# Solution M2: Cluster Setup and Management

# Task 1

**Challenge:**

Create a two node **Minikube**-based **Kubernetes** cluster and deploy a simple app on it. For example, **shekeriev/k8s-oracle** from the previous HW

**Solution:**

*Please note, that this solution requires access to a supported local virtualization solution*

To create a **minikube**-based two node **Kubernetes** cluster we must execute the following command

**minikube start --driver=hyperv --nodes=2**

Adjust the **--driver** option to match your situation. You can select one of the following **virtualbox**, **vmwarefusion**, **hyperv**, **vmware**, **docker**, or **ssh**. If not specified, the most suitable option will be auto detected and selected

Now, that we have our cluster, we can check if it is reachable

**kubectl cluster-info**

**kubectl get nodes -o wide**

Next, we can deploy a simple application on it (using the files from the previous HW)

**kubectl apply -f homework.yaml**

Then check the deployed resources

**kubectl get pods,svc -n homework -o wide**

And try to reach the application on http://<minikube-ip>:32000

**Minikube** **IP** address can be retrieved with

**minikube ip**

Or directly the service **URL** with

**minikube service list**

Once done, clean up the application with

**kubectl delete -f homework.yaml**

And then the **minikube** cluster with

**minikube delete**

# Task 2

**Challenge:**

Create a three node **KIND**-based **Kubernetes** cluster and deploy a simple app on it. For example, **shekeriev/k8s-oracle** from the previous HW

**Solution:**

*Please note, that this solution requires access to a local or remote* ***Docker*** *instance*

Prepare a cluster configuration manifest (**kind-cluster.yaml**) with the following content

kind: Cluster

apiVersion: kind.x-k8s.io/v1alpha4

nodes:

- role: control-plane

  extraPortMappings:

  - containerPort: 32000

    hostPort: 8080

- role: worker

- role: worker

Then start the cluster with

**kind create cluster --name hw --config kind-cluster.yaml**

Finally, switch the context to the new cluster

**kubectl config use-context kind-hw**

Ask for some information about the cluster

**kubectl cluster-info**

**kubectl get nodes -o wide**

Then spin up the application

**kubectl apply -f homework.yaml**

And check its components

**kubectl get pods,svc -n homework -o wide**

Try to reach the application on <http://localhost:8080>

Once done exploring, remove the application with

**kubectl delete -f homework.yaml**

And finally, remove the cluster with

**kind delete cluster --name hw**

# Task 3

**Challenge:**

Create a three-node vanilla **Kubernetes** cluster in a virtualization solution of your choice. You are free to use **Debian** (as demonstrated during the practice) or any other (from the same or another family) **Linux** distribution. Use **Vagrant** or a similar tool to do it in as much automated fashion as possible (***If you are new to Vagrant, don’t worry. Do the other two tasks and wait until the solution is published***)

**Solution:**

*Please note, that this solution requires access to a supported local virtualization solution.* ***VirtualBox*** *is the one selected for this. Also, you must have* ***Vagrant*** *installed*

Use the files (folder **task3**) provided with this solution

The main part is in the **Vagrantfile** file

There are several things to pay attention to:

* The selected virtual machine template image (**shekeriev/debian-10**). It is a generic one (just a bare installation). It may be changed to anything else (including to **shekeriev/debian-11**), but then the initialization scripts may need to be adjusted as well
* There are three initialization scripts (variables **$common**, **$k8scp**, **$k8swk**). The first one (**$common**) is applicable to all machines in the cluster as it is generic and prepares them for being part of the cluster. The second one (**$k8scp**) is just for the control plane node as it initializes the cluster (control plane) and installs a **POD Network** plugin. The third one (**$k8swk**) is applicable only to the worker nodes as it joins them to the cluster
* **Calico** is selected as a **POD Network** plugin. You may change it to one of the others, of course with the appropriate changes of the initialization script
* The version of **Kubernetes** components is controlled via a variable (**K8SVER**) in the first script (and then in the second). Set it to a particular version (for example, **1.21.6-00**) or **latest** (to install the latest available)
* The **Vagrantfile** itself may be further optimized and/or made more agile, but this is outside the scope of the current course

Once done with the adjustments, execute

**vagrant up**

To create the cluster and then

**vagrant ssh node1**

To open a session to **node 1**

There you may check the environment by executing

**kubectl cluster-info**

**kubectl get nodes -o wide**

And then, run the application

**kubectl apply -f /vagrant/homework.yaml**

And check its components

**kubectl get pods,svc -n homework -o wide**

Try to reach the application

**curl http://192.168.99.101:32000**

Once done exploring, remove the application with

**kubectl delete -f /vagrant/homework.yaml**

And finally, remove the cluster by closing the session and executing

**exit**

**vagrant destroy --force**