**Requirements:**

1. Create a Simple CRUD application (no UI required , only API's).
2. Create Operator to do deploy application and Mongodb with PVC for mongo ( should be able to expand Volume).
3. Generate Helm charts for deployments.

**Problem Statement:**

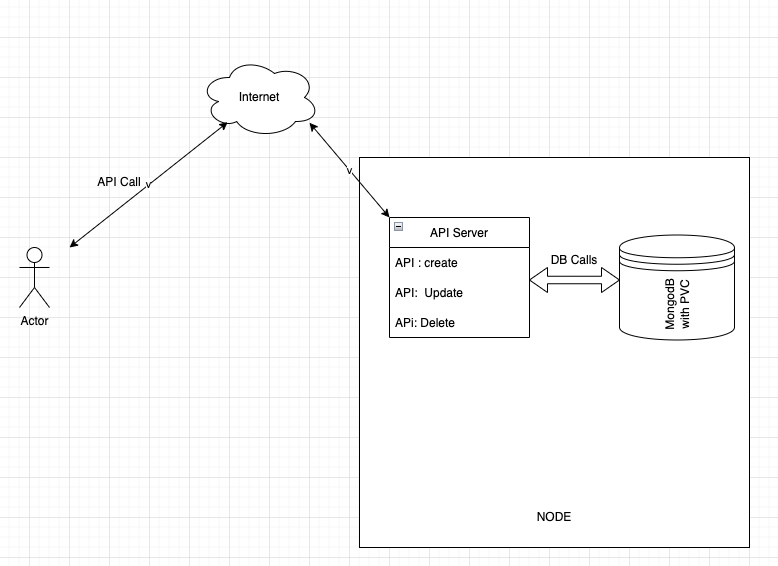
* Develop a stack which comprises of API server and Database.
* API server should be able to handle the CRUD operations.
* Restful CRUD API (Create, Read, Update, Delete) to be performed successfully in a Database.
* Export the above requirement to a docker image to be used as input for deployment to requirement No 2.

**Environment and Tools Used:**

* + 1. Docker ( Containerisation )
    2. Virtual Machine (ESXi) ( Virtualization)
    3. Python 3.0 ( Programming Language )
    4. Python Flask ( API server)
    5. MongoDB ( database)
    6. Minikube single node cluster
    7. Kopf
    8. Kubernetes

**Solution:**

First approach was to try the api server and mongo integration on the virtual machine and execute CRUD API methods ( GET , PUT, POST, DELETE) successfully



Steps performed:

* 1. Setting up the docker environment

sudo apt-get install -y apt-transport-https ca-certificates curl gnupg-agent software-properties-common

sudo apt-get install -y docker-ce docker-ce-cli containerd.io

docker –version

* 1. Pull the latest ‘mongo’ docker image to VM

docker pull mongo

docker pull mongo

Using default tag: latest

latest: Pulling from library/mongo

Digest: sha256:a89d79ddc5187f57b1270f87ec581b7cc6fd697efa12b8f1af72f3c4888d72b5

Status: Image is up to date for mongo:latest

docker.io/library/mongo:latest

* 1. Create a docker container for pulled image

docker create -it --name MongoDB\_dev -p 5000:27017 mongo

* 1. Start the mongodB container

docker start MongoDB\_dev

docker start MongoDB\_dev

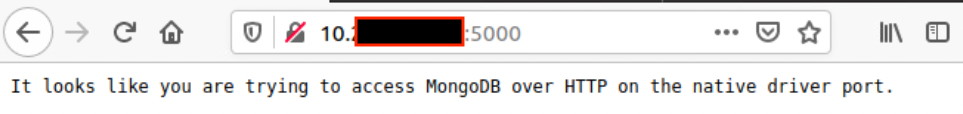
MongoDB\_dev

root@host:~/test\_proj# docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

21536dba5b97 mongo "docker-entrypoint.s…" 24 seconds ago Up 4 seconds 0.0.0.0:5000->27017/tcp, :::5000->27017/tcp MongoDB\_dev

* 1. Verification of MongoDB server:



MongoDB is accessible over 10.x.x.x:5000 port

* 1. Now create a python program using pymongo library for interacting with mongodb. This python program will create , update, read and delete records for a specific database in mongodb server

complete python file is attached in git repo for reference

* 1. Now we need to create a API server using FLASK in python

Python file is attached to git repo for reference . Dependent python libraries

are installed as part of requirements.txt

* 1. Run the flask server

root@host:~# python3 test\_api.py

\* Serving Flask app 'test\_api'

\* Debug mode: on

WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.

\* Running on all addresses (0.0.0.0)

\* Running on http://127.0.0.1:5010

\* Running on http://10.x.x.x:5010

Press CTRL+C to quit

\* Restarting with stat

\* Debugger is active!

\* Debugger PIN: 796-223-735

* 1. Validate the status using API for server in running status

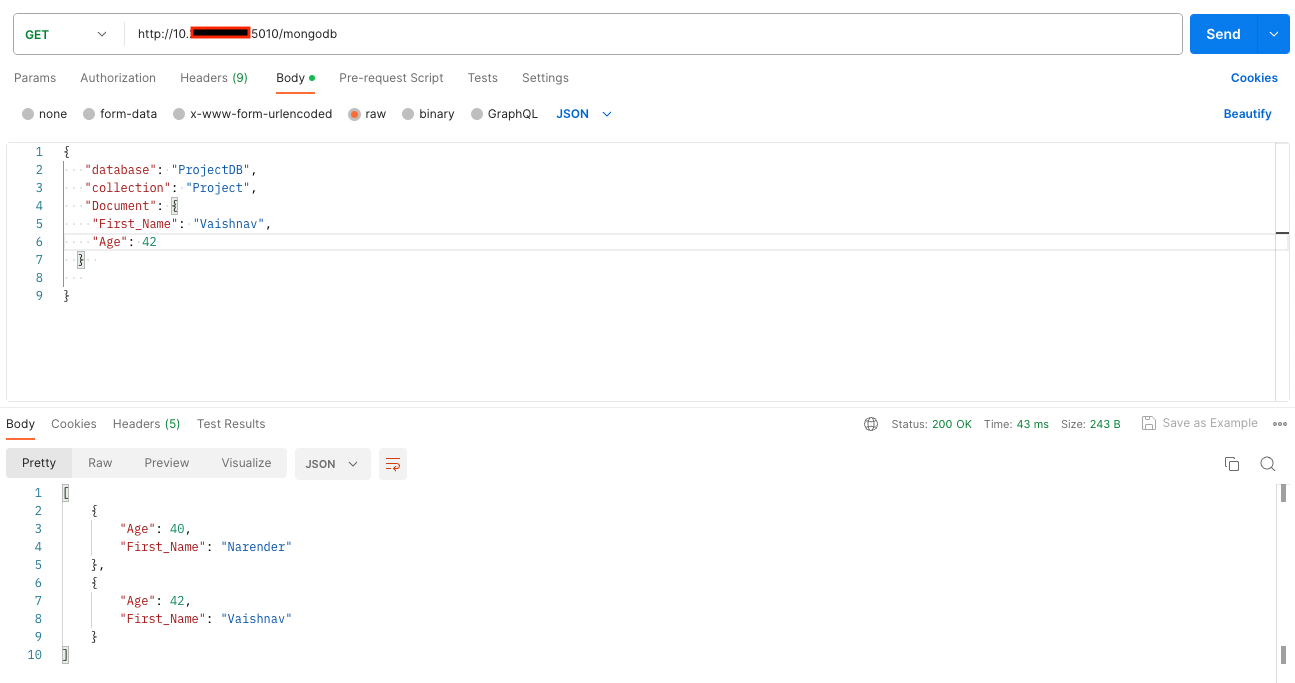
A screenshot of a computer

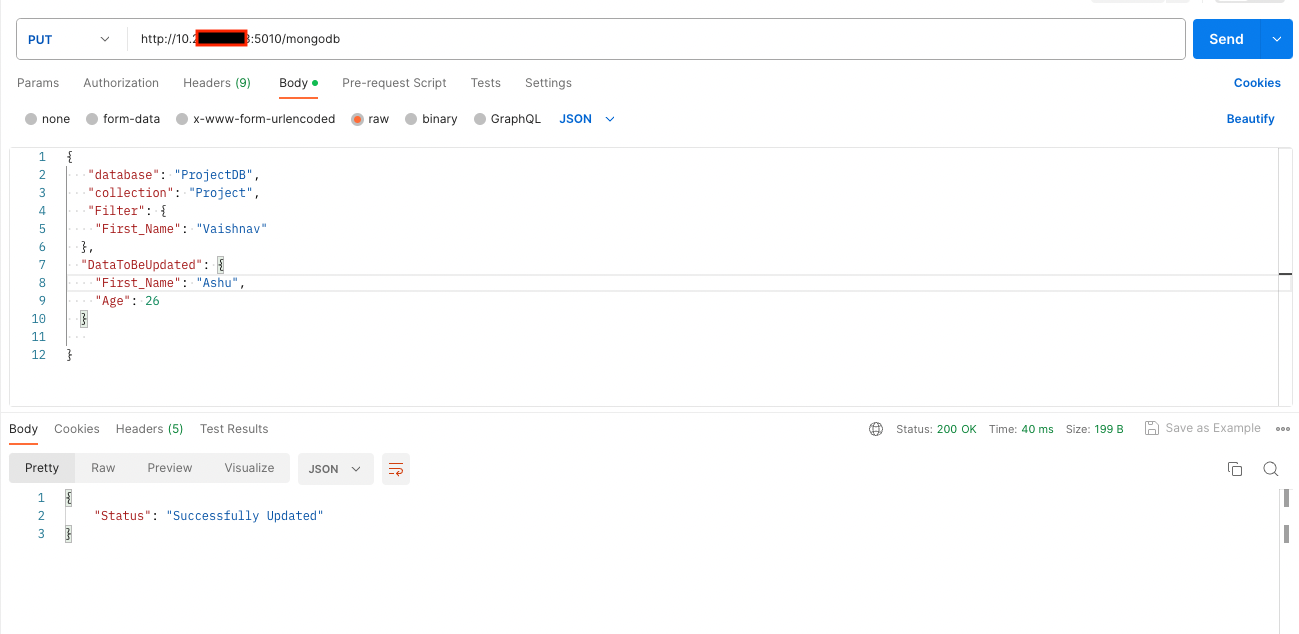
Description automatically generated

* 1. Perform the CRUD operations using POSTMAN on mongodb via API server

A screenshot of a computer

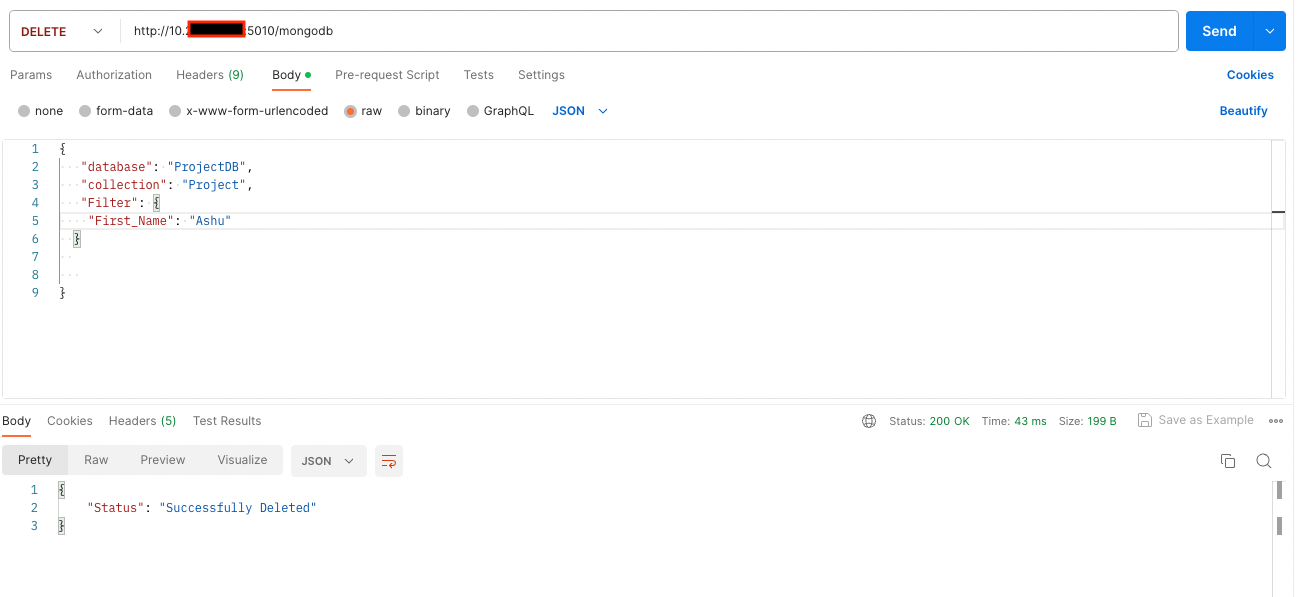
Description automatically generated

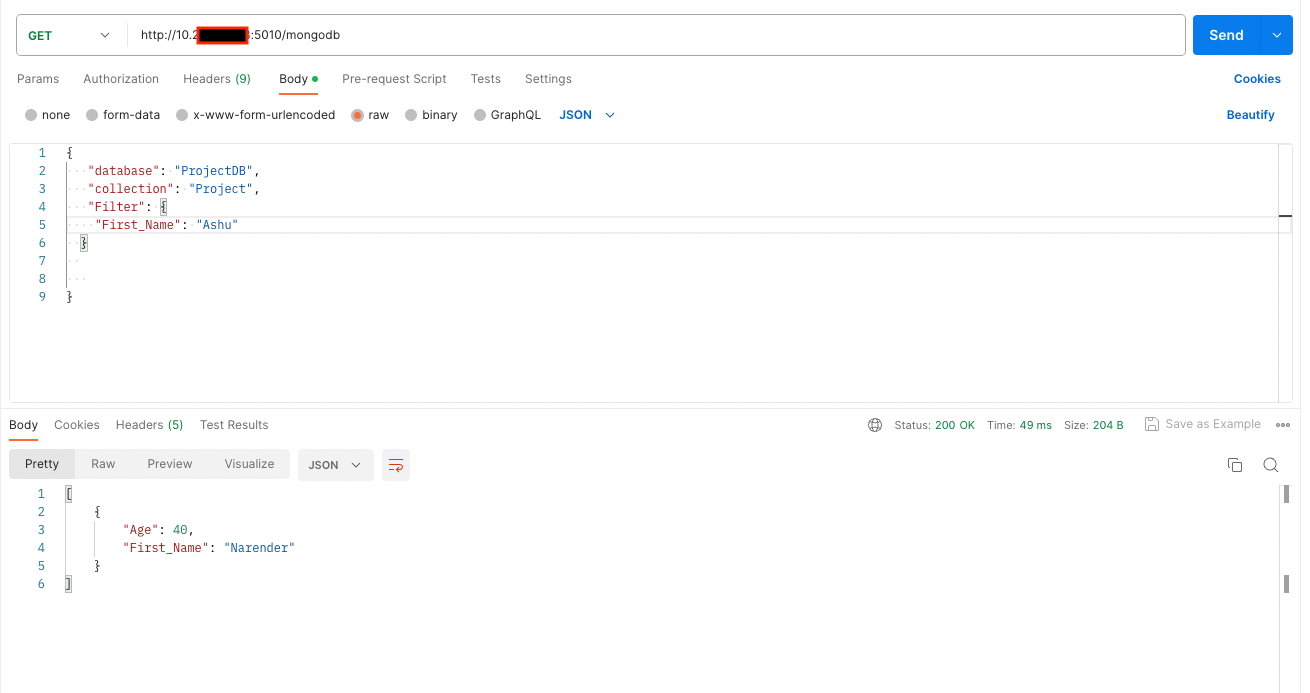




A screenshot of a computer

Description automatically generated





* 1. Now created a dockerfile having both the containers ( FLASK,Mongo) together so that the CRUD operations are targeted to API server and internally it is directed to MongoDB

DockerFile, requirements.txt, crud\_mongo.py files are attached to git repo for reference

root@ansible-host:~/test\_proj# cat requirements.txt

click==7.1.2

Flask==1.1.2

itsdangerous==1.1.0

Jinja2==2.11.2

MarkupSafe==1.1.1

pymongo==3.10.1

Werkzeug==1.0.1

root@ansible-host:~/test\_proj# cat docker-compose.yaml

version: "3"

services:

mymongo\_1:

image: "mongo"

ports:

- '27017:27017'

networks:

network:

ipv4\_address: 10.5.0.6

myreader:

build: .

depends\_on:

- mymongo\_1

ports:

- "5010:5010"

networks:

network:

ipv4\_address: 10.5.0.5

networks:

network:

driver: bridge

ipam:

config:

- subnet: 10.5.0.0/16

gateway: 10.5.0.1

root@ansible-host:~/test\_proj# cat Dockerfile

# Step 1 select default OS image

FROM python:3.10-slim-bullseye

# # Step 2 tell what you want to do

RUN apt-get update -y && apt-get install -y python3-pip

# # Step 3 Configure a software

# # Defining working directory

WORKDIR /app

# # Copy everything which is present in my docker directory to working (/app)

COPY /requirements.txt /app

RUN pip3 install --upgrade pip && pip3 install -r requirements.txt

COPY ["crud\_mongo.py", "/app"]

# Exposing an internal port

EXPOSE 5001

# Step 4 set default commands

# These are permanent commands i.e even if user will provide come commands those will be considered as argunemts of this command

ENTRYPOINT [ "python3" ]

# These commands will be replaced if user provides any command by himself

CMD ["crud\_mongo.py"]

* 1. Now create containers using docker compose as shown below

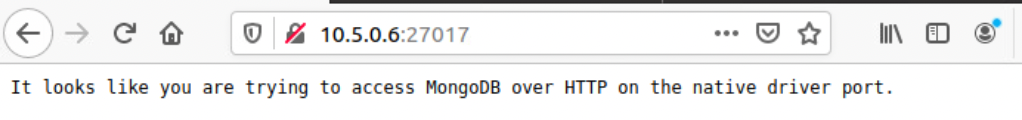
A close-up of a computer screen

Description automatically generated

A white text with black numbers

Description automatically generated

* 1. Validate the API and Mongodb server individually



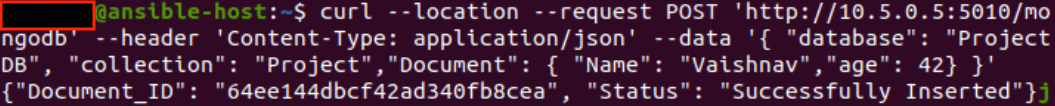
A white background with black lines

Description automatically generated

* 1. Now perform CRUD operations on API server which will redirect request to MongodB

A computer screen shot of white text

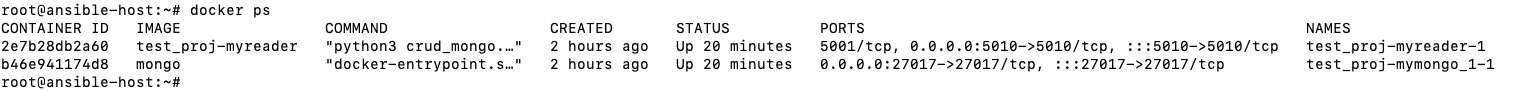
Description automatically generated



A computer screen shot of a computer code

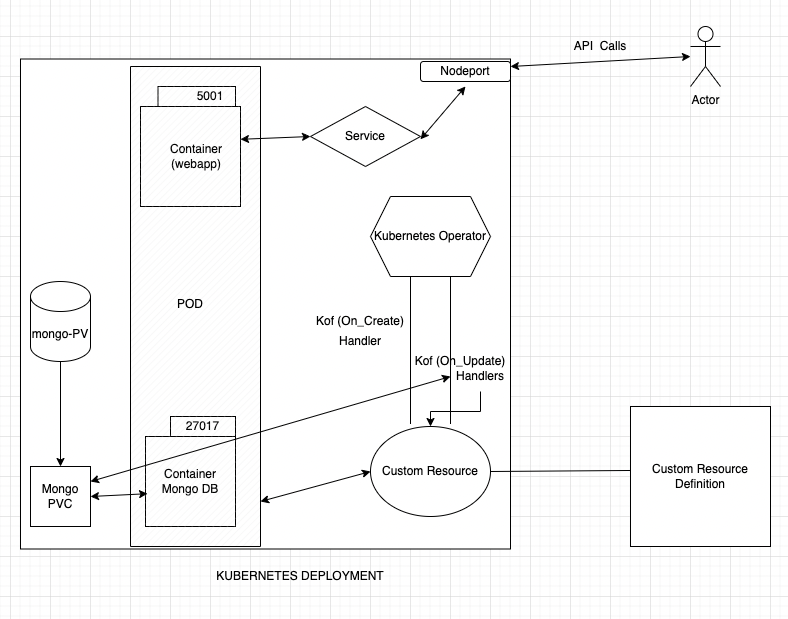
Description automatically generated

* 1. Docker images getting generated as part of the docker compose execution



Creation of Kubernetes Operator to deploy application and Mongodb with PVC for mongo ( should be able to expand Volume).

1. Design the workflow for Python based Kubernetes operator which involves the deployment of CRUD application with MongoDB and dynamic update of PVC for it



1. First create the Custom Resource Definition file ( ProjectCRD.yaml)

A screenshot of a computer program

Description automatically generated

1. Apply and bring up the CRD



1. Create a Custom Resource as per the requirements

A screenshot of a computer

Description automatically generated

1. Bringup the Custom Resource (ProjectCR.yaml)

**@ansible-host**:**~**$ kubectl get project

NAME AGE

project-new-object 118m

**@ansible-host**:**~**$

1. Create the operator file for creating the create deployment and update PVC scenario

import kopf

import kubernetes.client

from kubernetes.client.rest import ApiException

import yaml

**@kopf.on.create**('assessment.com', 'v1', 'projects')

def create\_fn(spec, \*\*kwargs):

name = kwargs["body"]["metadata"]["name"]

print("Name is %s\n" % name)

# Create the deployment spec

doc = yaml.safe\_load(f"""

apiVersion: apps/v1

kind: Deployment

metadata:

name: {name}-deployment

labels:

app: {name}

spec:

replicas: {spec.get('replicas', 1)}

selector:

matchLabels:

app: {name}

template:

metadata:

labels:

app: {name}

spec:

containers:

- name: webapp

image: webapp\_4:latest

ports:

- containerPort: 5001

imagePullPolicy: IfNotPresent

- name: mongo

image: mongo:latest

imagePullPolicy: IfNotPresent

volumeMounts:

- name: storage

mountPath: /data/db

volumes:

- name: storage

persistentVolumeClaim:

claimName: mongo-pvc

""")

# Make it our child: assign the namespace, name, labels, owner references, etc.

kopf.adopt(doc)

# Actually create an object by requesting the Kubernetes API.

api = kubernetes.client.AppsV1Api()

try:

depl = api.create\_namespaced\_deployment(namespace=doc['metadata']['namespace'], body=doc)

# Update the parent's status.

return {'children': [depl.metadata.uid]}

except ApiException as e:

print("Exception when calling AppsV1Api->create\_namespaced\_deployment: %s\n" % e)

**@kopf.on.update('assessment.com', 'v1', 'projects')**

def update\_fn(spec, name, \*\*kwargs):

size = spec.get('size', None)

if not size:

raise kopf.PermanentError(f"Size must be set. Got {size!r}.")

doc = yaml.safe\_load(f"""

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: mongo-pvc

#annotations:

# volume.beta.kubernetes.io/storage-class: standard

spec:

accessModes:

- ReadWriteOnce

resources:

requests:

storage: "{size}"

""")

kopf.adopt(doc)

# Actually patch an object by requesting the Kubernetes API.

api = kubernetes.client.CoreV1Api()

try:

depl = api.patch\_namespaced\_persistent\_volume\_claim(name='mongo-pvc', namespace='default', body=doc)

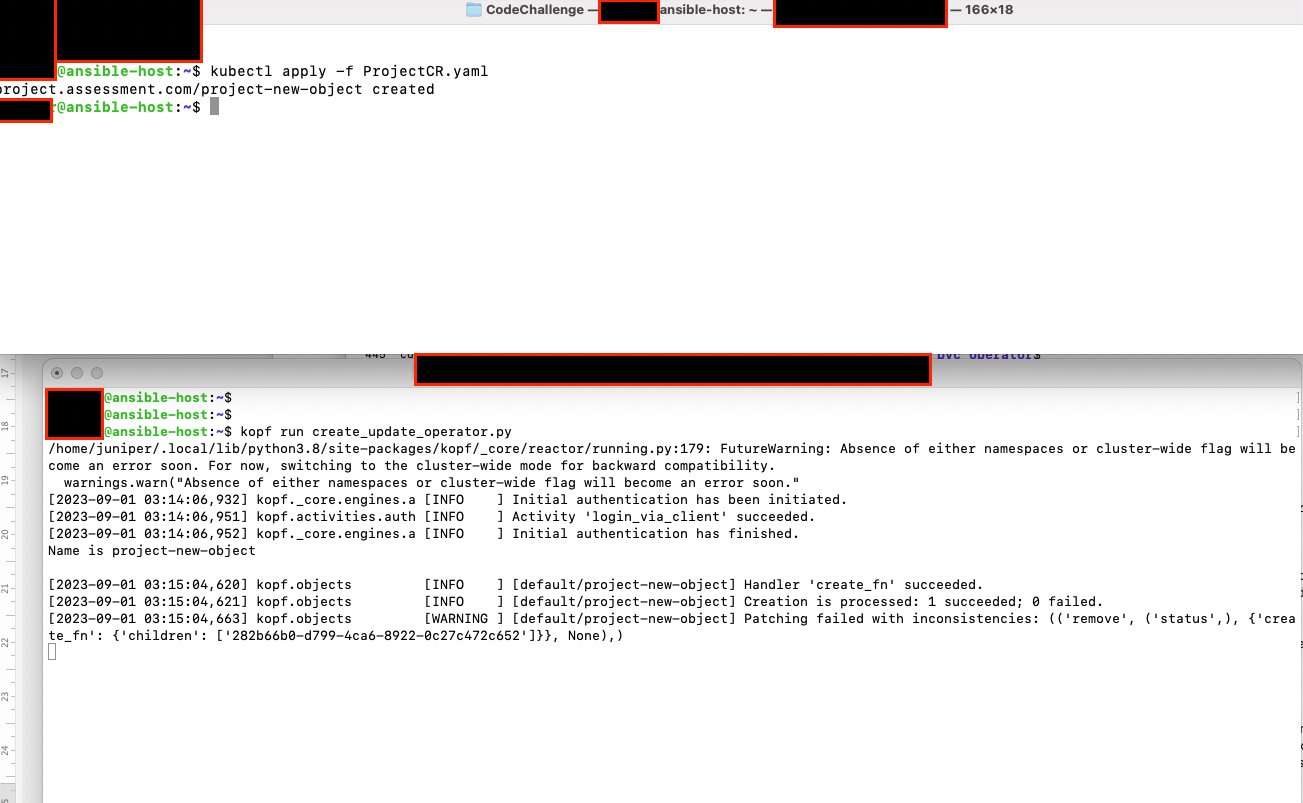
# Update the parent's status.

return {'children': [depl.metadata.uid]}

except ApiException as e:

print("Exception when calling CoreV1Api->patch\_namespaced\_persistent\_volume\_claim: %s\n" % e)

1. Run this operator.py file in separate terminal , so that when it detects a custom resource it should execute the handlers (on\_create, On\_update)



**@ansible-host**:**~**$ kubectl get deployments

NAME READY UP-TO-DATE AVAILABLE AGE

project-new-object-deployment 1/1 1 1 3m59s

**@ansible-host**:**~**$

**@ansible-host**:**~**$ kubectl get pods

NAME READY STATUS RESTARTS AGE

project-new-object-deployment-55b4b954f5-lg2zf 2/2 Running 0 4m7s

**@ansible-host**:**~**$ kubectl describe pods project-new-object-deployment-55b4b954f5-lg2zf

Name: project-new-object-deployment-55b4b954f5-lg2zf

Namespace: default

Priority: 0

Service Account: default

Node: minikube/192.168.49.2

Start Time: Fri, 01 Sep 2023 03:15:04 -0400

Labels: app=project-new-object

pod-template-hash=55b4b954f5

Annotations: <none>

Status: Running

IP: 10.244.0.64

IPs:

IP: 10.244.0.64

Controlled By: ReplicaSet/project-new-object-deployment-55b4b954f5

Containers:

webapp:

Container ID: docker://d85eb84632835167a876f86b53cfa1da6dabe345f9e7b2edccace9f30e8819f4

Image: webapp\_4:latest

Image ID: docker://sha256:1c04d3ea3fcd47afafaaf9740d103a1ea5babbdcd25504e39d42343969dc2118

Port: 5001/TCP

Host Port: 0/TCP

State: Running

Started: Fri, 01 Sep 2023 03:15:05 -0400

Ready: True

Restart Count: 0

Environment: <none>

Mounts:

/var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-9prwg (ro)

mongo:

Container ID: docker://3ec198cc751cf08a88435d554baf3ff73415588ffd87ed5272f99056a95aa70b

Image: mongo:latest

Image ID: docker-pullable://mongo@sha256:a89d79ddc5187f57b1270f87ec581b7cc6fd697efa12b8f1af72f3c4888d72b5

Port: <none>

Host Port: <none>

State: Running

Started: Fri, 01 Sep 2023 03:15:06 -0400

Ready: True

Restart Count: 0

Environment: <none>

Mounts:

/data/db from storage (rw)

/var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-9prwg (ro)

Conditions:

Type Status

Initialized True

Ready True

ContainersReady True

PodScheduled True

Volumes:

storage:

Type: PersistentVolumeClaim (a reference to a PersistentVolumeClaim in the same namespace)

ClaimName: mongo-pvc

ReadOnly: false

kube-api-access-9prwg:

Type: Projected (a volume that contains injected data from multiple sources)

TokenExpirationSeconds: 3607

ConfigMapName: kube-root-ca.crt

ConfigMapOptional: <nil>

DownwardAPI: true

QoS Class: BestEffort

Node-Selectors: <none>

Tolerations: node.kubernetes.io/not-ready:NoExecute op=Exists for 300s

node.kubernetes.io/unreachable:NoExecute op=Exists for 300s

Events:

Type Reason Age From Message

---- ------ ---- ---- -------

Normal Scheduled 4m25s default-scheduler Successfully assigned default/project-new-object-deployment-55b4b954f5-lg2zf to minikube

Normal Pulled 4m25s kubelet Container image "webapp\_4:latest" already present on machine

Normal Created 4m25s kubelet Created container webapp

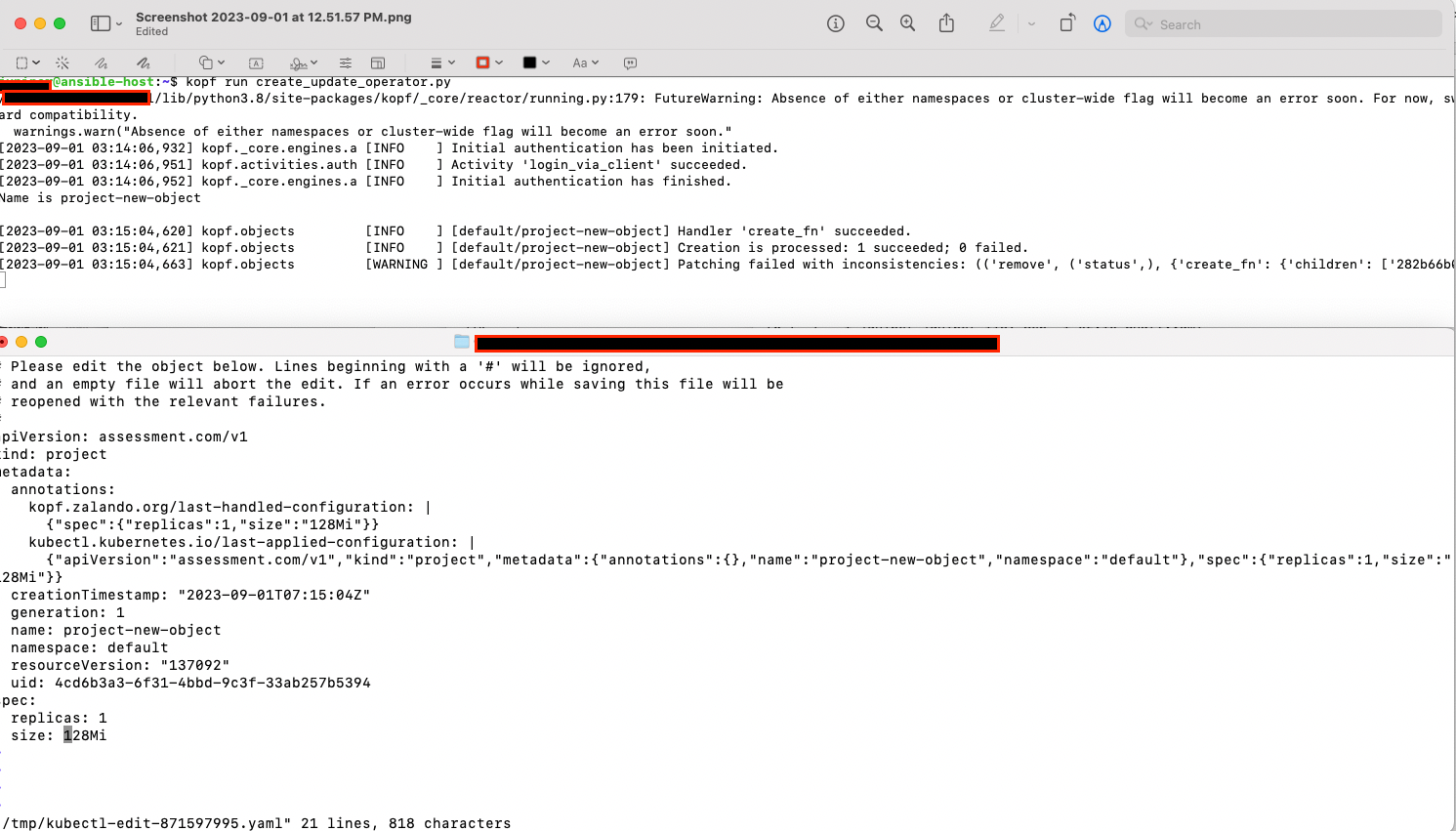
Normal Started 4m25s kubelet Started container webapp

Normal Pulled 4m25s kubelet Container image "mongo:latest" already present on machine

Normal Created 4m25s kubelet Created container mongo

Normal Started 4m24s kubelet Started container mongo

1. When Custom Resource Modfied to expand the mongodb PVC , the on\_update handler got triggered



The On\_update handler triggers to update the PVC for mongo DB

A screenshot of a computer

Description automatically generated