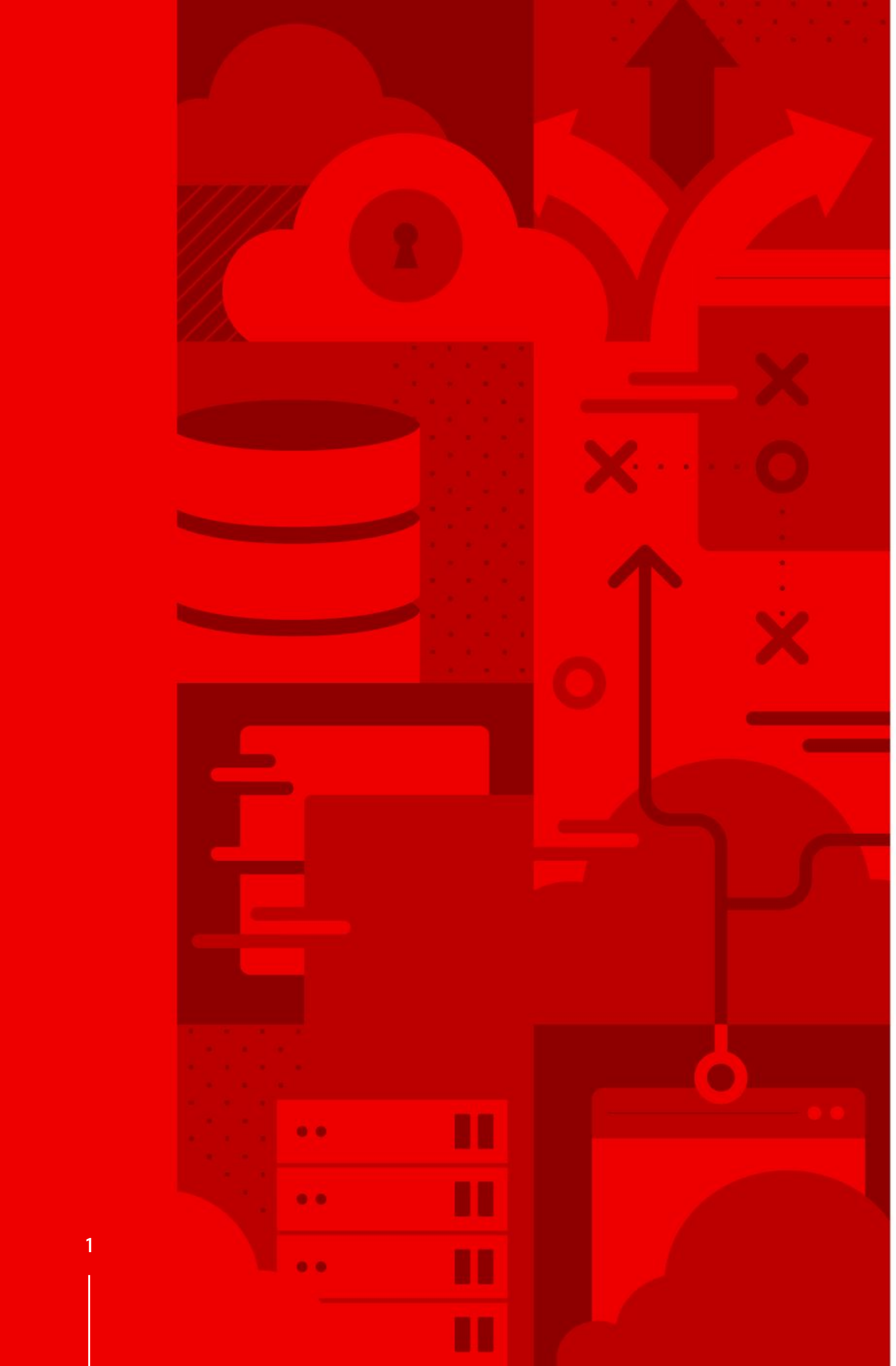


Helm for Developers

Kubernetes made easy(ier)

Andrew Block
Senior Principal Consultant

Developing Containerized Cloud Native Applications is Hard

While deploying containerized applications in a cloud environment yields many benefits, there is an increased level of ownership and work required to manage the entire set of required components.

Images

Applications deployed as images need to be managed and sourced from an image registry.

Application Configuration

Environment dependant values used by the application.

Infrastructure Configuration

Values used to specify the components to support the application.

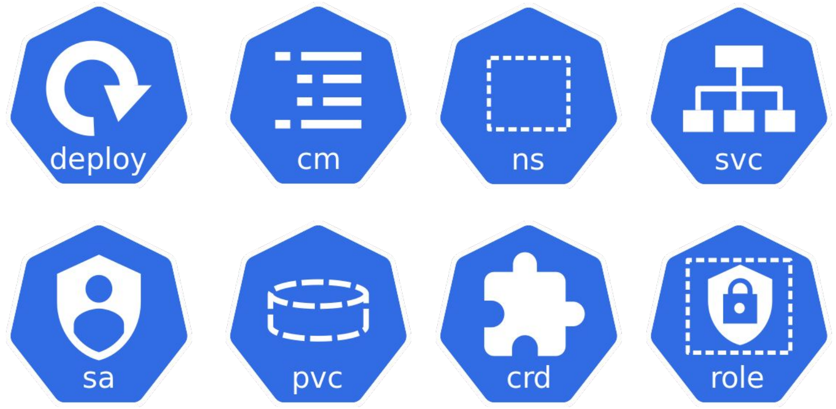
Declarative Configuration

Many cloud native platforms and frameworks specify their configurations via YAML formatted files.

Red Hat

Kubernetes Application Composition

Kubernetes contains a vast ecosystem of resources that can describe an application deployment



How do you manage it all effectively?

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: httpd-deployment
  labels:
    app: httpd
spec:
  replicas: 3
  selector:
    matchLabels:
      app: httpd
  template:
    metadata:
      labels:
        app: httpd
    spec:
      containers:
      - name: httpd
        image:
registry.redhat.io/rhscl/httpd-24-rhel7:2.4
        ports:
        - containerPort: 8080
```

Would it be nice if managing applications on Kubernetes was just like any other framework?

Package managers enable individuals with knowledge of an application the ability for to have another entity that may not have pre existing knowledge the ability to leverage the application successfully

- `yum install <name>`
- `apt-get install <name>`
- `brew install <name>`
- `choco install <name>`
- `pip install <name>`
- `npm install <name>`

...

HELM

Package manager for Kubernetes

CLOUD NATIVE COMPUTING FOUNDATION

Project Overview

- https://helm.sh/
- https://github.com/helm/helm

Top level CNCF Project

- 2016 – Joined CNCF
- 2020 – Graduated status

Active development community

- 13,000+ contributors
- 1,700+ contributing companies
- 9,500+ code commits

Helm Primary Components

CLI

The *helm* binary provides a mechanism for interacting with the helm ecosystem

Templates

Provides dynamic capabilities for Kubernetes resources that are to be instantiated

Charts

Packages representing Kubernetes deployable resources

Values

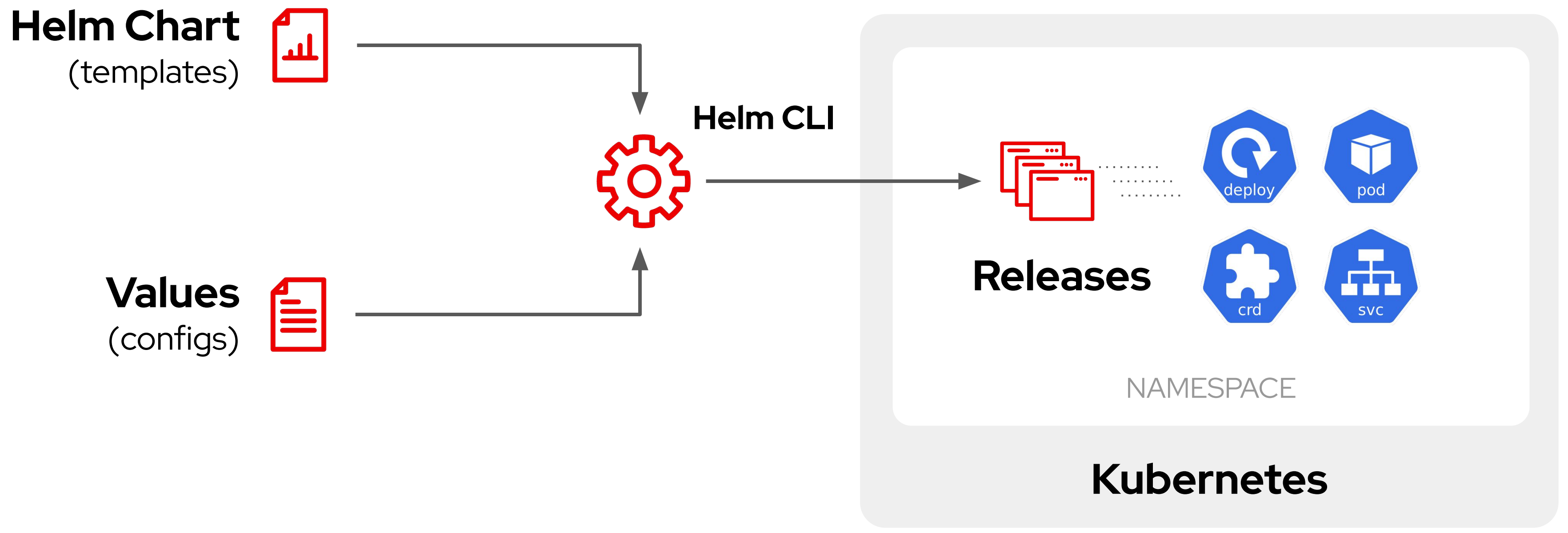
Configuration variables that are injected into templated resources

Revisions

Configurations of a chart at a particular point in time

Red Hat

Helm Primary Components



Helm Fundamentals

Understanding the basic concepts of Helm will provide the necessary information for creating your own charts

Creating and Deploying a Helm Chart

Zero to hero in a few short commands

- Creating a new Chart

```
$ helm create opendevhour
```

- Install Chart to Kubernetes cluster

```
$ helm install opendevhour
```

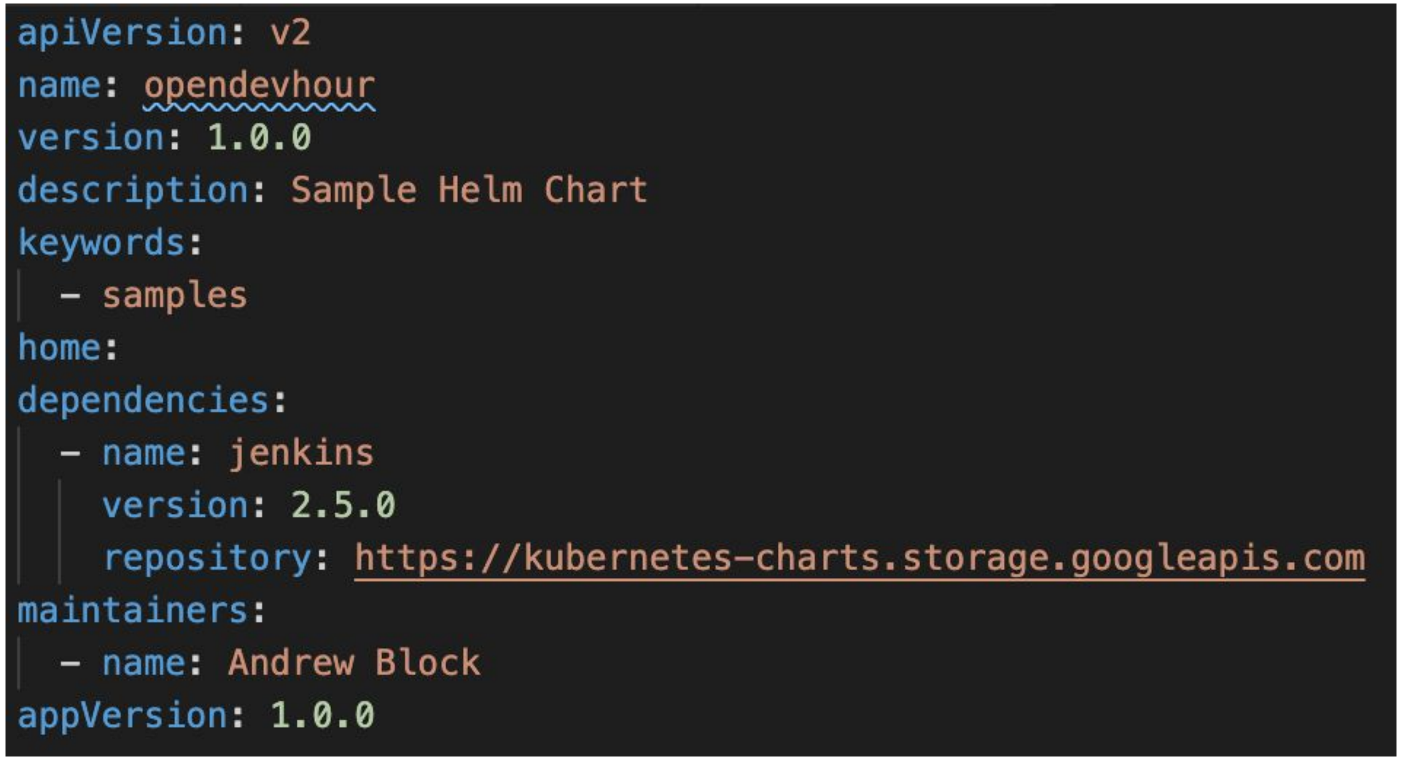
```
opendevhour/
  Chart.yaml          # Information about the chart
  LICENSE             # OPTIONAL: Chart license
  README.md           # OPTIONAL: README file
  values.yaml         # The default configuration values
  values.schema.json  # OPTIONAL: A JSON Schema for values
  charts/             # Dependency charts
  crds/               # Custom Resource Definitions
  templates/          # Directory of templates
  templates/NOTES.txt # OPTIONAL: Usage notes
```

Helm Chart Directory Structure

Red Hat

Chart.yaml

Helm metadata file



```yaml
apiVersion: v2
name: opendevhour
version: 1.0.0
description: Sample Helm Chart
keywords:
  - samples
home:
dependencies:
  - name: jenkins
    version: 2.5.0
    repository: https://kubernetes-charts.storage.googleapis.com
maintainers:
  - name: Andrew Block
appVersion: 1.0.0
```

Setting Chart Values

Values for a chart can be overridden by values contained in files or explicitly set

Files

```
$ helm install -f <values_file>
./opendevhour
```

Command Line

```
$ helm install --set foo=bar ./opendevhour
```

Multiple values can be specified

Red Hat

Managing Charts

- Listing Charts

```
$ helm list
```

- Upgrading a release

```
$ helm upgrade <release_name> <chart>
--set version=1.1
```

- Rolling back an upgrade

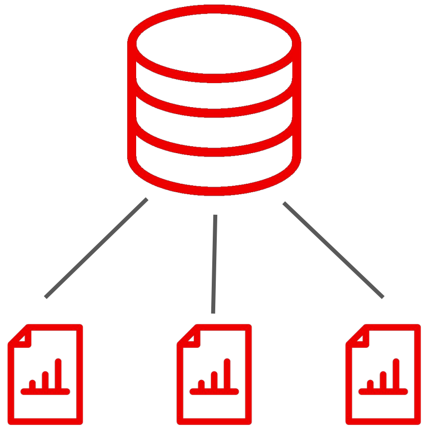
```
$ helm rollback <release> <revision>
```

- Uninstalling a Chart

```
$ helm uninstall <release>
```

Locating Charts in Repositories

Share and source charts from Repositories to accelerate productivity



Repository management

The `helm repo` command can be used to manage repositories

Installing charts from repositories

The helm repo subcommand

```
$ helm install redhat-cop/jenkins --generate-name
```

Searching for charts

Charts located within repositories can be searched by keywords

```
$ helm search repo nginx
```

Helm Templating

Programming Kubernetes resources

Templates and Values

Working together to bring your chart to life

Templates:

- Located under the *templates* directory
- Uses a combination of go templates and sprig functions

```
{{ .Values.replicaCount }}
```

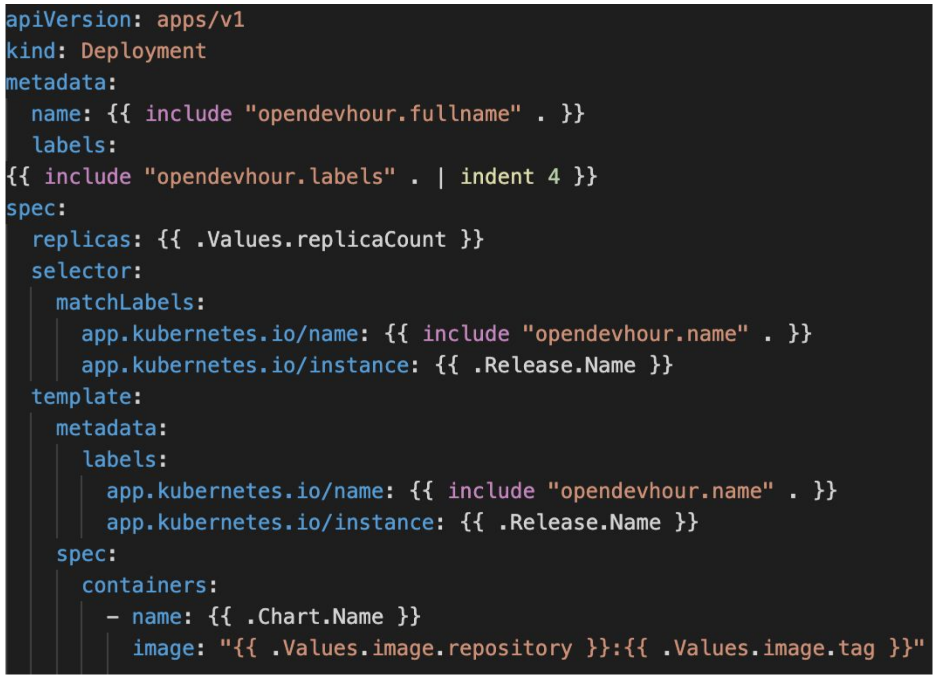
Values:

- Collection of key=value pairs to define the configuration of a chart
- *Values.yaml* is the default, baseline source

```
replicaCount: 2
```

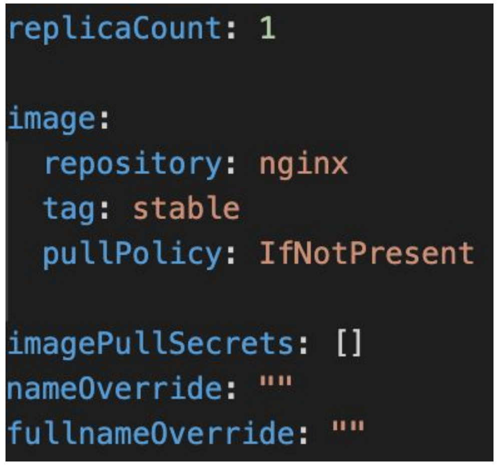
Red Hat

Templates and Values



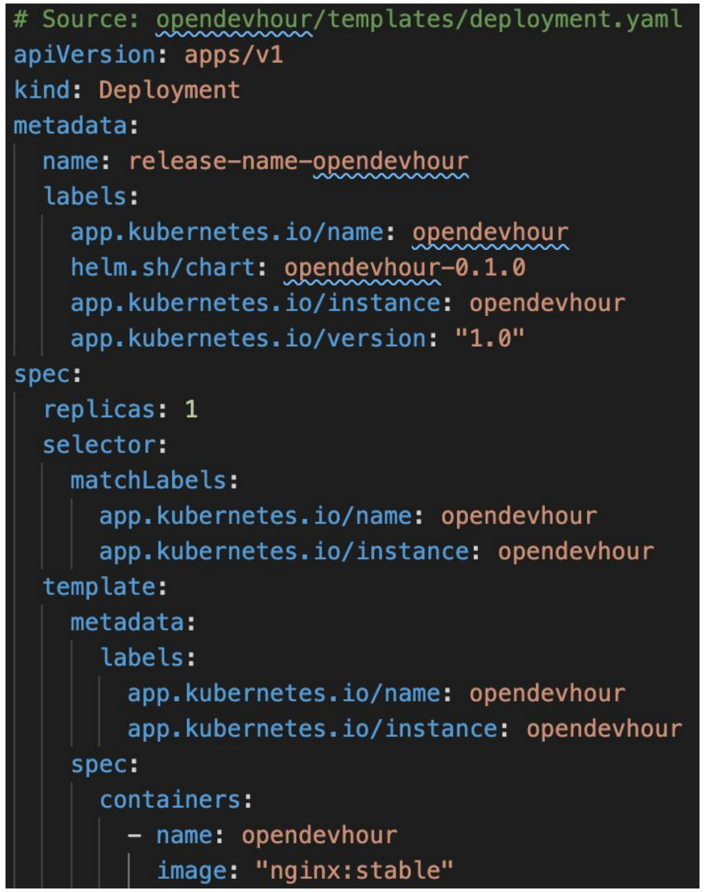
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ include "opendevhour.fullname" . }}
  labels:
{{ include "opendevhour.labels" . | indent 4 }}
spec:
  replicas: {{ .Values.replicaCount }}
  selector:
    matchLabels:
      app.kubernetes.io/name: {{ include "opendevhour.name" . }}
      app.kubernetes.io/instance: {{ .Release.Name }}
  template:
    metadata:
      labels:
        app.kubernetes.io/name: {{ include "opendevhour.name" . }}
        app.kubernetes.io/instance: {{ .Release.Name }}
    spec:
      containers:
        - name: {{ .Chart.Name }}
          image: "{{ .Values.image.repository }}:{{ .Values.image.tag }}"
```

Template



```yaml
replicaCount: 1

image:
  repository: nginx
  tag: stable
  pullPolicy: IfNotPresent

imagePullSecrets: []
nameOverride: ""
fullnameOverride: ""
```

Values



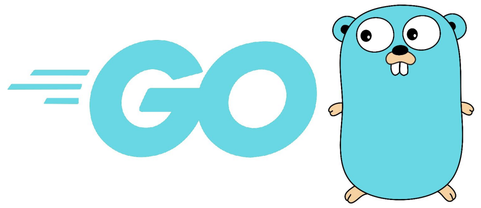
```yaml
# Source: opendevhour/templates/deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: release-name-opendevhour
  labels:
    app.kubernetes.io/name: opendevhour
    helm.sh/chart: opendevhour-0.1.0
    app.kubernetes.io/instance: opendevhour
    app.kubernetes.io/version: "1.0"
spec:
  replicas: 1
  selector:
    matchLabels:
      app.kubernetes.io/name: opendevhour
      app.kubernetes.io/instance: opendevhour
  template:
    metadata:
      labels:
        app.kubernetes.io/name: opendevhour
        app.kubernetes.io/instance: opendevhour
    spec:
      containers:
        - name: opendevhour
          image: "nginx:stable"
```

Leverage the Built in Objects

Helm exposes a variety of resources for developers that can be used within templates

Object	**Definition**
`.Chart`	Contents of the `Chart.yaml` file
`.Release`	Assets related to the release
`.Values`	Values associated with the chart
`.Files`	Provides access to files within the chart
`.Capabilities`	Characteristics of the Kubernetes environment
`.Template`	Information related to the template being executed

Template Functions



Over 60 functions are provided out of the box using a combination of go templates and Spring functions

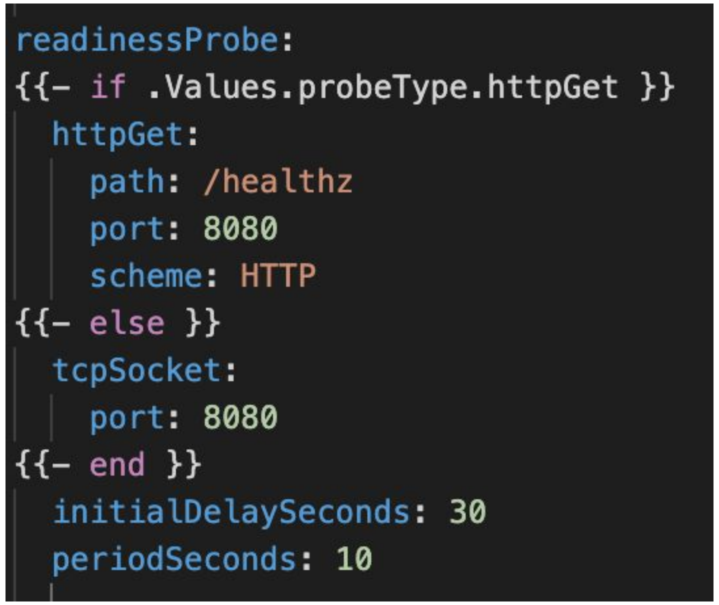
- Cryptographic and Security
- Date
- Dictionaries
- Encoding
- File Path
- Kubernetes and Chart
- Logic and Flow Control
- Lists
- Math
- Network
- Reflection
- Regular Expressions
- Semantic Versions
- String
- Type Conversion
- URL
- UUID

Red Hat

Flow Control

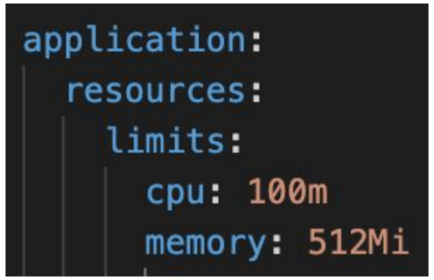
Control the flow of template generation

- `if/else` for creating conditional blocks
- `with` to specify a scope
- `range`, which provides a "for each"-style loop

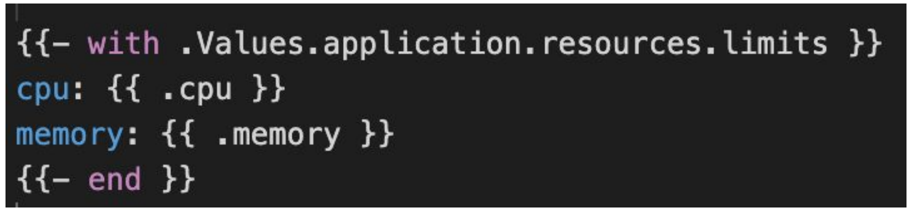


```
readinessProbe:
{{- if .Values.probeType.httpGet }}
  httpGet:
    path: /healthz
    port: 8080
    scheme: HTTP
{{- else }}
  tcpSocket:
    port: 8080
{{- end }}
  initialDelaySeconds: 30
  periodSeconds: 10
```

if/else



```
application:
  resources:
    limits:
      cpu: 100m
      memory: 512Mi
```



```
{{- with .Values.application.resources.limits }}
cpu: {{ .cpu }}
memory: {{ .memory }}
{{- end }}
```

with

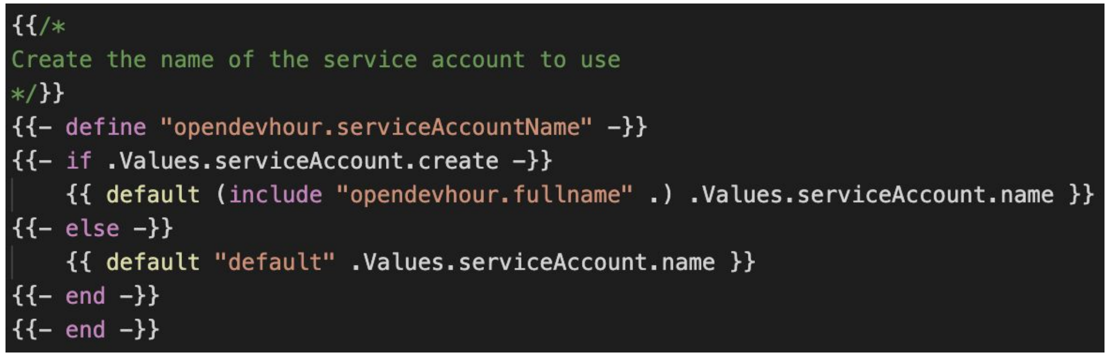
Named Templates

Template resources defined in one file and being used in another

- New charts create a `templates/_helpers.tpl` with boilerplate content
- Also known as *partials* or *subtemplates*
- A named template created using the `define` keyword
- `include` or `template` can be used to reference the named template
- Allows for dynamic, complex logic to be created

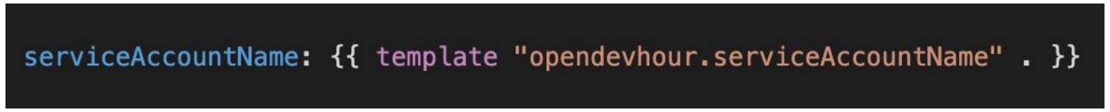
Named Templates

Named template defined in a *_helpers.tpl* file



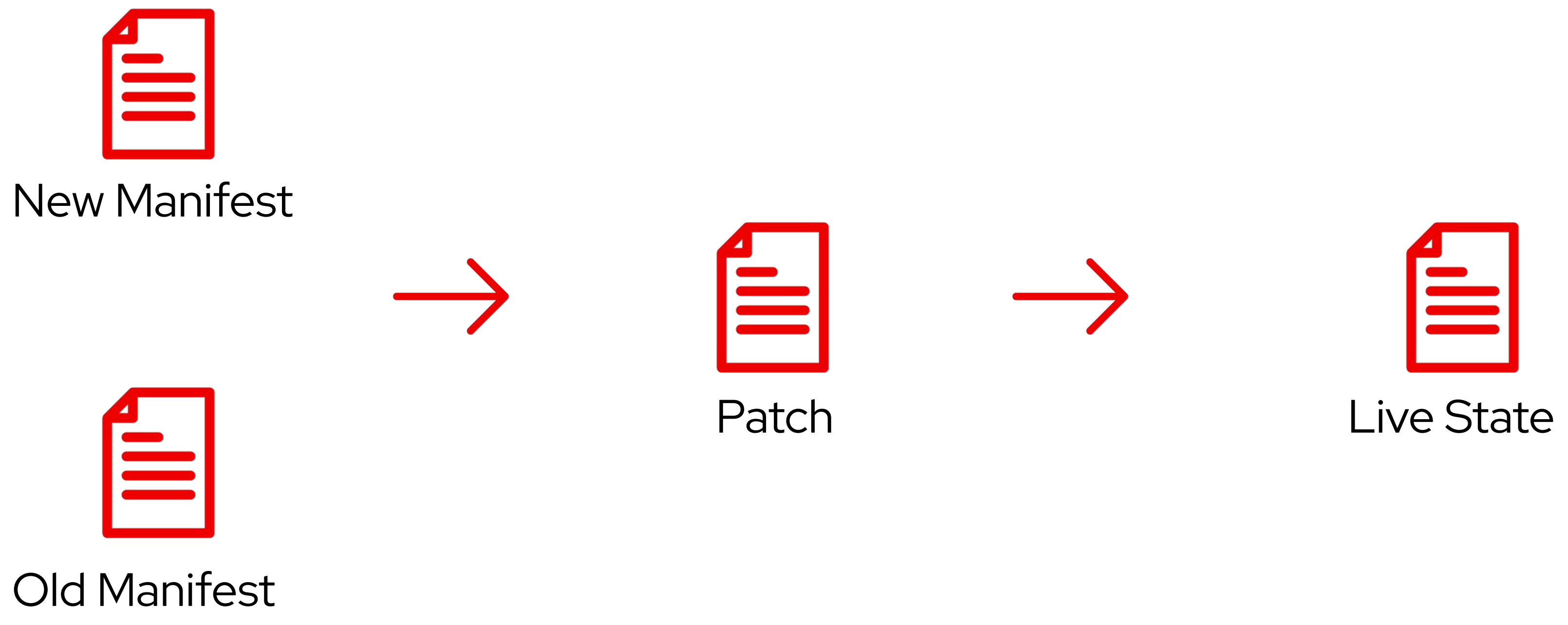
```
{{/*
Create the name of the service account to use
*/}}
{{- define "opendevhour.serviceAccountName" -}}
{{- if .Values.serviceAccount.create -}}
    {{ default (include "opendevhour.fullname" .) .Values.serviceAccount.name }}
{{- else -}}
    {{ default "default" .Values.serviceAccount.name }}
{{- end -}}
{{- end -}}
```

Inclusion in a deployment.yaml template



```
serviceAccountName: {{ template "opendevhour.serviceAccountName" . }}
```

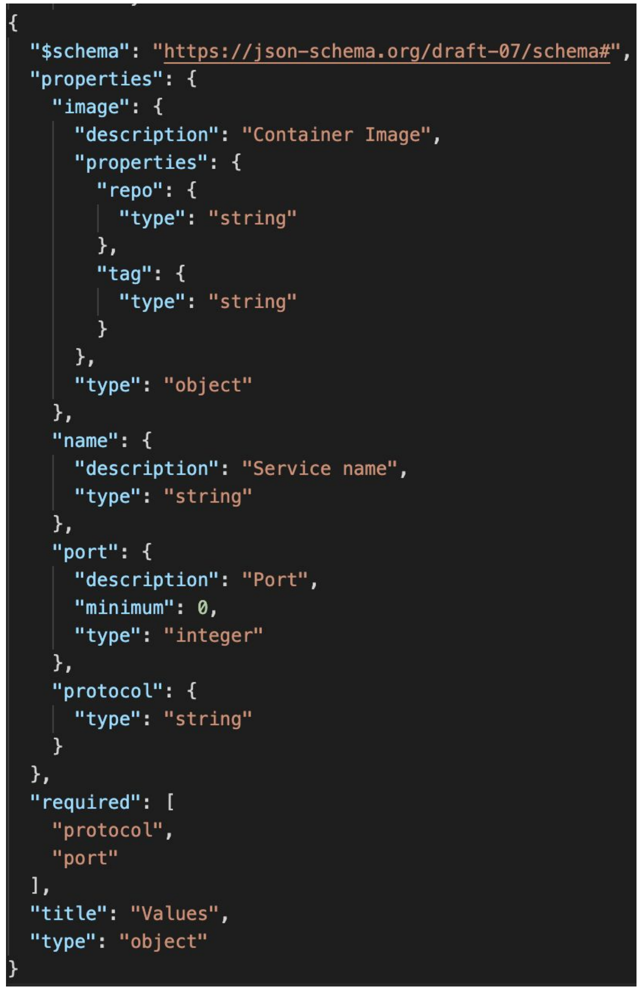
Three Way Strategic Merge



Improved method of applying manifests against existing resources

JSON Schema Validation

- Validation of Values for a chart using [JSON Schema]
- Defined in a `values.schema.json` file
- Validation occurs with `helm install`, `helm upgrade`, `helm lint` and `helm template`



```json
{
  "$schema": "https://json-schema.org/draft-07/schema#",
  "properties": {
    "image": {
      "description": "Container Image",
      "properties": {
        "repo": {
          "type": "string"
        },
        "tag": {
          "type": "string"
        }
      },
      "type": "object"
    },
    "name": {
      "description": "Service name",
      "type": "string"
    },
    "port": {
      "description": "Port",
      "minimum": 0,
      "type": "integer"
    },
    "protocol": {
      "type": "string"
    }
  },
  "required": [
    "protocol",
    "port"
  ],
  "title": "Values",
  "type": "object"
}
```

Full Stack Support

Helm provides the capabilities of managing the full lifecycle of an application and the integration with external components

“Hook”ing into the Lifecycle

Performing actions at different points of the Helm release process

- Commonly implemented as Kubernetes Jobs
- Declared using the `helm.sh/hook` annotation
- Enables full lifecycle management of Helm resources
- Common use cases:
  - Waiting for dependencies to be installed
  - Loading configurations prior to install
  - Database upgrades during chart upgrades

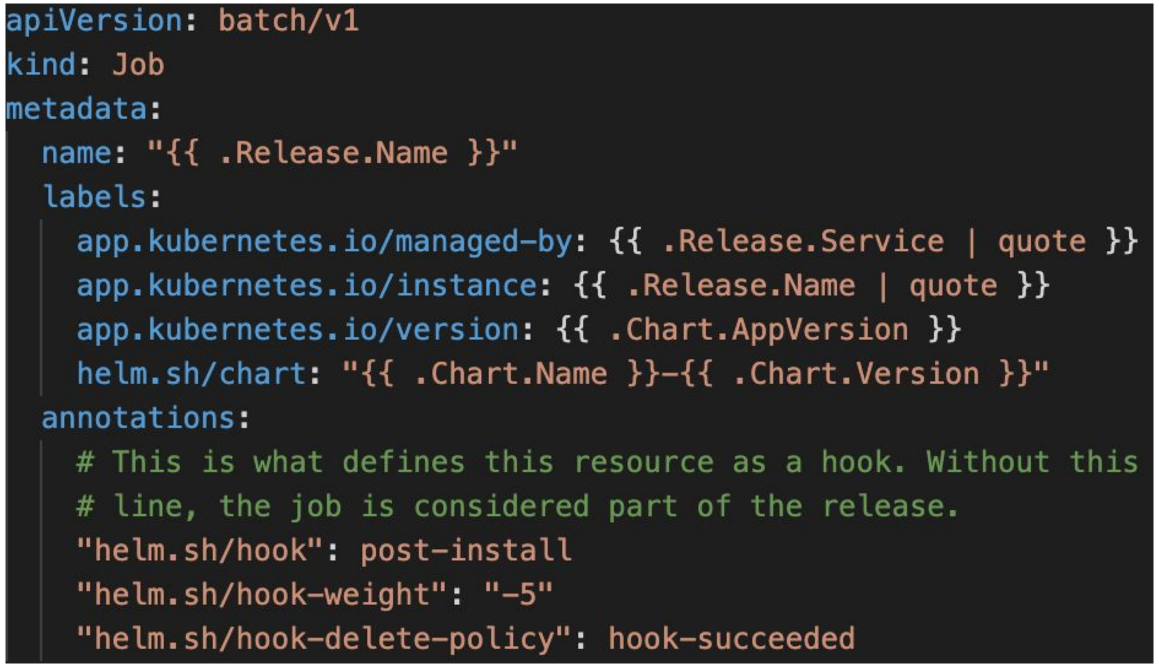
“Hook”ing into the Lifecycle

Hook execution points:

- `pre-install`, `post-install`, `pre-delete`, `post-delete`, `pre-upgrade`, `post-upgrade`, `pre-rollback`, `post-rollback`, and `test`

Deletion policies

- When hook related resources should be deleted
- `before-hook-creation`, `hook-succeeded`, and `hook-failed`



```yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: "{{ .Release.Name }}"
  labels:
    app.kubernetes.io/managed-by: {{ .Release.Service | quote }}
    app.kubernetes.io/instance: {{ .Release.Name | quote }}
    app.kubernetes.io/version: {{ .Chart.AppVersion }}
    helm.sh/chart: "{{ .Chart.Name }}-{{ .Chart.Version }}"
  annotations:
    # This is what defines this resource as a hook. Without this
    # line, the job is considered part of the release.
    "helm.sh/hook": post-install
    "helm.sh/hook-weight": "-5"
    "helm.sh/hook-delete-policy": hook-succeeded
```

Testing Charts

Testing can be performed to verify the integrity of chart resources and expected actions

```
# Install chart
$ helm install <release_name> <chart>

# Execute tests
$ helm test <release_name>
```

- Tests stored in the `templates/tests` directory
- Extension of Helm hooks
  - Resources annotated with `helm.sh: test`
- Executed via the `helm test` command
- Use cases:
  - Application availability
  - Proper resource rendering

Red Hat

Additional Testing Tools

yamllint

yamllint is a YAML Linter to verify the correctness of YAML formatted files

```
# Install yamllint
$ pip install yamllint

# Render Templates and Test
$ cat -n <(helm template <release_name>
<chart>)
```

Chart Testing CLI Tool

Ct is a utility to lint charts and validate charts in a running cluster

- Linting
    - Contains yamllint and Yamale tools
- Installing into a cluster
    - Deploys chart and execute test suites
- Conformance
    - Validates chart version incremented

Continuous Integration and Continuous Delivery

Manage Helm Charts in a similar fashion as any other application

Integrate into existing CI/CD Tools

The management of helm and subsequent releases can be integrated into existing CI/CD tools such as Jenkins, TravisCI and GitHub Actions (and many more)

Types of activities

Common activities as part of a CI/CD pipeline include *chart conformance*, *integration testing*, and *release management*

Leverage community assets

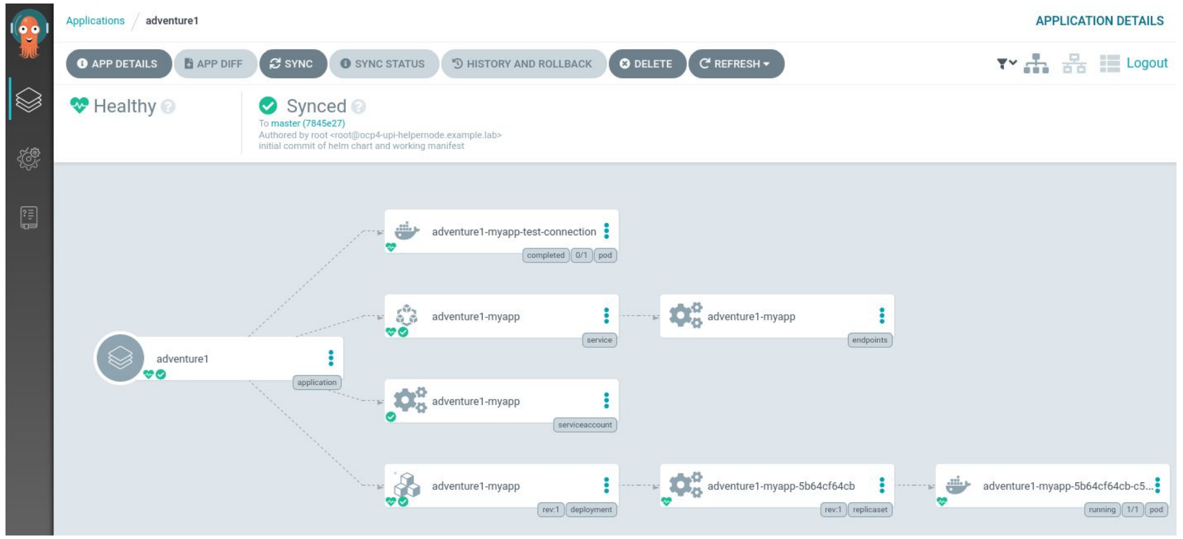
Existing resources are available in the community

- GitHub Actions
  - Chart Testing Action
  - Chart Releaser Action

GitOps

Management of Helm Charts and their releases declaratively

- Support with several popular Kubernetes GitOps tools
- Values file injection
- Setting individual parameters



To master (7845e27)
Authored by root <root@ocp4-upi-helpernode.example.lab>
initial commit of helm chart and working manifest

ArgoCD

Flux

Security

Helm has a robust security framework and has undergone a 3rd party security audit

Tiller Removal in Helm v3

Increases the overall security as cluster admin no longer required

Signed Binaries

Signed Helm CLI binaries including official RH released versions

Chart Provenance

Charts can be GPG signed and verified at install time

```
# Sign
$ helm package --sign <chart>

# Verify at install
$ helm install --verify <release_name> <chart>
```

Hosting Chart Repositories

Share your content with the world!

A chart repository is a web server that hosts an **index.yaml** metadata file and *optionally* a set of charts.

GitHub Pages

Hosting static content within repositories

Standalone Web Server

Provides dynamic capabilities for Kubernetes resources that are to be instantiated

Object Storage

Popular public cloud providers (such as AWS S3 and Google GCS)

Chart Museum

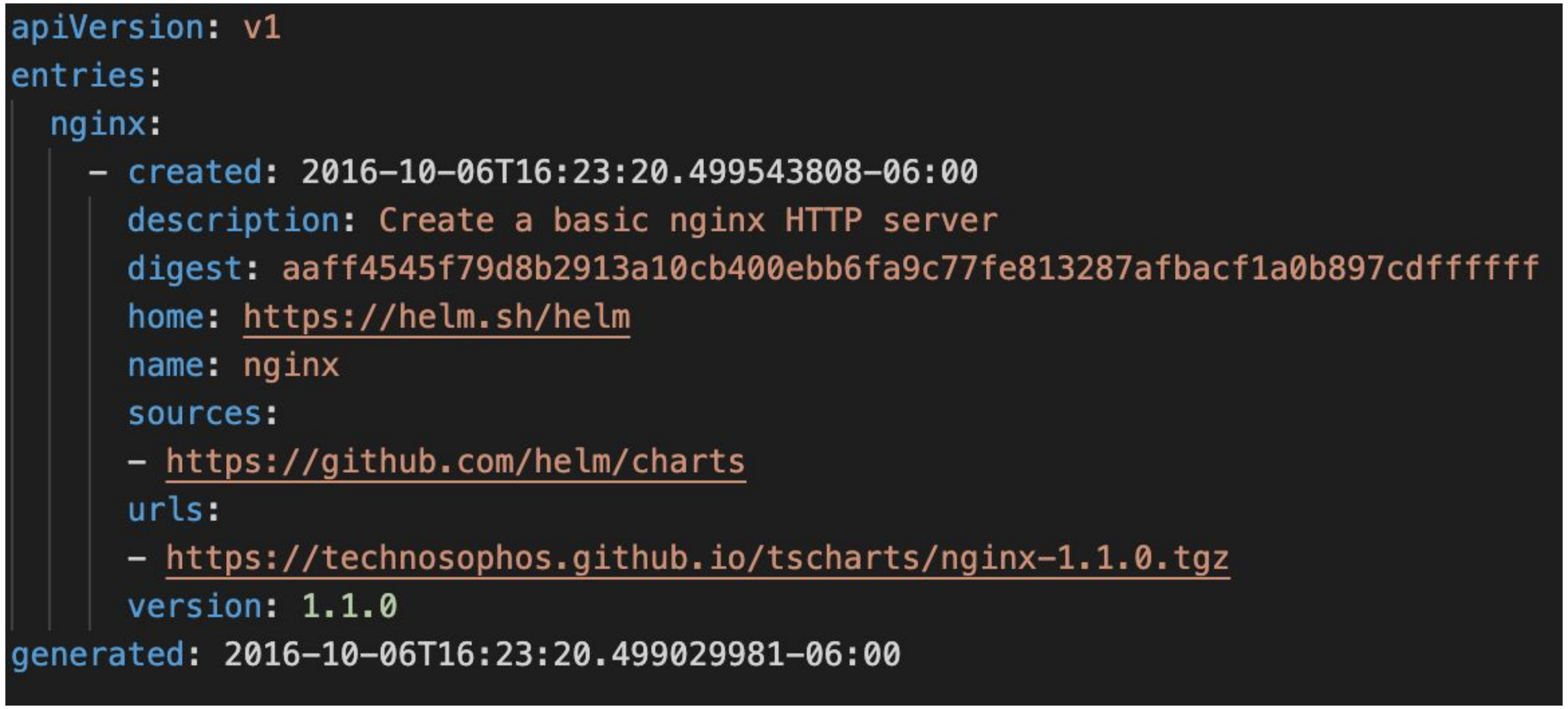
Open source Helm repository server

OCI Registry

Support for storing charts in OCI based registries (experimental)

index.yaml

Helm repository metadata file generated by `helm repo index` command



```yaml
apiVersion: v1
entries:
  nginx:
  - created: 2016-10-06T16:23:20.499543808-06:00
    description: Create a basic nginx HTTP server
    digest: aaff4545f79d8b2913a10cb400ebb6fa9c77fe813287afbacf1a0b897cdffffff
    home: https://helm.sh/helm
    name: nginx
    sources:
    - https://github.com/helm/charts
    urls:
    - https://technosophos.github.io/tscharts/nginx-1.1.0.tgz
    version: 1.1.0
generated: 2016-10-06T16:23:20.499029981-06:00
```

Finding Charts

Sharing and discovering Charts with the community

Helm Hub:

- Launched in 2018
- Provide a way to share Charts outside of the stable and incubator repositories

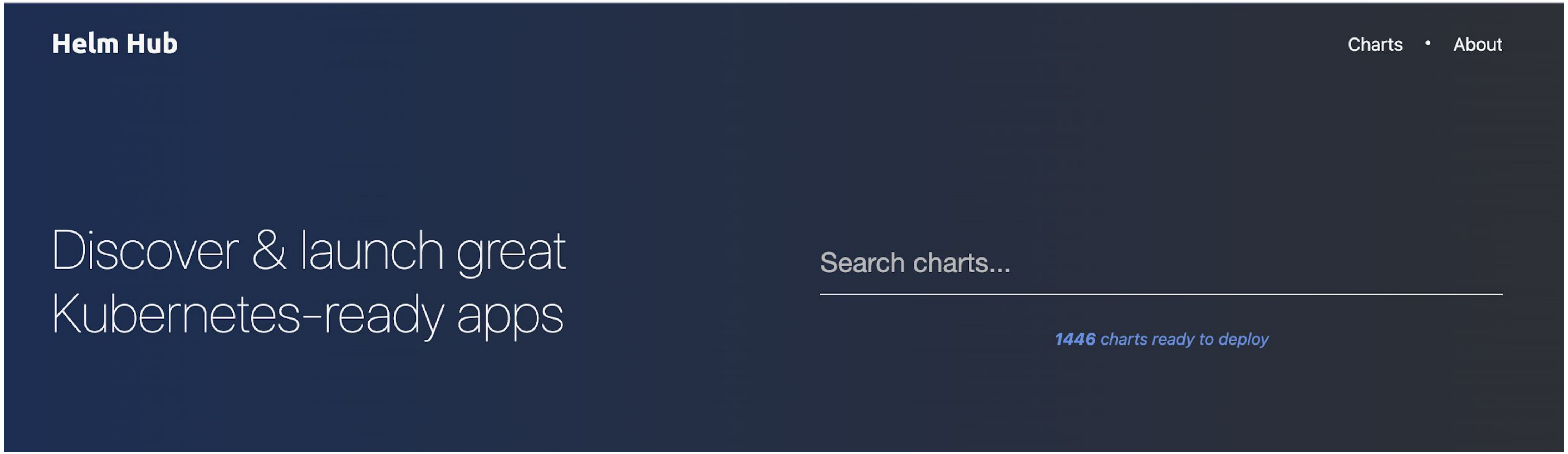
https://hub.helm.sh/

Artifact Hub:

- Launched in 2020
- Web based application for CNCF project
- Contains Helm Charts, OLM operators, OPA policies and Falco rules

https://artifacthub.io/

Helm Hub



Helm Hub

Charts • About

Discover & launch great Kubernetes-ready apps

Search charts...

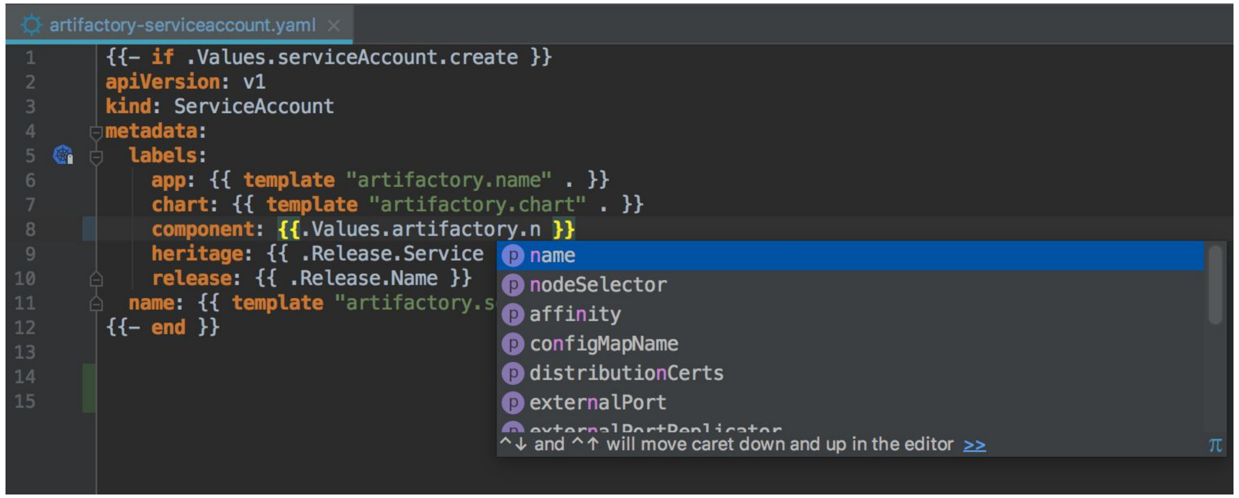
1446 charts ready to deploy

```
helm search hub
```

IDE Integration

Integration with several popular Integrated Development Environments

- Common features
  - Chart lifecycle
  - Preview template rendering
  - Dependency management
  - Visual editing



artifactory-serviceaccount.yaml

```
{{- if .Values.serviceAccount.create }}
apiVersion: v1
kind: ServiceAccount
metadata:
  labels:
    app: {{ template "artifactory.name" . }}
    chart: {{ template "artifactory.chart" . }}
    component: {{.Values.artifactory.n }}
    heritage: {{ .Release.Service
    release: {{ .Release.Name }}
  name: {{ template "artifactory.s
{{- end }}
```

```
name
nodeSelector
affinity
configMapName
distributionCerts
externalPort
externalPortReplicator
^↓ and ^↑ will move caret down and up in the editor >>
```

IntelliJ

VS Code

Helm Operators

Use the operator pattern to manage Helm charts

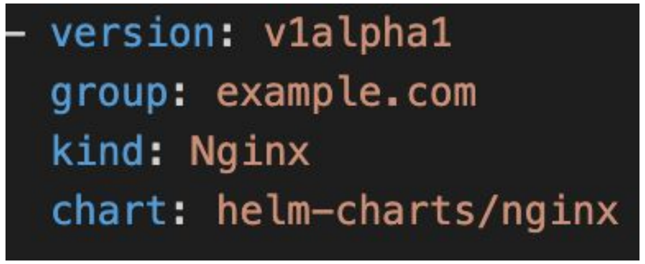
OPERATOR SDK

- operator-sdk supported feature
- Build new or existing helm charts
- Existing charts can be sourced from a remote url, repository, local directory or local archive
- Chart becomes a Custom Resource within the cluster
- Properties of Custom Resource `.spec` are injected as Chart values

```
# Create new Operator from scratch

$ operator-sdk new nginx-operator
--api-version=example.com/v1alpha1
--kind=Nginx --type=helm


# Create a new Operator from an existing
chart

$ operator-sdk new nginx-operator
--api-version=example.com/v1alpha1
--kind=Nginx --type=helm
--helm-repo=stable/nginx-ingress
```

Helm Operators

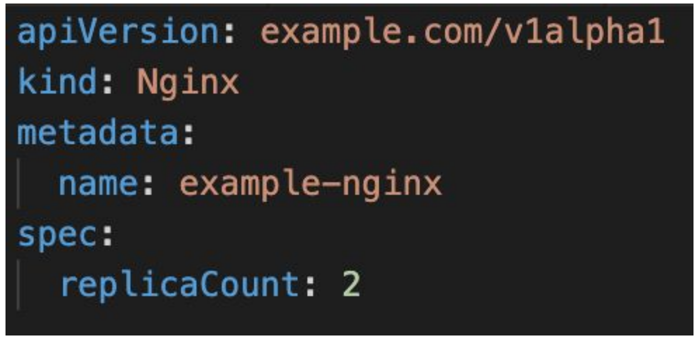
watches.yaml



```
- version: v1alpha1
  group: example.com
  kind: Nginx
  chart: helm-charts/nginx
```

→

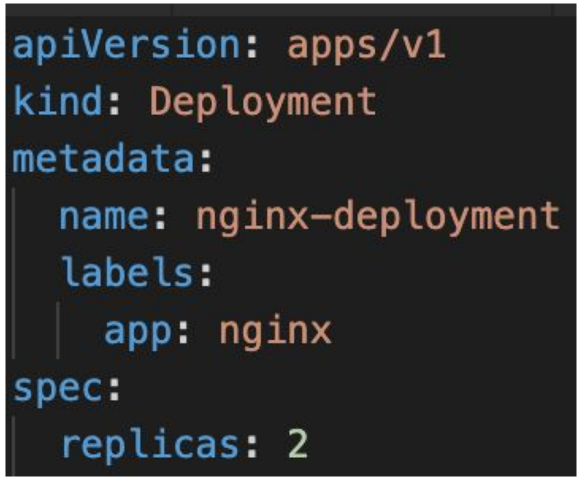
Nginx Custom Resource



```
apiVersion: example.com/v1alpha1
kind: Nginx
metadata:
  name: example-nginx
spec:
  replicaCount: 2
```

→

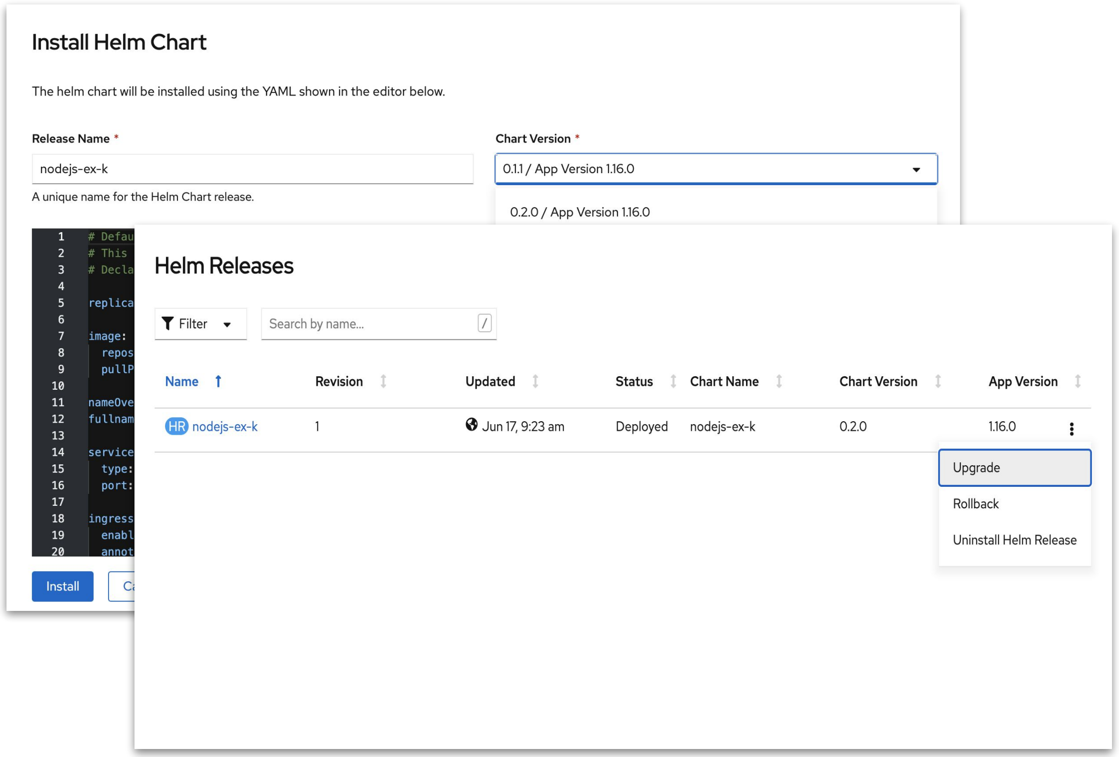
Rendered Deployment



```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 2
```

OpenShift Integration

- Official Helm binary available
- Chart documentation and metadata within web console
- Expose charts within a Helm repository
- Helm Release upgrade, uninstall and rollback



Install Helm Chart

The helm chart will be installed using the YAML shown in the editor below.

Release Name *: nodejs-ex-k

A unique name for the Helm Chart release.

Chart Version *: 0.1.1 / App Version 1.16.0

0.2.0 / App Version 1.16.0

```
# Defau
# This
# Decla

replica

image:
  repos
  pullP

nameOve
fullnam

service
  type:
  port:

ingress
  enabl
  annot
```

Helm Releases

Name	Revision	Updated	Status	Chart Name	Chart Version	App Version
HR nodejs-ex-k	1	Jun 17, 9:23 am	Deployed	nodejs-ex-k	0.2.0	1.16.0

Helm Resources

Extend your knowledge of the Helm ecosystem

Helm Documentation

https://helm.sh/docs/

Helm Project Repository

https://github.com/helm/helm

Slack

https://slack.kubernetes.io/ (#helm)

Interactive Lab

https://learn.openshift.com/developing-on-openshift/helm/

Learn Helm

https://www.packtpub.com/cloud-networking/learn-helm

Red Hat

OpenDevHour

https://red.ht/OpenDevHour

Upcoming events

- **Supersonic Secure Java with Quarkus,** SEP 14 | 16:00 CEST
- **Serverless stream processing of Debezium data change events with Kafka Streams and Knative,** OCT 20 | 16:00 CEST
- **Securing Microservices,** NOVEMBER
- **DevOps with Containers,** DECEMBER
- **Orchestrating microservices the cloud-native way,** JANUARY 2021

Past events

- **Helm for Developers,** AUGUST 18
- **Quarkus the black swan of Java,** JULY 23

Markus Eisele

Developer Adoption Lead

Red Hat

markus@redhat.com

Red Hat

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.

linkedin.com/company/red-hat

youtube.com/user/RedHatVideos

facebook.com/redhatinc

twitter.com/RedHat