12 - 03 Feb 2023 (Docker)

03 February 2023

20:42

# Docker Installation

## We can get image from

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

## Download Docker

**docker pull tomcat**



## Download specific version

**docker pull tomcat:8.5.40**

**docker pull docker.io/library/nginx:latest**

## Will show all downloaded images from docker hub

**docker image ls**

## Run the image

2 ways to run a cont from given image

1. Interactive / foreground mode (dev usecase)  
   **docker run nginx:latest**

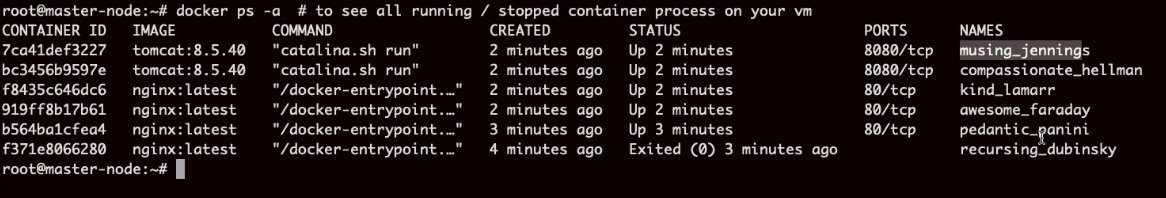


1. Detached / background mode (prod usecase)  
   docker run **-d** nginx:latest  
   docker run -d tomcat:8.4.5  
   we can run multiple instances of docker



## To see all running / stopped container process on your VM

**docker ps -a**



## Play with docker containers

**docker start <container ID>**

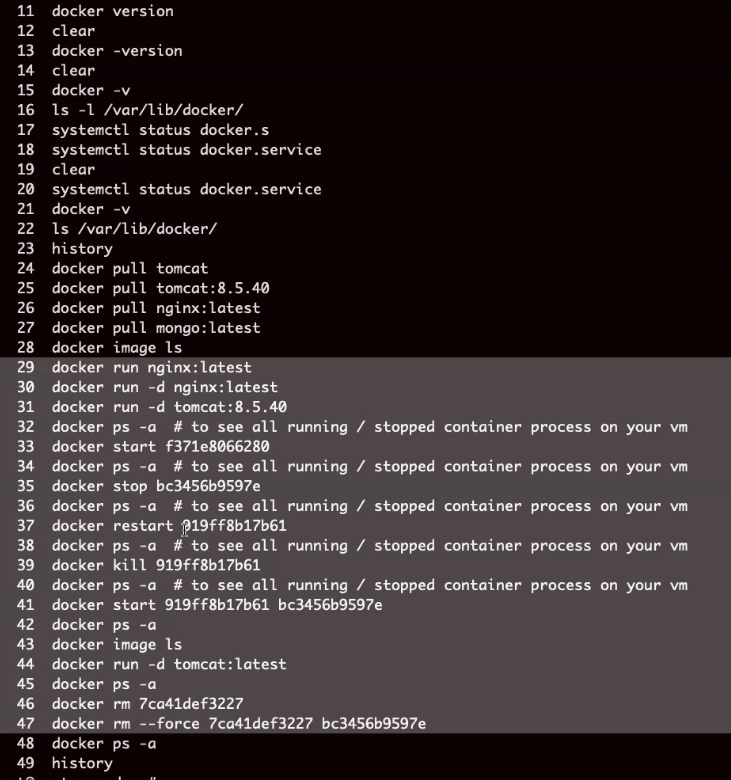
**docker stop <containerID> #graceful stop**

**docker restart <containerID>**

**docker kill <containerID> #force stop**

**Docker rm <containerID> #delete container**

**Docker rm <containerID> --force #force delete container**



# How to troubleshoot Docker containers

**Check the logs**

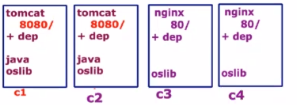
|  |  |
| --- | --- |
| **docker logs <container ID>** | **Less detail** |
| **docker inspect <container id>** | **More detail** |

**Every container has its own port number, its own IP address**

**To access container**

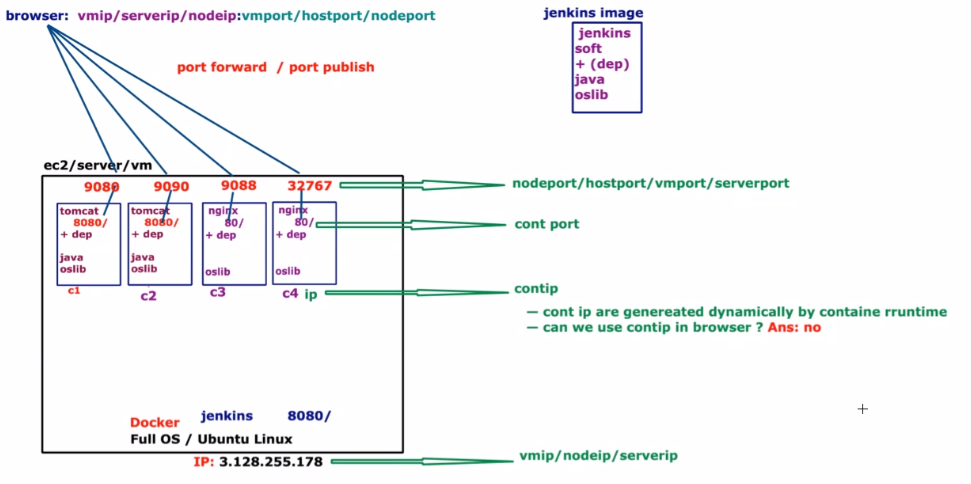
1. **Using curl  
   curl 172.17.0.2:80 #get this detail from inspect command**

**isolated containers. Each container has unique IP.**



## Port forward / publish concept

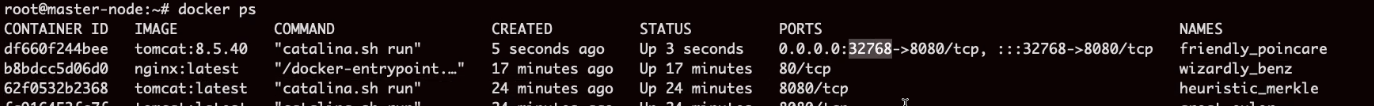
**We cannot access application directly. It has to go via vmIP:containerPort**



**With port number**

**docker run -d -P tocat:8.5.40 # -P is generating random number**

**docker ps**

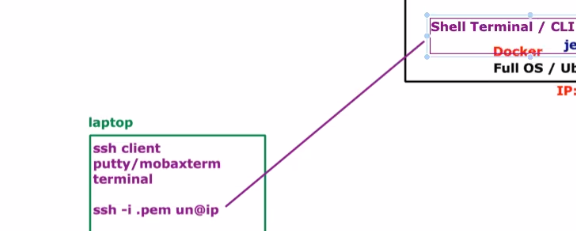


**docker run -d -p 9080:80 nginx:latest # -p allows to configure host port and container port**

**In browser test the accessebility**[**http://publicIP:nodeport**](http://publicIP:nodeport)

**Access the Container**

**This is how we access VM**

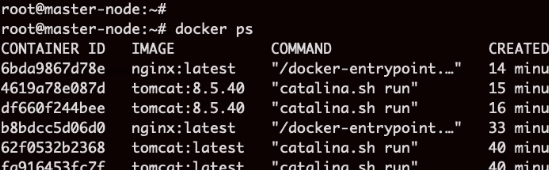


**No ssh for container.**

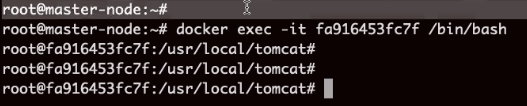
**To login to container**

**We use "docker exec" to login**

**Choose any container**



**docker exec -it <container id> /bin/bash # -it interactive terminal**

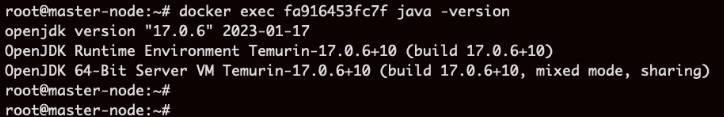


**Cat /etc/os-release**

**Exit #to exit**

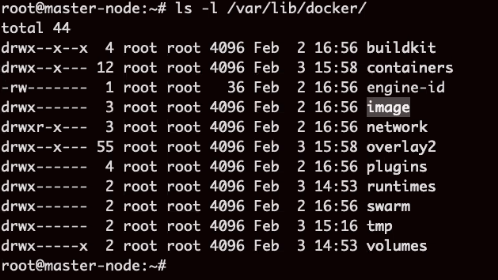
**Execute command on container from VM (without -it)**

**docker exec <containerID> java -version**



**Where image is stored?**

**In image folder**



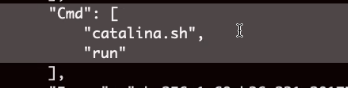
## 

## What is COMMAND column?



**How docker knows how to start these process**

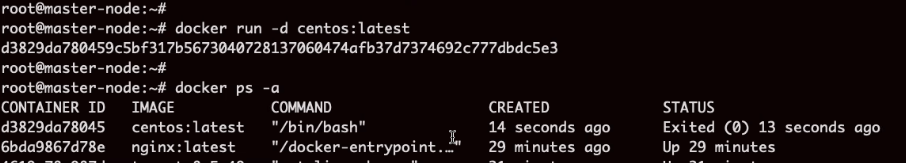
**Who created image has added some commands within image to tell docker which process to start.**



**docker pull centos:latest**

**docker run -d centos:latest**

**Container not running. See the command.**



**Any command we run on terminal/computer**

**Cmd creates a process**

**Program runs in memory**

**It is configured with /bin/bash**

**It executed and terminated.**

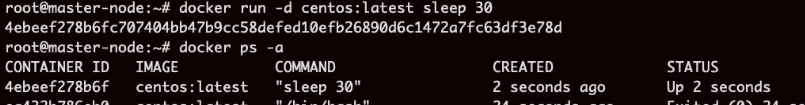
**Example ls command, it creates a process and ends. Sleep process stays there for 10 secs and then terminates**



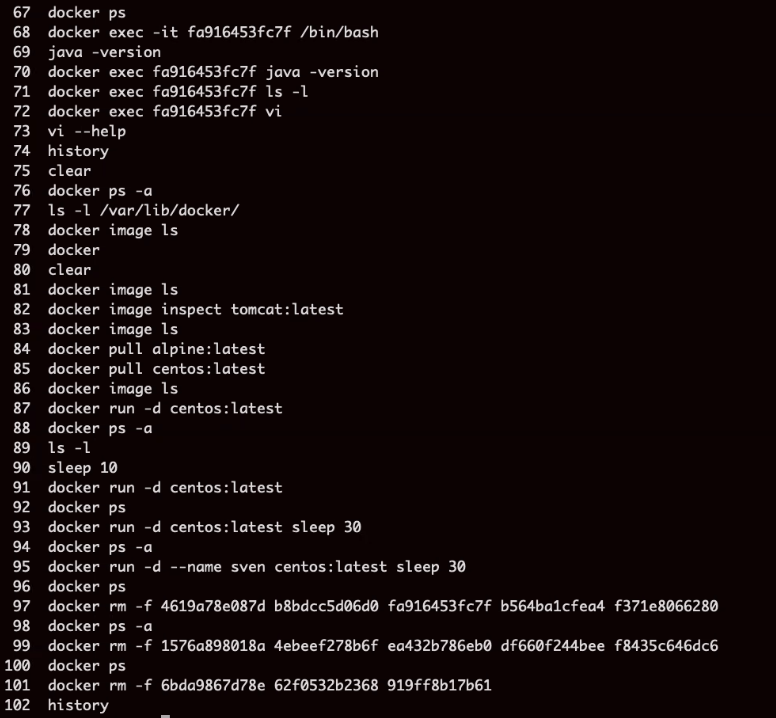
**Process is like heartbeat.**

**For container process is like heartbeat.**

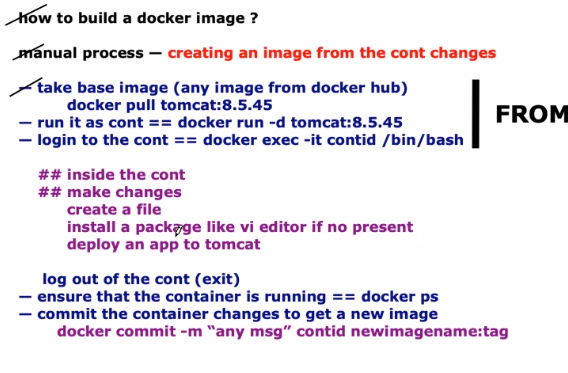
**Passed a sleep command in centos container**



**Summary**



# How do I construct or build my own image?



**Docker pull tomcat:8**

**Docker run -d tomcat:8**

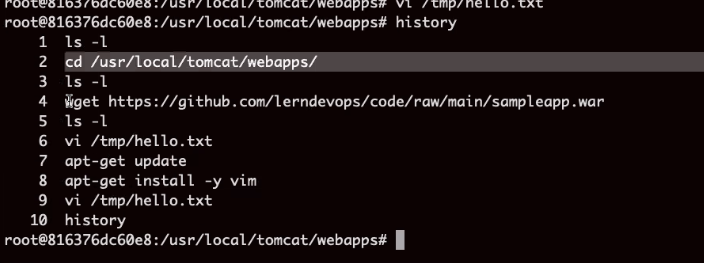
**docker ps**

**Docker exec -it contid /bin/bash**

**Cd /usr/local/toomcat/webapps/**

**wget past warfileURL**

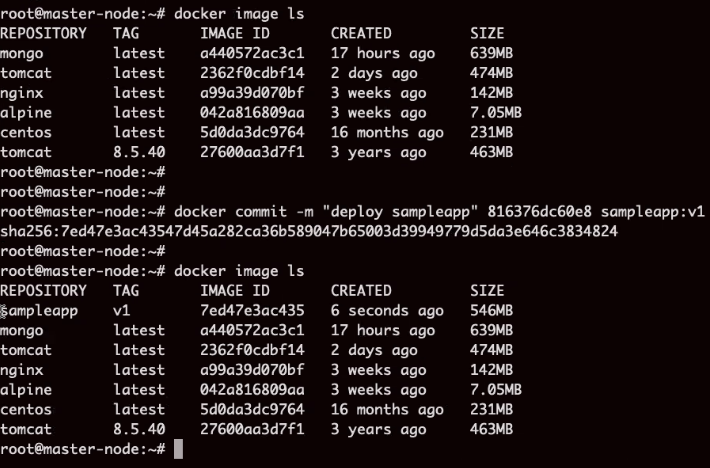
<https://github.com/lerndevops/code/raw/main/sampleapp.war>



**docker ps**

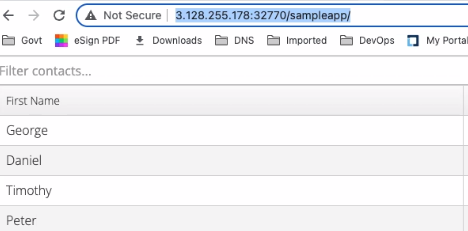
**Docker image ls**

**Docker commit -m "deployed sample app" ContainerID sampleapp:v1**

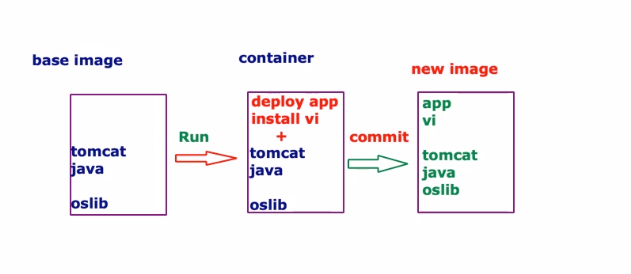


**Browse web site**

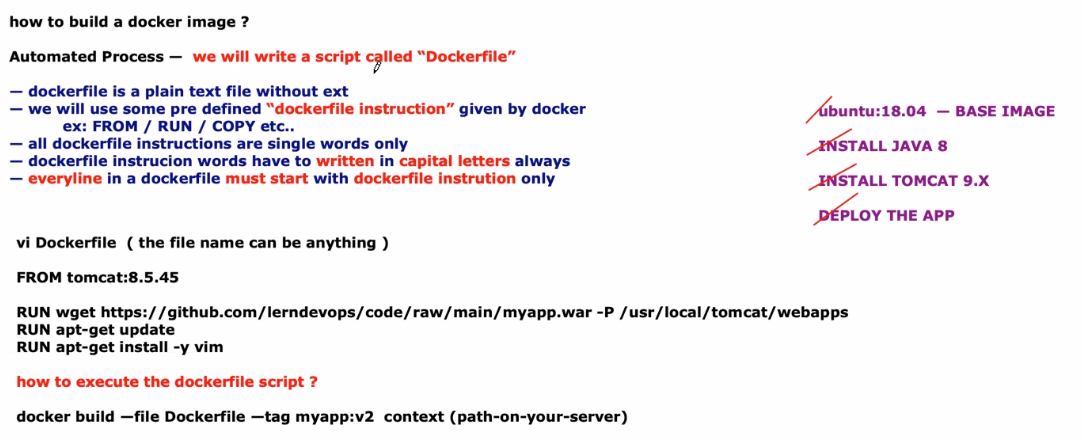
<http://3.128.255.178:32770/sampleapp>



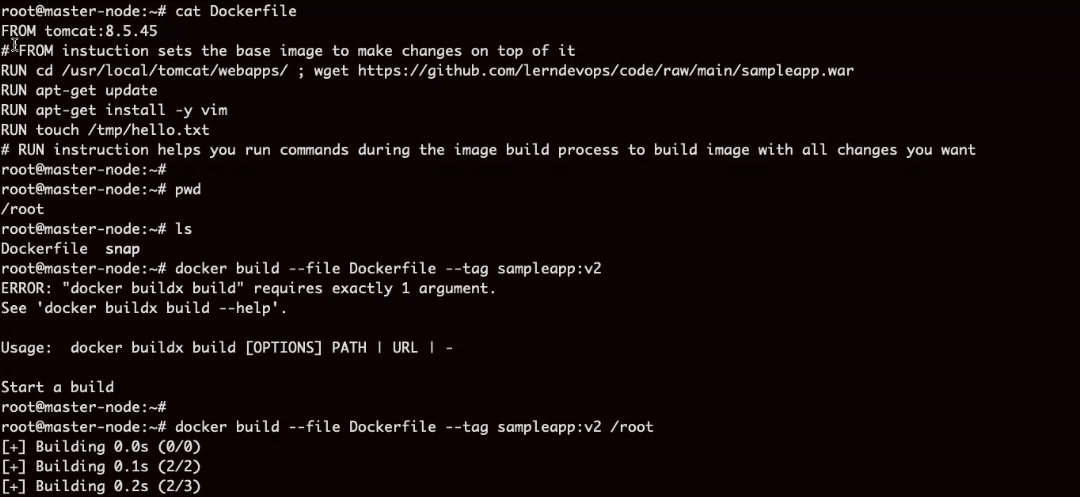
**Summary**

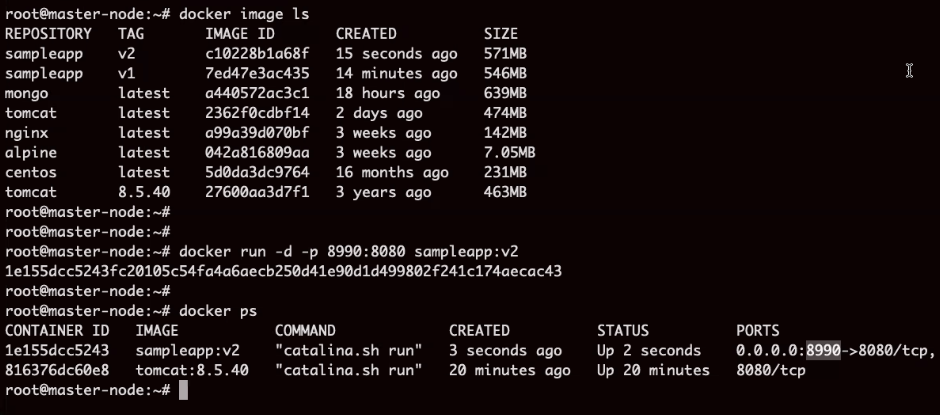


# Automate docker build image process

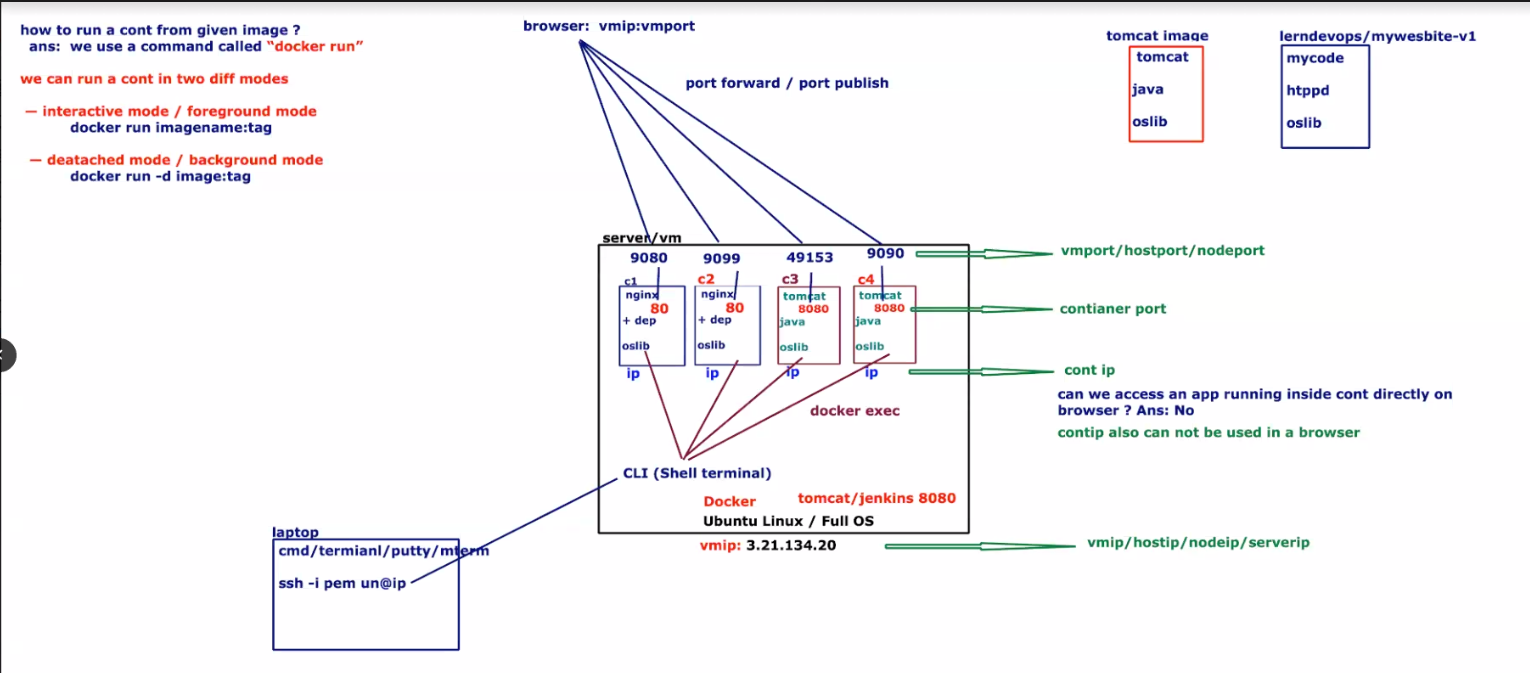


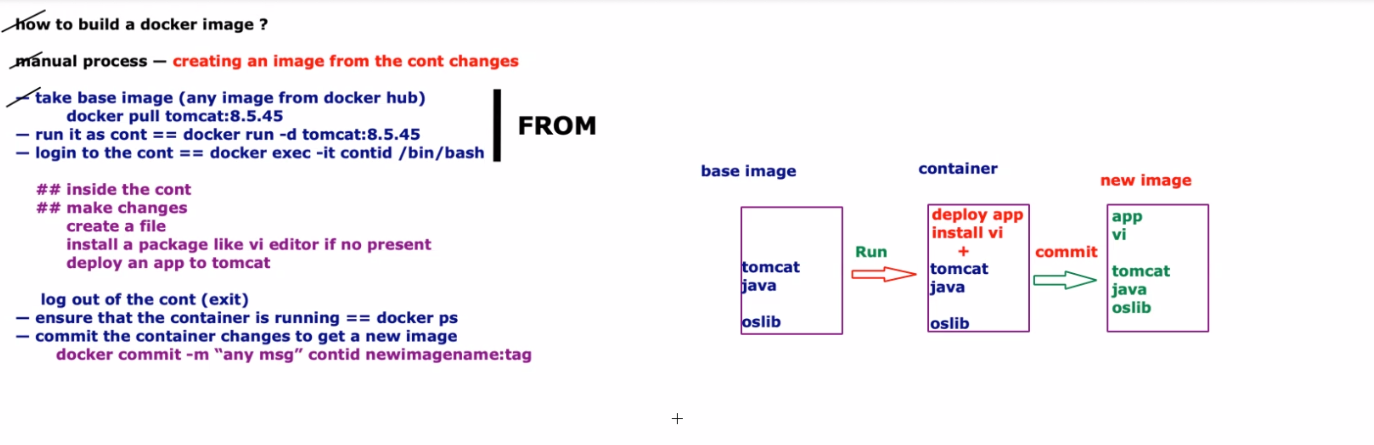
<https://github.com/lerndevops/code/raw/main/sampleapp.war>

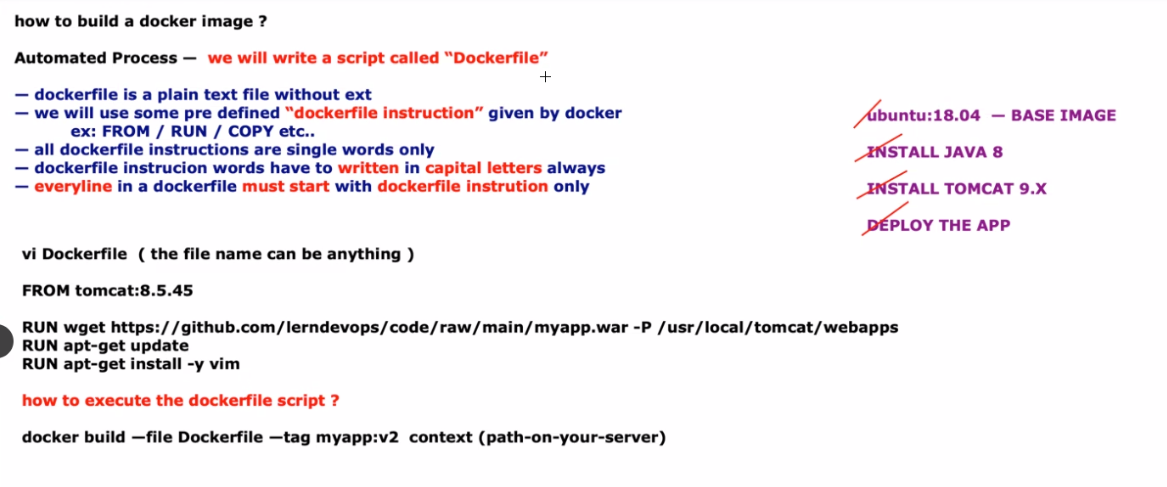


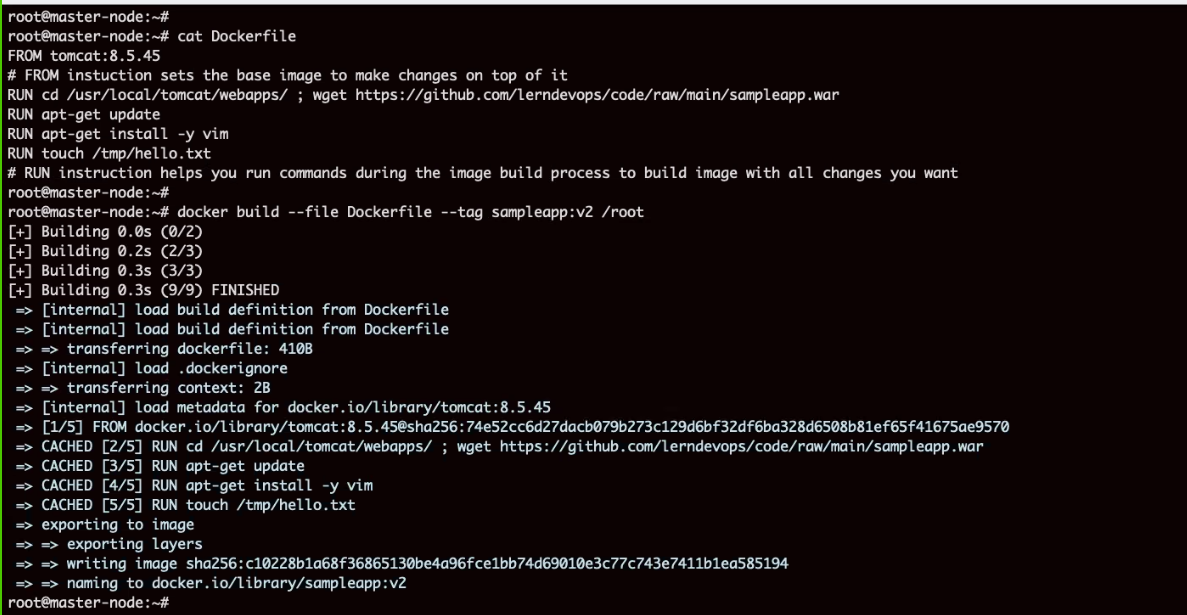


**Summary:**









**Next session advanced image build process will be covered**

13 - 06 Feb 2023 (Docker, Docker Volume)

06 February 2023

20:33

**Agenda:**

* Docker upload image
* Docker Volumes
* Docker Compose

In practical scenario we cannot download from internet

Installers will be saved on network location.

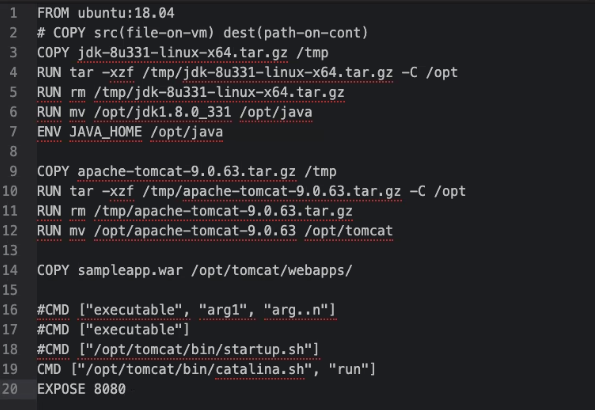
Jdk - # wget <https://github.com/lerndevops/code/raw/main/jdk-8u331-linux-x64.tar.gz>

Tomact - # wget <https://github.com/lerndevops/code/raw/main/apache-tomcat-9.0.63.tar.gz>

Sample.war file - # wget <https://github.com/lerndevops/code/raw/main/sampleapp.war>

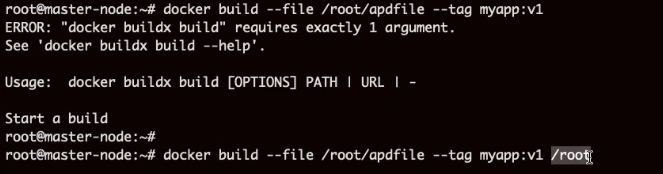
Download and install your own version

File Name: adfile

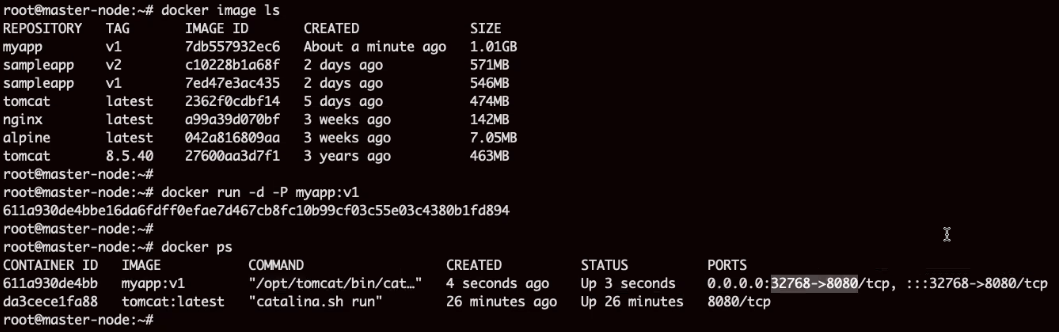


Root path cannot be given in copy command.

How docker will come to know where is the source file path



Build the Image



# How to share image

Push image to hub.docker.com

Resposotories --> create repository --> Name = **myapp**

--> Public --> create

## Authenticate other by login on to docker controller

Docker login

Username:

Password:

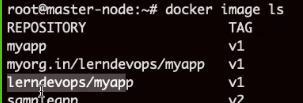
Docker push myapp:v1 (Access will be denied)

Tag image properly while building the image

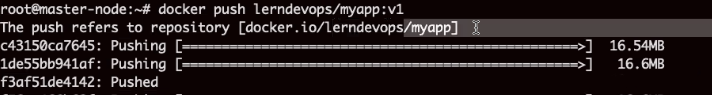


Docker images ls

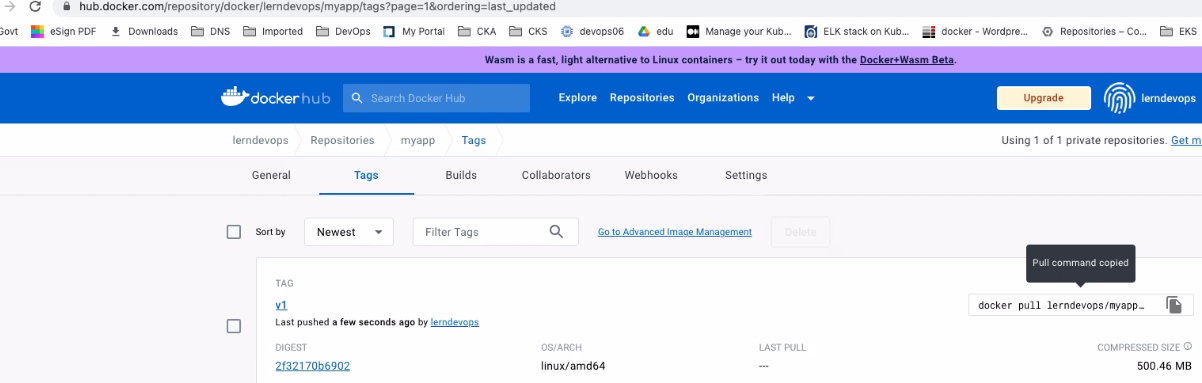
Account and repo



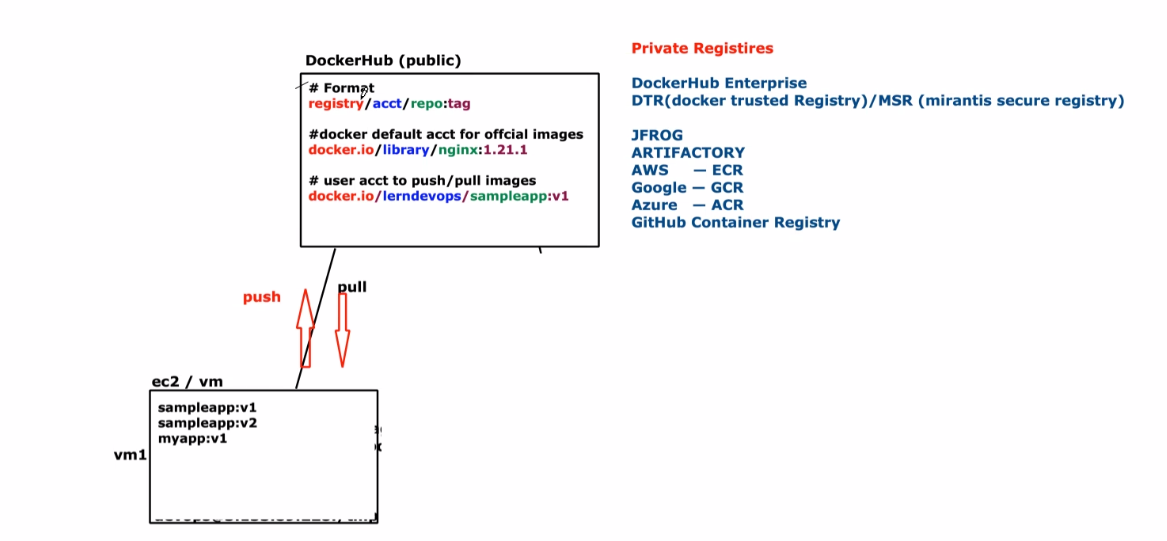
Push now



## Now others can download this image from docker hub

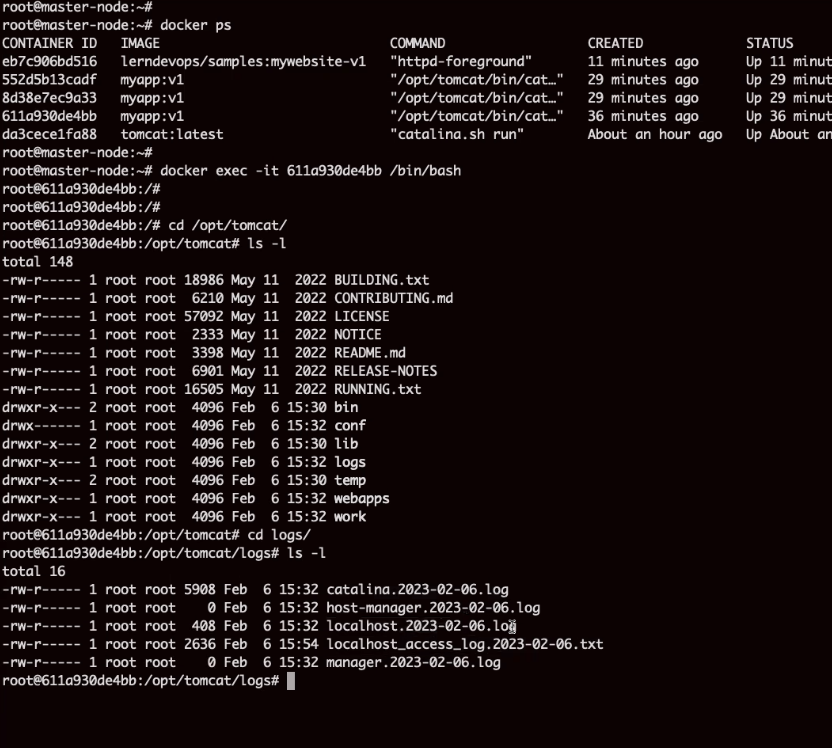


# Summary



# Docker Volumes

Logs are inside container



Once container deleted, logs gone

Use Volume Concept to make data persistent

1. Helps to attach and external storage to container
2. Helps to make the data persistent

Map container log to host path location

**Show all volumes**

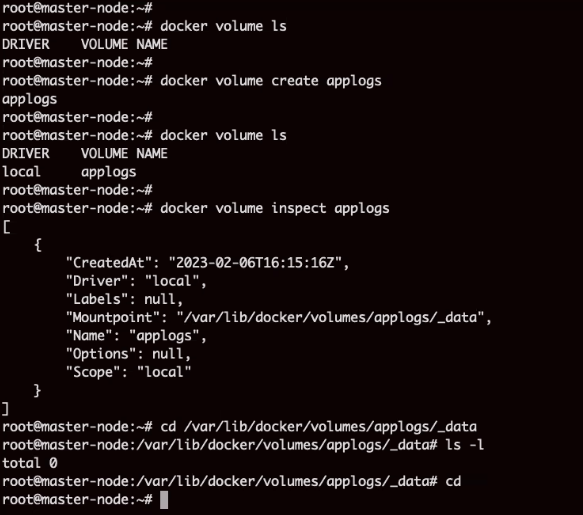
Docker volume ls

Docker volume create applogs

Docker volumes ls

Will show mountpoint where data is stored

Docker volume inspect applogs



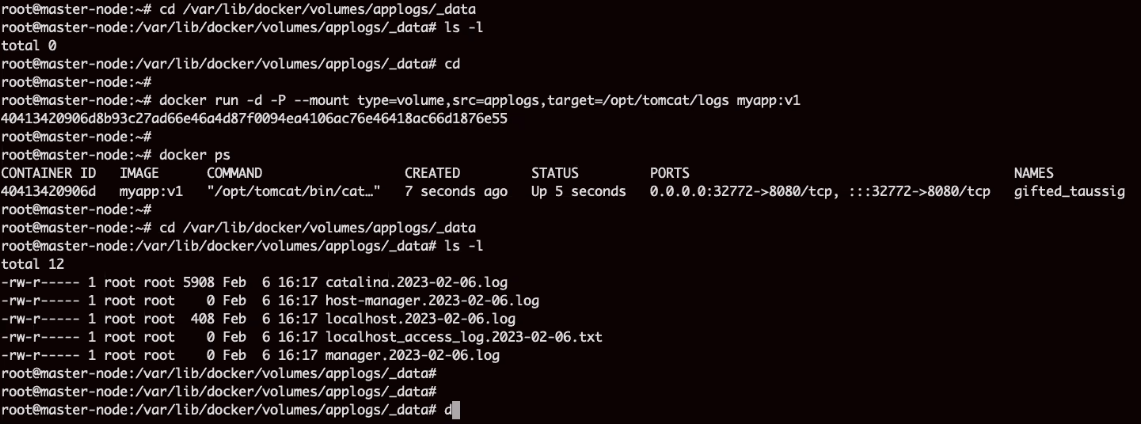
Cannot attach volume to running container

Attach a volume to container

docker run -d -P --mount type=volume,src=applogs,target=/opt/tomcat/logs myapp:v1



Log file created on local disk



Here volume is created and decided by Docker. It is called docker managed volume.

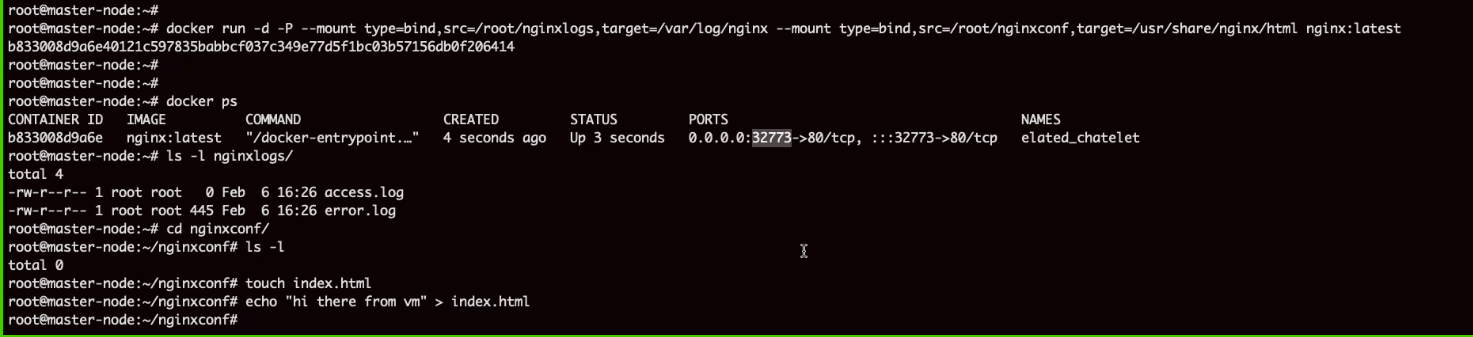
I want to set a custom path.

mkdir nginxlogs

mkdir nginxconf

docker run -d -P --mount type=**bind**,***src***=/root/nginxlogs,***target***=/var/log/nginx --mount type=**bind**,***src***=/root/nginxconfig,***target***=/usr/share/nginx/html ***nginx:latest***

One container has 2 volumes attached



Long syntax can be written like this



# Docker Compose

Thousands of uses hitting website. Run multiple instances to share the load.

We should automate container launch.

To write compose file, use YAML file.

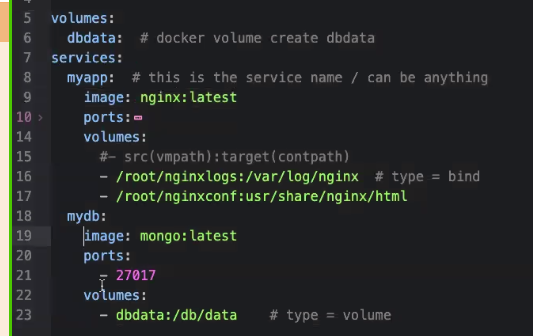
With docker compose we start creating a service.

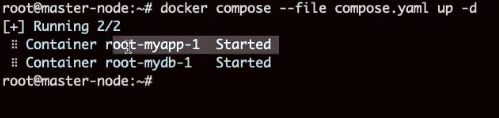
Service is a group of contains from single host only.

**Compos.yaml**

Docker compose --file compose.yaml up -d

By defailt it will try to crette one service for one





# 

# 

# 2 containers



# 



# 

# Bring down containers

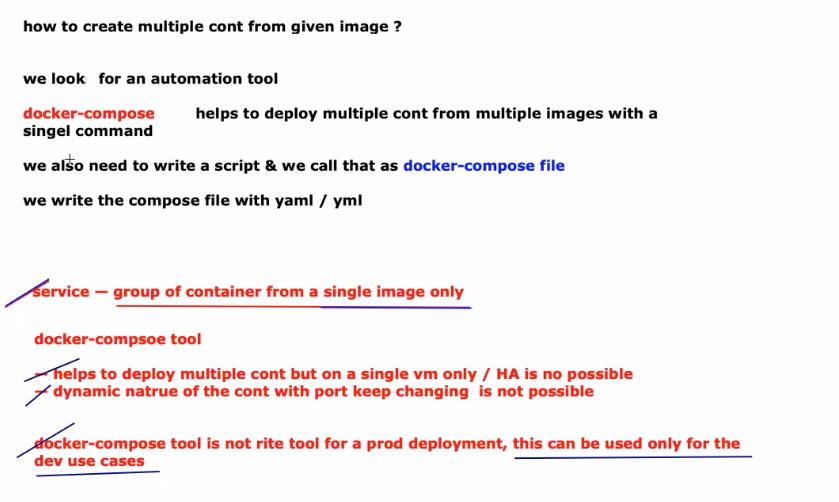


# 

# Compose is good for DEV Environment, not good for Production environment.

# When we spin more containers it keeps on changing port numbers. It is dynamic in nature.

# Compose Summary



**Tomorrow:**

Container orchestration tool

Kubernetes