

Demo – Introduction to containers

Pull a container

```
docker image ls          # no nginx container image
docker pull nginx        # pull nginx container image
docker image ls          # container image listed
```

Run a container

Simple base image

```
docker run --name mycontainer -ti ubuntu
> exit
```

Something more interesting: web server or proxy

- without exposing a port:

```
curl localhost:8080      # nothing listening locally
curl localhost:80        # nothing listening locally
docker run --name mywebserver -d --rm nginx
# we have nginx running now, but can't access it until we're "in" the container:
# like this:
docker exec -ti mywebserver /bin/bash
> apt-get update
> apt-get install curl
> curl localhost:80      # nginx is serving us data
> exit
docker stop mywebserver
```

- exposing a port to the outside:

```
curl localhost:8080      # nothing listening on 8080
docker run --name mywebserver -d --rm -p 8080:80 nginx
curl localhost:8080      # nginx answers from terminal on local machine
```

- attaching to the running process:

```
# as opposed to starting a new process with "docker exec", we can just attach also:
docker attach mywebserver
# in a different terminal window, generate requests:
curl localhost:8080
# see log entries pop up on attached terminal
> <CTRL C>
```

- inspecting the container logs:

```
# we can now also inspect the logs from a container:
docker logs mywebserver    # one log entry
curl localhost:8080        # hit the server once more to generate another log entry
docker logs mywebserver    # two log entries
docker stop mywebserver    # stop the nginx container
```

Mapping the file system

```
mkdir tmp
vim ./tmp/hello.html          # create a static html file to serve from nginx
docker run --name nginx -d \
    --rm -p 8080:80 \
    -v $PWD/tmp:/usr/share/nginx/html \
    nginx
docker exec -ti nginx /bin/bash # go inside container and show content of
                                # \usr\share\nginx\html
curl localhost:8080/hello.html # content is served by nginx
docker stop nginx              # stop the nginx container
```

Building a container

Instead of mapping files from the host into a container, a more sustainable solution that allows a container to capture all its dependencies is to simply build your own. This is easier than one might expect.

Execute below steps in an empty folder.

Create content to copy into container

```
# prepare content for nginx
echo "hello world" > hello.html
```

Create a Dockerfile

Create a file called **Dockerfile** with the contents below:

```
FROM nginx
COPY ./hello.html /usr/share/nginx/html
```

Build the container

```
# replace "xstof" with your own docker hub repository name
docker build -t xstof/hello .
# show the image
docker image ls
# push the image to docker hub
docker push xstof/hello
```