Helmfile Declarative Helm releases

START HERE

`helmfile` **`init`**
Check helm and plugins

`helmfile` **`sync`**
Sync all releases

`helmfile` **`repos`**
Add chart repositories

`helmfile` **`-e`** *`<env>`* **`apply`**
Pick an environment

`helmfile` **`diff`**
Preview changes

`helmfile` **`-l`** *`<key>=<val>`* **`apply`**
Only matching releases

`helmfile` **`apply`**
Deploy what changed

`helmfile` **`destroy`**
Remove all 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

`helmfile` **`list --output json`**
List releases as JSON

`helmfile` **`show-dag`**
Show deployment order

`helmfile` **`lint`**
Run helm lint on the charts

`helmfile` **`completion`** *`<shell>`*
Generate shell completion

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` **`diff --skip-diff-on-install`**
Skip the diff for new releases

DEPLOY (cont.)

`helmfile` **`template`**
Render manifests locally

`helmfile` **`destroy`**
Uninstall and purge releases

`helmfile` **`status`**
Run helm status per release

`helmfile` **`test`**
Run helm test on releases

`helmfile` **`write-values`**
Write values files per release

`helmfile` **`fetch --output-dir`** *`<dir>`*
Download charts for inspection

`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` **`-l dir=`***`<path> <cmd>`*
Select by defining directory

`helmfile` **`--allow-no-matching-release`** *`<cmd>`*
No error if nothing matches

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

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

TARGETING (cont.)

`helmfile` **`--kubeconfig`** *`<file> <cmd>`*
Use a kubeconfig file

STATE VALUES & OUTPUT

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

`helmfile` **`--state-values-file`** *`<file> <cmd>`*
Load state values from a file

`helmfile` **`print-env`**
Show parsed environment and values

`helmfile` **`build`**
Print the fully rendered state

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

`helmfile` **`-q`** *`<cmd>`*
Only warnings and errors

`helmfile` **`--no-color`** *`<cmd>`*
Plain output for CI

`helmfile` **`-f -`** *`<cmd>`*
Read the helmfile from stdin

`helmfile` **`-b`** *`<path> <cmd>`*
Use a specific helm binary

`helmfile` **`cache cleanup`**
Remove cached files

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 }}
```

Defaults for all releases

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

HELMFILE.YAML (cont.)

Dependencies and toggles

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

Hooks

```
hooks:
  - events: ["presync"]
    showlogs: true
    command: ./check.sh
  - events: ["postsync"]
    kubectlApply: {filename:
extra.yaml}
```

Split across files

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

Hook events

Event	Runs
`prepare`	after a release is loaded
`presync`	before install or upgrade
`postsync`	after install or upgrade
`preuninstall`	before uninstall
`postuninstall`	after uninstall
`cleanup`	after each release

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

Release fields such as installed: false uninstall the release on sync, handy for per-environment toggles.

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
```

Debug the rendered output

```
helmfile -e <env> template >
out.yaml
helmfile -e <env> build
helmfile show-dag
```

Bootstrap a project

```
helmfile create <name>
cd <name> && helmfile init
helmfile repos && helmfile diff
```