# Containers and Cloud Exam – 18 February 2024

Problems for the exam for the ["Containers and Cloud"](https://softuni.bg/trainings/4359/containers-and-cloud-january-2024) course @ SoftUni

## Run React app with Docker

You are provided with a **React** **app** with **MongoDB** **database** and **NodeJS** **backend**.

Your task is to create three containers for the three services and run them locally using Docker.

### Steps

Create two **Dockerfiles** – one for the **backend** service and one for the **frontend** service.

Push the two images to Docker Hub.

Create a **docker-compose.yml** file, following the steps below, and run the containers, using the **images** that you previously pushed to Docker Hub.

* Provide **Docker build contexts** for the frontend and backend services
* Expose the frontend **service** on port 3000and the backend **service** on port 80
* Create the following **volumes**:
  + For the mongo service, a **volume** with name "data" for the container's "/data/db" folder
  + For the backend service, a **volume** with name "logs" for the container's "logs" **folder** (include its **path**)
  + For the backend service, a **volume** for the container's "node\_modules" **folder** (include its path)
  + For the frontend service, **mount** the "src" **folder** of the frontend **service** to the "src" **folder** of the backend **service** (include folders' paths)
* The **three containers** should be in a **common network.**

**NOTE: Don't forget that you have to modify your connection string.**

### Requirements

Rename your two Dockerfiles to:

* Dockerfile-fe
* Dockerfile-be

If you don't rename them, there is a chance to overwrite one of the files with the other!

Provide the three files – the two Dockerfiles and the docker-compose.yml file, as well as a screenshoot of the running application.

Place all of the files in a folder named **{username}-task-1**, where **username** is your **SoftUni** username.

## Deploy an ASP.NET Core MVC app to Azure via Terraform

You are provided with a .NET application that consists of two projects – one for the **web application** and one for the **SQL Server database**.

Your task is to deploy the app to Azure via Terraform. Deploy the app using four Terraform configuration files – **main.tf**, **variables.tf**, **values.tfvars**, **outputs.tf**.

### Requirements

Provide the Terraform configuration files and an **screenshot** of the deployed app.

Place all of the files in a folder named **{username}-task-2**, where **username** is your **SoftUni** username.

## BONUS: Set up App Monitoring

Set up monitoring for the deployed .NET app in Azure. You should follow the steps and instructions below.

#### Set up Prometheus and Blackbox Exporter

Set the following configurations in the **prometheus-exam.yml file:**

* Scrape the target every 15 seconds
* Metrics should be accessed on **/probe**

#### Set up AlertManager

Set the following configurations in the **alertmanager-exam.yml file:**

* Set the timeout for alert resolution for 1 minute
* Specify the **webhook\_receiver (use the** [**web.hook**](https://webhook.site/) **website)**
* Specify that the alerts are sent to the **webhook\_receiver**
* Configure the alerting rules

Don't forget to change the configurations in the **prometheus-exam.yml** file

#### Set up Grafana

Add a **Prometheus Data Source** in Grafana.

Create a Grafana Dashboard and create a **histogram** for the **HTTP probe duration metric**, then export the Grafana **dashboard** as a **JSON** file.

### Requirements

Provide the **prometheus-exam.yml** and **alertmanager-exam.yml** configuration files and the **JSON export** file from **Grafana**.

Place all of the files in a folder named **{username}-task-3**, where **username** is your **SoftUni** username.

## Submission

Add all of the folders in an archive (**.zip**, **.rar**, **.7z**) and upload it to the **SULS** system.