Endocode 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

- 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.

A little bit of programming

Create an HTTP service that complies with the following requirements:

1. Written in any language you think is appropriate. (i.e. python,go, rust).
2. Command-line flag for the listening port (defaults to 8080) and environment variable override.
3. Use mostly standard libraries.
4. Please provide a Makefile (or use a similar tool) to define tasks
5. Three HTTP endpoints:
   - `/helloworld`
     --> returns "Hello Stranger"
   - `/helloworld?name=AlfredENeumann` (any filtered value)
     --> returns "Hello Alfred E Neumann" (camel-case gets cut by spaces)
   - `/versionz`
     --> returns a JSON with git hash and name of the project (needs to be compiled in)
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.

CI

Cool, we have an HTTP service running. Now we want to automate the process including testing of our changes. Let's set up a CI/CD pipeline.

1. Use a CI/CD tool you like. Github Actions might be straight forward. Jenkins is fine too (Install Jenkins locally in that case)
2. Create a test & deployment pipeline to build the HTTP service.
3. Create a stage/step in your pipeline for automated testing.
4. Save your Docker image somewhere you can retrieve it from later.

CD

Great! We have an HTTP service and a way of automatically creating it.
Now let's put it somewhere. That can be locally (with minikube) or cloud based (with GKE or similar)

1. Bring up a Kubernetes Cluster
2. Create the required Kubernetes files to deploy our HTTP service into the cluster.
3. Automate the previous step into your Jenkins pipeline.
4. 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:
    i. Shutdown timeout 30 seconds.
    ii. New requests are ignored.
    iii. Pod/container is killed after 30 seconds no matter what.

IaC

Heaven above. That is a lot already. If you still have an appetite, go ahead and make it even better. Let's automate it even more!

1. Create a Helm chart for our HTTP service.
2. Terraform the deployment of our HTTP service into our local cluster. Your CI Pipeline should now deploy everything into your cluster using Terraform.

K8s Debugging

for some reason our nginx container is not reachable via curl.
Can you check whats wrong?

deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        command: ["sleep"]
        ports:
        - containerPort: 80
```

service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
spec:
  type: NodePort
  selector:
    app: wrong_nginx
  ports:
    - port: 81
      targetPort: 81
      nodePort: 30007
```

GCP Architecture

Imagine, a customer has some python applications which are already packaged via docker. They want a very easy way with a minimal operational effort to run this application in GCP. Suggest a GCP service that would be an option and tell us why.

HAVE FUN, BE CREATIVE AND SURPRISE US!