

Intro to K8s Helm and Operators

Joshua Haupt

hauptj.com

joshua@hauptj.com

What is Kubernetes

- K8s is an open-source container orchestration platform for automating the deployment, scaling, and management of containerized software.
- Originally created by Google in 2014
- Maintained by the Cloud Native Computing Foundation (CNCF)

What is Containerization?

- Containerization is the packaging of software with operating system libraries and required dependencies to run it in a single executable package.
- Adoption became popular with the release of Docker

What is Helm?

- “Helm helps you manage Kubernetes applications — Helm Charts help you define, install, and upgrade even the most complex Kubernetes application.”
- “Charts are easy to create, version, share, and publish — so start using Helm and stop the copy-and-paste.”
- Source: helm.sh

Helm V2: Client Server model

- Helm V2: Two parts
 - Helm: client that runs on your local system
 - Tiller: server that runs inside your Kubernetes cluster
 - Manages installations of charts
 - Considered security risk as it requires root permissions
- EOL

Helm V3: Client Model

- Helm V3:
 - Tiller removed
 - All templating is now done on the client side

What it provides

- A package manager for Kubernetes with templating
 - Flow Control
 - Version control
 - Dependency management
- Written in Golang
 - Utilizes the [Go template](#) rendering engine

Flow Control

- Enables specifying variables for specific environments
 - Local, development, production
- Control Structures
 - If / else –conditional blocks
 - With – specify a scope
 - Like an “if” statement
 - Range – for / “for each” loop
- [Chart template guide on Flow Control](#)

Pipelining Example – if / else

```
{{- if eq .Values.stage "prod" -}}  
  debug: false  
  host: example.com  
  port: 443  
{{- else if eq .Values.stage "dev" -}}  
  debug: true  
  host: dev.example.com  
  port: 443  
{{- else -}}  
  debug: true  
  host: local.example.com  
  port: 8443  
{{- end -}}
```

Pipelining Example – if / else

```
{{- if eq .Values.stage "prod" -}}
  debug: false
  host: example.com
  port: 443
{{- else if eq .Values.stage "dev" -}}
  debug: true
  host: dev.example.com
  port: 443
{{- else -}}
  debug: true
  host: local.example.com
  port: 8443
{{- end -}}
```

- **Note:** a dash and a space removes whitespace
 - {{- :removes whitespace from the left
 - -}} : removes whitespace from the right

Pipelining Example - with

```
{{- with .Values.production -}}  
  debug: false  
  host: example.com  
  port: 443  
  log: {{ .path | quote }}  
{{- end -}}
```

- Note: (.) specifies the current scope
- With changes the scope
- Scope is reset with {{end}}
- Values.yaml:
 production:
 path: /dev/null

Flow Control - range

users: |-

```
{{- range .Values.users }}
```

```
- {{ . | title | quote }}
```

```
{{- end }}
```

• Values.yaml

users:

- Alice

- Bob

- Charlie

Version Control with ChartMuseum

- Open source repository for Helm charts
- Supports many 3rd party object storage platforms for storage
- Supports local storage on the filesystem
- github.com/helm/chartmuseum

GitLab Helm Chart Repo Support

- Basic support introduced in GitLab 14.1 in all tiers including gitlab.com SaaS
- [Helm charts in the Package Registry | GitLab](#)

Dependency Management

- Umbrella / parent charts that reference dependency charts
 - Example: [Bitnami Kube Prometheus Chart](#)

Creating a simple chart

- \$ helm create mychart
- \$ tree mychart

```
mychart
|-- Chart.yaml
|-- charts
|-- templates
|   |-- NOTES.txt
|   |-- _helpers.tpl
|   |-- deployment.yaml
|   |-- ingress.yaml
|   `-- service.yaml
`-- values.yaml
```

- The **templates directory** is where Helm reads YAML definitions for Kubernetes objects such as Services and Deployments
- **Chart.yaml** contains the metadata that describes and manages the version of the chart
- **Values.yaml** contains the default values that are set at the time of deployment

ChartMuseum via Docker

```
docker run --rm -it \  
-p 8080:8080 \  
-e DEBUG=1 \  
-e STORAGE=local \  
-e STORAGE_LOCAL_ROOTDIR=/charts \  
-v $(pwd)/charts:/charts \  
chartmuseum/chartmuseum:latest
```

Kubernetes Operators

- Extend functionality by defining custom resources and controllers
- An Operator is a K8s controller that is specifically designed to manage and operate a particular custom resource
- Enable the automation of tasks without complex scripting

Kubernetes Controllers

- Component of the K8s system responsible for managing and maintaining the desired states of the resources in a cluster

Live Walkthrough

- [Terraform](#)
- [Helm](#)
- [DigitalOcean Managed K8s](#)
- [Kube Prometheus Stack Helm Chart](#)
- [Nginx Ingress Controller](#)