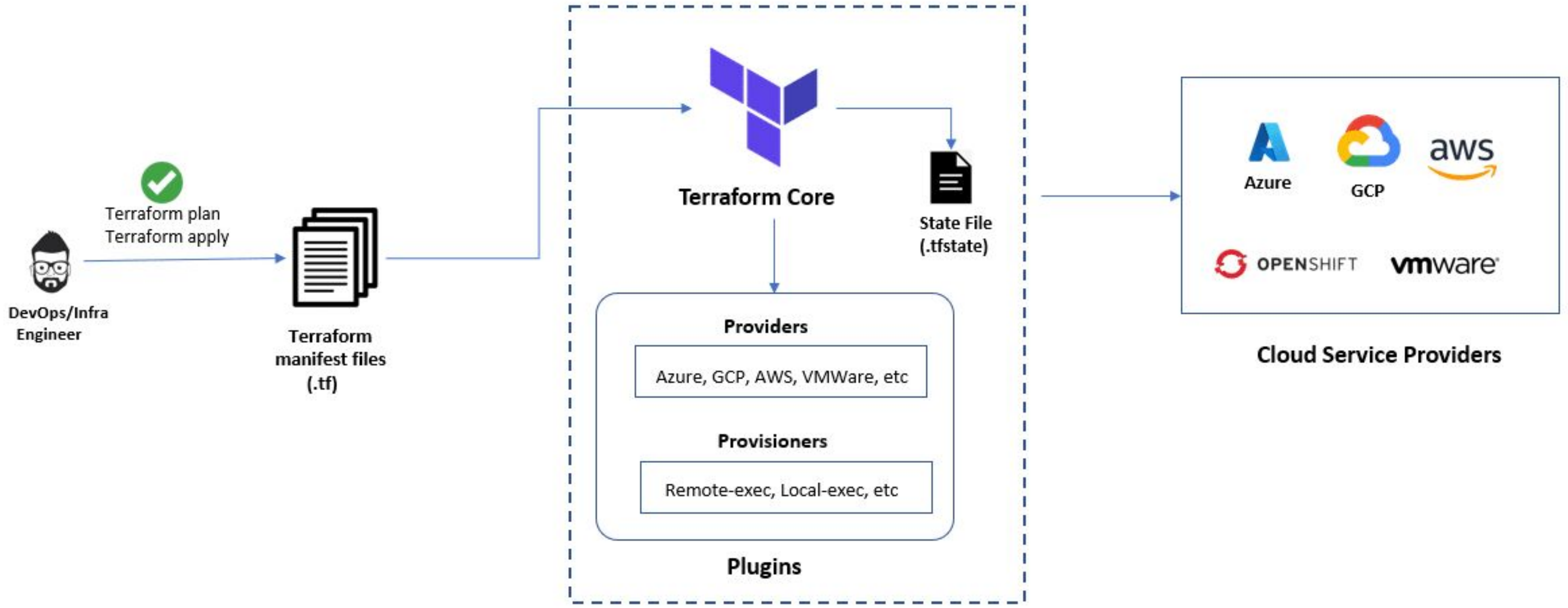
Key Features of Terraform

- **Declarative Language (HCL)**: Write configurations in a human-readable syntax.
- **State Management**: Track the current state of infrastructure and manage changes.
- **Modularity**: Use modules for reusable configurations.
- **Execution Plan**: Preview changes before applying them.
- **Providers and Resources**: Manage cloud services and resources.

Terraform Architecture

- **Configuration Files**: Define resources using .tf files.
- **Terraform Core**: Executes plans and applies configurations.
- **Providers**: Interfaces for managing different cloud services.
- **State Management**: Maintains information about resources.

Terraform Architecture

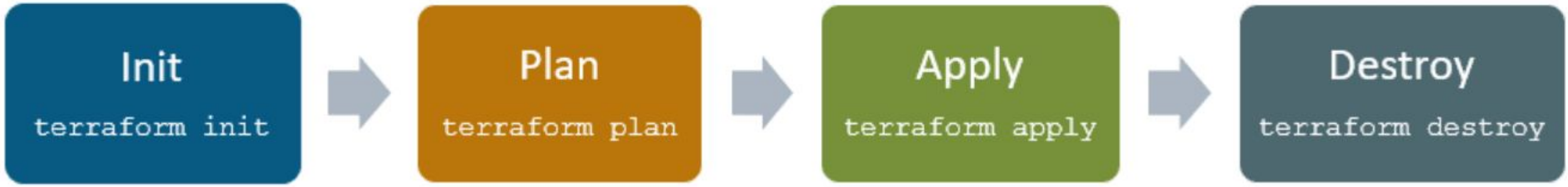


Terraform Configuration Files

- **main.tf**: Defines main resources and infrastructure.
- **variables.tf**: Stores variable definitions.
- **outputs.tf**: Specifies output values to display after deployment.
- **providers.tf**: Sets up cloud providers and their configurations.
- **terraform.tfvars**: Contains variable values for different environments.

Terraform Workflow

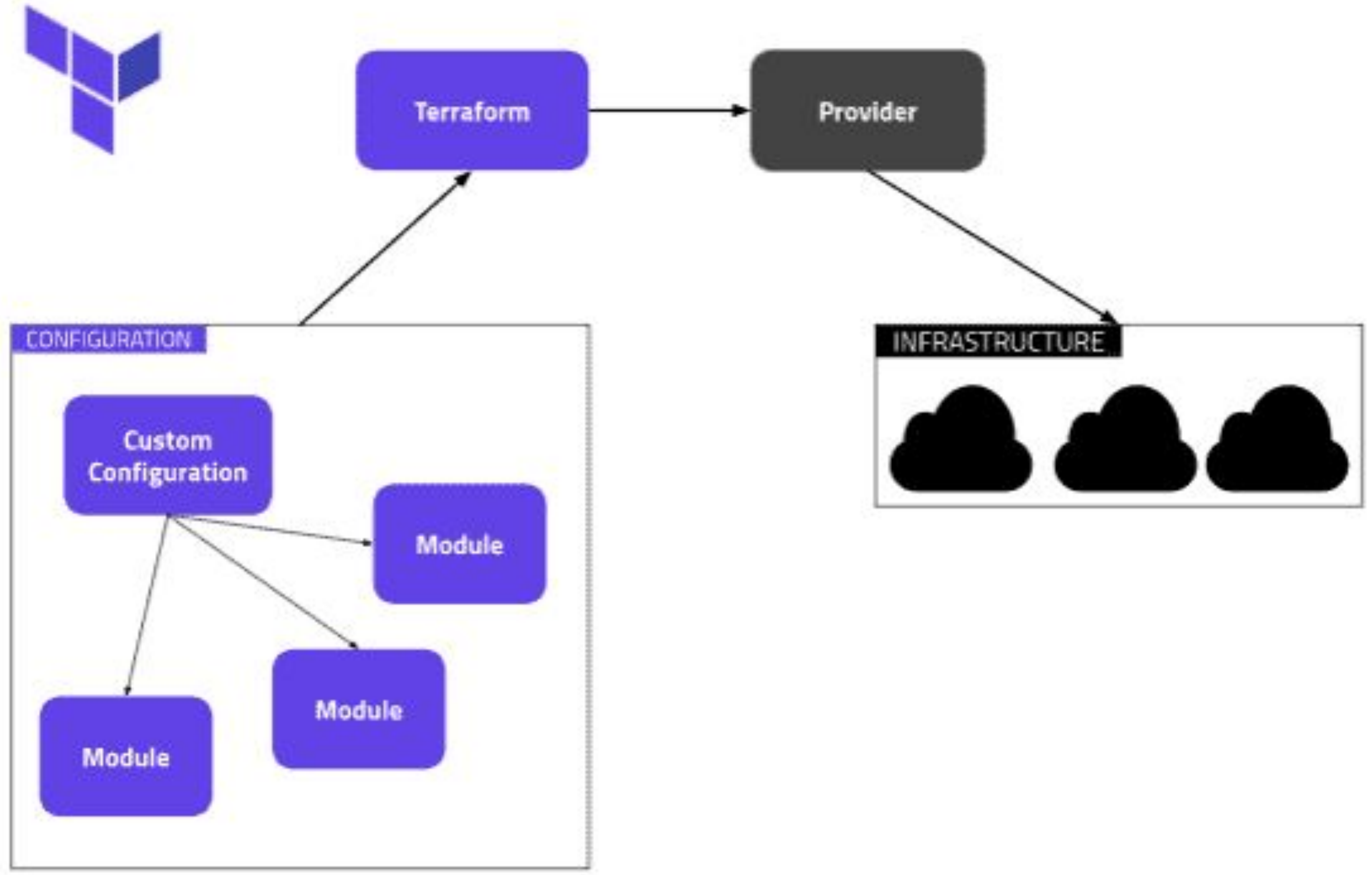
- **Write**: Define infrastructure using configuration files.
- **Initialize**: Run terraform init to set up the environment.
- **Plan**: Preview changes using terraform plan.
- **Apply**: Apply changes using terraform apply.
- **Destroy**: Remove infrastructure using terraform destroy.



Providers and Modules

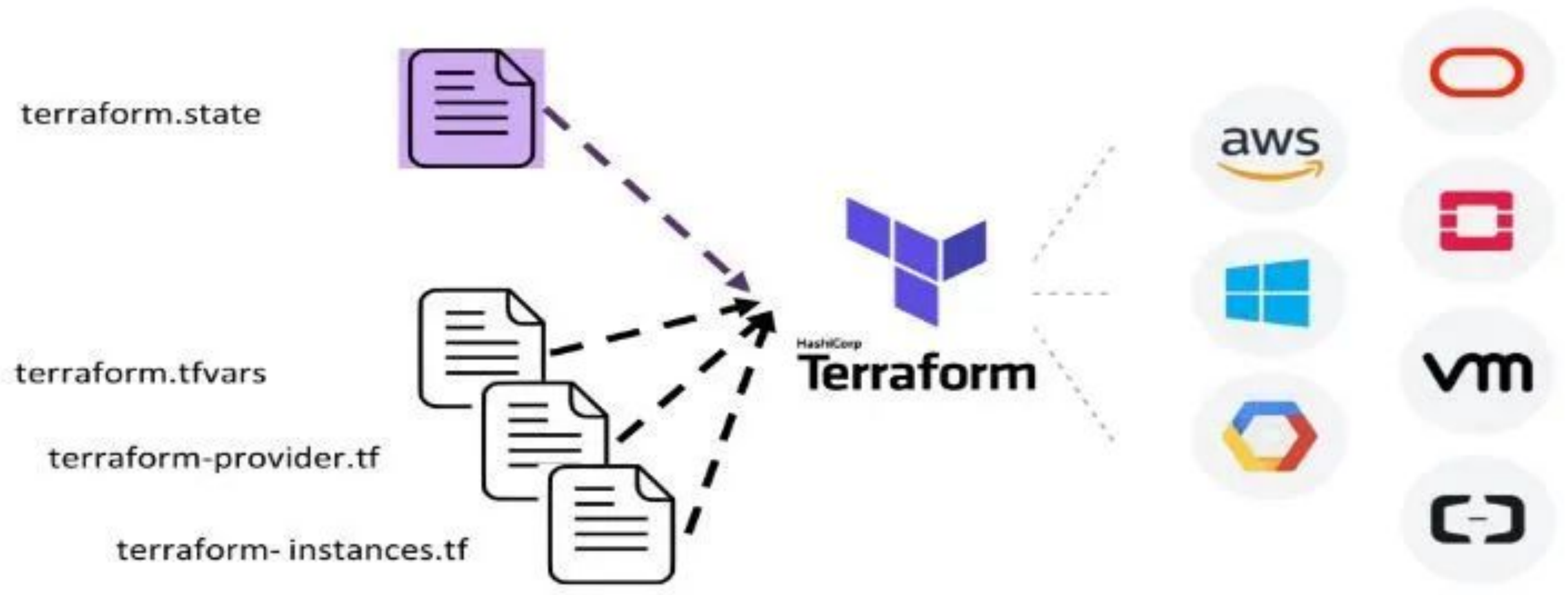
- **Providers**: Plugins for managing resources on specific platforms like AWS, Azure, and GCP.
- **Modules**: Reusable configurations that encapsulate related resources.

Use modules to organize code and manage infrastructure as building blocks.


Terraform
Provider
CONFIGURATION
Custom Configuration
Module
Module
Module
INFRASTRUCTURE

Terraform State

- The state file (terraform.tfstate) records information about managed resources.
- State is essential for tracking resource changes and dependencies.
- Use remote state for collaboration and consistency.
- **State Locking**: Prevents multiple operations on the same state.


terraform.state
terraform.tfvars
terraform-provider.tf
terraform- instances.tf
HashiCorp
Terraform
aws
vm

Terraform Commands

- **terraform init**: Initializes a new or existing Terraform configuration.
- **terraform plan**: Creates an execution plan for changes.
- **terraform apply**: Applies the execution plan.
- **terraform destroy**: Destroys all infrastructure managed by Terraform.
- **terraform validate**: Validates configuration files for syntax errors.
- **terraform fmt**: Formats configuration files.

Terraform Use Cases

- **Provisioning Cloud Resources**: Deploy and manage VMs, networks, and storage.
- **Multi-Cloud and Hybrid Environments**: Manage resources across multiple cloud providers.
- **CI/CD Automation**: Integrate Terraform with CI/CD pipelines for automated deployments.
- **Infrastructure as Code (IaC)**: Version-controlled, repeatable infrastructure deployments.
- **Immutable Infrastructure**: Ensure consistent environments by recreating resources from scratch.

Best Practices with Terraform

- Store configurations in a version control system like Git.
- Use remote state storage for shared state management.
- Implement state locking to prevent race conditions.
- Organize configurations using modules.
- Always run `terraform plan` before `terraform apply`.
- Use environment variables for sensitive data.

Labs

- Install Terraform - https://developer.hashicorp.com/terraform/install
- Provision an NGINX server in less than a minute using Docker on Windows.

  https://developer.hashicorp.com/terraform/tutorials/azure-get-started/install-cli

- Creating an Azure Resource Group with Terraform

  https://learn.microsoft.com/en-us/azure/developer/terraform/create-resource-group?tabs=azure-cli

- Deploying Static Website on Azure Storage using Terraform:

  https://learn.microsoft.com/en-us/azure/storage/blobs/storage-quickstart-static-website-terraform?tabs=azure-cli