DevOps Interview Assignment

DevOps Hiring Team, TeamViewer GmbH 14/03/2024

Abstract

In this document you will find three simple DevOps related tasks which you have to complete. Afterwards please send the solutions back to us as a zip archive containing **exactly** the following three files:

1. `main.go`
2. `Dockerfile`
3. `chart.tar.gz`

1 Programming

Below you can find the source code of a simple program written in *Golang* which calculates the mass of a aluminium sphere and a iron cube when calling the respective endpoint and passing the geometries dimension to it. The port on which the program listens has to be passed as a command-line parameter when calling the it.

Please complete the program by adding the missing go `types` and `functions` needed to do the calculations in the marked area between.

```go
package main

import (
  "fmt"
  "net/http"
  "strconv"
  "os"
  "math"
)

type Mass struct {
  Density float64
}

// --- ADD YOUR CODE ---




// --- BETWEEN THOSE LINES ---

func Handler(massVolume MassVolume) http.HandlerFunc {
  return func(w http.ResponseWriter, r *http.Request) {
    if dimension, err := strconv.ParseFloat(r.URL.Query().Get("dimension"), 64); err == nil {
      weight := massVolume.density() * massVolume.volume(dimension)
      w.Write([]byte(fmt.Sprintf("%.2f", math.Round(weight*100)/100)))
      return
    }
    w.WriteHeader(http.StatusBadRequest)
  }
}

func main() {
  port, err := strconv.Atoi(os.Args[1])
  if err != nil {
    panic(err)
  }

  aluminiumSphere := Sphere{Mass{Density: 2.710}}
  ironCube := Cube{Mass{Density: 7.874}}

  http.HandleFunc("/aluminium/sphere", Handler(aluminiumSphere))
  http.HandleFunc("/iron/cube", Handler(ironCube))

  if err := http.ListenAndServe(fmt.Sprintf(":%d", port), nil); err != nil {
    panic(err)
  }
}
```

2 Containerization

Create a **single** *Dockerfile* which compiles and containerizes the go program from 1. The resulting image should be less than 100MB of size (compressed), the application should not run as root user and the port command-line parameter should be made configurable by an environment variable which can be passed to the container when starting it.

3 Deployment

Create a simple *Helm Chart* which contains the resources required to deploy the container from 2 to a *Kubernetes* cluster. The port environment variable should be made configurable via the *Helm Charts* `values.yaml` file.