UNIVERSITY OF APPLIED SCIENCES UPPER AUSTRIA

Continuous Delivery in agile Software Development

Exercise 05 (accompanying Chapter „Technologies, Tools & Frameworks“)

FH-Prof. DI Dr. Marc Kurz

HAGENBERG | LINZ | STEYR | WELS

Information & Prerequisites

- In this exercise, you will create your own Kubernetes cluster hosted locally via minikube (https://minikube.sigs.k8s.io/docs/)

minikube quickly sets up a local Kubernetes cluster on macOS, Linux, and Windows. We proudly focus on helping application developers and new Kubernetes users.

```
~
~ time minikube start
😄 minikube v1.13.0 on Darwin 10.15.6
✨ Using the docker driver based on user configuration
👍 Starting control plane node minikube in cluster minikube
🔥 Creating docker container (CPUs=2, Memory=3892MB) ...
🐳 Preparing Kubernetes v1.19.0 on Docker 19.03.8 ...
🔎 Verifying Kubernetes components...
🌟 Enabled addons: default-storageclass, storage-provisioner
💡 kubectl not found. If you need it, try: 'minikube kubectl -- get pods -A'
🏄 Done! kubectl is now configured to use "minikube" by default

_____________________________________________________
Executed in   23.96 secs   fish           external
   usr time    1.66 secs  237.00 micros    1.66 secs
   sys time    0.78 secs  943.00 micros    0.78 secs

~
```

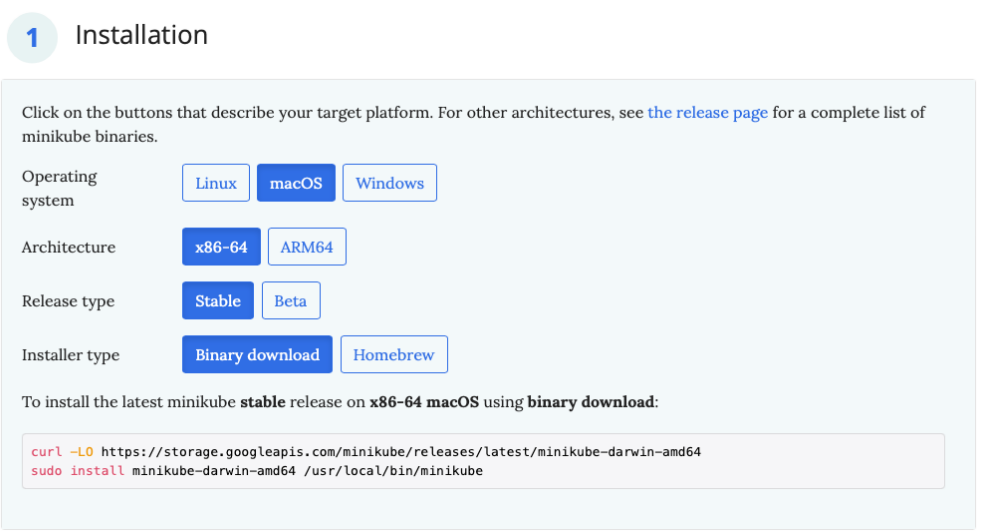
🎉 Latest Release: v1.30.1 - Apr 04, 2023 (changelog)

Instructions (Part 1) - Minikube

- Install Minikube on your computer (see https://minikube.sigs.k8s.io/docs/start/)

- minikube is local Kubernetes, focusing on making it easy to learn and develop for Kubernetes.
- All you need is Docker or a Virtual Machine environment, and Kubernetes is a single command away:

```
minikube start
```



Instructions (Part 1) - Minikube

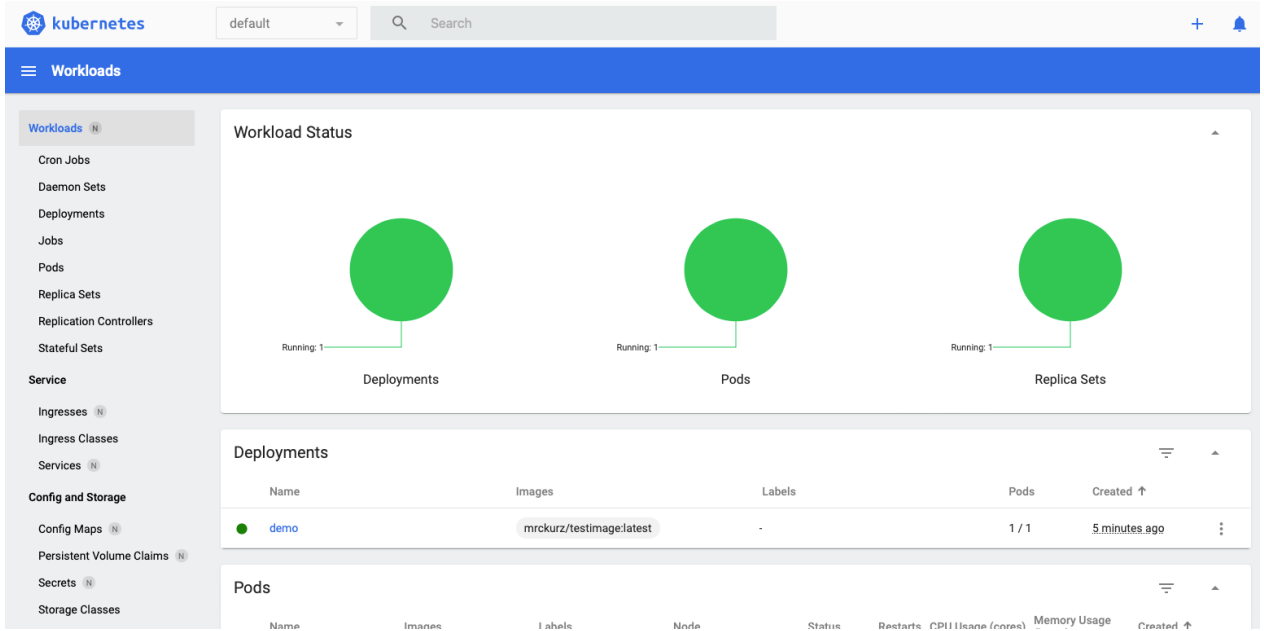
- Start your cluster

```
minikube start
```

```
Last login: Wed May  3 18:54:29 on ttys001
marckurz@Marcs-MacBook-Air ~ % minikube start
😄  minikube v1.30.1 on Darwin 13.3.1
✨  Using the docker driver based on existing profile
👍  Starting control plane node minikube in cluster minikube
🚜  Pulling base image ...
🔄  Restarting existing docker container for "minikube" ...
🐳  Preparing Kubernetes v1.26.3 on Docker 23.0.2 ...
🔗  Configuring bridge CNI (Container Networking Interface) ...
🔎  Verifying Kubernetes components...
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
    ▪ Using image docker.io/kubernetesui/dashboard:v2.7.0
    ▪ Using image docker.io/kubernetesui/metrics-scraper:v1.0.8
💡  Some dashboard features require the metrics-server addon. To enable all features please run:

        minikube addons enable metrics-server


🌟  Enabled addons: storage-provisioner, default-storageclass, dashboard

❗  /usr/local/bin/kubectl is version 1.24.0, which may have incompatibilities with Kubernetes 1.26.3.
    ▪ Want kubectl v1.26.3? Try 'minikube kubectl -- get pods -A'
🏄  Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

Instructions (Part 1) - Minikube

- To open the minikube dashboard in your browser use `minikube dashboard`



Description Part 2

- In this part, you will deploy a container in a Kubernetes cluster
- Thus, a pod will be created that runs the container.
- To access the running container, a port forward to a port on your local machine will be conducted.

- **Requirements**
  - > kubectl is connected to the Kubernetes cluster created in the previous part 1
  - > use one of the Docker containers that have as been created and pushed to DockerHub in a previous exercise (e.g. the webserver displaying the current time and date...)

Instructions (Part 2)

- In Kubernetes, the equivalent to docker container run is kubectl run. Use this command to run your container:
  > `kubectl run demo --image=mrckurz/testimage:latest --port=5000 --labels=app=demo`
- To verify that the container started and the app is running, use:
  > `kubectl get pods --selector app=demo`
- To forward your local port **9999** to the container port **5000**, use:
  > `kubectl port-forward demo 9999:5000`
- 🔍 Now, access the website http://localhost:9999. What is shown there?
- To delete the pod, use:
  > `kubectl delete pod demo`

Description Part 3

- In this part, you will specify a deployment (supervisor) for your pod by defining a deployment manifest (*deployment.yaml*).
- This manifest will be applied to your Kubernetes cluster to create a deployment resource, which automatically creates a pod.
- This pod is then running your container.
- Finally, you will delete the pod managed by a deployment in order to see that Kubernetes takes care of creating it again.

Instructions (Part 3)

- Create a file *deployment.yaml* with the following content and insert your DockerHub account details:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: demo
spec:
  replicas: 1
  selector:
    matchLabels:
      app: demo
  template:
    metadata:
      labels:
        app: demo
    spec:
      containers:
      - name: demo
        image: YOU-DOCKERHUB-ACCOUNT/testimage:latest
        ports:
        - containerPort: 5000
```

Instructions (Part 3)

- Apply the *deployment* to your Kubernetes cluster using:

  ```
  > kubectl apply -f deployment.yaml
  ```

- You can see all your active deployments (in your current namespace) by executing:

  ```
  > kubectl get deployments
  ```

  ```
  $ kubectl get deployments
  NAME   READY   UP-TO-DATE   AVAILABLE   AGE
  demo   0/1     1            0           31s
  ```

- To get more information on the demo deployment use:

  ```
  > kubectl describe deployment demo
  ```

- To forward your local port **9999** to the container port **8888**, use:

  ```
  > kubectl port-forward deployment/demo 9999:8888
  ```

Instructions (Part 3)

- 🔍 Now, access the website http://localhost:9999. What is shown there?
- Query the *pods* of your *deployment* with:

  ```
  > kubectl get pods
  ```

  ```
  NAME                    READY   STATUS    RESTARTS   AGE
  demo-bd8fd986b-as7dmf   1/1     Running   0          2m
  ```

- Delete the *pod* with:

  ```
  > kubectl delete pod --selector app=demo
  ```

- Now, query the *pods* of your *deployment* again:

  ```
  > kubectl get pods
  ```

- 🔍 What is your observation?

Description Part 4

- In this exercise, you will specify a service for your deployment by defining a service manifest (service.yaml).
- This manifest will be applied to your Kubernetes cluster to create a service resource.
- When the service is available, you can retrieve its *LoadBalancer Ingress* IP to access your demo app via a web browser.

Instructions (Part 4)

- Create a file *service.yaml* with the content shown on the right
  - The ports-section specifies that the Service forwards all traffic on port 80 to the Pod's port on 5000.
  - The selector tells the Service to which Pods the requests should be routed. Here, the requests are forwarded to any Pods matching the label app: demo.

```yaml
apiVersion: v1
kind: Service
metadata:
  name: demo
  labels:
    app: demo
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 5000
  selector:
    app: demo
  type: LoadBalancer
```

Instructions (Part 4)

- Apply the *service* to your Kubernetes cluster using:

  > `kubectl apply -f service.yaml`

- Your service now acts as *LoadBalancer* and received an externally accessible IP address. You can obtain this IP with:

  > `kubectl describe service demo`

```
marckurz@Marcs-MacBook-Air ex05_Kubernetes % kubectl describe service demo
Name:                     demo
Namespace:                default
Labels:                   app=demo
Annotations:              <none>
Selector:                 app=demo
Type:                     LoadBalancer
IP Family Policy:         SingleStack
IP Families:              IPv4
IP:                       10.98.219.36
IPs:                      10.98.219.36
Port:                     <unset>  80/TCP
TargetPort:               5000/TCP
NodePort:                 <unset>  32581/TCP
Endpoints:                10.244.0.15:5000
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
```

Instructions (Part 4)

- Use the IP address of the *LoadBalancer Ingress* to access your website.
- 🔍 Enter the IP in a browser.

Description Part 5

- In this exercise, you will create a namespace for your Kubernetes resources (i.e., deployments, services, pods).
- Afterwards, you will modify the deployment manifest from part 2, to create a deployment in this particular namespace.

Instructions (Part 5)

- Create a file *namespace.yaml* with the following content:

```yaml
apiVersion: v1
kind: Namespace
metadata:
  name: demo-environment
```

- Apply the *namespace* to your Kubernetes cluster using:
  > `kubectl apply -f namespace.yaml`
  > This will create the namespace demo-environment where you can add Kubernetes resources (i.e., deployments, services, pods) to.

Instructions (Part 5)

- Query all your namespaces with the following command and find your newly created namespace:

  > `kubectl get namespaces`

```
$ kubectl get namespaces
NAME                STATUS   AGE
default             Active   63m
demo-environment    Active   40s
kube-node-lease     Active   63m
kube-public         Active   63m
kube-system         Active   63m
```

Instructions (Part 5)

- Modify the deployment from part 3 so that the demo app will be deployed in the demo-environment namespace. Therefore, extend the metadata as follows:

```yaml
metadata:
  name: demo
  namespace: demo-environment
```

- Apply the modified deployment by executing:

```
> kubectl apply -f deployment.yaml
```

- Check the resources in the demo-environment namespace with:

```
> kubectl get deployments --namespace demo-environment
> kubectl get pods --namespace demo-environment
```

Exercise Submission

- Hand in the zip archive containing your results in form of a comprehensive documentations about your work (including screenshots) via Moodle no later than **May 16th, 23:55**

UNIVERSITY OF APPLIED SCIENCES UPPER AUSTRIA

Continuous Delivery in agile Software Development

Exercise 05 (accompanying Chapter „Technologies, Tools & Frameworks“)

FH-Prof. DI Dr. Marc Kurz

HAGENBERG | LINZ | STEYR | WELS