**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)

**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

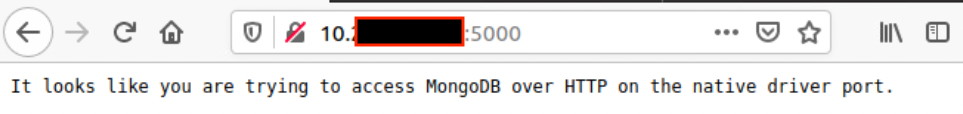
MongoDB\_dev

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

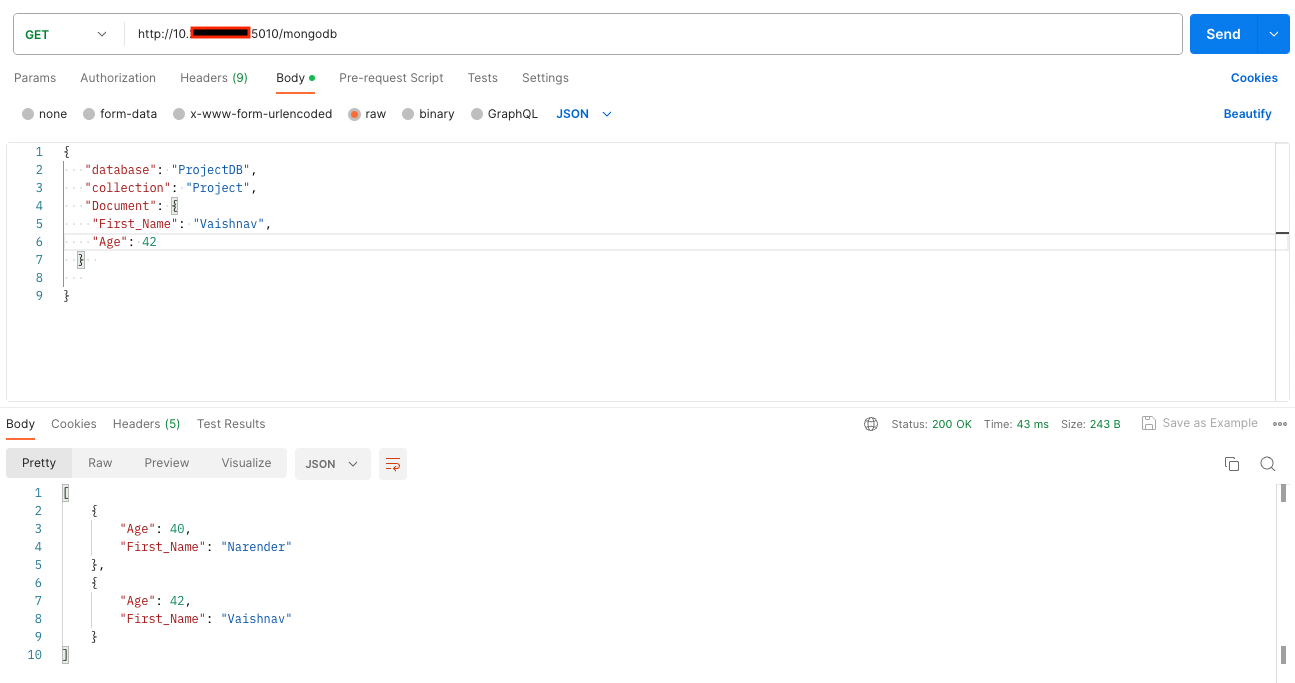
A screenshot of a computer

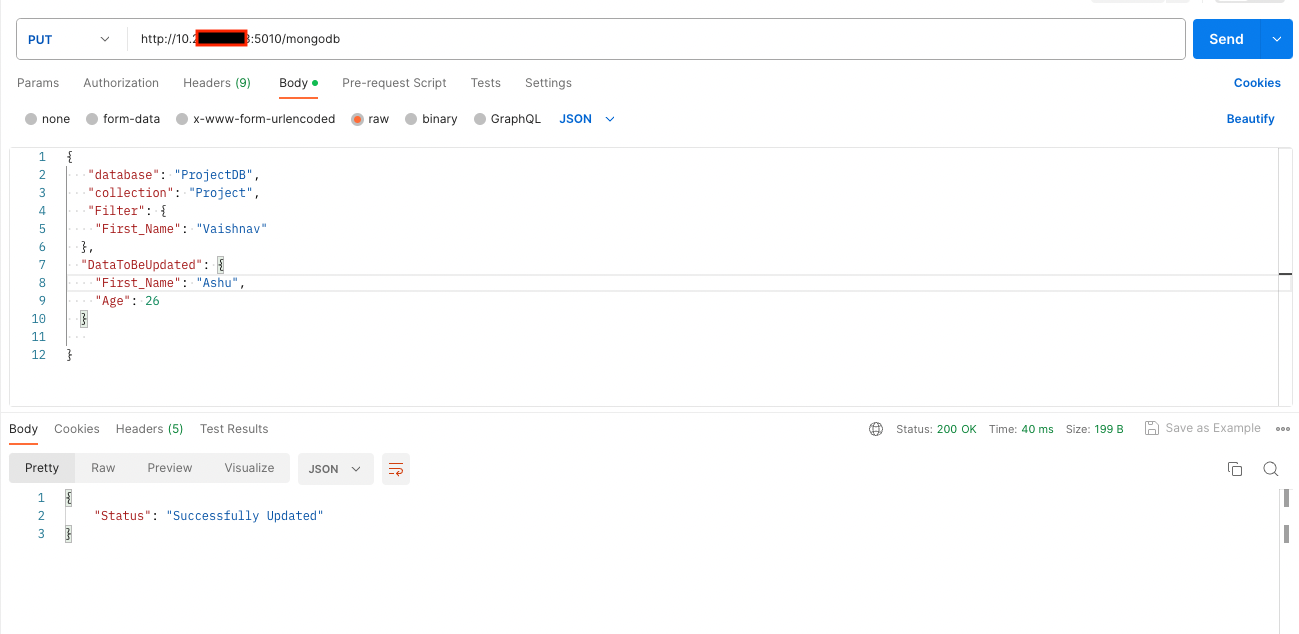
Description automatically generated

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

A screenshot of a computer

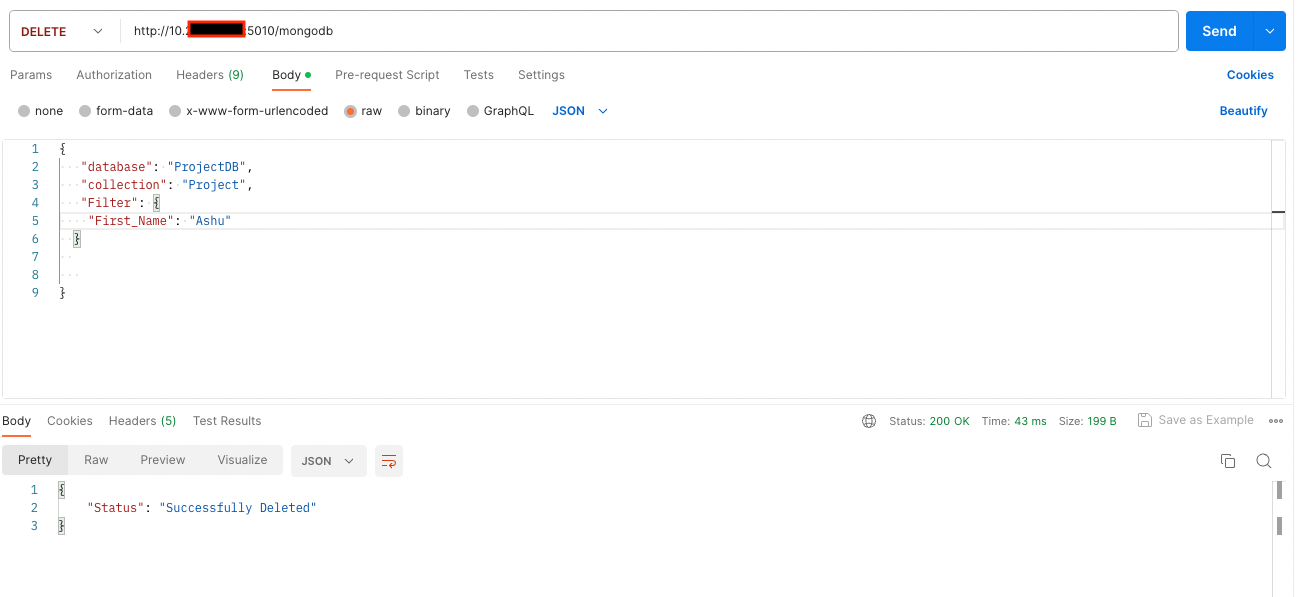
Description automatically generated

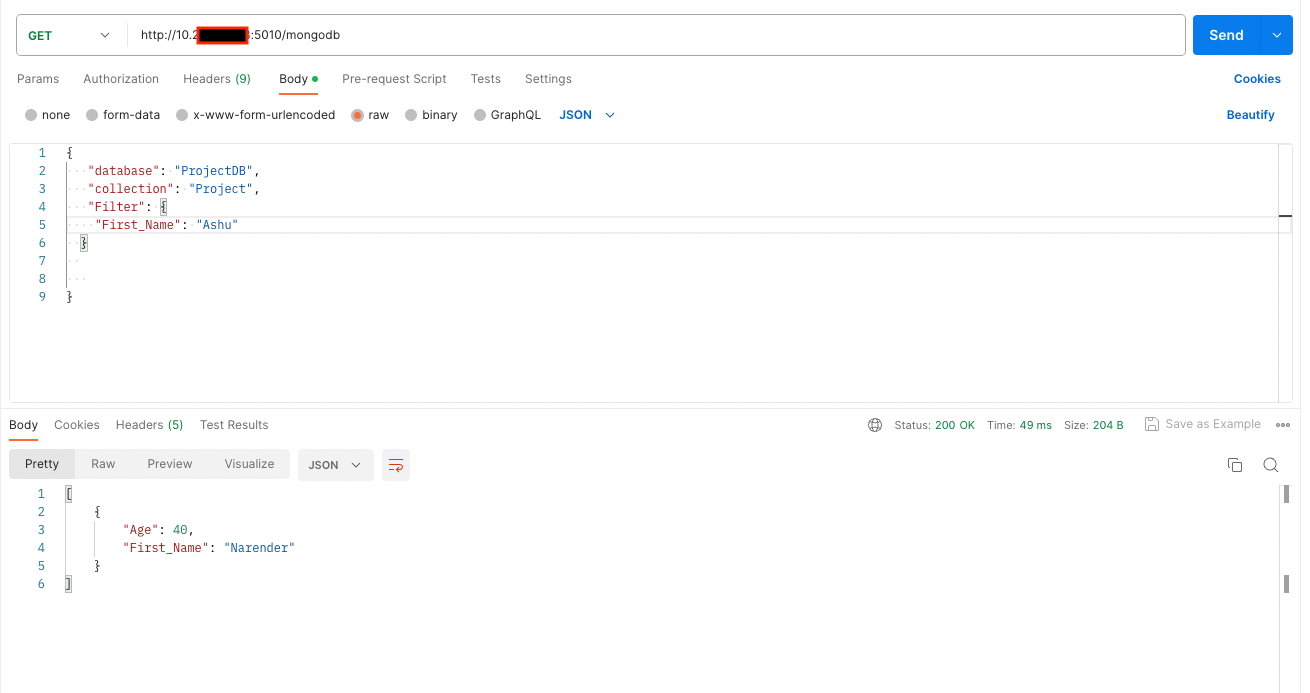




A screenshot of a computer

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

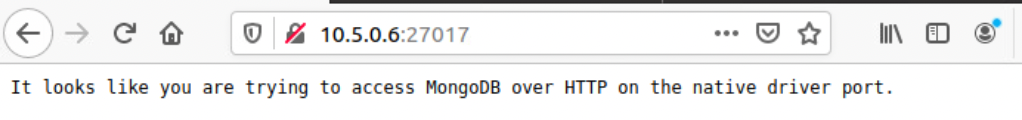
A close-up of a computer screen

Description automatically generated

A white text with black numbers

Description automatically generated

* 1. Validate the API and Mongodb server individually



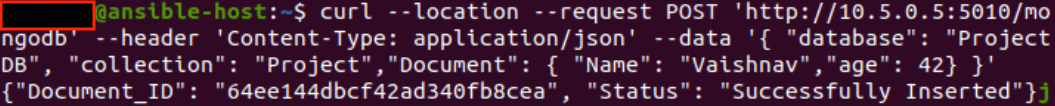
A white background with black lines

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

