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 init -reconfigure`
Ignore saved backend setup

`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

`terraform plan -destroy`
Preview a destroy

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

`terraform fmt`
Rewrite files in canonical style

WORKFLOW (cont.)

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

TARGETING & PLANS

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

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

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

`terraform show <planfile>`
Display a saved plan

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

`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

STATE (cont.)

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

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

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

PROVIDERS & MODULES

`terraform providers`
List required providers

`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
```