PE Recruitment | Pre-Interview Assignment 2 - Coding and DevOps

Coding and DevOps Challenge

As a first step in assessing your fitness for a role within the Google Cloud Platform (GCP) Platform Engineering team, we want you to showcase your technical knowledge and skills by completing a coding and devops challenge. Please be aware that there is no right or wrong solution, and you will have the oppurtunity to walk through your solution as a part of the later technical interview.

Task prerequisites

You will need a development environment to work on, for example a laptop/PC with the following tools/software installed, or you could use a cloud environment (as long as we can reproduce your implementation on a Kubernetes cluster).

- Python
- Docker (Docker for Mac/Windows would suffice)
- Kubernetes (Docker for Mac/Windows with Kubernetes enbaled would suffice)
- Helm
- Kubectl
- Ingress Nginx controller deployed on your Kubernetes cluster

Task description

1. **Write a Python API using the Fast API framework which:**
    i. Will take a request with an uploaded YAML file which contains key/values of a person.
    ii. The API should have a function to parse the YAML and push a record to a Redis store (instructions to setup Redis are further down this page). You can use any backend if you like, redis is simple enough and we are not concerned about the backend implementation.
    iii. The API should have a function to also retrieve data about a paticular person stored in the Redis store.
    iv. Try to production-ise your code as much as possible.
    v. Use best practises in all aspects of your implementation.

2. **Write a Python CLI client to interact with the API for both:**
    i. Uploading/updating a person record via a YAML file
    ii. Retrieving a person record - it should print the response to your terminal
    iii. Try to production-ise your code as much as possible.
    iv. Use best practises in all aspects of your implementation.

    The YAML file should follow this template:

```
first_name: <name>
last_name: <name>
age: <age>
```

3. **Containerise the Python API in a Dockerfile (with the intention of deploying the image on Kubernetes):**
    i. Use best practises in the Dockerfile

4. **Create a Helm chart to deploy the API image:**
    i. Use best practises in the chart and Kubernetes resources
    ii. Try to production-ise your code as much as possible in your environment
        a. There may be limitations to your Kubernetes environment
        b. Document any further improvements that could be made
    iii. The Deployment should be exposed via Nginx Ingress

You can use a standard Redis chart and image to deploy Redis:

```
helm repo add bitnami https://charts.bitnami.com/bitnami
helm install my-release bitnami/redis --set auth.password=rdBagRsVs4

# expose to localhost if required
kubectl port-forward --namespace default svc/my-release-redis-master 6379:6379
```

5. **Prove you can add, update and retrieve person records with your Python client against the deployed API:**
    i. Deploy the charts for Redis and the API
    ii. Produce a document/readme on how to test via your CLI, include:
        a. Sample input/output data used
        b. Screenshots
        c. Instructions for us to recreate
    iii. Push all code and documentation to a github repo and share the repo link.

Try to organise your repo using best practices as this will house all the code and automation for the API image, chart and all Python source.