# Containers and Cloud Exam Preparation

Problems for the exam for the ["Containers and Cloud"](https://softuni.bg/trainings/4332/containers-and-cloud-september-2023) course @ SoftUni

Your task is to deploy an app to Azure via Docker and Terraform and optionally – set up app monitoring and alert notification via Prometheus, AlertManager and Grafana.

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

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

### Steps

Your task is to deploy the app to Azure via Docker by executing the following steps:

#### Build a Custom Image ( pts.)

Create a **Dockerfile** in the **web app root directory** in the solution. The Dockerfile must contain **four** stages:

##### Base

This is the initial stage. Use the "**mcr.microsoft.com/dotnet/aspnet:6.0**" image. Expose ports **80** and **443** and set the working directory to **/app**.

##### Build

Use the "**mcr.microsoft.com/dotnet/sdk:6.0**" image. Set the working direcroty to **/src**. You should copy the **Homies** and **Homies.Data** projects into the container. Restore the packages, required by the **Homies** application. Copy all of the files into the current working directory. Build the application in **Release** mode. Don't forget to set the directory for the build artifacts.

##### Publish

Publish the application in **Release** mode. Set the working directory to **/app/publish**. Don't forget to exclude the **AppHost** when publishing.

##### Final

Set the working directory to **/app**. Copy the output from the "**Publish**" stage. Set the entrypoint for the container and the executable file.

**NOTE: Feel free to use the built-in Docker support of Visual Studio.**

#### Orchestrate Containers (pts.)

Create a **docker-compose.yaml** file. It should contain the version of the file, definitions of the **two services**: for the **database** and for the **web** **app** and **volumes** definition.

##### Database service

The file should contain:

* **container name**
* **image**
* **exposed ports: 1433:1433**
* **deployment configuration**
  + **resource reservations**
    - **CPUs: 2**
    - **Memory: 2GB**
* **environment variables**
* **volumes**

##### Web app service

The file should contain:

* **container name**
* **build context**
  + the **dockerfile path** for building the container
* **image**
* **exposed ports: 80:80**
* **restart policy**

##### Volumes

* Volume: **sqldata**
  + **Volume driver to use**
  + Additional volume driver options
    - **Share name**: **sql-volume**
    - **Storage account name: homiesstorageacc**

#### Create Azure Container Registry (pts.)

Create a resource group and a container registry.

##### Resource group

Name: **{username}homiessrg**

##### Container registry

Name: **{username}homiesscr**

#### Push Image to Azure Container Registry (pts.)

Push the image to your Azure container registry.

#### Create Azure Context (pts.)

Create an ACI context to associate Docker with your Azure subscription and resource group.

#### Deploy App to to Azure Container Instances (pts.)

Start the application in the Azure Container Instance.

#### Run the App in Azure (pts.)

Run the IP in Azure using the IP address of the application.

### Requirements

Provide the **Dockerfile** and the **docker-compose.yaml** files.

Provide **images** of the **Resource Group**, the **Container registry** and the **Container Instances** from Azure Portal, and from the running in a browser app.

Place all of the files in a folder named **{username}-docker-exam**.

## 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**.

### Steps

Your task is to deploy the app to Azure via Terraform by executing the following steps:

#### Create Azure Resource Group (pts.)

Create a **Terraform** configuration to deploy an **Azure** **resource** **group**.

#### Create App Service Plan (pts.)

Configure the Terraform configuration file.

#### Write and Apply a Terraform Configuration (pts.)

Configure the Terraform configuration file.

#### Separate Configuration to Multiple Files (pts.)

Separate the Terraform configuration file to multiple files:

##### main.tf

This should be the main Terraform configuration file.

##### variables.tf

This file should contain the variable declarations.

##### values.tfvars

This file should contain the values for the variables.

##### outputs.tf

This file should contain the output declarations.

#### Apply Configuration (pts.)

Deploy the app.

### Requirements

Provide the Terraform configuration files and an image of the deployed app.

Place all of the files in a folder named **{username}-terraform-exam**.

## BONUS: Set up App Monitoring

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

#### Set up Prometheus and Blackbox Exporter (pts.)

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 (pts.)

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 (pts.)

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}-monitor-exam**.

## Submission

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