Terraform Infrastructure as code

START HERE

`terraform` **`init`**
Download providers and modules

`terraform` **`state list`**
List tracked resources

`terraform` **`fmt`**
Format the files

`terraform` **`destroy`**
Remove everything managed

`terraform` **`validate`**
Check the configuration

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

`terraform` **`apply`** *`<file>`*
Apply the saved plan

`terraform` **`output`**
Show output values

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

CONCEPTS

Terraform compares your configuration with its state file, not with the real world alone. Lose the state and it forgets what it manages.

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

Never edit the state file by hand and keep it out of git; use a remote backend for teams.

WORKFLOW

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

`terraform` **`init -upgrade`**
Update providers and modules

`terraform` **`init -reconfigure`**
Ignore saved backend setup

`terraform` **`init -migrate-state`**
Move state to a new backend

`terraform` **`validate`**
Check syntax and consistency

`terraform` **`plan`**
Preview changes

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

WORKFLOW (cont.)

`terraform` **`apply`**
Plan, confirm, then apply

`terraform` **`apply`** *`<planfile>`*
Apply a saved plan

`terraform` **`apply -auto-approve`**
Skip the confirmation prompt
Meant for CI; never use it without a reviewed plan.

`terraform` **`destroy`**
Destroy all managed resources

`terraform` **`plan -destroy`**
Preview a destroy

`terraform` **`apply -refresh-only`**
Sync state with reality

`terraform` **`fmt`**
Rewrite files in canonical style

`terraform` **`fmt -recursive -check`**
Check formatting incl. subdirs

TARGETING & PLANS

`terraform` **`plan -target=`***`<addr>`*
Limit to one resource
For exceptions only; it can leave the rest out of sync.

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

TARGETING & PLANS (cont.)

`terraform` **`plan -refresh=false`**
Skip reading real state

`terraform` **`plan -parallelism=`***`<n>`*
Limit concurrent operations

`terraform` **`plan -detailed-exitcode`**
Exit 2 when changes exist

`terraform` **`show`** *`<planfile>`*
Display a saved plan

`terraform` **`show -json`** *`<planfile>`*
Plan as JSON

`terraform` **`graph`**
Dependency graph (DOT)

VARIABLES & OUTPUTS

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

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

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

`terraform.tfvars`
Loaded automatically
**.auto.tfvars files are also loaded automatically.*

VARIABLES & OUTPUTS (cont.)

`terraform` **`output`**
Show all outputs

`terraform` **`output`** *`<name>`*
Show a single output

`terraform` **`output -raw`** *`<name>`*
Plain value, no quotes

`terraform` **`output -json`**
Outputs as JSON

`terraform` **`console`**
Try expressions interactively

STATE

`terraform` **`state list`**
List tracked resources

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

`terraform` **`state mv`** *`<src>`* *`<dst>`*
Rename or move in state

`terraform` **`state rm`** *`<addr>`*
Forget a resource, keep it real

`terraform` **`state pull`**
Print the state as JSON

`terraform` **`import`** *`<addr>`* *`<id>`*
Adopt an existing resource
The resource block must exist in the config first; an import block is the declarative alternative.

STATE (cont.)

`terraform` **`taint`** *`<addr>`*
Mark for replacement
Superseded by apply -replace.

`terraform` **`untaint`** *`<addr>`*
Clear the taint mark

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

WORKSPACES

`terraform` **`workspace list`**
List workspaces

`terraform` **`workspace show`**
Show the current one

`terraform` **`workspace new`** *`<name>`*
Create and switch

`terraform` **`workspace select`** *`<name>`*
Switch workspace

`terraform` **`workspace delete`** *`<name>`*
Delete a workspace

Each workspace has its own state under the same configuration. Use ${terraform.workspace} to vary names.

PROVIDERS & MODULES

`terraform` **`providers`**
List required providers

PROVIDERS & MODULES (cont.)

`terraform` **`providers lock`**
Write the dependency lock file

`terraform` **`get`**
Download modules

`terraform` **`version`**
Show Terraform version

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

RECIPES

Reviewed deploy

```
terraform init
terraform fmt -check
terraform validate
terraform plan -out=tfplan
terraform apply tfplan
```

Per-environment variables

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

Rename without recreating

```
terraform state mv
aws_instance.old
aws_instance.new
# update the resource name in
the .tf file
terraform plan   # expect no
changes
```

Replace one broken resource

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