Helm CLI Cheat Sheet

Helm CLI Cheat Sheet

A practical quick-reference guide for Helm, the Kubernetes package manager. Commands are organized by workflow: installing, managing releases, working with charts, and troubleshooting.

Key Concepts

Before diving into commands, understand these core concepts:

Concept	Description
Chart	A package containing Kubernetes resource templates, default values, and metadata
Release	A specific instance of a chart running in a cluster with its own configuration
Revision	A numbered snapshot of a release, created on each install/upgrade/rollback
Repository	A collection of charts stored remotely (HTTP) or locally
Values	Configuration parameters that customize chart behavior

Values Files and Overriding

Helm merges values in this order (later overrides earlier):

1. Chart's `values.yaml` (defaults)
2. Parent chart values (for subcharts)
3. User-supplied `-f values.yaml` files (multiple allowed)
4. `--set` and `--set-file` arguments

Installation Commands

Basic Install

```
# Install a chart with default values
helm install my-release bitnami/nginx

# Install in a specific namespace (namespace must exist)
helm install my-release bitnami/nginx -n production
```

```
# Install with a custom release name from a local chart
helm install myapp ./my-chart
```

Install with Custom Values

```
# Override values from a file
helm install my-release bitnami/nginx -f values.yaml

# Override multiple values files (later files override earlier)
helm install my-release bitnami/nginx -f base.yaml -f override.yaml

# Set individual values inline
helm install my-release bitnami/nginx --set replicaCount=3

# Set nested values
helm install my-release bitnami/nginx --set image.tag=1.25.0

# Set multiple values
helm install my-release bitnami/nginx \
  --set replicaCount=3 \
  --set service.type=LoadBalancer
```

Install with Verification

```
# Dry run to see generated manifests without installing
helm install my-release bitnami/nginx --dry-run

# Dry run with debug output (shows computed values)
helm install my-release bitnami/nginx --dry-run --debug

# Verify chart before install (runs helm test after install)
helm install my-release bitnami/nginx --wait

# Install with timeout (default is 5m)
helm install my-release bitnami/nginx --timeout 10m
```

Install from Different Sources

```
# From OCI registry
helm install my-release oci://registry.example.com/charts/nginx:1.0.0

# From absolute URL
helm install my-release https://example.com/charts/nginx-1.0.0.tgz

# From a specific chart version
helm install my-release bitnami/nginx --version 15.0.0
```

Release Management

List Releases

```
# List releases in current namespace
helm list

# List releases in all namespaces
helm list -A

# List releases in a specific namespace
helm list -n production

# Show all releases including deleted (kept for history)
helm list --all

# Filter by release name pattern
helm list --filter 'my-*'

# Output in different formats
helm list -o json
helm list -o yaml
```

Release Status and History

```
# Get status of a release
helm status my-release

# Get status in a specific namespace
helm status my-release -n production

# View release history
helm history my-release

# History with maximum revisions
helm history my-release --max 10
```

Uninstall Releases

```
# Uninstall a release
helm uninstall my-release

# Uninstall from a specific namespace
helm uninstall my-release -n production

# Keep history after uninstall (allows rollback)
helm uninstall my-release --keep-history

# Dry run uninstall
helm uninstall my-release --dry-run
```

Upgrade and Rollback

Upgrade Releases

```
# Upgrade to latest chart version
helm upgrade my-release bitnami/nginx

# Upgrade with new values
helm upgrade my-release bitnami/nginx -f values.yaml

# Upgrade with inline values
helm upgrade my-release bitnami/nginx --set replicaCount=5

# Upgrade to a specific chart version
helm upgrade my-release bitnami/nginx --version 16.0.0

# Upgrade and install if release does not exist
helm upgrade --install my-release bitnami/nginx -f values.yaml

# Force resource updates through replacement
helm upgrade my-release bitnami/nginx --force
```

Upgrade with Safety Options

```
# Wait for resources to be ready
helm upgrade my-release bitnami/nginx --wait

# Wait with custom timeout
helm upgrade my-release bitnami/nginx --wait --timeout 10m

# Rollback on failure (automatic rollback if upgrade fails)
helm upgrade my-release bitnami/nginx --atomic

# Cleanup on failure (deletes new resources on failure)
helm upgrade my-release bitnami/nginx --cleanup-on-fail
```

Rollback Releases

```
# Rollback to previous revision
helm rollback my-release

# Rollback to a specific revision
helm rollback my-release 3

# Rollback in a specific namespace
helm rollback my-release 3 -n production

# Dry run rollback
helm rollback my-release --dry-run
```

Chart Operations

Create Charts

```
# Create a new chart skeleton
helm create my-chart
```

```
# Create with a specific starter
helm create my-chart --starter bitnami/nginx
```

Package Charts

```
# Package a chart directory into a .tgz file
helm package ./my-chart

# Package with a specific version
helm package ./my-chart --version 1.2.3

# Package with app version
helm package ./my-chart --app-version 2.0.0

# Package with signature
helm package ./my-chart --sign --key 'John Doe'
```

Lint and Validate

```
# Lint a chart for issues
helm lint ./my-chart

# Lint with strict mode (warnings as errors)
helm lint ./my-chart --strict

# Lint with values file
helm lint ./my-chart -f values.yaml
```

Pull and Download

```
# Download a chart without installing
helm pull bitnami/nginx

# Download a specific version
helm pull bitnami/nginx --version 15.0.0

# Download and extract
helm pull bitnami/nginx --untar

# Download to specific directory
helm pull bitnami/nginx -d ./charts/

# Download from OCI registry
helm pull oci://registry.example.com/charts/nginx:1.0.0
```

Inspect Charts

```
# Show all chart information
helm show all bitnami/nginx

# Show only chart.yaml contents
helm show chart bitnami/nginx
```

```
# Show default values
helm show values bitnami/nginx

# Show README
helm show readme bitnami/nginx

# Show from a packaged chart
helm show values ./nginx-15.0.0.tgz

# Combine outputs
helm show chart bitnami/nginx && helm show values bitnami/nginx
```

Repository Management

Add and List Repositories

```
# Add a chart repository
helm repo add bitnami https://charts.bitnami.com/bitnami

# Add with authentication
helm repo add private https://charts.example.com \
  --username user --password pass

# List configured repositories
helm repo list

# List output as JSON
helm repo list -o json
```

Update and Remove

```
# Update all repositories (fetch latest index)
helm repo update

# Update specific repository
helm repo update bitnami

# Remove a repository
helm repo remove bitnami

# Remove multiple repositories
helm repo remove bitnami stable
```

Index and Search

```
# Generate index file for a chart directory
helm repo index ./charts/

# Generate index with base URL
helm repo index ./charts/ --url https://charts.example.com

# Merge with existing index
```

```
helm repo index ./charts/ --merge index.yaml

# Search for charts by keyword
helm search repo nginx

# Search all repositories
helm search hub nginx

# Search with regex
helm search repo 'nginx$'

# Show all versions (not just latest)
helm search repo nginx --versions

# Filter by version constraint
helm search repo nginx --version '>=15.0.0'
```

Values Management

Inspect Values

```
# Get all computed values for a release
helm get values my-release

# Get values in a specific namespace
helm get values my-release -n production

# Get values for a specific revision
helm get values my-release --revision 3

# Get all values including defaults
helm get values my-release --all

# Output as YAML (default)
helm get values my-release -o yaml

# Output as JSON
helm get values my-release -o json
```

Set Values Syntax

```
# Simple value
--set name=value

# Nested value (uses dot notation)
--set image.tag=1.25.0

# Array/list values (use array index)
--set servers[0].name=web1
--set servers[1].name=web2

# Array shorthand
--set names={web1,web2,web3}

# Values with special characters (escape or quote)
```

```
--set "pod.annotations.kubernetes\.io/name"=myapp

# Read value from file
--set-file config=./config.txt

# Use string value (preserves type)
--set-string enabled=true
```

Values Files

```
# Use a values file
helm install my-release bitnami/nginx -f values.yaml

# Use multiple values files
helm install my-release bitnami/nginx \
  -f base.yaml \
  -f env/prod.yaml

# Generate values file template
helm show values bitnami/nginx > values.yaml
```

Template and Debug

Render Templates

```
# Render templates without installing
helm template my-release bitnami/nginx

# Render with values file
helm template my-release bitnami/nginx -f values.yaml

# Render to a directory
helm template my-release bitnami/nginx --output-dir ./rendered/

# Render specific templates only
helm template my-release bitnami/nginx -s templates/deployment.yaml

# Render with release name simulation
helm template my-release ./my-chart --release-name
```

Debug Commands

```
# Debug with dry run (shows computed values and manifests)
helm install my-release bitnami/nginx --dry-run --debug

# Debug template rendering issues
helm template my-release ./my-chart --debug

# Validate rendered manifests server-side
helm template my-release bitnami/nginx | kubectl apply --dry-run=client -f -
```

Dependency Management

Work with Dependencies

```
# Download chart dependencies
helm dependency update ./my-chart

# List dependencies
helm dependency list ./my-chart

# Build dependencies from Chart.lock
helm dependency build ./my-chart
```

Chart.yaml Dependencies

```
# Example Chart.yaml with dependencies
apiVersion: v2
name: my-app
version: 1.0.0
dependencies:
  - name: redis
    version: "17.x.x"
    repository: https://charts.bitnami.com/bitnami
    condition: redis.enabled
  - name: postgresql
    version: "12.x.x"
    repository: https://charts.bitnami.com/bitnami
    alias: database
```

Plugin Management

Install and Use Plugins

```
# Install a plugin
helm plugin install https://github.com/helm/helm-mapkubeapis

# Install from local directory
helm plugin install ./helm-plugin

# List installed plugins
helm plugin list

# Uninstall a plugin
helm plugin uninstall mapkubeapis

# Update a plugin
helm plugin update helm-diff
```

Release Information

Get Detailed Release Info

```
# Get user-supplied values
helm get values my-release

# Get rendered manifests
helm get manifest my-release

# Get release notes
helm get notes my-release

# Get hooks (job templates)
helm get hooks my-release

# Get everything combined
helm get all my-release

# Get from a specific revision
helm get manifest my-release --revision 2

# Get from specific namespace
helm get manifest my-release -n production
```

Testing

Test Releases

```
# Run helm tests for a release
helm test my-release

# Test with timeout
helm test my-release --timeout 5m

# Clean up test pods after running
helm test my-release --cleanup

# Test in a specific namespace
helm test my-release -n production

# View test logs
kubectl logs my-release-test-connection
```

Test Pod Example

Charts define tests in `templates/tests/`. A test hook looks like:

```
apiVersion: v1
kind: Pod
metadata:
  name: "{{ .Release.Name }}-test-connection"
  annotations:
    "helm.sh/hook": test
spec:
  containers:
```

```
    - name: wget
      image: busybox
      command: ['wget']
      args: ['{{ .Release.Name }}:{{ .Values.service.port }}']
  restartPolicy: Never
```

Common Patterns

Production Install Pattern

```
helm upgrade --install myapp ./chart \
  -n production \
  --create-namespace \
  -f values/production.yaml \
  --set image.tag=${CI_COMMIT_SHA} \
  --atomic \
  --timeout 10m \
  --wait \
  --history-max 10
```

Diff Before Upgrade (with helm-diff plugin)

```
# Preview changes before applying
helm diff upgrade my-release bitnami/nginx -f values.yaml

# Diff with three-way merge view
helm diff upgrade my-release bitnami/nginx --three-way-merge
```

Multi-Environment Values

```
# Directory structure
# values/
#   base.yaml
#   development.yaml
#   staging.yaml
#   production.yaml

helm upgrade --install myapp ./chart \
  -f values/base.yaml \
  -f values/production.yaml
```

Secret Management

```
# Set sensitive values from environment
helm install my-release bitnami/postgresql \
  --set postgresqlPassword="${DB_PASSWORD}" \
  --set-string postgresqlPassword="${DB_PASSWORD}"

# Use --set-file for secrets from files
helm install my-release ./chart \
  --set-file ssl.cert=/path/to/cert.pem
```

Quick Reference

Command	Purpose
`helm install`	Create a new release
`helm upgrade`	Update an existing release
`helm rollback`	Revert to a previous revision
`helm uninstall`	Delete a release
`helm list`	Show releases
`helm status`	Show release details
`helm history`	Show revision history
`helm get values`	Show release values
`helm get manifest`	Show rendered YAML
`helm repo add`	Add a chart repository
`helm repo update`	Fetch latest chart indexes
`helm search`	Find charts
`helm pull`	Download a chart
`helm template`	Render templates locally
`helm lint`	Validate chart syntax
`helm test`	Run chart tests

Next Steps

- Kubernetes kubectl Cheat Sheet — Essential kubectl commands
- Docker CLI Cheat Sheet — Build container images
- Kubernetes GitOps with ArgoCD — Declarative deployments with Helm