# Docker & Kubernetes workshop

## Install prerequisites

* Install Docker desktop
* Install minikube
* Create a DockerHub account.

<https://kubernetes.io/docs/tasks/tools/install-minikube/>

On win10, check that HyperV is installed. If not install virtuabox.

## Getting to know Docker basics

Most used commands:

* “Docker images” - list all downloded images in local repository.
* “docker ps” – list all running processes.
* “docker run” – instanciate a docker image.
* “docker kill” – Stop a docker instance.
* “docker exec” – run a command in a running docker instance.
* “docker build” – make a docker image.
* “docker tag” – tag a image.
* “docker push”- upload a docker image to a central repository.
* “docker system prune” – Clean up temp docker “layers” to free space.

### Running a container

1. In a commandline window – (powershell/cmd/sh/bash/etc.)
2. “docker run hello-world”
3. Notice, layers. Every image is built of x layers based on a base image.

### Attaching to a container

1. In a commandline window
2. “docker run –it ubuntu bash”

Now you have a fully functioning Ubuntu minimal installation running.

1. Type “exit” to exit to host system ( kills the container)

### Running a container detached

1. In a cmd.
2. “docker run –d nodered/node-red”
3. Echoes the process id name.
4. Notice that this node app is exposed through port 1880.
5. Also notice the folders.
6. Run “docker kill [process id]” to kill the running instance.

### Running with port mapping to host system

The container may contain services that might be reached from the host system. But these need to be exposed explicitly and mapped to a host port.

1. In a cmd.
2. Run “docker run –d nodered/node-red”
3. Echoes the process id name.
4. Notice that this node app is exposed through port 1880.
5. Also notice the folders.
6. Try to access the node application “localhost:1880”
7. This will not work because the port is not exposed to the host system.
8. Run “docker kill [process id]” to kill the running instance.
9. Run “docker run –d –p 1880:1880 nodered/node-red”
10. Try to access the node application “localhost:1880” again.
11. Now it works.
12. The format for the –p switch is :  
    “-p [hostport1]:[containerport1] [hostport2]:[containerport2]…….”.  
    Note: The order of the option switches must be before the other params.

### Running with folder mapping to host system

For earlier versions of Docker, the containers doesn’t persist their state. So anything done inside a running container is lost when the container is stopped/restarted or the host system is restarted. This is different in newer versions, but when running the container I Kubernetes, this is still true.

Try running node-red and make an API endpoint.

1. Make sure the node-red container is running (“docker ps”).
2. Go to the local address “localhost:1880”, and drag a “http in” and a “http response” box from the “network” part of the left menu onto the flow.
3. Drag a connection between the boxes.
4. Double click on the in box to set the url. Eks. “URL: /test”
5. Press the red “Deploy” button, and test the endpoint. “localhost:1880/test”
6. If you now kill the container and restart it again, the flow canvas is empty. So anything you do, is lost upon restart.
7. Kill the container again and start it with the following cmd line:  
   “docker run -it --rm -v C:\Users\Admin\dockerNk8s\node-red\data:/data -p 1880:1880 -d nodered/node-red”. ***Change the highlighted path to a local path that exist on your machine.***

## Making your own container

Make a docker image and push this to dockerhub.

1. Clone <https://github.com/kurt70/dockerk8sworkshop.git>
2. In cmd/bash/etc. go to the WebAppTest/ WebAppTest folder where the dockerfile is.
3. Compile the image by writing :  
   “docker build -t [yourdockeraccountname]/webapptest:latest .”
4. Run “docker images” to see your new image in your local repository.
5. To run it, “docker run -p 8080:80 -d [yourdockeraccountname]/webapptest”  
   If your port 8080 is used, change the port mapping to an available port.
6. In a browser go to “<http://localhost:8080>”.
7. Note that the program writes the hostname/machine name of the instance. We’ll use this later on in the k8s part.
8. Use “docker images” and note the size of the image. 261mb is huge.
9. Try finding a better and smaller base image. Ideally, the image should be 50-60 mb….
10. Try to push the image to your own dockerhub repository. Well need this later.