Technical University  
of Applied Sciences  
Würzburg-Schweinfurt

Faculty of Computer Science and Business Information Systems

Würzburg, on  
11.05.2023

# Cloud Native Computing

**Exercise: Kubernetes**

**3.5 Basics Kubernetes Instances: configmap**

Run the following commands and observe its result to learn more about – configmap

**-- Recreate the cluster --**

Create a YAML file named "nginx-configmap.yaml" that create the configmap with YAML configuration here and apply it.

**apiVersion: v1**

**kind: ConfigMap**

**metadata:**

**name: nginx-config**

**namespace: default**

**data:**

**index.html: |**

**"Welcome to Course: Kubernetes !"**

Apply nginx-configmap.yaml to create configmap

* **kubectl apply -f nginx-configmap.yaml**

Show all configmaps in the cluster

* **kubectl get configmap**

Display the value in the configmap we created

* **kubectl describe configmap nginx-config**
* **kubectl get configmap nginx-config -o yaml**

Create a pod using our newly created configmap with this YAML configuration. Save it under the file "nginx-pod-with-configmap.yaml" and apply it.

**apiVersion: v1**

**kind: Pod**

**metadata:**

**name: nginx-pod**

**spec:**

**containers:**

**- name: nginx-container**

**image: nginx**

**volumeMounts:**

**- name: nginx-page**

**mountPath: /usr/share/nginx/html/**

**volumes:**

**- name: nginx-page**

**configMap:**

**name: nginx-config**

**restartPolicy: Never**

Apply nginx-pod-with-configmap.yaml to create a pod

* **kubectl apply -f nginx-pod-with-configmap.yaml**

Check if the Nginx pod is already started

* **kubectl get pod nginx-pod**

Run "kubectl port-forward" to map port 80 of "nginx-pod" on our localhost at port 8080. Then use the URL "localhost:8080" in your web browser to see the response from nginx

* **kubectl port-forward nginx-pod 8080:80**
* **localhost:8080**

**3.6 Basics Kubernetes Instances: service**

Run the following commands and observe its result to learn more about – service

**-- Recreate the cluster --**

Create a deployment with nginx:alpine image named nginx-depl with 3 pods

* **kubectl create deployment nginx-depl --image nginx:alpine --replicas 3**

Generate a ClusterIP service for deployment named "nginx-svc" that has an IP address of 8080 and is mapped to port 80 of pods by creating a YAML file named "nginx-service-clusterip.yaml" with YAML configuration here and apply it.

**apiVersion: v1**

**kind: Service**

**metadata:**

**labels:**

**app: nginx-depl**

**name: nginx-svc**

**spec:**

**ports:**

**- port: 8080**

**protocol: TCP**

**targetPort: 80**

**selector:**

**app: nginx-depl**

**type: ClusterIP**

Apply nginx-service-clusterip.yaml to start "nginx-svc" service

* **kubectl apply -f nginx-service-clusterip.yaml**

View all services in the cluster

* **kubectl get service**

Display a detail of the nginx-svc service

* **kubectl describe service nginx-svc**

Run the kubectl exec command in the nginx-depl deployment pod and run the following curl command to see the output of Nginx (names from the pod). Now we show the response from the DNS and IP address of the service

* **kubectl exec -it <pod-name> -- sh**
* **curl nginx-svc:8080**
* **curl <Service-IP>:8080**

Delete the nginx-svc service

* **kubectl delete service nginx-svc**

Create a service for a pod in the nginx-depl deployment using the imperative command

* **kubectl expose deployment nginx-depl --name nginx-svc --port 8080 --target-port 80**
* **kubectl get service**

Delete the nginx-svc service

* **kubectl delete service nginx-svc**

Generate a NodePort service for the deployment named "nginx-svc" that has a Port of 8080 and is mapped to port 80 by deployment and a NodePort to port 32000 by creating a YAML file named "nginx-service-nodeport.yaml" with YAML configuration here and apply it.

**apiVersion: v1**

**kind: Service**

**metadata:**

**labels:**

**app: nginx-depl**

**name: nginx-svc**

**spec:**

**ports:**

**- port: 8080**

**protocol: TCP**

**targetPort: 80**

**nodePort: 32000**

**selector:**

**app: nginx-depl**

**type: NodePort**

Apply nginx-service-nodeport.yaml to start "nginx" service

* **kubectl apply -f nginx-service-nodeport.yaml**

Show all services in the cluster and show an IP address of nodes in the cluster

* **kubectl get service -o wide**
* **kubectl get node -o wide**

Use the URL "localhost:32000" in your web browser to see the response from nginx

* **localhost:32000**

Enter "<node-IP>:32000" and " <node-IP>:8080" in the web browser (e.g. Firefox). You should see the response from nginx only from port 32000

* **<node-IP>:32000"**

Delete the nginx-svc service

* **kubectl delete service nginx-svc**

Generate a LoadBalancer service named "nginx-svc" that has a Port of 8080 and is mapped to port 80 by nginx-deployment and a NodePort to port 32000 by creating a YAML file named "nginx-service-loadbalancer.yaml" with YAML configuration here and apply it.

**apiVersion: v1**

**kind: Service**

**metadata:**

**labels:**

**app: nginx-depl**

**name: nginx-svc**

**spec:**

**ports:**

**- port: 8080**

**protocol: TCP**

**targetPort: 80**

**nodePort: 32000**

**selector:**

**app: nginx-depl**

**type: LoadBalancer**

Apply nginx-service-loadbalancer.yaml to start "nginx" service

* **kubectl apply -f nginx-service-loadbalancer.yaml**

Show all services in the cluster and Show an IP address of nodes in the cluster

* **kubectl get service -o wide**
* **kubectl get node -o wide**

Enter "localhost:32000" and "localhost:8080" in the web browser (e.g. Firefox). You should see the response from nginx from both endpoints

* **localhost:32000**
* **localhost:8080**

Delete the nginx-svc service

* **kubectl delete service nginx-svc**

**3.7 Basics Kubernetes Instances: namespace**

Run the following commands and observe its result to learn more about – namespace

**-- Recreate the cluster --**

Show all namespaces in the cluster

* **kubectl get namespace**

Show all pods in each namespace

* **kubectl get pod --all-namespaces**

Create a pod named "nginx" with image "nginx" in namespace "default"

* **kubectl run nginx --image nginx**

Create a "myapp" namespace

* **kubectl create namespace myapp**

Create a second pod named nginx with image nginx in namespace myapp.

* **kubectl run nginx --image nginx -n myapp**

When we show a pod without specifying a namespace, only pods in the "default" namespace are shown

* **kubectl get pod**

To show pods in the "myapp" namespace, we can run:

* **kubectl get pod -n myapp**
* **kubectl get pod --all-namespaces**

Create a service for an nginx pod in the myapp namespace and an nginx pod in the default namespace

* **kubectl expose pod nginx --name nginx-myapp-svc --port 8080 --target-port 80 -n myapp**
* **kubectl expose pod nginx --name nginx-svc --port 8080 --target-port 80**

Test which service is reachable from the pod "nginx" in the namespace "default".

* **kubectl exec -it nginx -- curl nginx-myapp-svc:8080 ## not reachable**
* **kubectl exec -it nginx -- curl nginx-myapp-svc.myapp:8080 ## reachable**
* **kubectl exec -it nginx -- curl nginx-svc:8080 ## reachable**

**3.8 Basics Kubernetes Instances: PV and PVC**

Run the following commands and observe its result to learn more about – PV and PVC

**-- Recreate the cluster --**

Create a PV "nginx-pv-volume" using 1 Gi storage on our local machine at "/mnt/data" path by creating a YAML file named "nginx-pv.yaml" with YAML configuration here and apply it.

**apiVersion: v1**

**kind: PersistentVolume**

**metadata:**

**name: nginx-pv-volume**

**labels:**

**type: local**

**spec:**

**storageClassName: local-path**

**capacity:**

**storage: 1Gi**

**accessModes:**

**- ReadWriteMany**

**hostPath:**

**path: "/mnt/data"**

Apply nginx-pv.yaml to start "nginx" service

* **kubectl apply -f nginx-pv.yaml**

Show PV in cluster. The status of PV should be "available".

* **kubectl get pv**

Create a PVC named "nginx-pv-claim" that takes up 500 Mi from a PV by creating a YAML file named "nginx-pvc.yaml" with YAML configuration here and apply it.

**apiVersion: v1**

**kind: PersistentVolumeClaim**

**metadata:**

**name: nginx-pv-claim**

**spec:**

**storageClassName: local-path**

**accessModes:**

**- ReadWriteMany**

**resources:**

**requests:**

**storage: 500Mi**

Apply nginx-pvc.yaml to start "nginx" service

* **kubectl apply -f nginx-pvc.yaml**

Show PVC in cluster. The status of PVC should be "Pending". Even though PV has already been created, PVC remains in pending state waiting for some pods to use it

* **kubectl get pvc**

Create a nginx image deployment named nginx-depl with the following YAML file by creating a YAML file named "nginx-deployment-pvc.yaml" with YAML configuration here and apply it.

**apiVersion: apps/v1**

**kind: Deployment**

**metadata:**

**name: nginx-depl**

**labels:**

**app: nginx**

**tier: frontend**

**spec:**

**replicas: 3**

**selector:**

**matchLabels:**

**tier: frontend**

**template:**

**metadata:**

**labels:**

**tier: frontend**

**spec:**

**volumes:**

**- name: nginx-pv-storage**

**persistentVolumeClaim:**

**claimName: nginx-pv-claim**

**containers:**

**- name: nginx**

**image: nginx**

**volumeMounts:**

**- mountPath: "/usr/share/nginx/html"**

**name: nginx-pv-storage**

Apply nginx-deployment-pvc.yaml to start "nginx" service

* **kubectl apply -f nginx-deployment-pvc.yaml**

Show PVC in cluster. The status of PVC is now "Bound" because the pod is using this PVC.

* **kubectl get pvc**

The status of PV will also be "Bound" as the PVC use this PV.

* **kubectl get pv**

**3.9 Basics Kubernetes Instances: StatefulSet**

Run the following commands and observe its result to learn more about – StatefulSet

**-- Recreate the cluster --**

Create a headless service for statefulset. We need to create it before statefulset by creating a YAML file named "headless-service.yaml" with YAML configuration here and apply it.

**apiVersion: v1**

**kind: Service**

**metadata:**

**name: nginx-svc-headless**

**labels:**

**app: myapp**

**spec:**

**ports:**

**- name: nginx-port**

**port: 80**

**clusterIP: None**

**selector:**

**app: myapp**

Apply headless-service.yaml to start "nginx" service

* **kubectl apply -f headless-service.yaml**

Show a detail of the nginx-svc-headless service. We will see that it has no IP address.

* **kubectl describe service nginx-svc-headless**

View which pods are connected to this service. We won't see a pod yet

* **kubectl describe endpoints nginx-svc-headless**

Create a StatefulSet with an "nginx" image using the following YAML file. We also need to specify a headless service and a PVC template by creating a YAML file named "nginx-statefulset.yaml" with YAML configuration here and apply it.

**apiVersion: apps/v1**

**kind: StatefulSet**

**metadata:**

**name: nginx-statefulset**

**spec:**

**selector:**

**matchLabels:**

**app: myapp**

**serviceName: nginx-svc-headless**

**replicas: 5**

**template:**

**metadata:**

**labels:**

**app: myapp**

**spec:**

**containers:**

**- name: myapp**

**image: nginx**

**volumeMounts:**

**- name: www**

**mountPath: /usr/share/nginx/html**

**volumeClaimTemplates:**

**- metadata:**

**name: www**

**spec:**

**accessModes: [ "ReadWriteOnce" ]**

**storageClassName: "local-path"**

**resources:**

**requests:**

**storage: 100Mi**

Apply nginx-statefulset.yaml to start "nginx" service

* **kubectl apply -f nginx-statefulset.yaml**

Show StatefulSet's pods by IP address. We will see how to name and create the pods. The pods are launched one at a time.

* **kubectl get pods -o wide**

View PV and PVC in the cluster. We will see that PV and PVC have been created for each pod.

* **kubectl get pv**
* **kubectl get pvc**

Now we see pods that were tied to the headless service.

* **kubectl describe endpoints nginx-svc-headless**

See a response from Nginx Pods with the name of Headless Service

* **kubectl exec -it nginx-statefulset-0 -- curl nginx-svc-headless**

View a response from Nginx pods directly using a pod's DNS

* **kubectl exec -it nginx-statefulset-0 -- curl nginx-statefulset-0.nginx-svc-headless**

Use the "kubectl exec" command to get into the pods and modify an Nginx HTML page.

* **kubectl exec -it nginx-statefulset-0 -- sh**
* **echo "hello from nginx-statefulset-0 !" > /usr/share/nginx/html/index.html**

View a response from Nginx pods directly using a pod's DNS. We'll see the text we just added

* **kubectl exec -it nginx-statefulset-0 -- curl nginx-statefulset-0.nginx-svc-headless**

We're trying to reduce the number of pods in the StatefulSet and show the order in which the pods are terminated. We will see that the pod with the highest number is terminated first.

* **kubectl scale statefulset nginx-statefulset --replicas 2**

View PV and PVC in the cluster. We will see that PV and PVC are not erased.

* **kubectl get pv**
* **kubectl get pvc**

Let's try deleting a pod and see how it gets recreated

* **kubectl delete pod nginx-statefulset-0**
* **kubectl get pod -o wide**

Replay a response from Nginx pods directly using a pod's DNS. We'll "still" see the text we just added

* **kubectl exec -it nginx-statefulset-0 -- curl nginx-statefulset-0.nginx-svc-headless**