Terraform — Practical, Complete Documentation

1. Introduction

Terraform is an Infrastructure-as-Code (IaC) tool that lets you define cloud resources using declarative configuration (HCL). Terraform maintains a state file that maps your configuration to actual resources.

2. Basic Terraform Commands

terraform init

Initializes working directory, downloads providers, initializes backend, fetches modules.

terraform plan -out=tfplan

Creates an execution plan, compares configuration and state, shows what will change.

terraform apply

Executes the plan and updates real resources + updates state.

terraform refresh (Deprecated)

Refresh used to update the state file with real-world attributes but is deprecated.

Modern Refresh Equivalent:

terraform plan -refresh-only -out=refresh.plan

Does NOT update the state file. It only generates a plan.

State only updates after:

terraform apply refresh.plan

3. Additional commands

terraform fmt, validate, show, state, import.

4. Terraform Modules

Modules group related resources into reusable units.

Manual Module Example:

modules/ec2/main.tf, variables.tf, outputs.tf

Used via module "name" { source = "../modules/ec2" }

Terraform Registry Modules:

source = "terraform-aws-modules/vpc/aws"

5. Real-World Scenarios

Scenario 1: EKS add-on created manually not detected by refresh

Terraform cannot detect resources not in state. It only refreshes tracked resources.

Scenario 2: SG description changed manually but refresh didn't show drift

Some AWS attributes are not returned through APIs or not tracked by provider schema.

Scenario 3: Hard-coded AMI no longer exists → plan succeeds, apply fails

Plan does not validate AMIs. Apply validates at runtime. Use data "aws_ami" for dynamic AMI lookup.

Scenario 4: Importing Manual Resource

Create matching tf resource → terraform import resource.id → terraform plan → reconcile.

6. What refresh can and cannot detect

Can detect:

Changes in tracked resources (instance type, tags, rules, etc.)

Cannot detect:

New resources, deleted resources, attributes not exposed by AWS, invalid AMIs, etc.

7. Best Practices

Avoid hardcoding AMI IDs, use modules, use refresh-only plans, avoid console changes, use ignore_changes sparingly.