Terraform Version Constraints

Two Primitives: Arguments and Blocks

HCL has exactly two syntax constructs. Once you recognize them, every Terraform configuration becomes predictable.

An argument assigns a value to a name:

region = "us-central1"

A block is a named container for related configuration:

resource "google_storage_bucket" "training_data" {
name = "my-ml-project-training-data-2026"
location = "US"

Provider Block and Version Constraints

The terraform block sits at the top of your configuration and declares two things: the minimum Terraform core version yo

```
terraform {
required_version = "~> 1.15"
required_providers {
source = "hashicorp/google"
version = "~> 6.0"
provider "google" {
```

The Core Workflow: init → plan → apply → destroy

Terraform's four-command cycle is deliberate. Each step is safe to re-run and progressively more consequential.

terraform init downloads provider binaries into .terraform/ and writes .terraform.lock.hcl. It is idempotent — running i

terraform plan reads current state, compares it to your configuration, and proposes change actions. No cloud resources a

terraform apply presents the plan one final time and prompts for confirmation. After you type yes, Terraform creates, up

terraform destroy is a convenience alias for terraform apply -destroy. It runs the same apply engine in destroy-planning

<Callout type="warning">

The -auto-approve flag skips the interactive confirmation prompt on apply and destroy. Only use it in locked CI environm

Reading Terraform Plan Output

Four symbols appear in plan output, and misreading them is the most common source of production surprises:

-/+ demands the closest scrutiny. It appears when an argument that cannot be changed in place is modified — for example,

A plan ending with Plan: 0 to add, 0 to change, 0 to destroy is Terraform's confirmation that your running infrastructur

Locking Versions with .terraform.lock.hcl

After terraform init, examine the generated lock file:
provider "registry.terraform.io/hashicorp/google" {
version = "6.14.1"
constraints = "~> 6.0"
"h1:ABC123xyz...",
"zh:DEF456uvw...",
version records the exact provider binary installed — 6.14.1 — selected within the ~> 6.0 range. constraints echoes the

Hands-On Exercise

Goal: Provision a GCS training-data bucket and walk the complete init → plan → apply → destroy lifecycle.

1. Create main.tf with the configuration from the worked example in this chapter (substitute your GCP project ID).
2. Run terraform init. Confirm .terraform.lock.hcl was created and lists hashicorp/google with an exact version.
3. Run terraform plan. Verify exactly one + resource in the output and note the planned bucket name.
4. Run terraform apply. At the prompt, type yes. Confirm "Apply complete! Resources: 1 added, 0 changed, 0 destroyed."
5. Open the GCS console — the bucket must be visible with the name from step 3.
6. Run terraform plan -destroy. Read the - symbol on the bucket entry and confirm no other resources appear.

Try It Next

Complete the exercises for:

Terraform Version Constraints

academy.kspl.tech