Setting up AiO-Kubernetes

This method is less common but offers the advantage that no nested virtualization is required. Also it gives nice insight in the workings of Kubernetes.

Notice that this procedure only is supported on Ubuntu 20.04 workstation and server. Support for CentOS 7 has been dropped and removed from this guide.

WARNING: you cannot copy paste from this document, but will have to type all commands.

1. Install Ubuntu 20.04 workstation or server, meeting the following minimal requirements
    a. 20 GB disk space
    b. 4 GB RAM recommended (2 GB minimal)
    c. 2 CPU's
    d. No Swap
    e. One ordinary user with sudo privileges must be present. In this document I'll assume the username "student". Change this according to your local setup.
2. Install some packages
    a. **sudo apt install git vim -y**
3. As ordinary user, clone the course Git repository that is required for the course you are taking
    a. **git clone https://github.com/sandervanvugt/kubernetes** for Kubernetes in 4 Hours
    b. **git clone https://github.com/sandervanvugt/ckad** for CKAD
    c. **git clone https://github.com/sandervanvugt/microservices** for Microservices
    d. **git clone https://github.com/sandervanvugt/cka** for CKA
4. Run the setup scripts using root privileges
    a. **cd /ckad** (or **cd /cka**) (or whichever GitHub repository you have cloned)
    b. **sudo ./setup-container.sh**
    c. **sudo ./setup-kubetools-ubuntu.sh**
5. Still from a root shell, install a Kubernetes master node
    a. **sudo kubeadm init --pod-network-cidr=10.10.0.0/16**
6. Everything from this point is done in a user shell. Set up the kubectl client:
    a. **cd ~**
    b. **mkdir .kube**
    c. **sudo cp -i /etc/kubernetes/admin.conf .kube/config**
    d. **sudo chown student:student .kube/config**
7. Set up the Calico networking agent
    a. **kubectl create -f https://docs.projectcalico.org/manifests/tigera-operator.yaml**
    b. **wget https://docs.projectcalico.org/manifests/custom-resources.yaml**

c. You now need to define the Pod network, which by default is set to 192.168.0.0/24, which in general is a bad idea. I suggest setting it to 10.10.0.0 - make sure this address range is not yet used for something else!
d. **sed -i -e s/192.168.0.0/10.10.0.0/g custom-resources.yaml**
e. **kubectl create -f custom-resources.yaml**
f. **kubectl get pods -n calico-system**: wait until all pods show a state of Ready, this can take about 5 minutes!

8. By default, user Pods cannot run on the Kubernetes control node. Use the following command to remove the taint so that you can schedule nodes on it:
   kubectl taint nodes --all node-role.kubernetes.io/master-
9. Type **kubectl get all** to verify the cluster works.