TERRAFORM QUICK REFERENCE

Providers, resources, variables, state, modules

Basics

Core Workflow

```
terraform init      # install providers & modules
terraform plan      # preview changes
terraform apply     # apply changes
terraform destroy   # tear down all resources
```

Essential Commands

`terraform init`	Initialize working directory, download providers
`terraform plan`	Show execution plan without applying
`terraform apply`	Apply changes to infrastructure
`terraform destroy`	Destroy all managed resources
`terraform fmt`	Format `.tf` files to canonical style
`terraform validate`	Check configuration syntax
`terraform show`	Display current state or plan
`terraform output`	Print output values

Providers

Provider Configuration

```
terraform {
  required_providers {
    aws = { source = "hashicorp/aws", version = "~> 5.0" }
  }
}
provider "aws" {
  region = "us-east-1"
}
```

Provider Notes

`source`	Registry address (`hashicorp/aws`, `hashicorp/google`)
`version`	Version constraint (`~> 5.0`, `>= 3.0, < 4.0`)
`.terraform.lock.hcl`	Lock file — commit to version control
`alias`	Use multiple configs for the same provider

Resources

Resource Blocks

```
resource "aws_instance" "web" {
  ami           = "ami-0c55b159cbfafe1f0"
  instance_type = "t3.micro"
  tags = { Name = "web-server" }
}
```

Resource Meta-Arguments

`depends_on`	Explicit dependency on another resource
`count`	Create multiple instances (`count = 3`)
`for_each`	Create instances from a map or set
`provider`	Select a non-default provider alias
`lifecycle`	Customize create/update/destroy behavior

Referencing Resources

```
# type.name.attribute
aws_instance.web.id
aws_instance.web.public_ip
aws_vpc.main.cidr_block
```

Variables

Declaring Variables

```
variable "region" {
  type    = string
  default = "us-east-1"
}
variable "instance_count" {
  type        = number
  description = "Number of instances"
}
```

Setting Variable Values

`-var 'region=us-west-2'`	CLI flag
`-var-file=prod.tfvars`	Load from a `.tfvars` file
`terraform.tfvars`	Auto-loaded if present
`TF_VAR_region`	Environment variable
`Interactive prompt`	Asked at plan/apply if no default

Variable Types

`string`	`"us-east-1"`
`number`	`42`
`bool`	`true` / `false`
`list(string)`	`["a", "b"]`
`map(string)`	`{ key = "val" }`
`object({...})`	Structured type with named attributes

Outputs

Defining Outputs

```
output "instance_ip" {
  value       = aws_instance.web.public_ip
  description = "Public IP of the web server"
}
output "db_password" {
  value     = random_password.db.result
  sensitive = true
}
```

Output Commands

`terraform output`	Print all outputs
`terraform output instance_ip`	Print a specific output
`terraform output -json`	JSON format for scripting
`sensitive = true`	Hide value from CLI output
`module.vpc.vpc_id`	Access child module outputs

State

Remote Backend

```
terraform {
  backend "s3" {
    bucket = "my-tf-state"
    key    = "prod/terraform.tfstate"
    region = "us-east-1"
  }
}
```

State Commands

`terraform state list`	List all resources in state
`terraform state show <addr>`	Show attributes of a resource
`terraform state mv <src> <dst>`	Rename / move a resource in state
`terraform state rm <addr>`	Remove resource from state (keep infra)
`terraform state pull`	Download remote state to stdout
`terraform import <addr> <id>`	Import existing infra into state

Modules

Using Modules

```
module "vpc" {
  source  = "terraform-aws-modules/vpc/aws"
  version = "~> 5.0"
  cidr    = "10.0.0.0/16"
}
```

Module Sources

`"./modules/vpc"`	Local path
`"terraform-aws-modules/vpc/aws"`	Terraform Registry
`"github.com/org/repo//module"`	GitHub repository
`"s3::https://bucket/module.zip"`	S3 bucket

Module Structure

```
modules/vpc/
  main.tf         # resources
  variables.tf    # input variables
  outputs.tf      # output values
```

Data Sources

Reading Existing Resources

```
data "aws_ami" "ubuntu" {
  most_recent = true
  filter {
    name   = "name"
    values = ["ubuntu/images/hvm-ssd/*"]
  }
  owners = ["099720109477"]
}
```

Common Data Sources

`data.aws_ami`	Look up an AMI by filters
`data.aws_vpc`	Look up existing VPC
`data.aws_caller_identity`	Current AWS account ID
`data.aws_region`	Current AWS region
`data.terraform_remote_state`	Read outputs from another state file
`data.external`	Run an external program for data

Lifecycle

Lifecycle Rules

```
resource "aws_instance" "web" {
  lifecycle {
    create_before_destroy = true
    prevent_destroy       = true
    ignore_changes        = [tags]
  }
}
```

Lifecycle Options

`create_before_destroy`	Create replacement before destroying old
`prevent_destroy`	Error if `terraform destroy` targets this
`ignore_changes`	Don't detect drift on listed attributes
`replace_triggered_by`	Force replacement when referenced resource changes
`precondition`	Validate assumptions before apply
`postcondition`	Validate results after apply

Common Patterns

Loops & Conditionals

```
# for_each with a map
resource "aws_iam_user" "users" {
  for_each = toset(["alice", "bob"])
  name     = each.value
}
# conditional resource
count = var.create_db ? 1 : 0
```

Useful Functions

`file("key.pub")`	Read file contents
`join(", ", list)`	Join list into string
`lookup(map, key, default)`	Map lookup with fallback
`length(list)`	Number of elements
`toset(["a", "b"])`	Convert list to set (for for_each)
`try(expr, fallback)`	Return fallback if expr errors
`templatefile(path, vars)`	Render a template file