**Kubernetes a practical example**

Project Description:

It is an online application that can calculate the Fibonacci sequence

The application consists of 5 docker components:

1. Client: It is a Reactjs application for user interaction (see the code :Client/src)
2. Server: API server to read the data from the DB (Redis or Postgree) auszulesen (see the code Server/index.js)
3. Worker: Responsible for the Fibonacci sequence calculation (see the code: Worker/index.js)
4. Redis Docker container
5. Postgres Docker Container

As an alternative to client, server and worker you can use the 3 Docker images:

stephengrider/multi-client <https://hub.docker.com/r/stephengrider/multi-client/>

stephengrider/multi-server <https://hub.docker.com/r/stephengrider/multi-server/>

stephengrider/multi-worker <https://hub.docker.com/r/stephengrider/multi-worker/>

To operate this microservice application, we are facing a Big challenge:

We must …

… Start container, stop container, monitoring container and update container

… Document the deployment steps

… manage the access data.

… manage the volume.

…expose the Applicatios.

… manage the network traffic.

… scale the applications up.

… monitor all containers and services.

…

And here comes kubernetes into play, all the challenges mentioned above can be orchestrated by Kubernetes.

Let's start with the basics:

In order to start play with Kubernetes, the Minikube tool from Google is suitable.

Minikube is a Kubernetescluster consisting of one node for the development. It is available on windows, mac and linux. <https://kubernetes.io/docs/tasks/tools/install-minikube/>

The most important terms at kubernetes

1. Pods: A grouping of more than one container

Now let's create our first pods. To create a Pods we have 2 options.

1. Imperative mode „kubectl run multi-client --image=stephengrider/multi-client --port=3000“

To validate the Pods creation, enter the following command „kubectl get pods“

1. Declarative mode using yaml file „infrastructure as code“

**apiVersion: apps/v1**

**kind: Deployment**

**metadata:**

**name: client-deployment**

**spec:**

**replicas: 1**

**selector:**

**matchLabels:**

**component: web**

**template:**

**metadata:**

**labels:**

**component: web**

**spec:**

**containers:**

**- name: client**

**image: stephengrider/multi-client**

**ports:**

**- containerPort: 3000**

to excute the desired pods on the yaml file we shoud run the folowing command

„kubectl apply -f .\client-deployment.yaml“

1. Service: An abstract way to expose an application running on a set of Pods as a network service.

To acces to our the **multi-client** Container we muss expose the application using the service

**apiVersion: v1**

**kind: Service**

**metadata:**

**name: client-cluster-ip-service**

**spec:**

**type: ClusterIP**

**selector:**

**component: web**

**ports:**

**- port: 3000**

**targetPort: 3000**

to expose the application we shoud run the folowing command

„kubectl apply -f .\client-cluster-ip-service.yaml“

1. **Replicas:** to define the number off the pods
2. ingress-service: exposes HTTP and HTTPS routes from outside the cluster to services within the cluster. Traffic routing is controlled by rules defined on the Ingress resource.

the yaml files for our Fibonacci sequence calculator project are in the ks8 folder.

for every component we have 2 yaml file 1 one to deploy the container and the second one to expose the application.

* Client component: we have client-deployment.yaml and client-cluster-ip-service.yaml
* Server component. server-deployment.yaml & server-cluster-ip-service.yaml
* Redis Container: redis-deployment.yaml & server-cluster-ip-service.yaml
* Postgres Container: postgres-deployment.yaml & postgres-cluster-ip-service.yaml
* Worker component: worker-deployment.yaml file
* to manage the network trafic we need ingress-service.yaml file
* to manage the Volume we need database-persistent-volume-claim.yaml file
* to manage the Postgres database secret we need to run this command : „kubectl create secret generic pgpassword --from-literal PGPASSWORD=12345asdf“

let now deploy our Fibonacci sequence calculator project.

We can do it in der steps

1. „kubectl create secret generic pgpassword --from-literal PGPASSWORD=12345asdf“
2. kubectl apply -f .\

to access the application just run „minikube ip“ to egt the cluster ip address

& navigate tot he this cluster adress with your browser

you need to refresh the page after submit.

Thanks and have fun by trying