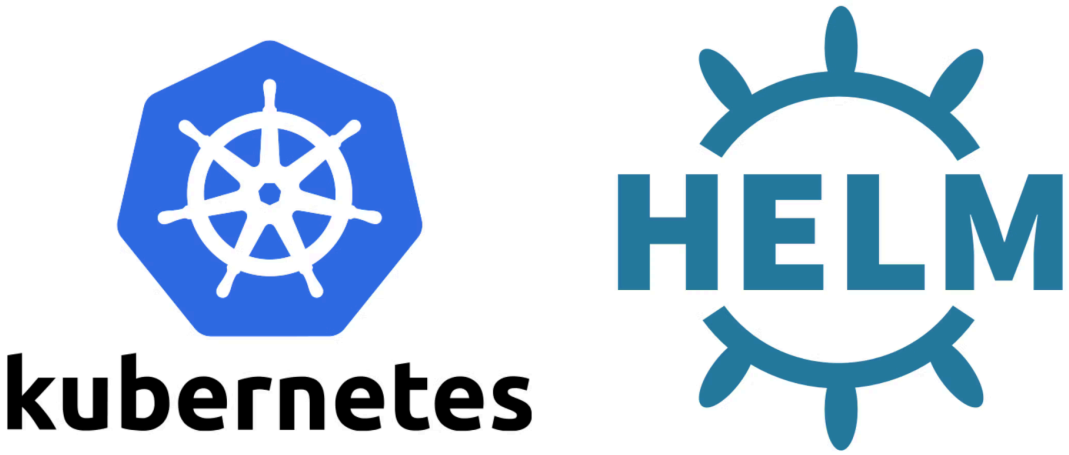
Helm for Kubernetes

Author: Zayan Ahmed | **Estimated Reading time:** 6 min

Introduction

Helm is a powerful package manager for Kubernetes, designed to simplify the deployment, management, and scaling of applications within Kubernetes clusters. Often referred to as the "Kubernetes Package Manager," Helm helps streamline the process of creating, sharing, and maintaining Kubernetes application configurations



Key Concepts

1. Charts

A Helm Chart is a collection of YAML files that describe a set of Kubernetes resources. Think of it as a template for deploying an application. Charts can include deployments, services, ingress rules, ConfigMaps, secrets, and more.

- **Chart Structure**:

```
mychart/
├── Chart.yaml          # Metadata about the chart
├── values.yaml         # Default configuration values
├── templates/          # Templates for Kubernetes resources
```

```
└── charts/         # Dependencies (sub-charts)
```

2. Releases

A Helm Release is a deployed instance of a Helm Chart. Each release is unique within a Kubernetes namespace and has a version history, allowing for upgrades and rollbacks.

3. Repositories

Helm Repositories are where charts are stored and shared. Popular repositories include Artifact Hub and Bitnami.

Benefits of Helm

- **Simplified Deployments**: Automates the generation and application of Kubernetes resource configurations.
- **Versioning**: Tracks changes and allows rollback to previous release versions.
- **Reusability**: Enables reuse of Charts across environments and projects.
- **Community Support**: A large library of pre-built Charts for common applications.

Installing Helm

1. Prerequisites

- Kubernetes cluster up and running.
- `kubectl` installed and configured.

2. Install Helm

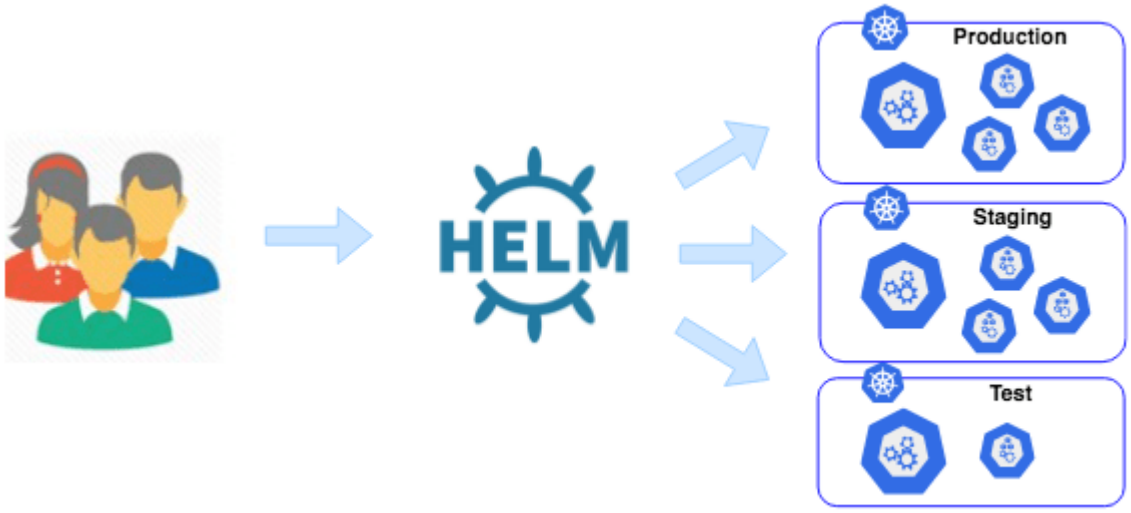
- **Linux/macOS**:

```
curl https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3 | bash
```

- **Windows**: Use Chocolatey or download the binary from the official Helm releases page.

3. Verify Installation

```
helm version
```



Common Helm Commands

Adding a Repository

```
helm repo add <repo_name> <repo_url>

helm repo update
```

Example:

```
helm repo add bitnami https://charts.bitnami.com/bitnami
```

Searching for Charts

```
helm search repo <chart_name>
```

Installing a Chart

```
helm install <release_name> <chart_name>
```

Example:

```
helm install my-nginx bitnami/nginx
```

Listing Releases

```
helm list
```

Upgrading a Release

```
helm upgrade <release_name> <chart_name>
```

Rolling Back a Release

```
helm rollback <release_name> <revision_number>
```

Uninstalling a Release

```
helm uninstall <release_name>
```

Customizing Helm Charts

Using `values.yaml`

Helm allows customization of Charts using a `values.yaml` file. Modify the file to specify custom configurations before installation or upgrade.

Overriding Values Inline

You can override specific values without editing `values.yaml` using the `--set` flag:

```
helm install my-release bitnami/nginx --set service.type=NodePort
```

Helm Best Practices

1. **Version Control for Charts**: Store your Charts in a version-controlled repository.
2. **Parameterize Configurations**: Use `values.yaml` extensively to make Charts environment-agnostic.
3. **Use Chart Dependencies**: Simplify deployments by using sub-charts for complex applications.
4. **Enable RBAC**: Secure Helm operations with role-based access control.
5. **Regularly Update Charts**: Keep Charts updated to incorporate the latest features and fixes.

Advanced Features

1. Chart Hooks

Helm supports hooks to execute tasks at specific points in the lifecycle of a release (e.g., pre-install, post-install). Define hooks in the `templates` directory.

2. Chart Dependencies

Use the `requirements.yaml` file to specify dependent Charts. Helm will fetch and manage these dependencies automatically.

3. Helm Plugins

Extend Helm's functionality with plugins. Install plugins using:

helm plugin install <plugin_url>

```
helm plugin install <plugin_url>
```

4. Helmfile

For managing multiple Helm releases, consider using Helmfile, which allows you to define releases declaratively in a single YAML file.

Troubleshooting

- **Common Issues**:
  - "Failed to download chart": Check the repository URL or update the repo.
  - "Release already exists": Use `helm uninstall` to clean up stale releases.

Debugging Commands:
helm get all <release_name>

- helm template <chart_name> --debug

```
helm get all <release_name>

helm template <chart_name> --debug
```

Conclusion

Helm is a vital tool for managing Kubernetes applications effectively. By leveraging its templating, versioning, and community-driven Charts, you can streamline application deployment and maintenance while adhering to Kubernetes best practices.

Follow me on **LinkedIn** for more 😊