OpenTofu Open-source infrastructure as code

START HERE

`tofu init`
Download providers and modules

`tofu state list`
List tracked resources

`tofu fmt`
Format the files

`tofu destroy`
Remove everything managed

`tofu validate`
Check the configuration

`tofu plan -out=<file>`
Preview and save changes

`tofu apply <file>`
Apply the saved plan

`tofu output`
Show output values

Config .tf files → `init` → **Working dir** providers, modules → `plan` → **Plan** diff vs. state → `apply` → **Infrastructure** state updated

CONCEPTS

OpenTofu is the open-source fork of Terraform: same HCL and workflow, command is tofu. Configuration and state stay compatible for typical setups.

Always read the plan before applying. Saving it with -out guarantees apply does exactly what you reviewed.

State may hold secrets. Keep it out of git, use a remote backend, and consider OpenTofu's built-in state encryption.

WORKFLOW

`tofu init`
Prepare the working directory
Run again after adding a provider, module or backend.

`tofu init -upgrade`
Update providers and modules

`tofu init -reconfigure`
Ignore saved backend setup

`tofu init -migrate-state`
Move state to a new backend

`tofu validate`
Check configuration syntax

`tofu plan`
Preview changes

`tofu plan -out=<file>`
Save the plan to a file

WORKFLOW (cont.)

`tofu apply`
Create or update infrastructure

`tofu apply <planfile>`
Apply a saved plan

`tofu apply -auto-approve`
Skip the confirmation prompt

`tofu destroy`
Remove all managed resources

`tofu plan -destroy`
Preview a destroy

`tofu plan -refresh-only`
Preview drift against reality

`tofu fmt`
Format files in place

`tofu fmt -recursive -check`
Check formatting in CI

TARGETING & PLANS

`tofu plan -target=<addr>`
Limit to one resource

`tofu plan -exclude=<addr>`
Skip a resource and its dependents
Documented for exceptional cases only.

`tofu apply -replace=<addr>`
Force recreating a resource

`tofu plan -refresh=false`
Skip refreshing real state

`tofu plan -parallelism=<n>`
Limit concurrent operations

TARGETING & PLANS (cont.)

`tofu plan -detailed-exitcode`
Exit 2 when changes exist

`tofu plan -generate-config-out=<file>`
Draft HCL for import blocks
Experimental; the file must not exist yet.

`tofu show <planfile>`
Read a saved plan

`tofu show -json <planfile>`
Plan as JSON

`tofu graph`
Emit a Graphviz dependency graph

VARIABLES & OUTPUTS

`tofu plan -var '<name>=<value>'`
Set one variable

`tofu plan -var-file=<file>`
Load variables from a file

`TF_VAR_<name>=<value>`
Set a variable via environment

`terraform.tfvars`
Auto-loaded variable file

`tofu output`
Show all outputs

`tofu output <name>`
Show one output

VARIABLES & OUTPUTS (cont.)

`tofu output -raw <name>`
Plain value for scripts

`tofu output -json`
All outputs as JSON

`tofu console`
Try expressions interactively

STATE

`tofu state list`
List tracked resources

`tofu state show <addr>`
Show one resource's attributes

`tofu state mv <src> <dst>`
Rename or move in state

`tofu state rm <addr>`
Forget a resource, keep the real one

`tofu state pull`
Print remote state as JSON

`tofu state push <file>`
Overwrite state from a file

`tofu import <addr> <id>`
Adopt an existing resource

`tofu taint <addr>`
Mark for recreation

`tofu untaint <addr>`
Clear the taint mark

`tofu force-unlock <id>`
Release a stuck state lock

`tofu refresh`
Sync state with reality

WORKSPACES

`tofu workspace list`
List workspaces

`tofu workspace show`
Show the current workspace

`tofu workspace new <name>`
Create and switch

`tofu workspace select <name>`
Switch workspace

`tofu workspace delete <name>`
Delete an empty workspace

Each workspace has its own state under the same configuration; it is not a strong isolation boundary.

TESTING & MODULES

`tofu test`
Run *.tftest.hcl test files
Creates real infrastructure, then destroys it.

`tofu test -filter=<file>`
Run one test file

`tofu test -verbose`
Print plan or state per run

`tofu test -test-directory=<dir>`
Use another test folder

`tofu providers`
Show required providers

TESTING & MODULES (cont.)

`tofu get`
Fetch remote modules

`tofu metadata functions -json`
List built-in functions

`tofu login`
Store credentials for a registry host

`tofu version`
Show the OpenTofu version

`tofu -chdir=<dir> <cmd>`
Run in another directory

RECIPES

Reviewed deploy

```
tofu init
tofu fmt -check && tofu validate
tofu plan -out=tfplan
tofu show tfplan
tofu apply tfplan
```

Per-environment variables

```
tofu workspace select prod
tofu plan -var-file=prod.tfvars -out=prod.plan
tofu apply prod.plan
```

Rename without recreating

```
tofu state mv aws_instance.old aws_instance.new
tofu plan   # expect no changes
```

Replace one broken resource

```
tofu plan -replace=aws_instance.web -out=fix.plan
tofu apply fix.plan
```