# TAKE HOME CHALLENGE

## Containerization, Deployment & Orchestrations of a Simple Ruby application

Submitted By,

**DELWIN NORONHA**

delwin.noronha1007@gmail.com

Date: 19 Dec 2023

## Contents

## CHALLENGE

Below, you will find the challenge(s) that we would like you to work on. We're looking forward to seeing your skills put into action, and the task will also offer you more insights on the role and industry that we operate in.

```
require 'socket'

server  = TCPServer.new('0.0.0.0', 80)

loop {
  client  = server.accept
  request = client.readpartial(2048)

  method, path, version = request.lines[0].split

  puts "#{method} #{path} #{version}"

  if path == "/healthcheck"
    client.write("OK")
  else
    client.write("Well, hello there!")
  end

  client.close
}
```

https://github.com/sawasy/http_server/blob/main/http_server.rb

---

## Overview

Given the provided Ruby app, design a pipeline and deployment strategyfollowing the requirements illustrated below.

## Modules

1. Pipeline design
2. Deploy
3. Improvements

### Module 1 - Pipeline design

We have a homogenous mixture of environments based on cloud as well ason-premise Kubernetes clusters. Some of our applications run on multiple instances while others are standalone. The configuration of the auxiliary services (databases, caches etc.) might be different depending on the environment.
Design a pipeline that fits the needs of such an infrastructure for our app. (GitHub actions, helm, Argo CD are preferred).

### Module 2 - Deploy

Run the app on local k8s cluster.

#### Part 1 - Containerize

Build a secure, scalable and robust container image.

#### Part 2 - Manifests

Write helm manifests to deploy the application to local Kubernetes cluster.
Should satisfy following:

- Highly available and load balanced
- Ensuring the application is started before served with traffic
- Safeguards for ensuring healthy life cycle of applications
- Ensure zero downtime
- Endpoints of the web application are accessible outside the cluster

### Module 3 - Improvements

This is an open-ended assignment, you are free to introduce changes, in the assigned time frame, to the application or in the instrumentation to meet your standards in terms of security, availability, reliability and observability.

## APPROACH AND DESIGN CONSIDERATIONS

The below considerations were taken to satisfy the mentioned requirements.

1) Containerize the application – This can be achieved using **Docker**. Below best practices are followed
    a. Use smallest possible base image / multistage build – to reduce image size + security
    b. Prevent root access in container - Security
    c. Order commands in Dockerfile to obtain highest level of caching – this will reduce build time and image size.
2) Manifests – **Helm** is used to manifest the deployments to Kubernetes cluster
3) Auxiliary services configuration – Can be achieved using Helm values. We can have different **helm values files** for different environment / applications. This will allow to configure different environment specific values like
    a. Database hosts
    b. Cache details
    c. Any other configurable endpoints, etc.

    We can also make use of **secret management** tools like Vault, Secret Manager, GitHub secrets etc. to store and retrieve environment specific sensitive data like passwords.
4) Deploy to multiple Clusters – We can use **Argo CD** tool to manage multi-cluster deployments (both cloud and on-prem). This will follow GitOps and make it very easy to deploy applications to multiple clusters with least effort.
5) Local deployment – **K3D** Kubernetes is used. It has its own local container registry as well, where images can be imported. Docker in docker concept.
6) Load balancing – Using K3D provided **Traefik** LB.
7) CICD Pipeline – **Github actions** is used.
8) For high availability – **Autocscalers**, multi pods, affinity rules. (additionally – regional clusters, capacity planning, logging, monitoring, alerting etc)
9) **Readiness and Liveliness probes** – Used to make sure application is started before serving traffic, and ensure application health is regularly checked.
10) Zero downtime – Achieved using **rolling updates** and multiple pod replicas**.**
11) Application endpoints accessible outside cluster – Using **loadbalancer** service, ingress
12) TLS is not used, as this is a local setup. It is always recommended to used TLS encryption for production workloads.
13) This application is currently tested only in Linux / amd64 machine. Docker base image would require to be changed to support other architectures.
14) The application deployment is manifested with helm using minimal required values.
15) Applications will run on local host network. (localhost – 127.0.0.1)
16) ArgoCD UI will be exposed using port forwarding, as this is local setup.
17) Argo CD sync time has been set to 5s, it requires 1 time login.
18) The setup is limited with features required only for the challenge requirements.
19) Have fun implementing this setup!!

Additional Improvements / Considerations

1) **Security** – Enable RBACs in clusters, Use secret management tools, code scanning, proper auth processes, TLS, cert-manager.
2) **Reliability** – Logging, monitoring, alerting, tracing, auto scaling
3) **Observability** – We can use tools like Prometheus, Grafana, Jaeger for monitoring, alerting and tracing. This won't be done for this challenge, considering the limited time given.
4) **Availability** – Multi region deployments, disaster recovery plans, capacity planning

## SETUP

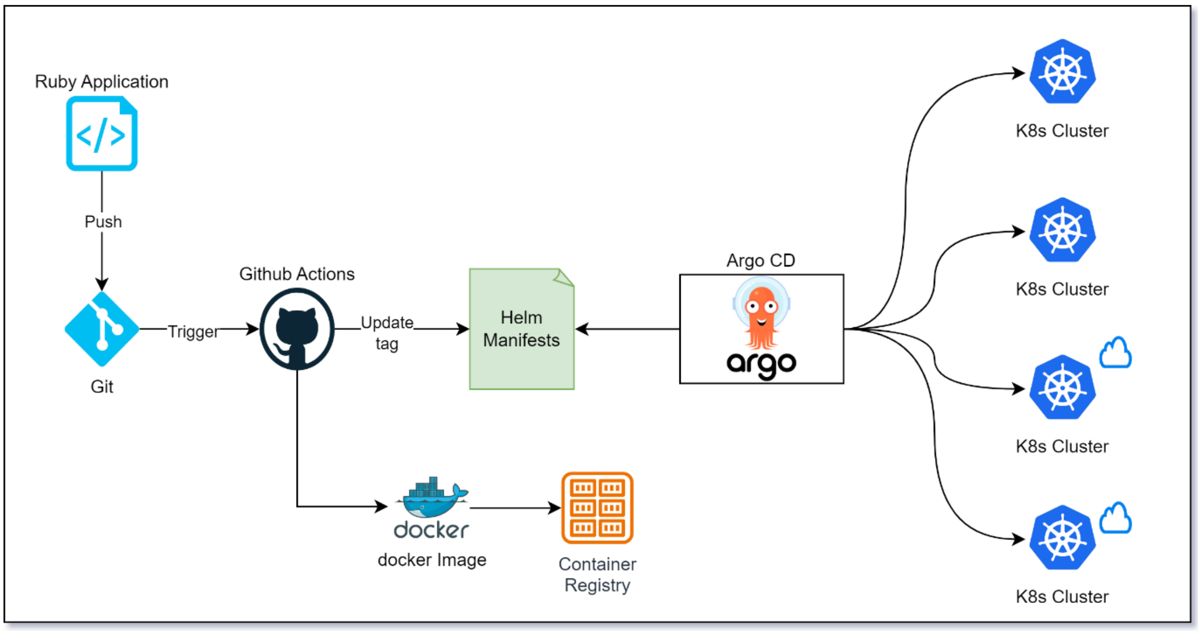
The below tools will be used

1) **Docker** – For containerization
2) **K3D** – To deploy K8s cluster locally and for local container registry.
3) **Helm** – To manifest Kubernetes deployments
4) **Argo CD** – For Kubernetes deployments

To install the setup, follow instructions mentioned in **README** attached with the code.

Detailed installation guide with screenshots provided in this document.

Below is a high-level diagram of this setup.



## PRE-REQUISITES

Make sure below tools are present in the system,

- Kubectl - to interact to Kubernetes cluster. https://kubernetes.io/docs/tasks/tools/
- Docker engine
- Bash
- Helm - https://helm.sh/docs/intro/install/
- Argo CD CLI (optional)
- K3D – To deploy K8s cluster locally and for local container registry - https://k3d.io/v5.6.0/#installation

Internet connection is required to download the necessary docker images and dependencies.

** Note: If this setup is being run behind a proxy, please make sure the necessary proxy configuration is done in the system.

## INSTALLATION

1) Clone / Download the GitHub repo to your local - [https://github.com/dnoronh/ruby-app-devops.git](https://github.com/dnoronh/ruby-app-devops.git)

2) In bash, run the install_setup.sh shell script using command

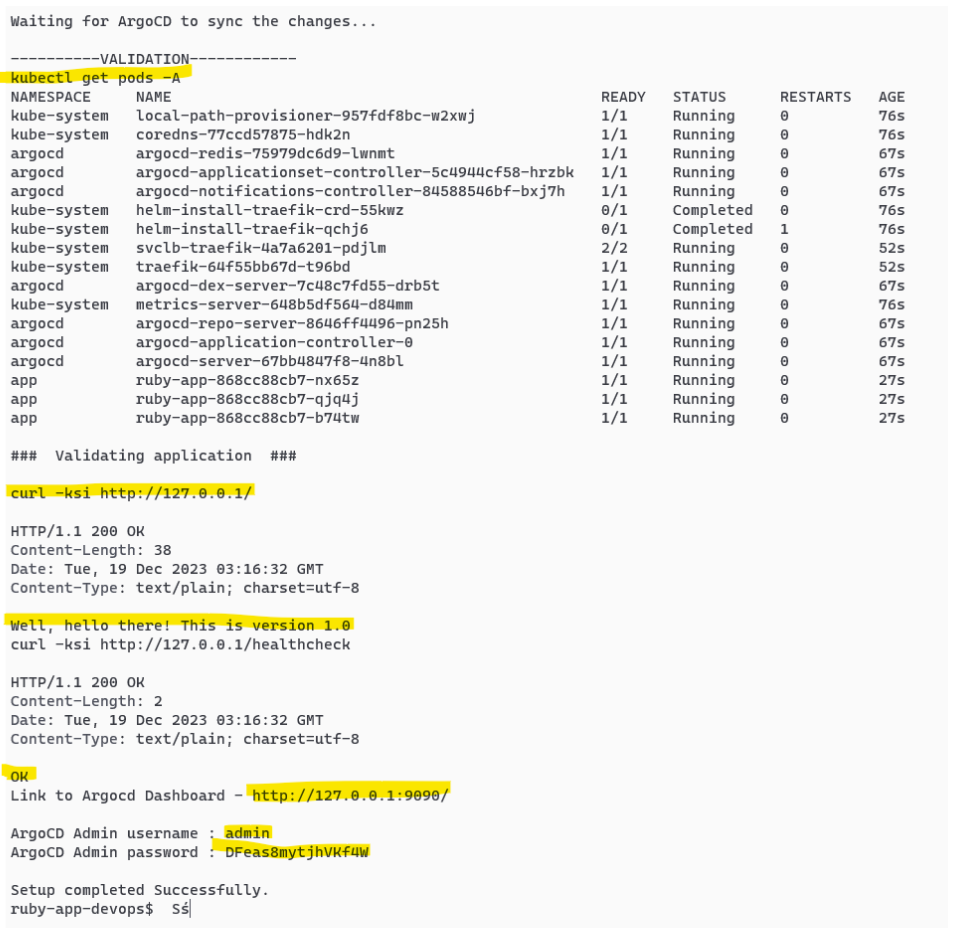
   ```
   ./install_setup.sh
   ```

   ```
   ruby-app-devops$ ./install_setup.sh
   ```

   The script will perform the below actions

   1) Create K3D cluster
   2) Create docker image 1.0 for ruby app
   3) Create docker image 2.0 for ruby app
   4) Deploy Argo CD tool using helm charts
   5) Create Argo CD application for ruby app
   6) Port forwarding of Argo CD service.

This below can be seen post installation



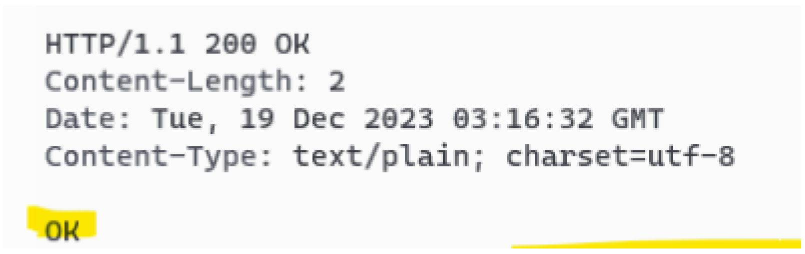
```
Waiting for ArgoCD to sync the changes...

----------VALIDATION------------
kubectl get pods -A
NAMESPACE     NAME                                               READY   STATUS      RESTARTS   AGE
kube-system   local-path-provisioner-957fdf8bc-w2xwj             1/1     Running     0          76s
kube-system   coredns-77ccd57875-hdk2n                           1/1     Running     0          76s
argocd        argocd-redis-75979dc6d9-lwnmt                      1/1     Running     0          67s
argocd        argocd-applicationset-controller-5c4944cf58-hrzbk  1/1     Running     0          67s
argocd        argocd-notifications-controller-84588546bf-bxj7h   1/1     Running     0          67s
kube-system   helm-install-traefik-crd-55kwz                     0/1     Completed   0          76s
kube-system   helm-install-traefik-qchj6                         0/1     Completed   1          76s
kube-system   svclb-traefik-4a7a6201-pdjlm                       2/2     Running     0          52s
kube-system   traefik-64f55bb67d-t96bd                           1/1     Running     0          52s
argocd        argocd-dex-server-7c48c7fd55-drb5t                 1/1     Running     0          67s
kube-system   metrics-server-648b5df564-d84mm                    1/1     Running     0          76s
argocd        argocd-repo-server-8646ff4496-pn25h                1/1     Running     0          67s
argocd        argocd-application-controller-0                    1/1     Running     0          67s
argocd        argocd-server-67bb4847f8-4n8bl                     1/1     Running     0          67s
app           ruby-app-868cc88cb7-nx65z                          1/1     Running     0          27s
app           ruby-app-868cc88cb7-qjq4j                          1/1     Running     0          27s
app           ruby-app-868cc88cb7-b74tw                          1/1     Running     0          27s

###  Validating application  ###

curl -ksi http://127.0.0.1/

HTTP/1.1 200 OK
Content-Length: 38
Date: Tue, 19 Dec 2023 03:16:32 GMT
Content-Type: text/plain; charset=utf-8

Well, hello there! This is version 1.0
curl -ksi http://127.0.0.1/healthcheck

HTTP/1.1 200 OK
Content-Length: 2
Date: Tue, 19 Dec 2023 03:16:32 GMT
Content-Type: text/plain; charset=utf-8

OK
Link to Argocd Dashboard - http://127.0.0.1:9090/

ArgoCD Admin username : admin
ArgoCD Admin password : DFeas8mytjhVKf4W

Setup completed Successfully.
ruby-app-devops$  Sś
```

Please use the mentioned Argo CD **username** and **password** for first time login to UI

## VALIDATION

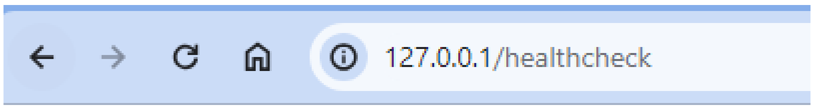
1) It will take around 1 min for the setup to complete. Once completed please validate if pods are up and running.
   **`kubectl get pods -A`**

2) Validate if able to reach the ruby application
   **`curl -kis http://127.0.0.1/healthcheck`**

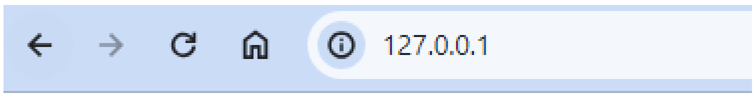
   

   ```
   HTTP/1.1 200 OK
   Content-Length: 2
   Date: Tue, 19 Dec 2023 03:16:32 GMT
   Content-Type: text/plain; charset=utf-8

   OK
   ```

   We can validate the same in bowser as well



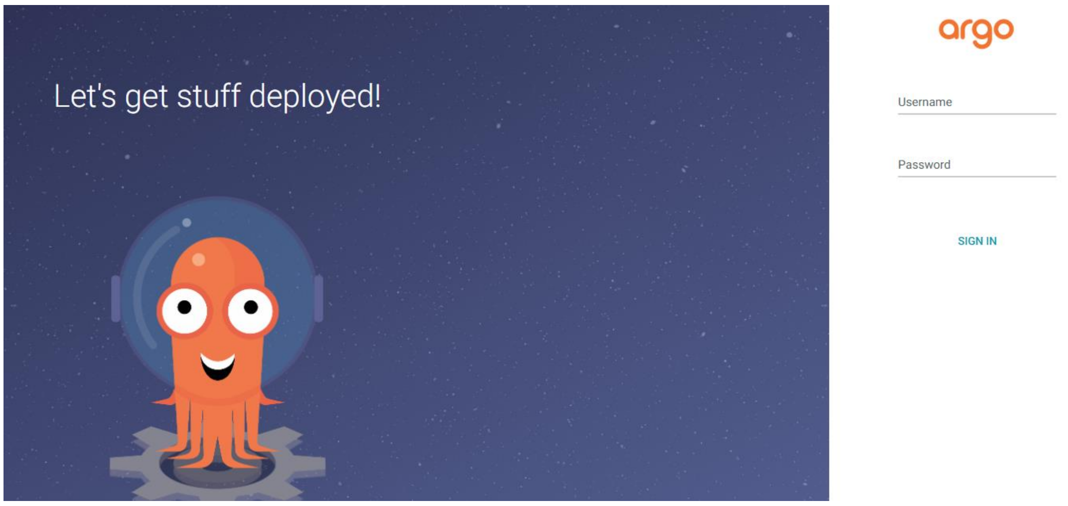
```
OK
```



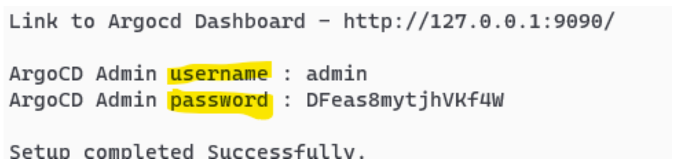
```
Well, hello there! This is version 1.0
```

3) Validate if you are able to open Argo CD dashboard using below link.

http://127.0.0.1:9090/



Enter the username password provided in the installation output logs



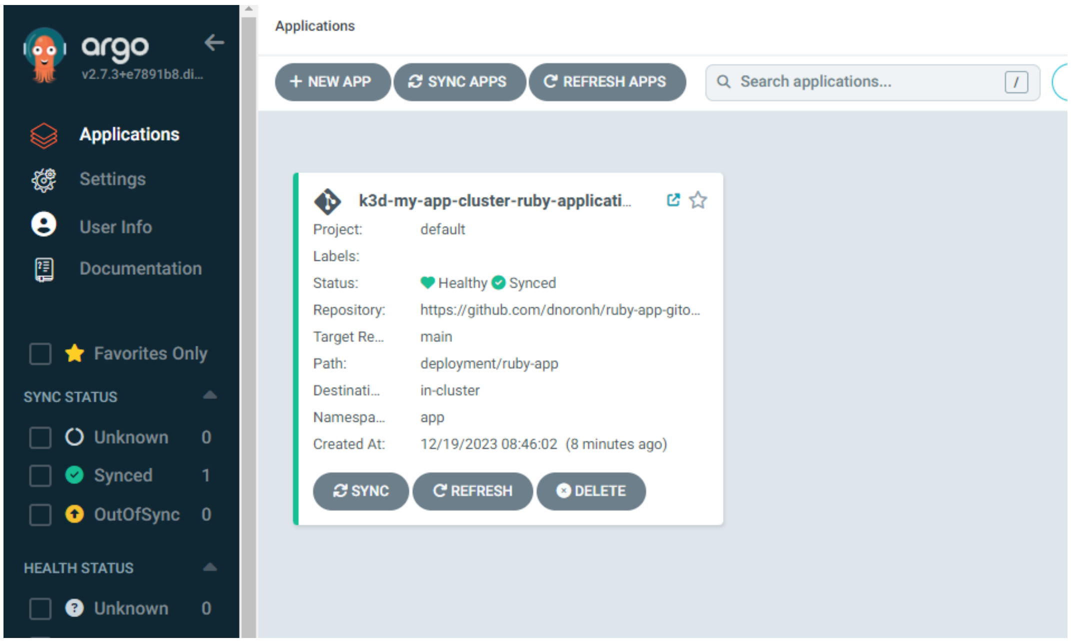
```
Link to Argocd Dashboard - http://127.0.0.1:9090/

ArgoCD Admin username : admin
ArgoCD Admin password : DFeas8mytjhVKf4W

Setup completed Successfully.
```

If you are not able to open the UI, please validate if port forward is enabled

Command **ps -f | grep port-forward**

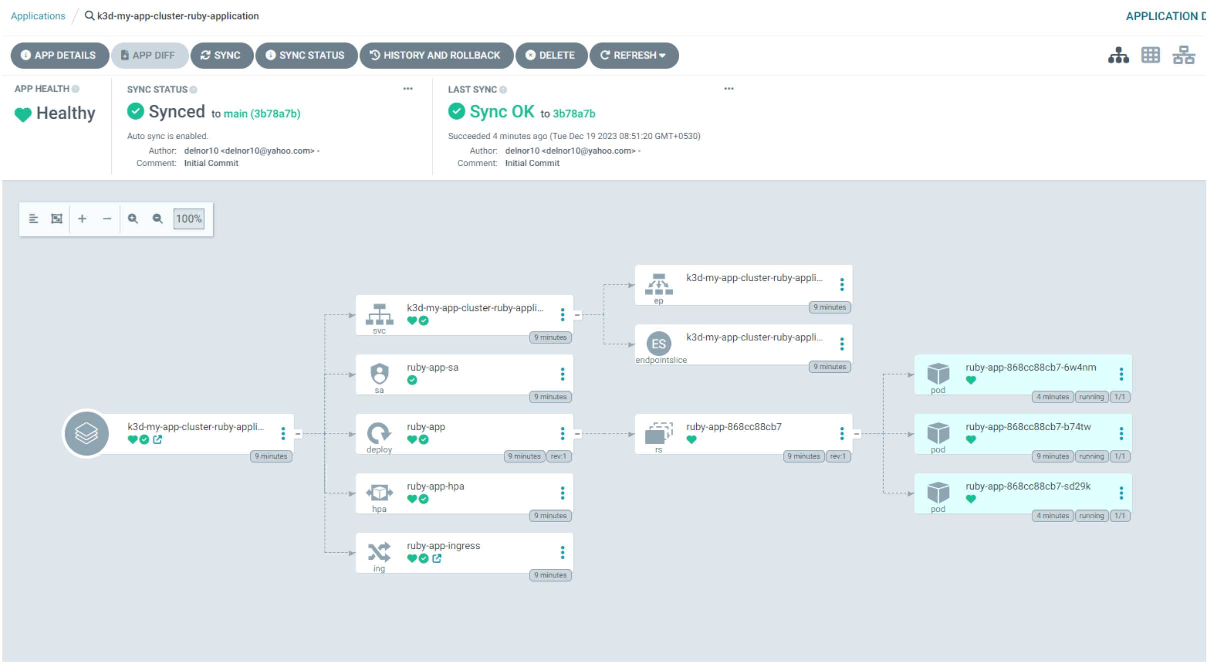
Run command below command to enable port forwarding if not already enabled

```
kubectl port-forward svc/argocd-server -n argocd 9090:80 > /dev/null &
```



k3d-my-app-cluster-ruby-applicati...

| | |
|---|---|
| Project: | default |
| Labels: | |
| Status: | Healthy Synced |
| Repository: | https://github.com/dnoronh/ruby-app-gito... |
| Target Re... | main |
| Path: | deployment/ruby-app |
| Destinati... | in-cluster |
| Namespa... | app |
| Created At: | 12/19/2023 08:46:02 (8 minutes ago) |

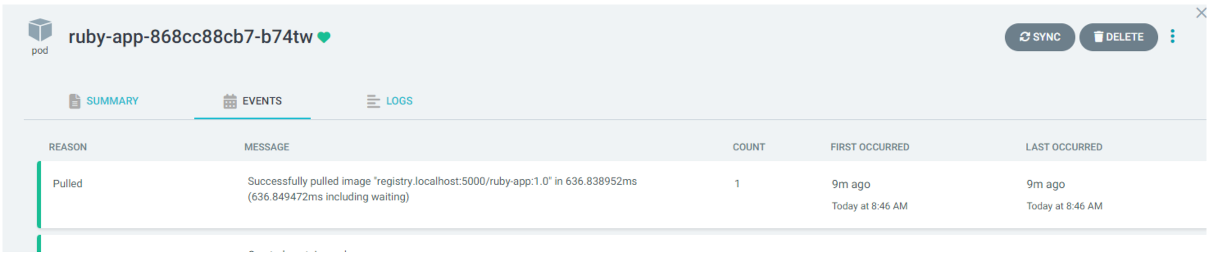

Applications
k3d-my-app-cluster-ruby-application
APPLICATION D
APP DETAILS
APP DIFF
SYNC
SYNC STATUS
HISTORY AND ROLLBACK
DELETE
REFRESH
100%
k3d-my-app-cluster-ruby-appli...
k3d-my-app-cluster-ruby-appli...
svc
ruby-app-sa
sa
ruby-app
deploy
ruby-app-hpa
hpa
ruby-app-ingress
ing
k3d-my-app-cluster-ruby-appli...
ep
k3d-my-app-cluster-ruby-appli...
endpointslice
ruby-app-868cc88cb7
rs
ruby-app-868cc88cb7-6w4nm
ruby-app-868cc88cb7-b74tw
ruby-app-868cc88cb7-sd29k
pod
9 minutes
4 minutes
running
1/1
rev:1

APP HEALTH
Healthy

SYNC STATUS
Synced to main (3b78a7b)
Auto sync is enabled.
Author: delnor10 <delnor10@yahoo.com> -
Comment: Initial Commit

LAST SYNC
Sync OK to 3b78a7b
Succeeded 4 minutes ago (Tue Dec 19 2023 08:51:20 GMT+0530)
Author: delnor10 <delnor10@yahoo.com> -
Comment: Initial Commit

We can see that image version '1.0' is deployed to Kubernetes via Argo CD



ruby-app-868cc88cb7-b74tw

| REASON | MESSAGE | COUNT | FIRST OCCURRED | LAST OCCURRED |
|---|---|---|---|---|
| Pulled | Successfully pulled image "registry.localhost:5000/ruby-app:1.0" in 636.838952ms (636.849472ms including waiting) | 1 | 9m ago Today at 8:46 AM | 9m ago Today at 8:46 AM |

As best practices for GitOps, we have the manifests in a different Gihub Repo. Argo CD will be monitoring this repository. - https://github.com/dnoronh/ruby-app-gitops-repo

## DEPLOY NEW IMAGE – The GitOps Way

To perform the build and deploy of version 2.0 of the app, we will use **GitHub Actions** workflow.

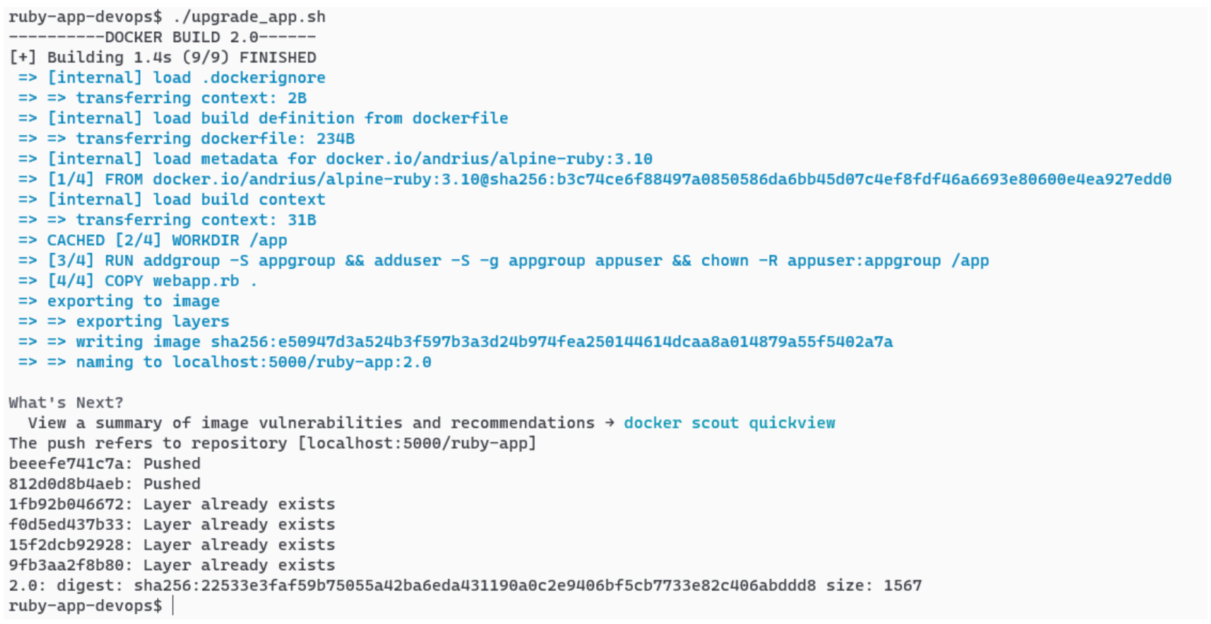
GitHub actions pipeline will perform the below steps

1) Build the docker image
2) Tag and push the image to registry (this step would be done manually in our case, as we are using a local registry)
3) Update the image tag in manifest (helm values) – Ideally it is better to use short **GIT_SHA** for the tags, but in this demo, we will use the tag '2.0' for the newer version.

Argo CD will sync the changes to the cluster and deploy the new version 2.0

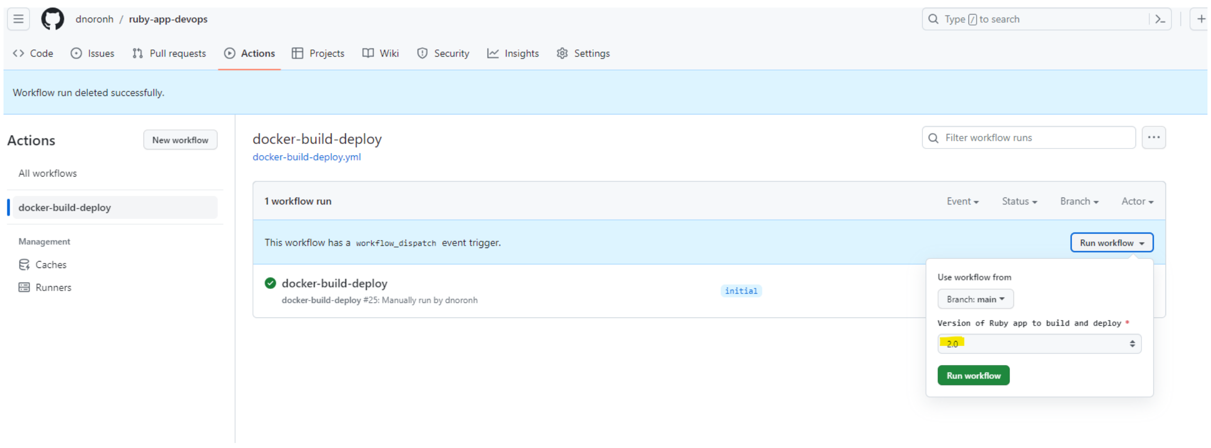
To demonstrate this in our local cluster,

In our case, image 2.0 version is already built during installation. We just need to run the Github actions pipeline to update the tag in helm values.



```
ruby-app-devops$ ./upgrade_app.sh
----------DOCKER BUILD 2.0------
[+] Building 1.4s (9/9) FINISHED
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load build definition from dockerfile
 => => transferring dockerfile: 234B
 => [internal] load metadata for docker.io/andrius/alpine-ruby:3.10
 => [1/4] FROM docker.io/andrius/alpine-ruby:3.10@sha256:b3c74ce6f88497a0850586da6bb45d07c4ef8fdf46a6693e80600e4ea927edd0
 => [internal] load build context
 => => transferring context: 31B
 => CACHED [2/4] WORKDIR /app
 => [3/4] RUN addgroup -S appgroup && adduser -S -g appgroup appuser && chown -R appuser:appgroup /app
 => [4/4] COPY webapp.rb .
 => exporting to image
 => => exporting layers
 => => writing image sha256:e50947d3a524b3f597b3a3d24b974fea250144614dcaa8a014879a55f5402a7a
 => => naming to localhost:5000/ruby-app:2.0

What's Next?
  View a summary of image vulnerabilities and recommendations → docker scout quickview
The push refers to repository [localhost:5000/ruby-app]
beeefe741c7a: Pushed
812d0d8b4aeb: Pushed
1fb92b046672: Layer already exists
f0d5ed437b33: Layer already exists
15f2dcb92928: Layer already exists
9fb3aa2f8b80: Layer already exists
2.0: digest: sha256:22533e3faf59b75055a42ba6eda431190a0c2e9406bf5cb7733e82c406abddd8 size: 1567
ruby-app-devops$
```

Run the GitHub actions workflow – https://github.com/dnoronh/ruby-app-devops/actions/workflows/docker-build-deploy.yml


dnoronh / ruby-app-devops

Workflow run deleted successfully.

docker-build-deploy
docker-build-deploy.yml

1 workflow run

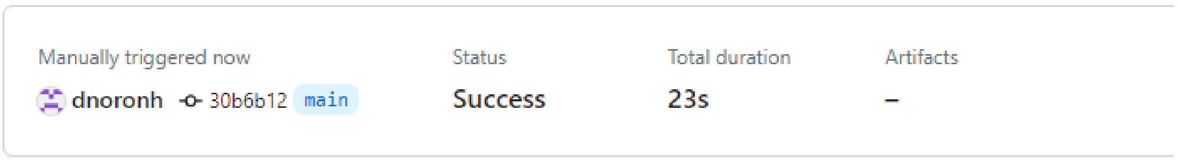
This workflow has a `workflow_dispatch` event trigger.

docker-build-deploy
docker-build-deploy #25: Manually run by dnoronh `initial`

Use workflow from
Branch: main

Version of Ruby app to build and deploy *
2.0

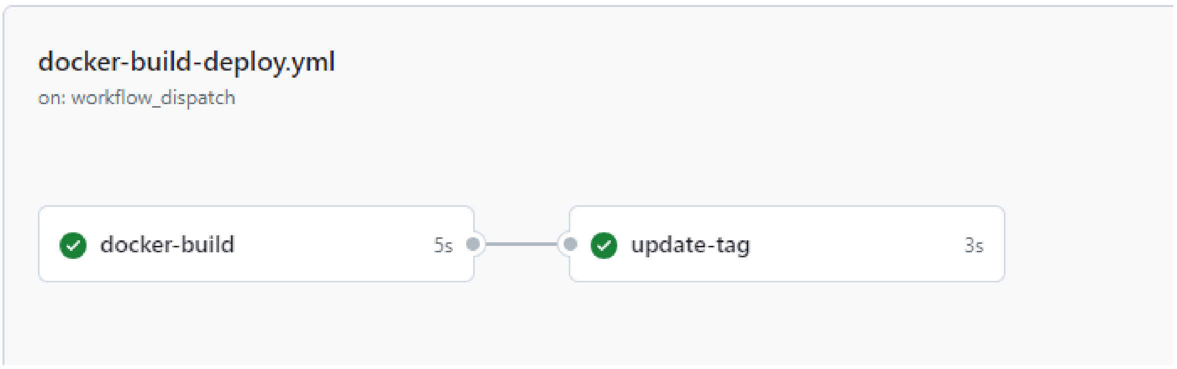
This workflow will perform a dummy docker build, and update the image version in manifest repo to 2.0


Manually triggered now
dnoronh 30b6b12 main

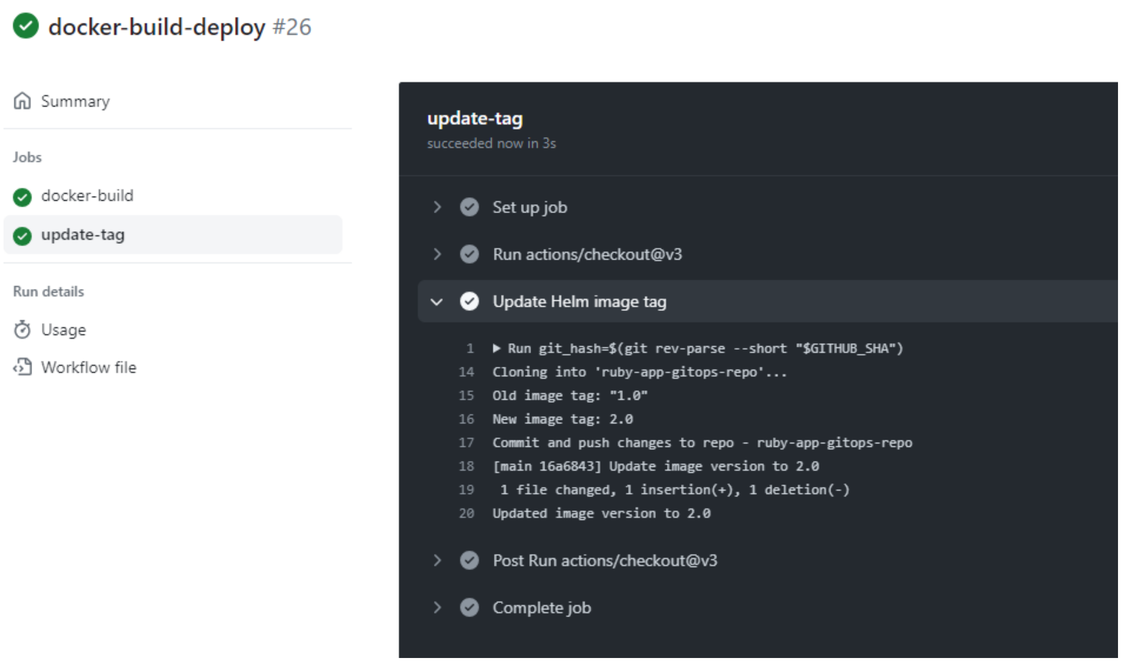
Status
Success

Total duration
23s

Artifacts
–


docker-build-deploy.yml
on: workflow_dispatch

docker-build 5s
update-tag 3s


docker-build-deploy #26

update-tag
succeeded now in 3s

Set up job
Run actions/checkout@v3
Update Helm image tag

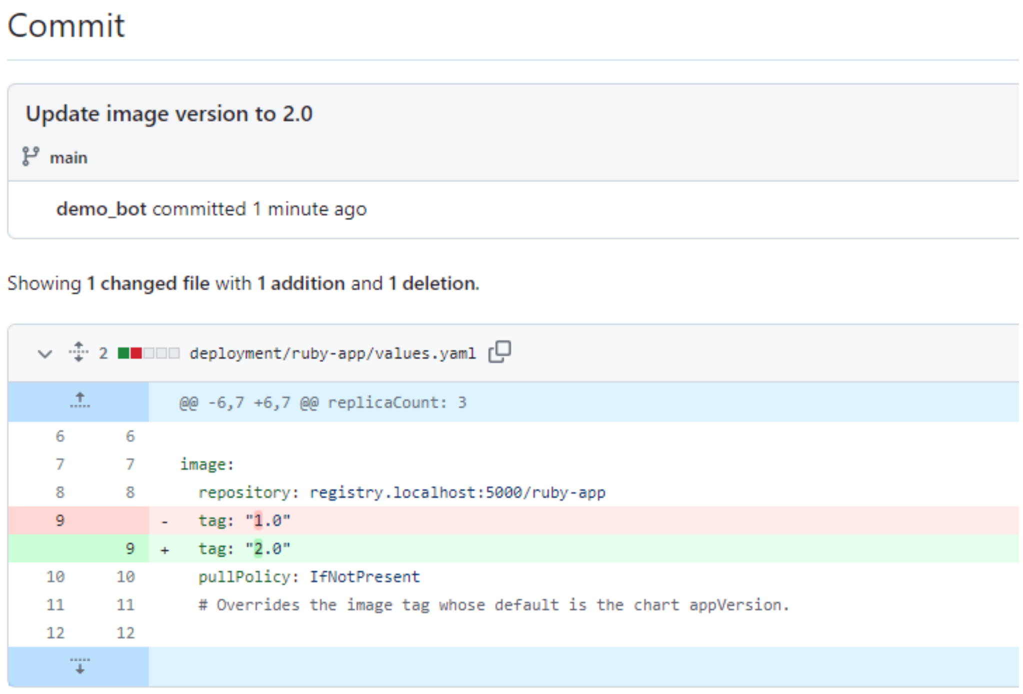
```
▶ Run git_hash=$(git rev-parse --short "$GITHUB_SHA")
Cloning into 'ruby-app-gitops-repo'...
Old image tag: "1.0"
New image tag: 2.0
Commit and push changes to repo - ruby-app-gitops-repo
[main 16a6843] Update image version to 2.0
 1 file changed, 1 insertion(+), 1 deletion(-)
Updated image version to 2.0
```

Post Run actions/checkout@v3
Complete job

Let’s validate the changes done

Go to https://github.com/dnoronh/ruby-app-gitops-repo

View the latest commit done



## Commit

**Update image version to 2.0**

**main**

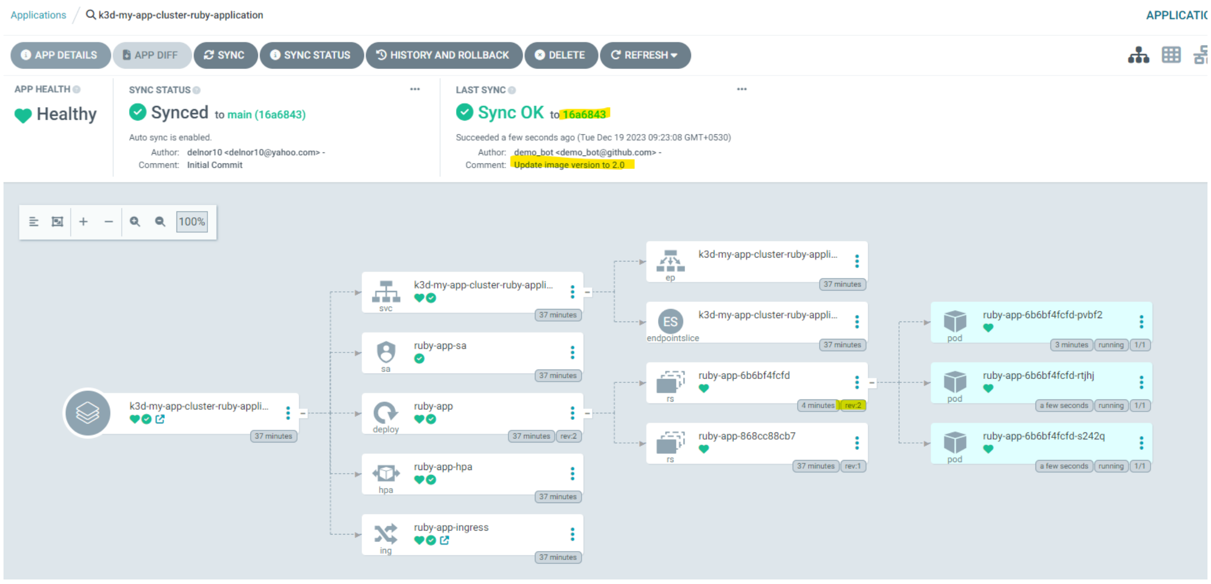
**demo_bot** committed 1 minute ago

Showing **1 changed file** with **1 addition** and **1 deletion.**

2 🟩🟥⬜⬜⬜ `deployment/ruby-app/values.yaml`

```
@@ -6,7 +6,7 @@ replicaCount: 3

 image:
   repository: registry.localhost:5000/ruby-app
-  tag: "1.0"
+  tag: "2.0"
   pullPolicy: IfNotPresent
   # Overrides the image tag whose default is the chart appVersion.

```

Now observe the changes in Argo CD or by sending a request to the app



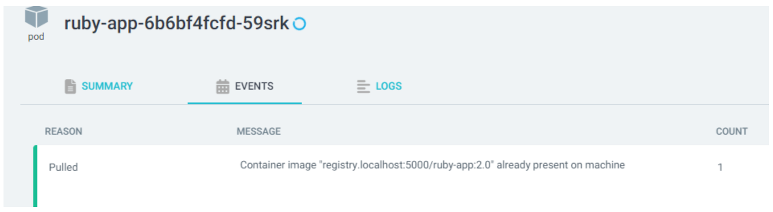
Applications / k3d-my-app-cluster-ruby-application

APP HEALTH
Healthy

SYNC STATUS
Synced to main (16a6843)
Auto sync is enabled.
Author: delnor10 <delnor10@yahoo.com> -
Comment: Initial Commit

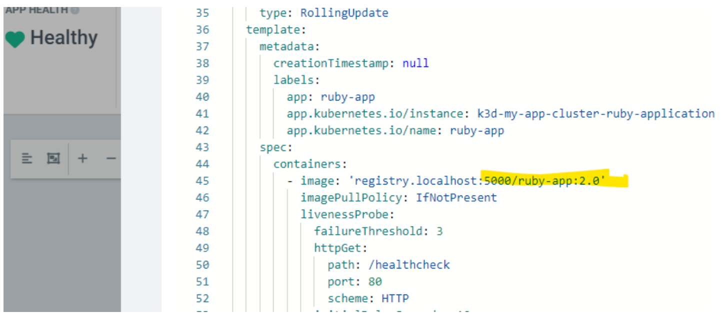
LAST SYNC
Sync OK to 16a6843
Succeeded a few seconds ago (Tue Dec 19 2023 09:23:08 GMT+0530)
Author: demo_bot <demo_bot@github.com> -
Comment: Update image version to 2.0

You can observe now that version 2.0 of the app is deployed



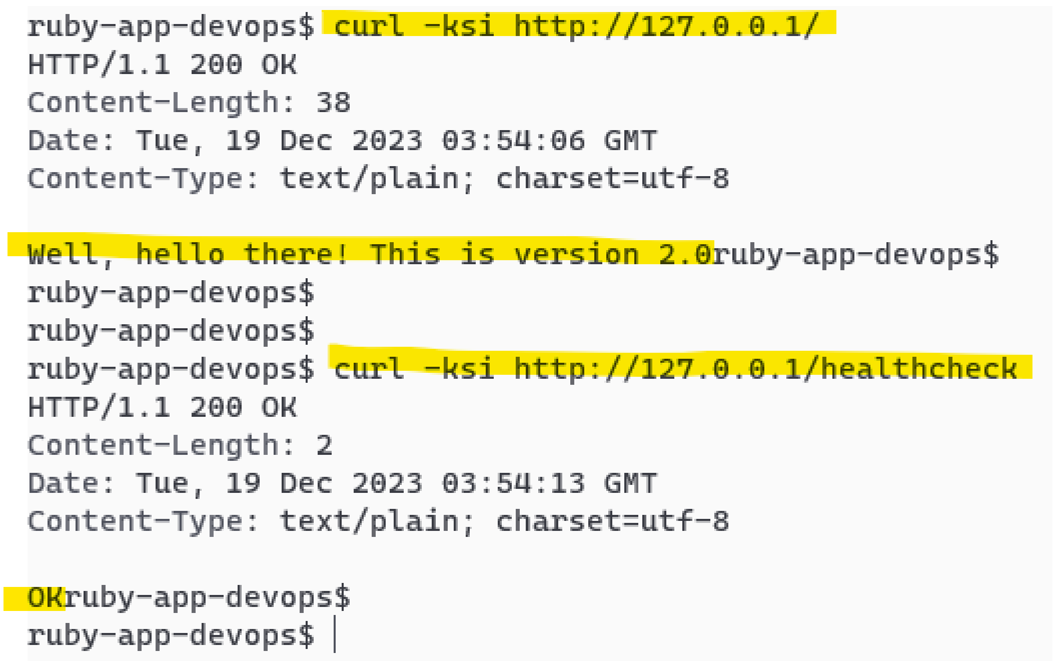
ruby-app-6b6bf4fcfd-59srk

| REASON | MESSAGE | COUNT |
|---|---|---|
| Pulled | Container image "registry.localhost:5000/ruby-app:2.0" already present on machine | 1 |



```
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: ruby-app
        app.kubernetes.io/instance: k3d-my-app-cluster-ruby-application
        app.kubernetes.io/name: ruby-app
    spec:
      containers:
        - image: 'registry.localhost:5000/ruby-app:2.0'
          imagePullPolicy: IfNotPresent
          livenessProbe:
            failureThreshold: 3
            httpGet:
              path: /healthcheck
              port: 80
              scheme: HTTP
```

Validate the application endpoints



```
ruby-app-devops$ curl -ksi http://127.0.0.1/
HTTP/1.1 200 OK
Content-Length: 38
Date: Tue, 19 Dec 2023 03:54:06 GMT
Content-Type: text/plain; charset=utf-8

Well, hello there! This is version 2.0ruby-app-devops$
ruby-app-devops$
ruby-app-devops$
ruby-app-devops$ curl -ksi http://127.0.0.1/healthcheck
HTTP/1.1 200 OK
Content-Length: 2
Date: Tue, 19 Dec 2023 03:54:13 GMT
Content-Type: text/plain; charset=utf-8

OKruby-app-devops$
ruby-app-devops$ |
```

We can observe that Argo CD has deployed the latest version of the image and it is running successfully.

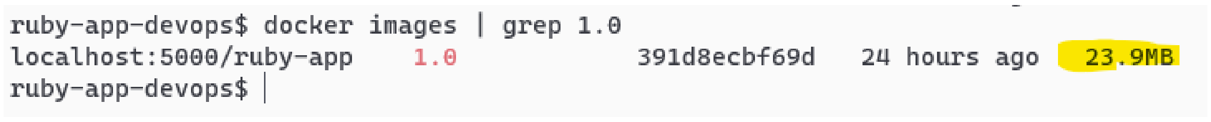
## CLEANUP

To cleanup the setup, run the below command

```
./install_setup.sh cleanup
```

## RESULTS

1) **Containerization** – The docker image for the Ruby app is built following industry best practices to achieve lower image sizes and best security. Additionally, non-root access is provided to the container. We can see that the final image is only 24MB. This is a huge advantage when it comes to microservices architecture.



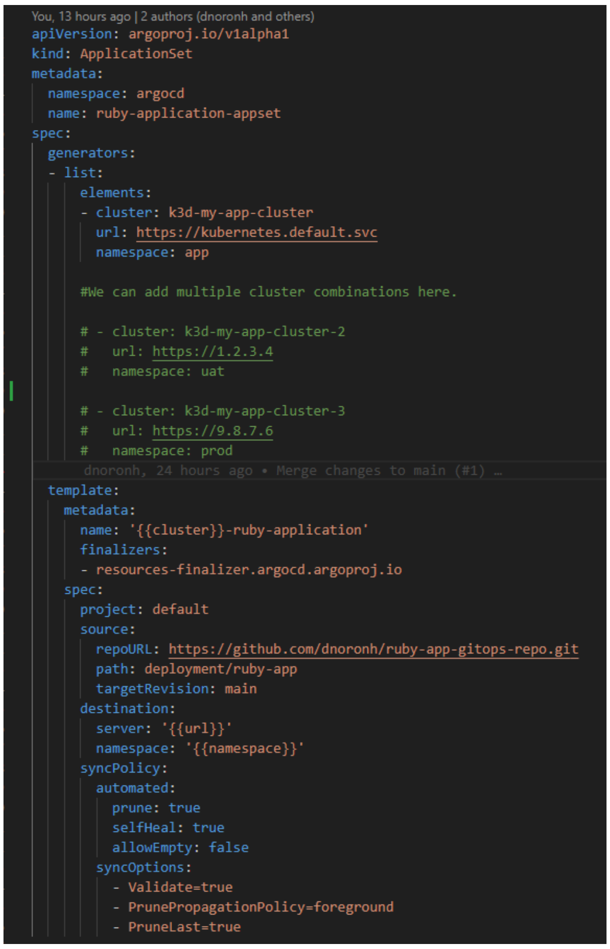
```
ruby-app-devops$ docker images | grep 1.0
localhost:5000/ruby-app    1.0          391d8ecbf69d   24 hours ago   23.9MB
ruby-app-devops$ |
```

2) **GitHub actions** – With Github actions workflows, we were able to observe the automated docker build as well as updating of the latest image tag in the manifest. This allows continuous build and deployment for faster releases. Additionally, we can add code / container scanning as part of the build pipeline to make the application more secure.

3) **Helm** – Using helm we were able to easily manifest the Kubernetes deployments. Helm values files introduce a flexible approach to configuration, enabling developers to customize settings for each instance and environment effortlessly. This eliminates the need to hardcode configurations directly into the codebase, and the same code can be seamlessly deployed across diverse environments, promoting consistency and efficiency. The values can be any interface connection details, ports, values that differ over environments.

4) **Argo CD** – Using Argo CD and its application Set object we were able to easily deploy the application to the Kubernetes cluster, and we also observed how application Set can be used to deploy the application to multiple clusters and namespaces irrespective of where they are hosted. This flexibility ensures that our application deployment remains agile and adaptable to varying infrastructural demands.

Below is the Application Set configuration used for our deployment. This can be further enhanced to support deployments to multiple clusters. Clusters can be setup in Argo CD using Kubernetes Secrets.



```
You, 13 hours ago | 2 authors (dnoronh and others)
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  namespace: argocd
  name: ruby-application-appset
spec:
  generators:
  - list:
      elements:
      - cluster: k3d-my-app-cluster
        url: https://kubernetes.default.svc
        namespace: app

      #We can add multiple cluster combinations here.

      # - cluster: k3d-my-app-cluster-2
      #   url: https://1.2.3.4
      #   namespace: uat

      # - cluster: k3d-my-app-cluster-3
      #   url: https://9.8.7.6
      #   namespace: prod
        dnoronh, 24 hours ago • Merge changes to main (#1) …
  template:
    metadata:
      name: '{{cluster}}-ruby-application'
      finalizers:
      - resources-finalizer.argocd.argoproj.io
    spec:
      project: default
      source:
        repoURL: https://github.com/dnoronh/ruby-app-gitops-repo.git
        path: deployment/ruby-app
        targetRevision: main
      destination:
        server: '{{url}}'
        namespace: '{{namespace}}'
      syncPolicy:
        automated:
          prune: true
          selfHeal: true
          allowEmpty: false
        syncOptions:
          - Validate=true
          - PrunePropagationPolicy=foreground
          - PruneLast=true
```

5) **K3D** - is a highly effective tool for swiftly creating local Kubernetes clusters for testing purposes. Its easy configurability and effortless destruction make it advantageous, especially in the context of the Docker in Docker concept.

In this scenario, we could setup multiple K3D clusters and use Argo CD to demonstrate multi-cluster deployments, but due to short time available, I am restricting it to single cluster.

## CONCLUSION

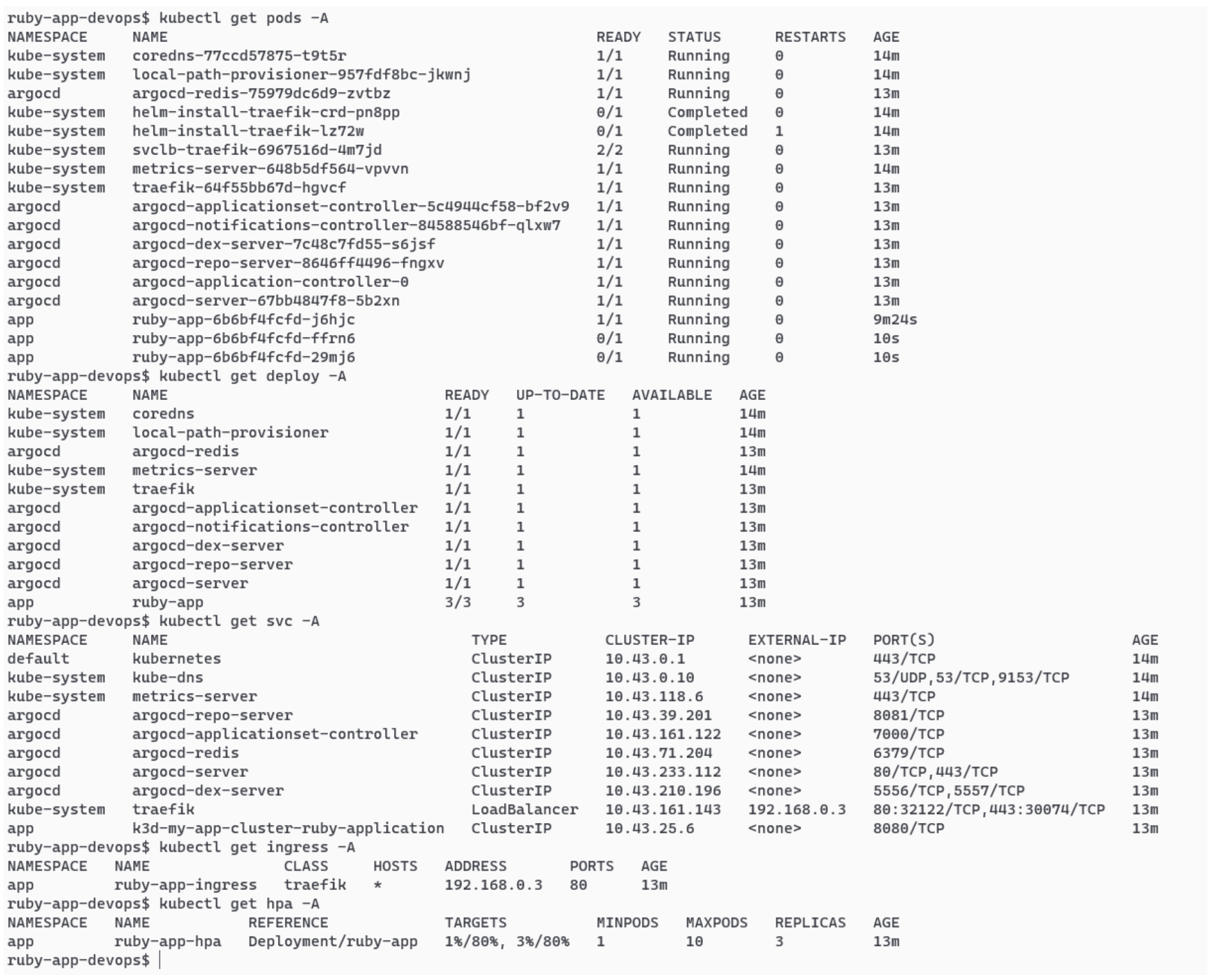
Through this challenge, a flexible solution was devised for the continuous build and deployment of a Ruby application in a hybrid, multi-cluster Kubernetes environment.

The pipeline, orchestrated using GitHub Actions, automated the containerization of the Ruby application based on industry best practices. Helm was instrumental in generating Kubernetes manifests, providing flexibility for deployments tailored to specific instances and environments. Argo CD streamlined the deployment process following the GitOps model, ensuring the deployment of the latest application version across relevant instances.

The integration of observability tools such as Prometheus, Grafana, Jaeger, and Datadog further enhances the reliability of applications hosted in the clusters. This deployment and improvement strategy positions the Ruby app for sustained success in dynamic and demanding operational landscapes.

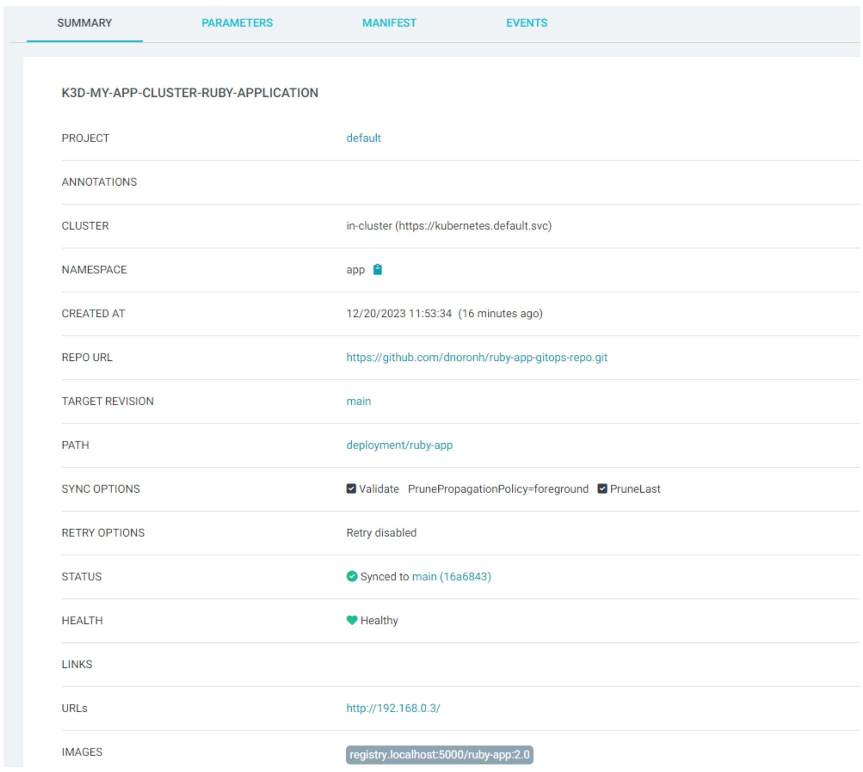
Few Additional screenshots

Few relevant K8s objects



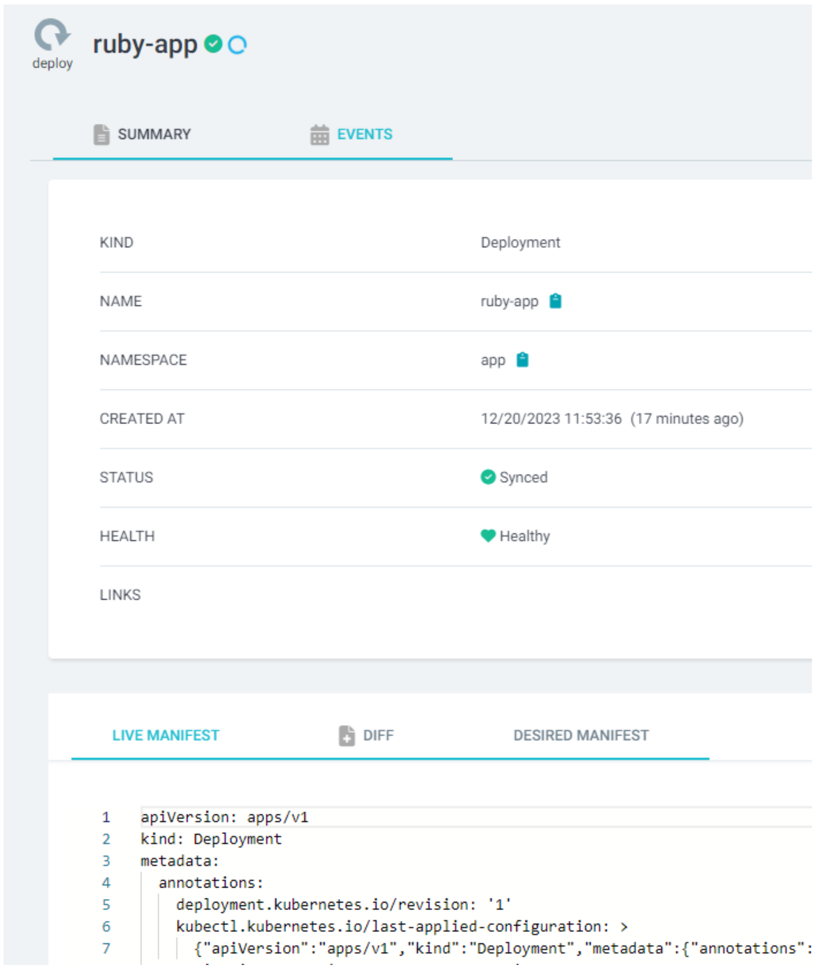
```
ruby-app-devops$ kubectl get pods -A
NAMESPACE     NAME                                              READY   STATUS      RESTARTS   AGE
kube-system   coredns-77ccd57875-t9t5r                          1/1     Running     0          14m
kube-system   local-path-provisioner-957fdf8bc-jkwnj            1/1     Running     0          14m
argocd        argocd-redis-75979dc6d9-zvtbz                     1/1     Running     0          13m
kube-system   helm-install-traefik-crd-pn8pp                    0/1     Completed   0          14m
kube-system   helm-install-traefik-lz72w                        0/1     Completed   1          14m
kube-system   svclb-traefik-6967516d-4m7jd                      2/2     Running     0          13m
kube-system   metrics-server-648b5df564-vpvvn                   1/1     Running     0          14m
kube-system   traefik-64f55bb67d-hgvcf                          1/1     Running     0          13m
argocd        argocd-applicationset-controller-5c4944cf58-bf2v9   1/1     Running     0          13m
argocd        argocd-notifications-controller-84588546bf-qlxw7    1/1     Running     0          13m
argocd        argocd-dex-server-7c48c7fd55-s6jsf                1/1     Running     0          13m
argocd        argocd-repo-server-8646ff4496-fngxv               1/1     Running     0          13m
argocd        argocd-application-controller-0                   1/1     Running     0          13m
argocd        argocd-server-67bb4847f8-5b2xn                    1/1     Running     0          13m
app           ruby-app-6b6bf4fcfd-j6hjc                         1/1     Running     0          9m24s
app           ruby-app-6b6bf4fcfd-ffrn6                         0/1     Running     0          10s
app           ruby-app-6b6bf4fcfd-29mj6                         0/1     Running     0          10s
ruby-app-devops$ kubectl get deploy -A
NAMESPACE     NAME                               READY   UP-TO-DATE   AVAILABLE   AGE
kube-system   coredns                            1/1     1            1           14m
kube-system   local-path-provisioner             1/1     1            1           14m
argocd        argocd-redis                       1/1     1            1           13m
kube-system   metrics-server                     1/1     1            1           14m
kube-system   traefik                            1/1     1            1           13m
argocd        argocd-applicationset-controller   1/1     1            1           13m
argocd        argocd-notifications-controller    1/1     1            1           13m
argocd        argocd-dex-server                  1/1     1            1           13m
argocd        argocd-repo-server                 1/1     1            1           13m
argocd        argocd-server                      1/1     1            1           13m
app           ruby-app                           3/3     3            3           13m
ruby-app-devops$ kubectl get svc -A
NAMESPACE     NAME                                  TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)                      AGE
default       kubernetes                            ClusterIP      10.43.0.1       <none>        443/TCP                      14m
kube-system   kube-dns                              ClusterIP      10.43.0.10      <none>        53/UDP,53/TCP,9153/TCP       14m
kube-system   metrics-server                        ClusterIP      10.43.118.6     <none>        443/TCP                      14m
argocd        argocd-repo-server                    ClusterIP      10.43.39.201    <none>        8081/TCP                     13m
argocd        argocd-applicationset-controller      ClusterIP      10.43.161.122   <none>        7000/TCP                     13m
argocd        argocd-redis                          ClusterIP      10.43.71.204    <none>        6379/TCP                     13m
argocd        argocd-server                         ClusterIP      10.43.233.112   <none>        80/TCP,443/TCP               13m
argocd        argocd-dex-server                     ClusterIP      10.43.210.196   <none>        5556/TCP,5557/TCP            13m
kube-system   traefik                               LoadBalancer   10.43.161.143   192.168.0.3   80:32122/TCP,443:30074/TCP   13m
app           k3d-my-app-cluster-ruby-application   ClusterIP      10.43.25.6      <none>        8080/TCP                     13m
ruby-app-devops$ kubectl get ingress -A
NAMESPACE   NAME               CLASS     HOSTS   ADDRESS       PORTS   AGE
app         ruby-app-ingress   traefik   *       192.168.0.3   80      13m
ruby-app-devops$ kubectl get hpa -A
NAMESPACE   NAME           REFERENCE             TARGETS          MINPODS   MAXPODS   REPLICAS   AGE
app         ruby-app-hpa   Deployment/ruby-app   1%/80%, 3%/80%   1         10        3          13m
ruby-app-devops$ |
```

Screenshots from argoCD



K3D-MY-APP-CLUSTER-RUBY-APPLICATION

| | |
|---|---|
| PROJECT | default |
| ANNOTATIONS | |
| CLUSTER | in-cluster (https://kubernetes.default.svc) |
| NAMESPACE | app |
| CREATED AT | 12/20/2023 11:53:34 (16 minutes ago) |
| REPO URL | https://github.com/dnoronh/ruby-app-gitops-repo.git |
| TARGET REVISION | main |
| PATH | deployment/ruby-app |
| SYNC OPTIONS | ✓ Validate PrunePropagationPolicy=foreground ✓ PruneLast |
| RETRY OPTIONS | Retry disabled |
| STATUS | ✓ Synced to main (16a6843) |
| HEALTH | Healthy |
| LINKS | |
| URLs | http://192.168.0.3/ |
| IMAGES | registry.localhost:5000/ruby-app:2.0 |



| | |
|---|---|
| KIND | Deployment |
| NAME | ruby-app |
| NAMESPACE | app |
| CREATED AT | 12/20/2023 11:53:36 (17 minutes ago) |
| STATUS | ✓ Synced |
| HEALTH | Healthy |
| LINKS | |

```
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
    deployment.kubernetes.io/revision: '1'
    kubectl.kubernetes.io/last-applied-configuration: >
      {"apiVersion":"apps/v1","kind":"Deployment","metadata":{"annotations":
```