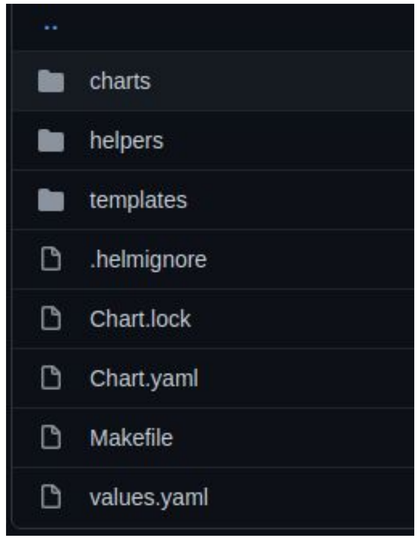
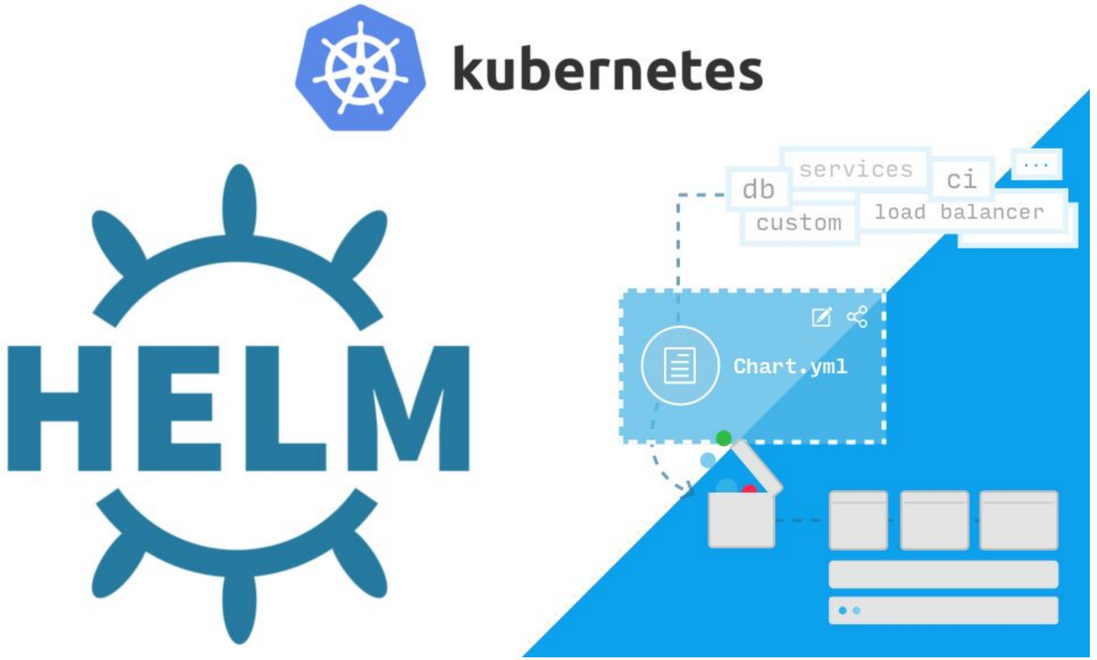
Kubernetes Helm Charts Overview

Contents

What is Helm?

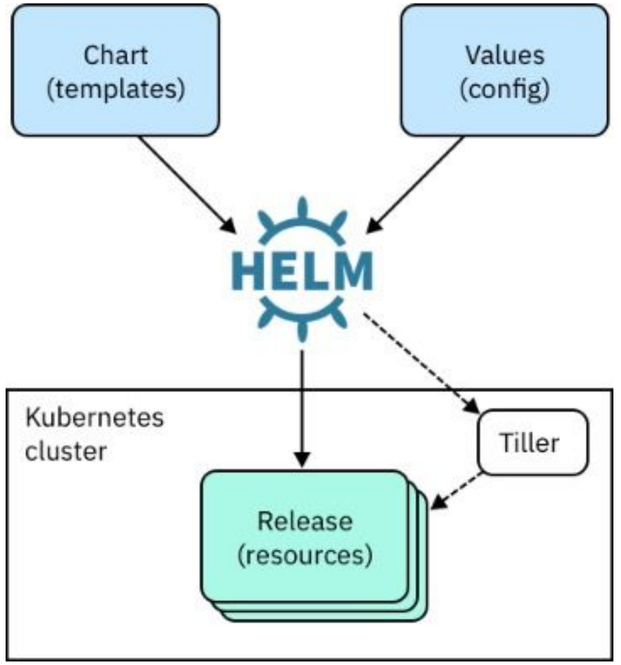
Helm is a k8s packet manager and a chart management tool. With it, you can install applications and deploy applications for k8s as you would through *apt* or *yam*. In simple words, Helm is *templatizing* tool for YAML manifests.





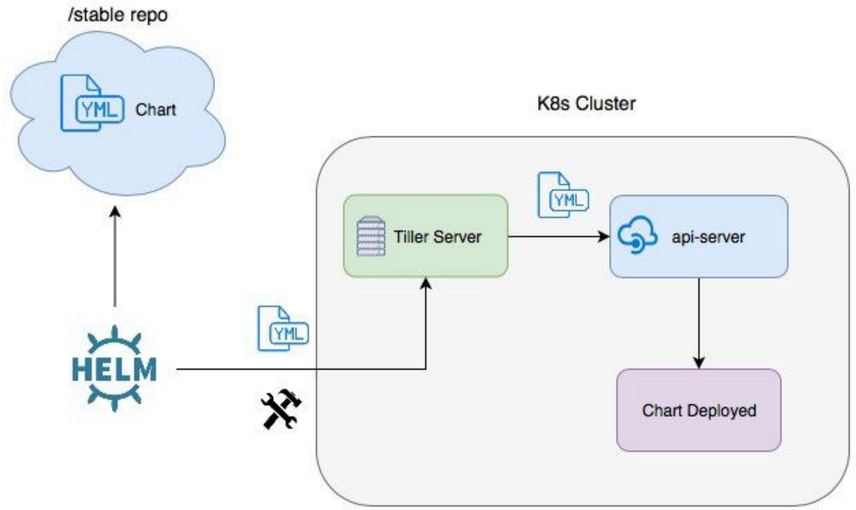
Helm features

- Helm Charts provide the ability to leverage Kubernetes packages through the click of a button or single CLI command
- Helm offers is the ability to customize application configurations during deployment
- Helm automatically maintains a database of all the versions of your releases
- Scalability
- Standards, templates and templates again…



Helm disadvantages

- A good amount of cons are directed towards Helm V2 and the introduction (and later removal in Helm V3) of **Tiller**. In a nutshell, Tiller is the in-cluster portion of Helm that runs the commands and Charts for Helm. Because Tiller had unfettered access to the Kubernetes Cluster, it became a sore point for cluster security
- Helm V2 was storing secrets or sensitive information in Kubernetes ConfigMaps in plain text
- Helm can be overkill for simple deployments

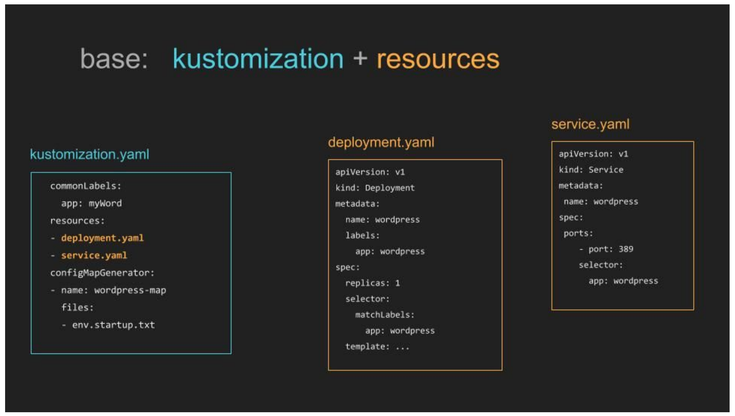


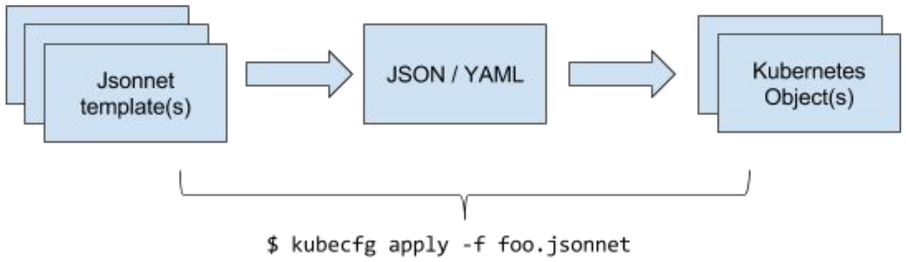
Helm Alternatives

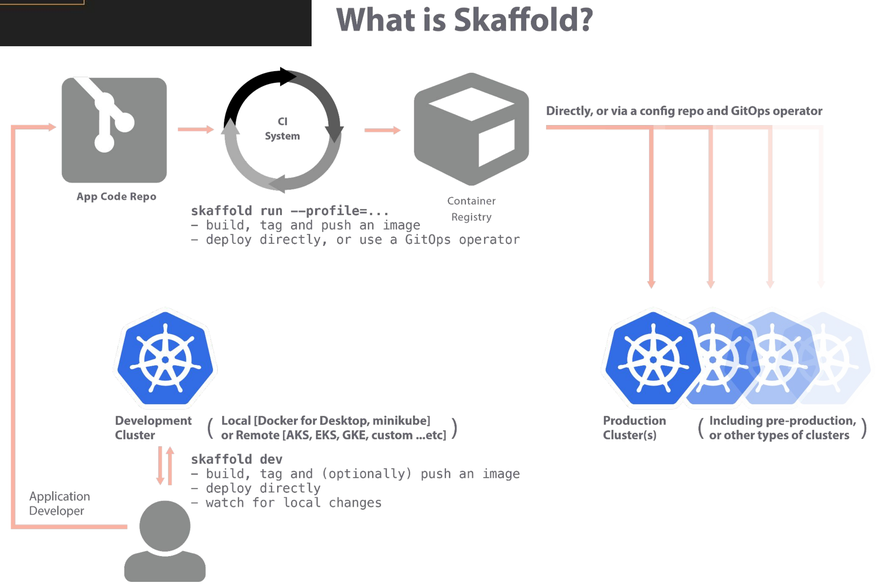
Some tools for object-oriented approach for k8s templating, allowing for complex and relationship-based templates to be created:

- ➢ Kustomize
- ➢ Jsonnet/Ksonnet
- ➢ Skaffold

All of them has pros and cons…







Helm Repositories

Charts Repository is an HTTP server for storing and distributing charts: a place where charts can be stored and shared.

```
$ helm create example-chart
$ helm package example-chart
$ helm repo index helm-local-repo/
$ helm repo add
$ helm repo -h
$ helm repo list
$ helm repo update
$ helm repo remove
$ helm search hub
```

```
$ tree example-chart/
example-chart/
|-- Chart.yaml
|-- charts
|-- templates
|   |-- NOTES.txt
|   |-- _helpers.tpl
|   |-- deployment.yaml
|   |-- hpa.yaml
|   |-- ingress.yaml
|   |-- service.yaml
|   |-- serviceaccount.yaml
|   `-- tests
|       `-- test-connection.yaml
`-- values.yaml

3 directories, 10 files
```

bitnami/**charts**

Bitnami Helm Charts

bitnami by vmware

1k Contributors | 340 Issues | 5k Stars | 5k Forks

ArtifactHUB

Find, install and publish Kubernetes packages

Helm Chart basic commands

You can get a helm chart in the following ways:

1. By pulling it from a repository

```
helm repo add bitnami https://charts.bitnami.com/bitnami
OR
helm repo add stable https://charts.helm.sh/stable
OR
helm repo add stable https://kubernetes-charts.storage.googleapis.com
helm repo update
helm repo list
helm search
helm show chart
helm install nginx bitnami/nginx
helm history
helm rollback
```

```
vasilii@vasilii-sora:~/Github/unkai-helm-chart$ helm install nginx bitnami/nginx
WARNING: Kubernetes configuration file is group-readable. This is insecure. Location: /home/vasilii/.kube/config
WARNING: Kubernetes configuration file is world-readable. This is insecure. Location: /home/vasilii/.kube/config
NAME: nginx
LAST DEPLOYED: Thu Jan 27 09:34:20
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
CHART NAME: nginx
CHART VERSION: 9.7.5
APP VERSION: 1.21.6

** Please be patient while the chart is being deployed **

NGINX can be accessed through the following DNS name from within your cluster:

    nginx.default.svc.cluster.local (port 80)

To access NGINX from outside the cluster, follow the steps below:
```

2. By using the code and install locally

```
git clone https://github.com/prometheus-community/helm-charts.git
cd charts/kube-prometheus-stack/
helm lint
helm install prometheus prometheus-community/kube-prometheus-stack

OR

helm --kube-context ${KUBERNETES_CONTEXT}   \
     install -n ${CHART_NAMESPACE} <release> \
     <name> -f ${CHART_VALUES} --debug --dry-run

---

helm status unkai
helm upgrade
helm uninstall unkai
```

```
vasilii@vasilii-sora:~/Github/unkai-helm-chart$ helm --kube-context minikube install -n default unkai unkai -f unkai/values.yaml
WARNING: Kubernetes configuration file is group-readable. This is insecure. Location: /home/vasilii/.kube/config
WARNING: Kubernetes configuration file is world-readable. This is insecure. Location: /home/vasilii/.kube/config
W0127 09:43:32.909407   44429 warnings.go:70] policy/v1beta1 PodSecurityPolicy is deprecated in v1.21+, unavailable in v1.25+
W0127 09:43:36.134181   44429 warnings.go:70] policy/v1beta1 PodSecurityPolicy is deprecated in v1.21+, unavailable in v1.25+
W0127 09:43:36.139359   44429 warnings.go:70] policy/v1beta1 PodSecurityPolicy is deprecated in v1.21+, unavailable in v1.25+
NAME: unkai
LAST DEPLOYED: Thu Jan 27 09:43:30
NAMESPACE: default
STATUS: deployed
REVISION: 1
```

Helm Values

Everything works, but now all the data in *ConfigMap* is static. How to override values?

There are two methods to override default values:

1. Using the **--set** flag:

```
helm upgrade --install <service> -f <service>-values.yaml
<service>-100.0.112+xxxx.tgz  --set <service>.image.tag=9.0.1xx.xx.xx
```

1. Using the **--values** flag in *values.yaml*:

```
global:
    postgresql:
        postgresqlPassword: "postgres" # new default password
```

```
$ helm repo add bitnami https://charts.bitnami.com/bitnami
$ helm install my-release bitnami/postgresql -f values.yml
```

Helm Hooks

Helm **hooks** are arbitrary Kubernetes resources marked with a special annotation *helm.sh/hook*.
Several hooks are suitable for applying migrations:

- **pre-install** - triggered at the first Helm release of the application after processing all templates, but before creating resources in Kubernetes;
- **pre-upgrade** - triggered when updating the Helm release and executed, like pre-install, after processing templates, but before creating resources in Kubernetes.
- …

```
kind: Job
metadata:
  name: somejob
  annotations:
    "helm.sh/hook": pre-upgrade,pre-install
    "helm.sh/hook-weight": "1"
```

Helm Hooks Case

Case: *I want to generate a password in a Helm template. However the password will be changed when the release is upgraded. Is there a way to check if a password was previously generated and then use the existing value?*

```
apiVersion: v1
kind: Secret
metadata:
  name: {{ .secret.name }}
  annotations:
    "helm.sh/hook": "pre-install"
    "helm.sh/hook-delete-policy": "before-hook-creation"
  labels:
    app: {{ template "helm-random-secret.name" . }}
    chart: {{ template "helm-random-secret.chart" . }}
    release: {{ .Release.Name }}
    heritage: {{ .Release.Service }}
data:
  some-password: {{ { .secretKey | b64enc | quote }}
```