Nomad Workload orchestrator

START HERE

`nomad agent -dev`
Throwaway local cluster

`nomad alloc logs <alloc> <task>`
Read task output

`export NOMAD_ADDR=<url>`
Point the CLI at a server

`nomad job stop <job>`
Stop the job

`nomad job init`
Write an example job file

`nomad job plan <file>`
Dry run, show the diff

`nomad job run <file>`
Submit the job

`nomad job status <job>`
Allocations and health

Job your spec file → *evaluation* → **Eval** scheduler run → *places* → **Allocation** group on a node → *runs* → **Task** container, process → *logs, exec* →

CONCEPTS

You submit a job; Nomad evaluates it and places allocations on client nodes. Status commands walk that chain.

job plan exits 0 for no change, 1 when allocations would change, 255 on error. Handy in CI.

Job, alloc and node IDs can be shortened to a unique prefix.

CONNECT

`export NOMAD_ADDR=<url>`
Set the server address
Default is http://127.0.0.1:4646

`export NOMAD_TOKEN=<secret-id>`
Authenticate with an ACL token
Same as -token; the flag wins

`export NOMAD_NAMESPACE=<ns>`
Work in a namespace
*Default is default; * means all authorized*

`nomad server members`
List servers and leader

`nomad version`
Print the CLI version

RUN JOBS

`nomad job init`
Write example.nomad.hcl

`nomad job validate <file>`
Check a job file offline

`nomad job plan <file>`
Dry run with a diff
Prints the check-index to pass to job run

`nomad job run <file>`
Submit or update a job

`nomad job run -detach <file>`
Submit without monitoring
Prints the evaluation ID

`nomad job run -check-index <n> <file>`
Submit only if unchanged since plan

`nomad job run -var <key>=<value> <file>`
Set a job variable

`nomad job run -preserve-counts <file>`
Keep group counts when updating

INSPECT JOBS

`nomad status`
List all jobs

`nomad job status <job>`
Summary, allocations, deployment

`nomad job inspect <job>`
Show the stored job as JSON

INSPECT JOBS (cont.)

`nomad job history <job>`
List versions of a job

`nomad job allocs <job>`
List allocations of a job

CHANGE AND STOP

`nomad job stop <job>`
Stop a job
Job stays visible as dead

`nomad job stop -purge <job>`
Stop and remove from the state

`nomad job scale <job> <group> <count>`
Set a task group count

`nomad job revert <job> <version>`
Roll back to an older version

`nomad job restart <job>`
Restart allocations in place

`nomad job promote <job>`
Promote canaries

ALLOCATIONS

`nomad alloc status <alloc>`
Task states and events

`nomad alloc logs <alloc> <task>`
Print task stdout

`nomad alloc logs -f <alloc> <task>`
Follow logs

ALLOCATIONS (cont.)

`nomad alloc logs -stderr <alloc> <task>`
Print task stderr

`nomad alloc logs -job <job>`
Logs of a random alloc of the job
Add -task or -group when the job has several

`nomad alloc exec <alloc> <cmd>`
Run a command in a task
Add -task <task> if the group has several tasks

`nomad alloc exec -job <job> /bin/sh`
Shell into a job's task

`nomad alloc fs <alloc> [<path>]`
Browse the allocation directory

`nomad alloc restart <alloc>`
Restart a task in place

NODES

`nomad node status`
List client nodes

`nomad node status <node>`
Resources and allocations of a node

`nomad node drain -enable <node>`
Migrate allocations away
Deadline defaults to 1h; add -yes to skip the prompt

NODES (cont.)

`nomad node drain -disable <node>`
Cancel a drain

`nomad node eligibility -disable <node>`
Stop new placements

`nomad node eligibility -enable <node>`
Allow placements again

EVALS AND DEPLOYMENTS

`nomad eval status <eval>`
Why a placement failed
Look for the failed placement section

`nomad deployment status <id>`
Progress of a rolling update

`nomad deployment list`
List deployments

`nomad deployment promote <id>`
Promote healthy canaries

VARIABLES AND NAMESPACES

`nomad var put <path> <key>=<value>`
Write a variable

`nomad var get <path>`
Read a variable

`nomad var list [<prefix>]`
List variables

VARIABLES AND NAMESPACES (cont.)

`nomad namespace list`
List namespaces

ACL AND OPERATE

`nomad acl bootstrap`
Create the first management token
Works once per cluster; keep the secret

`nomad operator raft list-peers`
Show the Raft peers

RECIPES

Plan, then apply exactly that plan

```
nomad job plan web.nomad.hcl
# copy the Job Modify Index from the output
nomad job run -check-index <n> web.nomad.hcl
```

Drain a node for maintenance

```
nomad node drain -enable -yes <node>
nomad node status <node>
# ...maintenance...
nomad node drain -disable <node>
nomad node eligibility -enable <node>
```

Debug a failing task

```
nomad job status <job>
nomad alloc status <alloc>
nomad alloc logs -stderr <alloc> <task>
nomad alloc exec -task <task> <alloc> /bin/sh
```