# Mastering Helm

## 1. Introduction to Helm

* 1.1. What is Helm?
* 1.2. Benefits of Using Helm
* 1.3. Helm Architecture Overview
* 1.4. Helm 3 vs. Previous Versions
* 1.5. Key Concepts and Terminology

## 2. Getting Started

* 2.1. System Requirements
* 2.2. Installing Helm
  + 2.2.1. Installation on macOS
  + 2.2.2. Installation on Windows
  + 2.2.3. Installation on Linux
* 2.3. Setting Up a Kubernetes Cluster
  + 2.3.1. Using Minikube
  + 2.3.2. Cloud Provider Clusters (EKS, GKE, AKS)
* 2.4. Verifying the Installation

## 3. Helm Basics

* 3.1. Understanding Helm Charts
  + 3.1.1. Chart Structure and Components
  + 3.1.2. Chart Repositories
* 3.2. Using Official and Stable Charts
* 3.3. Installing Your First Chart
  + 3.3.1. The helm install Command
  + 3.3.2. Customizing Installations with Values
* 3.4. Managing Releases
  + 3.4.1. Listing Releases
  + 3.4.2. Upgrading Releases
  + 3.4.3. Rolling Back Releases
  + 3.4.4. Uninstalling Releases

## 4. Deep Dive into Charts

* 4.1. Anatomy of a Chart
* 4.2. The Chart.yaml File
* 4.3. The values.yaml File
* 4.4. Templates and the Templates Directory
* 4.5. Chart Dependencies
  + 4.5.1. Managing Dependencies with requirements.yaml
  + 4.5.2. Updating Dependencies

## 5. Creating Custom Charts

* 5.1. Using helm create
* 5.2. Customizing Templates
  + 5.2.1. Introduction to Go Templates
  + 5.2.2. Template Functions and Pipelines
  + 5.2.3. Control Structures (if, else, range)
* 5.3. Working with Variables and Values
* 5.4. Packaging and Sharing Charts
  + 5.4.1. The helm package Command
  + 5.4.2. Hosting Charts on a Repository
  + 5.4.3. Using ChartMuseum and Artifact Hub

## 6. Advanced Template Techniques

* 6.1. Using Named Templates and Partials
* 6.2. Template Inheritance with \_helpers.tpl
* 6.3. Advanced Functions and Sprig Library
* 6.4. Implementing Conditional Logic
* 6.5. Handling Complex Data Structures

## 7. Chart Development Best Practices

* 7.1. Chart File Organization
* 7.2. Writing Maintainable Templates
* 7.3. Managing Default Values
* 7.4. Versioning Charts with Semantic Versioning
* 7.5. Documentation and Annotations

## 8. Helm Hooks and Lifecycle Events

* 8.1. Introduction to Helm Hooks
* 8.2. Defining Hooks in Templates
* 8.3. Hook Weights and Deletion Policies
* 8.4. Common Use Cases for Hooks
* 8.5. Troubleshooting Hooks

## 9. Security and Access Control

* 9.1. Understanding Kubernetes RBAC
* 9.2. Helm Security Best Practices
* 9.3. Managing Sensitive Data with Secrets
* 9.4. Integrating with External Secret Management Tools
* 9.5. Verifying Chart Signatures

## 10. Helm Plugins and Extensions

* 10.1. Discovering and Installing Plugins
* 10.2. Popular Helm Plugins
  + 10.2.1. Helm Diff
  + 10.2.2. Helm Secrets
  + 10.2.3. Helm Unittest
* 10.3. Developing Custom Plugins

## 11. Integrating Helm with CI/CD Pipelines

* 11.1. The Role of Helm in DevOps
* 11.2. Automating Deployments with Jenkins
* 11.3. Using Helm with GitLab CI/CD
* 11.4. Implementing GitOps with Helm and Flux
* 11.5. Continuous Deployment Strategies

## 12. Helmfile and Managing Multiple Releases

* 12.1. Introduction to Helmfile
* 12.2. Installing and Setting Up Helmfile
* 12.3. Defining Multiple Releases
* 12.4. Managing Environments and Overlays
* 12.5. Helmfile Commands and Workflow

## 13. Advanced Helm Practices

* 13.1. Chart Testing and Validation
  + 13.1.1. Using helm lint
  + 13.1.2. Writing Unit Tests with Helm Unittest
* 13.2. Performance Tuning and Optimization
* 13.3. Debugging Techniques
  + 13.3.1. The helm get Commands
  + 13.3.2. Template Debugging with --debug
* 13.4. Managing Large Scale Deployments

## 14. Helm and Kubernetes Ecosystem

* 14.1. Integrating Helm with Other Tools
  + 14.1.1. Helm and Kustomize
  + 14.1.2. Helm and Istio
* 14.2. Helm Operator Patterns
* 14.3. Service Mesh Deployments with Helm
* 14.4. Stateful Applications and Helm

## 15. Helm API and SDK

* 15.1. Understanding the Helm Go SDK
* 15.2. Using the Helm SDK in Applications
* 15.3. Interacting with the Helm API
* 15.4. Building Custom Deployment Tools

## 16. Upgrading and Migrating Helm

* 16.1. Migration from Helm 2 to Helm 3
* 16.2. Handling Deprecated Features
* 16.3. Ensuring Backward Compatibility
* 16.4. Helm Upgrade Strategies

## 17. Community, Contribution, and Further Learning

* 17.1. Helm Governance and Roadmap
* 17.2. Contributing to Helm Projects
* 17.3. Community Resources and Support
* 17.4. Keeping Up with Latest Developments
* 17.5. Recommended Books and Courses

## 18. Appendices

* 18.1. Helm Command Reference
* 18.2. Glossary of Terms
* 18.3. Troubleshooting Guide
* 18.4. YAML Primer for Helm Users
* 18.5. Template Function Reference

————————

By following this comprehensive guide, you'll progress from a Helm beginner to an expert, mastering all aspects of Helm for Kubernetes package management.

#software/containers/orchestration/helm