Helm Charts with Kubernetes

Michael Levan

Principal Consultant, Content Creator, Public Speaker, and Advisor on all things Kubernetes

Pearson

Prerequisites

- *A Kubernetes cluster (wherever.. Even Minikube)*
- *Helm installed*

Section 1: Welcome and Setup

- What is a package manager?
- What is Helm?
- Helm setup and installation

What’s A Package Manager?

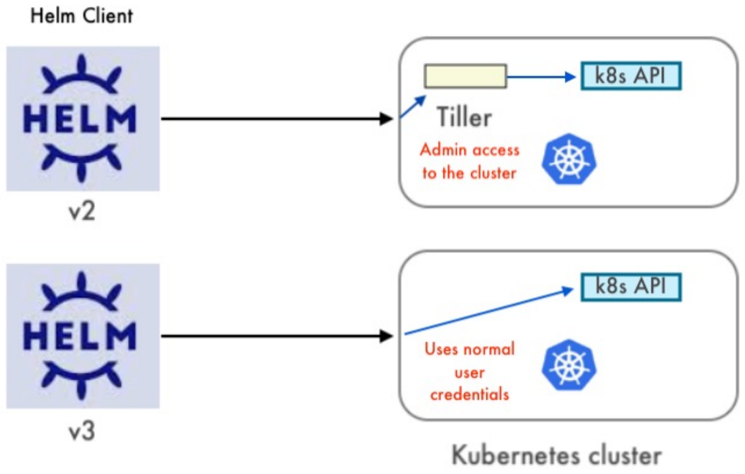
- *Package Manager* keeps track of what software

What Is Helm?

- A collection of charts (more about charts in the next segment)
- Bundled with one or more Kubernetes manifests

A Quick Note On Helm 2 vs Helm 3

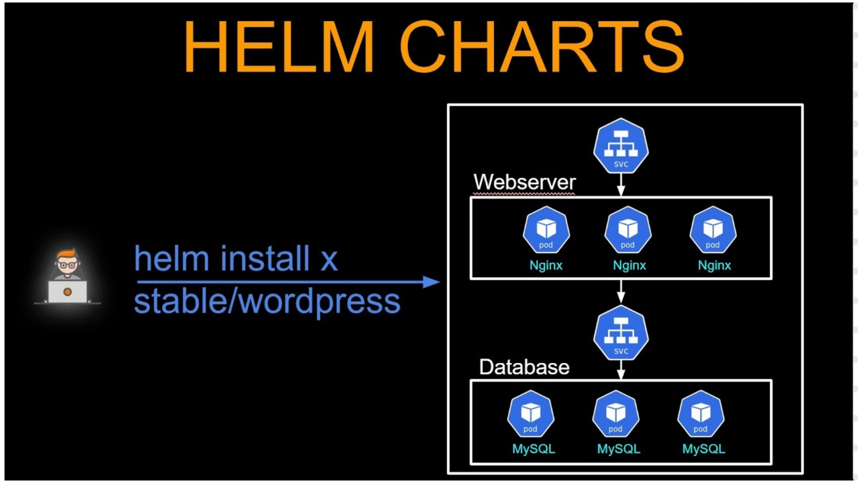
- Removal of Tiller



Hands-On: Install Helm

Section 2: Introduction To Helm

- Helm overview and history
- Why Helm is needed for every environment
- Why should you use Helm?
- What is a Chart?
- Creating a chart



Helm Overview

- Originally created by DeisLabs and donated to CNCF
- Goal is to help manage k8s manifests in an easier fashion
- Helm supports Kubernetes natively
- Release history/versioning (like version control in GitHub)

Pearson

Helm History

- Graduated CNCF project
- Started in 2016 (Helm2)
- Helm3 released in 2019

Why Is Helm Needed

- Manage multiple applications in one place
- Without Helm, there’s a massive amount of configuration sprawl

Why Helm For All Environments?

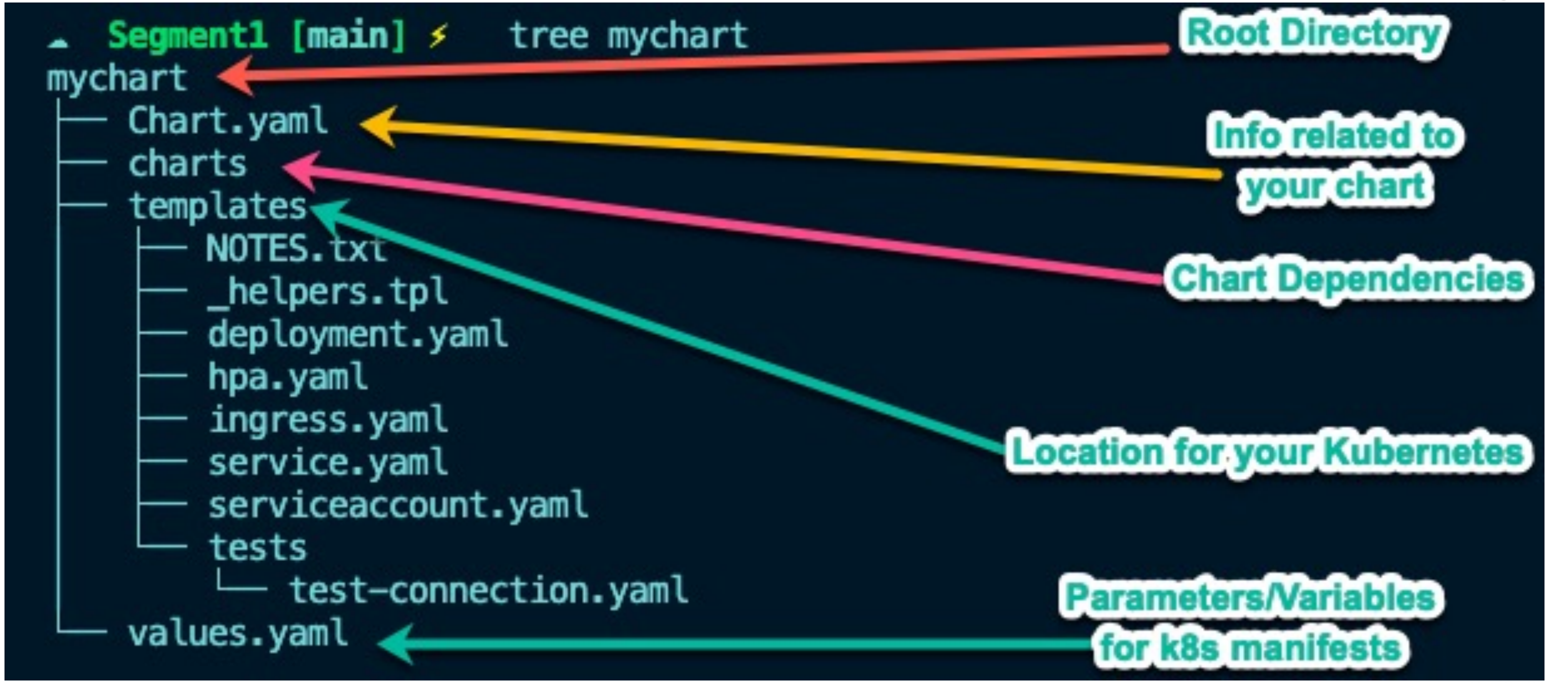
- Consolidate configuration sprawl
  - Dev
  - Staging
  - UAT
  - QA
  - Prod

Pearson

Charts Consists Of…

- chart.yaml
- values.yaml
- Templates Directory
- Other charts (sometimes)

Chart Architecture



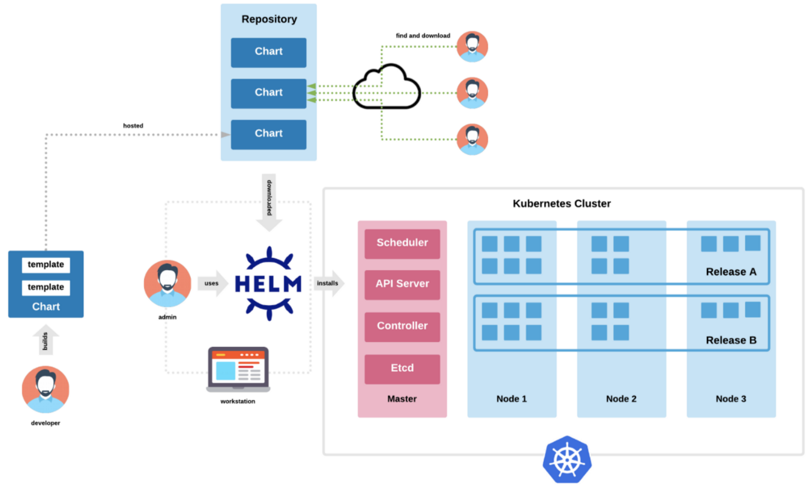
```
▲  Segment1 [main] ⚡  tree mychart
mychart
├── Chart.yaml
├── charts
├── templates
│   ├── NOTES.txt
│   ├── _helpers.tpl
│   ├── deployment.yaml
│   ├── hpa.yaml
│   ├── ingress.yaml
│   ├── service.yaml
│   ├── serviceaccount.yaml
│   └── tests
│       └── test-connection.yaml
└── values.yaml
```

Root Directory
Info related to your chart
Chart Dependencies
Location for your Kubernetes
Parameters/Variables for k8s manifests

Hands-On: Create A Helm Chart

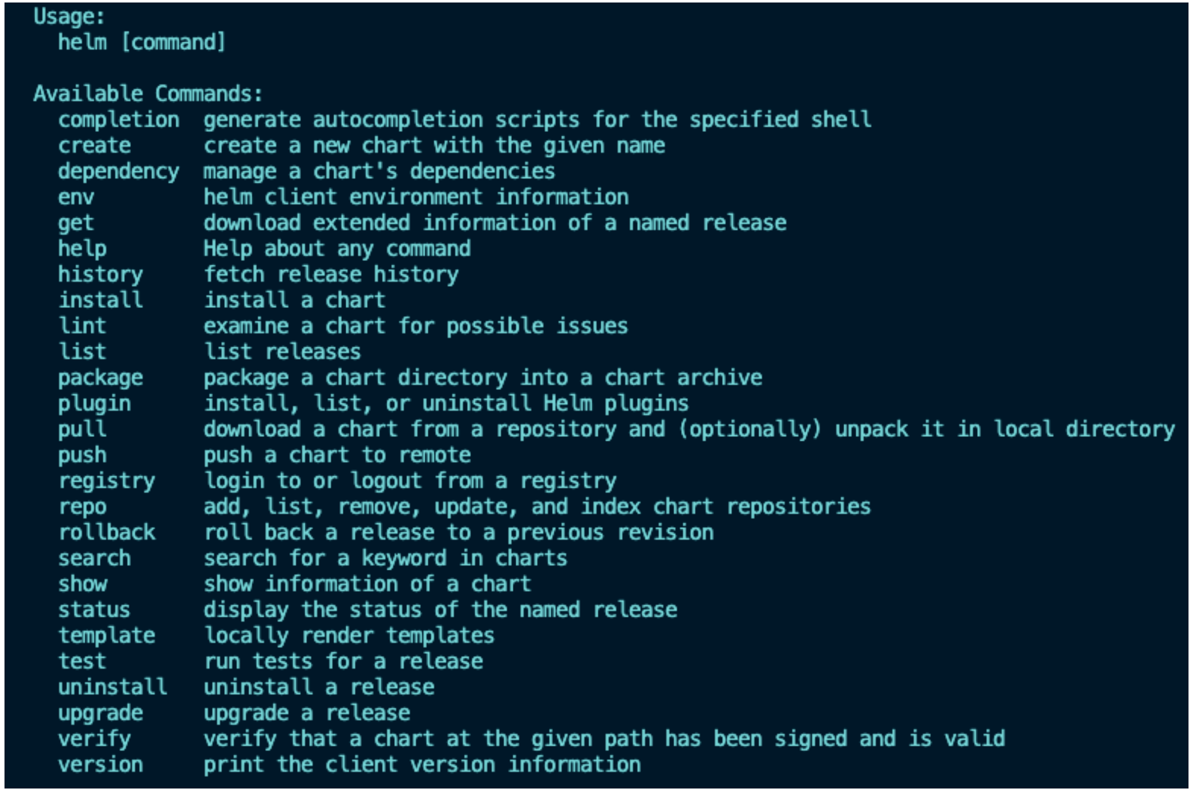
Section 3: Managing Your Apps With Helm

- Commands overview
- Exploring Chart lifecycle and management
- Understand how to deploy, update, rollback, and delete charts
- Real-world Helm testing (--dry-run)
- Generate Helm logs and confirmation of environments



Pearson

Commands Overview



```
Usage:
  helm [command]

Available Commands:
  completion  generate autocompletion scripts for the specified shell
  create      create a new chart with the given name
  dependency  manage a chart's dependencies
  env         helm client environment information
  get         download extended information of a named release
  help        Help about any command
  history     fetch release history
  install     install a chart
  lint        examine a chart for possible issues
  list        list releases
  package     package a chart directory into a chart archive
  plugin      install, list, or uninstall Helm plugins
  pull        download a chart from a repository and (optionally) unpack it in local directory
  push        push a chart to remote
  registry    login to or logout from a registry
  repo        add, list, remove, update, and index chart repositories
  rollback    roll back a release to a previous revision
  search      search for a keyword in charts
  show        show information of a chart
  status      display the status of the named release
  template    locally render templates
  test        run tests for a release
  uninstall   uninstall a release
  upgrade     upgrade a release
  verify      verify that a chart at the given path has been signed and is valid
  version     print the client version information
```

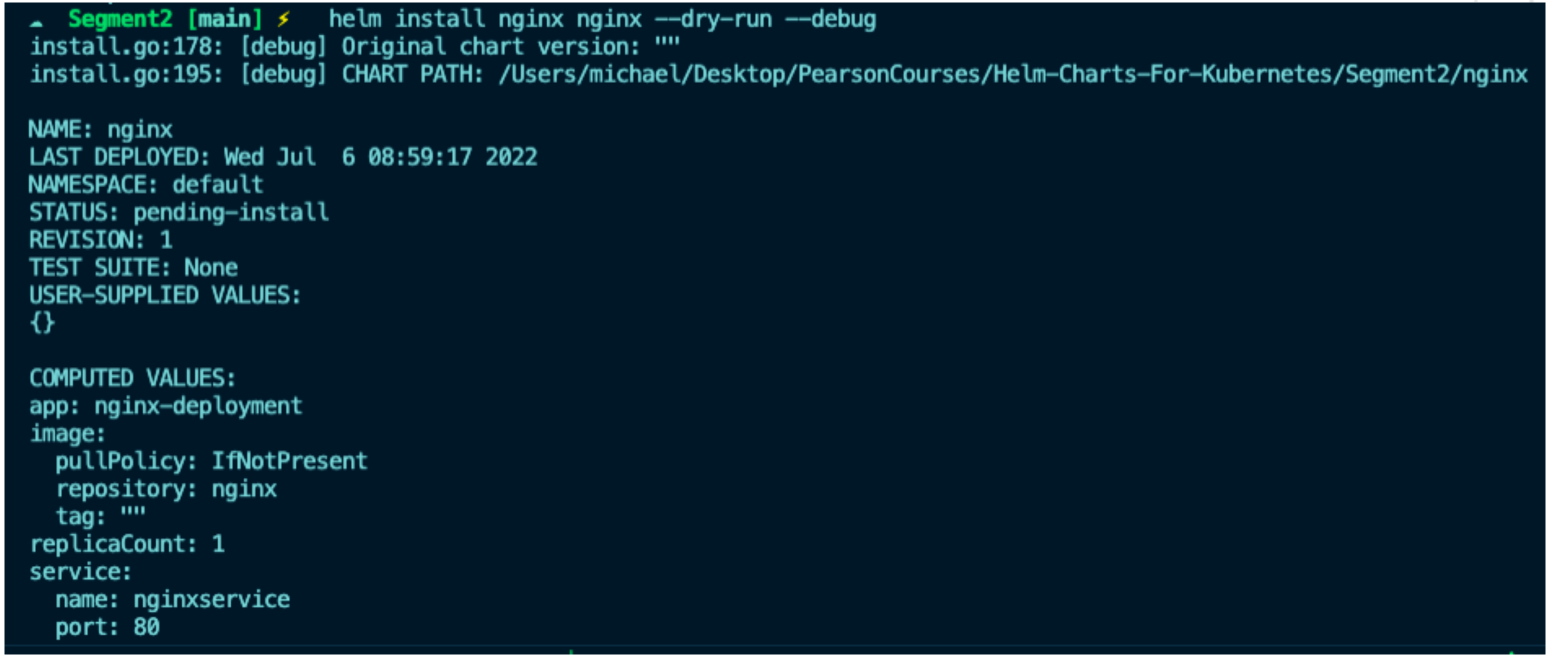
Chart Lifecycle And Management

- `pre-install` Executes after templates are rendered, but before any resources are created in Kubernetes
- `post-install` Executes after all resources are loaded into Kubernetes
- `pre-delete` Executes on a deletion request before any resources are deleted from Kubernetes
- `post-delete` Executes on a deletion request after all of the release's resources have been deleted
- `pre-upgrade` Executes on an upgrade request after templates are rendered, but before any resources are updated
- `post-upgrade` Executes on an upgrade request after all resources have been upgraded
- `pre-rollback` Executes on a rollback request after templates are rendered, but before any resources are rolled back
- `post-rollback` Executes on a rollback request after all resources have been modified
- `test` Executes when the Helm test subcommand is invoked

Understand How To Deploy, Update, Rollback, And Delete Charts

- helm install
- helm upgrade
- helm rollback
- helm uninstall

Helm Dry Run And Testing



```
▲  Segment2 [main] ⚡  helm install nginx nginx --dry-run --debug
install.go:178: [debug] Original chart version: ""
install.go:195: [debug] CHART PATH: /Users/michael/Desktop/PearsonCourses/Helm-Charts-For-Kubernetes/Segment2/nginx

NAME: nginx
LAST DEPLOYED: Wed Jul  6 08:59:17 2022
NAMESPACE: default
STATUS: pending-install
REVISION: 1
TEST SUITE: None
USER-SUPPLIED VALUES:
{}

COMPUTED VALUES:
app: nginx-deployment
image:
  pullPolicy: IfNotPresent
  repository: nginx
  tag: ""
replicaCount: 1
service:
  name: nginxservice
  port: 80
```

Generate Helm logs And Confirmation Of Environments

```
server:
  remoteWrite:
  - url: "<Your Metrics instance remote_write endpoint>"
    basic_auth:
      username: <your_grafana_cloud_prometheus_username>
      password: <your_grafana_cloud_API_key>
```

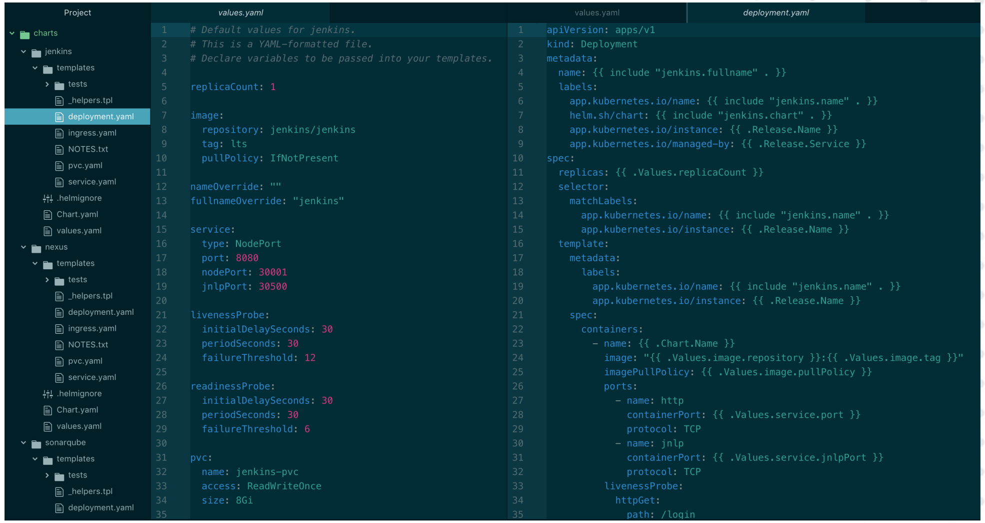
- helm show
- helm list
- helm history

Hands-On

- Update, roll back, and delete charts
- Use –-dry-run and --debug

Section 4: Using Templates To Define Applications And Control Releases

- Create truly generic charts by exploring chart structure and values.yaml
- Use Go templates
- Understand how templating functions provide better control and validation in the Chart configuration

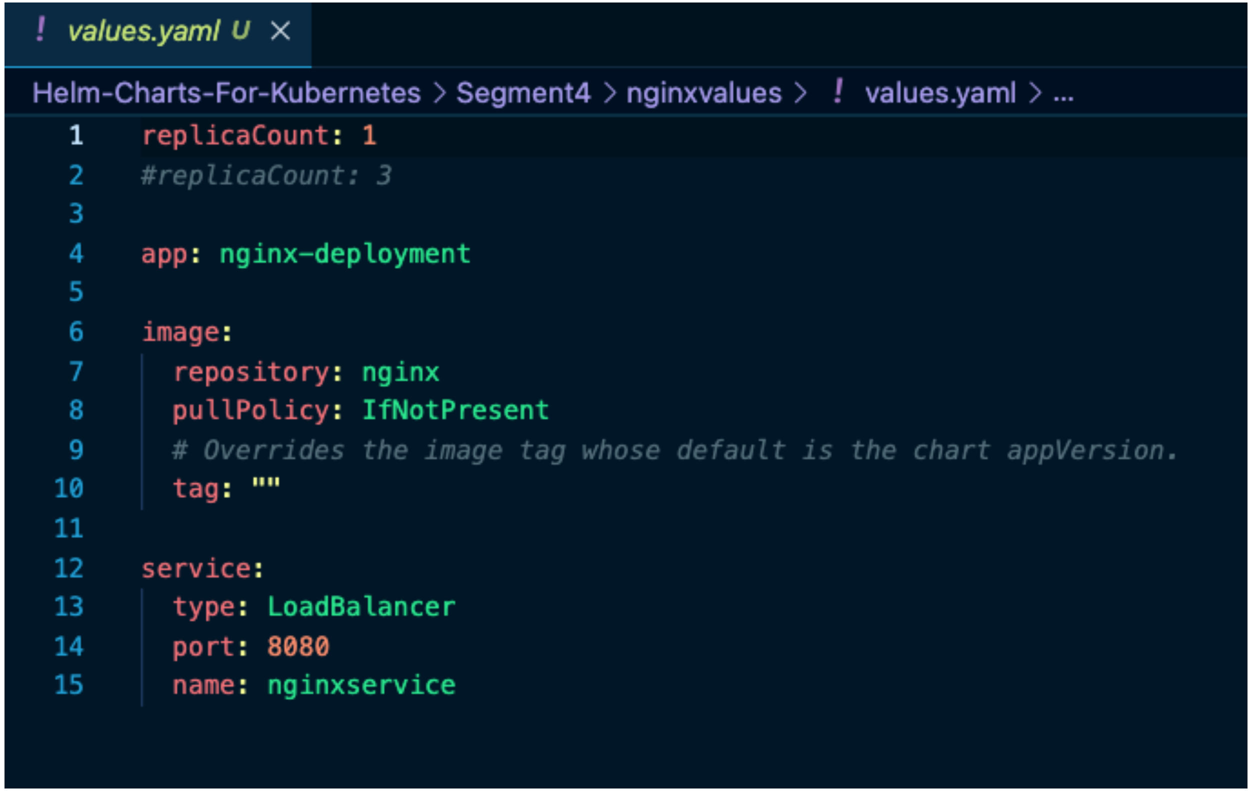


```
charts
  jenkins
    templates
      tests
      _helpers.tpl
      deployment.yaml
      ingress.yaml
      NOTES.txt
      pvc.yaml
      service.yaml
    .helmignore
    Chart.yaml
    values.yaml
  nexus
    templates
      tests
      _helpers.tpl
      deployment.yaml
      ingress.yaml
      NOTES.txt
      pvc.yaml
      service.yaml
    .helmignore
    Chart.yaml
    values.yaml
  sonarqube
    templates
      tests
      _helpers.tpl
      deployment.yaml
```

```yaml
# Default values for jenkins.
# This is a YAML-formatted file.
# Declare variables to be passed into your templates.

replicaCount: 1

image:
  repository: jenkins/jenkins
  tag: lts
  pullPolicy: IfNotPresent

nameOverride: ""
fullnameOverride: "jenkins"

service:
  type: NodePort
  port: 8080
  nodePort: 30001
  jnlpPort: 30500

livenessProbe:
  initialDelaySeconds: 30
  periodSeconds: 30
  failureThreshold: 12

readinessProbe:
  initialDelaySeconds: 30
  periodSeconds: 30
  failureThreshold: 6

pvc:
  name: jenkins-pvc
  access: ReadWriteOnce
  size: 8Gi

```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ include "jenkins.fullname" . }}
  labels:
    app.kubernetes.io/name: {{ include "jenkins.name" . }}
    helm.sh/chart: {{ include "jenkins.chart" . }}
    app.kubernetes.io/instance: {{ .Release.Name }}
    app.kubernetes.io/managed-by: {{ .Release.Service }}
spec:
  replicas: {{ .Values.replicaCount }}
  selector:
    matchLabels:
      app.kubernetes.io/name: {{ include "jenkins.name" . }}
      app.kubernetes.io/instance: {{ .Release.Name }}
  template:
    metadata:
      labels:
        app.kubernetes.io/name: {{ include "jenkins.name" . }}
        app.kubernetes.io/instance: {{ .Release.Name }}
    spec:
      containers:
        - name: {{ .Chart.Name }}
          image: "{{ .Values.image.repository }}:{{ .Values.image.tag }}"
          imagePullPolicy: {{ .Values.image.pullPolicy }}
          ports:
            - name: http
              containerPort: {{ .Values.service.port }}
              protocol: TCP
            - name: jnlp
              containerPort: {{ .Values.service.jnlpPort }}
              protocol: TCP
          livenessProbe:
            httpGet:
              path: /login
```

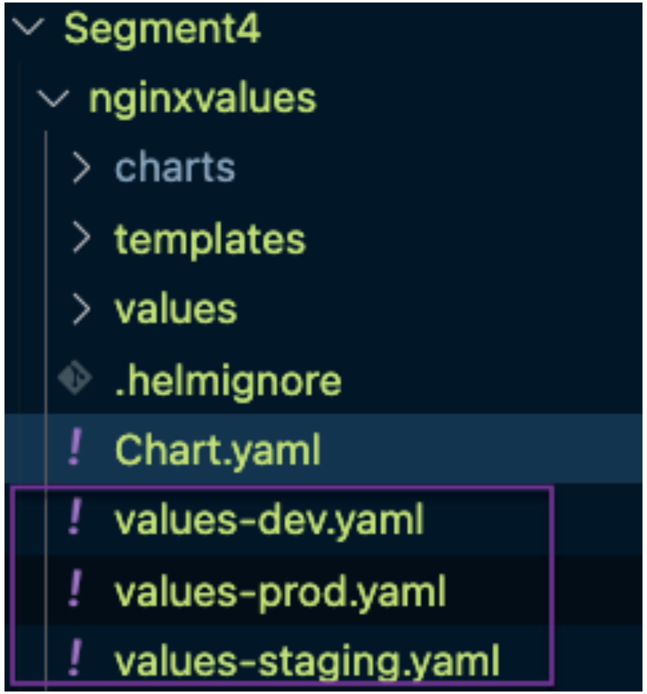
Values.yaml


! values.yaml U ×
Helm-Charts-For-Kubernetes > Segment4 > nginxvalues > ! values.yaml > ...

```yaml
replicaCount: 1
#replicaCount: 3

app: nginx-deployment

image:
  repository: nginx
  pullPolicy: IfNotPresent
  # Overrides the image tag whose default is the chart appVersion.
  tag: ""

service:
  type: LoadBalancer
  port: 8080
  name: nginxservice
```

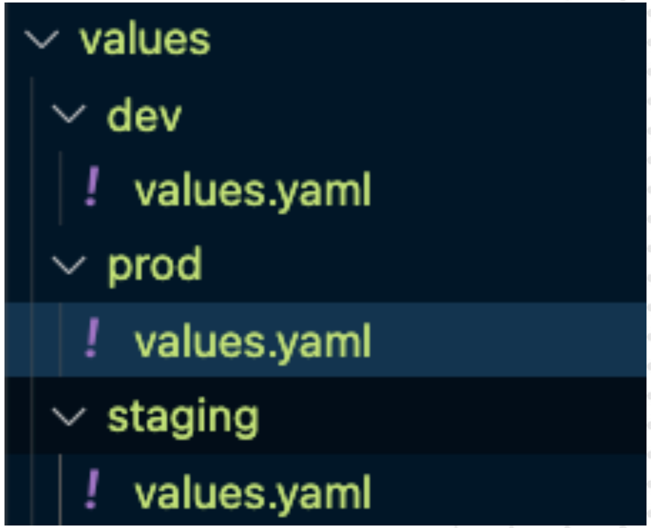
Pearson

Values.yaml Structure



```
Segment4
  nginxvalues
    charts
    templates
    values
    .helmignore
    Chart.yaml
    values-dev.yaml
    values-prod.yaml
    values-staging.yaml
```

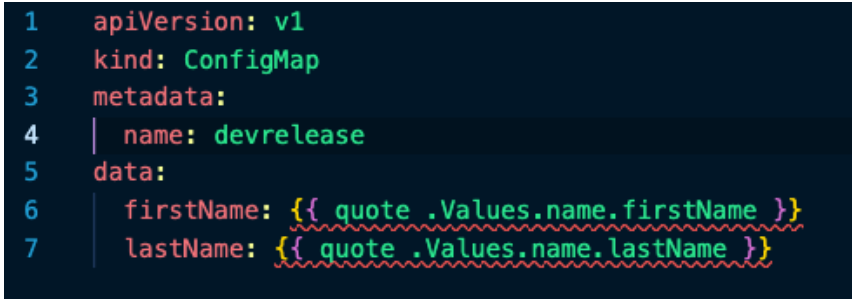
VS



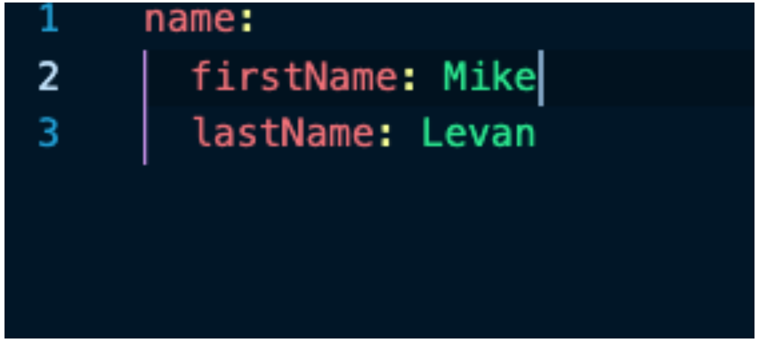
```
values
  dev
    values.yaml
  prod
    values.yaml
  staging
    values.yaml
```

Pearson

Go Templates (template functions)

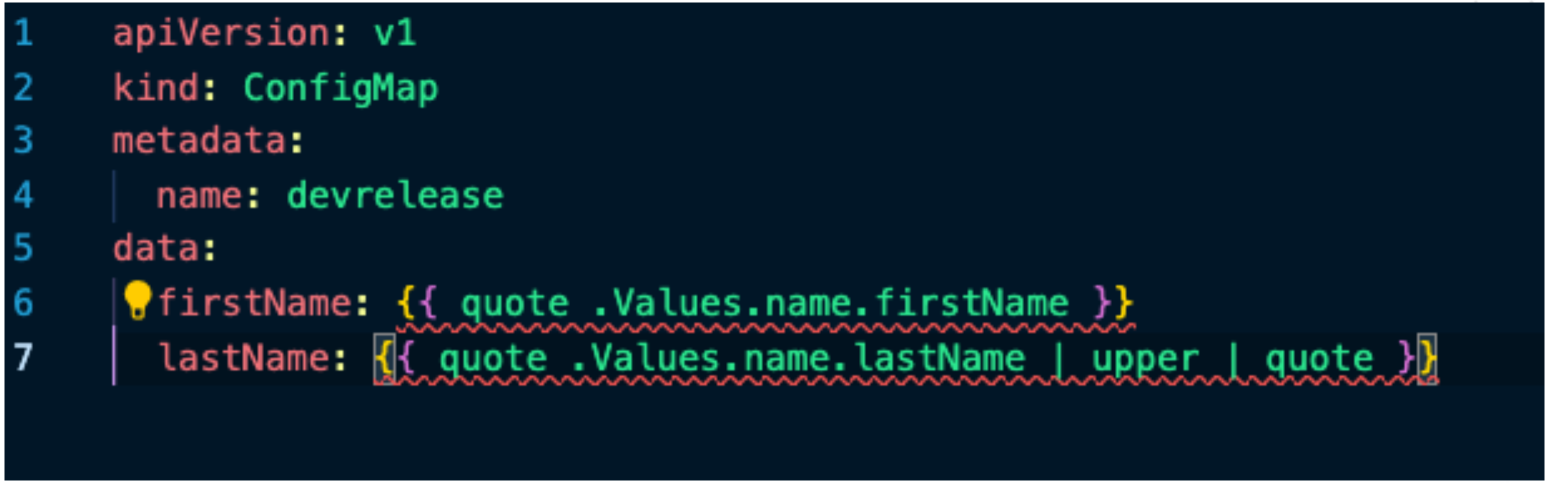


```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: devrelease
data:
  firstName: {{ quote .Values.name.firstName }}
  lastName: {{ quote .Values.name.lastName }}
```



```yaml
name:
  firstName: Mike
  lastName: Levan
```

Pipeline Templates



```
apiVersion: v1
kind: ConfigMap
metadata:
  name: devrelease
data:
  firstName: {{ quote .Values.name.firstName }}
  lastName: {{ quote .Values.name.lastName | upper | quote }}
```

How Templating Functions Provide Better Control

- Manipulate data

Helm Chart Best Practices

- 1: General Conventions
- 2: Value Files
- 3: Templates
- 4: Dependencies
- 5: Labels and Annotations
- 6: Pod and Pod Templates
- 7: CRDs
- 8: RBAC

General Conventions

- Chart names
- Hyphens
- Upper case
- Lower case
- Dots (.)

Value Files

- Lower case names
- Words should be camelCase

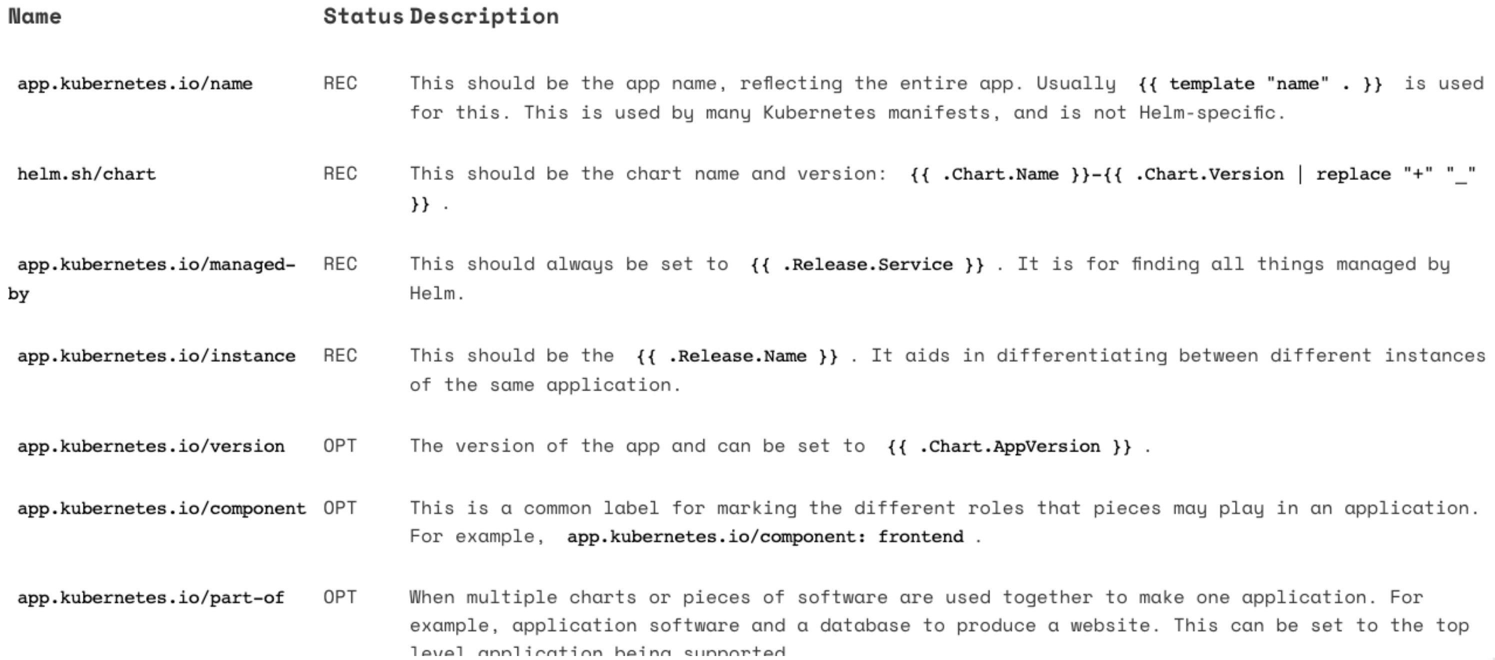
Templates

- Directory structure

Dependencies

- Use ranges for versions for patching

Labels



Name	Status	Description
app.kubernetes.io/name	REC	This should be the app name, reflecting the entire app. Usually `{{ template "name" . }}` is used for this. This is used by many Kubernetes manifests, and is not Helm-specific.
helm.sh/chart	REC	This should be the chart name and version: `{{ .Chart.Name }}-{{ .Chart.Version | replace "+" "_" }}` .
app.kubernetes.io/managed-by	REC	This should always be set to `{{ .Release.Service }}` . It is for finding all things managed by Helm.
app.kubernetes.io/instance	REC	This should be the `{{ .Release.Name }}` . It aids in differentiating between different instances of the same application.
app.kubernetes.io/version	OPT	The version of the app and can be set to `{{ .Chart.AppVersion }}` .
app.kubernetes.io/component	OPT	This is a common label for marking the different roles that pieces may play in an application. For example, `app.kubernetes.io/component: frontend` .
app.kubernetes.io/part-of	OPT	When multiple charts or pieces of software are used together to make one application. For example, application software and a database to produce a website. This can be set to the top level application being supported

Pod Specs and Pod Templates

- Fixed tag for container images
- imagePull Policies

CRDs

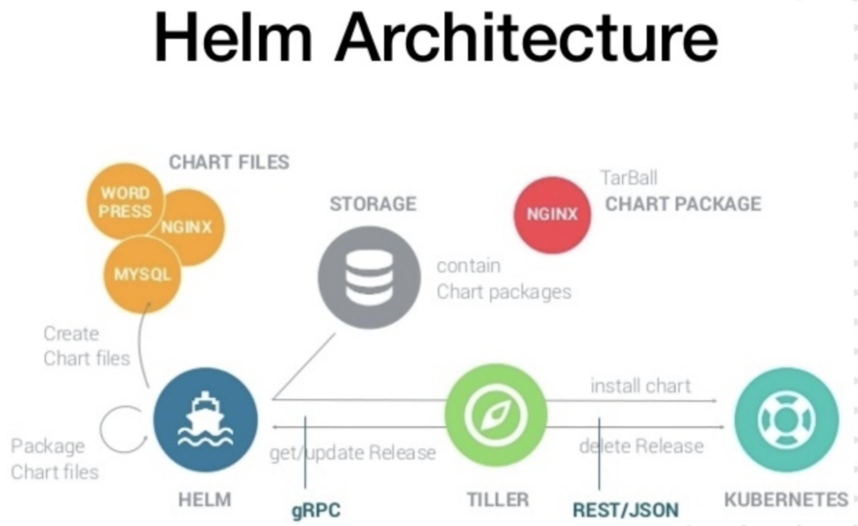
- Designed for speed

RBAC

- Use proper policies for users, groups, and service accounts that are running Pods

Section 5: Create A Real-World Helm Chart

- Common use-case for Helm in production
- Deploy an official environment with Helm



Helm Chart Production Use-Case

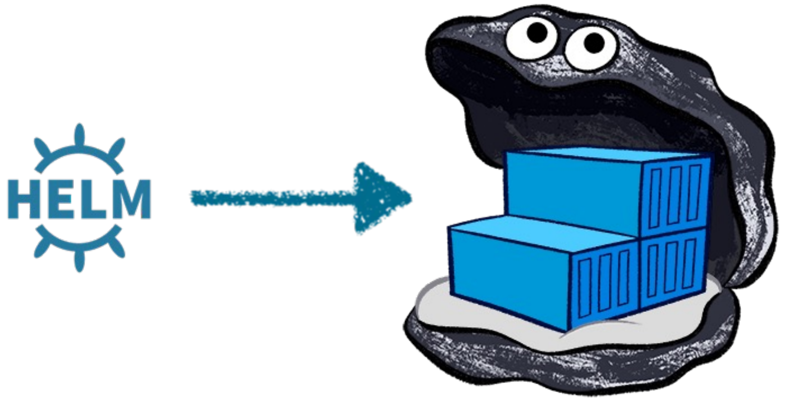
- Manage multiple environments
- Code base that you already know
- Sets a standard
- Packaged up like an app

Pearson

Real-World Demo Environment

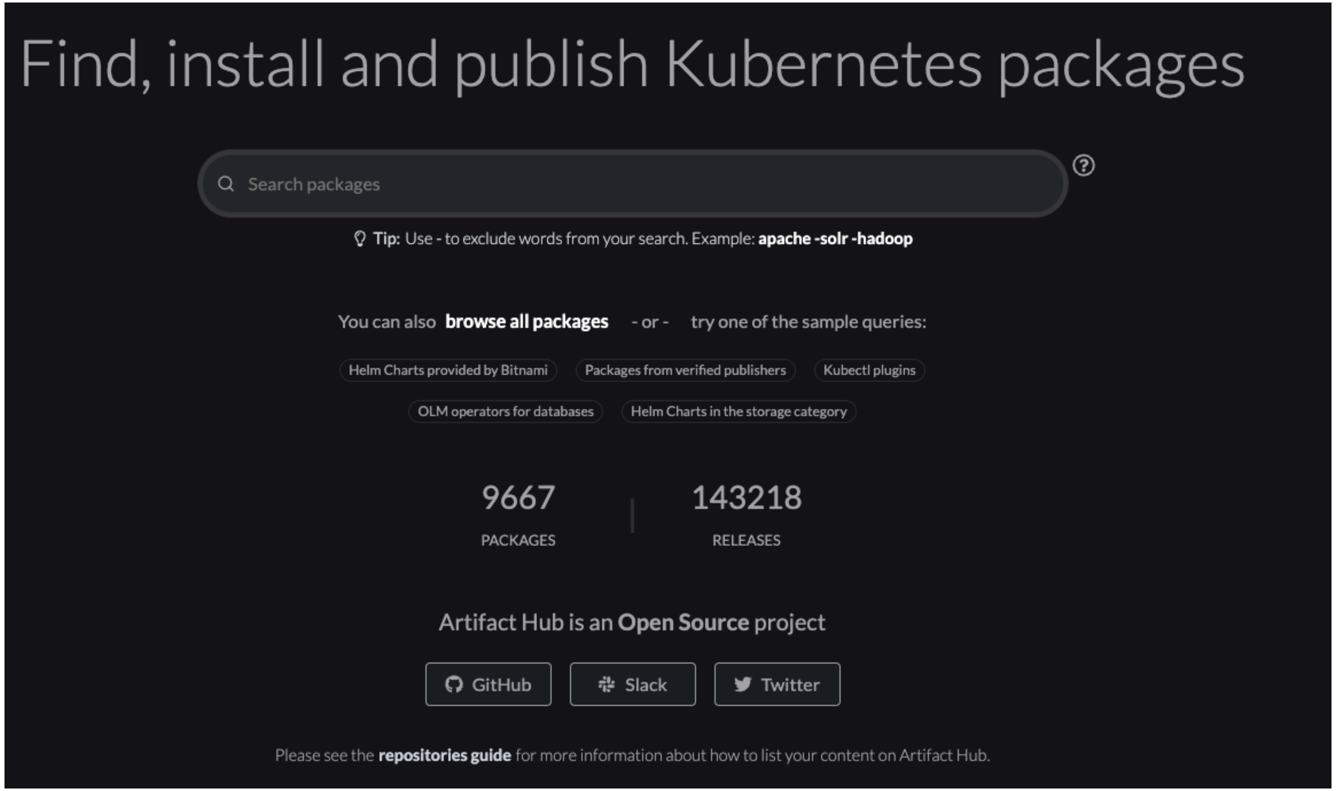
- Prometheus
- Ingress-nginx

Section 6: How To Get Involved With The Helm Project?



- Share a chart with the world
- Use Public registries

Hosting a Helm chart on Artifact Hub



Find, install and publish Kubernetes packages

Search packages

Tip: Use - to exclude words from your search. Example: **apache -solr -hadoop**

You can also **browse all packages** - or - try one of the sample queries:

Helm Charts provided by Bitnami | Packages from verified publishers | Kubectl plugins | OLM operators for databases | Helm Charts in the storage category

9667 PACKAGES | 143218 RELEASES

Artifact Hub is an **Open Source** project

GitHub | Slack | Twitter

Please see the **repositories guide** for more information about how to list your content on Artifact Hub.

Creating Your Own Repo

- Google Cloud Storage
- CloudSmith
- JFrog Artifactory
- GitHub Pages
- GitLab Package Registry

Demo

- Push a Helm Chart to GitHub Pages
- Push a Helm Chart to artifacthub.io