Agility. Security. Delivered.

Hands-on Helm

Templating your way to a brighter future

About Coveros

- Coveros helps companies accelerate the delivery of secure, reliable software using agile methods

- Coveros Services
  - Agile transformations & coaching
  - Software development
  - Testing & Automation
  - DevOps and continuous integration
  - Software security assurance & testing

- Agile, DevOps, Test Automation, and Security training

- Open Source Products
  - SecureCI - Secure DevOps toolchain
  - Selenified - Agile Test Framework
  - Codeveros - Microservice Reference App

Select Clients

About Your Instructor

Bob Foster is a Technical Manager with over 20 years experience as a senior software engineer and team lead. Bob has proven experience in all aspects of software development and delivery (including DevOps, CI/CD, and infrastructure as code). He is a firm believer in the benefits gained by software teams when following Agile, CI/CD, and DevOps methodologies and has been successful in helping drive Agile/DevOps cultural shifts in both individual teams and organizations. Bob feels strongly that feedback is crucial to the success of a project and following the disciplines of Agile, CI/CD and DevOps will help provide visibility and feedback early and often throughout the entirety of a project, improving its chance for success.

Bob Foster
bob.foster@coveros.com

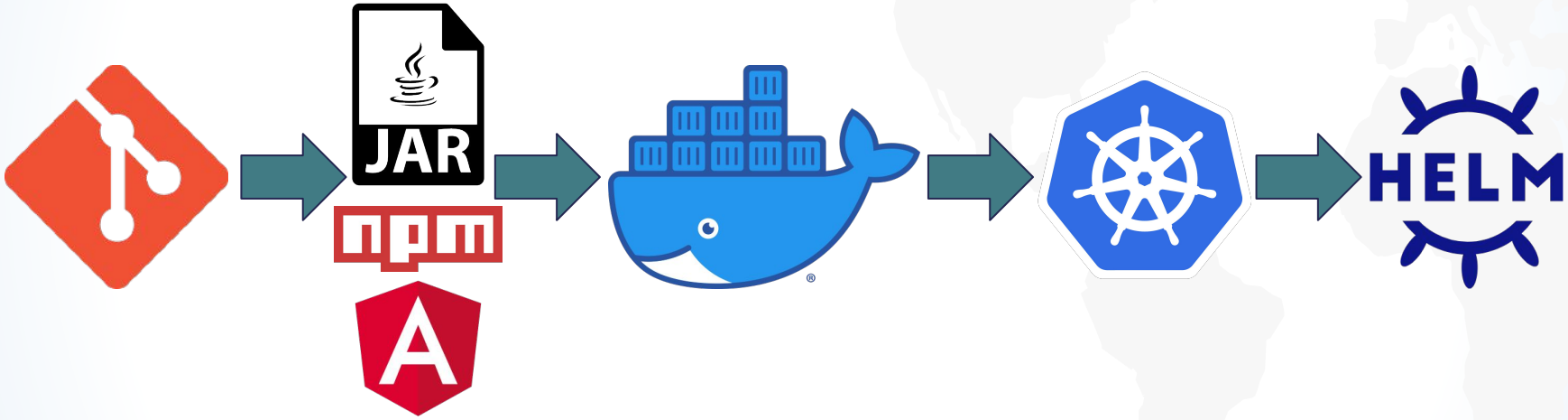
Learning Objectives

At the conclusion of the course, students will be able to:

- Understand the building blocks leading up to and including Helm
- Search, discover, download, and install existing Helm charts
- Upgrade, customize, rollback, and remove Helm releases
- Create and publish custom Helm charts using best practices, patterns, and recommendations

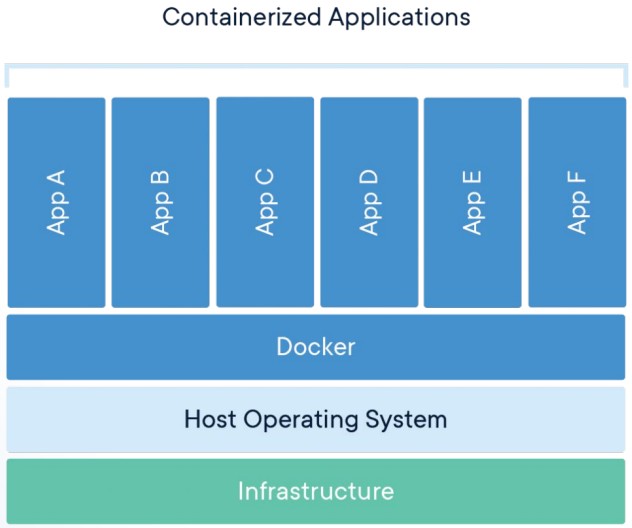
Building Blocks

Getting to Helm



Docker

- A Docker Image encapsulates everything needed to execute a process
- A Docker Container is a configured, running instance of a Docker image




```
# Dockerfile

FROM node:14.15-buster

# Create service directory
WORKDIR /usr/src/server

# Install dependencies
COPY package*.json
RUN npm ci --quiet --only=production

# Copy source code
COPY . .

EXPOSE 8080

# Start service
CMD ["npm", "start"]
```

Running a container

Create the Docker Image...

```
$ docker build -t user-service:1.0 .
```

...and then run a Docker Container

```
$ docker run -d -p 8080:8080 user-service:1.0
```

The local port 8080 is mapped to the running Docker container

```
$ curl localhost:8080
```

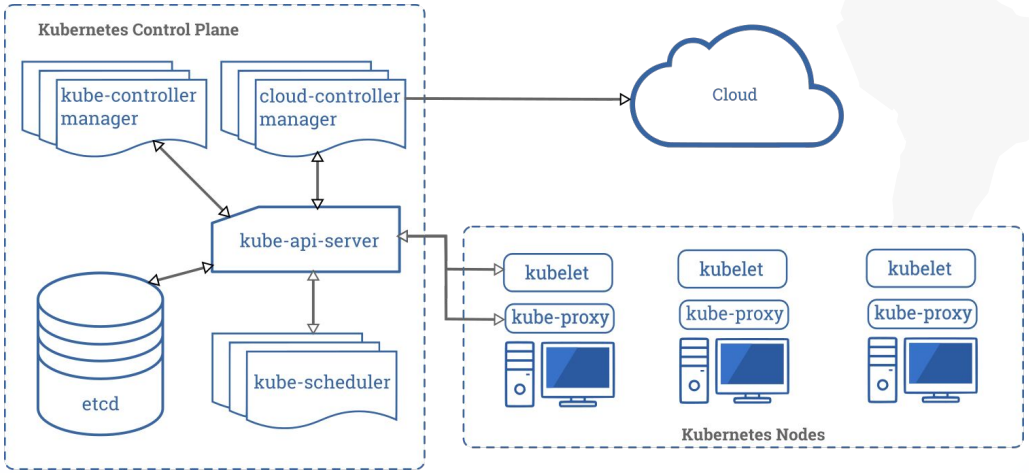
Docker Limitations

Managing Docker containers via the command line is brittle, error-prone, and difficult to organize into larger, more complex multi-service applications

Only sufficient for isolated, short-lived, processes

Kubernetes

- Orchestrates container deployment, scaling, and management
- Service Discovery, Self-healing, Extensible
- Objects describe desired state, Kubernetes works to ensure the actual state matches the specified desired state
- Manifest files used to define object specifications (yaml or json)



Pod

- Smallest, deployable Kubernetes object
- Encapsulates one or more running containers
- Represents actual state
- Instead of directly running Docker containers like in the previous example, containers are run inside Pods on the cluster
- Able to group multiple containers together into a single Pod in order to share resources
  - Sidecar containers
- Not typically configured directly
  - Desired state specified by a controller, handing responsibility of maintaining actual state to Kubernetes

Pod Specification

```
kind: Pod
apiVersion: v1
metadata:
  name: user-service-56df847669-qcjc8
  labels:
    app.kubernetes.io/name: user-service
spec:
  containers:
    - name: user-service
      image: 'coveros/codeveros-user-service:1.0'
      ports:
        - name: http
          containerPort: 8080
...
```

Deployment

- Common Kubernetes controller used to declare a desired state
- Kubernetes is constantly monitoring the actual cluster state, making the needed changes to ensure it matches the declared state
- A Deployment causes a ReplicaSet to be created, which in turn causes Pods to be deployed
- Other controllers include: DaemonSet, StatefulSet, and Job

Deployment Specification

```
apiVersion: Deployment
kind: Deployment
metadata:
  name: user-service
  labels:
    app.kubernetes.io/name: user-service
spec:
  replicas: 2
  selector:
    matchLabels:
      app.kubernetes.io/name: user-service
  template:
    metadata:
      labels:
        app.kubernetes.io/name: user-service
    spec:
      containers:
        - name: user-service
          image: coveros/user-service:1.0
          ports:
            - name: http
              containerPort: 8080
...
```

Service

- Facilitates service discovery and load balancing
- Pods can be stood up and torn down at any time, and deployed to any node in the cluster
  - Pods assigned unique IP addresses in the cluster network
  - Potentially short-lived, unreliable endpoints
- Services provide a known, stable, and reliable endpoint
  - Defined rules reflected in the node's kube-proxy
- kube-proxy is responsible for balancing requests between multiple replicas
- Possible to use Services to front more than just Pods

Service Specification

```
apiVersion: v1
kind: Service
metadata:
  name: user-service
spec:
  ports:
    - protocol: TCP
      port: 9080
      targetPort: 8080
      nodePort: 30001
  selector:
    app.kubernetes.io/name: user-service
  type: ClusterIP
```

Volume

- Running Containers are ephemeral, their state exists in memory
- Volumes allow data to be persisted across Container restarts
- Volumes enable data sharing between Containers within a Pod
- Volumes only exist for the lifetime of a Pod
- Volumes can be backed by a number of different sources
  - ConfigMap, emptyDir, persistentVolumeClaim etc.
  - Some types allow persisting data across Pod lifecycle changes
- PersistentVolumeClaims can be used to maintain data across Pod lifecycle changes and share data between Pods

ConfigMap

- Allows storing configuration data for other objects to use
- Stored as Key-Value pairs
- Container Volumes can be backed by a ConfigMap, used to define such things as a config file, environment variables and more

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: nginx-conf
data:
  default.conf: |
    server {
      listen 80;
      location /health-check {
        ...
      }
      ...
    }
```

```
apiVersion: v1
kind: Pod
metadata:
  name: codeveros-ui
spec:
  volumes:
    - name: nginx-config
      configMap:
        name: nginx-conf
  containers:
    - name: codeveros-ui
      volumeMounts:
       - name: nginx-config
         mountPath: /etc/nginx/conf.d/
```

Namespace

- Virtual clusters that organize and separate related Kubernetes resources
- Provide scoping
- default namespace used if not specified
- Resource names need to be unique within a namespace
- Certain objects only accessible within a Namespace
  - For example, Secrets are only accessible by objects in the same namespace
- Affects DNS configuration

Setting the Stage

But Wait, There's More!

There are many other Docker and Kubernetes details beyond the scope of this training. These high-level concepts will help focus the Helm discussion

Bring on Helm

What is Helm?

Helm is a CNCF graduated project that streamlines creating, installing, and managing Kubernetes applications

Provides

- Templating
- Packaging
- Versioning
- Distribution

Main Components

- Charts
- Repositories
- Releases

Templating

- Kubernetes resource manifests are defined as static YAML or JSON
- Manually maintaining manifests is doable only when deploying to a single environment or if there will be limited changes
  - Requires manually updating every affected object
- Templating allows you to organize the configurable attributes and optional features into a centralized location, requiring a single change that is propagated to all relevant resources
- Define boilerplate and shared configuration in a single location
- Alternatives options:
  - kustomize
  - kpt
  - cdk8s

Packaging

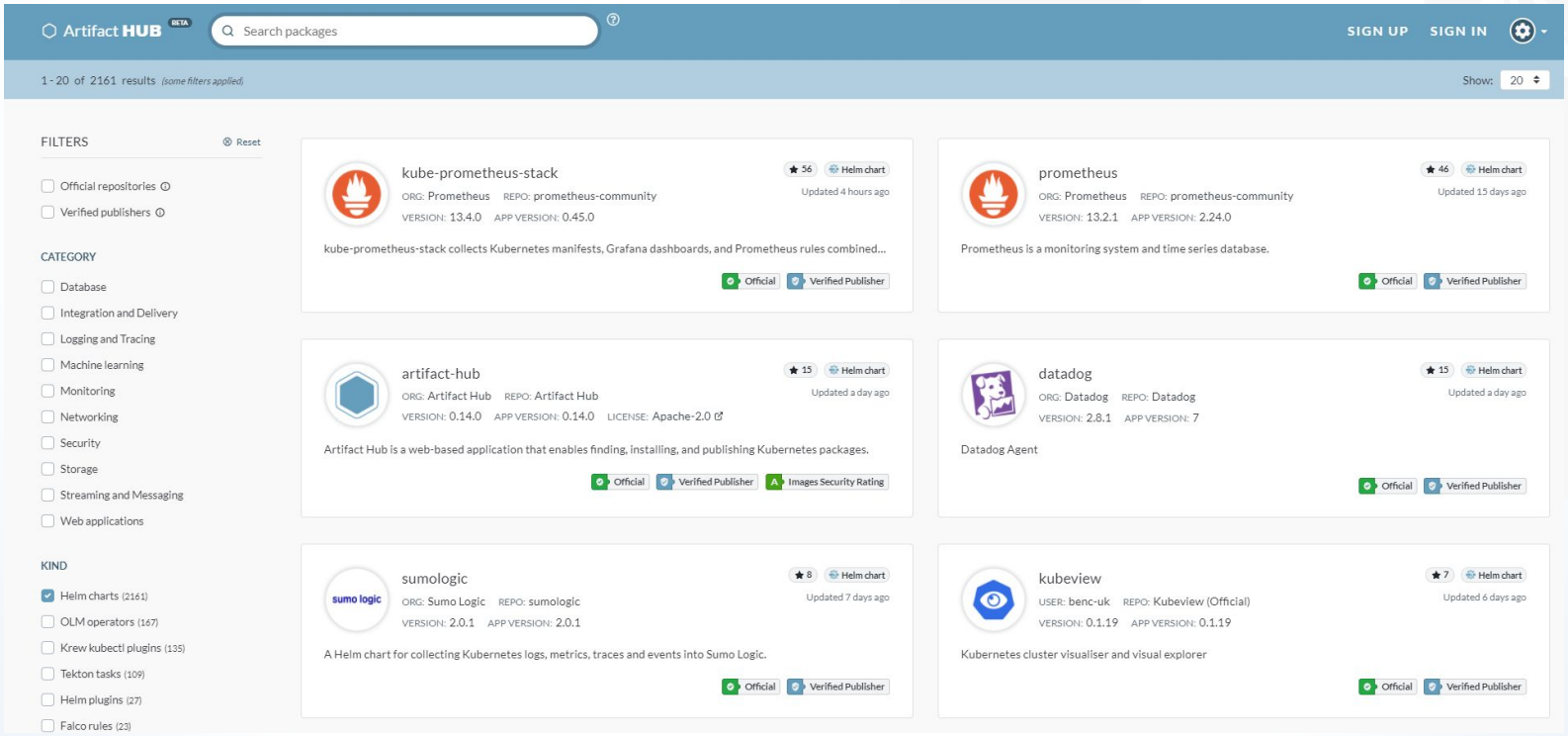
- Helm bundles together files and configuration into charts
- Charts can be packaged together as an archived (tgz) file
- Packaging charts enables easy sharing and distribution
- Kubernetes applications become an installable artifact just like a .msi, .dmg, .deb, .rpm etc.

Versioning

- Helm charts are semantically versioned
  - Major.Minor.Patch
- Allows for selecting a specific version to install
- Enables controlled upgrading and rolling back of installed charts

Distribution

- Makes charts available for others to use through chart repositories
- Artifact Hub is the CNCF-maintained public chart discovery tool
- Repositories include Harbor, Nexus, Chart Museum and any publicly accessible directory with an index.yaml



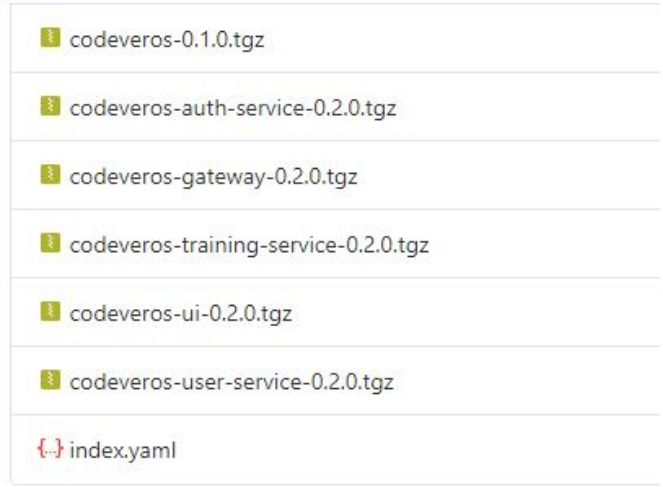
https://artifacthub.io

Chart Basics

- Main Helm artifact
- Fully defines a Kubernetes application
- Contains all the resources, templates, and configuration necessary to run an application in a Kubernetes cluster
- Versioned, packaged, and published to Helm repositories
- Installed into Kubernetes clusters as a release

Repository Basics

- Contains Helm charts available for download and installation
- HTTP server with index.yaml file
- Can contain packaged charts or the index.yaml file can point to external locations



https://coveros.github.io/codeveros

index.yaml 2.95 KB

```yaml
apiVersion: v1
entries:
  codeveros:
  - apiVersion: v2
    created: "2020-05-22T15:41:04.5051716-04:00"
    dependencies:
    - name: codeveros-user-service
      repository: https://coveros.gitlab.io/codeveros/charts
      version: 0.2.0
    - name: codeveros-training-service
      repository: https://coveros.gitlab.io/codeveros/charts
      version: 0.2.0
    - name: codeveros-auth-service
      repository: https://coveros.gitlab.io/codeveros/charts
      version: 0.2.0
    - name: codeveros-gateway
      repository: https://coveros.gitlab.io/codeveros/charts
      version: 0.2.0
    - name: codeveros-ui
      repository: https://coveros.gitlab.io/codeveros/charts
      version: 0.2.0
    description: An umbrella chart for CODEveros
    digest: 2588a6bba3cce56cde819ec1b5f539e353c639de1d7bfc94364cfdb90528444f
    name: codeveros
    type: application
    urls:
    - codeveros-0.1.0.tgz
    version: 0.1.0
```

Release Basics

- Running instance of a chart combined with configuration
- Similar relationship to a Helm chart as a Docker container to Docker image
- User-definable, unique name enables installing the same chart many times into the same cluster namespace

Helm Client

- helm list
- helm repo
- helm search
- helm install
- helm uninstall
- helm upgrade
- helm show
- helm status
- helm rollback
- helm pull
- helm create
- helm package
- helm dependency
- helm lint
- helm get
- ...and more

These will be discussed in detail throughout this tutorial. At anytime, can get additional information with the `--help` argument.

```
$ helm --help

$ helm list --help
```

Exercise: Environment Setup

Environment Overview

- You may either use a Coveros-provided Ubuntu-based AWS Instance or install the tools on your own machine or VM
  - The instructions will assume an Ubuntu 20 CLI-based setup
- **MicroK8s** is the one-node Kubernetes cluster we will use
- **Helm** is the main application and the focus of this course
- **Lens** and the **Kubernetes Dashboard** provide a clean user interface to visualize the cluster state and to observe changes
- **kubectl** is the command-line tool for interacting with the Kubernetes cluster
- **Codeveros** is the reference web application we will use throughout the training

Exercise Objectives

- Connecting to the EC2 Instance
- Installing MicroK8s, Helm, the Dashboard, and Lens
- Installing a simple Helm chart to verify setup

Connecting to VM

- You may either use a course-provided AWS EC2 instance, or complete the exercises using your own VM or machine
- All instructions will assume you are using the instructor-provided Ubuntu EC2 instance
- The instructor should have issued each student an EC2 IP address
- The EC2 instances are configured with a super user account with the following credentials:
  - Username: helmuser, password: helmpassword
  - You will be required to update your password upon initial connection

Connecting to VM

1. SSH into your EC2 Instance

```
$ ssh helmuser@<EC2_IP>
```

2. Enter the initial password -- helmpassword -- at the prompt

```
User username "helmuser".
helmuser@<EC2_IP>'s password: < helmpassword >
```

3. Update your password, by first entering the current password, then your new password, and then retyping your new password

```
WARNING! Your Password has expired.
You must change your password now and login again!
Changing password for helmuser.
Current password: < helmpassword >
New password:
Retype new password:
```

4. SSH into your EC2 instance using your updated password

MicroK8s

- MicroK8s is a simple one-node Kubernetes cluster installation
- Maintained by Canonical
- Fully-conformant, tracks upstream Kubernetes releases
- Originally built for Ubuntu-based distributions, also supports Windows, MacOS and other Linux distributions that support snapd
- Current version is 1.20
- https://microk8s.io/

Install MicroK8s

5. Install MicroK8s 1.20

```
$ sudo snap install microk8s --classic --channel=1.20/stable
```

6. Add microk8s group to your user account

```
$ sudo usermod -aG microk8s $USER
$ sudo chown -f -R $USER ~/.kube
```

7. Exit and reconnect to instance in order to apply group change

Configure MicroK8s

8. Ensure all MicroK8s services are running

```
$ microk8s inspect
```

```
helmuser@ip-172-31-1-140:~$ microk8s inspect
Inspecting Certificates
Inspecting services
  Service snap.microk8s.daemon-cluster-agent is running
  Service snap.microk8s.daemon-containerd is running
  Service snap.microk8s.daemon-apiserver is running
  Service snap.microk8s.daemon-apiserver-kicker is running
  Service snap.microk8s.daemon-control-plane-kicker is running
  Service snap.microk8s.daemon-proxy is running
  Service snap.microk8s.daemon-kubelet is running
  Service snap.microk8s.daemon-scheduler is running
  Service snap.microk8s.daemon-controller-manager is running
```

Reinspect until all are running!

```
Inspecting services
  Service snap.microk8s.daemon-cluster-agent is running
FAIL:  Service snap.microk8s.daemon-containerd is not running
```

9. Enable Add-ons

```
$ microk8s enable dns ingress storage
```

10. Alias kubectl and copy the kubeconfig to your user directory

```
$ alias kubectl='microk8s kubectl'
$ echo "alias kubectl='microk8s kubectl'" >> ~/.bash_aliases
$ microk8s config > ~/.kube/config
$ chmod 600 ~/.kube/config
```

Update MicroK8s Certs

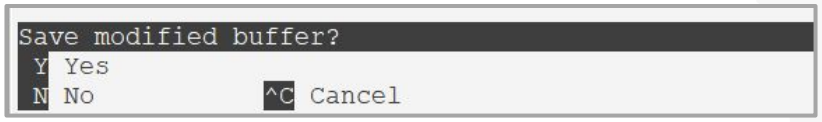
11. Add Local IP address to alt_names

```
$ nano /var/snap/microk8s/current/certs/csr.conf.template
```

```
[ alt_names ]
DNS.1 = kubernetes
DNS.2 = kubernetes.default
DNS.3 = kubernetes.default.svc
DNS.4 = kubernetes.default.svc.cluster
DNS.5 = kubernetes.default.svc.cluster.local
IP.1 = 127.0.0.1
IP.2 = 10.152.183.1
IP.4 = 3.131.83.73
#MOREIPS
```

← **Add the IP address to IP.4**
(Your IP address will be different)

12. To save and exit nano, press Ctrl+X, then press Y, then press Enter



13. Refresh the Certs

```
$ sudo microk8s refresh-certs
```

Install Helm

14. Download the Helm 3 installer script

```
$ curl -fsSL -o get_helm.sh \
https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3
```

15. Run the Helm installer script

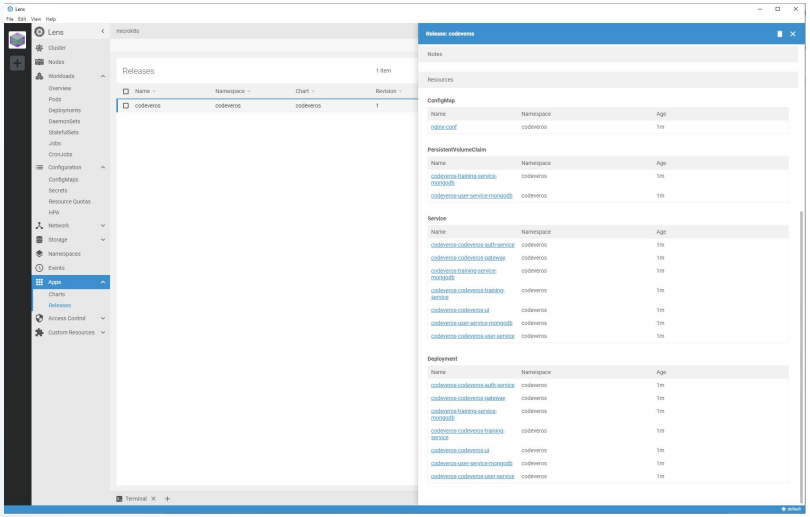
```
$ chmod 700 get_helm.sh && ./get_helm.sh && rm -rf get_helm.sh
```

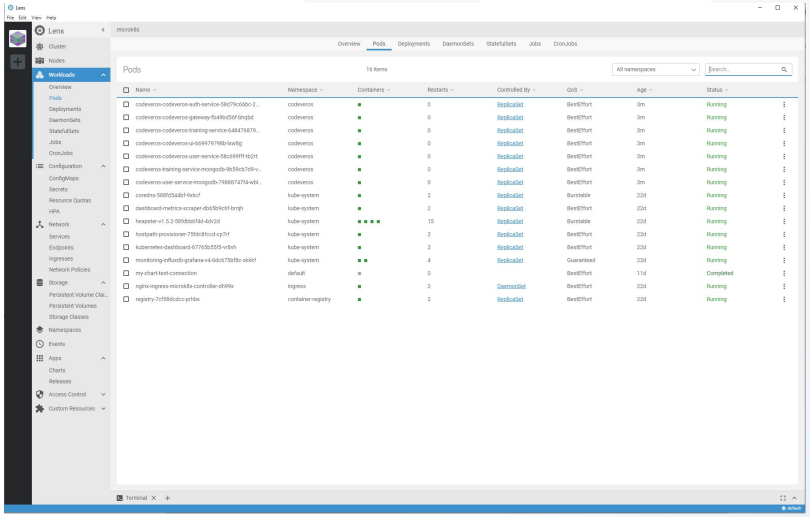
16. Enable kubectl and helm bash completion

```
$ sudo apt install -y bash-completion
$ source <(kubectl completion bash)
$ source <(helm completion bash)
$ echo "source <(kubectl completion bash)" >> ~/.bashrc
$ echo "source <(helm completion bash)" >> ~/.bashrc
```

Lens

- Lens is an open-source, separately installed Kubernetes IDE, allowing you to interact with one or more Kubernetes clusters
- Provides increased visibility, real time statistics, log streams, Helm application support, and hands-on troubleshooting capabilities
- https://k8slens.dev





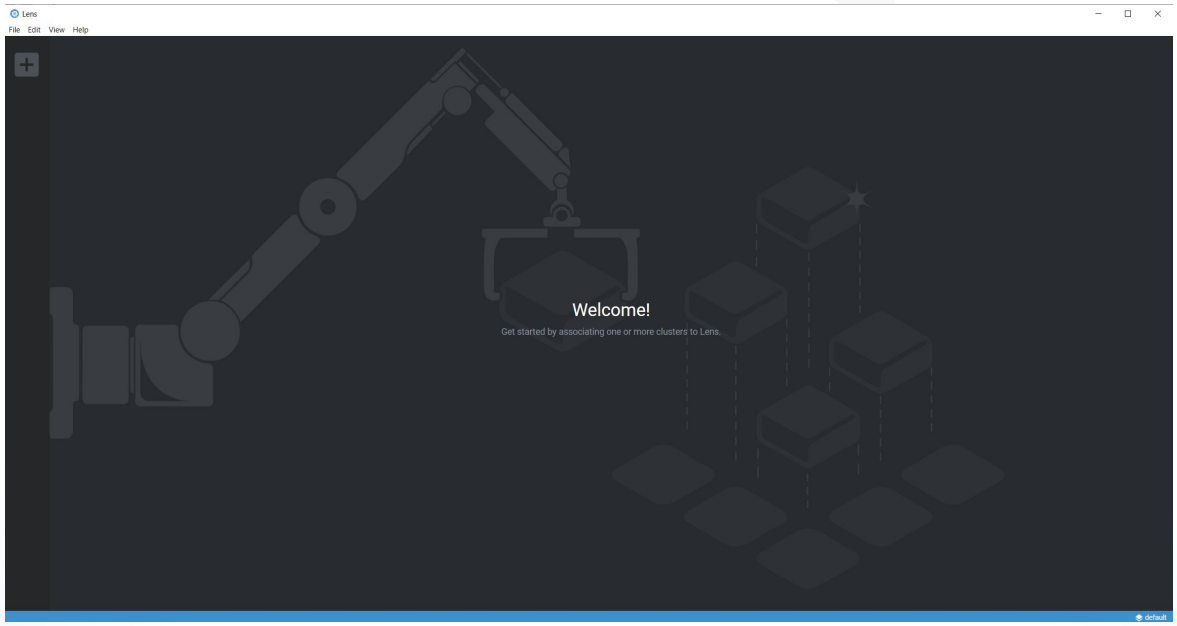
Download Lens Installer

The following steps are to be performed on your local machine, not the AWS instance. Lens is a GUI application that can be used to connect to remote clusters:

17. Go to https://k8slens.dev/ to download the version based on your local operating system
    - Also available as a snap: `sudo snap install kontena-lens --classic`

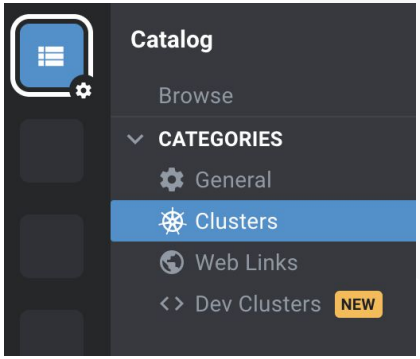
Install Lens

18. Execute the downloaded binary to have it run through the installation process
    - Lens will either automatically launch after installation or will be ready to be manually started
    - You will be asked to create a Lens ID for Login, follow the instructions to create the Lens ID

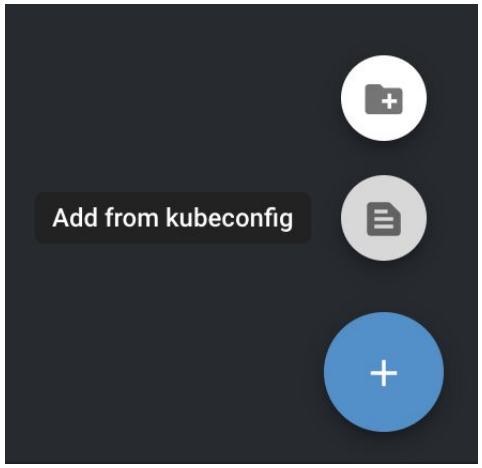


Add Cluster to Lens

19. Click the “Cluster” menu option on the left-hand menu



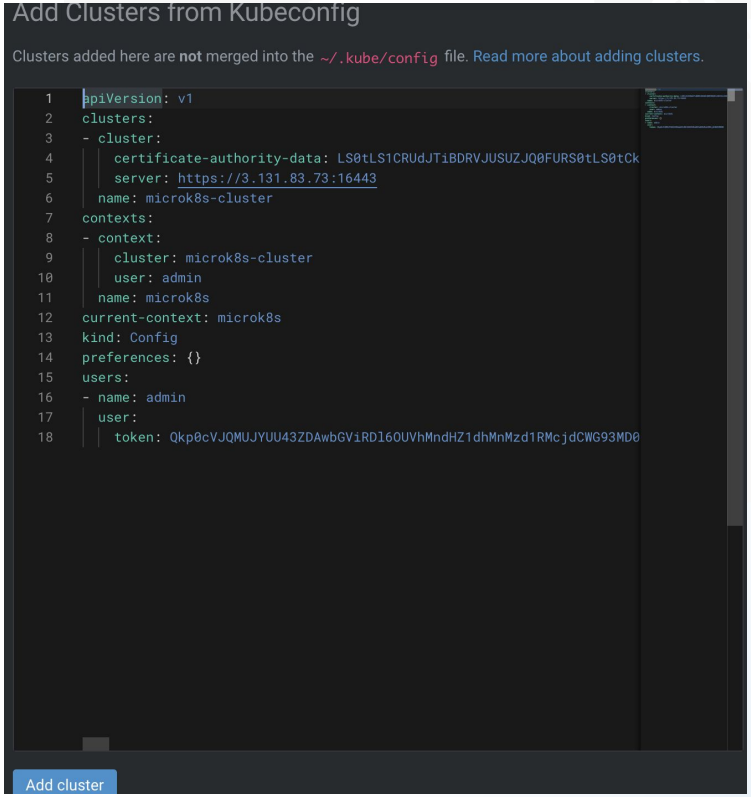
20. Hover over the “Add” icon and Select "Add from kubeconfig”



Add Cluster to Lens

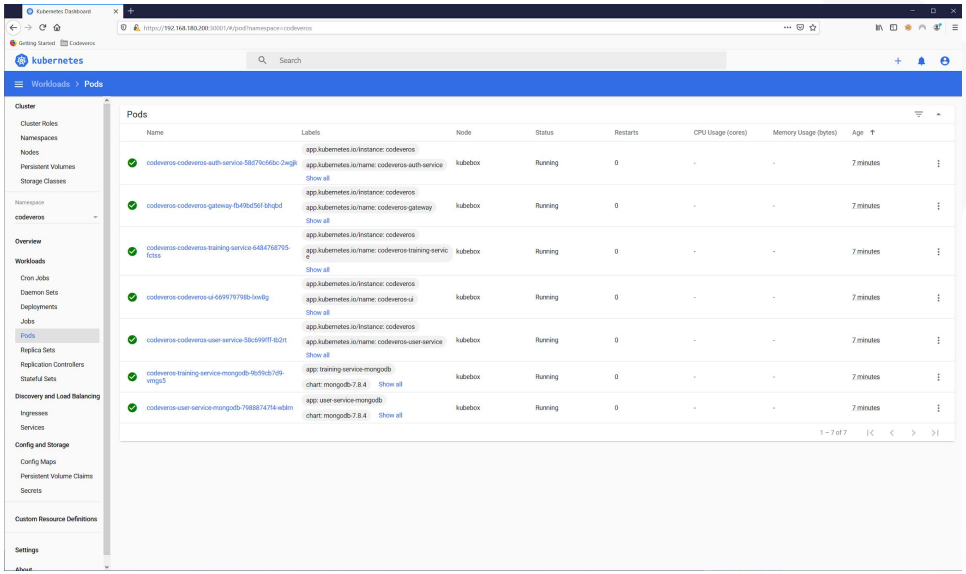
21. Copy the contents of your ~/.kube/config file on the AWS EC2 instance, paste it into Lens, and click "Add cluster".

```
helmuser@ip-172-31-1-140:~$ cat ~/.kube/config
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: LS0tLS1CRUdJTiBDRVJ
NRFV4T0RJek1UZGFGdzB6TVRBeU1ETXhPREl6TVRkYQpNQmN4Rl
lQUDlLYW1jZmIwMk8KWjYzeGZjMjZMRmh0Y05PVWFnSnZhbC9NZ
2JxV0p0WnVOSWJraUJnK21QT1ZWZjNDUEhWS2JNdk4xNWpPbGM1
N1p1eGNYSk9mMENBd0VBQWFOUU1FNHdIUVlEVlIwT0JCWUVGTzE
nZ0VCQURWUXZ3T3Z5WVpOTkZLdAozeVFucFlQMWcyQkI0dndBbz
IKenNsWFNETXliNHk2MjZqN1NMME9ZV0VrK096ci9zVGNSK3hyF
XZqNklRQmFIdVFKTVNTOUg0VzVrcW83b2hZZWVIR1BWZkZaZGFQ
    server: https://172.31.1.140:16443
  name: microk8s-cluster
contexts:
- context:
    cluster: microk8s-cluster
    user: admin
  name: microk8s
current-context: microk8s
kind: Config
preferences: {}
users:
- name: admin
  user:
    token: YU5kcHgvdFVYSnhubDBqZTA5SXVRSXFtdGNHbWVp
```



Kubernetes Dashboard

- The Dashboard is a simple web-based Kubernetes user interface
- Provided by Kubernetes as a set of manifests deployable into a cluster providing insight into that cluster
- Available as a MicroK8s add-on for easy installation
  - Does not require installing a separate desktop tool



Kubernetes Dashboard

22. Enable Dashboard add-on

```
$ microk8s enable dashboard
```

23. Create Patch File

```
$ cat <<EOF > patch.yaml
spec:
  type: NodePort
  ports:
    - port: 443
      nodePort: 30001
EOF
```

24. Patch the Dashboard Service to make the dashboard accessible on port 30001

```
$ kubectl -n kube-system patch svc kubernetes-dashboard -p "$(cat
patch.yaml)"
```

Kubernetes Dashboard

25. Retrieve token from secret

```
$ token=$(kubectl -n kube-system get secret | grep default-token | cut -d " " -f1)

$ kubectl -n kube-system describe secret $token
```

```
Name:         default-token-j6mmb
Namespace:    kube-system
Labels:       <none>
Annotations:  kubernetes.io/service-account.name: default
              kubernetes.io/service-account.uid: c678f14a-0ae1-40a6-8302-73c8dcb761ff

Type:  kubernetes.io/service-account-token

Data
====
ca.crt:     1103 bytes
namespace:  11 bytes
token:      eyJhbGciOiJSUzI1NiIsImtpZCI6IkdFamhwZUpBREdSNjNIWk8wVXNYQXFpbm9yeTVhUDJYUU9mSWpuNHVRU0EifQ.eyJpc3
Wllc3BhY2UiOiJrdWJlLXN5c3RlbSIsImtlYmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VjcmV0Lm5hbWUiOiJkZWZhdWx0LXRva2VuLW
mFlbHQiLCJrdWJlcm5ldGVzLmlvL3N1cnZpY2VhY2NvdW50L3N1cnZpY2UtYWNjb3VudC5laWQiOiJjNjc4ZjE0YS0wYWUxLTQwYTYtODMwMi
CJ9.K3DYQ649Ngkho2zPh5yYQdQ_SIohO30HzvKdVpn-2v3SdgtjmWZLJ29qeRh_NJGr--OgEhvskocPEXXAv1LlxdfkW0T0LuFiEMMdS9pFT
```

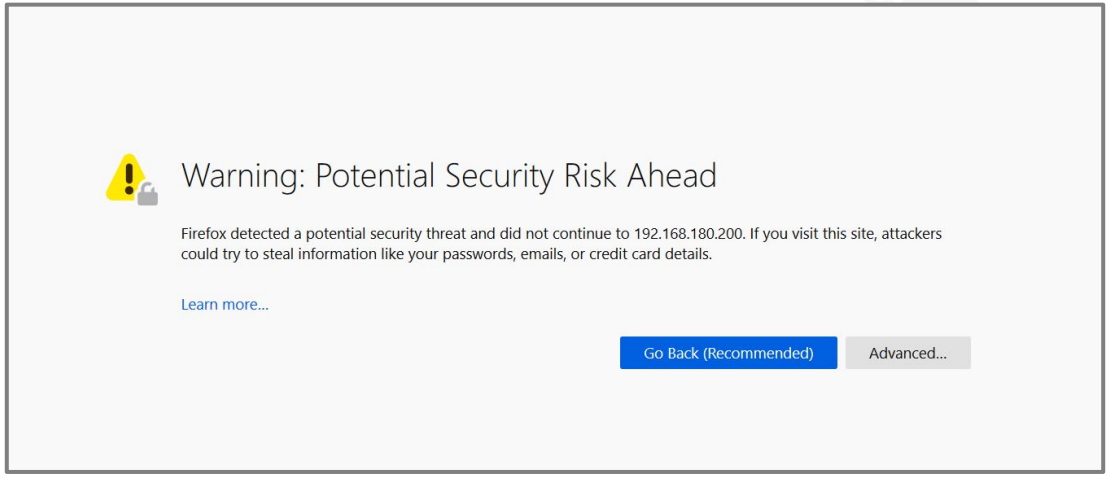
26. Save the token value into a file for easy retrieval

```
$ echo "<TOKEN_HERE>" >> ~/dashboard-token.out
```

Kubernetes Dashboard

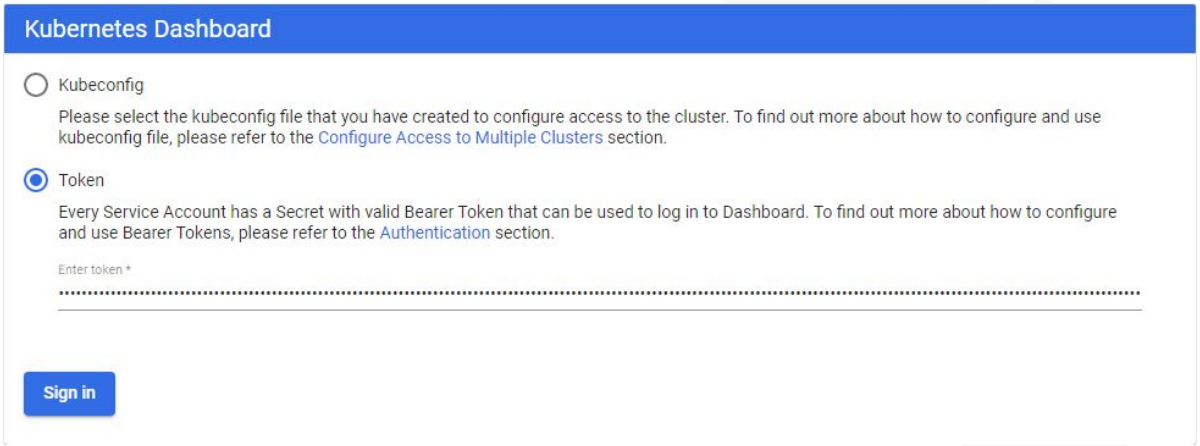
27. In your browser, access https://<EC2 IP>:30001, using either the EC2 instance IP address, or whatever the IP is that is running MicroK8s

There will be a security warning that you will need to accept. This is just a warning about the untrusted certificate that is initially loaded



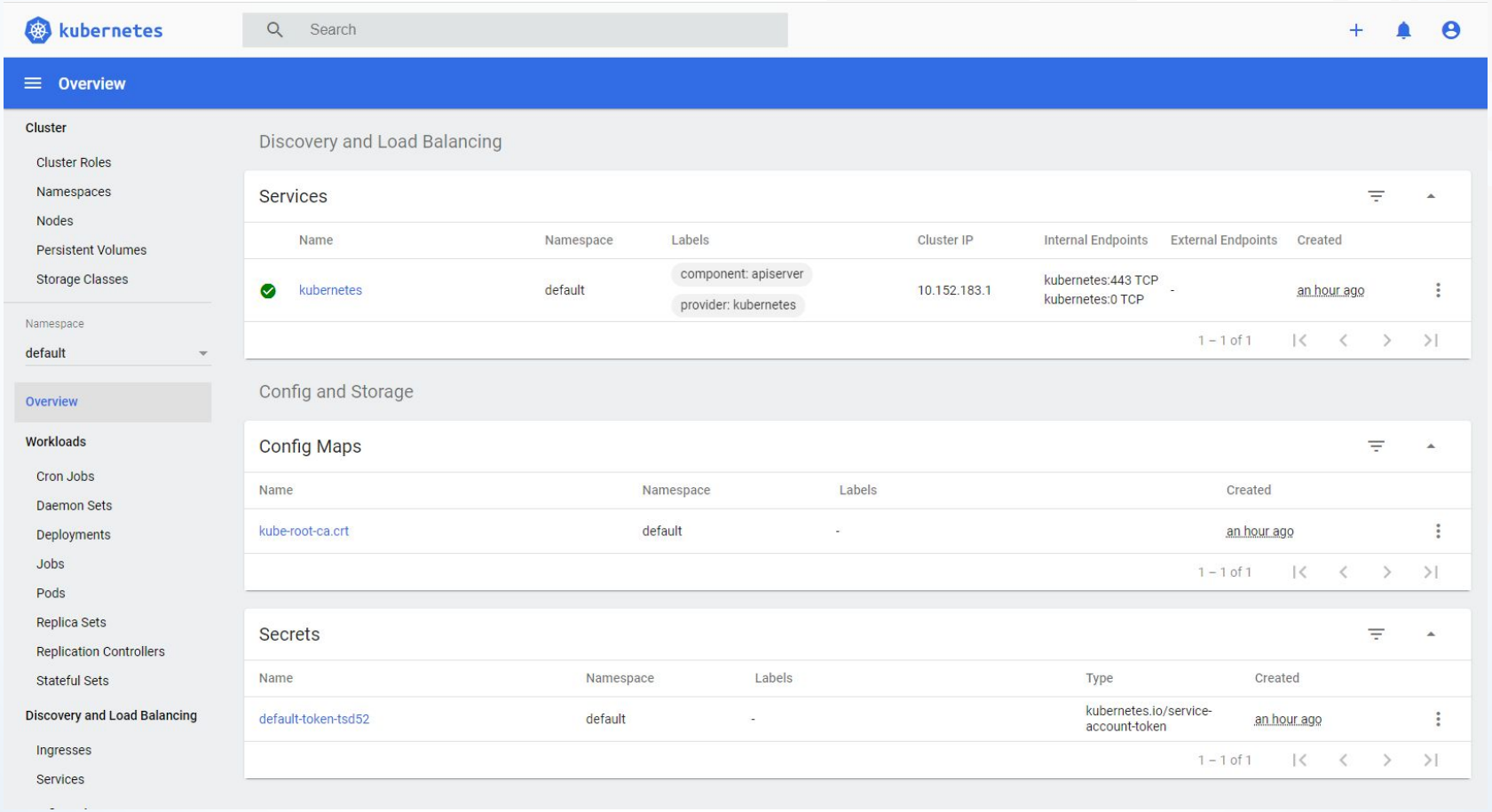
Kubernetes Dashboard

28. After accepting the security warning, a login prompt requires the previously copied token



Kubernetes Dashboard

You should now be authenticated and able to view the cluster data



Verify Environment Setup

29. Add the Coveros Helm Repository

```
$ helm repo add coveros https://coveros.github.io/charts
```

30. Install the Hello World example chart

```
$ helm install example-chart coveros/example-chart
```

31. Verify that the chart successfully deployed

```
$ helm ls
```

32. Use Lens or the Kubernetes Dashboard to observe the added resources

33. Clean-up the installed chart release

```
$ helm uninstall example-chart
```

Installing a Chart

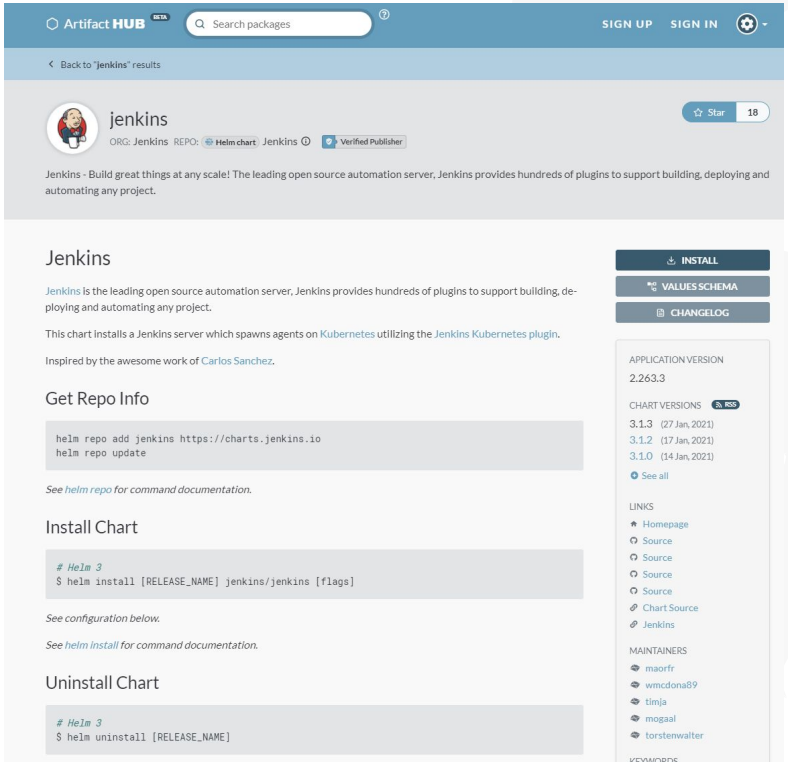
helm install

Chart Installation

- An installed chart is a release
- Available charts can be discovered and installed from repositories
- Artifact Hub is the CNCF-maintained chart discovery service
- Helm commands
  - repo
  - search
  - pull
  - install
  - list/ls
  - status

Artifact Hub

- CNCF-maintained web application
- Aggregates many Helm repositories (doesn't host any charts)
- Provides chart search and discovery user interface



https://artifacthub.io/packages/helm/jenkinsci/jenkins

Helm Client - helm repo

- Command used to add, remove, and list chart repositories
- Also used to generate an index.yaml file for creating a repository
- At a minimum, a saved chart repository has a name and URL
- Allows installing charts using the saved repository name

```
$ helm repo add jenkins https://charts.jenkins.io

$ helm install jenkins jenkins/jenkins
```

Helm Client - helm search

- Search for charts in Artifact Hub and in saved repositories
- `helm search hub` searches Artifact Hub or another configured hub instance
- `helm search repo` searches the saved repositories
  - `helm repo update` should be used to update local index

```
$ helm search hub jenkins
URL                          CHART VERSION APP VERSION  DESCRIPTION
https://artifacthub.io/... 3.1.8         2.263.3      Jenkins - Build…
...

$ helm search repo jenkins
NAME                         CHART VERSION APP VERSION  DESCRIPTION
jenkins/jenkins              3.1.8         2.263.3      Jenkins - Build…
```

Helm Client - helm pull

- Downloads a Helm chart archive without installing it
- Useful for retrieving a chart for inspection, modification or inclusion as a dependency for another chart
- Manually downloading dependencies is not recommended for production use-cases

```
$ helm pull jenkins/jenkins

$ ls
jenkins-3.1.8.tgz

$ helm pull jenkins/jenkins --untar

$ ls jenkins/
CHANGELOG.md Chart.yaml README.md templates tests values.yaml
```

Helm Client - helm install

- Installs a helm chart, creating a new release
- Takes a release name, any value overrides and the chart as a name reference, URL, or path to an archive file or directory
- Many other options can be set including the Kubernetes namespace in which to install the chart

```
$ helm install jenkins jenkins/jenkins

$ helm ls

NAME     NAMESPACE  REVISION UPDATED    STATUS   CHART
jenkins  default    1        2021-02-04 deployed jenkins-3.1.8
```

Helm Client - helm list/ls

- Displays the installed releases, identifying upgrade candidates
- Defaults to showing the charts in the Kubernetes default namespace
- Can change the namespace queried as well as displaying all installed releases
  - `-n jenkins-ns`
  - `--all-namespaces`

```
$ helm install -n jenkins-ns jenkins jenkins/jenkins

$ helm ls
NAME     NAMESPACE  REVISION UPDATED    STATUS   CHART

$ helm ls --all-namespaces

NAME     NAMESPACE  REVISION UPDATED    STATUS   CHART
jenkins  jenkins-ns 1        2021-02-05 deployed jenkins-3.1.8
```

Helm Client - helm status

- Displays similar information as **`helm ls`**, but about a single release
- As with **`helm ls`**, the namespace needs to be specified if the release is not installed in the default namespace
  - `-n jenkins-ns`
  - `--all-namespaces`

```
$ helm install -n jenkins-ns jenkins jenkins/jenkins

$ helm status -n jenkins-ns jenkins
NAME: jenkins
LAST DEPLOYED: Fri Feb 5 16:08:18 2021
NAMESPACE: jenkins-ns
STATUS: deployed
REVISION: 1
NOTES:
...
```

Installing a Release

```
$ helm repo add codeveros https://coveros.github.io/codeveros

"codeveros" has been added to your repositories

$ kubectl create namespace codeveros-ns

namespace/codeveros-ns created

$ helm install -n codeveros-ns codeveros-rel codeveros/codeveros

NAME: codeveros-release
LAST DEPLOYED: Fri Feb 5 14:43:23 2021
NAMESPACE: codeveros-ns
STATUS: deployed
REVISION: 1

$ helm ls -n codeveros-ns

NAME          NAMESPACE     REVISION UPDATED    STATUS   CHART
codeveros-rel codeveros-ns  1        2021-02-05 deployed codeveros-0.2.0
```

Exercise: Installing a Helm Chart

Exercise Objectives

- Adding a Helm repository
- Searching for a Helm chart
- Install Codeveros chart from Coveros repository
- Verifying the installation on the command line and using Lens or Dashboard
- Navigating to Codeveros to show it has been successfully deployed

Add Repository

1. Add Codeveros Repository

```
$ helm repo add codeveros https://coveros.github.io/codeveros
```

2. List Repositories

```
$ helm repo list
```

3. Update Local Repository Cache

```
$ helm repo update
```

```
helmuser@ip-172-31-1-140:~$ helm repo add codeveros https://coveros.github.io/codeveros
"codeveros" has been added to your repositories
helmuser@ip-172-31-1-140:~$ helm repo list
NAME            URL
coveros         https://coveros.github.io/charts
codeveros       https://coveros.github.io/codeveros
helmuser@ip-172-31-1-140:~$ helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "coveros" chart repository
...Successfully got an update from the "codeveros" chart repository
Update Complete. □Happy Helming!□
```

Searching for a Chart

4. Search for Codeveros Chart in Artifact Hub

```
$ helm search hub codeveros
```

5. Search for Codeveros Chart in Saved Repositories

```
$ helm search repo codeveros
```

```
helmuser@ip-172-31-1-140:~$ helm search hub codeveros
No results found
helmuser@ip-172-31-1-140:~$ helm search repo codeveros
NAME                                    CHART VERSION   APP VERSION     DESCRIPTION
codeveros/codeveros                     0.2.0           1.0.0           An umbrella chart for CODEveros
codeveros/codeveros-auth-service        0.5.0           1.1             Codeveros Auth Service Helm chart
codeveros/codeveros-gateway             0.4.0           1.1             Codeveros Gateway Helm chart
codeveros/codeveros-training-service    0.5.0           1.1             Codeveros Training Service Helm chart
codeveros/codeveros-ui                  0.6.0           1.1             Codeveros Angular UI Helm chart
codeveros/codeveros-user-service        0.5.0           1.1             Codeveros User Service Helm chart
```

What is the difference?

Installing a Chart

8. Install Codeveros Chart

```
$ helm install codeveros codeveros/codeveros
```

```
helmuser@ip-172-31-1-140:~$ helm install codeveros codeveros/codeveros
NAME: codeveros
LAST DEPLOYED: Fri Feb  5 19:03:27 2021
NAMESPACE: default
STATUS: deployed
REVISION: 1
```

Verify installation using CLI

6. Use helm to list installed releases

```
$ helm ls
```

7. Use helm to get the status of the specific release

```
$ helm status codeveros
```

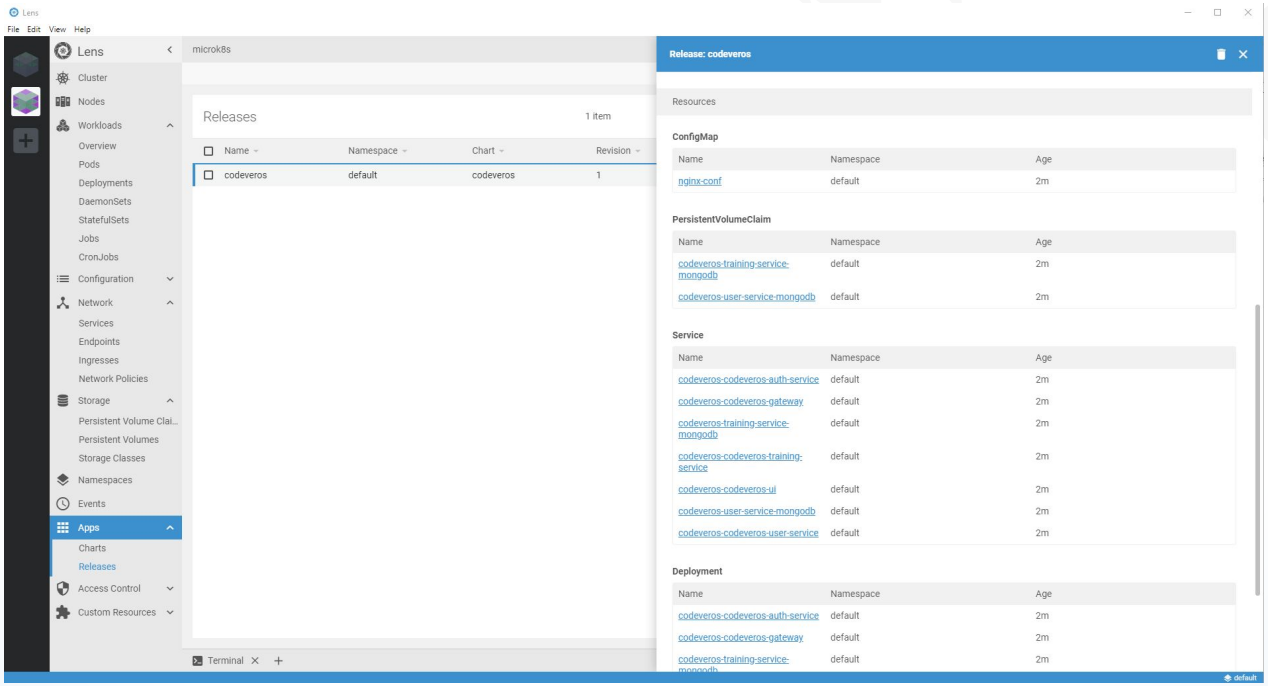
8. Use kubectl to view installed resources

```
$ kubectl get pods
```

```
helmuser@ip-172-31-1-140:~$ helm ls
NAME            NAMESPACE       REVISION        UPDATED                                 STATUS          CHART           APP VERSION
codeveros       default         1               2021-02-05 19:03:27.204442174 +0000 UTC deployed        codeveros-0.2.0 1.0.0
helmuser@ip-172-31-1-140:~$ helm status codeveros
NAME: codeveros
LAST DEPLOYED: Fri Feb  5 19:03:27 2021
NAMESPACE: default
STATUS: deployed
REVISION: 1
helmuser@ip-172-31-1-140:~$ kubectl get pods
NAME                                            READY   STATUS    RESTARTS   AGE
codeveros-codeveros-ui-794c766d88-k64cl         1/1     Running   0          74s
codeveros-user-service-mongodb-584bcf745b-nzsn7 1/1     Running   0          74s
codeveros-codeveros-user-service-65fcf5dc75-gk82q       1/1     Running   0          74s
codeveros-codeveros-training-service-6cd449fbd9-zblh8   0/1     Running   0          74s
codeveros-codeveros-gateway-757f956b7c-jncff    1/1     Running   0          74s
codeveros-7cfd67c477-xmkwb                      1/1     Running   0          74s
codeveros-training-service-mongodb-7bc475d7fc-9zgp7     0/1     Running   0          74s
codeveros-codeveros-auth-service-67d7654f6f-m5l6d       1/1     Running   0          74s
```

Verify Installation using Lens

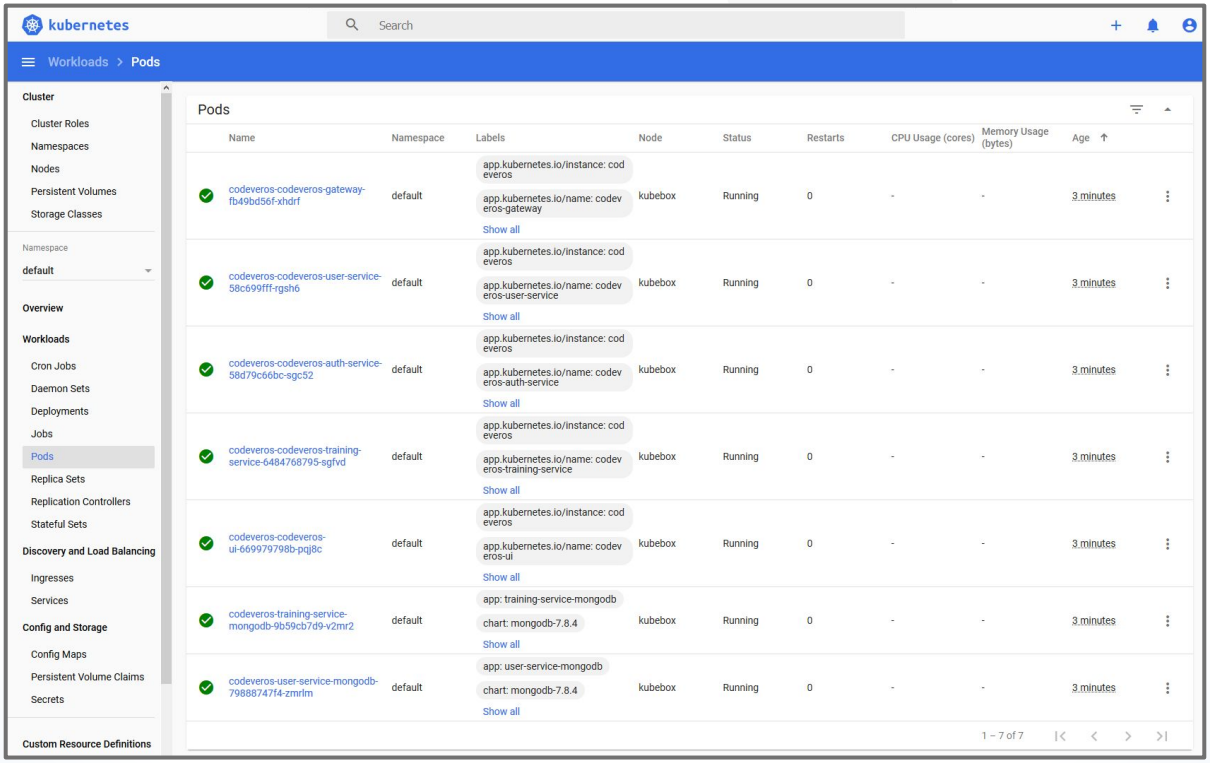
9. Connect to the cluster by clicking the icon in the left-side menu
10. Click on Apps to expand that menu choice
11. Select Releases to show the installed Helm releases
12. Click on the codeveros release to open the right sidenav which lists the configuration and installed resources



Verify Installation using Dashboard

13. Navigate to https://<EC2 IP>:30001 and sign in using the token retrieved in the Environment Setup

14. Select Pods from the left-menu under the Workloads section



Connect to Codeveros

Codeveros is distributed as an umbrella chart composed of a number of other charts. The entry point into Codeveros is the codeveros-proxy service. By default, this service is dynamically assigned a nodePort based on availability.

15. Determine nodePort assigned to the codeveros-proxy service

```
$ kubectl describe svc codeveros-proxy
```

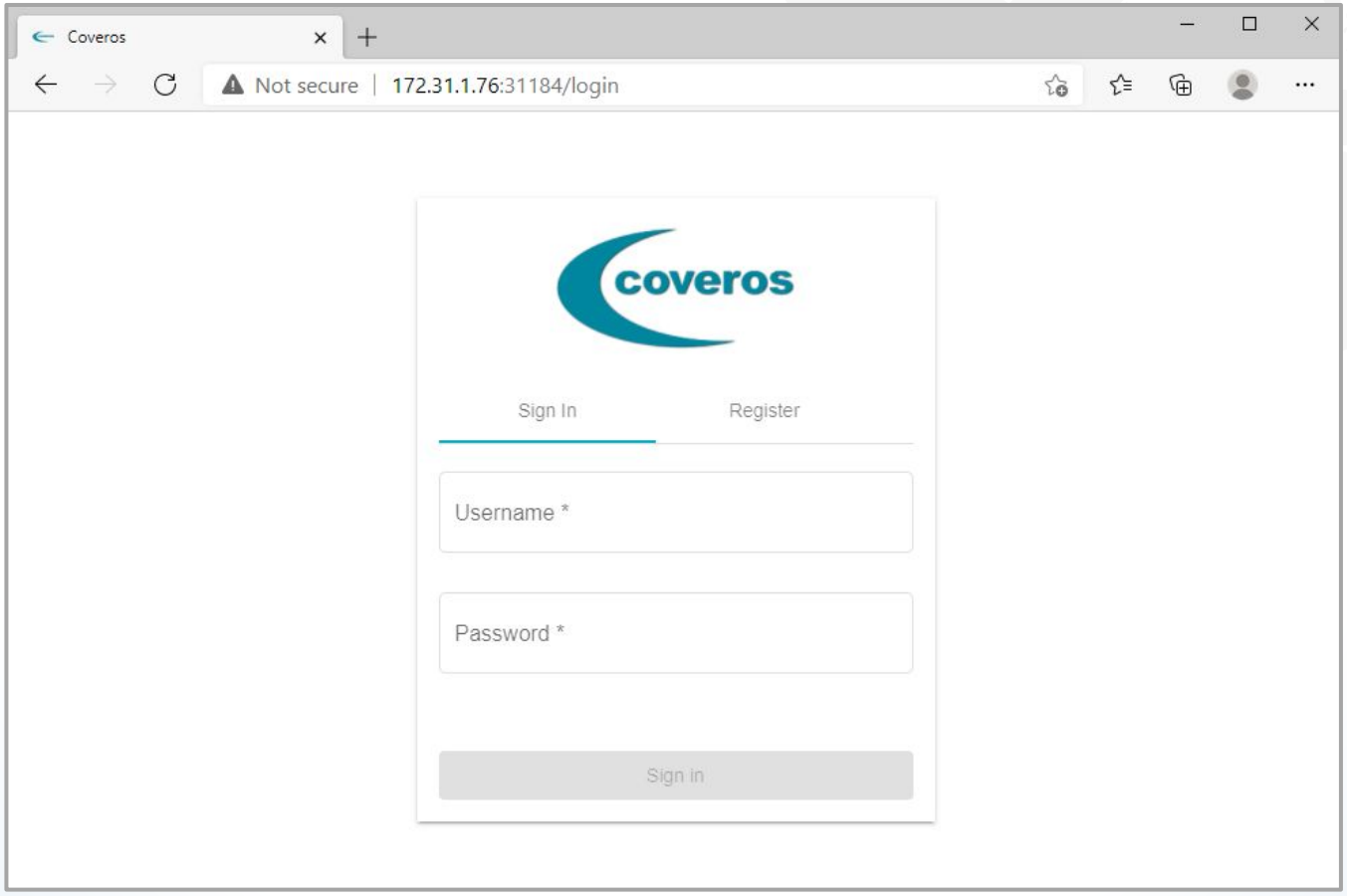
```
helmuser@ip-172-31-1-76:~$ kubectl describe svc codeveros-proxy
Name:                     codeveros-proxy
Namespace:                default
Labels:                   app.kubernetes.io/instance=codeveros
                          app.kubernetes.io/managed-by=Helm
                          app.kubernetes.io/name=proxy
                          helm.sh/chart=proxy-8.5.3
Annotations:              meta.helm.sh/release-name: codeveros
                          meta.helm.sh/release-namespace: default
Selector:                 app.kubernetes.io/instance=codeveros,app.kul
Type:                     NodePort
IP Families:              <none>
IP:                       10.152.183.84
IPs:                      10.152.183.84
Port:                     http  80/TCP
TargetPort:               http/TCP
NodePort:                 http  31184/TCP
Endpoints:                <none>
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
```

In this example, the assigned nodePort is 31184, but it will be different for every student

Try finding this same information using Lens or Kubernetes Dashboard

Connect to Codeveros

16. In a browser, navigate to http://<EC2 IP>:<nodePort> to verify that Codeveros was properly deployed



Upgrading a Release

helm upgrade

Upgrading a Release

- Upgrade a release when there is a newer chart version
- Upgrade a release to modify the configured values
- Helm commands
  - show
  - upgrade

Helm Client - helm show values

- Display chart information including metadata, readme, and defined values
- One way to discover the available values that can be overridden to customize or upgrade a release
  - The alternatives are to either pull the chart and view the file, or view it directly in its source code repository (must view the correct version)
- `helm show values <CHART>` displays the chart's values.yaml file

```
$ helm show values jenkins/jenkins

# Default values for jenkins
# This is a YAML-formatted file.
# Declare name/value pairs to be passed into your templates.
...
agent:
  enabled: true
  image: "jenkins/jenkins-slave"
...
```

Helm Client - helm upgrade

- Upgrades an installed release by providing overridden values and/or a newer chart version
- At a minimum, the release name and chart must be specified, the specified chart version can be the current one installed
- Can individually override values through the `--set` argument or provide an entire YAML override file with the `-f` argument

```
$ helm install jenkins jenkins/jenkins --version=3.1.7

$ helm ls
NAME     NAMESPACE  REVISION UPDATED    STATUS   CHART
jenkins  default    1        2021-02-05 deployed jenkins-3.1.7

$ helm upgrade jenkins jenkins/jenkins

$ helm ls
NAME     NAMESPACE  REVISION UPDATED    STATUS   CHART
jenkins  default    2        2021-02-05 deployed jenkins-3.1.8
```

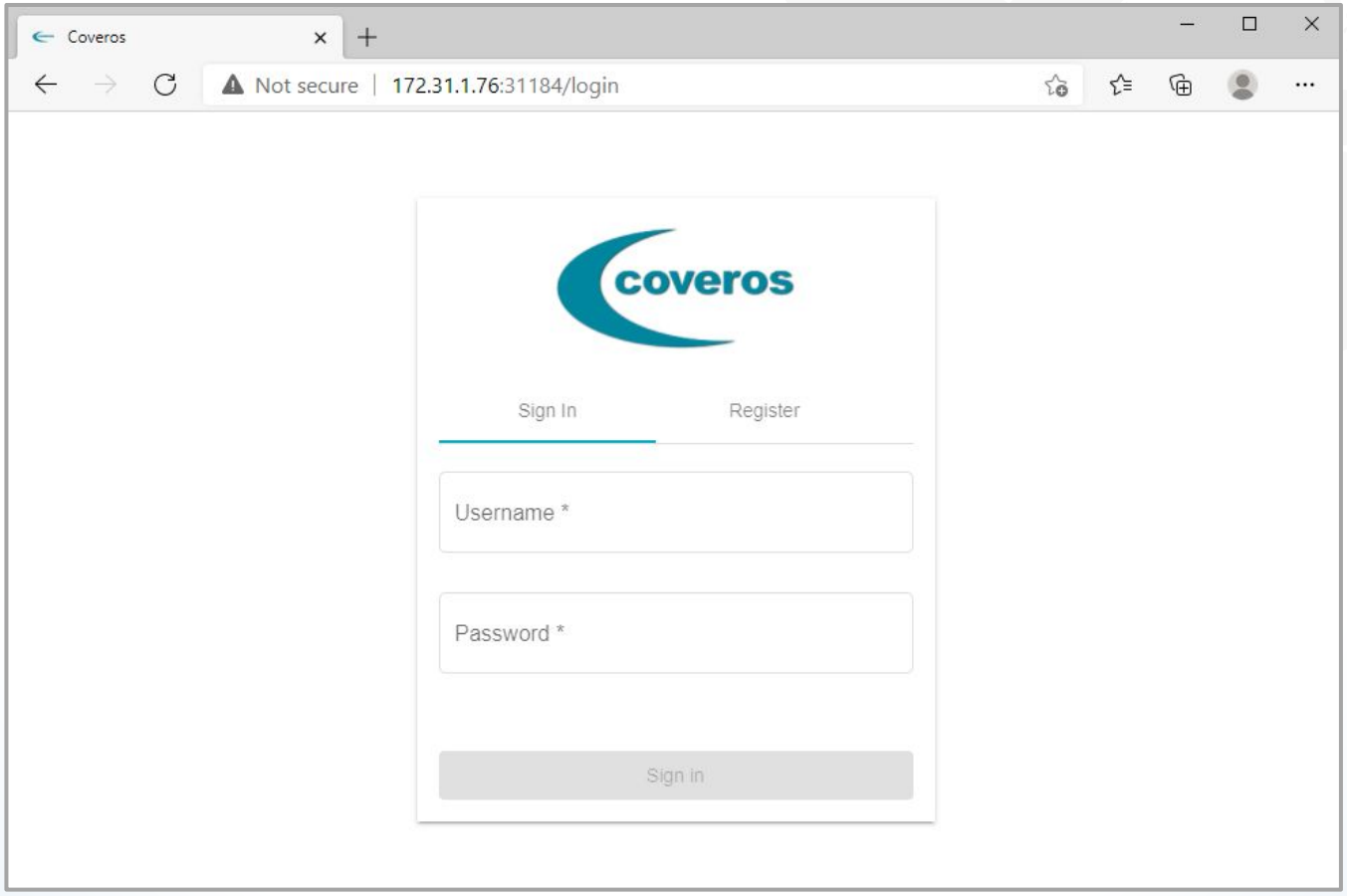
Exercise: Upgrading a Release

Exercise Objectives

- Display the available chart values
- Display the current release configuration
- Create a overrides.yaml file and customize some of the default configuration of the installed chart
- Upgrade the chart by updating the port configuration
- Walk through the changes to examine the current release status
- Verify that Codeveros is available on the updated port

Verify Codeveros is Running

1. Using the previous URL and port, navigate to http://<EC2 IP>:<nodePort> to verify that Codeveros is still running



Determine the Configurable Values

As discussed in the previous exercise, The **codeveros** chart is composed of a number of other charts. The **codeveros** chart defines the Codeveros entry point.

2. Display the **codeveros** default values

```
$ helm show values codeveros/codeveros
```

```
helmuser@ip-172-31-1-76:~$ helm show values codeveros/codeveros
codeveros-ui:
  ingress:
    enabled: false

codeveros-gateway:
  ingress:
    enabled: false

proxy:
  enabled: true
  service:
    type: NodePort
  existingServerBlockConfigmap: "{{ .Release.Name }}-proxy-config"
```

Note: The values displayed for you may be slightly different depending on the current state of the **codeveros** chart

Remember, when the nodePort is undefined, Kubernetes assigns an available value

Determine the Current Configuration

3. Use **`helm get values`** to review the current release configuration
   a. The **`-a`** argument gets all values, not just user-defined ones

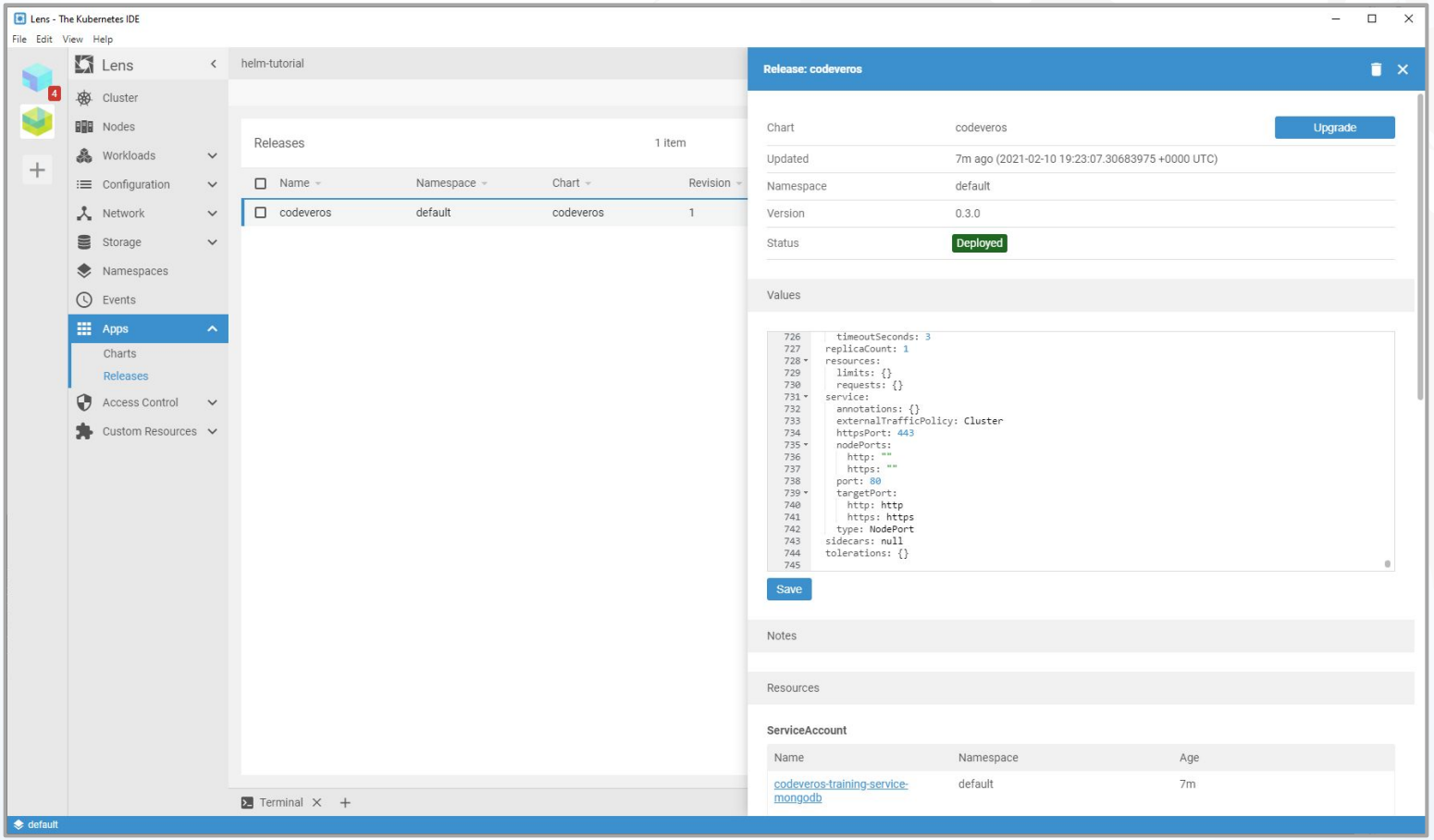
```
$ helm get values -a codeveros
```

```
service:
  annotations: {}
  externalTrafficPolicy: Cluster
  httpsPort: 443
  nodePorts:
    http: ""
    https: ""
  port: 80
  targetPort:
    http: http
    https: https
  type: NodePort
sidecars: null
tolerations: {}
```

The output may be somewhat overwhelming as every defined value is displayed, but the relevant section is contained under the **`proxy.service`** attribute. It should be located at the bottom, as the configuration is alphabetically sorted. This shows that the nodePort value is not set.

Determine the Current Configuration

The release configuration is also available through Lens



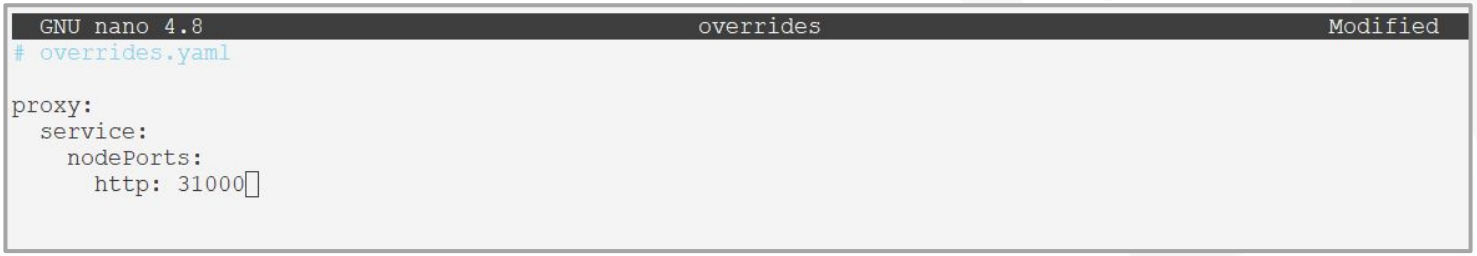
Create a Values File

4. Create a YAML file to define the codeveros service nodePort

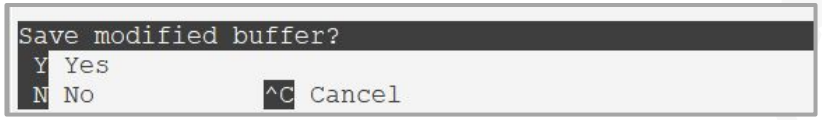
```
$ nano overrides.yaml
```

```
# overrides.yaml

proxy:
  service:
    nodePorts:
      http: 31000
```



5. To save and exit nano, press Ctrl+X, then press Y, then press Enter



Upgrade the Release

6. Observe the current release status

```
$ helm ls
$ helm status codeveros
```

```
helmuser@ip-172-31-1-76:~$ helm ls
NAME            NAMESPACE       REVISION        UPDATED                                 STATUS          CHART           APP VERSION
codeveros       default         1               2021-02-10 19:23:07.30683975 +0000 UTC  deployed        codeveros-0.3.0 1.0.0
helmuser@ip-172-31-1-76:~$ helm status codeveros
NAME: codeveros
LAST DEPLOYED: Wed Feb 10 19:23:07 2021
NAMESPACE: default
STATUS: deployed
REVISION: 1
```

7. Upgrade the release using the overrides.yaml config file

```
$ helm upgrade -f overrides.yaml codeveros codeveros/codeveros
```

```
helmuser@ip-172-31-1-76:~$ helm upgrade -f overrides.yaml codeveros codeveros/codeveros
Release "codeveros" has been upgraded. Happy Helming!
NAME: codeveros
LAST DEPLOYED: Wed Feb 10 19:36:48 2021
NAMESPACE: default
STATUS: deployed
REVISION: 2
```

Review the Changes

8. Observe the current release status

```
$ helm ls
$ helm status codeveros
```

```
helmuser@ip-172-31-1-76:~$ helm ls
NAME            NAMESPACE       REVISION        UPDATED                                 STATUS          CHART           APP VERSION
codeveros       default         2               2021-02-10 19:36:48.506714594 +0000 UTC deployed        codeveros-0.3.0 1.0.0
helmuser@ip-172-31-1-76:~$ helm status codeveros
NAME: codeveros
LAST DEPLOYED: Wed Feb 10 19:36:48 2021
NAMESPACE: default
STATUS: deployed
REVISION: 2
```

9. Observe the current release configuration

```
$ helm get values -a codeveros
```

```
service:
  annotations: {}
  externalTrafficPolicy: Cluster
  httpsPort: 443
  nodePorts:
    http: 31000
    https: ""
  port: 80
  targetPort:
    http: http
    https: https
  type: NodePort
sidecars: null
tolerations: {}
```

Notice how the **`nodePorts.http`** value is now set to 31000. Compare this to the output from Revision 1, retrieved in step 3.

Review the Changes

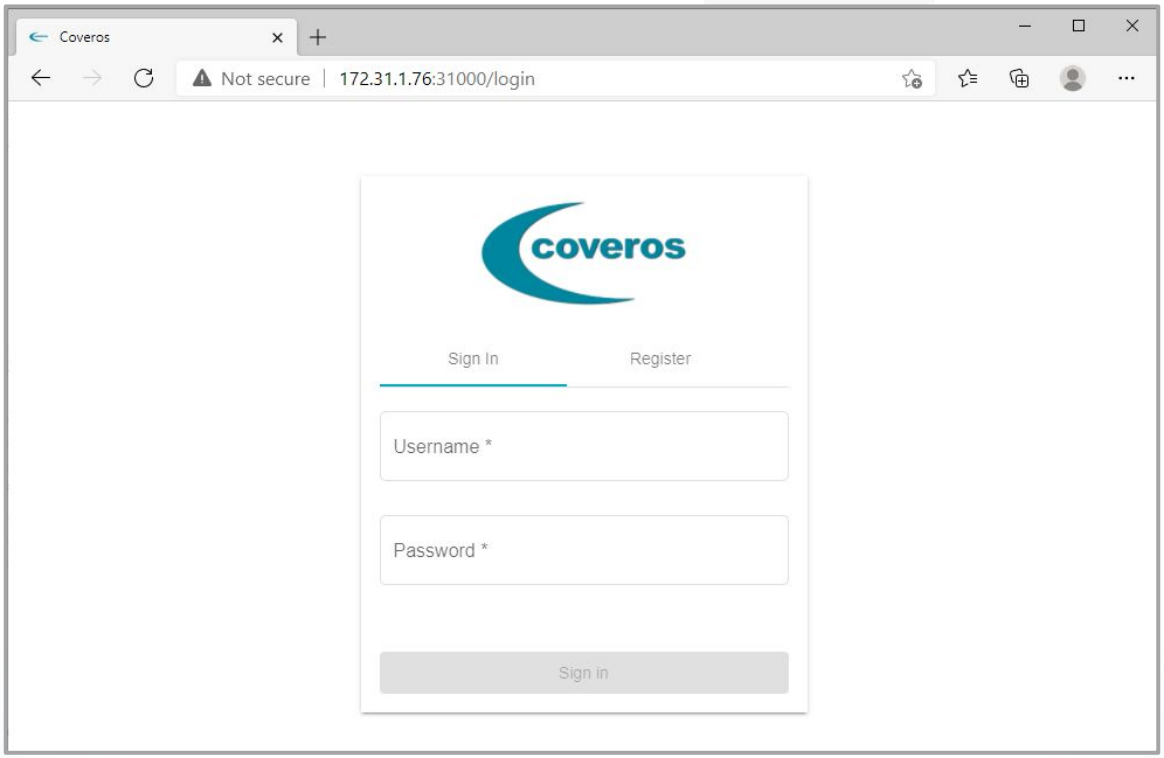
10. Observe the updated codeveros-proxy service configuration, focusing on the NodePort value

```
$ kubectl describe svc codeveros-proxy
```

```
helmuser@ip-172-31-1-76:~$ kubectl describe svc codeveros-proxy
Name:                     codeveros-proxy
Namespace:                default
Labels:                   app.kubernetes.io/instance=codeveros
                          app.kubernetes.io/managed-by=Helm
                          app.kubernetes.io/name=proxy
                          helm.sh/chart=proxy-8.5.3
Annotations:              meta.helm.sh/release-name: codeveros
                          meta.helm.sh/release-namespace: default
Selector:                 app.kubernetes.io/instance=codeveros,app.kubernetes.io/name=proxy
Type:                     NodePort
IP Families:              <none>
IP:                       10.152.183.84
IPs:                      10.152.183.84
Port:                     http  80/TCP
TargetPort:               http/TCP
NodePort:                 http  31000/TCP
Endpoints:                10.1.138.207:8080
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
```

Connect to Codeveros

11. In a browser, navigate to http://<EC2 IP>:<original nodePort> to verify that Codeveros is no longer available on that port, and then navigate to http://<EC2 IP>:31000 to verify that Codeveros has properly been updated to the new nodePort



Rollback and Uninstall

Hit the Undo Button

Helm Client - history

- Use **`helm history`** to view past revisions
- Useful when preparing to rollback a failed release

```
$ helm history jenkins
REVISION  UPDATED   STATUS     CHART          APP VERSION  DESCRIPTION
1         Fri Feb 5 superseded jenkins-3.1.7  2.263.3      Install complete
2         Fri Feb 5 deployed   jenkins-3.1.8  2.263.3      Upgrade complete
```

Helm Client - helm rollback

- Restores an installed release to a prior revision
- Useful when issues are found in an updated release
- Rolls back to previous release by default
- Can specify the exact revision as a command line parameter
- Creates a new revision

```
$ helm rollback jenkins 1
Rollback was a success! Happy Helming!

$ helm history jenkins
REVISION  UPDATED   STATUS     CHART          APP VERSION  DESCRIPTION
1         Fri Feb 5 superseded jenkins-3.1.7  2.263.3      Install complete
2         Fri Feb 5 superseded jenkins-3.1.8  2.263.3      Upgrade complete
3         Fri Feb 5 deployed   jenkins-3.1.7  2.263.3      Rollback to 1
```

Helm Client - helm uninstall

- Uninstall a release if it is no longer needed
- Removes the Kubernetes resources managed by the release
- Removes the release history by default

```
$ helm ls
NAME     NAMESPACE  REVISION UPDATED    STATUS   CHART
jenkins  default    3        2021-02-05 deployed jenkins-3.1.7

$ helm uninstall jenkins
release "jenkins" uninstalled

$ helm ls
NAME     NAMESPACE  REVISION UPDATED    STATUS   CHART

$ helm status jenkins
Error: release: not found
```

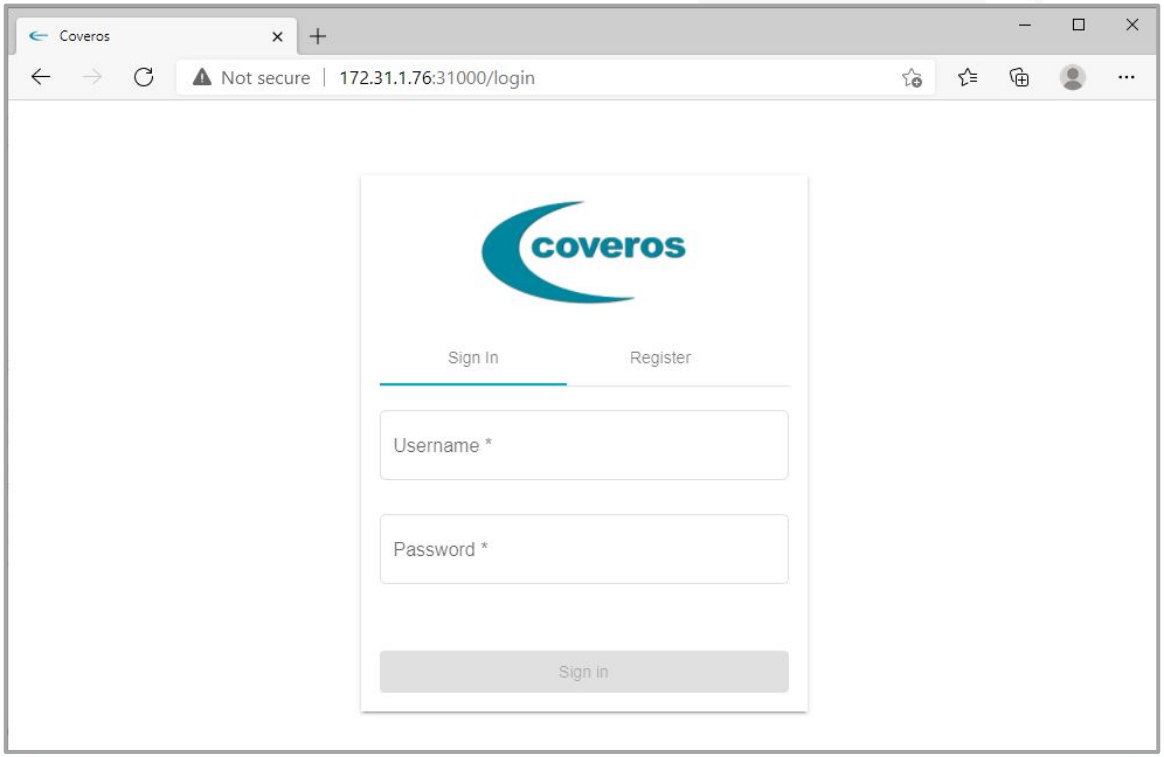
Exercise: Rollback and Uninstall

Exercise Objectives

- Review the current state and history before and after rollback
- Rollback a release to a prior revision
- Uninstall a release
- Verify that the release has been removed and Codeveros is no longer accessible

Connect to Codeveros

1. In a browser, navigate to http://<EC2 IP>:31000 to verify that Codeveros is still accessible



Review Release History

2. Review the release status

```
$ helm ls
$ helm status codeveros
$ helm history codeveros
```

```
helmuser@ip-172-31-1-76:~$ helm ls
NAME            NAMESPACE       REVISION        UPDATED                                 STATUS          CHART           APP VERSION
codeveros       default         2               2021-02-10 19:36:48.506714594 +0000 UTC deployed        codeveros-0.3.0 1.0.0
helmuser@ip-172-31-1-76:~$ helm status codeveros
NAME: codeveros
LAST DEPLOYED: Wed Feb 10 19:36:48 2021
NAMESPACE: default
STATUS: deployed
REVISION: 2
helmuser@ip-172-31-1-76:~$ helm history codeveros
REVISION        UPDATED                         STATUS          CHART           APP VERSION     DESCRIPTION
1               Wed Feb 10 19:23:07 2021        superseded      codeveros-0.3.0 1.0.0           Install complete
2               Wed Feb 10 19:36:48 2021        deployed        codeveros-0.3.0 1.0.0           Upgrade complete
```

There are two revisions: when codeveros was originally installed, and when the nodePort was updated

Rollback the Release

3. Rollback to the first revision

```
$ helm rollback codeveros 1
```

4. Review the release status

```
$ helm ls
$ helm status codeveros
$ helm history codeveros
```

```
helmuser@ip-172-31-1-76:~$ helm rollback codeveros 1
Rollback was a success! Happy Helming!
helmuser@ip-172-31-1-76:~$ helm ls
NAME            NAMESPACE       REVISION        UPDATED                                 STATUS          CHART           APP VERSION
codeveros       default         3               2021-02-10 19:42:01.97897634 +0000 UTC  deployed        codeveros-0.3.0 1.0.0
helmuser@ip-172-31-1-76:~$ helm status codeveros
NAME: codeveros
LAST DEPLOYED: Wed Feb 10 19:42:01 2021
NAMESPACE: default
STATUS: deployed
REVISION: 3
helmuser@ip-172-31-1-76:~$ helm history codeveros
REVISION        UPDATED                         STATUS          CHART           APP VERSION     DESCRIPTION
1               Wed Feb 10 19:23:07 2021        superseded      codeveros-0.3.0 1.0.0           Install complete
2               Wed Feb 10 19:36:48 2021        superseded      codeveros-0.3.0 1.0.0           Upgrade complete
3               Wed Feb 10 19:42:01 2021        deployed        codeveros-0.3.0 1.0.0           Rollback to 1
```

Revision 2 was superseded and the current revision is now 3

Review the Release Configuration

5. Review the release configuration

```
$ helm get values -a codeveros
```

```
service:
  annotations: {}
  externalTrafficPolicy: Cluster
  httpsPort: 443
  nodePorts:
    http: ""
    https: ""
  port: 80
  targetPort:
    http: http
    https: https
  type: NodePort
sidecars: null
tolerations: {}
```

Notice how the **`nodePorts.http`** value is no longer set, it has been reverted to the Revision 1 state

Review the Service Configuration

6. Observe the updated codeveros service configuration, focusing on the NodePort value

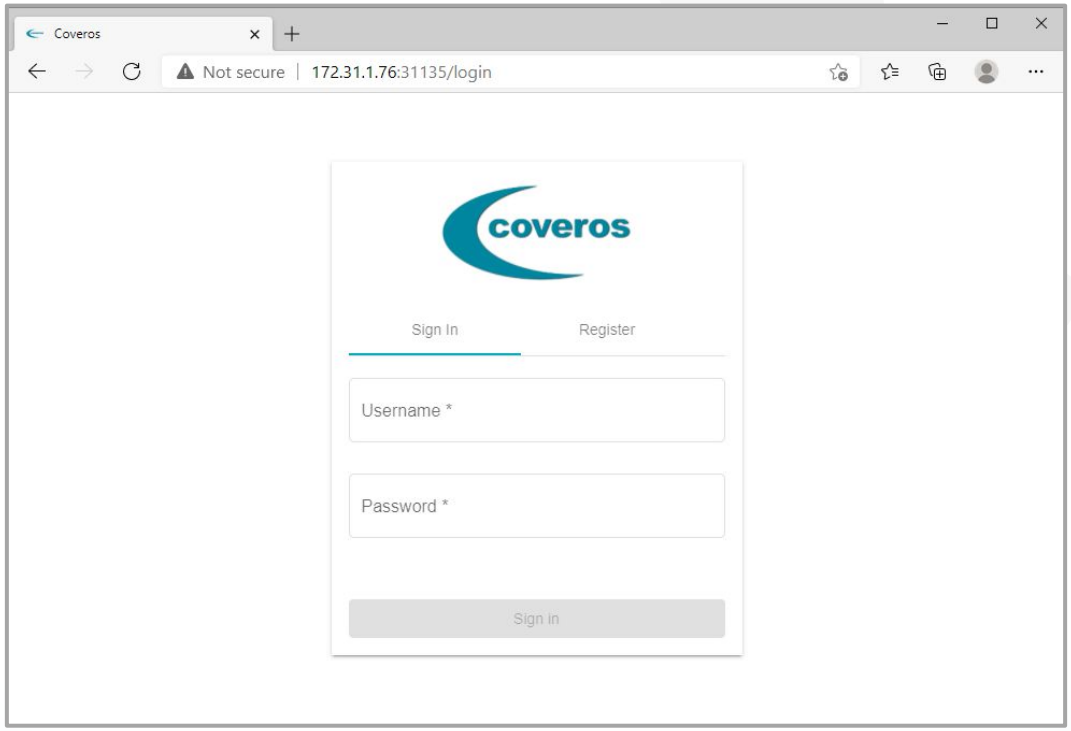
```
$ kubectl describe svc codeveros-proxy
```

```
helmuser@ip-172-31-1-76:~$ kubectl describe svc codeveros-proxy
Name:                     codeveros-proxy
Namespace:                default
Labels:                   app.kubernetes.io/instance=codeveros
                          app.kubernetes.io/managed-by=Helm
                          app.kubernetes.io/name=proxy
                          helm.sh/chart=proxy-8.5.3
Annotations:              meta.helm.sh/release-name: codeveros
                          meta.helm.sh/release-namespace: default
Selector:                 app.kubernetes.io/instance=codeveros,app.kubernetes.io/name=proxy
Type:                     NodePort
IP Families:              <none>
IP:                       10.152.183.84
IPs:                      10.152.183.84
Port:                     http  80/TCP
TargetPort:               http/TCP
NodePort:                 http  31135/TCP
Endpoints:                10.1.138.207:8080
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
```

Notice that the NodePort may no longer be 31000 as it was in Revision 2. But it is likely also not the original port value that existed in Revision 1. Why is that? Use this NodePort value for the next step.

Connect to Codeveros

7. In a browser, navigate to http://<EC2 IP>:31000 to verify that Codeveros is no longer available on that port (if it has changed), and then navigate to http://<EC2 IP>:<nodePort> to verify that Codeveros has properly been updated to the new nodePort



Uninstall the Release

8. Uninstall the release

```
$ helm uninstall codeveros
```

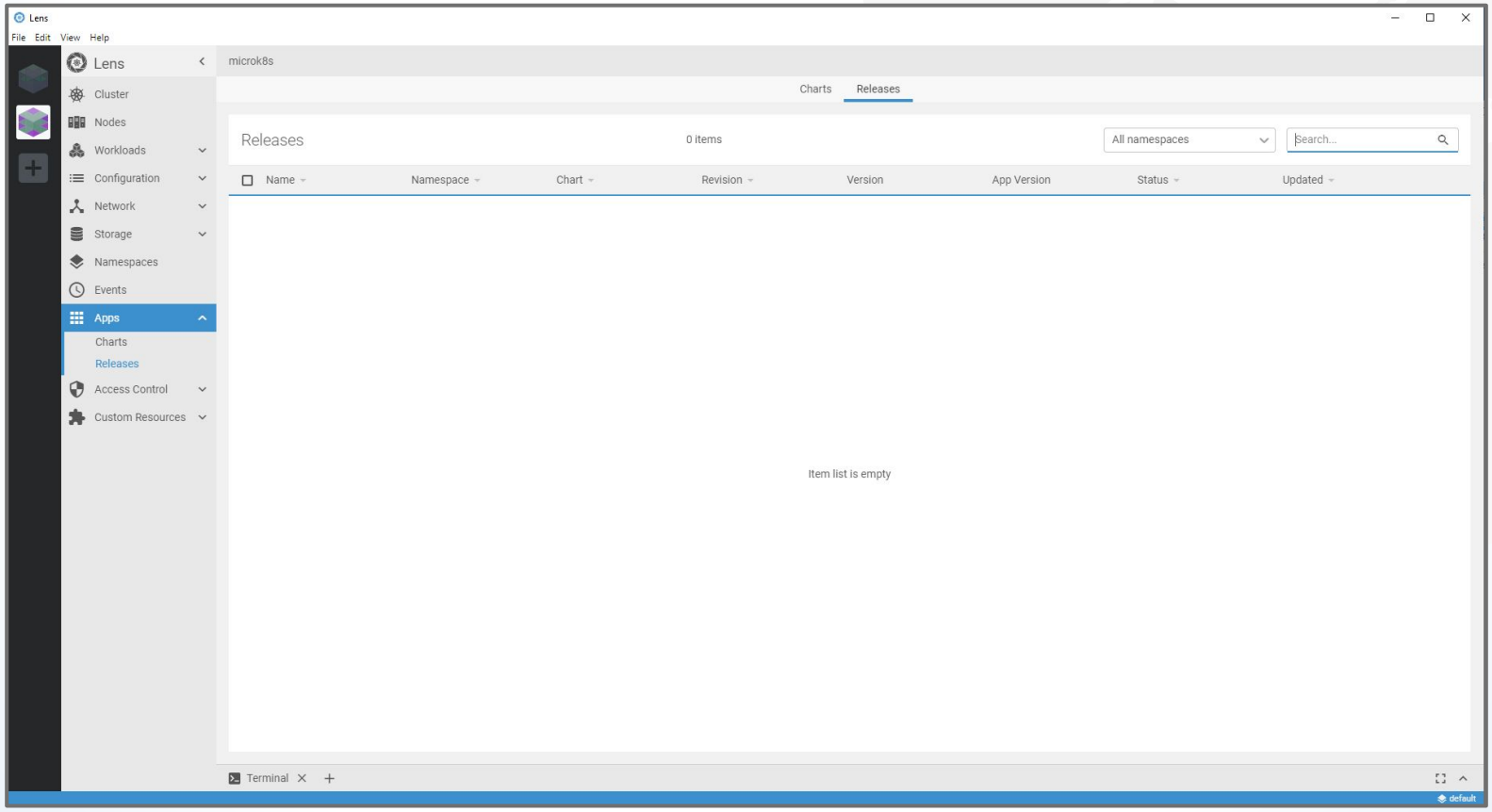
9. Verify the uninstallation

```
$ helm ls
$ helm status codeveros
$ helm history codeveros
$ kubectl describe service codeveros-proxy
```

```
helmuser@ip-172-31-1-76:~$ helm uninstall codeveros
helmrelease "codeveros" uninstalled
helmuser@ip-172-31-1-76:~$ helm ls
NAME    NAMESPACE       REVISION        UPDATED STATUS  CHART   APP VERSION
helmuser@ip-172-31-1-76:~$ helm status codeveros
Error: release: not found
helmuser@ip-172-31-1-76:~$ helm history codeveros
Error: release: not found
helmuser@ip-172-31-1-76:~$ kubectl describe service codeveros-proxy
Error from server (NotFound): services "codeveros-proxy" not found
```

Verify Uninstallation

10. Use Lens or the Kubernetes Dashboard to verify the release was uninstalled



Connect to Codeveros

11. Verify that you are no longer able to connect to Codeveros on any of the previous configured ports

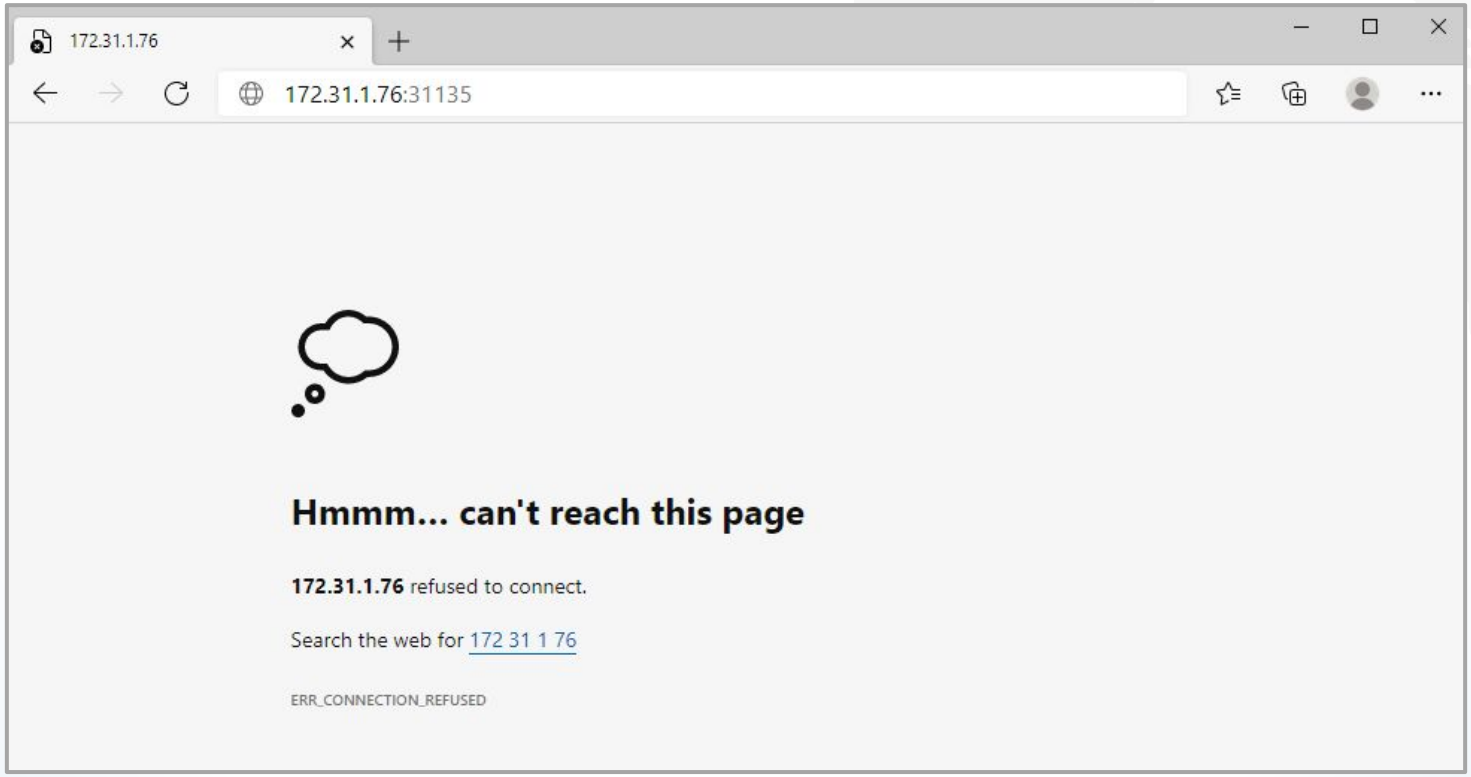


Chart Creation

helm create my-chart

Helm Client - helm create

- Generates an installable skeleton chart with the required files, as well as commonly used templates and design patterns
- Nginx-based chart by default
- Can specify a different starter scaffold with `--starter`

```
$ helm create my-new-chart
Creating my-new-chart

$ ls my-new-chart/
Chart.yaml charts templates values.yaml

$ helm install my-new-chart my-new-chart/
Name: my-new-chart
LAST DEPLOYED: Fri Feb 5 20:26:15 2021
NAMESPACE: default
STATUS: deployed
REVISION: 1
NOTES: ...
```

Chart Creation

There is no requirement to using **`helm create`** to generate a new chart, though it can speed up the initial creation process and assist with structuring files based on recommended practices, especially if using a custom starter scaffold

The minimum requirement for a chart is a Chart.yaml file with the name, version, and apiVersion defined

```
useless-chart/
  Chart.yaml
```

```
# useless-chart/Chart.yaml
apiVersion: v2
name: useless-chart
version: 1.0.0
```

```
$ helm install useless-chart useless-chart/
NAME: useless-chart
LAST DEPLOYED: Fri Feb 5 20:27:43 2021
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

Helm Chart Components

Key Components

- Chart.yaml
- values.yaml
- templates folder
- charts folder
- templates/NOTES.txt

Other Components

- LICENSE
- README.md
- crds folder
- values.schema.json

Chart.yaml

- Defines the chart metadata
- Fields include
  - apiVersion *required* - Set to v2 for Helm 3.x support
  - name *required*
  - description
  - version *required* - Chart version
  - appVersion
  - dependencies
  - ...other fields
- Values accessible in templates under the `.Charts` object
- Version must follow semantic versioning

Chart.yaml Example

```
# Chart.yaml

apiVersion: v2
name: codeveros
description: An umbrella chart for Codeveros
type: application
version: 0.2.0
appVersion: 1.0.0
dependencies:
  - name: codeveros-user-service
    version: 0.5.0
    repository: https://coveros.github.io/codeveros
  - name: codeveros-training-service
    version: 0.5.0
    repository: https://coveros.github.io/codeveros
  - name: codeveros-auth-service
    version: 0.5.0
    repository: https://coveros.github.io/codeveros
  - name: codeveros-gateway
    version: 0.4.0
    repository: https://coveros.github.io/codeveros
  - name: codeveros-ui
    version: 0.6.0
    repository: https://coveros.github.io/codeveros
```

values.yaml

- Plain YAML key value pair file
- Defines the default values available on the `.Values` object, central to the ability to create configurable template files
- Lowest level of value definition, overridden by parent charts and by user-supplied values

```
# values.yaml

service:
  type: ClusterIP
  port: 8080
```

```
# templates/service.yaml

...
spec:
  type: {{ .Values.service.type }}
  ports:
    - port: {{ .Values.service.port }}
...
```

templates Folder

- Contains the chart's template files
- All files will be run through the Helm template renderer so non-template files are not allowed in the templates folder
- Files that start with a underscore (_) indicate they define named templates that do not generate Kubernetes resources
- All other files -- besides NOTES.txt -- are expected to generate Kubernetes resources

charts folder

- Stores child/dependent charts
- Stored as archives or as expanded chart directories
- Installation will fail if any charts listed in the Chart.yaml dependencies section are not present in the charts directory
- All enabled charts located in the charts directory -- even if not specified as a dependency in Chart.yaml -- will be installed
- Charts specified as a dependency in Chart.yaml can be automatically retrieved using the `helm dependency` cli command

templates/NOTES.txt

- After a chart is installed or upgraded, the NOTES.txt content is printed for the user
- It is configurable like every other file in the templates directory
- Useful for printing out installation and usage instructions based on the defined configuration

```
# templates/NOTES.txt
Well done, you just installed {{ .Release.Name }}

Service is available at {{ .Values.service.port }}
```

```
$ helm install codeveros-test codeveros/codeveros
NAME: codeveros-test
...
NOTES:
Well done, you just installed codeveros-test

Service is available at 8080
```

Exercise: Create a Chart

Exercise Objectives

- Use **`helm create`** to generate an nginx-based starter chart
- Review the chart files and directory structure
- Modify the chart metadata and default values
- Install the chart and observe the results

Chart Creation

1. Use **`helm create`** to generate an nginx-based starter chart

```
$ helm create my-new-chart
```

2. Review the generated files and directory structure

```
$ ls my-new-chart/
$ ls my-new-chart/charts/
$ ls my-new-chart/templates/
$ ls my-new-chart/templates/tests/
```

```
helmuser@ip-172-31-1-140:~$ helm create my-new-chart
Creating my-new-chart
helmuser@ip-172-31-1-140:~$ ls my-new-chart/
Chart.yaml  charts  templates  values.yaml
helmuser@ip-172-31-1-140:~$ ls my-new-chart/charts/
helmuser@ip-172-31-1-140:~$ ls my-new-chart/templates/
NOTES.txt  _helpers.tpl  deployment.yaml  hpa.yaml  ingress.yaml  service.yaml  serviceaccount.yaml  tests
helmuser@ip-172-31-1-140:~$ ls my-new-chart/templates/tests/
test-connection.yaml
```

Update Chart.yaml

3. Review generated Chart.yaml

```
$ cat my-new-chart/Chart.yaml
```

4. Personalize the Chart description (e.g. "My first Helm Chart")

```
$ nano my-new-chart/Chart.yaml
```

```
helmuser@ip-172-31-1-140:~$ cat my-new-chart/Chart.yaml
apiVersion: v2
name: my-new-chart
description: A Helm chart for Kubernetes

# A chart can be either an 'application' or a 'library' chart.
#
# Application charts are a collection of templates that can be packaged into versioned archives
# to be deployed.
#
# Library charts provide useful utilities or functions for the chart developer. They're included as
# a dependency of application charts to inject those utilities and functions into the rendering
# pipeline. Library charts do not define any templates and therefore cannot be deployed.
type: application

# This is the chart version. This version number should be incremented each time you make changes
# to the chart and its templates, including the app version.
# Versions are expected to follow Semantic Versioning (https://semver.org/)
version: 0.1.0

# This is the version number of the application being deployed. This version number should be
# incremented each time you make changes to the application. Versions are not expected to
# follow Semantic Versioning. They should reflect the version the application is using.
# It is recommended to use it with quotes.
appVersion: "1.16.0"
```

```
  GNU nano 4.8                                  my-new-chart/Chart.yaml                                  Modified
apiVersion: v2
name: my-new-chart
description: My first Helm Chart

# A chart can be either an 'application' or a 'library' chart.
```

Update values.yaml

5. Review the generated values.yaml

```
$ cat my-new-chart/values.yaml
```

6. Update the service configuration to enable external connectivity

```
$ nano my-new-chart/values.yaml
```

```
GNU nano 4.8                                    my-new-chart/values.yaml
  # runAsUser: 1000

service:
  type: ClusterIP
  port: 80
```

```
GNU nano 4.8                                    my-new-chart/values.yaml
  # runAsUser: 1000

service:
  type: NodePort
  port: 80
```

Don't forget to save your changes!

Review Service Template

7. View the generated service.yaml template

```
$ cat my-new-chart/templates/service.yaml
```

```
helmuser@ip-172-31-1-140:~$ cat my-new-chart/templates/service.yaml
apiVersion: v1
kind: Service
metadata:
  name: {{ include "my-new-chart.fullname" . }}
  labels:
    {{- include "my-new-chart.labels" . | nindent 4 }}
spec:
  type: {{ .Values.service.type }}
  ports:
    - port: {{ .Values.service.port }}
      targetPort: http
      protocol: TCP
      name: http
  selector:
    {{- include "my-new-chart.selectorLabels" . | nindent 4 }}
```

In later sections, we will learn what all this syntax means, and how to create our own templates. For now, it is sufficient to know that this template produces a Kubernetes Service, and the spec.type value: **`{{ .Values.service.type }}`** is where the value you just set to **`NodePort`** is used.

Install the Chart

8. Install your new chart from its directory

```
$ helm install my-new-chart my-new-chart
```

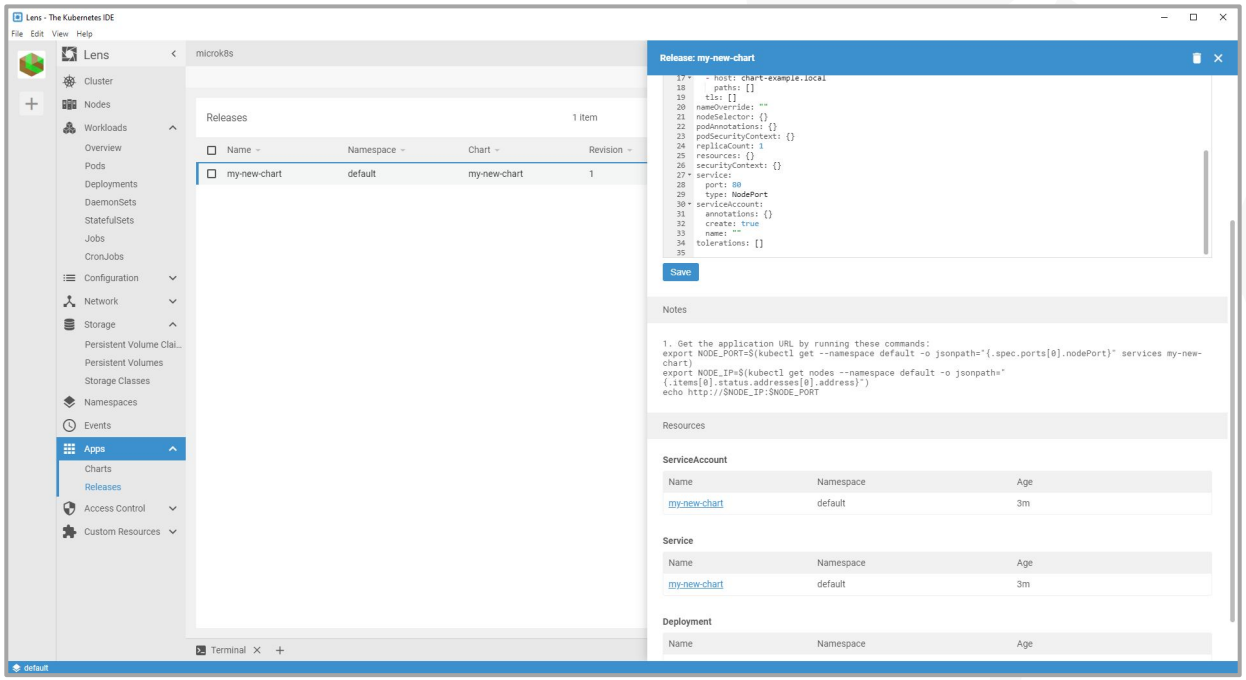
9. Follow the instructions in the installation notes to retrieve the URL

```
helmuser@ip-172-31-1-140:~$ helm install my-new-chart my-new-chart/
NAME: my-new-chart
LAST DEPLOYED: Fri Feb  5 20:39:11 2021
NAMESPACE: default
STATUS: deployed
REVISION: 1
NOTES:
1. Get the application URL by running these commands:
  export NODE_PORT=$(kubectl get --namespace default -o jsonpath="{.spec.ports[0].nodePort}" services my-new-chart)
  export NODE_IP=$(kubectl get nodes --namespace default -o jsonpath="{.items[0].status.addresses[0].address}")
  echo http://$NODE_IP:$NODE_PORT
helmuser@ip-172-31-1-140:~$  export NODE_PORT=$(kubectl get --namespace default -o jsonpath="{.spec.ports[0].nodePort}" services my-new-chart)
helmuser@ip-172-31-1-140:~$ export NODE_IP=$(kubectl get nodes --namespace default -o jsonpath="{.items[0].status.addresses[0].address}")
helmuser@ip-172-31-1-140:~$ echo http://$NODE_IP:$NODE_PORT
http://172.31.1.140:31809
```

In this example, the application is available at http://172.31.1.140:31809

Viewing the Release

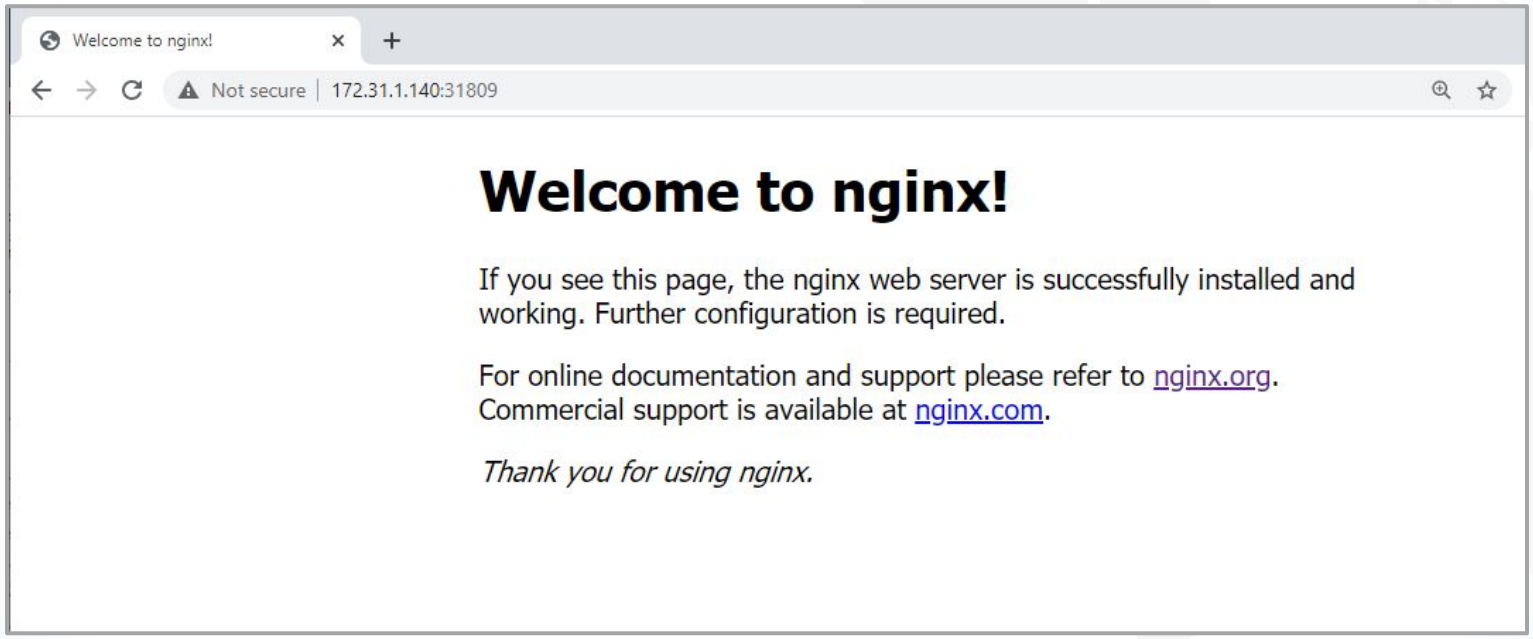
10. Using kubectl, helm, Lens, and the Kubernetes Dashboard, take some time and explore the resources that have created



For example in Lens, here are the resources attached to the release, and the applied configuration

Connect to the Application

11. Use the URL from Step 9 to navigate to the application (** You will need to replace the IP address with the provided public IP address)



12. Uninstall the release when finished

```
$ helm uninstall my-new-chart
```

Helm Templating

{{ .Values.subTitle }}

Templating Basics

- YAML Primer
- Variable usage
- Dot (.) cursor
- Built-in objects

YAML Primer

- Superset of JSON, any valid JSON is valid YAML
- YAML focused on human readability
- Structure denoted by whitespace indentation not braces
- Whitespace required as part of separators (:), (-)
  - Eliminates need to escape special characters
- Composed of collections (lists and maps) and scalars

YAML Scalars

- Strings, integers, floats, booleans etc
- YAML automatically detects simple types without the need for explicit typing
- Enclosing quotes only needed for certain special characters or disambiguating a string from another type

```
aNumber: 123                  # Auto detected as integer
aString: This is a String     # Auto detected as string
aNumberStr: "123"             # Disambiguated as string
aBoolean: Yes                 # Auto detected as boolean
aBooleanStr: "Yes"            # Disambiguated as string
```

YAML Lists

- Also known as sequences or arrays
- Block format denotes members by leading hyphen
- Inline format encloses members in square brackets, separated by commas
- Recommend using block format for most purposes

```
listFormats:
  - block
  - inline
```

```
listFormats:[block,inline]
```

YAML Maps

- Also known as hash, dictionary, associative array
- Key/Value pairs separated with colon space (key: value)
- Block format separates members by line
- Inline format encloses members in curly brackets, separated by commas
- Recommend block format for most purposes

```
service:
  type: ClusterIP
  port: 8080
```

```
service: {type: ClusterIP, port: 8080}
```

YAML Collections

Lists can contain maps rather than basic scalars

```
ports:
  - name: http
    targetPort: 8080
    port: 8080
  - name: docker
    targetPort: 5003
    port: 5003
```

Variable Usage

- Helm templates are Go templates
- Data evaluations and control structures delimited by double curly braces "{{" and "}}"
- For example, if there is a variable $fullName that equals the string "user-service" then:

```
apiVersion: v1
kind: Service
metadata:
  name: {{ $fullName }}
...
```

```
apiVersion: v1
kind: Service
metadata:
  name: user-service
...
```

Dot (.) Cursor

- Templates are executed by applying them to a data structure
- Dot (.) refers to the value at the current location in the structure
- The built-in and top-level objects are referenced with a leading (.)
  - {{ .Values.imageTag }}
- Various mechanisms exist to change dot's scope, altering the location it references
  - range, with, etc.

Built-in Objects

Top level objects accessible in templates

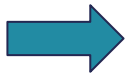
- Release
- Values
- Chart
- Capabilities
- Files
- Template

Values Object

- Top-level object used to access values defined in values.yaml and from user-supplied sources

```
# values.yaml
service:
  type: ClusterIP
  port: 8080
```

```
apiVersion: v1
kind: Service
spec:
  type: {{ .Values.service.type }}
  ports:
    - port: {{ .Values.service.port }}
      protocol: TCP
      targetPort: http
```



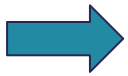
```
apiVersion: v1
kind: Service
spec:
  type: ClusterIP
  ports:
  -   port: 8080
      protocol: TCP
      targetPort: http
```

Release Object

- Includes chart release metadata such as the name, namespace, revision, etc.
- Name, Namespace, IsUpgrade, IsInstall, Revision, Service
- For example, using {{ .Release.Name }} inserts the release name into a template

```
$ helm install my-release coveros/codeveros
```

```
apiVersion: v1
kind: Service
metadata:
  labels:
    name: {{ .Release.Name }}
spec:
...
```



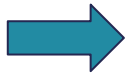
```
apiVersion: v1
kind: Service
metadata:
  labels:
    name: my-release
spec:
...
```

Chart Object

- Includes all chart metadata defined in Chart.yaml
- Name, Version, AppVersion etc
- For example, using {{ .Chart.Name }} inserts the chart name into a template

```
# Chart.yaml
apiVersion: v2
name: user-service
version: 1.0.0
...
```

```
apiVersion: v1
kind: Service
metadata:
  labels:
    chartName: {{ .Chart.Name }}
    chartVersion: {{ .Chart.Version }}
...
```



```
apiVersion: v1
kind: Service
metadata:
  labels:
    chartName: user-service
    chartVersion: 1.0.0
...
```

Capabilities Object

- Used to access cluster information
- APIVersion describes supported API versions
- KubeVersion is the Kubernetes version

```
{{- if semverCompare ">=1.20-0" .Capabilities.KubeVersion.Version -}}
apiVersion: networking.k8s.io/v1
{{- else -}}
apiVersion: networking.k8s.io/v1beta1
{{- end -}}
kind: Ingress
...
```

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
...
```

Files and Template Objects

- Files is used to access non-template files, to inject unrendered content into templates.
  - Discussed in greater detail in a later section
- Template used to access information about the current template
  - Name and Basepath information

Advanced Templating

Functions and Pipelines

Functions

- Template functions accept input arguments and produce output
- Used for transforming and evaluating data
- Helm includes the standard Go templating functions as well as the Sprig Go library that provides over 100 more
  - https://github.com/Masterminds/sprig

```
{{ functionName arg1 }}
```

```
{{ printf "%s-%s" .Chart.Name .Chart.Version }}
```

Pipelines

- Chain template commands together to sequentially transform data
- Pipeline commands are either simple values or functions
- The previous command's output become the next command's input
- Commands separated with the pipeline character '|'

```
{{ .Release.Name | trunc 63 | upper | trimSuffix "-" }}
```

Function Structuring

- Functions can either be structured as a function call with input arguments or as a chained pipeline
- Chained pipelines send the result of the previous evaluation as the function's last input argument
- Both ways are equivalent, though chaining is more common

```
{{ functionName arg1 }}
```

```
{{ arg1 | functionName }}
```

```
{{ functionName arg1 arg2 }}
```

```
{{ arg2 | functionName arg1 }}
```

Grouping

- Parentheses can be used for grouping
- Useful when needing to chain together complex pipelines or multiple operations

```
{{ arg1 | funcOne (funcTwo (arg2 | funcThree)) | funcFour }}
```

```
{{ .Values.existing | default (include "chart.fullname" .) }}
```

```
{{ range (.paths | default (list "/")) }}
```

```
{{ if (and (eq .Values.type "NodePort") (not (empty .Values.nodePort))) }}
```

A Sample of Functions

- default
- quote
- toYaml
- indent/nindent
- printf
- required
- tpl
- lookup
- Comparison operators (eq, ne, lt, le, gt, ge)
- Boolean functions (and, or, not)
- ...and many, many, more

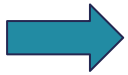
Functions - default

- Allows outputting a default value if the input evaluates to false

```
# templates/configmap.yaml
...
data:
  first: {{ default "Bobby" .Values.firstName }}
  last: {{ .Values.lastName | default "Foster" }}
```

```
# values.yaml

firstName: Bob
lastName:
```



```
...
data:
  first: Bob
  last: Foster
```

```
# values.yaml

firstName:
lastName: Two Shoes
```

```
...
data:
  first: Bobby
  last: Two Shoes
```

(In practice, static defaults should be set in values.yaml)

Functions - quote

- Forces the rendered manifest to wrap content in quotes
- Helps disambiguate string values when strings are required

```
# values.yaml
db:
  port: 8080
  host: db-service
```

```
# templates/deployment.yaml
...
spec:
  containers:
    - env:
        - name: DB_PORT
          value: {{ .Values.db.port | quote }}
```

```
...
spec:
  containers:
    - env:
        - name: DB_PORT
          value: "8080"
```

Functions - toYaml

- Used to render blocks of YAML in templated resource files
- Useful when further processing is not needed in the template

```
# values.yaml

ingress:
  labels:
    kubernetes.io/ingress.class: nginx
    kubernetes.io/tls-acme: true
```

```
# templates/ingress.yaml
...
metadata:
  labels:
    {{- toYaml .Values.ingress.labels | nindent 4 }}
...
```

Functions - indent/nindent

- YAML uses whitespace to denote structure, unlike JSON which uses curly braces ({...}) to enclose and scope
- Helm generates Kubernetes YAML manifest files, so the output must be properly formatted
- indent and nindent are used to define the indentation level of the rendered manifest
- nindent adds an initial newline character
- Every line of text is indented by the specified amount

Functions - indent/nindent

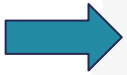
```
# values.yaml

labels:
  first: hi
  second: hello
```

```
labels:
{{ toYaml .Values.labels }}
```

```
labels:
first: hi
second: hello
```

```
labels:
{{ toYaml .Values.labels | indent 2 }}
```



```
labels:
  first: hi
  second: hello
```

```
labels: {{toYaml .Values.labels | nindent 2}}
```

```
labels:
  first: hi
  second: hello
```

```
labels:
  {{- toYaml .Values.labels | nindent 2 }}
```

```
labels:
  first: hi
  second: hello
```

Functions - printf

- One of many String manipulation functions
- Outputs formatted string based on provided format specifier

```
{{ printf "%s-%s" .Chart.Name .Chart.Version }}
```

```
{{ .Chart.Name | printf "The chart name is %s" }}
```

Functions - required

- Specifies a required value, causing the chart installation to fail if the value is empty

```
{{ arg1 | required "Error Message" }}
```

```
{{ required "Error Message" arg1 }}
```

```
{{ required "You must supply a name" .Values.name }}
```

Functions - tpl

- Allows evaluating strings as templates
- Enables defining dynamic values, since the values.yaml file is itself not dynamic

```
{{ tpl TEMPLATE_STRING VALUES }}
```

```
# values.yaml
name: "Bob"
fullName: "{{ .Values.name }} Foster"
```

```
# templates/configmap.yaml
...
data:
  name: {{ tpl .Values.fullName }}
  literal: {{ .Values.fullName }}
```

```
# output
...
data:
  name: Bob Foster
  literal: "{{ .Values.name }} Foster"
```

Functions - lookup

- Used to access information on resources running in the cluster
- Namespace and Name fields are optional, which allows retrieving multiple resources
- Retrieves a single resource as a map, retrieves a lists of resources as a map with an "items" field
- Mimics ‘kubectl get’ functionality

```
{{ lookup apiVersion kind namespace name }}
```

```
{{ $myPod := (lookup "v1" "Pod" "codeveros" "user-service") }}
podName: {{ $myPod.metadata.name }}


podNames:
{{- range $index, $pod := (lookup "v1" "Pod" "codeveros" "") }}
  - {{ $pod.metadata.name }}
{{- end }}
```

Functions - Comparisons

- Comparison operators are functions
  - eq - Equality check
  - ne - Negative equality check
  - lt - Less than
  - le - Less than or equal
  - gt - Greater than
  - ge - Greater than or equal
- Can be combined with boolean functions -- and, not, or -- and grouping () to create complex statements

```
{{ if (and (eq arg1 arg2) (or (not (empty arg3)) (lt arg4 arg5))) }}
  {{/* DO SOMETHING */}}
{{ end }}
```

Advanced Templating

Named Templates

Partials

- Helm assumes files in the templates/ folder define Kubernetes resource manifests
- Files in the templates/ folder that start with _ do not generate a Kubernetes manifest file
- By convention: templates/_helpers.tpl
- Can define partial templates in _ files, and reference them in other chart templates
- Named template partials are globally accessible during chart install, so must be uniquely named
  - Can access child chart partials in a parent chart and vice versa

define

- The define action creates a named template

```
{{- define "my-chart.my-template" }}
  # template content
{{- end }}
```

```
# templates/_helpers.tpl

{{- define "codeveros-user-service.fullname" -}}
  {{- if .Values.fullnameOverride }}
    {{- .Values.fullnameOverride | trunc 63 | trimSuffix "-" }}
  {{- else }}
    {{- $name := default .Chart.Name .Values.nameOverride }}
    {{- if contains $name .Release.Name }}
      {{- .Release.Name | trunc 63 | trimSuffix "-" }}
    {{- else }}
      {{- printf "%s-%s" .Release.Name $name | trunc 63 | trimSuffix "-" }}
    {{- end }}
  {{- end }}
{{- end }}
```

template

- The template action is used to execute a named template
- Accepts a template name, and an optional pipeline used to define dot (.) in the template
- Used to inline a template into another template
- Creates reusable functionality

```
{{ template "name" pipeline }}
```

```
# templates/service.yaml
...
metadata:
  name: {{ template "codeveros-user-service.fullname" . }}
...
```

include

- template is a Go provided template action, its output cannot be chained together to form a larger pipeline
- Helm provides the include function to enable further processing of the named template output
- For example, you must use include instead of template in order to specify the indentation level of the output
- Preferable to use include over template

```
{{ include "name" pipeline }}
```

```
# templates/service.yaml
...
metadata:
  labels:
    {{- include "codeveros-user-service.labels" . | nindent 4 }}
...
```

block

- Go provided action that combines template definition (define) and execution (template) actions
- Allows developers to provide default implementation that is overridden later
- Not recommended for use in Helm charts, use include instead
  - If the same block is defined multiple times, the selected default implementation is not guaranteed

```
#templates/service.yaml
...
metadata:
  name: {{ block "user-service.fullname" . }}Default Name{{ end }}
...
```

Advanced Templating

Flow Control Actions

Flow Control Actions

- if / else
- range
- with

Flow Control - if / else

- Create conditional blocks that increase the chart's configurability
- Optional 'else' block and any number of 'else if' blocks
- If the conditional check evaluates to true, the block is executed
  - False is: False, 0, null, empty: arrays, slices, maps, or strings

```
{{- if .Values.ingress.enabled -}}
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  ...
spec:
  ...
{{- end -}}
```

```
{{- if PIPELINE -}}
T0
{{- else if OTHER_PIPELINE -}}
T1
{{- else -}}
T2
{{- end -}}
```

Flow Control - range

- Loops through the elements of a collection
- Inside the range block, dot (.) is set to the current element

```
# values.yaml
ingress:
  hosts:
    - host: foo.bar.com
      paths:
        - /foo
        - /bar
    - paths:
        - /
        - /baz
```

```
# templates/ingress.yaml
...
spec:
  rules:
    {{- range .Values.ingress.hosts }}
    - http:
        paths:
          {{- range .paths }}
          - path: {{ . }}
            backend:
              serviceName: my-service
              servicePort: 8080
          {{- end }}
        {{- if .host }}
        host: {{ .host | quote }}
        {{- end }}
     {{- end }}
```

Flow Control - range

- Can also loop over maps
- Assign the key and value to variables

```
# values.yaml
user:
  firstName: Bob
  lastName: Foster
  age: 62
  username: bfoster
```

```
# templates/configmap.yaml
...
data:
{{- range $key, $value := .Values.user }}
  {{ $key }}: {{ $value }}
{{- end }}
...
```

Flow Control - with

- Modifies dot's (.) scope
- Within the block, (.) refers to the with pipeline value
- Restricting scope prevents accessing higher-level objects
- If the with pipeline value is empty, nothing is executed

```
# values.yaml
car: Datsun
user:
  firstName: Bob
  lastName: Foster
  age: 62
  username: bfoster
```

```
{{ with PIPELINE }}
# execute this
{{ end }}
```

```
{{ /* INVALID */ }}
data:
  {{- with .Values.user }}
  firstName: {{ .firstName }}
  lastName: {{ .lastName }}
  car: {{ .Values.car }}
  {{- end }}
```

```
{{ /* VALID */ }}
data:
  {{- with .Values.user }}
  firstName: {{ .firstName }}
  lastName: {{ .lastName }}
  {{- end }}
  car: {{ .Values.car }}
```

Advanced Templating

What's Left?

What's Left?

- Template Variables
- Comments
- Accessing Files
- Whitespace Chomping
- Multi-line strings

Template Variables

- Named reference
- Scoped to block in which it is defined
- $ is a special global variable that references the root context
- Variables begin with $
- Variables initialized with := operator
- Initialized variables can be reassigned with = operator
- Useful in blocks of code that alter the dot (.) reference and require access to higher scoped objects
- Useful for defining once and using multiple times

Variable Examples

```
# templates/_helpers.tpl
{{- define "user-service.fullname" -}}
{{- $name := default .Chart.Name .Values.nameOverride }}
{{- printf "%s-%s" .Release.Name $name }}
{{- end }}
```

```
# templates/ingress.yaml
{{- $fullName := include "user-service.fullname" . -}}
{{- $svcPort := .Values.service.port -}}
...
metadata:
  name: {{ $fullName }}
spec:
  rules:
    {{- range .Values.ingress.hosts }}
    - host {{ .host | quote }}
      http:
        {{- range .paths }}
        paths: {{ . }}
        backend:
          serviceName: {{ $fullName }}
          servicePort: {{ $svcPort }}
...
```

Comments

- Template comments aren't rendered in final manifest files and should focus on documenting specific features with the Helm template

```
{{/* This is a template comment /*}}
```

- YAML comments are useful for viewing once the resource has been deployed to the cluster

```
# This is a YAML comment
```

Accessing Files

- Top-level .Files object with a number of different file accessor methods
- Used to import the contents of files and inject their literal value into templates without processing
- Accessed files must be stored outside templates folder
- `.Files.Get <path>` retrieves the contents of the file at <path>
- `.Files.Glob <pattern>` returns files that match a glob pattern
  - Useful for operating on multiple files
  - Map keyed by path, value is the file's byte content
- `.Files.Lines <path>` returns an array of the file lines at <path>
  - Useful for operating on individual lines
- There are a number of different path helper functions

Accessing Files Example

```
my-chart/
  Chart.yaml
  values.yaml
  templates/
    configmap.yaml
  files/
    file1.txt
    file2.txt
    file3.conf
    file4.conf
```

```
# templates/configmap.yaml

apiVersion: v1
kind: ConfigMap
metadata:
  name: my-chart-config
data:
  afile: |-
    # comment to ensure proper initial indent
    {{- .Files.Get "files/file1.txt" | nindent 4 }}
  {{- range $path, $_ := .Files.Glob "files/*.conf" }}
  a-{{ $path | base }}: |-
    # converting byte content
    {{- $_ | toString | nindent 4 }}
    # or using .File.Get on the path
    {{- $.Files.Get $path | nindent 4 }}
  {{- end }}
  {{- (.Files.Glob "files/*").AsConfig | nindent 2 }}
```

Whitespace Chomping

- Directionally removes whitespace
- Rendered manifest files replace template lines with whitespace
- Chomping is important for creating well-formed YAML
- {{- leftChomp }} removes whitespace to the left
- {{ rightChomp -}} removes whitespace to the right
- {{- doubleChomp -}} removes whitespace on both sides

Advanced String Usage

- Strings can be inline or span multiple lines
- Inline strings can be unquoted, single-quoted, or double-quoted
  - Use unquoted strings where possible
  - Use quotes to include certain characters and disambiguate strings from other types
- Multi-line strings denoted with (|) or (>)
  - (|) preserves line breaks and indentation
  - (>) folds content, removing line breaks
  - (-) strips final trailing newline using (|-) or (>-)
  - (+) retains all trailing whitespace (|+) or (>+)
  - Content must be correctly indented based on YAML rules
  - The first line of text enforces the left most indentation
  - Content not folded if it includes inner indentation

Advanced String Usage

```
# templates/configmap.yaml

data:
  unquoted: unquoted text block
  single: 'this: is a "single-quoted" block'
  double: "this: is a \"double-quoted\" block"
  number: '123'
  valid-indent: |
    this is a properly
      indented multi-line
  invalid-indent: |
      this is not properly
    indented, and will fail
  newline: |
    single trailing newline

  chomp: |-
    the - chomps trailing newline

  preserve: |+
    the + preserves whitespace

  folded: >
    this is folded,
    removing line-breaks,
    but keeping trailing \n
```

Exercise: Create a Codeveros Chart

Exercise Objectives

- Use **`helm create`** to generate a starter chart named codeveros-user-service
- Delete the chart files that are not required for the codeveros-user-service chart
- Modify the chart metadata, default values, and template files
- Install the chart and observe the results

Chart Creation

1. Use **`helm create`** to generate a new chart

```
$ helm create codeveros-user-service
```

2. Remove the following file (The codeveros-user-service will not make use of horizontal pod scaling)

```
$ rm codeveros-user-service/templates/hpa.yaml
```

```
helmuser@ip-172-31-1-140:~$ helm create codeveros-user-service
Creating codeveros-user-service
helmuser@ip-172-31-1-140:~$ rm codeveros-user-service/templates/hpa.yaml
```

Update Chart.yaml

3. Open the generated Chart.yaml file

```
$ nano codeveros-user-service/Chart.yaml
```

4. Update the appVersion to: `"1.1"`. (Sets the image tag).

```
  GNU nano 4.8                                codeveros-user-service/Chart.yaml
apiVersion: v2
name: codeveros-user-service
description: A Helm chart for Kubernetes

# A chart can be either an 'application' or a 'library' chart.
#
# Application charts are a collection of templates that can be packaged into versioned archives
# to be deployed.
#
# Library charts provide useful utilities or functions for the chart developer. They're included as
# a dependency of application charts to inject those utilities and functions into the rendering
# pipeline. Library charts do not define any templates and therefore cannot be deployed.
type: application

# This is the chart version. This version number should be incremented each time you make changes
# to the chart and its templates, including the app version.
# Versions are expected to follow Semantic Versioning (https://semver.org/)
version: 0.1.0

# This is the version number of the application being deployed. This version number should be
# incremented each time you make changes to the application. Versions are not expected to
# follow Semantic Versioning. They should reflect the version the application is using.
# It is recommended to use it with quotes.
appVersion: "1.1"
```

Update values.yaml

5. Open the generated values.yaml file

```
$ nano codeveros-user-service/values.yaml
```

6. Update the image repository to: `coveros/codeveros-user-service`

7. Update the service port to: `8080`

```yaml
replicaCount: 1

image:
  repository: coveros/codeveros-user-service
  pullPolicy: IfNotPresent
  # Overrides the image tag whose default is the chart appVersion.
  tag: ""

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

serviceAccount:
  # Specifies whether a service account should be created
  create: true
  # Annotations to add to the service account
  annotations: {}
  # The name of the service account to use.
  # If not set and create is true, a name is generated using the fullname template
  name: ""

podAnnotations: {}

podSecurityContext: {}
  # fsGroup: 2000

securityContext: {}
  # capabilities:
  #   drop:
  #   - ALL
  # readOnlyRootFilesystem: true
  # runAsNonRoot: true
  # runAsUser: 1000

service:
  type: ClusterIP
  port: 8080
```

Update Deployment Template

8. Open the generated deployment.yaml template

```
$ nano codeveros-user-service/templates/deployment.yaml
```

9. Update the containerPort to: `8080`

10. Update the the livenessProbe and readinessProbe path to: `/health-check`

```
          ports:
            - name: http
              containerPort: 8080
              protocol: TCP
          livenessProbe:
            httpGet:
              path: /health-check
              port: http
          readinessProbe:
            httpGet:
              path: /health-check
              port: http
```

Add MongoDB Dependency

11. The codeveros-user-service has a dependency on MongoDB and the chart needs to be updated accordingly

12. Open the generated Chart.yaml file

```
$ nano codeveros-user-service/Chart.yaml
```

13. Add the MongoDB dependency to the bottom of the file

```
dependencies:
  - name: mongodb
    version: 12.1.16
    repository: https://charts.bitnami.com/bitnami
```

Add MongoDB Values

14. Open the generated values.yaml file

```
$ nano codeveros-user-service/values.yaml
```

15. Add the following MongoDB values to bottom of the file

```
env:
  dbPort: 27017
  dbDatabase: users

mongodb:
  auth:
    enabled: false
  nameOverride: user-service-mongodb
```

Add Environment Variables

16. Open the generated deployment.yaml template

```
$ nano codeveros-user-service/templates/deployment.yaml
```

17. Add the following three environment variables to the container:
    - DB_HOST
    - DB_PORT
    - DB_DATABASE

```
          ports:
            - name: http
              containerPort: 8080
              protocol: TCP
          livenessProbe:
            httpGet:
              path: /health-check
              port: http
          readinessProbe:
            httpGet:
              path: /health-check
              port: http
          env:
            - name: DB_HOST
              value: "{{ .Release.Name }}-user-service-mongodb"
            - name: DB_PORT
              value: {{ .Values.env.dbPort | quote }}
            - name: DB_DATABASE
              value: {{ .Values.env.dbDatabase }}
```

Install the Chart

19. Install your new chart from its directory

```
$ helm install codeveros-user-service codeveros-user-service
```

20. An error should occur, as mentioned earlier in the tutorial, an installation will fail if any charts listed in the Chart.yaml dependencies section are not present in the charts directory

```
helmuser@ip-172-31-1-140:~$ helm install codeveros-user-service codeveros-user-service/
Error: found in Chart.yaml, but missing in charts/ directory: mongodb
```

Install Chart Dependencies

21. View a list of chart dependencies

```
$ helm dependency list codeveros-user-service/
```

22. Update the dependencies and view the list again

```
$ helm dependency update codeveros-user-service/
$ helm dependency list codeveros-user-service/
```

```
helmuser@ip-172-31-1-140:~$ helm dependency list codeveros-user-service/
NAME    VERSION REPOSITORY                                      STATUS
mongodb 10.6.1  https://charts.bitnami.com/bitnami      missing

helmuser@ip-172-31-1-140:~$ helm dependency update codeveros-user-service/
Getting updates for unmanaged Helm repositories...
...Successfully got an update from the "https://charts.bitnami.com/bitnami" chart repository
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "coveros" chart repository
...Successfully got an update from the "codeveros" chart repository
...Successfully got an update from the "jenkins" chart repository
Update Complete. □Happy Helming!□
Saving 1 charts
Downloading mongodb from repo https://charts.bitnami.com/bitnami
Deleting outdated charts
helmuser@ip-172-31-1-140:~$ helm dependency list codeveros-user-service/
NAME    VERSION REPOSITORY                                      STATUS
mongodb 10.6.1  https://charts.bitnami.com/bitnami      ok
```

Install the Chart

23. Install your new chart from its directory

```
$ helm install codeveros-user-service codeveros-user-service
```

24. Retrieve the Kubernetes Service Cluster IP

```
$ kubectl get svc codeveros-user-service -o jsonpath="{.spec.clusterIP}"
```

25. Use cURL to verify the release (should return `OK`)

```
$ curl <CLUSTER_IP>:8080/health-check
```

26. Using Lens, take some time and explore the resource that has been created

27. Uninstall the release when finished

```
$ helm uninstall codeveros-user-service
```

Chart Dependencies

Umbrella Charts

Chart Dependencies

- Charts can depend on other charts (aka subcharts or child charts)
- Dependencies are installed when a chart is installed
- A chart can override a subchart's default values
- Child templates are accessible by the parent chart
- Parent charts can import child chart values
- Parent charts can define global values
- Dependencies must be present prior to packaging and publishing a chart
- An umbrella chart is a chart composed of subcharts along with global configuration

Defining Dependencies

- Specified by the Chart.yaml dependencies field or manually added to charts directory
- Prefer defining dependencies in Chart.yaml
- Manually adding subcharts can be useful if the subchart is not used itself or by any other charts, not a typical use case

```
# Chart.yaml

...
dependencies:
  - name: codeveros-user-service
    version: 0.5.0
    repository: https://<chart-repo-url>
  - name: codeveros-training-service
    version: 0.5.0
    repository: "@codeveros"
```

```
my-chart/
  Chart.yaml
  charts/
    my-chart-1.0.0.tgz
    my-other-chart/
      Chart.yaml
      ...
```

Helm Client - helm dependency

- In order to install, update or package a chart, its declared dependencies must exist in the charts directory
- Use helm dependency to download the dependencies
- `helm dependency update` updates the downloaded charts in the charts folder based on the configured Chart.yaml dependencies field
  - Generates a Chart.lock file that tracks the charts installed
  - Doesn't affect manually added charts (not specified in Chart.yaml)
  - Declared dependencies present in the charts folder -- but whose version is different than specified in Chart.yaml -- are removed
- `helm dependency build` uses the Chart.lock file to rebuild the charts folder
  - If Chart.lock not present, operates the same as `helm dependency update`

Helm Client - helm dependency

```
$ ls codeveros/
Chart.yaml

$ helm install codeveros codeveros/
Error: found in Chart.yaml, but missing in charts/ directory: …

$ helm dependency update codeveros/

$ ls codeveros/
Chart.lock Chart.yaml charts

$ ls codeveros/charts/
codeveros-auth-service-0.5.0.tgz codeveros-gateway-0.5.0.tgz …

$ helm install codeveros codeveros/
NAME: codeveros
...
```

Overriding Children Values

- A parent's values.yaml overrides a subchart's value.yaml
- Referenced by the chart name or the alias (if defined)

```
# Chart.yaml
...
dependencies:
  - name: my-subchart
    version: 1.0.0
  - name: my-other-subchart
    version: 1.0.0
    alias: the-chart
```

```
# values.yaml

my-subchart:
  service:
    type: NodePort
    nodePort: 30005


the-chart:
  service:
    type: NodePort
    nodePort: 30004
```

```
# my-subchart
# values.yaml

service:
  type: ClusterIP
  port: 8080
...
```

Sharing Templates

- Defined templates are global and accessible by every chart
- Recommend prefixing template names using chart name in order to minimize the possibility of collisions between charts
- Named templates are not rendered at the time they are defined, they act more like functions

```
# my-chart templates/_helpers.tpl

{{- define "my-chart.chart" }}
{{ printf "%s-%s" .Chart.Name .Chart.Version }}
{{- end }}
```

```
# my-umbrella-chart templates/configmap.yaml

data:
  chartname: {{- include "my-chart.chart" . }}
```

What value is rendered in this configmap?

Sharing Values

- Subcharts can export values which are imported by parent charts
- Parent charts can import values not specified as an export
- Creates a stronger coupling between parent and child so use with caution

```
# my-chart Chart.yaml

dependencies:
  - name: my-subchart
    import-values:
      - toExport
      - child: toImport
        parent: imported
```

```
# my-subchart Chart.yaml

exports:
  toExport:
    aVal: one
    bVal: two
```

```
# my-subchart values.yaml

toImport:
  cVal: three
```

```
# my-chart values.yaml
imported:
  cVal: four
  dVal: five
```

```
# my-chart configmap.yaml

data:
  aVal: {{ .Values.aVal }}           # one
  bVal: {{ .Values.bVal }}           # two
  cVal: {{ .Values.imported.cVal }}  # three
  dVal: {{ .Values.imported.dVal }}  # five
```

Global Values

- Parent charts can define global values which are accessible by child charts
- Child chart global values are not accessible by a parent chart
- Parent chart globals override child chart globals
- Similar to importing child values, creates a tighter coupling
- Allows defining a common value once and having it propagated to all child charts

```
# my-chart values.yaml

global:
  name: My Chart
```

```
# Every child chart template now
# has access to this value

{{ .Values.global.name }}
```

Exercise: Create an Umbrella Chart

Exercise Slides

- Create Codeveros Umbrella chart
- Download dependencies
- Install Umbrella Chart
- Use Lens or Dashboard to view
- Browse to Codeveros and register a user, play around

Chart Creation

1. Create the codeveros umbrella chart folder structure

```
$ mkdir -p codeveros/charts
$ touch codeveros/Chart.yaml
$ touch codeveros/values.yaml
```

2. Verify the structure

```
$ ls codeveros
```

```
helmuser@ip-172-31-1-76:~$ mkdir -p codeveros/charts
helmuser@ip-172-31-1-76:~$ touch codeveros/Chart.yaml
helmuser@ip-172-31-1-76:~$ touch codeveros/values.yaml
helmuser@ip-172-31-1-76:~$ ls codeveros
Chart.yaml  charts  values.yaml
```

Create Chart.yaml

2. Create Chart.yaml with the sub-chart dependencies

```
$ nano codeveros/Chart.yaml
```

```
apiVersion: v2
name: codeveros
description: An umbrella chart for Codeveros
type: application
version: 1.0.0
appVersion: 1.0.0
dependencies:
  - name: codeveros-training-service
    version: ~0.5.0
    repository: https://coveros.github.io/codeveros
  - name: codeveros-auth-service
    version: ~0.5.0
    repository: https://coveros.github.io/codeveros
  - name: codeveros-gateway
    version: ~0.6.2
    repository: https://coveros.github.io/codeveros
  - name: codeveros-ui
    version: ~0.8.2
    repository: https://coveros.github.io/codeveros
```

***codeveros-user-service is excluded because we will use the chart we created in the previous exercise**

Create values.yaml

3. Enable the codeveros-ui and codeveros-gateway ingresses using values.yaml to override the sub-chart default values. The ingress is the Codeveros external entrypoint, making it accessible on port 80 and 443.

```
$ nano codeveros/values.yaml
```

```
  GNU nano 4.8                          codeveros/values.yaml
codeveros-ui:
  ingress:
    enabled: true

codeveros-gateway:
  ingress:
    enabled: true
```

Install the Chart

4. Install the umbrella chart, providing the release name and chart directory

```
$ helm install codeveros codeveros
```

5. An error should occur, as mentioned earlier in the tutorial, an installation will fail if any charts listed in the Chart.yaml dependencies section are not present in the charts directory

```
helmuser@ip-172-31-1-140:~$ helm install codeveros codeveros/
Error: found in Chart.yaml, but missing in charts/ directory: codeveros-training-service,
 codeveros-auth-service, codeveros-gateway, codeveros-ui
```

Install Chart Dependencies

6. View and install the chart dependencies

```
$ helm dependency list codeveros
$ helm dependency update codeveros
$ helm dependency list codeveros
```

```
helmuser@ip-172-31-1-76:~$ helm dependency list codeveros
NAME                            VERSION REPOSITORY                                      STATUS
codeveros-training-service      0.5.0   https://coveros.github.io/codeveros     missing
codeveros-auth-service          0.5.0   https://coveros.github.io/codeveros     missing
codeveros-gateway               0.5.0   https://coveros.github.io/codeveros     missing
codeveros-ui                    0.7.0   https://coveros.github.io/codeveros     missing

helmuser@ip-172-31-1-76:~$ helm dependency update codeveros
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "codeveros" chart repository
...Successfully got an update from the "coveros" chart repository
Update Complete. □Happy Helming!□
Saving 4 charts
Downloading codeveros-training-service from repo https://coveros.github.io/codeveros
Downloading codeveros-auth-service from repo https://coveros.github.io/codeveros
Downloading codeveros-gateway from repo https://coveros.github.io/codeveros
Downloading codeveros-ui from repo https://coveros.github.io/codeveros
Deleting outdated charts
helmuser@ip-172-31-1-76:~$ helm dependency list codeveros
NAME                            VERSION REPOSITORY                                      STATUS
codeveros-training-service      0.5.0   https://coveros.github.io/codeveros     ok
codeveros-auth-service          0.5.0   https://coveros.github.io/codeveros     ok
codeveros-gateway               0.5.0   https://coveros.github.io/codeveros     ok
codeveros-ui                    0.7.0   https://coveros.github.io/codeveros     ok
```

Add codeveros-user-service

7. Copy the codeveros-user-service chart to the codeveros charts directory

```
$ cp -r codeveros-user-service/ codeveros/charts/
```

8. Install the umbrella chart from the helm directory

```
$ helm install codeveros codeveros
```

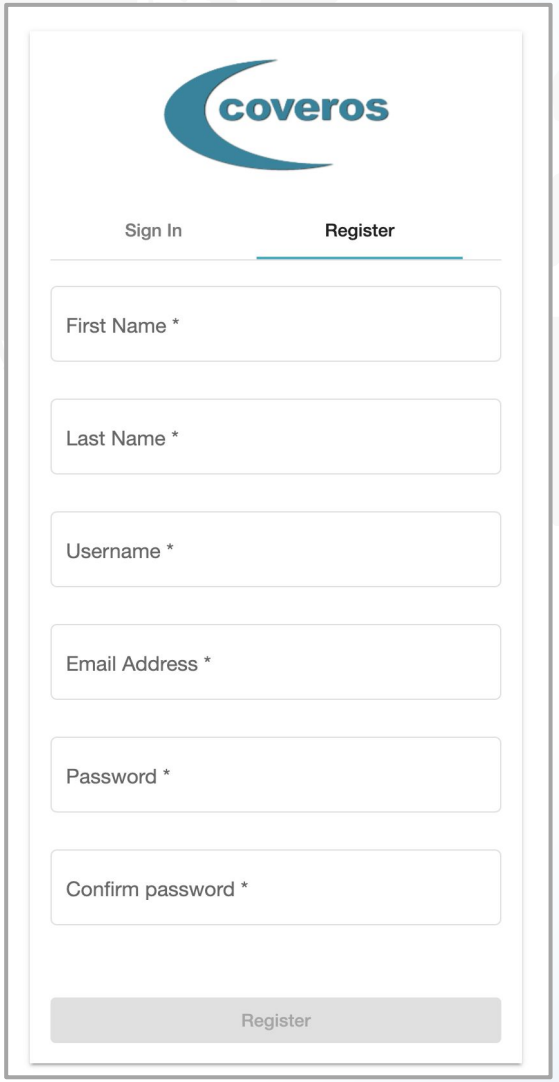
```
helmuser@ip-172-31-1-140:~$ cp -r codeveros-user-service/ codeveros/charts/
helmuser@ip-172-31-1-140:~$ helm install codeveros codeveros
NAME: codeveros
LAST DEPLOYED: Mon Feb  8 18:52:42 2021
NAMESPACE: default
STATUS: deployed
REVISION: 1
```

Browse to Codeveros

9. Browse to http://<EC2 IP>

10. Register a User, play around

11. Using Lens, take some time and explore the resources that have been created



Best Practices, Patterns, and Recommendations

General

- Only make configurable the things that need to be configurable
  - Can be added in later versions as needed
- Lint and test your charts
- Optional resources should be wrapped in conditional ‘enabled’ value checks
- Use .helmignore to blacklist files from the chart archive
- Organize files accessed by .Files into a files directory
- Define resources using the latest stable API version
- Ensure compatibility with the previous stable Kubernetes release

Formatting and Style

- Prefer YAML to JSON, prefer block format to inline format
- YAML should be indented with two spaces each level
- Include whitespace between template braces and content
  - {{ .Values.foo }}
- Chomp whitespace to render well-formatted resource manifests
- Can use nindent and chomping to enable both indented template directives and well-formed rendered manifests

Chart Metadata

- Chart Names should be composed of lowercase letters and numbers separated by dashes (e.g. my-chart)
- Chart directory name should match the chart name
- Every change to a chart should cause the version to be bumped
- Follow semantic versioning
  - Major.Minor.Patch-Status+Build
  - Status identifier and Build identifier are optional
  - Stable charts start at 1.0.0
- Provide as much Chart.yaml metadata as possible

Defining Values

- Lower camelcase naming (e.g. servicePort)
- Descriptively named to understand usage
- Prefer flat to nested structure
  - servicePort vs service.port
- Sufficiently comment values in values.yaml and in README.md
  - Describe usage and default values
- Consider how easily the values can be overridden
  - Maps can be customized more easily than arrays, since overriding an array requires defining the complete structure
- Define defaults in values.yaml rather than relying on default function

Using Images

- Define Image repository and tag as separately configurable fields
- Use specific image tags, don't use latest or leave undefined
- Set Chart.yaml appVersion to the image tag of the main process
- In the template where the image is specified, default the tag to AppVersion, but allow it to be overridden by a chart value
  - Default chart value to empty string
- Add imagePullPolicy as a configurable value, default to ifNotPresent

```
image: "{{ .Values.image.repository }}:{{ .Values.image.tag | default .Chart.AppVersion }}"
```

```
# values.yaml
image:
  repository: coveros/codeveros-user-service
  tag:
```

```
# Chart.yaml
name: codeveros-user-service
appVersion: 1.0.0
...
```

Create These Named Templates

At minimum, define these named templates in _helpers.tpl

- my-chart.name
  - Defaults to chart name, overridden by nameOverride value
- my-chart.fullname
  - Defaults to chart name or if chart name is not a substring of the release name, then defaults to a combination of release name and chart name, overridden by fullnameOverride
- my-chart.chart
  - Combines chart name and chart version
- my-chart.labels
  - Defines the common set of labels to apply to each resource
- my-chart.selectorLabels
  - Defines the subset of labels used for connecting services and pods

Structuring Templates

- Templates must be placed in the templates/ folder
- Use *.yaml file extension
  - Can use *.tpl for files that only define partials (e.g. _helpers.tpl)
- Use lower kabob case to name files (separate words with dashes)
- Use the resource type as the filename (e.g. configmap.yaml)
  - If multiple components, include in filename (e.g. configmap-agent.yaml)
- Define each resource in its own template file
- All defined templates should be namespaced with chart name to avoid conflicts

Templates

- Use "my-chart.fullname" to define resource names
  - If multiple resources of the same type are needed, add the component as a suffix (e.g. name: {{ include "my-chart.fullname" . }}-agent)
- Use include instead of template
- Don’t specify default values or make things configurable unnecessarily
- Use YAML comments and template comments appropriately
- Don’t specify the namespace
- Don’t use block

Dependencies

- Consider using version ranges
  - Semantic version ranges like ~1.1.0 or ^1.1.0
  - Chart.lock will peg the exact versions
- If using version ranges, commit Chart.lock to version control, and use `helm dependency build` during CI, to replicate local testing
- Use HTTPS repository URLs when possible
- Prefer repository URLs to repository aliases to limit dependency on external setup
- Track all dependencies in Chart.yaml to the extent possible
- Use conditions or tags to enable optional chart dependencies

Labels and Annotations

- Use labels to identify and classify resources
- Define standard labels as named templates
- Use annotations to add non-identifying metadata to resources
  - Commonly used for configuration (e.g. customizing an ingress controller or defining chart hooks)
  - Any other non-identifying relevant metadata
- If additional labels and annotations may be needed, add configurable values
- The different resource types should each have their own configurable label and annotation values
- Prefer defining default label and annotation values as empty maps
  - Depending on how they are used in templates, can optionally comment them out completely

Standard Labels

- Define the following standard labels ("my-chart.labels") and apply to each resource
  - app.kubernetes.io/name: {{ include "my-chart.name" . }}
  - app.kubernetes.io/instance: {{ .Release.Name }}
  - app.kubernetes.io/managed-by: {{ .Release.Service }}
  - helm.sh/chart: {{ include "my-chart.chart" . }}
  - app.kubernetes.io/version: {{ .Chart.AppVersion | quote }}
- Add app.kubernetes.io/component label if multiple components of the same resource type are needed
- Add app.kubernetes.io/part-of label If the helm chart is part of a larger whole
- Organize selector labels as template ("my-chart.selectorLabels")
  - app.kubernetes.io/name and app.kubernetes.io/instance, and app.kubernetes.io/component if defined
  - Used by Deployment matchLabels and Service selectors

Linting and Testing

- Lint and Test your charts
- `helm lint` provides simple, opinionated linting, verifies the chart files are well-formed
- Other linters: yamllint (code style), yamale (schema validator)
- `helm install` arguments `--debug` and `--dry-run` used to simulate installation and identify problems prior to attempting installation
- `helm template` used to output rendered resource to ensure template is configured correctly
- `helm test` executes chart tests against an installed release
  - Helm templates annotated with: `"helm.sh/hook": test`

Supporting Documentation

- README.md
  - Describe usage
  - Include table listing of all configuration values, including descriptions and default values
  - Include any relevant information needed for upgrading between versions
- CHANGELOG.md
  - Keep a running log of changes for each released version
- NOTES.txt
  - Provide accurate and useful usage and access instructions

Questions?