Terraform Infrastructure as code

CONCEPTS

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

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

> 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

`terraform init -upgrade`
Update providers and modules

`terraform validate`
Check syntax and consistency

`terraform plan`
Preview changes

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

`terraform apply`
Plan, confirm, then apply

`terraform apply <planfile>`
Apply a saved plan

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

`terraform destroy`
Destroy all managed resources

WORKFLOW (cont.)

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

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 output`
Show all outputs

`terraform output <name>`
Show a single output

STATE

`terraform state list`
List tracked resources

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

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

PROVIDERS & MODULES

`terraform version`
Show Terraform version