Docker Running Notes

Docker : Docker is containerisation tool ,provides packaging the application and bundles all dependencies using docker we can create containers

what is container in docker???

A Docker container image is a lightweight, and executable package. It includes everything needed to run an application like code, runtime (java) , system tools, system libraries and settings.

Docker is a tool designed to make it easier to create, deploy, and run applications by using containers. Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and ship it all out as one package. ... And importantly, Docker is open source.

DOCKER ARCHITECTURE

virtualization vs containerization

export PS1='\u@\h# '

docker -v
docker --version
docker info
docker images
docker pull <iname>

docker run amazonlinx
-- container will be created and exited immediately
docker run -it <iid/imagename>

---enter into container

docker run -it --name <contname> <iid/imagename>
exit --> container gets stoped

ctrl + p + q

docker ps -a

docker ps (only running containers)
docker start cid
docker stop cid
docker pause cid
docker unpause cid

docker info

docker attach cid --> This command attaches your terminal to the running container's primary process (PID 1). It connects you to the standard input (STDIN), output (STDOUT), and error (STDERR)

It doesn't start a new process or give you a shell by default.

docker exec -it cid /bin/bash

This command executes a new process (/bin/bash, a shell) inside the running container and connects your terminal to it interactively

Starts a new shell session inside the container, independent of the main process.

docker rm cid
docker rm -f cid
docker rmi iid
docker rmi iid --force
docker rmi -f iid

docker run -itd <iid/imagename>

---

docker commit <cid>

docker tag iid <reponame>:<tag>

docker stats cid --> Display a live stream of container(s) resource usage statistics
docker top cid --> Display the running processes of a container

---

docekr rm $(docker ps -aq)
docker rmi $(docker images -q)

---

docker ps -al
87 docker images
88 docker pull amazonlinux
89 docker search amazonlinux
90 docker images
91 docker run amazonlinux
92 dockper ps -a

```
 93  docker ps -a
 94  docker run -it amazonlinux
 95  docker --version
 96  docker -v
 97  docker info
 98  docker rm -f $(docker ps -aq)
 99  docker rmi -f $(docker images -q)
100  docker info
101  docker images
102  docker search amazonlinux
103  docker pull amazonlinux
104  docker images
105  docker run c95cccf4246a
106  docker ps -a
107  docker run -it amazonlinux:latest
108  docker ps -a
109  docker run -it --name cont1 c95cccf4246a
110  docker ps -a
111  docker images
112  docker rename trusting_lamport cont2
113  docker ps -a
114  docker rename elated_brown cont3
115  docker ps -a
116  docker start cont2
117  docker start cont3
118  docker ps -a
119  docker stop cont3
120  docker ps -a
121  docker stop cont2
122  docker ps -a
123  docker pause cont1
124  docker ps -a
125  docker pause cont2
126  docker unpause 6ef9dcbe702c
127  docker ps -a
128  docker ps
129  docker ps -a
130  docker run -it --name cont4 amazonlinux
131  docker ps
132  docker attach cont4
133  docker ps
134  docker exec -it cont4 /bin/bash
135  docker ps
136  docker images
137  docker ps
```

```
138  docker ps -a
139  docker rm cont3
140  docker ps -a
141  docker rm cont4
142  docker rm -f cont4
143  docker ps -a
144  docker images
145  docker rmi c95cccf4246a
146  docker rmi -f c95cccf4246a
147  docker ps -a
148  docker rm cont1 cont2
149  docker rm -f cont1 cont2
150  docker ps
151  docker ps -a
152  docker images
153  docker rmi c95cccf4246a
154  docker images
155  docker run -it --name cont1 amazonlinux
156  docker ps
157  docker exec -it cont1 /bin/bash
158  docker ps
159  docker exec -it cont2 /bin/bash
160  docker run -it --name cont2 amazonlinux
161  docker ps
162  docker ps
163  docker commit cont1
164  docker images
165  docker run -it --name c3 842e85e4da4b
166  docker run -it --name c4 842e85e4da4b
167  docker ps
168  docker ps -a
169  docker images
170  docker tag 842e85e4da4b amazonlinux:git
171  docker iamges
172  docker images
173  docker run -it --name c5 amazonlinux:git
174  docker ps
175  docker commit c5
176  docker images
177  docker tag a1812a02562c amazonlinux:git_java
178  docker images
179  docker run -it --name cont6 amazonlinux:git_java
180  docker ps -a
181  docker run -itd --name cont7 amazonlinux
182  docker ps -a
```

```
 183  docker ps -al
 184  top
 185  docker stats cont7
 186  docker top cont7
 187  docker exec -it cont7 /bin/bash
 188  docker top cont7
 189  docekr ps -a
 190  docker ps -a
 191  docker ps -aq
 192  docker rm -f $(docker ps -aq)
 193  docker ps -a
 194  docker images
 195  docker images -q
 196  docker rmi -f $(docker images -q)
```

DATE : 27 MAR 2025

docker hub account creation

creation of Image repo in your DH account

docker image push to - Image repo in your DH account

==========
LAUNCHING A WEB APPLICATION:
httpd

```
yum install httpd -y
systemctl start httpd
cd /var/www/html/
vim index.html
```

publicIP:80
==========

port maping /port forwards

docker run -it --name <contname> -p HP:CP amazonlinux

docker run -it --name fb_httpd_cont2 -p 8081:80 amazonlinux

```
yum install httpd -y
cd /usr/sbin/
./httpd
```

cd /var/www/html/
yum install vim -y
vim index.html

==========

docker cp src dest
docker cp index.html fb_httpd_cont2:/var/www/html/index.html
Successfully copied 77.8kB to fb_httpd_cont2:/var/www/html/index.html
---
docker run -it -h <hostname> ubuntu
adduser kiran in container --> useradd kiran

docker exec -it -u kiran CID /bin/bash

==========

```
 221  docker -v
 222  systemctl status docker
 223  systemctl enable docker
 224  systemctl start docker
 225  systemctl status docker
 226  docker images
 227  docker ps
 228  docker ps -a
 229  docker rm -f $(docker ps -aq)
 230  docker rmi $(docker images -q)
 231  docker images
 232  docker run -name cont1 amazonlinux
 233  docker run --name cont1 amazonlinux
 234  docke ps -a
 235  docker ps -a
 236  docker run -itd --name cont2 amazonlinux
 237  docker ps -a
 238  docker run -it --name cont3 amazonlinux sleep 10
 239  docker ps -a
 240  docker run -itd --name cont4 amazonlinux sleep 20
 241  docker ps -a
 242  docker run -it --name cont5 amazonlinux
 243  docker ps
 244  docker commit cont5
 245  docker images
 246  docker tag 246983fcf6c6 amazonlinux:git
 247  docker images
 248  docker push amazonlinux:git
```

```
249  docker login
250  docker images
251  docker tag 246983fcf6c6 devopshubg333/batch15d:git
252  docker images
253  docker push devopshubg333/batch15d:git
254  docker rm -f $(docker ps -qa)
255  docker rmi -f $(docker images -q)
256  docker images
257  docker pull devopshubg333/batch15d:git
258  docker images
259  docker run -it --name c1 devopshubg333/batch15d:git
260  yum install httpd -y
261  systemctl status httpd
262  systemctl start httpd
263  systemctl status httpd
264  cd /var/www/html/
265  ll
266  vim index.html
267  vim index.html
268  docker iamges
269  docker images
270  docker run -it --name fb_httpd_cont1 amazonlinux
271  docker ps
272  docker run -it --name fb_httpd_cont2 -p 8081:80 amazonlinux
273  docker ps
274  docker ps
275  docker exec fb_httpd_cont2 /bin/bash
276  docker exec -it fb_httpd_cont2 /bin/bash
277  docker ps
278  ll
279  cp -rp index.html /opt/
280  ll
281  cd /opt/
282  ll
283  docker cp index.html fb_httpd_cont2:/var/www/html/index.html
284  docker images
285  docker pull ubuntu
286  docker run --name c1 ubuntu
287  docker run --name c2 ubuntu
288  docker run -it --name c3 ubuntu
289  docker ps
290  docker exec -it -u madhu c3 /bin/bash
291  docker run -it --name c4 -h httpdserver ubuntu
292  history
```

DAY 3: 28 MARCH 25

==========
HOW TO PUSH IMAGE TO DOCKER HUB :

docker login
uname :
password:

docker push devopshubg333/batch6:git

==========

TO CHECK IMAGE HISTORY :

docker history <i.name>:tag

docker logs cid
==========
docker home or root directory

/var/lib/docker
==========
Docker Container - ephemeral nature :
Docker containers are ephemeral by design, meaning they are temporary and stateless. Once a container stops or is deleted, any data or changes made inside it during its runtime are lost unless explicitly preserved
DOCKER VOLUMES

docker volume create my_volume
docker run -d -v my_volume:/data my_image

docker volume help

Usage: docker volume COMMAND

Manage volumes

Commands:
  create   Create a volume
  inspect   Display detailed information on one or more volumes
  ls   List volumes
  prune   Remove unused local volumes
  rm   Remove one or more volumes
  update   Update a volume (cluster volumes only)

Run 'docker volume COMMAND --help' for more information on a command.

BIND MOUNTS

docker run -v /host/path:/container/path my-image
-v HostVolumePATH:ContVolumePATH

docker run -it --name cont2 -v /opt:/opt/logs iname

==========

docker system prune
docker container prune
docker image prune
docker volume prune

==========

Docker file :

how to build --?
docker build -t <devopshubg333/batch11:tag> .

---
httpd dockerfile:

FROM amazonlinux
RUN yum install httpd -y
COPY index.html /var/www/html/index.html
CMD ["/usr/sbin/httpd","-D","FOREGROUND"]
EXPOSE 80
---

docker run -it -p 8090:80 --name cont1 iid
==========
29 MARCH 2025

Dockerizing sample python web-application :

======
Code : /opt/app.py

```
import os
from flask import Flask
app = Flask(__name__)

@app.route("/")
def main():
   return "Welcome to Batch15!"

@app.route('/how are you')
def hello():
   return 'I am good, how about you?'

if __name__ == "__main__":
   app.run()
```

==========

Commands :

```
yum update -y
yum install python3 python3-pip pip -y
pip install flask
FLASK_APP=/opt/app.py flask run --host=0.0.0.0 --port=8080
```

==========

Dockerfile : (1. if OS is amazonlinux :)

```
FROM amazonlinux
LABEL maintainer="MADHU KIRAN <devopstraininghub@gmail.com>"
RUN yum install python3 python3-pip pip -y
RUN pip install flask
COPY app.py /opt/app.py
CMD FLASK_APP=/opt/app.py flask run --host=0.0.0.0 --port=8080
EXPOSE 8080
```

==========

2. if OS is ubuntu :

```
FROM ubuntu:latest
LABEL maintainer="KIRAN <devopstraininghub@gmail.com>"

# Update the package list and install necessary dependencies
RUN apt-get update -y && apt-get install -y python3 python3-pip

# Install Flask via pip
RUN pip3 install flask
```

```
# Copy your app.py into the container
COPY app.py /opt/app.py

# Set the environment variable for Flask app and run the Flask server
CMD FLASK_APP=/opt/app.py flask run --host=0.0.0.0 --port=8080

# Expose the port Flask will run on
EXPOSE 8080
```

---

3. Dockerfile using python base image :

```
FROM python:3.9

# Copy the Flask application file to the container
COPY app.py /opt/app.py

# Set the FLASK_APP environment variable
ENV FLASK_APP=/opt/app.py

# Expose the port on which the Flask application will run
EXPOSE 8080

# Run the Flask application
CMD ["flask", "run", "--host=0.0.0.0", "--port=8080"]
CMD FLASK_APP=/opt/app.py flask run --host=0.0.0.0 --port=8080
```

==========

DOCKERFILE INSTUCTIONS DEEP DIVE :

FROM
RUN
COPY
ADD
COPY VS ADD

CMD
ENTRYPOINT
EXPOSE

WORKDIR
LABEL

ARG
ENV

ADD

RUN VS
CMD VS ENTRY POINT
---
cat Dockerfile
FROM amazonlinux
RUN yum install iputils -y
ENTRYPOINT ["ping"]
CMD ["www.google.com"]

RUN executes command(s) in a new layer and creates a new image. E.g., it is often used for installing software packages

CMD sets default command and/or parameters, which can be overwritten from command line when docker container runs

RUN - DURING IMAGE BUILDING

CMD - DURING CONTAINER CREATIING - default instruction and we can override it
ENTRYPOINT configures a container that will run as an executable – can not be overridden but APPENDED & GIVEN HIGH PRIORITY

exec

COPY - COPY SRC TO DEST (LOCALLY)
ADD - REMOTE URL CAN ALSO BE COPIED
==========

DATE: 1 APRIL 2025

Tomcat Docker file :

[root@ip-172-31-86-3 opt]# cat Dockerfile.tomcat
FROM amazonlinux:latest
LABEL maintainer="madhu <devopstraininghub@gmail.com>"
RUN yum install java -y && yum install tar gzip -y
WORKDIR /opt
ADD https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.102/bin/apache-tomcat-9.0.102.tar.gz .

```
# Extract the tar.gz file
RUN tar -xvf apache-tomcat-9.0.102.tar.gz && rm apache-tomcat-9.0.102.tar.gz


RUN sed -i 's/"127\\.\\d+\\.\\d+\\.\\d+|::1|0:0:0:0:0:0:0:1"/".*"/g' /opt/apache-tomcat-
9.0.102/webapps/manager/META-INF/context.xml
WORKDIR /opt/apache-tomcat-9.0.102/conf/
RUN rm -rf tomcat-users.xml
RUN  echo '<?xml version="1.0" encoding="utf-8"?> \
      <tomcat-users> \
      <role rolename="manager-gui"/> \
      <user username="tomcat" password="tomcat" roles="manager-gui, manager-script, manager-
status"/> \
      </tomcat-users>' > tomcat-users.xml

CMD ["/opt/apache-tomcat-9.0.102/bin/catalina.sh", "run"]

EXPOSE 8080


================================================================================
# Use Amazon Linux base image
FROM amazonlinux:latest

# Install necessary packages
RUN yum install -y java-1.8.0-openjdk-devel zip gzip tar && \
   yum clean all

# Set environment variables
ENV TOMCAT_VERSION=9.0.89 \
   CATALINA_HOME=/opt/tomcat \
   PATH=$CATALINA_HOME/bin:$PATH

# Download and extract Tomcat
RUN curl -O https://dlcdn.apache.org/tomcat/tomcat-9/v${TOMCAT_VERSION}/bin/apache-tomcat-
${TOMCAT_VERSION}.tar.gz && \
   tar -xf apache-tomcat-${TOMCAT_VERSION}.tar.gz -C /opt && \
   rm apache-tomcat-${TOMCAT_VERSION}.tar.gz && \
   ln -s /opt/apache-tomcat-${TOMCAT_VERSION} $CATALINA_HOME

# Update Tomcat manager context to allow access from any IP
RUN sed -i 's/"127\\.\\d+\\.\\d+\\.\\d+|::1|0:0:0:0:0:0:0:1"/".*"/g'
$CATALINA_HOME/webapps/manager/META-INF/context.xml
```

```
# Overwrite the default Tomcat users file with a new one containing desired users and roles
RUN echo '<?xml version="1.0" encoding="utf-8"?> \
    <tomcat-users> \
    <role rolename="manager-gui"/> \
    <role rolename="manager-script"/> \
    <role rolename="manager-status"/> \
    <user username="tomcat" password="Tomcat" roles="manager-gui, manager-script, manager-status"/> \
    </tomcat-users>' > $CATALINA_HOME/conf/tomcat-users.xml

# Expose Tomcat port
EXPOSE 8080

# Start Tomcat
CMD ["catalina.sh", "run"]
```

==========

jenkins image --

docker pull jenkins/jenkins:lts

==========

2 APR 2025

==========

IMAGE OPTIMIZATION :

MULTI STAGE DOCKER FILES:

MINIMAL OS IN DOCKER IAMGES -- BASE IMAGE

USE MINIMAL BASE IMAGE / DISTROLESS IMAGES

MULTISTATE DOCKERFILE & DISTROLESS

IMAGE OPTIMIZATION :

SMALL IMAGES - BETTER SECURITY -- FAST SPIN UP & DEPLOYMENT

YOU need only run time & runtime executable in image - no need of buid stage dependencies (Unnecessary packages)

Larger Docker images typically contain more software packages, dependencies, and potentially unnecessary files. This increases the attack surface, providing more potential entry points for

attackers to exploit vulnerabilities. Every additional package or file included in the image represents a potential avenue for attackers to gain unauthorized access or execute malicious code.

To mitigate these security risks, it's important to follow best practices for Docker image security:

Minimize Image Size: Trim unnecessary files, dependencies, and packages from the Docker image to reduce its size and attack surface.

Use Official Base Images: Start with official and minimal base images provided by Docker or trusted organizations, as they are typically well-maintained and regularly updated with security patches.

Regularly Update Images: Keep Docker images up-to-date by regularly rebuilding them with the latest security patches and software updates.

Scan Images for Vulnerabilities: Utilize Docker security scanning tools to detect and remediate known vulnerabilities in Docker images.

Implement Least Privilege: Follow the principle of least privilege when configuring user permissions and access controls within Docker containers to minimize the potential impact of security breaches.

==========

```
##artifact build stage
FROM maven AS buildstage
RUN mkdir /opt/mindcircuit15
WORKDIR /opt/mindcircuit15
COPY . .
RUN mvn clean install    ## artifact -- .war

### tomcat deploy stage
FROM tomcat
WORKDIR webapps
COPY --from=buildstage /opt/mindcircuit15/target/*.war .
RUN rm -rf ROOT && mv *.war ROOT.war
EXPOSE 8080
```

==========

3 APR 2025

```
sudo usermod -aG docker ec2-user
sudo usermod -aG docker ubuntu
```

calculator.go
=========

```
package main

import (
	"bufio"
	"fmt"
	"os"
	"strconv"
	"strings"
)

func main() {
	fmt.Println("HELLO BATCH15 AWS DEVOPS CHMAPS , I am a calculator app ....")

	for {
		// Read input from the user
		reader := bufio.NewReader(os.Stdin)
		fmt.Print("Enter any calculation (Example: 1 + 2 (or) 2 * 5 -> Please maintain spaces
as shown in example): ")
		text, _ := reader.ReadString('\n')

		// Trim the newline character from the input
		text = strings.TrimSpace(text)

		// Check if the user entered "exit" to quit the program
		if text == "exit" {
			break
		}

		// Split the input into two parts: the left operand and the right operand
		parts := strings.Split(text, " ")
		if len(parts) != 3 {
			fmt.Println("Invalid input. Try again.")
			continue
		}

		// Convert the operands to integers
		left, err := strconv.Atoi(parts[0])
		if err != nil {
			fmt.Println("Invalid input. Try again.")
			continue
		}
```

```
		right, err := strconv.Atoi(parts[2])
		if err != nil {
			fmt.Println("Invalid input. Try again.")
			continue
		}

		// Perform the calculation based on the operator
		var result int
		switch parts[1] {
		case "+":
			result = left + right
		case "-":
			result = left - right
		case "*":
			result = left * right
		case "/":
			result = left / right
		default:
			fmt.Println("Invalid operator. Try again.")
			continue
		}

		// Print the result
		fmt.Printf("Result: %d\n", result)
	}
}
```

==========

STEPS:

yum update
yum install -y go
GO111MODULE=off
go build calculator.go

==========

[root@ip-172-31-40-108 opt]# cat Dockerfile.calculator

```
FROM amazonlinux
RUN yum update && yum install -y go
ENV GO111MODULE=off
WORKDIR /opt/calculatorapp
COPY . .
RUN go build calculator.go
```

```
ENTRYPOINT ["/opt/calculatorapp/calculator"]
**********************************************************************

[root@ip-172-31-40-108 opt]# cat Dockerfile.ubuntu
FROM ubuntu

RUN apt-get update && apt-get install -y golang-go
ENV GO111MODULE=off
WORKDIR /opt/calculatorapp
COPY . .
RUN go build calculator.go

ENTRYPOINT ["/opt/calculatorapp/calculator"]

================================================

[root@ip-172-31-40-108 opt]# cat Dockerfile-multistage

FROM amazonlinux AS buildstage
RUN yum update && yum install -y go
ENV GO111MODULE=off
WORKDIR /opt/calculatorapp
COPY . .
RUN go build calculator.go

FROM scratch
COPY --from=buildstage /opt/calculatorapp/calculator /opt/calculatorapp/calculator
ENTRYPOINT ["/opt/calculatorapp/calculator"]

----------------------------------------


code is in my GIT HUB - clone it

git clone https://github.com/devopstraininghub/multi-stage-dockerfileex.git


Ref URLS :

https://docs.docker.com/build/building/multi-stage/
https://devopscube.com/reduce-docker-image-size/
======================================================================
docker network list /ls

docker network inspect bridge
```

docker network create --subnet 10.81.0.0/16 --gateway 10.81.0.1 --ip-range 10.81.3.0/24 --driver bridge batch13nw
docker run -itd --net batch15nw --ip 10.81.0.20 <iname>

docker network rm batch13nw

Give/Type the command to create a Docker Container?
docker run --rm -d --name app1 --publish 8080:80 \
-e APPNAME=app1 -e APPENV=dev -v /app1:/var/www/html \
nginx:latest

13. How to check the logs of a container?
docker logs container1 #To check stdout logs of the container.
docker logs -f container1

---
 mkdir calculatorapp
  94 cd calculatorapp/
  95 ll
  96 vim calapp.go
  97 yum install -y go
  98 GO111MODULE=off
  99 ll
 100 go build calapp.go
 101 ll
 102 ./calapp
 103 ll
 104 rm -rf calapp
 105 ll
 106 vim Dockerfile
 107 docker build -t calimg:v1 -f Dockerfile .
 108 docker iamges
 109 docker images
 110 docke run --rm -it calimg
 111 docker run --rm -it calimg
 112 docker run --rm -it calimg:v1
 113 ll
 114 vim Dockerfile
 115 mv calapp.go calculator.go
 116 ll
 117 docker build -t calimg:v1 -f Dockerfile .
 118 docker images
 119 docker run --rm -it calimg
 120 docker run --rm -it calimg:v1

```
121  ll
122  vim Dockerfile_multistage
123  ll
124  docker build -t calimg:v2 -f Dockerfile_multistage .
125  vim Dockerfile_multistage
126  docker build -t calimg:v2 -f Dockerfile_multistage .
127  vim Dockerfile_multistage
128  docker build -t calimg:v2 -f Dockerfile_multistage .
129  docker iamges
130  docker images
131  docker run --rm -it calimg:v2
132  docker ps -a
133  docker images
134  systmctl status docker
135  systemctl status docker
136  docker rmi $(docker images -q)
137  cd /opt/
138  ll
139  rm -rf *
140  ll
141  mkdir calculatorapp
142  ll
143  cd calculatorapp/
144  ll
145  vim calculator.go
146  yum update
147  yum install go -y
148  GO111MODULE=off
149  ll
150  go build calculator.go
151  ll
152  ./calculator
153  ll
154  rm -rf calculator
155  ll
156  vim Dockerfile
157  docker build -t calculatorapp:v1 -f Dockerfile .
158  docker images
159  docker history be8cb570ca06
160  docker images
161  docker run --rm -it --name ct1 calculatorapp:v1
162  ll
163  vim Dockerfile.multistage
164  docker buid -t calculatorapp:multistage_v2 -f Dockerfile.multistage .
165  vim Dockerfile.multistage
```

```
166  docker build -t calculatorapp:multistage_v2 -f Dockerfile.multistage .
167  docker images
168  docker run --rm -it --name ct2 calculatorapp:multistage_v2
169  hostname
170  hostname  -I
171  ifconfig
172  hostname  -I
173  ifconfig
174  docker network ls
175  docker ps -a
176  docker run -it --name ct1 ubuntu
177  docker run -it --name ct2 ubuntu
178  docker run -it --name ct3 ubuntu
179  docker run -it --name ct4 ubuntu
180  docker network ls
181  docker network inspect bridge
182  docker network create testnw
183  docker network ls
184  docker network inspect testnw
185  docker network create --subnet 10.81.0.0/16 batch15nw
186  docker network ls
187  docker network inspect batch15nw
188  docker run -it --name cc1 --network batch15nw ubuntu
189  docker run -it --name cc2 --network batch15nw ubuntu
190  docker run -it --name cc3 --network batch15nw --ip 10.81.8.88 ubuntu
191  docker network ls
192  docker network inspect batch15nw
193  docker network ls
194  docker network rm batch15nw
195  docker rm -f $(docker ps -qa)
196  docker network rm batch15nw
197  docker network rm testnw
198  docker network ls
199  ifconfig
200  docker run -it --name hsotnwtct --network host ubuntu
201  docker network ls
202  history
```

==========

4 APR 2025

DOCKER COMPOSE :

docker-compose.yml

- HELPS TO STRUCTURE YOUR COMMANDS
- SIMPLIFIES CONTAINER MANAGEMENT
- EASIER TO MAKE CHANGES AND SEE CONFIGURATION
- DECLARATIVE APPROACH - DEFINING THE DESIRED STATE

Docker Compose is a tool used to define and manage multi-container Docker applications. It allows you to configure your application's services, networks, and volumes in a single YAML file, making it easier to manage complex applications that require multiple containers.

Key Features and Uses:

Multi-Container Management:
Service Definition: Specify how each service (container) should be built, run, and networked with others.

Environment Configuration:
Use environment variables and configuration options to customize the behavior of your services.

Networking:
Automatically creates a network for your services, allowing them to communicate with each other using service names as hostnames.

Volume Management:
Easily manage persistent data by defining volumes for services.

Scaling:
Easily scale services up or down using simple commands, making it ideal for development and production environments.

docker pull wordpress
docker run -itd -p 8091:80 wordpress

==========

docker compose Installation :

sudo curl -L https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m) -o /usr/local/bin/docker-compose

```
sudo chmod +x /usr/local/bin/docker-compose

docker-compose version

==================================
docker-compose.yml /yaml

version: "3.3"

services:
  db:
    image: mysql:5.7
    volumes:
      - db_data:/var/lib/mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: somewordpress
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wordpress
      MYSQL_PASSWORD: wordpress

  wordpress:
    depends_on:
      - db
    image: wordpress:latest
    volumes:
      - wordpress_data:/var/www/html
    ports:
      - "8000:80"
    restart: always
    environment:
      WORDPRESS_DB_HOST: db
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress
      WORDPRESS_DB_NAME: wordpress
volumes:
  db_data: {}
  wordpress_data: {}

=========
  docker-compose up -d
  docker-compose down --volumes
```

==========
8 APR 2025:

SONARQUBE - SCA
& TRIVY

Ubuntu:

```
# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "${UBUNTU_CODENAME:-$VERSION_CODENAME}") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update

sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-
plugin -y
```

==========
to build :
mvn clean install
---

to scan using sonar:

mvn sonar:sonar
http://ec2-18-234-233-210.compute-1.amazonaws.com:9000/

sonar token :
squ_fb5fdb469f3cbfe23a88554ae7cd3ba7e3c29263
==========

```
stage('sonarqube scan') {
        steps {
           echo 'scanning project'
           sh 'ls -ltrh'

           sh ''' mvn sonar:sonar \\
```

```
                -Dsonar.host.url=http://ec2-18-234-233-210.compute-1.amazonaws.com:9000 \\
                -Dsonar.login=squ_fb5fdb469f3cbfe23a88554ae7cd3ba7e3c29263'''
        }
          }
```

==========

jenkins on ubuntu:

```
sudo apt update
sudo apt install -y openjdk-17-jdk
java -version


curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee \
  /usr/share/keyrings/jenkins-keyring.asc > /dev/null
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
  https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
  /etc/apt/sources.list.d/jenkins.list > /dev/null


sudo apt update
sudo apt install -y jenkins


sudo systemctl start jenkins
sudo systemctl enable jenkins
```

==========

TRIVY :

DOCKER IMAGE SCANNING :

Trivy is an open-source vulnerability scanner designed to detect security issues in container images, file systems, and Git repositories.

```
rpm -ivh https://github.com/aquasecurity/trivy/releases/download/v0.18.3/trivy_0.18.3_Linux-64bit.rpm

rpm -uvh
```

1. Scanning a Docker Image

```
  trivy image <image_name>
```

2. Scanning a Local Directory

trivy fs <directory_path>

3. Outputting Results in JSON Format

trivy image <image_name> --format json --output results.json

4. Scanning for Specific Severity Levels

trivy image <image_name> --severity HIGH,CRITICAL

==========
==========

Using Trivy in CI/CD:

```
pipeline {
    agent any

          tools{
      maven 'maven3'

    }

    stages {
        stage('Git checkout') {
            steps {
              git branch: 'main', url: 'https://github.com/devopstraininghub/mindcircuit14.git'
            }
        }
        stage('Compile stage maven') {
            steps {
                sh 'mvn clean install'
            }
        }

        stage('SONARQUBE ANALYSIS ') {
            steps {
                echo " SONARQUBE ANALYSIS"
                                        sh ''' mvn sonar:sonar \\
                    -Dsonar.host.url=http://3.80.128.201:9000/ \\
```

```
                -Dsonar.login=squ_464b30c7f49ad6b99a5d0cd66bb6b88ccc5a7df3'''

        }
      }

     stage('docker build') {
        steps {

              sh 'docker build -t devopshubg333/mc14:v1 .'

        }
      }

       stage('image scan using trivt') {
        steps {
          sh "trivy image --format json --output results.json devopshubg333/mc14:v1"

        }
      }

     stage('docker image push') {
        steps {
         script{
          withCredentials([string(credentialsId: 'dockerhub', variable: 'dockerhub')]) {
            sh 'docker login -u devopshubg333 -p ${dockerhub}'
         }
              sh 'docker push devopshubg333/mc14:v1'
           }

        }
      }

     }

                    }
```

==========

How do you pass secrets to a Docker image?

It’s not recommended to pass secrets directly to a Docker image. Instead, consider using the following methods:

Docker Secrets: Manage sensitive data in Docker Swarm.
Docker Configs: Store non-sensitive configuration data.
Docker Volumes: Use volumes to keep secrets outside the image.
Environment Variables: Set environment variables at runtime.
Docker Compose Secrets: Handle secrets within Docker Compose.
Kubernetes Secrets: Use Kubernetes for managing sensitive information.
Secret Management Services: Utilize services like AWS Secrets Manager, HashiCorp Vault.

Example:
To pass AWS Access Key and Secret Key to a Docker container, you can use:
docker run -e AWS_ACCESS_KEY_ID=AKxxxxxxxxxxxxxxxxx -e
AWS_SECRET_ACCESS_KEY=wJalxxxxxxxxxx myimage

For build-time secrets in Docker images, use build arguments:

docker build --build-arg AWS_ACCESS_KEY_ID=AKIAIxxxxx --build-arg
AWS_SECRET_ACCESS_KEY=wJalrXUtnFxxxxxxx.
These methods help ensure that sensitive data remains secure and is not included directly in the image.

=========

How to save/export a running container to tar file?

docker export CONT.NAME/ID > mcapp.tar

========

How to Convert existing image to tar file?

docker save imagename:latest -o tomcat_demo.tar
==========

How Import tar file as a image?

docker import < mcapp.tar

==========