# Setup Guide

This guide was written as a support for the different Kubernetes courses that I'm teaching. This guide does not intend to show all possible solutions but covers a few options to use Kubernetes. Other options do exist, but I cannot support them in my courses. For questions: mail@sandervanvugt.nl

version 2.5, 20 August 2021: added Ubuntu-only update for Minikube based on Docker

## Setting up Minikube on a Linux virtual machine

Because of ongoing issues with Minikube on MacOS / VMware combination, I have developed a new procedure that works with Docker containers and is only supported on Ubuntu 20.04.

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

1. Make sure you have either a fully installed Ubuntu 20.04 (virtual) machine.
    a. 8 GB RAM recommended, 4 GB RAM minimal
    b. 2 CPU cores
    c. 20 GB minimal disk space
2. Install git, vim and bash completion
    a. On Ubuntu: **sudo apt install git vim**
3. As ordinary user with sudo privileges, clone the course Git repository
    a. **git clone https://github.com/sandervanvugt/kubernetes** for Kubernetes in 4 Hours
4. Change into the cloned git repo and run the **minikube-docker-setup.sh** script
    a. **cd kubernetes**
    b. **./minikube-docker-setup.sh**
5. When you see the message that the script is ready, manually run the following command:
    a. **minikube start --memory=4096 --vm-driver=docker**
6. Once Minikube has started successfully, you'll see a message that it has started. Type **kubectl get all** to verify the minikube cluster is up and running

## 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 currently only is supported on CentOS 7 with Docker. That does not mean that it doesn't work on Ubuntu, it just means that I haven't had time yet to figure it all out and give full support :-)

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

1. Install a Centos 7 (NOT 8); minimal installation will do. Theoretically, this works on Ubuntu as well but I haven't had the time yet to make my scripts Ubuntu compatible - hopefully soon.
    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. On CentOS: **sudo yum install git vim bash-completion wget**
3. As ordinary user, clone the course Git repository that is required for this course
    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-docker.sh**
    c. **sudo ./setup-kubetools.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.

## Using Hosted Kubernetes in GCE

Kubernetes is supported by many cloud providers. As I cannot be all-inclusive, but most of all, as Google has donated Kubernetes to the world, I like to do something back and explain the procedure on GCE only.

1. From a browser, go to **https://cloud.google.com** and click **Sign in**.
2. Click **Go to console**
3. Select **Compute > Kubernetes Engine > Clusters**
4. Click **Create Cluster**
5. Click **My first cluster**
6. Click **Create Now** and wait a few minutes - typically not more than 5.
7. Once the cluster appears, click **Connect**
8. You'll see a command **gcloud container ...** Click **Run in Cloud Shell** to run it. This will deploy a cloud shell machine, which may take one or two minutes.
   (NOTE: If you don't see the command, this is what you need to type in a cloud shell: **gcloud containers cluster get-credentials cluster-name --zone us-central1-c --project your-project-123**. Make sure the change the cluster name, project name as well as the zone to what applies to your environment.
9. Press Enter to run the command, and click **Authorize** to authorize the cloud shell
10. Type **kubectl get all** to check that the Kubernetes cluster works.