Helm Kubernetes package manager

START HERE

`helm repo add <name> <url>`
Add a chart repository

`helm list`
Show installed releases

`helm search repo <keyword>`
Find a chart

`helm upgrade <release> <chart>`
Apply new chart or values

`helm show values <chart>`
See what you can configure

`helm rollback <release>`
Go back one revision

`helm install <release> <chart>`
Deploy a chart

`helm uninstall <release>`
Remove a release

Repository chart index → `helm pull` / `install` → **Chart + values** templates → `helm template` → **Manifests** rendered YAML → `helm install` → **Release** revisions in cluster

CONCEPTS

A chart is the package, a release is one installation of it. Each upgrade creates a new numbered revision.

Helm uses your current kubectl context. Add -n <namespace> to every command, or releases land in the default namespace.

Docs now show Helm 4: --atomic became --rollback-on-failure. Helm 3 still uses --atomic.

REPOSITORIES

`helm repo add <name> <url>`
Add a chart repository

`helm repo update`
Refresh local chart indexes
Run it before searching or installing to see new chart versions.

`helm repo list`
List configured repositories

`helm search repo <keyword>`
Search added repositories

`helm search repo <chart> --versions`
List all versions of a chart

`helm search hub <keyword>`
Search Artifact Hub

`helm repo remove <name>`
Remove a repository

`helm pull <chart> --untar`
Download and unpack a chart

INSPECT CHARTS

`helm show values <chart>`
Print the default values
Save it to a file as a starting point for your own overrides.

`helm show readme <chart>`
Show the chart README

`helm show chart <chart>`
Show chart metadata

`helm show crds <chart>`
Show bundled CRDs

INSTALL

`helm install <release> <chart>`
Install a chart as a release

INSTALL (cont.)

`helm install <release> <chart> -n <ns> --create-namespace`
Install into a new namespace

`helm install <chart> --generate-name`
Let Helm pick the release name

`helm install <release> <chart> -f <values.yaml>`
Install with a values file
With several -f files the rightmost wins.

`helm install <release> <chart> --set <key>=<val>`
Override a single value
--set wins over -f. Use --set-string to force a string.

`helm install <release> <chart> --version <ver>`
Pin the chart version
Accepts an exact version or a range such as ^2.0.0.

`helm install <release> <chart> --wait --timeout 5m`
Wait until resources are ready

`helm install <release> <chart> --dry-run=client`
Simulate without a cluster
Use =server to validate against the cluster.

`helm install <release> <chart> --rollback-on-failure`
Undo a failed install
Helm 3 spelling: --atomic.

`helm install <release> ./<dir>`
Install a local chart directory

RELEASES

`helm list`
List releases in this namespace

`helm list -A`
List releases in all namespaces

`helm list --failed`
Show only failed releases

`helm status <release>`
Show release status and notes

`helm get values <release>`
Values you supplied

`helm get values <release> --all`
Values including chart defaults

`helm get manifest <release>`
The manifests as deployed

RELEASES (cont.)

`helm get notes <release>`
Print the release notes

`helm history <release>`
List revisions of a release

UPGRADE & ROLLBACK

`helm upgrade <release> <chart> -f <values.yaml>`
Upgrade with a values file

`helm upgrade --install <release> <chart>`
Install or upgrade in one command
The usual form in scripts and CI: it works whether or not the release exists.

`helm upgrade <release> <chart> --set <key>=<val>`
Change one value
Without --reuse-values, values from earlier --set flags fall back to chart defaults.

`helm upgrade <release> <chart> --reuse-values`
Keep previous values, merge new

`helm upgrade <release> <chart> --dry-run=server`
Check an upgrade against the cluster

`helm upgrade <release> <chart> --rollback-on-failure`
Roll back if the upgrade fails
Helm 3 spelling: --atomic.

`helm rollback <release>`
Roll back to the previous revision

`helm rollback <release> <revision>`
Roll back to a given revision
Find the number with helm history. A rollback creates a new revision.

`helm uninstall <release>`
Remove a release

TEMPLATING & DEBUG

`helm template <release> <chart> -f <values.yaml>`
Render manifests locally

`helm template <release> <chart> -s templates/<file>`
Render a single template
-s is --show-only.

TEMPLATING & DEBUG (cont.)

`helm template <release> <chart> --debug`
Verbose output while rendering
Shows rendered YAML even when it is invalid.

`helm lint <chart-dir>`
Check a chart for problems

`helm create <name>`
Scaffold a new chart

`helm package <chart-dir>`
Build a chart archive (.tgz)

`helm dependency update <chart-dir>`
Fetch dependencies into charts/

Values precedence (low to high)

Source	Wins over
`chart values.yaml`	nothing
`-f file`	chart defaults
`later -f file`	earlier -f
`--set`	all files

RECIPES

Deploy a public chart

```
helm repo add bitnami https://
charts.bitnami.com/bitnami
helm repo update
helm show values bitnami/nginx >
values.yaml
# edit values.yaml
helm install web bitnami/nginx -
n web --create-namespace -f
values.yaml
```

Preview before upgrading

```
helm template web bitnami/nginx
-f values.yaml > new.yaml
helm get manifest web -n web >
old.yaml
diff old.yaml new.yaml
helm upgrade web bitnami/nginx -
n web -f values.yaml
```

Recover from a bad upgrade

```
helm history web -n web
helm rollback web <revision> -n
web
helm status web -n web
```

CI deploy step

```
helm upgrade --install web ./
chart \
  -n web --create-namespace \
  -f values-prod.yaml --wait --
timeout 10m
```

HELP

`helm <command> --help`
Help for a command

`helm version`
Print the client version