terraform_cheatsheet

Terraform describes infrastructure as code: init downloads providers, plan shows what will change, apply makes it so, and the state file remembers what exists. These are the commands you run every day, grouped by task. OpenTofu accepts the same commands: switch to tofu or a tf alias and every command updates. Click a command to copy it.

:workflow

7 cmds

Init, plan, apply, destroy.

Download providers and set up the backend

```
$ terraform init
```

Show what will change

```
$ terraform plan
```

Apply the changes

```
$ terraform apply
```

Apply without the confirmation prompt

```
$ terraform apply -auto-approve
```

Save a plan, then apply exactly that plan

```
$ terraform plan -out=tfplan   $ terraform apply tfplan
```

Preview a destroy

```
$ terraform plan -destroy
```

Destroy everything in the configuration

```
$ terraform destroy
```

:plan

8 cmds

Variables, targets and replacements.

Set a variable

```
$ terraform plan -var='region=us-east-1'
```

Use a variables file

```
$ terraform plan -var-file=prod.tfvars
```

Plan only one resource and its dependencies

```
$ terraform plan -target=aws_instance.web
```

Force a resource to be recreated

```
$ terraform apply -replace=aws_instance.web
```

Sync state with real infrastructure only

```
$ terraform apply -refresh-only
```

Exit code 2 when there are changes (for CI)

```
$ terraform plan -detailed-exitcode
```

Wait for the state lock

```
$ terraform apply -lock-timeout=5m
```

Run more operations at once

```
$ terraform apply -parallelism=20
```

:init

8 cmds

Backends, provider versions and modules.

Upgrade providers and modules within their constraints

```
$ terraform init -upgrade
```

Reconfigure the backend, ignoring the old one

```
$ terraform init -reconfigure
```

Move state to a new backend

```
$ terraform init -migrate-state
```

Pass backend settings from a file

```
$ terraform init -backend-config=backend.hcl
```

List the providers in use

```
$ terraform providers
```

Lock provider hashes for several platforms

```
$ terraform providers lock -platform=linux_amd64 -platform=darwin_arm64
```

Update modules

```
$ terraform get -update
```

Show the CLI and provider versions

```
$ terraform version
```

:state

8 cmds

What Terraform thinks exists.

List resources in state

```
$ terraform state list
```

Show one resource

```
$ terraform state show aws_instance.web
```

Show the whole state or a saved plan

```
$ terraform show   $ terraform show -json tfplan
```

Rename or move a resource

```
$ terraform state mv aws_instance.web aws_instance.app
```

Move a resource into a module

```
$ terraform state mv aws_instance.web module.app.aws_instance.web
```

Stop managing a resource without destroying it

```
$ terraform state rm aws_instance.web
```

Download the remote state

```
$ terraform state pull > terraform.tfstate
```

Release a stuck lock

```
$ terraform force-unlock LOCK_ID
```

:import

3 cmds

Bring existing infrastructure under Terraform.

Import a resource into state

```
$ terraform import aws_instance.web i-0abc123def456
```

Write config for import blocks

```
$ terraform plan -generate-config-out=generated.tf
```

Replace a provider source in state

```
$ terraform state replace-provider hashicorp/aws registry.example.com/acme/aws
```

:workspaces

5 cmds

Several states for one configuration.

List workspaces

```
$ terraform workspace list
```

Show the current workspace

```
$ terraform workspace show
```

Create a workspace

```
$ terraform workspace new staging
```

Switch workspace

```
$ terraform workspace select staging
```

Delete a workspace

```
$ terraform workspace delete staging
```

:outputs

5 cmds

Read values back out.

Show all outputs

```
$ terraform output
```

Print one output without quotes

```
$ terraform output -raw db_endpoint
```

Outputs as JSON

```
$ terraform output -json
```

Try expressions against the state

```
$ terraform console
```

Draw the dependency graph

```
$ terraform graph | dot -Tsvg > graph.svg
```

:code

5 cmds

Keep the configuration clean and correct.

Format files in the current folder

```
$ terraform fmt
```

Check formatting everywhere (for CI)

```
$ terraform fmt -recursive -check
```

Validate the configuration

```
$ terraform validate
```

Run .tftest.hcl tests

```
$ terraform test
```

Log in to HCP Terraform

```
$ terraform login
```

:debug

5 cmds

Environment variables that change how the CLI behaves.

Debug logs

```
$ TF_LOG=DEBUG terraform plan
```

Trace logs to a file

```
$ TF_LOG=TRACE TF_LOG_PATH=./tf.log terraform apply
```

Set a variable from the environment

```
$ TF_VAR_region=us-east-1 terraform plan
```

Pick the workspace from the environment

```
$ TF_WORKSPACE=staging terraform plan
```

Run in another folder

```
$ terraform -chdir=envs/prod plan
```

Updated Nov 8, 2026 · checked against Terraform 1.x and OpenTofu 1.x · sources: Terraform CLI commands, OpenTofu CLI commands

Made by Lucas Bittencourt. Free, no ads, no tracking. Spotted a mistake? Tell me.