**1. Create a New Helm Chart**

We will create a Helm chart that manages a web application (e.g., Nginx) with multiple Kubernetes objects.

helm create myapp-chart

This will create the following structure:

myapp-chart/

├── charts/

├── templates/

├── values.yaml

├── Chart.yaml

└── .helmignore

**2. Modify values.yaml**

Here’s how the values.yaml will look after modifications:

replicaCount: 2

image:

repository: nginx

tag: stable

service:

type: ClusterIP

port: 80

configmap:

appConfig: |

message: Welcome to our web app!

**3. Modify deployment.yaml in the templates/ Directory**

We'll add environment variables from a **ConfigMap** and expose ports via **Service**.

apiVersion: apps/v1

kind: Deployment

metadata:

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

labels:

app: {{ .Release.Name }}

spec:

replicas: {{ .Values.replicaCount }}

selector:

matchLabels:

app: {{ .Release.Name }}

template:

metadata:

labels:

app: {{ .Release.Name }}

spec:

containers:

- name: myapp-container

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

ports:

- containerPort: 80

env:

- name: APP\_MESSAGE

valueFrom:

configMapKeyRef:

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

key: message

**4. Add a service.yaml in the templates/ Directory**

Define a **Service** to expose the application.

apiVersion: v1

kind: Service

metadata:

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

labels:

app: {{ .Release.Name }}

spec:

type: {{ .Values.service.type }}

ports:

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

targetPort: 80

selector:

app: {{ .Release.Name }}

**5. Add a configmap.yaml in the templates/ Directory**

We'll create a **ConfigMap** that holds a message to be displayed by the application.

apiVersion: v1

kind: ConfigMap

metadata:

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

data:

message: {{ .Values.configmap.appConfig | quote }}

**6. Package the Chart**

Your chart now manages:

* A **Deployment** of 2 replicas running **Nginx**.
* A **ConfigMap** to configure the application.
* A **Service** exposing port 80 internally using ClusterIP.

Let’s play around with it!

REMOVE below files :

rm -rf serviceaccount.yaml ingress.yaml hpa.yaml

1102 rm -rf tests/

1113 rm -rf NOTES.txt

**7. Install the Helm Chart**

Deploy your chart into a Kubernetes cluster:

helm install mywebapp ./myapp-chart

Check the deployment and service status:

kubectl get deployments

kubectl get svc

kubectl get configmaps

**8. Helm Operations to Explore**

Now that the chart is deployed, let’s cover some key Helm commands and concepts.

**8.1 View Deployed Resources**

To view the resources that Helm has deployed:

helm list

To inspect the full manifest (all the resources Helm has deployed):

helm get manifest mywebapp

**8.2 Customize and Upgrade**

Modify the values.yaml file. For instance, change the number of replicas:

replicaCount: 3

Then, upgrade your release:

helm upgrade mywebapp ./myapp-chart

Check the updated deployment status:

kubectl get deployments

**8.3 Rollback**

Let’s say the upgrade didn't go as expected. You can rollback to a previous version:

helm rollback mywebapp 1

Check the history of releases:

helm history mywebapp

**8.4 Uninstall the Chart**

To clean up and uninstall your chart:

helm uninstall mywebapp

You can verify that the resources are removed:

kubectl get all

**8.5 Templating and Debugging**

You can render your chart locally to debug it before installing it:

helm template ./myapp-chart

If you want to dry-run the install (render the manifest without applying it):

helm install --dry-run --debug mywebapp ./myapp-chart

SOME MORE EASY EXAMPLES TO UNDERSTAND HELM :

https://github.com/Azure-Samples/helm-charts/tree/master/chart-source

**8.6 Package and Share the Chart**

You can package the chart to distribute it:

helm package ./myapp-chart

This will create a .tgz file that you can share or store in a Helm chart repository.

To share your Helm chart by publishing it to a Helm chart repository, there are a few steps you need to follow. Helm charts can be hosted in different types of repositories, such as:

1. **GitHub Pages**
2. **Google Cloud Storage (GCS)**
3. **S3 Buckets**
4. **Private Helm Repository Tools** (e.g., ChartMuseum, JFrog Artifactory)

Here’s how you can publish to a **GitHub Pages** Helm repository as an example:

**1. Package Your Helm Chart**

First, you need to package your Helm chart into a .tgz file:

helm package ./myapp-chart

This command will generate a .tgz file, such as:

myapp-chart-0.1.0.tgz

**2. Set Up GitHub Pages as a Helm Repository**

You can use GitHub Pages to host your Helm chart repository. Here's how:

1. **Create a GitHub Repository**:
   * Create a new repository on GitHub (e.g., my-helm-charts).
2. **Add the Chart to Your GitHub Repository**:
   * Clone your repository locally:

git clone https://github.com/[your-username]/my-helm-charts.git

cd my-helm-charts

* + Move your packaged chart (myapp-chart-0.1.0.tgz) into the repository:

mv ../myapp-chart-0.1.0.tgz .

1. **Generate an index.yaml File**: Helm repositories need an index.yaml file to serve as an index of available charts. Use the helm repo index command to generate this file:

helm repo index . --url https://[your-username].github.io/my-helm-charts/

This will create an index.yaml file that lists all the available charts in your repository. Make sure the --url flag points to your GitHub Pages URL.

1. **Commit and Push the Files to GitHub**: Add the chart and index to your GitHub repository and push the changes:

git add .

git commit -m "Add myapp-chart"

git push origin main

1. **Enable GitHub Pages**:
   * Go to your repository's **Settings** on GitHub.
   * Scroll down to the **GitHub Pages** section.
   * Under **Source**, select the main branch and the /root folder.
   * Save the changes. Your repository is now served at https://[your-username].github.io/my-helm-charts/.

**3. Add the Repository to Helm**

To use this repository in Helm, you or anyone else can add it with the helm repo add command:

helm repo add my-charts https://[your-username].github.io/my-helm-charts/

**4. Install the Chart from the Repository**

Now you can install the chart directly from your GitHub repository:

helm search repo my-charts

helm install myapp my-charts/myapp-chart --version 0.1.0

**5. Update the Repository**

If you need to add more charts or update an existing chart, follow these steps:

1. **Package the new chart**:

helm package ./new-chart

1. **Move the .tgz file to the repository**:

mv new-chart-0.1.1.tgz ./my-helm-charts/

1. **Regenerate the index.yaml**:

helm repo index . --url https://[your-username].github.io/my-helm-charts/

1. **Push the updated files to GitHub**:

bash

Copy code

git add .

git commit -m "Update charts"

git push origin main

**Summary:**

* **Package** your chart using helm package.
* **Create** a GitHub repository for hosting the chart.
* **Generate** the index.yaml file using helm repo index.
* **Host** the repository on GitHub Pages.
* **Use** the repository by adding it to Helm with helm repo add.