**What is Helm?**

1. Helm is a **Package Manager** for Kubernetes.
   1. Helm is a tool for managing Kubernetes Packages called **Charts.**
   2. A chart is a **collection of files** that describe a related set of Kubernetes resources.

Eg: Install mysql Package

helm install mysql stable/mysql

helm upgrade mysql stable/mysql

Diagram

Description automatically generated

Note: Helm Compares Old Chart, New Chart and Live State and creates a new Patch that updates the cluster.

1. Helm is a **Templating Engine**.
   1. We can define a common blueprint (template) for multiple YAML files.
   2. In a template file, dynamics values replaced their placeholders. The values come from a file called values.yaml.

A screenshot of a computer

Description automatically generated with medium confidence

1. Helm is used for **Release Management**
   1. A **release** is a running instance of a chart, combined with a specific config.
   2. The **config** contains configuration information that can be merged into a packaged chart to create a releasable object.
   3. Helm also keeps a release history of all deployed charts, so you can go back to a previous release if something went wrong.

### Definitions:

* **Chart** is the definition of our Application
* **Release** is the instance of the chart running in the Kubernetes cluster.
* **Release** **revision**: Change the existing YAML files and update the chart.
* **Chart** **Version** refers to the change in chart file structure. You may add **new YAML** files to chart.

**Helm can do the following:**

1. Create new charts from scratch.
2. Package charts into chart archive (tgz) files.
3. Interact with chart repositories where charts are stored.
4. Install and uninstall charts into an existing Kubernetes cluster.
5. Manage the release cycle of charts that have been installed with Helm.

**Installing Helm**

<https://helm.sh/docs/intro/install/>

Diagram

Description automatically generated

**Installing on Windows**

choco install kubernetes-helm

**Installing on Linux**

**curl** <https://get.helm.sh/helm-v3.8.2-darwin-amd64.tar.gz>

tar -zxvf helm-v3.0.0-linux-amd64.tar.gz

sudo mv linux-amd64/helm /usr/local/bin/helm

**helm version --short**

**To add existing Packages - Installing Official Helm Charts Repository**

**helm repo add stable** <https://charts.helm.sh/stable>

**Working with Helm Charts**

Charts allow you to version your manifest files too, just like we do with Node.js or any other package. This lets you install specific chart versions, which means keeping specific configurations for your infrastructure in the form of code.

**To create a Helm Chart:**

helm create mywebchart

**This creates the following folders and files**

**mywebchart/**

**chart.yaml** # A YAML file containing information about the chart

**values.yaml** # The default configuration values for the templates in the chart.

**charts/** # A directory containing any charts upon which this chart depends.

**templates/**  # Templates Directory, with values, will generate valid Kubernetes manifest files.

**crds**/ # Custom Resource Definitions

README.md # OPTIONAL: A human-readable README file

LICENSE # OPTIONAL: A plain text file containing the license for the chart

templates/NOTES.txt # OPTIONAL: A plain text file containing short usage notes

values.schema.json # OPTIONAL: A JSON Schema for imposing a structure on the values.yaml file

Note: Helm reserves use of the **charts/, crds/,** and **templates/** directories, and of the listed file names. Other files will be left as they are.

### Text Description automatically generated with medium confidence

### Overview Chart.yaml

### Graphical user interface, text, application Description automatically generated

### apiVersion – Helm API Version (V2 is for Helm3)

### appVersion – Application Version

### version – Chart Version

**Helm Commands**

* helm create [chart]
* helm install [release] [chart] Install a Release
* helm upgrade [release] [chart] Upgrade a Release version
* helm rollback [release] [version] Rollback to a Release version
* helm history [release] Print Release history
* helm status [release] Display Release status
* helm get all [release] Show details of a release
* helm uninstall [release] Uninstall a Release
* helm list List Release

**Install chart into Kubernetes Cluster:**

* helm create mywebappchart
* helm install **--dry-run** demo-mywebapp mywebappchart
* helm install demo-mywebapp mywebappchart
* helm list --short
* helm get manifest demo-mywebapp
* helm get all demo-mywebapp

**Version updates:**

1. helm status demo-mywebapp

Note the REVISION=1

1. **Change the App Version in Chart.yaml**
2. **Change the Image to new version in deployment.yaml**
3. helm upgrade demo-mywebapp mywebappchart
4. helm status demo-mywebapp

Note the REVISION=2 and kubectl

**Rollback:**

1. helm history demo-mywebapp
2. helm rollback demo-mywebapp 1
3. helm history demo-mywebapp

**Uninstall:**

helm uninstall demo-mywebapp

**Helm Templates**

**Helm Template Engine:**

It works on client side.

Diagram

Description automatically generated

**Replacing place holders with their values from different sources:**

1. **Using values from Chart.yaml.**

Diagram

Description automatically generated

1. Using values as per Release

A picture containing table

Description automatically generated

1. **Using Values from From Kubernetes Cluster:**

Diagram

Description automatically generated

1. **Values and Sub-charts**

Parent can **override** the values in subchart

Timeline

Description automatically generated

Note: In Parent values.yaml, backend is the subchart name.

1. Using **global** parent values.yaml properties can be used in child charts.

In Parent, values.yaml

. . .

**global**:

K1: V1

In YAML files of either the parent chart or child chart the value of K1 can be fetched using **{{ .Values.global.K1 }}**

**Example**

**Deployment.yaml**

apiVersion: apps/v1

kind: Deployment

metadata:

  name: {{.Release.Name}}-{{.Chart.Name}}

spec:

  replicas: {{.Values.replicaCount}}

  selector:

    matchLabels:

      app: {{.Release.Name}}-{{.Chart.Name}}-app

  template:

    metadata:

      name: {{.Release.Name}}-{{.Chart.Name}}-pod

      labels:

        app: {{.Release.Name}}-{{.Chart.Name}}-app

    spec:

      containers:

      - name: {{.Release.Name}}-{{.Chart.Name}}-con

        image: {{.Values.image.repository}}:{{.Values.image.tag}}

        ports:

        - containerPort: 80

**Service.yaml**

apiVersion: v1

kind: Service

metadata:

  name: {{ .Release.Name }}-{{ .Chart.Name }}-service

spec:

  type: {{.Values.service.type}}

  selector:

    app: {{ .Release.Name }}-{{ .Chart.Name }}-app

  ports:

    - protocol: TCP

      port: {{ .Values.service.port }}

      targetPort: 80

**Values.yaml**

replicaCount: 2

image:

  repository: nginx

  tag: "1.17.0"

service:

  type: LoadBalancer

  port: 8090

**Test the template using**

helm template mywebapp

helm install demo1 mywebapp --dry-run

**Run the chart**

helm install demo1 mywebapp

**Upgrade the chart**

change the value of image tag in values.yaml

helm upgrade demo-mywebapp mywebapp

**Override Values in values.yaml:**

**You can also override the values in Values.yaml using the below command**

helm install demo-mywebapp .\mywebappchart\ **--values** my-values.yaml

Graphical user interface, text, application

Description automatically generated

Graphical user interface

Description automatically generated with medium confidence

helm upgrade demo1 mywebapp --set service.port=8091

**Functions and Pipelines**

Graphical user interface, application, email

Description automatically generated

|  |  |
| --- | --- |
| **Function** | **Pipe** |
| default default\_value value | value | default default\_value |
| quote value | value | quote |
| upper value | value | upper |
| trunc value 63 | value | trunc 63 |
| trimSuffix "-" value | value | trimSuffix "-" |
| b64enc value | value | b64enc |
| randAlphaNum 10 | value | randAlphaNum 10 |
| toYaml value | value | toYaml |
| printf format value | list value … | join " -" |

Values.yaml

service:

  type: NodePort

  name: myservice-with-a-very-long name-that-will-be truncated-by-trunc and-trimsuffix-and which-is-too-long for-Kubernetes

  labels

    port: 80

mongodbRootPassword: T9rYGFAMGE

Examples:

* {{ Values.service.name | default .Chart.Name }} #Chart Name will be used as default value for Values.servce.name doesn’t exist
* {{ .Values.service.name | trunc 63 | trimSuffix "-" }}
* {{ .Values.mongodbRootPassword | b64enc | quote }}

**Modifying Scope using With and Indentation**

**Scoping using With**

apiVersion: v1

kind: Service

metadata:

  name: {{ .Release.Name }}-{{ .Chart.Name }}-service

spec:

  selector:

    app: nginx-{{.Release.Name}}-{{.Chart.Name}}-app

{{ with .Values.service}}

  type: {{ .type }}

  ports:

    - protocol: TCP

      port: {{ .port | default "8080"}}

      targetPort: 80

{{ end }}

**Trimming NewLine and Adding Indentation**

apiVersion: v1

kind: Service

metadata:

  name: nginx-service-{{ .Release.Name }}-{{ .Chart.Name }}

spec:

  selector:

    app: nginx-{{.Release.Name}}-{{.Chart.Name}}-app

  type: {{ .Values.service.type }}

  ports:

    - protocol: TCP

      port:

{{- with .Values.service -}}

      {{ indent 1 .port | default "8080"}}

{{- end }}

      targetPort: 80

**Formatted Output:**

apiVersion: v1

kind: Service

metadata:

  name: nginx-service-{{ printf "%s-%s" .Release.Name .Chart.Name }}

spec:

  selector:

    app: nginx-{{.Release.Name}}-{{.Chart.Name}}-app

{{ with .Values.service}}

  type: {{ .type }}

  ports:

    - protocol: TCP

      port: {{ .port | default "8080"}}

      targetPort: 80

{{ end }}

**Conditions and Logical Operators**

**Conditions:**

Text

Description automatically generated with medium confidence

Table

Description automatically generated

Examples:

{{- if **and** .Values.adminEmail (**or** .Values.serviceAccountJson .existingSecret) }}

{{- if (**and** (**eq** .Values.service.type "NodePort") (**not** (empty .Values.service.nfsNodePort))) }}

{{- if **or** .Values.rbac.pspEnabled (**and** .Values.rbac.namespaced (**or** .Values.sidecar.dashboards.enabled .Values.sidecar.datasources.enabled)) }}

**Loops:**

Diagram

Description automatically generated with medium confidence

**Variables**

apiVersion: v1

kind: Service

{{- $serviceSelector := printf "%s-%s" .Release.Name .Chart.Name }}

metadata:

  name: {{ $serviceSelector }}-service

spec:

{{- with .Values.service}}

  type: {{ .type }}

  selector:

    app: {{ $serviceSelector }}-app

  ports:

    - protocol: TCP

      port: {{ .port | default "8080"}}

      targetPort: 80

{{ end }}

**Calling Helper Functions and Sub-templates**

**Mywebapp/templates/\_helpers.tpl**

{{- define "mychart.fullname" -}}

{{- if .Values.fullnameOverride -}}

{{- .Values.fullnameOverride | trunc 63 | trimSuffix "-" -}}

{{- else -}}

{{- printf "%s-%s" .Release.Name $name | trunc 63 | trimSuffix "-" -}}

{{- end -}}

{{- end -}}

**Deployment.yaml**

apiVersion: apps/v1

kind: Deployment

metadata:

  name: {{ include "mychart.fullname" . }}-dep

spec:

  replicas: {{.Values.replicaCount}}

. . .

Note: Files prefixed by ‘\_’ are not rendered as Kubernetes object

**Working with Helm Repositories**

Table

Description automatically generated

Registry

Repository

Packages/charts

**Get the list of local Helm Repositories**

1. helm repo list

**Note:** No repository by default in Helm 3

1. Goto **https://hub.helm.sh**
2. Search nginx and select the first result.
3. helm repo add bitnami <https://charts.bitnami.com/bitnami>
4. helm search repo nginx
5. helm inspect **chart** bitnami/nginx #Inspect values [all | readme | chart | values ]
6. helm pull bitnami/nginx #fetch can be used instead of pull
7. helm install demo-nginx bitnami/nginx

**Helm Dependency**

* **helm dependency list:** list the dependencies for the given chart
* **helm dependency update**: update charts/ based on the contents of Chart.yaml
* **helm dependency build:** rebuild the charts/ directory based on the Chart.lock file

1. **Update Chart.yaml**

apiVersion: v2

name: mywebapp

description: A Helm chart for Kubernetes

type: application

version: 0.1.0

appVersion: "1.16.0"

**dependencies:**

  - name: mysql

    version: ~8.9.0

    repository: https://charts.bitnami.com/bitnami

1. helm dependency list mywebapp
2. helm dependency update mywebapp

Note: Chart.lock file is added to chart with hardcoded versions.

**Install Chart along with dependency**

1. helm install demo-mywebapp mywebapp
2. kubectl get pods

Note: the pods related to mysql are created along with our custom pods

**Publishing in a Repository**

1. Create the index.yaml

* helm package mywebapp
* helm repo index .

1. Upload charts’ archives and index file to an http server: Nginx, Apache, Cloud or use **Chartmuseum**

ChartMuseum is an open-source [Helm Chart Repository](https://helm.sh/docs/topics/chart_repository) server written in Go (Golang), with support for cloud storage backends, including [Google Cloud Storage](https://cloud.google.com/storage), [Amazon S3](https://aws.amazon.com/s3), [Microsoft Azure Blob Storage](https://azure.microsoft.com/en-us/services/storage/blobs), [Alibaba Cloud OSS Storage](https://www.alibabacloud.com/product/oss) and [Openstack Object Storage](https://developer.openstack.org/api-ref/object-store" \t "_blank).

1. Run the Chartmuseum repository locally from a Docker Image.

* docker run --rm -u 0 -it -p 8080:8080 -e DEBUG=1 -e STORAGE=local -e STORAGE\_LOCAL\_ROOTDIR=/charts -v charts:/charts ghcr.io/helm/chartmuseum:v0.14.0

1. Upload the package to Repository

* curl --data-binary "@mywebapp-0.1.0.tgz" http://localhost:8080/api/charts