Helmfile Declarative Helm releases

helmfile.yaml desired state → `template + values` → **Helm releases** charts + values → `diff / apply / sync` → **Cluster** kube-context

CONCEPTS

helmfile.yaml declares which Helm releases belong on a cluster; Helmfile makes the cluster match it.

apply runs diff first and syncs only when there are changes; sync always upgrades every selected release.

Needs the helm-diff plugin; helmfile init offers to install it.

PROJECT

helmfile init
Check helm and install plugins

helmfile create *<name>*
Scaffold a project layout

helmfile version
Print the Helmfile version

helmfile repos
Add the repositories from the file

helmfile deps
Update chart deps and lock files

helmfile list
List releases in the state file

PROJECT (cont.)

helmfile show-dag
Show deployment order

helmfile lint
Run helm lint on the charts

DEPLOY

helmfile diff
Show pending changes

helmfile apply
Diff, then sync only if changed

helmfile apply -i
Confirm before changing the cluster

helmfile sync
Upgrade --install every release

helmfile sync --wait --timeout *<sec>*
Wait until resources are ready

helmfile template
Render manifests locally

helmfile destroy
Uninstall and purge releases

helmfile status
Run helm status per release

helmfile apply --skip-deps
Skip repo update and dep build

TARGETING

helmfile -f *<file or dir> <cmd>*
Use another helmfile or directory

helmfile -e *<env> <cmd>*
Choose environment (default: default)

helmfile -l name=*<release> <cmd>*
Select one release

helmfile -l *<key>=<val>,<key2>=<val2> <cmd>*
Labels must all match (AND)

helmfile -l *<k>=<a>* **-l** *<k>=<b> <cmd>*
Repeated selectors match any (OR)

helmfile -l *<key>≠<val> <cmd>*
Exclude by label

helmfile -l namespace=*<ns> <cmd>*
Select by namespace

helmfile -l chart=*<chart> <cmd>*
Select by chart name

helmfile -n *<ns> <cmd>*
Set the namespace

helmfile --kube-context *<ctx> <cmd>*
Use a kubectl context

STATE VALUES & OUTPUT

helmfile --state-values-set *<key>=<val> <cmd>*
Override a state value

helmfile build
Print the fully rendered state

helmfile --debug *<cmd>*
Verbose helm output

HELMFILE.YAML

Minimal file

```
repositories:
  - name: bitnami
    url: https://charts.
bitnami.com/bitnami
---
releases:
  - name: cache
    namespace: data
    chart: bitnami/redis
    version: <x.y.z>
    values:
      - values/redis.yaml
```

Environments

```
environments:
  default:
    values: [env/
default.yaml]
  production:
    values: [env/
production.yaml]
---
# in releases:
  installed:
{{ eq .Environment.Name
"production" | toYaml }}
```

HELMFILE.YAML (cont.)

Defaults for all releases

```
helmDefaults:
  wait: true
  timeout: 600
  createNamespace: true
  historyMax: 10
```

Dependencies and toggles

```
releases:
  - name: app
    chart: ./charts/app
    needs: [data/cache]
    condition: app.enabled
    labels: {tier: web}
```

Split across files

```
bases:
  - environments.yaml
helmfiles:
  - path: apps/*/
helmfile.yaml
    missingFileHandler: Warn
```

Put .gotmpl on any file that uses Go templating, e.g. values.yaml.gotmpl; use {{ .Values.x }} for environment values.

RECIPES

Review then ship

```
helmfile -e staging diff
helmfile -e staging apply -i
```

Target one release

```
helmfile -e production -l
name=<release> diff
helmfile -e production -l
name=<release> apply
```