DOCKER 03

1. Create a image from running container.
   - List docker images
   - Run the images in container
   - Httpd is image
   - Docker run -d -p 81:80 httpd
   - Docker ps - it is running
   - Now, create a image from running container
   - Docker ps
   - Take id
   - Docker commit (image id) (new name): {tag name}

```
[root@ip-172-31-14-76 ec2-user]# docker run -d -p 81:80 httpd
061a05318ab02b224d3ee9b7e512649282b5ec993d36b8ce6c504afae765cb54
[root@ip-172-31-14-76 ec2-user]# docker commit bee0c099fb40 khttpd:v1
sha256:f299dd3190d36e43929d5ef2c56447a03cfb180b0e8c84fa848b86522130f779
[root@ip-172-31-14-76 ec2-user]# docker images
REPOSITORY                                                      TAG       IMAGE ID       CREATED          SIZE
khttpd                                                          v1        f299dd3190d3   3 seconds ago    117MB
mynginx1                                                        v1        4093486913a1   9 minutes ago    152MB
ubuntu1                                                         own1      7283ef487814   13 minutes ago   78.1MB
kamal/myamazon-nginx                                            latest    3b41700f929c   22 hours ago     213MB
kamalguntur/myamazon-nginx                                      latest    3b41700f929c   22 hours ago     213MB
myamazon-nginx                                                  latest    3b41700f929c   22 hours ago     213MB
414691912691.dkr.ecr.us-east-1.amazonaws.com/myamazon-nginx     latest    3b41700f929c   22 hours ago     213MB
```

2 . Copy image from local machine to docker server and load the image.

- **Save image**
- docker save -o myamazon-nginx.tar myamazon-nginx:latest
- 2️ **Remove image**
- docker rmi myamazon-nginx:latest
- 3️ **Load image back**
- docker load -i myamazon-nginx.tar
- 4️ **Verify**
- docker images

```
[root@ip-172-31-14-76 ec2-user]# docker save -o myamazon-nginx.tar myamazon-nginx:latest
[root@ip-172-31-14-76 ec2-user]# docker rmi myamazon-nginx:latest
Untagged: myamazon-nginx:latest
[root@ip-172-31-14-76 ec2-user]# docker load -i myamazon-nginx.tar
Loaded image: myamazon-nginx:latest
[root@ip-172-31-14-76 ec2-user]# docker images
REPOSITORY                                          TAG       IMAGE ID       CREATED       SIZE
kimage                                              k1        d738e095707c   2 days ago    11.3MB
mynginx1                                            v1        4093486913a1   3 days ago    152MB
kamal/myamazon-nginx                                latest    3b41700f929c   4 days ago    213MB
```

3 . Create Docker image using alpine and customize with tomcat.

STEP 1: Create a project folder

mkdir alpine-tomcat

cd alpine-tomcat

Check:

pwd

- **STEP 2: Create Dockerfile**

vi Dockerfile

Press **i** and paste **ENTIRE content below** 👇

```
# STEP 1: Use Alpine Linux
FROM alpine:3.19

# STEP 2: Install Java and required tools
RUN apk update && \
    apk add --no-cache openjdk17-jre curl tar bash

# STEP 3: Set Tomcat environment variables
ENV CATALINA_HOME=/opt/tomcat
ENV PATH=$CATALINA_HOME/bin:$PATH

# STEP 4: Create Tomcat directory
RUN mkdir -p $CATALINA_HOME

# STEP 5: Download and extract Tomcat (WORKING URL)
```

RUN curl -fsSL https://archive.apache.org/dist/tomcat/tomcat-9/v9.0.85/bin/apache-tomcat-9.0.85.tar.gz \
| tar -xz --strip-components=1 -C $CATALINA_HOME

STEP 6: Remove default webapps (optional)
RUN rm -rf $CATALINA_HOME/webapps/*

STEP 7: Expose Tomcat port
EXPOSE 8080

STEP 8: Start Tomcat
CMD ["catalina.sh", "run"]
Save and exit:

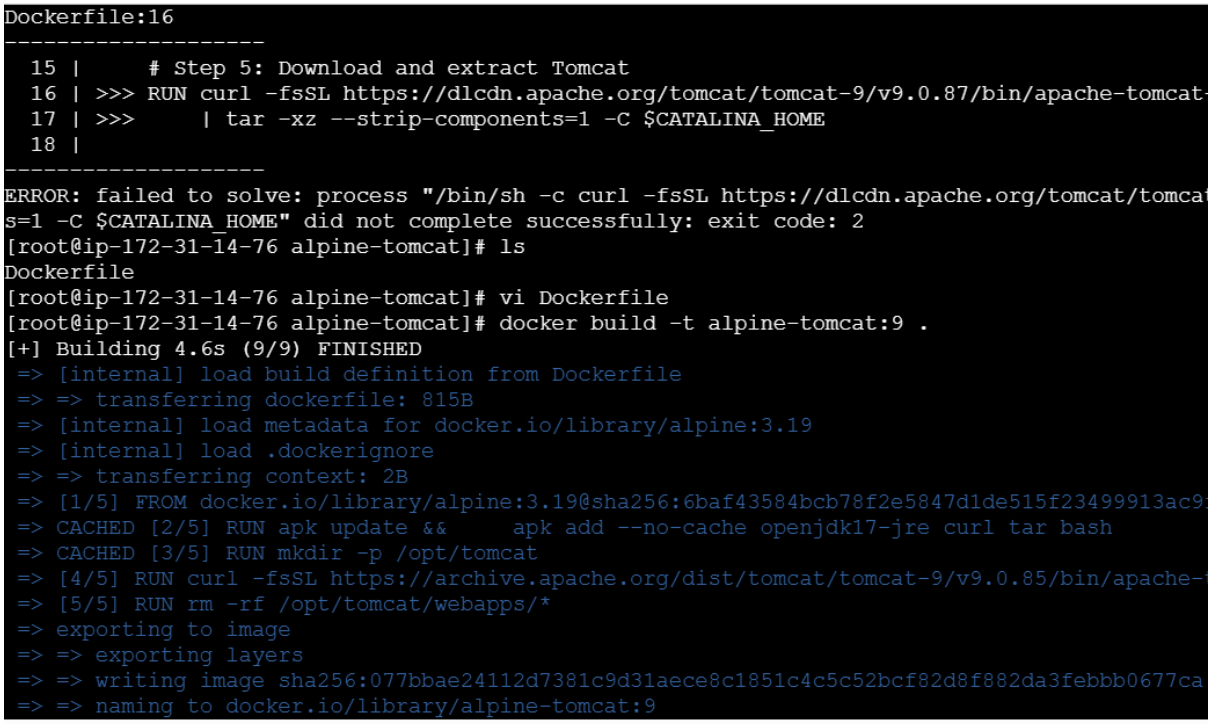
- Esc
- :wq
- Enter

🔹 **STEP 3: Build Docker image**

Make sure Dockerfile exists:
ls
Build image:
docker build -t alpine-tomcat:9 .
Wait until build finishes.
Verify:
docker images | grep alpine-tomcat

🔹 **STEP 4: Run Tomcat container**

docker run -d --name mytomcat -p 8080:8080 alpine-tomcat:9
Check container:
docker ps



- **STEP 5: Check Tomcat logs**

docker logs mytomcat

Expected:

Server startup in xxxx ms

- **STEP 6: Access Tomcat in browser**

Open:

http://<EC2_PUBLIC_IP>:8080

⚠️ EC2 Security Group:

- Allow **TCP 8080**
- Source: 0.0.0.0/0

4 . Create single stage and multi stage docker file using this source code: https://github.com/betawins/multi-stage-example.git

SINGLE-STAGE Dockerfile

Create file

vi Dockerfile.single

Press i and paste 👇

```
FROM golang:1.20

WORKDIR /app

COPY . .

RUN go build -o myapp main.go

EXPOSE 8080

CMD ["./myapp"]
```

Save:

Esc → :wq → Enter

Build single-stage image

```
docker build -f Dockerfile.single -t go-single .
```

Run single-stage container

```
docker run -d -p 8080:8080 --name go-single-container go-single
```

Test in browser:

http://<EC2_PUBLIC_IP>:8080

🔹 **STEP 3: MULTI-STAGE Dockerfile**

Create file

vi Dockerfile.multi

Press i and paste 👇

Stage 1: Build

FROM golang:1.20 AS builder

WORKDIR /app

COPY . .

RUN CGO_ENABLED=0 GOOS=linux GOARCH=amd64 go build -o myapp main.go

Stage 2: Runtime

FROM alpine:3.19

WORKDIR /app

COPY --from=builder /app/myapp .

EXPOSE 8080

CMD ["./myapp"]

Save:

Esc → :wq → Enter

Build multi-stage image

docker build -f Dockerfile.multi -t go-multi .

Run multi-stage container

docker run -d -p 8081:8080 --name go-multi-container go-multi

```
------
 > [4/5] RUN go mod download:
0.809 go: no modules specified (see 'go help mod download')
------
Dockerfile.single:13
--------------------
  11 |
  12 |     # Step 4: Download dependencies
  13 | >>> RUN go mod download
  14 |
  15 |     # Step 5: Build application
--------------------
ERROR: failed to solve: process "/bin/sh -c go mod download" did not complete successfully: exit code: 1
[root@ip-172-31-14-76 multi-stage-example]# docker images | grep go-single
[root@ip-172-31-14-76 multi-stage-example]# ls
Dockerfile  Dockerfile.single  README.md  mvnw  mvnw.cmd  pom.xml  src
[root@ip-172-31-14-76 multi-stage-example]# vi Dockerfile.single
[root@ip-172-31-14-76 multi-stage-example]# vi Dockerfile.single
[root@ip-172-31-14-76 multi-stage-example]# docker build -f Dockerfile.single -t go-single .
[+] Building 0.8s (8/8) FINISHED
 => [internal] load build definition from Dockerfile.single
 => => transferring dockerfile: 295B
 => [internal] load metadata for docker.io/library/golang:1.20
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/4] FROM docker.io/library/golang:1.20@sha256:8f9af7094d0cb27cc783c697ac5ba25efdc4da35f8526db21f7aebb0b0b4f18a
 => [internal] load build context
 => => transferring context: 7.54kB
```

5 . Install docker compose and execute sample application.

Download Docker Compose

sudo curl -L "https://github.com/docker/compose/releases/download/v2.29.2/docker-compose-linux-x86_64" \

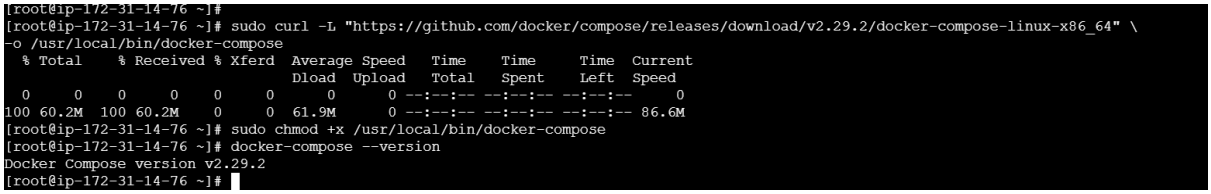
-o /usr/local/bin/docker-compose

✅ STEP 3: Give execute permission

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

✅ STEP 4: Verify installation

docker-compose –version



Create project directory

mkdir docker-compose-demo

cd docker-compose-demo

✅ STEP 6: Create docker-compose.yml file

vi docker-compose.yml

Press i and paste 👇

version: "3.8"

services:

  web:

    image: nginx:latest

    container_name: my-nginx

    ports:

      - "8080:80"

Save:

Esc → :wq → Enter

✅ STEP 7: Start application using Docker Compose

docker-compose up -d

Expected output:

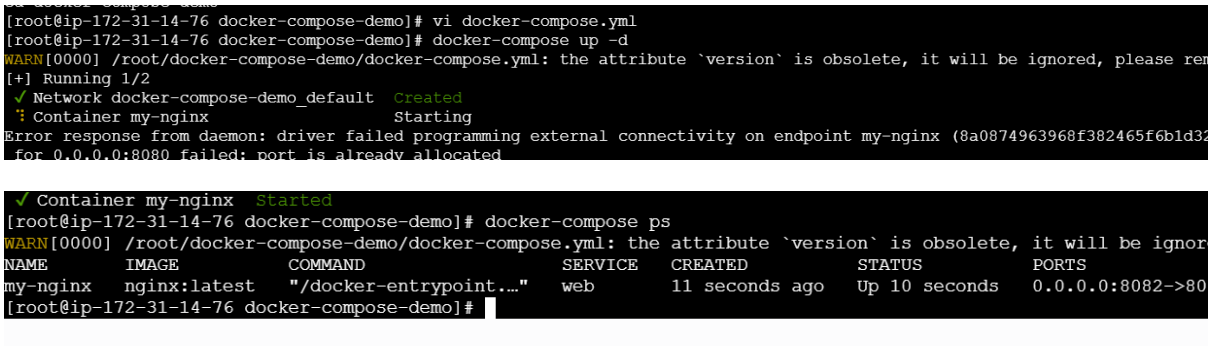
Creating my-nginx ... done

✅ STEP 8: Verify container

docker-compose ps

or

docker ps



6 . Implement solution to scan images when pushed to docker registry.

Run a image

Go to repo in dockerhub

Setting

Select first option

Docker commands

Public view

To push a new tag to this repository:

```
docker push kamalguntur/myamazon-nginx:tagname
```

General Tags Image Management BETA Collaborators Webhooks Settings

Image security insight settings

Features and controls that help you uncover, understand, and fix issues with your container images in Docker Hub

You've reached the repository limit set by your current plan. To secure more repositories with Docker Scout, **upgrade your plan.** Upgrade

- (•) Docker Scout image analysis RECOMMENDED
  Know when new CVEs impact your images, learn where they're introduced, and get recommendations for remediation options. Enable repos in bulk on Scout Dashboard
- ( ) Static scanning
  Images will be scanned once when pushed and the vulnerability report saved at that point in time.
- ( ) None

General Tags Image Management BETA Collaborators Webhooks Settings

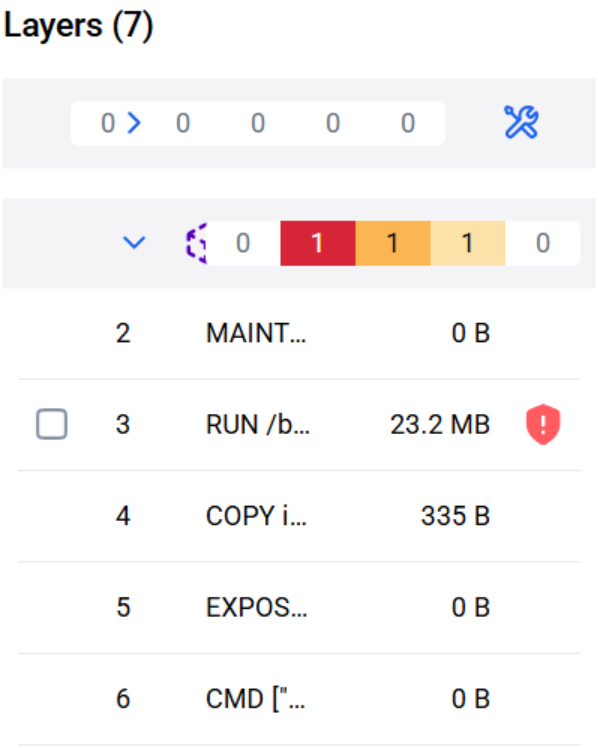
Tags

Analyzed by docker scout

This repository contains 1 tag(s).

Tag	OS	Type	Vulnerabilities					Pulled	Pushed
latest		Image	0	1	1	1	0	3 days	4 days

See all



1. Create a Jenkins pipeline to create a docker image and push the image to Docker hub.

- **STEP 1: Create an EC2 / Linux server**
- Launch EC2 (Amazon Linux / Ubuntu)
- Open ports in Security Group:
- **22** (SSH)
- **8080** (Jenkins)
- SSH into the server
- ---
- 🔹 **STEP 2: Install Docker on the server**
- sudo yum install docker -y # Amazon Linux
- sudo systemctl start docker
- sudo systemctl enable docker
- docker --version
- ---

- 🔹 **STEP 3: Run Jenkins using Docker (correct way)**
- docker run -d \
- --name jenkins \
- -p 8080:8080 \
- -p 50000:50000 \
- -v jenkins_home:/var/jenkins_home \
- -v /var/run/docker.sock:/var/run/docker.sock \
- jenkins/jenkins:lts
- 👉 This allows Jenkins to talk to Docker on the host.
- ---
- 🔹 **STEP 4: Install Docker CLI inside Jenkins container**
- docker exec -it -u root jenkins bash
- Inside container:
- apt update
- apt install -y docker.io
- docker --version
- exit
- ---
- 🔹 **STEP 5: Fix Docker permission for Jenkins user**
- docker exec -it -u root jenkins bash
- groupadd docker || true
- usermod -aG docker jenkins
- chmod 666 /var/run/docker.sock
- exit
- Restart Jenkins:
- docker restart jenkins
- Verify:
- docker exec -it jenkins docker ps
- ✅ If this works, Jenkins can run Docker.

```
[root@ip-172-31-21-9 ec2-user]# docker restart jenkins
jenkins
[root@ip-172-31-21-9 ec2-user]# docker exec -it jenkins docker ps
CONTAINER ID   IMAGE                COMMAND                  CREATED         STATUS         PORTS
               NAMES
5ca211faa3b2   jenkins/jenkins:lts   "/usr/bin/tini -- /u…"   3 minutes ago   Up 6 seconds   0.0.0.0:8080->8080/tcp, :::8080->8080/tcp, 0.0.0.0:50000->50000/tcp, :::5000
0->50000/tcp   jenkins
[root@ip-172-31-21-9 ec2-user]# docker exec -it jenkins cat /var/jenkins_home/secrets/initialAdminPassword
c74f11cfe9414e5dbc4686dc53174ea1
[root@ip-172-31-21-9 ec2-user]# ls
docker-jenkins-demo
[root@ip-172-31-21-9 ec2-user]# docker exec -it jenkins bash
jenkins@5ca211faa3b2:/$ cat <<EOF > Dockerfile
FROM nginx:alpine
COPY index.html /usr/share/nginx/html/index.html
EOF
bash: Dockerfile: Permission denied
jenkins@5ca211faa3b2:/$ exit
exit
[root@ip-172-31-21-9 ec2-user]# docker exec -it -u root jenkins bash
root@5ca211faa3b2:/# mkdir -p /var/jenkins_home/workspace/docker
cd /var/jenkins_home/workspace/docker
root@5ca211faa3b2:/var/jenkins_home/workspace/docker# cat <<EOF > Dockerfile
```

- ---
- 🔹 **STEP 6: Open Jenkins in browser**
- http://<EC2-PUBLIC-IP>:8080
- Get initial password:
- docker exec -it jenkins cat /var/jenkins_home/secrets/initialAdminPassword
- Install suggested plugins
- Create admin user
- Login to Jenkins
- ---
- 🔹 **STEP 7: Add Docker Hub credentials in Jenkins**
- Jenkins → **Manage Jenkins → Credentials → Global → Add Credentials**
- Kind: Username with password
- Username: Docker Hub username
- Password: Docker Hub password / token
- ID: docker_hub_cr
- Save

Global credentials (unrestricted)

Credentials that should be available irrespective of domain specification to requirements matching.

	ID	Name	Kind	Descriptio
	docker_hub_cr	kamalguntur/******	Username with password	

- ---

- 🔹 **STEP 8: Create Jenkins Pipeline job**
- Jenkins Dashboard → **New Item**
- Name: docker
- Type: **Pipeline**
- Click **OK**
- ---
- 🔹 **STEP 9: Create Dockerfile in Jenkins workspace**
- Go inside Jenkins container:
- docker exec -it -u root jenkins bash
- cd /var/jenkins_home/workspace/docker
- Create files:
- cat <<EOF > Dockerfile
- FROM nginx:alpine
- COPY index.html /usr/share/nginx/html/index.html
- EOF
- cat <<EOF > index.html
- <h1>Hello from Jenkins Docker Pipeline</h1>
- EOF
- Verify:
- ls
- Exit:
- exit
- ---
- 🔹 **STEP 10: Paste Jenkins Pipeline (FINAL)**
- pipeline {
- agent any
-
- environment {
- IMAGE_NAME = "kamalguntur/kamal"

```
        IMAGE_TAG  = "tagname"
        DOCKER_CREDS = "docker_hub_cr"
    }

    stages {

        stage('Build Docker Image') {
            steps {
                sh "docker build -t ${IMAGE_NAME}:${IMAGE_TAG} ."
            }
        }

        stage('Login to Docker Hub') {
            steps {
                withCredentials([usernamePassword(
                    credentialsId: 'docker_hub_cr',
                    usernameVariable: 'DOCKER_USER',
                    passwordVariable: 'DOCKER_PASS'
                )]) {
                    sh 'echo $DOCKER_PASS | docker login -u
$DOCKER_USER --password-stdin'
                }
            }
        }

        stage('Push Docker Image') {
            steps {
                sh "docker push ${IMAGE_NAME}:${IMAGE_TAG}"
            }
```

- }
- }
- }
- Click **Save**

Pipeline script

Script ?

```
pipeline {
    agent any

    environment {
        IMAGE_NAME = "kamalguntur/kamal"
        IMAGE_TAG  = "tagname"
        DOCKER_CREDS = "docker_hub_cr"
    }

    stages {

        stage('Build Docker Image') {
            steps {
                sh "docker build -t ${IMAGE_NAME}:${IMAGE_TAG} ."
            }
```

-
- ---
- 🔹 **STEP 11: Run the pipeline**
- Click **Build Now**
- Expected output:
- Build Docker Image SUCCESS
- Login to Docker Hub SUCCESS
- Push Docker Image SUCCESS
- ---
- 🔹 **STEP 12: Verify on Docker Hub**
- Open:
- https://hub.docker.com/r/kamalguntur/kamal
- You should see:
- kamal:tagname

Repositories / kamal / General

kamalguntur/kamal

Last pushed less than a minute ago • Repository size: 21.9 MB • 0 • 0

my own repo and own image and container created

Add a category

General Tags Image Management BETA Collaborators Webhooks Settings

Tags

DOCKER SCOUT INACTIVE
Activate

This repository contains 0 tag(s).

Tag	OS	Type	Pulled	Pushed
tagname		Image	less than 1 day	2 minutes

See all

-