DOCKER

**Install**

<https://store.docker.com/>

Download docker CE

Enable Hyper V in windows

**Tweak Docker for Windows**

Right click on it settings

Tweak terminal and shell

Install-Module -Scope CurrentUser posh-docker

IN powershell as admin :

Set-ExecutionPolicy RemoreSigned

Now when we write docker and tab it gives us the commands ….

<https://docs.docker.com/docker-for-windows/#test-your-installation>

**DOCKER CONTAINERS**

The general idea of docker is to have a container that has everything in it. It has a webserver also like nginx.

If you run a command it means you can talk to the server.

***Create a Nginx container***

**Image vs Container**

Image is the binaries and libraries and source code that all make your application.

Container is the running instance of that image. You can have many containers all running the same image.

In our shell

docker container run --publish 80:80 nginx

If we type localhost we can see now we are running an nginx.

Docker looked for an image called nginx. It started a new container for us to use. The publish command tells to use the port 80.

If we add detach in mac or linux it runs in the background.

In windows it runs with just pressing ctrl c.

docker container run --publish 80:80 nginx

***Common Container management commands***

The - - or – are options for the container like name ports etc

docker version → checks version

docker info → more info about the configuration of our files(number of containers running etc)

docker → list of commands that dockers uses

We have managements commands which are like categories for other commands.

docker <command> <sub-command> command is for management commands.

You can use either command subcommand or directly docker <subcommand>

Old commands will work with both ways. For example :

docker container run

docker run

docker container ls → list our containers and show info about them like container ID ,Image ports etc

docker container ls -a→ list our containers that we have run but also shows names that are unique

docker container stop and the three top digits of the container ID stops the container.(we can stop multiple containers)

docker container run --publish 80:80 --name webhost nginx

or docker run -p 8080

Publishing ports is always in HOST:CONTAINER format

We tell to run a container and use port 80 and the name of the container will be host and the image is nginx.

First it looks locally in image cache if doesn't find anything will look at image repository Docker Hub. It will download it and store it in the image cache.

If we don't specify a version it will have the latest. Creates new container based on that image and prepares to start.

Gives it a virtual IP on private network inside docker engine.

Open up a port we specified and forwards to port 80 in container.

Starts container by using the cmd in the image dockerfile.

docker container logs nameofthecontainer → show us the logs of the container like requests etc

docker container top nameofhtecontainer → The process running inside of the container

docker container COMMAND --help → We can see commands for the container

docker container rm → can take multiple three beginning letters of id and removes the containers

**You can t remove a running container so you must stop it**

docker container run --rm -it ubuntu bash - >remove the container when you stop it

docker container rm -f → We can force to stop and remove(we can have multiple values)

-- env option or -e → pass enviroment basic settings for example in mysql MYSQL\_RANDOM\_ROOT\_PASSWORD = yes

docker container start -ai nameofthecontainer

docker image ls – we can see image sizes and list that we have downloaded

docker pull imagenameorrepository

***Docker networking basics***

Each container can't run in the same port as other conainers so we need to assign them different cotnainers.

nginx should listen on 80:80

httpd (apache)8080:80

mysql 3306:3306

**CLI process monitoring**

docker container top – process list in one container

docker container inspect – details of one container config(it returns a json array)

docekr container stats – performance stats for all containers (ram,cpu,etc)

**Getting a Shell Inside Containers(No SSH needed)**

docker container run -it – start new container interactively. t emulates an ssh and i keeps session open.

docker container exec -it -run additional command in existing container

docker container exec -it webhost bash

docker container run -it --name proxy nginx bash

This will run bash inside nginx and the number after is the container id

To get out of the shell type exit

Since it was nginx with a shell when we exited the shell the service stopped

You can run a full ubuntu

docker container run -it - - name ubuntu ubuntu

So download ubuntu and place me on the prompt of this new container.

A container with ubuntu has minimal configuration so you must download extra configurations.

To start again the container

docker container start -ai nameofthecontainer

If we want to enter the shell in a container that exists

docker container exec -it mysql bash

We can see with ps aux processes run but we need to intall it in our docker container

apt-get update && apt-get install procps

With exec when we exit the container doesn t stop since it exec run an additional process

**Docker Networks:CLI process monitoring**

docker container port <container>

We can see which ports are forwarding traffic to tha container from the host to the container

Each container connected to a private vistrtual network “bridge”

Each virtual network routes through NAT firewall on host IP

All containers can talk each other in same network without using p

Best practices is to create a new virtual network for each app

Example network “my\_web\_app” for mysql and php/apache containers

Example network “my\_api” for mongo and nodejs containers

Anything can be changed and configured in docker

You can create multiple new networks

Attach containers to more than one virtual network or none

Skip virtual networks and use host IP(--net = host )

Check for container IP

docker container inspect --format ‘{{ .NetworkSettings.IPAddress}}’ <container>

Image that a virtual network is a bridge and when we use port we allow it to connect with our ethrnet network in that port. Different containers can talk each other if in same virtual network.

You can’t have two containers listening to port 80 on the host level. If you add same port we will get an error.

**Docker Networks: CLI Management of Virtual Networks**

docker network ls -> Show networks

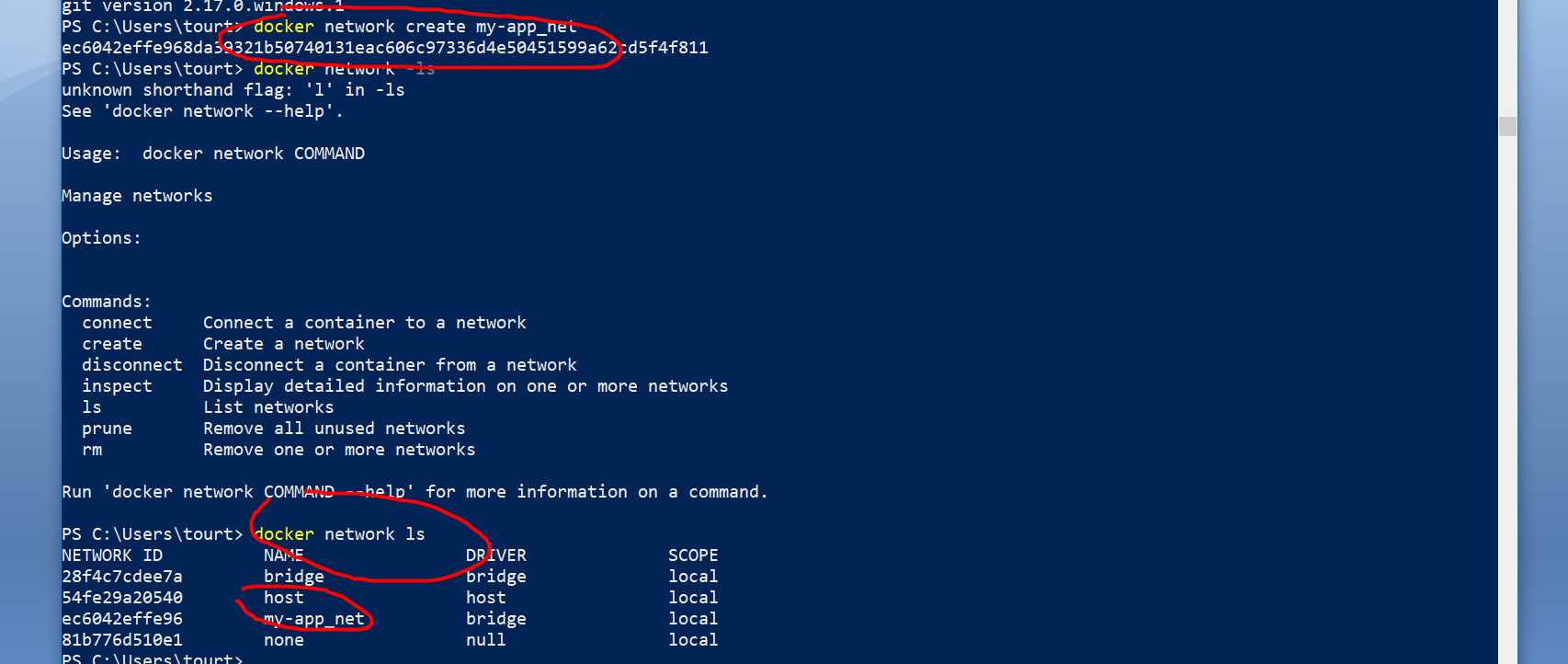
docker network inspect < name-> Inspect a network

docker network create - - driver -> create a network and third party drivers

docker network connect nameofnetwork nameofcontainer-> attach network to container

docker network disconnect nameofnetwork nameofcontainer -> detach network from container

##### Create a network



### Assing container to network

docker container run –d --name newnginx --network my\_app\_net nginx -> Will run an nginx container in the network my\_app\_net and if we run

or

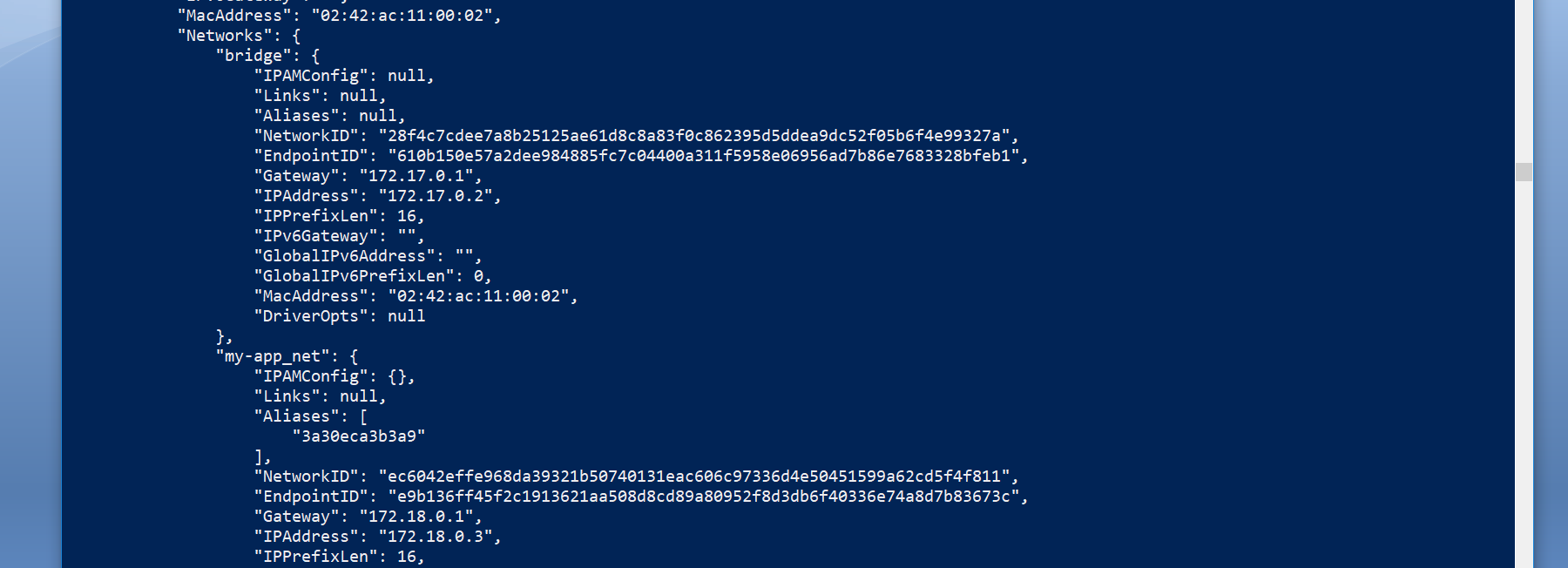
docker container run –d --name newnginx --net my\_app\_net nginx

##### Make a container have a second network

docker network connect my-app\_net webhost

docker container inspect webhost

we will see it belongs to two networks



## Docker Networks :DNS

### FORGET IPS

Since IPs change and are too dynamic its wrong to use IPs for our containers. For that we have a solution DNS naming.

If we have created our own network which is not the default bridge network it automatically gets a new feature which is automatic dns resolution for all the containers on that virtual network using their container names.

### Creating an alias for DNS

First create a network

Then create the containers you want to belong to that network by using -- net alias search

docker container run -d --net besmart --net-alias search elasticsearch:2

So we have give an alias to network called search

docker container run --rm --net besmart alpine nslookup search

It will return how many records we have in search alias

Address 1: 172.20.0.2 search.besmart

Address 2: 172.20.0.3 search.besmart

docker container run --rm --net besmart centos curl -s search:9200

it will return one record for besmart network in search alias in port 9200

## Images

What ‘s an image?

It s the application binaries and dependencies.

Metadata about the image data and how to run the image.

So it has the all the binaries the application has so you can run it in the host which provides the kernel. The difference is that it doesn t load a full operating system like a virtual machine does. It just starts an application.

You can have anything you want in it like Ubuntu distro with apt,apache, php etc.

### Docker Hub(the apt package system or nuget for containers)

### <https://hub.docker.com/>

Here we can find images. Anyone can put an image. Official are the ones most used. They are marked as official and they don’t have slashes. So a docker repository is an image!

Images can have tags.

docker container run --publish 80:80 nginx

For example the above run a container with the image nginx. If we haven’t download it with

docker pull nginx

It will download it and create the container.

We can download version we want:

docker pull nginx:1.11.9

## Image layers : Image Cache

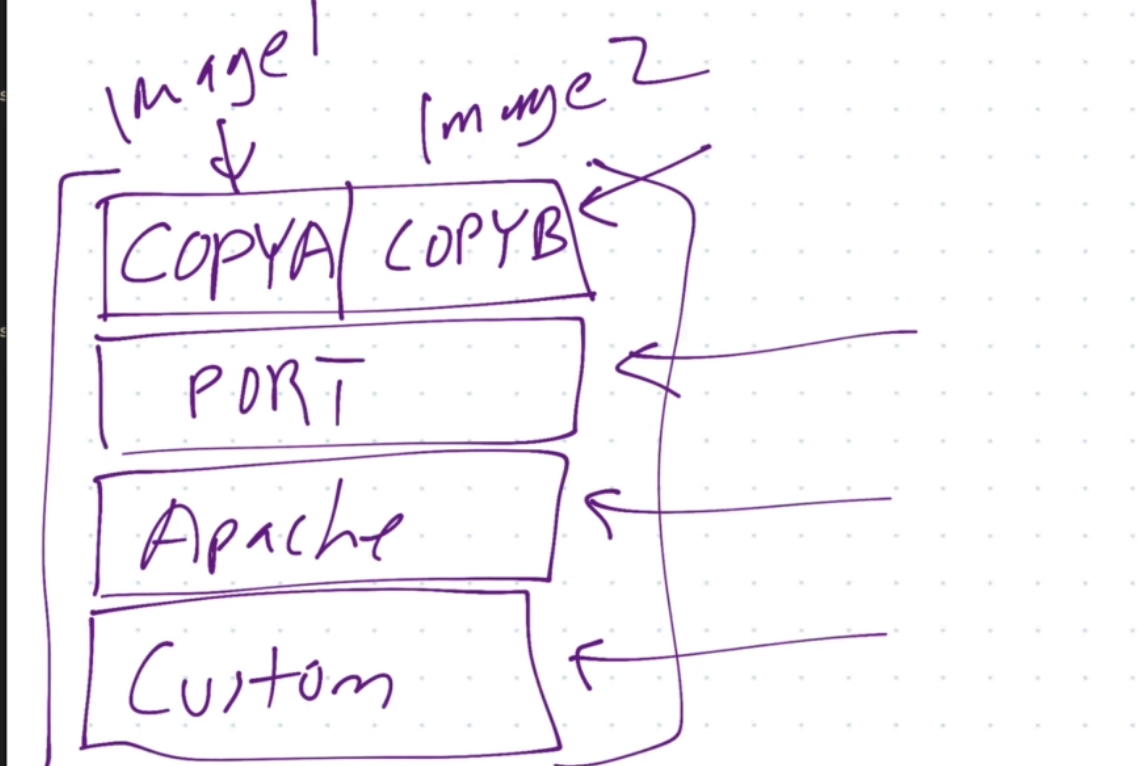
Images have layers. Every layer has a unique sha. Images can use layers that are used by another image. Docker works that way that they are stored cached so if an image uses a layer that is used by another image it won t create it again. That s why sometimes it says it would create it locally.

docker history <imagename>: History of the image layers. Every image starts from the very beginning with a blank layer known as scratch. Every set of changes happens after that on the file system in the image is another file.

docker image inspect <imagename> : It returns a json with metadata, author, default ports ,architecture etc

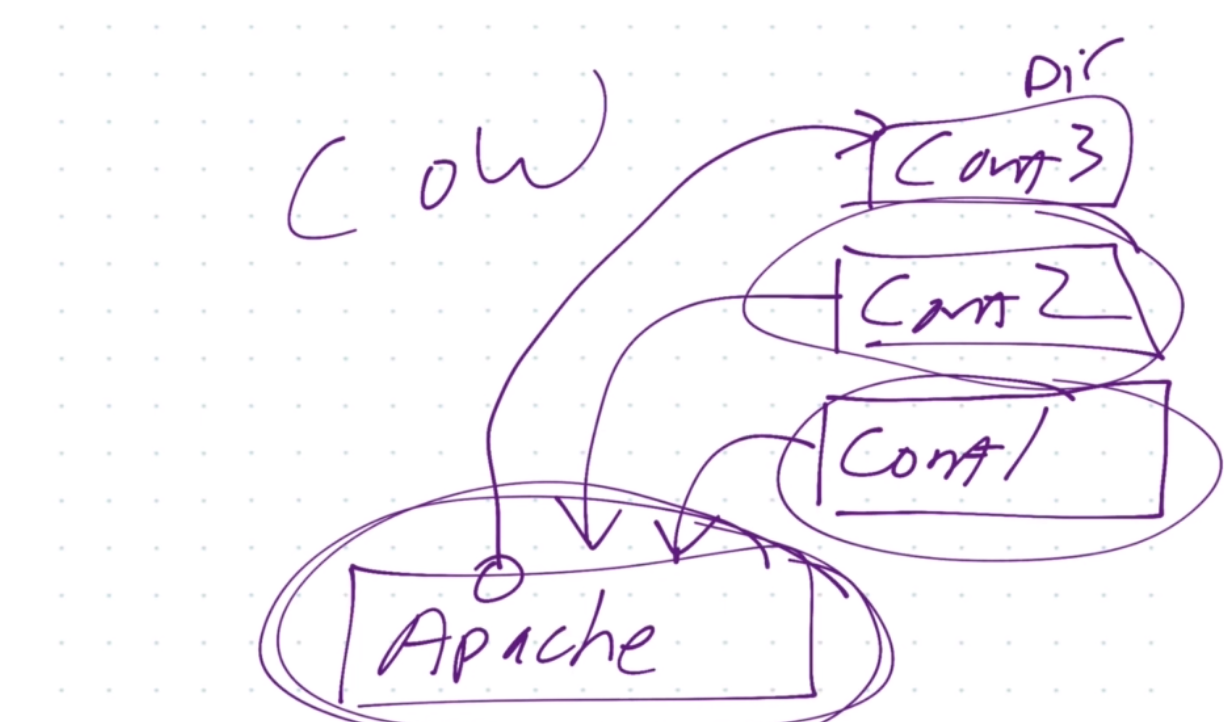
## Example

In the below example we have created a custom image with first layer apache second layer port one layer then we copy the docker file for 2 different sites. We will only have stored 5 files and not 8 files .



### Containers layering

When we create a container it creates a new single read/write layer which sits on top of image.



### Image tagging and pushing to docker hub

docker image ls

They have the repository name, tag, imageId .IF they are not official they will have the user/image or organization/image. Many times tags are versions. Different tags aren’t new images. Latest tag means default. Tagging is related to image ID.

Create a tag target image

docker image tag --help

Usage: docker image tag SOURCE\_IMAGE[:TAG] TARGET\_IMAGE[:TAG]

Create a tag TARGET\_IMAGE that refers to SOURCE\_IMAGE

docker push <imagename> : pushes the image to your repositories afgter you have loggeind in with the below command

docker login : log in to dockhub account

## Creating our own images

A Dockerfile looks like a shell script but it s not. It has it own language unique to docker. It ‘s stanza/command is its own layer.

### Commands/Stanzas

From : it’s in every Dockerfile which is required and specifies the distribution. It s the minimal.

Env : Environment variables. Very important for containers since we set keys and values for container building and running containers.

Run: It executes shell commands inside this container as it’s building.

The && means that all commands in that layer/stanza since it saves a little time and space.

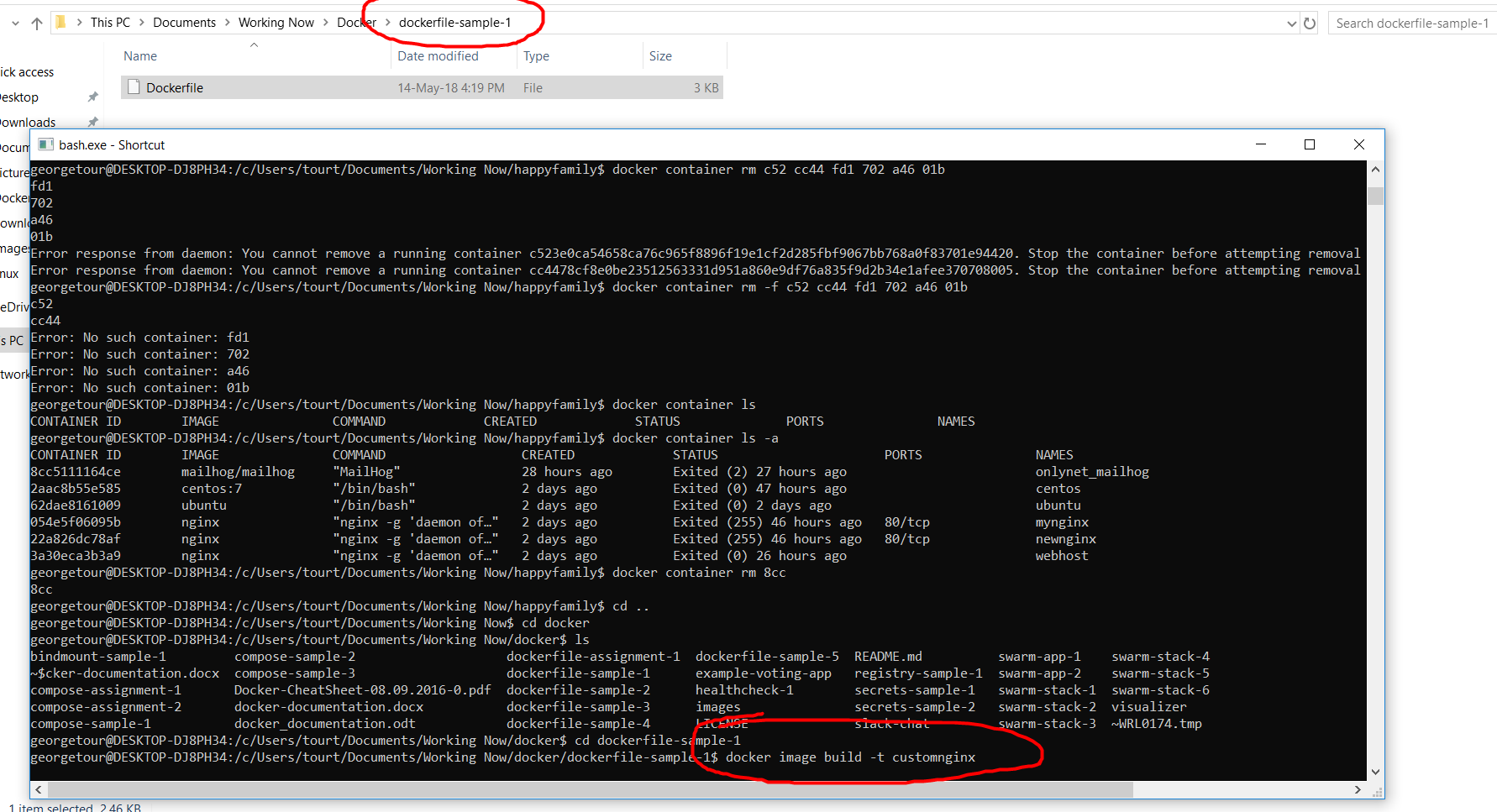
Expose : ports that will be exposed if we have –p command

CMD: required command that runs everytime you launch a new container or restart a stopped container

### Build image

Build image with tag customnginx in thius directory which that s what the . is .

docker image build –t customnginx .



Docker holds steps for changes so if you make a change in your docker file it will only apply this change whenever we run again the container.

If you can use an official image you can do it and add it in FROM.

WORKDIR

The WORKDIR instruction sets the working directory for any RUN, CMD, ENTRYPOINT, COPY and ADD instructions that follow it in the Dockerfile

Change default nginx directory toworkdir directory for html files

WORKDIR /usr/share/nginx/html

COPY copy the source code from local or build server to container images

COPY index.html index.html

To upload the image to dockerhub

Docker image tag nginx-with-html:latest georgetourtsin/nginx-with-html:latest

After all the above we build it

docker build –t testnode .

Then you run it

#### docker container run --rm -p 80:3000 testnode

## Dockerize

It means creating your own dockerfile for an existing app or service and you msut make it work properly in a container. You should match all images.