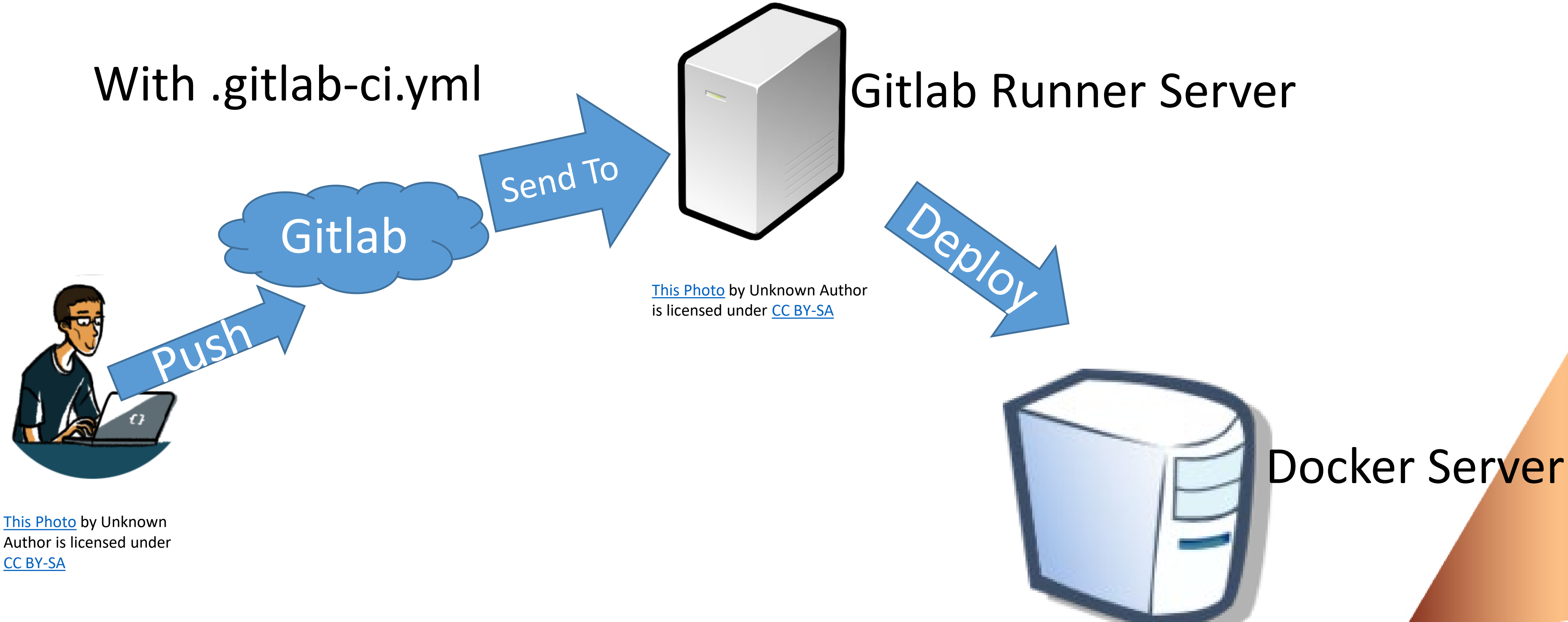
Docker CI

SE234 Advance Software Development

College of Arts, Media and Technology
Chiang Mai University

Automated Deployment



With .gitlab-ci.yml

Push

Gitlab

Send To

Gitlab Runner Server

Deploy

Docker Server

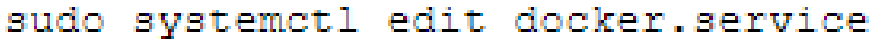
This Photo by Unknown Author is licensed under CC BY-SA

This Photo by Unknown Author is licensed under CC BY-SA

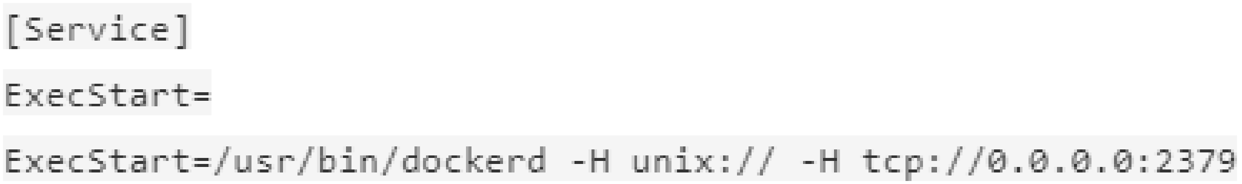
This Photo by Unknown Author is licensed under CC BY-SA

Docker Machine

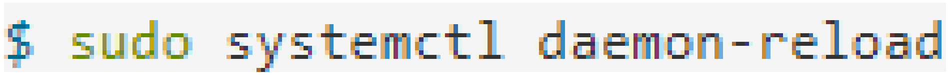
- Docker can be run from other computer
- The dockerd must be provided
- The port specify where to run the docker command



```
sudo systemctl edit docker.service
```

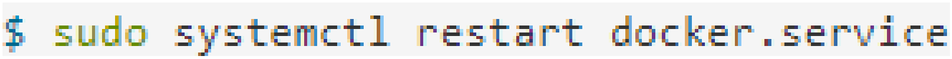


```
[Service]
ExecStart=
ExecStart=/usr/bin/dockerd -H unix:// -H tcp://0.0.0.0:2379
```

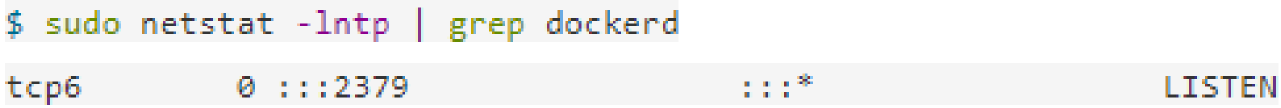


```
$ sudo systemctl daemon-reload
```

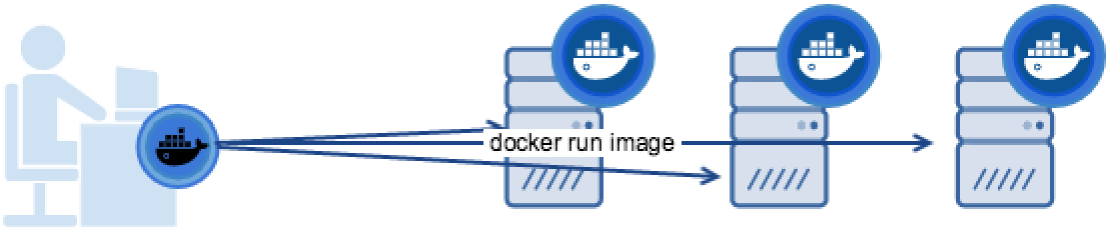
Docker Machine



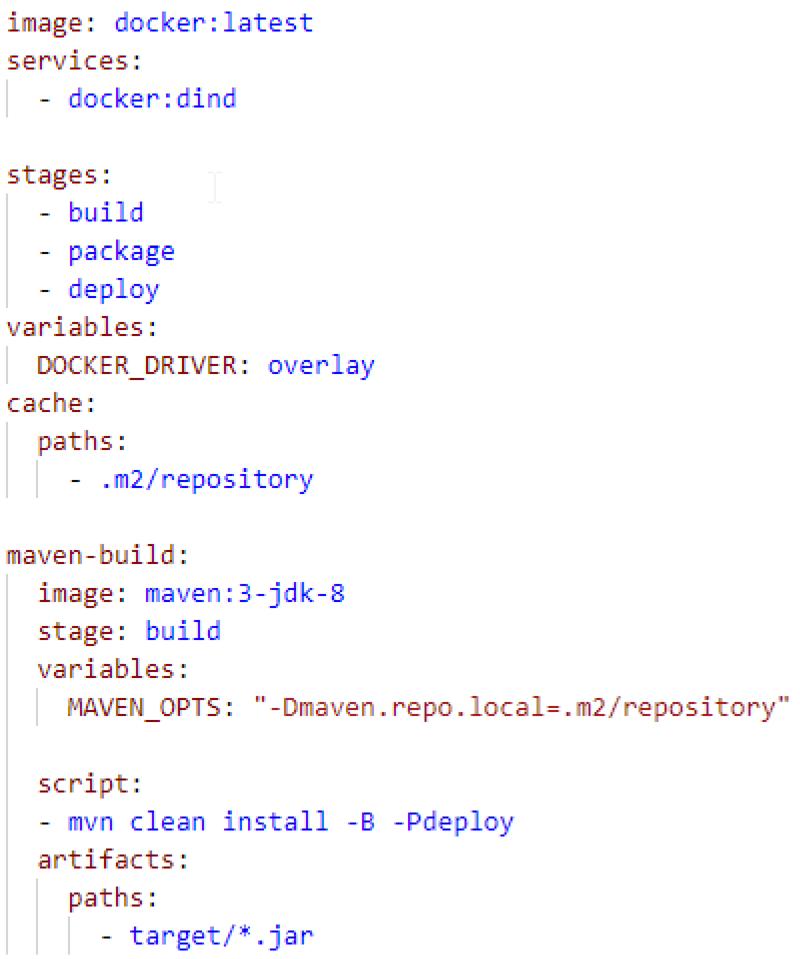
```
$ sudo systemctl restart docker.service
```



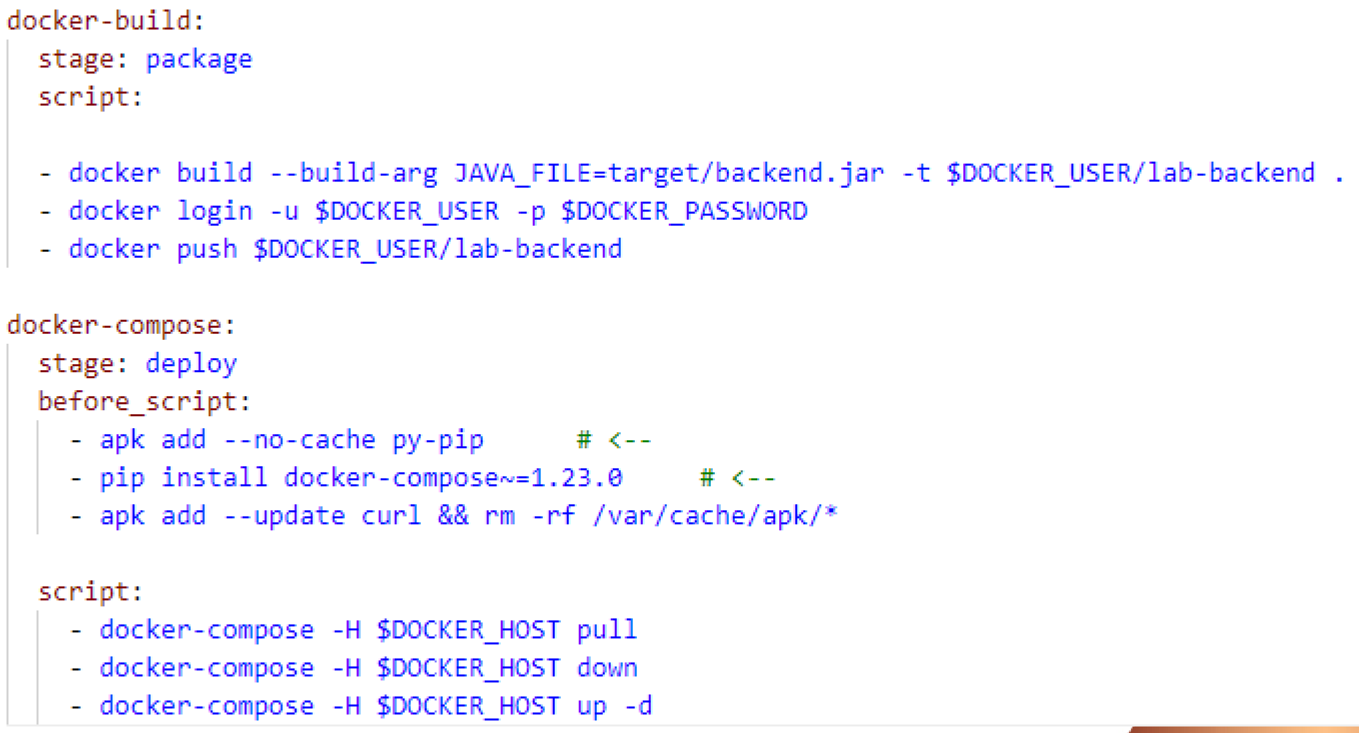
```
$ sudo netstat -lntp | grep dockerd
tcp6       0 :::2379                :::*                   LISTEN
```



Build First

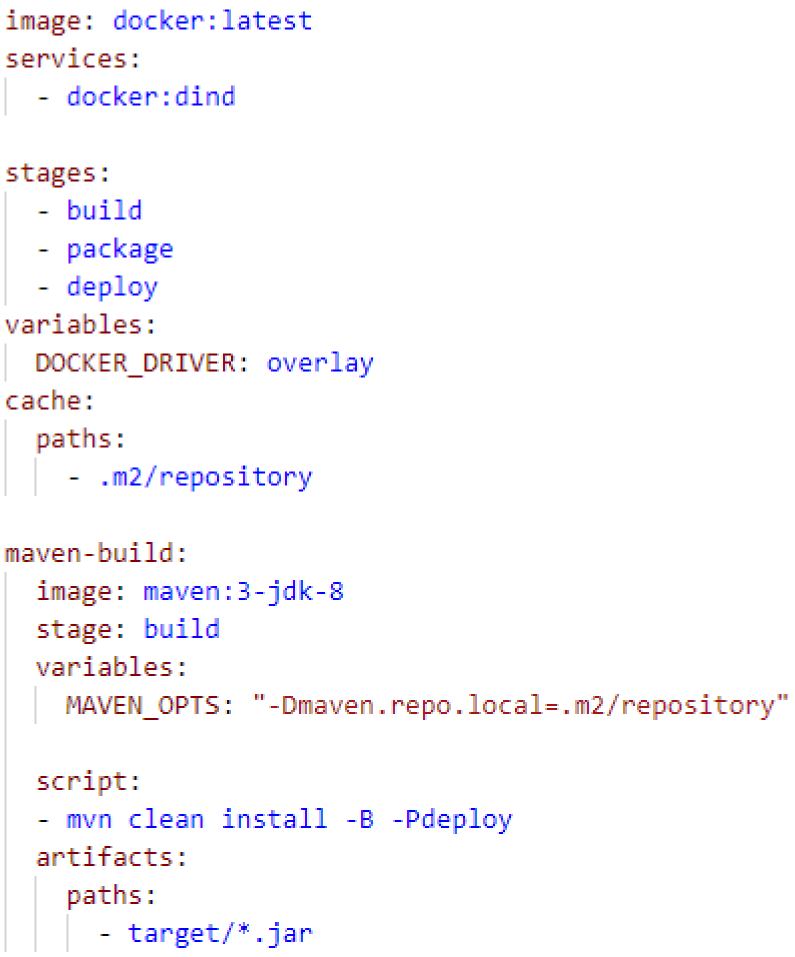


```yaml
image: docker:latest
services:
  - docker:dind

stages:
  - build
  - package
  - deploy
variables:
  DOCKER_DRIVER: overlay
cache:
  paths:
    - .m2/repository

maven-build:
  image: maven:3-jdk-8
  stage: build
  variables:
    MAVEN_OPTS: "-Dmaven.repo.local=.m2/repository"

  script:
  - mvn clean install -B -Pdeploy
  artifacts:
    paths:
      - target/*.jar
```



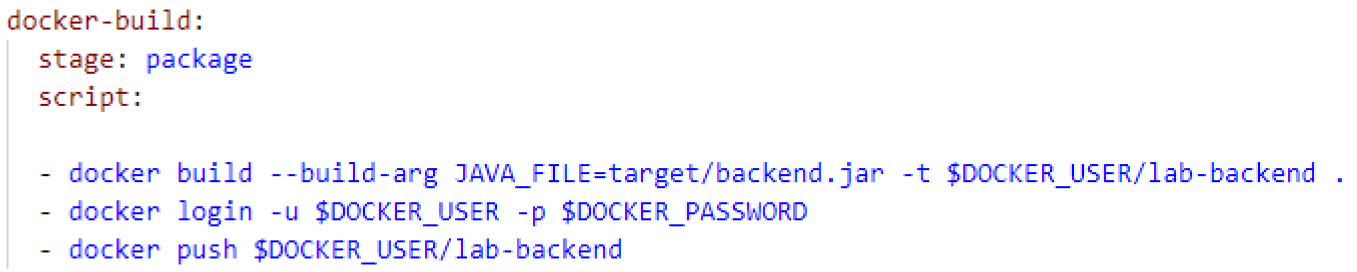
```yaml
docker-build:
  stage: package
  script:

  - docker build --build-arg JAVA_FILE=target/backend.jar -t $DOCKER_USER/lab-backend .
  - docker login -u $DOCKER_USER -p $DOCKER_PASSWORD
  - docker push $DOCKER_USER/lab-backend

docker-compose:
  stage: deploy
  before_script:
    - apk add --no-cache py-pip      # <--
    - pip install docker-compose~=1.23.0     # <--
    - apk add --update curl && rm -rf /var/cache/apk/*

  script:
    - docker-compose -H $DOCKER_HOST pull
    - docker-compose -H $DOCKER_HOST down
    - docker-compose -H $DOCKER_HOST up -d
```

Build First

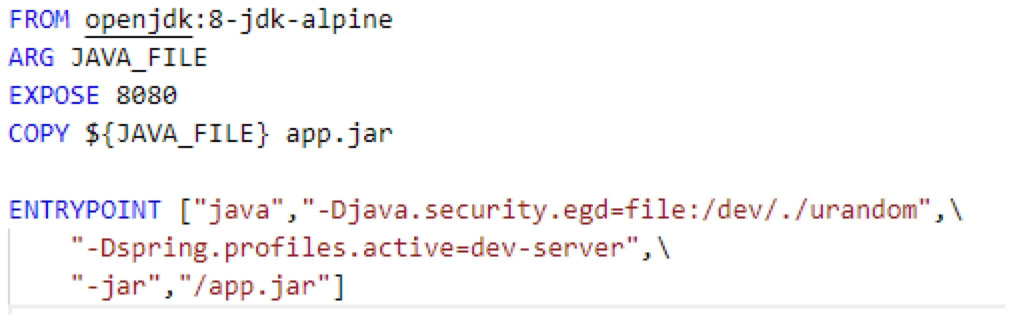


```
image: docker:latest
services:
  - docker:dind

stages:
  - build
  - package
  - deploy
variables:
  DOCKER_DRIVER: overlay
cache:
  paths:
    - .m2/repository

maven-build:
  image: maven:3-jdk-8
  stage: build
  variables:
    MAVEN_OPTS: "-Dmaven.repo.local=.m2/repository"

  script:
  - mvn clean install -B -Pdeploy
  artifacts:
    paths:
      - target/*.jar
```

Then Create Image

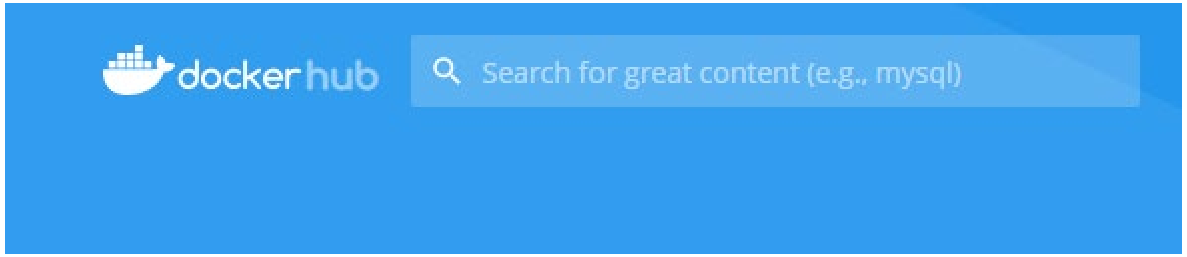


```
docker-build:
  stage: package
  script:

  - docker build --build-arg JAVA_FILE=target/backend.jar -t $DOCKER_USER/lab-backend .
  - docker login -u $DOCKER_USER -p $DOCKER_PASSWORD
  - docker push $DOCKER_USER/lab-backend
```



```
FROM openjdk:8-jdk-alpine
ARG JAVA_FILE
EXPOSE 8080
COPY ${JAVA_FILE} app.jar

ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom",\
    "-Dspring.profiles.active=dev-server",\
    "-jar","/app.jar"]
```

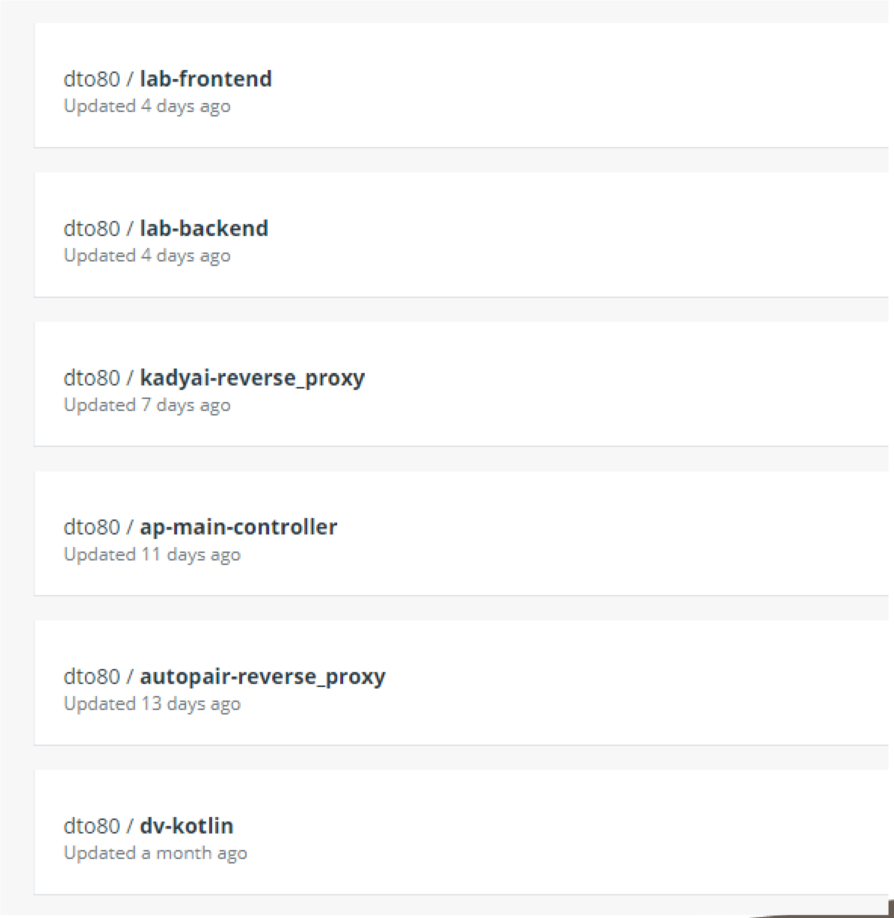
Docker Remote Repository



- Maintain most of the public docker images
- We can use other images as well

Repository for all Docker images

- Your own



dto80 / **lab-frontend**
Updated 4 days ago

dto80 / **lab-backend**
Updated 4 days ago

dto80 / **kadyai-reverse_proxy**
Updated 7 days ago

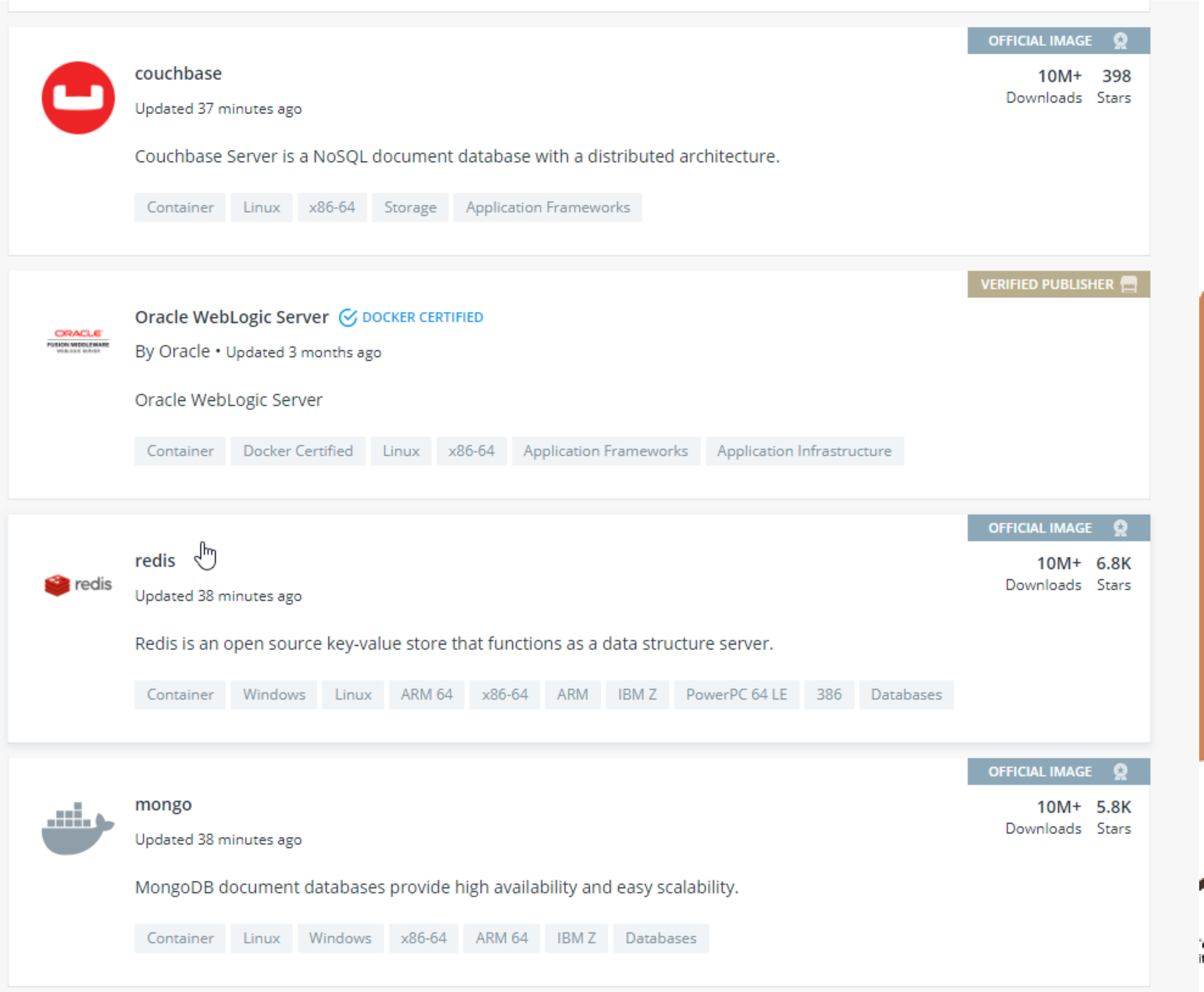
dto80 / **ap-main-controller**
Updated 11 days ago

dto80 / **autopair-reverse_proxy**
Updated 13 days ago

dto80 / **dv-kotlin**
Updated a month ago

Repository for all Docker images

- Open source application



OFFICIAL IMAGE

couchbase

Updated 37 minutes ago

10M+ Downloads 398 Stars

Couchbase Server is a NoSQL document database with a distributed architecture.

Container | Linux | x86-64 | Storage | Application Frameworks

VERIFIED PUBLISHER

Oracle WebLogic Server DOCKER CERTIFIED

By Oracle • Updated 3 months ago

Oracle WebLogic Server

Container | Docker Certified | Linux | x86-64 | Application Frameworks | Application Infrastructure

OFFICIAL IMAGE

redis

Updated 38 minutes ago

10M+ Downloads 6.8K Stars

Redis is an open source key-value store that functions as a data structure server.

Container | Windows | Linux | ARM 64 | x86-64 | ARM | IBM Z | PowerPC 64 LE | 386 | Databases

OFFICIAL IMAGE

mongo

Updated 38 minutes ago

10M+ Downloads 5.8K Stars

MongoDB document databases provide high availability and easy scalability.

Container | Linux | Windows | x86-64 | ARM 64 | IBM Z | Databases

Docker container run

- Will search the local repository
- By default
  - If none of the image is in the local computer
  - Download from the Docker hub

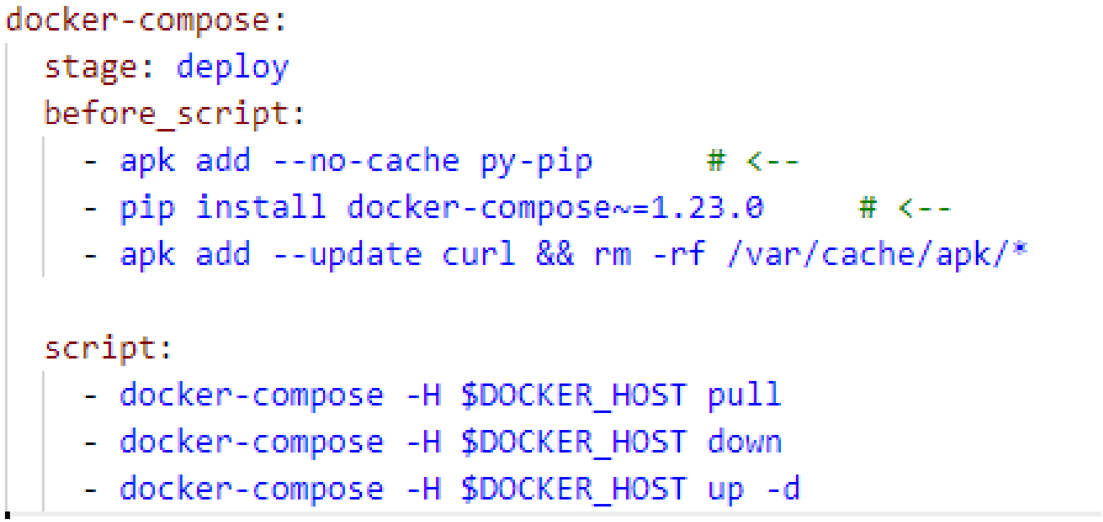
Docker Push

- Create the image in the local computer
- Push it to the Docker hub
- The name must start with the docker hub username
- And then the slash (/) and then any name
- Tag can be used to version the images
  - dto80/lab:latest dto80/lab:1.0

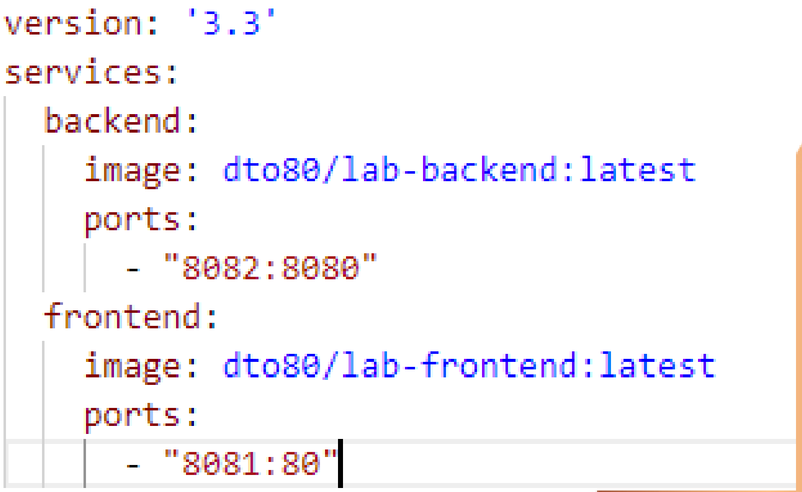
Docker pull

- If the image is in the local computer
  - When run the container it will not download the image
  - Even there is a newer version
- Use the pull command to force the docker machine to check the new version of the images (especially when you do not put the tag, or use the latest tag)
- If the newer version is in the docker-hub, docker will download a new version

Run Container



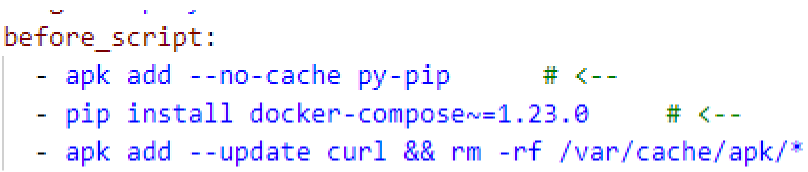
```
docker-compose:
  stage: deploy
  before_script:
    - apk add --no-cache py-pip      # <--
    - pip install docker-compose~=1.23.0     # <--
    - apk add --update curl && rm -rf /var/cache/apk/*

  script:
    - docker-compose -H $DOCKER_HOST pull
    - docker-compose -H $DOCKER_HOST down
    - docker-compose -H $DOCKER_HOST up -d
```



```
version: '3.3'
services:
  backend:
    image: dto80/lab-backend:latest
    ports:
      - "8082:8080"
  frontend:
    image: dto80/lab-frontend:latest
    ports:
      - "8081:80"
```

Docker-compose

- Not provided by Gitlab runner
  - Provided by ourselves



```
before_script:
  - apk add --no-cache py-pip      # <--
  - pip install docker-compose~=1.23.0     # <--
  - apk add --update curl && rm -rf /var/cache/apk/*
```

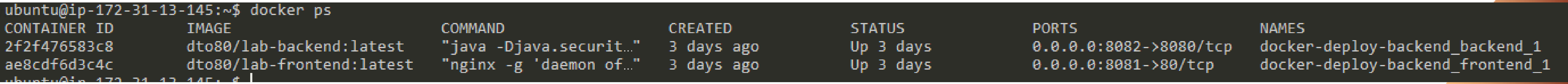
- Before script
  - Define that it’s not a script
  - But we need to prepare

Docker-compose pull

- Force to pull all images in the docker-compose file

Docker compose down

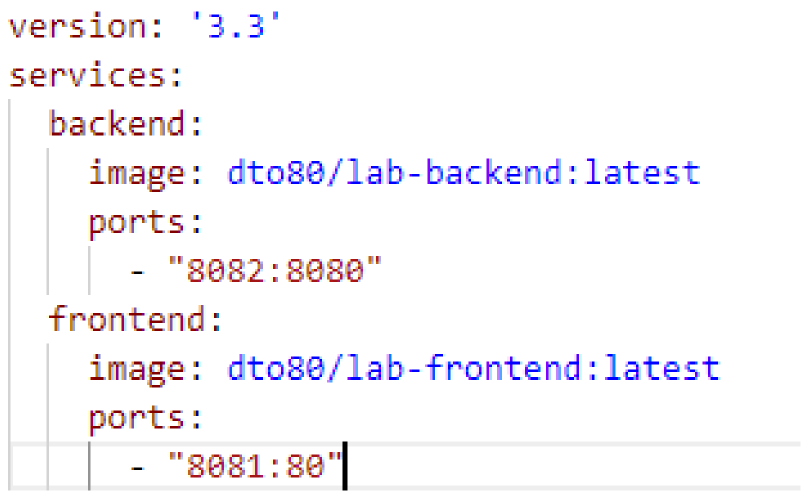
- Force all previous container in the docker-file to be down
- How do we know which container will be down?



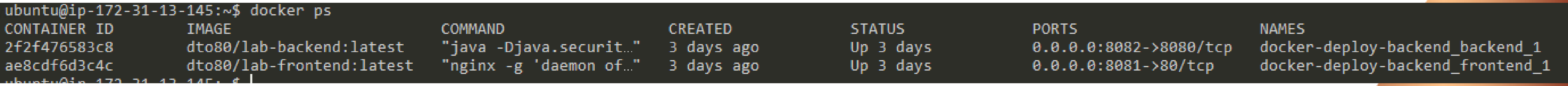
```
ubuntu@ip-172-31-13-145:~$ docker ps
CONTAINER ID        IMAGE                        COMMAND                  CREATED             STATUS              PORTS                    NAMES
2f2f476583c8        dto80/lab-backend:latest     "java -Djava.securit…"   3 days ago          Up 3 days           0.0.0.0:8082->8080/tcp   docker-deploy-backend_backend_1
ae8cdf6d3c4c        dto80/lab-frontend:latest    "nginx -g 'daemon of…"   3 days ago          Up 3 days           0.0.0.0:8081->80/tcp     docker-deploy-backend_frontend_1
ubuntu@ip-172-31-13-145:~$ 
```

Docker compose up

- Run all the images in the containers



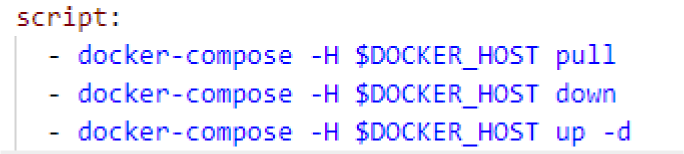
```yaml
version: '3.3'
services:
  backend:
    image: dto80/lab-backend:latest
    ports:
      - "8082:8080"
  frontend:
    image: dto80/lab-frontend:latest
    ports:
      - "8081:80"
```



```
ubuntu@ip-172-31-13-145:~$ docker ps
CONTAINER ID        IMAGE                        COMMAND                  CREATED             STATUS              PORTS                    NAMES
2f2f476583c8        dto80/lab-backend:latest     "java -Djava.securit…"   3 days ago          Up 3 days           0.0.0.0:8082->8080/tcp   docker-deploy-backend_backend_1
ae8cdf6d3c4c        dto80/lab-frontend:latest    "nginx -g 'daemon of…"   3 days ago          Up 3 days           0.0.0.0:8081->80/tcp     docker-deploy-backend_frontend_1
ubuntu@ip-172-31-13-145:~$
```

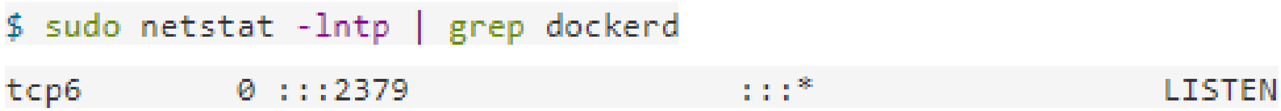
Docker compose Host

- -H command
- Specify where the operation will be run
- Variable show the host number



```
script:
  - docker-compose -H $DOCKER_HOST pull
  - docker-compose -H $DOCKER_HOST down
  - docker-compose -H $DOCKER_HOST up -d
```

- Define by tcp://1.1.1.1:2379



```
$ sudo netstat -lntp | grep dockerd
```

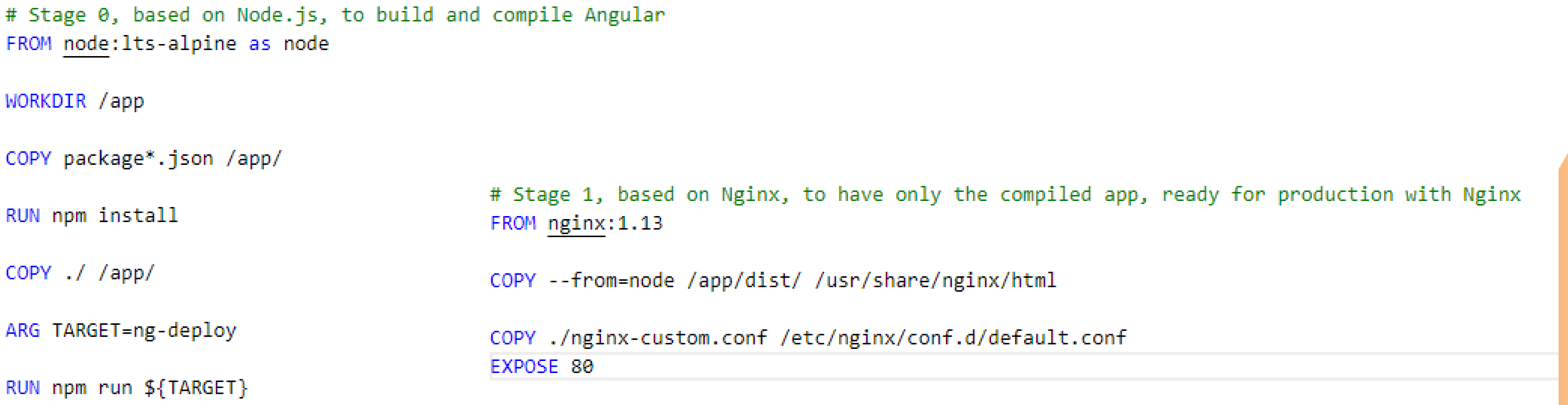
```
tcp6       0 :::2379                  :::*                    LISTEN
```

Hint

- Git lab CI
- Docker in Docker
  - Not all required resources are provided
  - Need to Install it manually

Multi-Stage Build

- Sometime build may be run by docker
  - Not by the gitlab-ci



```
# Stage 0, based on Node.js, to build and compile Angular
FROM node:lts-alpine as node

WORKDIR /app

COPY package*.json /app/

RUN npm install

COPY ./ /app/

ARG TARGET=ng-deploy

RUN npm run ${TARGET}
```

```
# Stage 1, based on Nginx, to have only the compiled app, ready for production with Nginx
FROM nginx:1.13

COPY --from=node /app/dist/ /usr/share/nginx/html

COPY ./nginx-custom.conf /etc/nginx/conf.d/default.conf
EXPOSE 80
```

Q & A

IF YOU HAVE 10 CHOCOLATE CAKES AND
SOMEONE ASKS FOR 2, HOW MANY DO U HAVE LEFT?
10...THAT BITCH AIN'T GETTING SHIT!!!