# Solution M3: Kubernetes

Main goal is to build further on what was demonstrated during the practice

## Task 1

The following sequence of actions can be used to solve the task:

* Create an empty folder, for example **C:\HW-3-1**
* Download the **site** folder from <https://github.com/shekeriev/two-docker-images>
* Start the **minikube** with (or another driver)

**minikube start --driver virtualbox**

* Copy the **site** folder to the **minikube** virtual machine (default password is tcuser)

**scp -r site docker@<minikube-ip>:.**

* Create a **main.tf** file on the host with the following content:

provider "kubernetes" {

  config\_path = "~/.kube/config"

}

resource "kubernetes\_pod" "dob-mysql" {

  metadata {

    name = "dob-mysql"

    labels = {

      App = "mysql"

    }

  }

  spec {

    container {

      image = "shekeriev/dob-w3-mysql"

      name  = "dob-mysql"

      env {

        name  = "MYSQL\_ROOT\_PASSWORD"

        value = "12345"

      }

      port {

        container\_port = 3306

      }

    }

    hostname = "dob-mysql"

  }

}

resource "kubernetes\_pod" "dob-php" {

  metadata {

    name = "dob-php"

    labels = {

      App = "php"

    }

  }

  spec {

    container {

      image = "shekeriev/dob-w3-php"

      name  = "dob-php"

      port {

        container\_port = 80

      }

      volume\_mount {

        mount\_path = "/var/www/html/"

        name       = "site-data"

      }

    }

    hostname = "dob-php"

    volume {

      name = "site-data"

      host\_path {

        path = "/home/docker/site"

      }

    }

  }

}

resource "kubernetes\_service" "dob-mysql" {

  metadata {

    name = "dob-mysql"

  }

  spec {

    selector = {

      App = "mysql"

    }

    port {

      port = 3306

    }

  }

}

resource "kubernetes\_service" "dob-php" {

  metadata {

    name = "dob-php"

  }

  spec {

    selector = {

      App = "php"

    }

    port {

      port        = 80

      target\_port = 80

      node\_port   = 30001

    }

    type = "NodePort"

  }

}

* Save and close the file
* Initialize the provider

**terraform init**

* Check what will be created

**terraform plan**

* Run the solution with

**terraform apply**

* In a browser tab navigate to http://<minikube-ip>:30001
* Destroy the environment

**terraform destroy**

The files are provided as a complementary compressed archive

## Task 2

The following sequence of actions can be used to solve the task:

* Create an empty folder, for example **C:\HW-3-2**
* Create a set of YAML files in **C:\HW-3-2** for every component of the solution
* First of the files could be named **web-pod.yml** and with the following content:

apiVersion: v1

kind: Pod

metadata:

  name: hw-web-pod

  labels:

    app: hw-web

spec:

  containers:

  - name: hw-web-container

    image: shekeriev/dob-w3-php

    ports:

    - containerPort: 80

    volumeMounts:

    - mountPath: /var/www/html/

      name: web-site

  volumes:

  - name: web-site

    hostPath:

      path: /home/docker/site

      type: Directory

* Next, we must create a service file named **web-svc.yml** with the following content:

apiVersion: v1

kind: Service

metadata:

  name: hw-web-svc

  labels:

    app: hw-web

spec:

  type: NodePort

  ports:

  - port: 80

    nodePort: 30001

    protocol: TCP

  selector:

    app: hw-web

* Then, we could create the database related file named **db-pod.yml** with the following content:

apiVersion: v1

kind: Pod

metadata:

  name: hw-db-pod

  labels:

    app: hw-db

spec:

  containers:

  - name: hw-db-container

    image: shekeriev/dob-w3-mysql

    env:

    - name: MYSQL\_ROOT\_PASSWORD

      value: "12345"

    ports:

    - containerPort: 3306

* And finally, we must create the database service **db-svc.yml** with the following content:

apiVersion: v1

kind: Service

metadata:

  name: dob-mysql

  labels:

    app: hw-db

spec:

  ports:

  - port: 3306

    protocol: TCP

  selector:

    app: hw-db

* We must ensure that the **site** folder is present on the **Kubernetes** node
* Then, we can test the solution so far
* It should be working. Delete the pods and the services created during the test
* Now, it is time to wrap each pod definition into deployment and then store them in a single **deployment.yaml** file with the following content:

apiVersion: apps/v1

kind: Deployment

metadata:

  name: hw-web-deployment

  labels:

    app: hw-web

spec:

  replicas: 1

  selector:

    matchLabels:

      app: hw-web

  template:

    metadata:

      labels:

        app: hw-web

    spec:

      containers:

      - name: hw-web-container

        image: shekeriev/dob-w3-php

        ports:

        - containerPort: 80

        volumeMounts:

        - mountPath: /var/www/html/

          name: web-site

      volumes:

      - name: web-site

        hostPath:

          path: /home/docker/site

          type: Directory

---

apiVersion: apps/v1

kind: Deployment

metadata:

  name: hw-db-deployment

  labels:

    app: hw-db

spec:

  replicas: 1

  selector:

    matchLabels:

      app: hw-db

  template:

    metadata:

      labels:

        app: hw-db

    spec:

      containers:

      - name: hw-db-container

        image: shekeriev/dob-w3-mysql

        env:

        - name: MYSQL\_ROOT\_PASSWORD

          value: "12345"

        ports:

        - containerPort: 3306

* Similar approach we will apply to two services, but will store them in **service.yaml** file:

apiVersion: v1

kind: Service

metadata:

  name: hw-web-svc

  labels:

    app: hw-web

spec:

  type: NodePort

  ports:

  - port: 80

    nodePort: 30001

    protocol: TCP

  selector:

    app: hw-web

---

apiVersion: v1

kind: Service

metadata:

  name: dob-mysql

  labels:

    app: hw-db

spec:

  ports:

  - port: 3306

    protocol: TCP

  selector:

    app: hw-db

* Test that the solution is still working by deploying both files **deployment.yaml** and **service.yaml**
* Once, that you are sure that everything is working as expected, remove the deployed solution, and continue
* Create a basic chart by executing

**helm create homework3**

* Copy the two **YAML** files (**deployment.yaml** and **service.yaml)** to the **templates** subfolder and overwrite the existing ones
* Install our template with

**helm install hw-test homework3**

* Check that everything is deployed with

**kubectl get pods,svc**

* Open a browser tab and check if the application is working (kubectl port-forward homework3-xxxxxxxxx-yyyy 8080:80 and open localhost)
* Uninstall the chart with

**helm uninstall hw-test**

Of course, even though our chart is working, it is far from what is expected from a well created chart

There is much more to be done here

More information can be found here: <https://helm.sh/docs/chart_template_guide/>