Docker Challenge

Hello and thank you for taking the time to do this challenge.

The goal of this challenge is to get a rough overview of your coding, testing, automation and documentation skills

Description

- Write an HTTP service in any language you think is appropriate. (i.e. java, go, python)
- Create your own git repository for this challenge
- Commit often! Instead of one final big commit, we would like to see small commits that build up to the result of your test, instead of one final commit with all the code.
- When you are ready, just send us the link of your repository with your solution

Level 1

Create an HTTP service that complies with the following requirements:

1. Command-line flag for the listening port (defaults to 8080) and environment variable override.

2. Use mostly standard libraries.

3. Three HTTP endpoints:

- /helloworld --> returns "Hello Stranger"
- /helloworld?name=AliAhmadi (any filtered value) --> returns "Hello AliAhmadi" (camel-case gets cut by spaces)
- /author --> returns "Ali Ahmadi" (put your name and family. hard code it in code)

5. Test your API with Postman

6. A structured log is written to standard out with:

- ISO date
- HTTP status
- Request

7. Write a readme file with usage examples.

8. Unit testing of all functionalities (flags, endpoints, etc.).

9. A production-ready Dockerfile, so the service can run safely inside a container.

10. Documentation where it makes sense

Level 2

Great! We have an HTTP service. Now let's put it somewhere.

1. Install kind (https://kind.sigs.k8s.io/) or any other lightweight, locally installable, cluster manager you prefer.

2. Create the required Kubernetes files to deploy our HTTP service into the cluster.

3. Our HTTP server should now also have an appropriate health check mechanisms. Add an endpoint (like /healthz) to the http server and define liveness and readiness prob parameters in pod spec for that. For more info take a look at https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-startup-probes/ link.

Bounce score

Our HTTP server should now also have an appropriate shutdown sequence. Add a graceful shutdown to your container and pod definitions so it will shutdown according to the following requirements:

- Shutdown timeout 30 seconds.

- New requests are ignored.

- Pod/container is killed after 30 seconds no matter what.