Kubernetes in 6h- Hands ons

Hands on #1

```
kubectl cluster-info
kubectl get nodes
kubectl get nodes -o wide
```

Hands on #2

```
kubectl config view
```

Hands on #3

Running pod

```
kubectl run my-httpd --image=httpd --port=80 --restart=Never
kubectl get pods
kubectl port-forward my-httpd 80
# Check in browser: http://localhost:80
```

Accessing container console

```
kubectl -it exec my-httpd bash
echo "<html><body><h1>Sphere.it K8s workshop</h1></body></html>" > htdocs/index.html
exit
```

(Check `http://localhost:80`)

Accessing application logs

```
kubectl logs my-httpd
```

Information on pod

```
kubectl describe pod/my-httpd
kubectl get pods -o wide
```

Hands on #4

```
kubectl delete pod/my-httpd
kubectl get pods
```

Hands on #5

Running deployment

```
kubectl get deployments
kubectl run my-nginx --image=nginx:1.17.0 --port=80
kubectl get deployments
kubectl describe deployment/my-nginx
```

Killing pod belonging to deployment

```
kubectl get pods
kubectl delete pod/<podName>
kubectl get deployments
kubectl get pods
```

```
kubectl port-forward deployment/my-nginx 80
```

Scaling

```
kubectl scale deployments/my-nginx --replicas 3
kubectl get deployments
kubectl scale deployments/my-nginx --replicas 1
```

Updating version

```
kubectl exec <my-nginx-pod-name> -- nginx -v
kubectl set image deployments/my-nginx my-nginx=nginx:1.17.3
kubectl exec <my-nginx-pod-name> -- nginx -v
```

Rollbacking to previous version

```
kubectl rollout undo deployments/my-nginx
kubectl exec <my-nginx-pod-name> -- nginx -v
```

Hands on #6

Create service exposing existing deployment

```
kubectl expose deployment/my-nginx
kubectl get services
```

Check out load balancing

```
kubectl proxy
# in browser: http://localhost:8001/api/v1/namespaces/default/services/my-nginx/proxy/
kubectl -it exec <my-nginx-pod-name> bash
echo "<html><body><h1>Kubernetes in 6h workshop</h1></body></html>" >
/usr/share/nginx/html/index.html
exit
# refresh browser
kubectl scale deployments/my-nginx --replicas 3
# refresh browser multiple times
kubectl get pods
kubectl delete pod/<pod-name>
```

Create deployment and service at once

```
kubectl run another-nginx --image=nginx --replicas=4 --expose --port=80
```

What happens when we port-forward service?

```
kubectl port-forward service/my-nginx 80
```

Hands on #7

Print object details

```
kubectl get service/my-nginx -o yaml
kubectl get pod/<pod-name> -o yaml
```

Dry run of kubernetes command and viewing object details

```
kubectl run my-nginx --image=nginx --replicas=5 --expose --port=80 -o yaml --dry-run
```

Hands on #8

Task: prepare YAML file configuration for Jenkins master deployment

Details:

- see configuration of already exisitng deployments and adopt it for different Jenkins image
- Docker image: **jenkins**
- Use as name for deployment: `my-jenkins`
- Expose port 80

Then create my-jenkins deployment with following command:

```
kubectl create -f jenkins-master-deployment.yaml
```

Then:

- Update config file exposing also port 5000
- Update existing object with new configuration:

```
kubectl replace -f jenkins-master-deployment.yaml
```

Observe what happened

Hands on #8b

Task: prepare YAML file configuration for Jenkins master service

Details:

- see configuration of already exisitng **service** and adopt it for different Jenkins image
- Use as name for deployment: `my-jenkins`
- Expose port 80 and 5000
- Figure out how to refer to already created deployment

Then create my-jenkins service with following command:

```
kubectl create -f jenkins-master-service.yaml
```

Forward Jenkins http port to localhost

```
kubectl port-forward service/my-jenkins 8080
```

Open in browser
http://localhost:8080/
Complete initialization of Jenkins master
You can find initialization password in Jenkins-master logs. (How to get the logs? Check Hands on #3)

Hands on #9

Task: prepare YAML file configuration for Jenkins slave deployment

Details:

- Docker image: jenkins/jnlp-slave
- Via Jenkins-master UI, configure new slave to be started (guide)
- Familiarize yourself with container documentation, to find out what container arguments are needed
    - Master url needs to have http:// prefix
- Deployment configuration
    - Number of replicas: 1
    - Use as name for deployment: `my-jenkins-slave`
    - Pass container arguments either via .**spec.template.spec.containers.image.env** (example) or .**spec.template.spec.containers.image.args** (example)
- For debug reasons use **kubectl log**

Declaratively apply given configuration to cluster:

```
kubectl apply -f jenkins-jnlp-slave-deployment.yaml
```

Apply this configuration again and observer output

Hands on #10

Create configMap object using following command

```
kubectl create configmap jenkins-agent1-config --from-literal=JENKINS_SECRET=<secret>
--from-literal=JENKINS_AGENT_NAME=agent1
```

View created config

```
kubectl get configmap
kubectl get configmap/jenkins-agent1-config -o yaml
kubectl describe configmap/jenkins-agent1-config
```

Modify Jenkins jnlp slave YAML config file to use properties from this config map (see examples below)

Apply the changes:

```
kubectl apply -f jenkins-jnlp-slave-deployment.yaml
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    run: my-jenkins-slave
  name: my-jenkins-slave
spec:
  replicas: 1
  selector:
    matchLabels:
      run: my-jenkins-slave
  template:
    metadata:
      labels:
        run: my-jenkins-slave
    spec:
      containers:
        - image: jenkins/jnlp-slave
          name: my-jenkins-slave
          resources: {}
          env:
            - name: JENKINS_URL
              value: 'http://10.104.75.83:8080'
          envFrom:
            - configMapRef:
                name: jenkins-agent1-config
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    run: my-jenkins-slave
  name: my-jenkins-slave
spec:
  replicas: 1
  selector:
    matchLabels:
      run: my-jenkins-slave
  template:
    metadata:
      labels:
        run: my-jenkins-slave
    spec:
      containers:
        - image: jenkins/jnlp-slave
          name: my-jenkins-slave
          resources: {}
          args: [-url,http://10.104.75.83:8080,
$(JENKINS_SECRET), $(JENKINS_AGENT_NAME)]
          envFrom:
            - configMapRef:
                name: jenkins-agent1-config
```

Hands on #11

Task:

- Create Persistent Volume Claim (using cofing file and kubectl –f apply)
- Modify jenkins-master-service.yaml configuration to have persisted volume mounted under **/var/jenkins_home**
- Apply changes to Jenkins master using **kubectl –f apply**
- Execute initial setup again (explain why you need to do it again)
- Kill Jenkins master pod and refresh Jenkins UI – what should you expect?
- Redo Jenkins slave configuration steps for newly configured master

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-jenkins
spec:
  replicas: 1
  selector:
    matchLabels:
      run: my-jenkins
  template:
    metadata:
      labels:
        run: my-jenkins
    spec:
      volumes:
        - name: jenkins-storage
          persistentVolumeClaim:
            claimName: jenkins-pvc-dynamic
      containers:
        - image: jenkins
          name: my-jenkins
          ports:
            - containerPort: 8080
            - containerPort: 50000
          volumeMounts:
            - mountPath: "/var/jenkins_home"
              name: jenkins-storage
```

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: jenkins-pvc-dynamic
spec:
  storageClassName: standard
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 3Gi
```