Terragrunt Terraform orchestration

CONCEPTS

Terragrunt is a wrapper around OpenTofu/Terraform. A unit is a directory with a terragrunt.hcl; a stack is a tree of units run together. Unknown commands such as plan pass through to tofu/terraform.

Discover find units in tree
↓ filter
Order DAG from dependencies
↓ per unit
Init + run auto-init, then tofu cmd
↓ state
Result summary of all units

Use -- to separate Terragrunt flags from tofu/terraform arguments: terragrunt run -- plan -no-color. The old run-all and hclfmt forms are deprecated in favour of run -- all and hcl fmt.

--all adds -auto-approve to apply and destroy automatically. Add --no-auto-approve if you want the prompt back.

RUN

terragrunt run plan
Plan the unit in this directory

terragrunt run apply
Apply the unit in this directory

terragrunt run init
Initialise the unit

terragrunt run destroy
Destroy the unit

terragrunt run output
Show unit outputs

terragrunt plan
Shortcut for run plan

terragrunt run --all plan
Plan all units below here

terragrunt run --all apply
Apply all units in order

terragrunt run --all destroy
Destroy all, dependents first

terragrunt run --graph plan
Plan this unit and its dependents

RUN (cont.)

terragrunt run --all --parallelism *<n>* **plan**
Cap concurrent units

terragrunt run --all --no-auto-approve apply
Keep the apply prompt

terragrunt run --source *<path>* **plan**
Override the module source

terragrunt run --tf-path *<bin>* **plan**
Pick tofu or terraform binary

SELECT UNITS

--filter '*<glob>***'**
Only units matching the path

--filter '!*<glob>***'**
Exclude matching units

--working-dir *<dir>*
Run as if in another directory

CONFIG BLOCKS

Parent terragrunt.hcl: backend and shared code

```
remote_state {
  backend = "s3"
  generate = { path =
"backend.tf", if_exists =
"overwrite_terragrunt" }
  config = {
    bucket = "my-state"
    key    =
"${path_relative_to_include()}/
tofu.tfstate"
    region = "eu-central-1"
  }
}
```

Block purpose

Block	Use
include	Inherit a parent file
inputs	Variables passed to the module
dependency	Outputs of another unit
dependencies	Ordering without outputs
remote_state	Backend setup
generate	Write helper files
locals	Named values in the file
feature	Flags set with --feature

mock_outputs let plan work before the dependency has been applied. Without them, plan on a fresh stack fails on missing outputs.

FUNCTIONS

find_in_parent_folders()
Nearest parent terragrunt file

path_relative_to_include()
Unit path relative to parent

get_terragrunt_dir()
Directory of this terragrunt.hcl

get_env("*<NAME>***", "***<default>***")**
Read an environment variable

DISCOVER

terragrunt find
List units and stacks

terragrunt find --json
Machine-readable list

terragrunt find --dag
Sort by dependency order

terragrunt list
List configurations

terragrunt render --json
Resolved config, includes merged

FORMAT AND VALIDATE

terragrunt hcl fmt
Format HCL files in place

terragrunt hcl fmt --check
Fail if formatting differs

terragrunt hcl validate
Validate HCL files

STACKS AND BACKEND

terragrunt stack generate
Create units from stack.hcl

terragrunt stack run *<cmd>*
Run a command on a stack

terragrunt backend bootstrap
Create the state backend

RECIPES

Plan only a subset

```
terragrunt run --all --filter
'prod/**' -- plan
terragrunt run --all --filter '!
prod/legacy/**' -- plan
terragrunt run --all --
parallelism 4 -- plan -no-color
```