Antony Kervazo-Canut

Helm for Teenagers

SUITABLE FOR ADULTS

Introduction

Helm is a package manager for Kubernetes. It allows developers and system administrators to easily deploy, configure, and manage applications on Kubernetes clusters. Using Helm, you can package your applications along with all their dependencies into a standardized format called a "chart".

- Simplification of Deployment: Helm reduces the complexity of Kubernetes deployments by managing application packages in the form of charts.
- Configuration Management: Helm charts allow for easy and repeatable configuration of Kubernetes applications, enabling the deployment of different instances of the application with varied configurations.
- Reusability: Helm charts are designed to be shared and reused, thus facilitating collaboration and standardizing deployments within an organization or the community.
- Dependency Management: Helm can automatically download and install the dependencies needed for a chart, simplifying the management of complex applications.

In summary, Helm is an essential tool for any Kubernetes user, offering a standardized and efficient method for deploying applications. It encapsulates best deployment practices in Kubernetes, allowing teams to focus on application development rather than the management of their deployment.

Installation of Helm

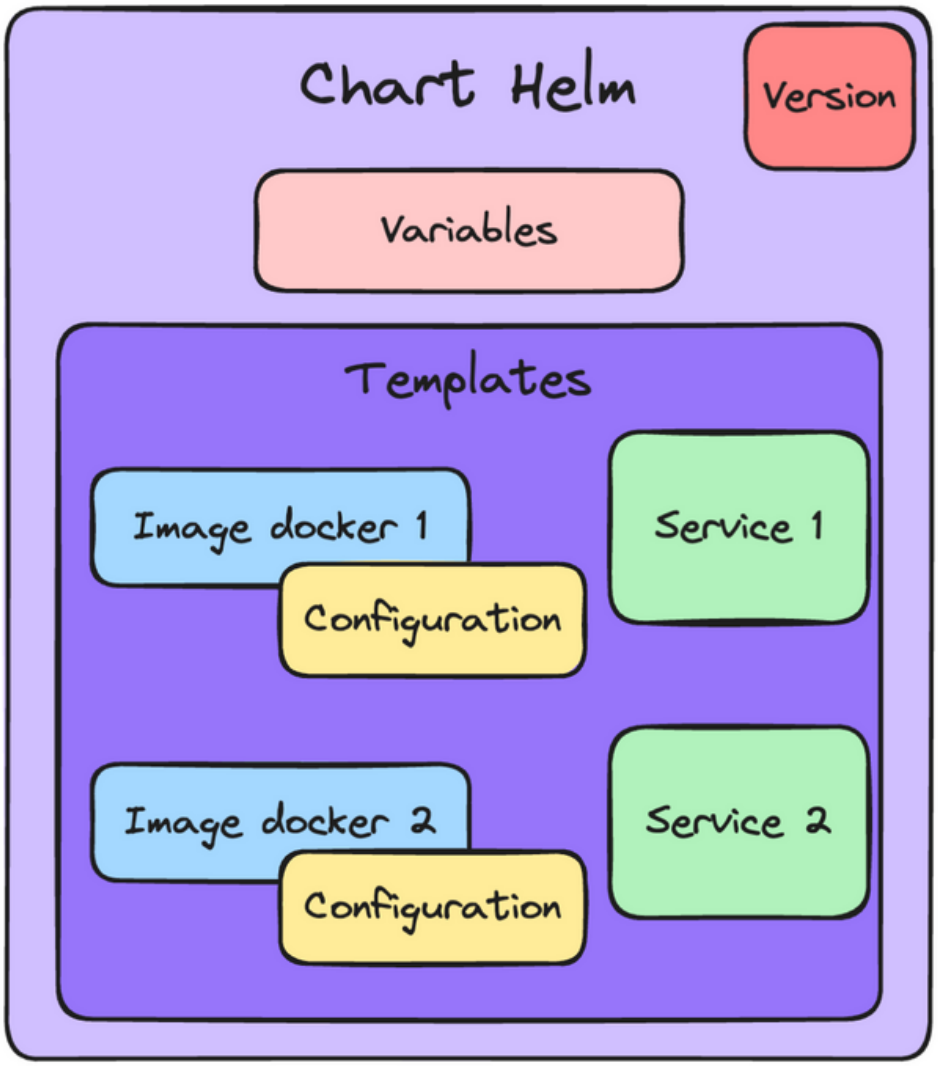
```
### LINUX ###
# Download the Helm installation script
# wget [URL] - command to download files from the Internet
wget
https://raw.githubusercontent.com/helm/helm/main/scripts/get-
helm-3

# Make the script executable
# chmod +x [file] - changes permissions to make a file
executable
chmod +x get-helm-3

# Execute the installation script
# ./[script] - executes a script in the shell
./get-helm-3

### WINDOWS ###
# Installation via Chocolatey, a package manager for Windows
# choco install [package] - command to install packages with
Chocolatey
choco install kubernetes-helm

### MACOS ###
# Installation via Homebrew, a package manager for macOS
# brew install [package] - command to install packages with
Homebrew
brew install helm
```

Helm configuration

```
# Add a public chart repository
# helm repo add [name] [URL] - adds a new chart repository
helm repo add stable https://charts.helm.sh/stable

# Add a private chart repository with authentication
# helm repo add [name] [URL] --username [username] --password
[password]
# Replace [username] and [password] with your credentials
helm repo add private-repo https://myprivaterepo.com --
username myuser --password mypassword

# Update chart repository information
# helm repo update - updates the list of available charts in
added repositories
helm repo update

# List chart repositories
# helm repo list - displays all configured chart repositories
helm repo list
```

Helm Charts

A Helm chart is a package that simplifies the deployment and management of applications on Kubernetes. Think of it as a recipe containing instructions and files needed to install an application on Kubernetes. Each chart includes configuration files and templates that can be customized to adapt the application to different environments or specific needs.

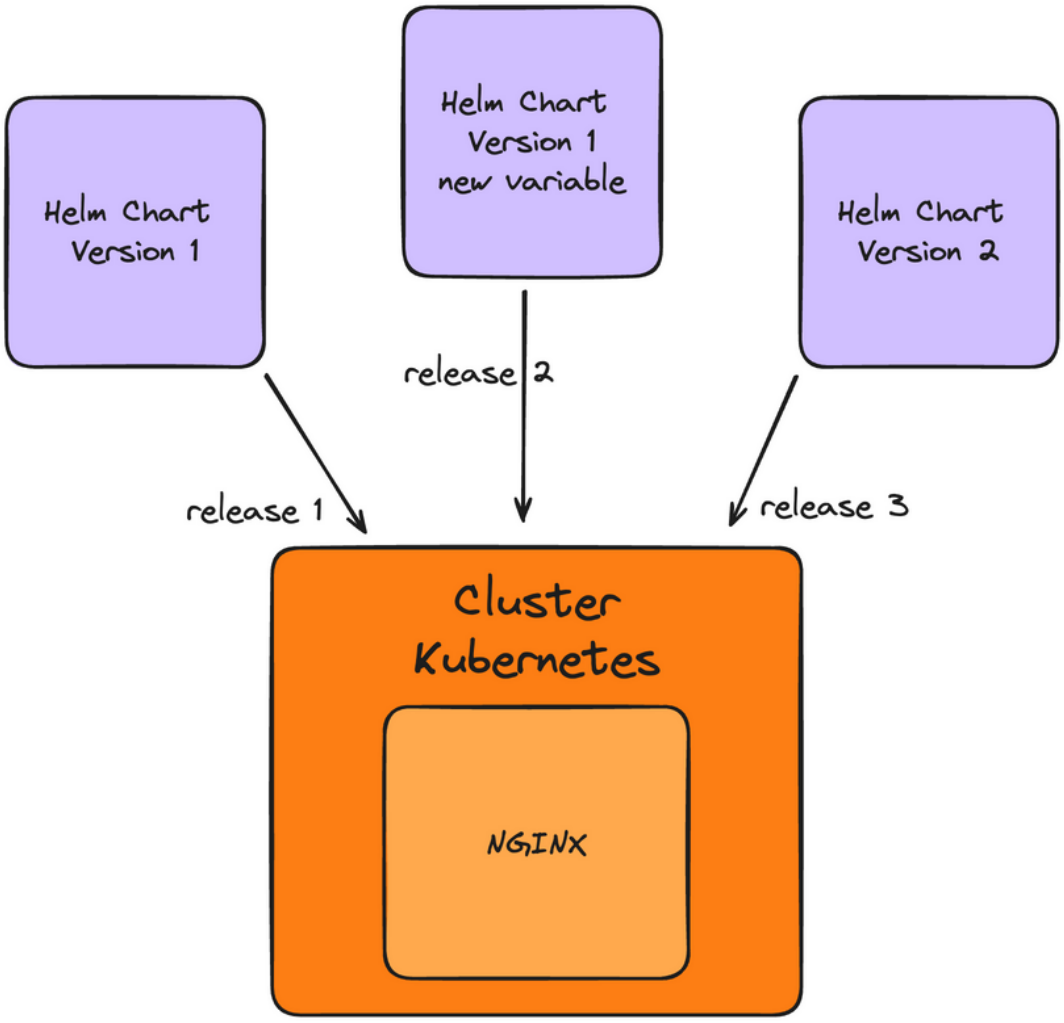


Installing a Chart

```
# Search for charts in repositories
# helm search repo [keyword] - searches for charts by keyword
helm search repo nginx

# Install a chart
# helm install [release_name] [chart_name] - installs a chart
into the cluster
# Replace [release_name] with the desired name for your
deployment
# Replace [chart_name] with the name of the chart to install
helm install my-nginx stable/nginx

# Install a chart with custom configuration
# helm install [release_name] [chart_name] --set [parameter]=
[value]
# Replace [release_name] and [chart_name] with the
appropriate names
# Replace [parameter] with the name of the variable to
configure and [value] with the desired value
helm install my-nginx stable/nginx --set
service.type=NodePort

# List chart repositories
# helm repo list - displays all configured chart repositories
helm repo list
```

Releases

Once a chart is deployed, Helm creates a "release". A release is an instance of a chart running in a Kubernetes cluster. Helm allows for efficient management of these releases.



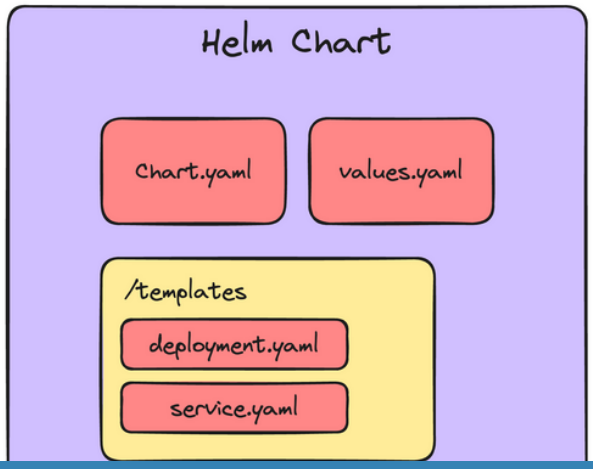
Releases Managment

```
# Update a release
# helm upgrade [release_name] [chart_name] - updates the
specified release
# You can also pass additional parameters with --set or by
using a values file
helm upgrade my-nginx stable/nginx --set
service.type=LoadBalancer

# Roll back to a previous version of a release
# helm rollback [release_name] [revision] - reverts to a
previous version of the release
# Replace [revision] with the revision number you wish to
roll back to
helm rollback my-nginx 1

# Check the status of a release
# helm status [release_name] - displays the current status of
the release
helm status my-nginx

# Delete a release
# helm uninstall [release_name] - removes the release from
the cluster
helm uninstall my-nginx
```

Creating a Chart

A Helm chart is composed of several files and directories. The typical structure of a chart is as follows:

- Chart.yaml: The metadata file of the chart, containing information such as the name, version, and a description.
- values.yaml: The default configuration file, where you define the values used by the chart's templates.
- templates/: A directory containing the YAML templates for Kubernetes resources, which will be customized at installation using Go Template and the variables from the values file.
- charts/: An optional directory containing dependencies in the form of charts.




```
# Create a new Helm chart
# helm create [chart_name] - creates a basic structure for a
new chart
helm create mychart
```

Creating a Chart

```yaml
apiVersion: v2
name: mychart
description: A simple example of a Helm chart
version: 0.1.0
```

Chart.yaml

```yaml
replicaCount: 1
image:
  repository: nginx
  tag: "stable"
  pullPolicy: IfNotPresent
service:
  type: ClusterIP
  port: 80
```

values.yaml

Creating a Chart

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ include "mychart.fullname" . }}
  labels:
    {{- include "mychart.labels" . | nindent 4 }}
spec:
  replicas: {{ .Values.replicaCount }}
  selector:
    matchLabels:
      {{- include "mychart.selectorLabels" . | nindent 6 }}
  template:
    metadata:
      labels:
        {{- include "mychart.selectorLabels" . | nindent 8 }}
    spec:
      containers:
        - name: {{ .Chart.Name }}
          image: "{{ .Values.image.repository }}:{{
.Values.image.tag }}"
          imagePullPolicy: {{ .Values.image.pullPolicy }}
          ports:
            - containerPort: {{ .Values.service.port }}
```

deployment.yaml

Creating a Chart

```yaml
apiVersion: v1
kind: Service
metadata:
  name: {{ include "mychart.fullname" . }}
  labels:
    {{- include "mychart.labels" . | nindent 4 }}
spec:
  type: {{ .Values.service.type }}
  ports:
    - port: {{ .Values.service.port }}
      targetPort: {{ .Values.service.port }}
  selector:
    {{- include "mychart.selectorLabels" . | nindent 4 }}
```

service.yaml

Dependencies in a Chart

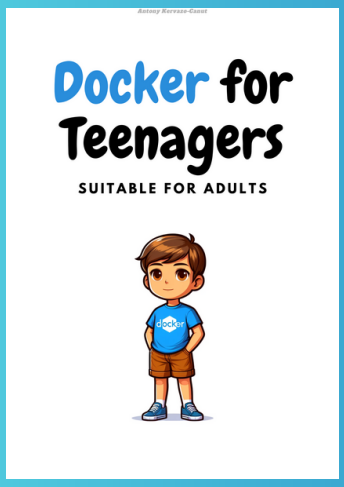
Helm charts can depend on other charts, which is useful for complex applications requiring multiple components.

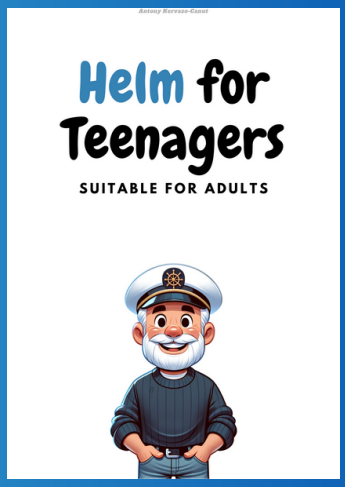
When working with complex charts having multiple dependencies, it is important to check compatibility, ensuring that the versions of the dependent charts are compatible with each other.

```yaml
dependencies:
  - name: nginx
    version: "1.16.0"
    repository: "https://charts.helm.sh/stable"
```

```
# Update the chart's dependencies
# helm dependency update [path_to_chart]
helm dependency update ./mychart
```

In the same collection





ANTONYCANUT

ANTONY KERVAZO-CANUT