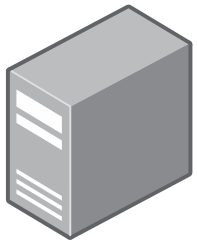
The Four Step Process

CODE

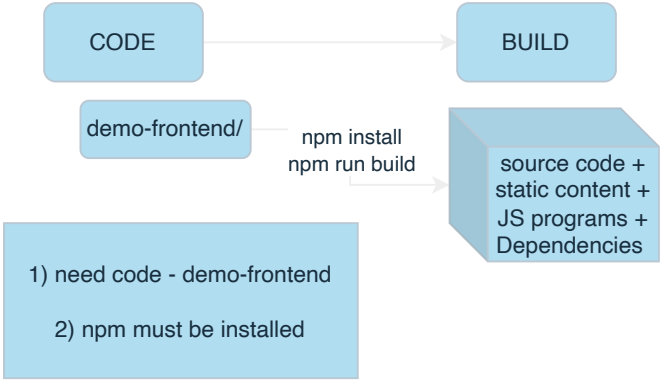
BUILD (artifacts, executable)

STORE

DEPLOY



BUILD TOOLS



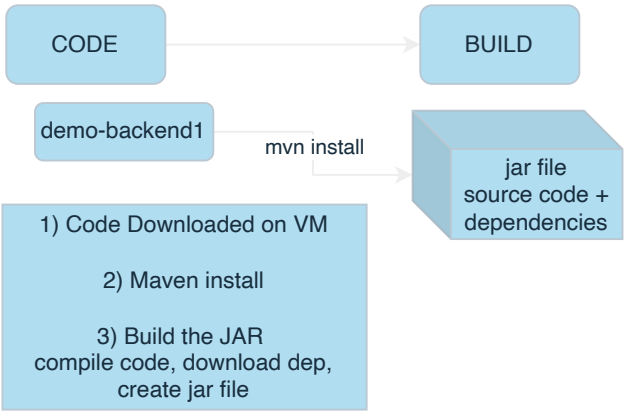
BUILD THE FRONTEND PROJECT ON THE AWS MACHINE

1) Copy the code from GitHub
git clone https://github.com/TekspotEdu/microserviceapp.git

2) Install NPM
apt-get update && apt-get install -y npm

3) Build the frontend project
cd microserviceapp/demo-frontend
npm install # Download all dependencies to a local folder "node_nodules"
npm run build # Create the final webapplication folder "build"

BUILD TOOL - Maven

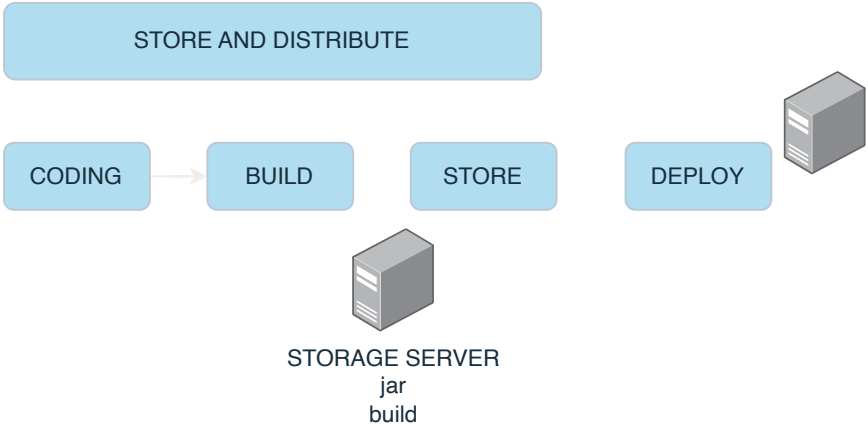


BUILD JAVA PROJECT ON AWS MACHINE

1) Clone the code
git clone https://github.com/TekspotEdu/microserviceapp.git

2) Install Maven
apt-get update && apt-get install -y maven

3) Build the java project using maven
cd microserviceapp/demo-backend1
mvn install # compile code, Download dep, jar file created in a folder "target"



SETUP the AWS VM as a STORAGE SERVER

1) Login to the VM and compress the build folder

```
cd microserviceapp/demo-frontend
tar -czvf demo-frontend-1.0.0.tar.gz build
```

2) Setup the machine as a Webserver

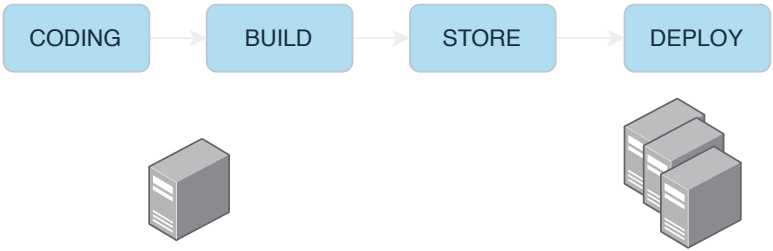
```
#install and start nginx
apt-get update && apt-get install -y nginx
service nginx start

# copy the files to /var/www/html
cd microserviceapp/demo-frontend
cp demo-frontend-1.0.0.tar.gz /var/www/html
cd ../demo-backend1
cp target/sentiment-analysis-web-0.0.2-SNAPSHOT.jar /var/www/html
```

3) Open Port 80 on AWS

http://13.127.234.6/demo-frontend-1.0.0.tar.gz

http://13.127.234.6/sentiment-analysis-web-0.0.2-SNAPSHOT.jar

DEPLOY THE JAVA APP (demo-backend1)



DEPLOY THE PYTHON APP ON the AWS VM "deploy-server-python"

1) Python version must be installed
apt-get update && apt-get install -y python3

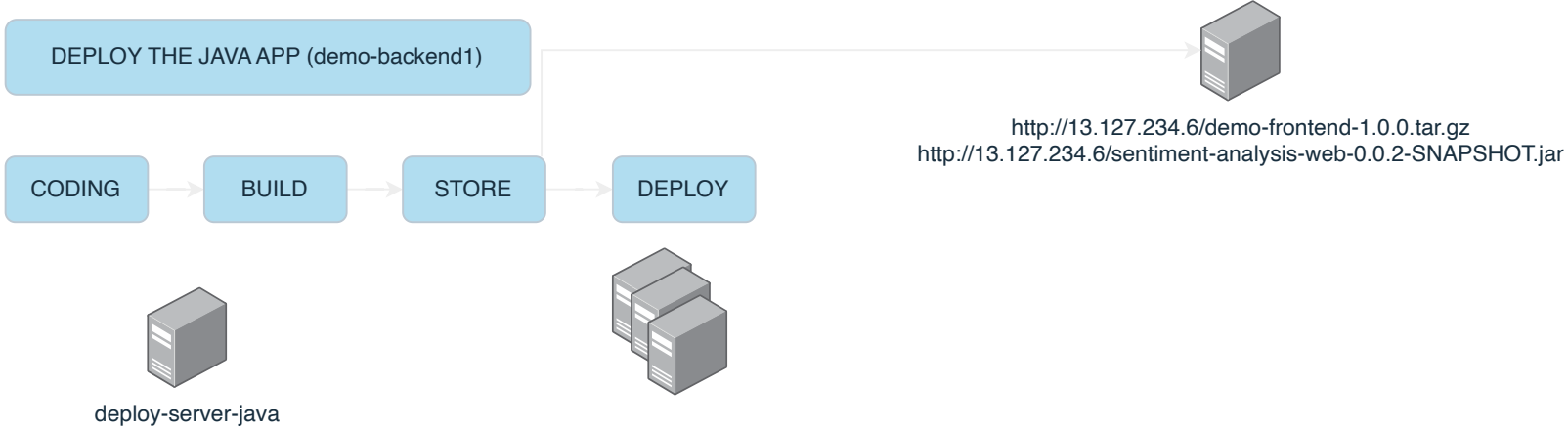
2) Clone the code from GitHub
git clone https://github.com/TekspotEdu/microserviceapp.git

3) Make sure that all App dependencies are installed
#Install PIP
apt-get update && apt-get install -y python3-pip

#Install App Dependencies
cd microserviceapp/demo-backend2/sa
pip3 install -r requirements.txt

4) Run the Python application
python3 sentiment_analysis.py

http://172.31.34.90:5000



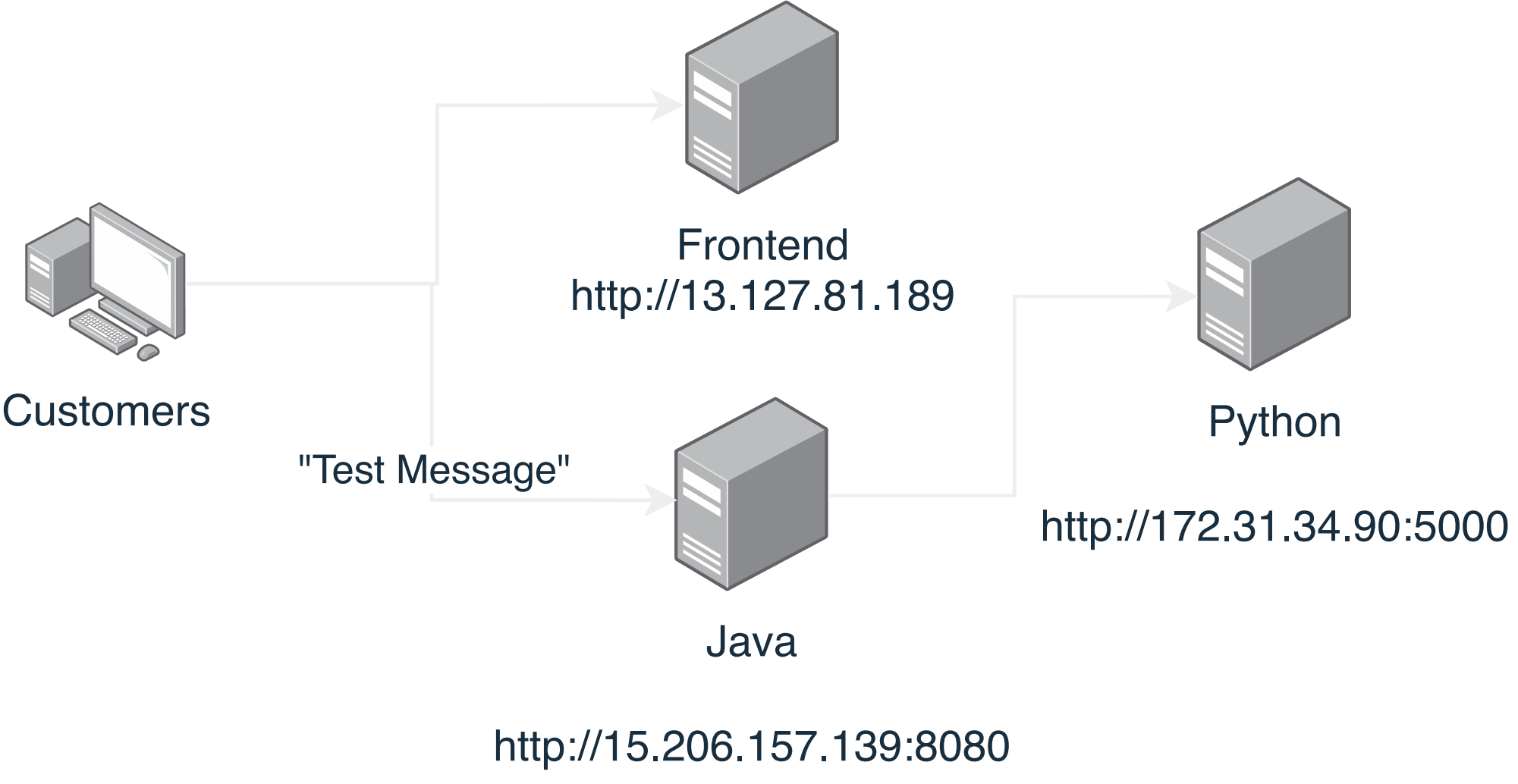
DEPLOY THE JAVA APP ON the AWS VM "deploy-server-java"

1) Install JDK 11
apt-get update && apt-get install -y openjdk-11-jdk

2) Download the JAR file
wget "http://13.127.234.6/sentiment-analysis-web-0.0.2-SNAPSHOT.jar"

3) Run the JAVA APPLICATION (jar file)
java -jar sentiment-analysis-web-0.0.2-SNAPSHOT.jar --sa.logic.api.url http://172.31.34.90:5000

http://15.206.157.139:8080


Frontend
http://13.127.81.189
Customers
Python
"Test Message"
http://172.31.34.90:5000
Java
http://15.206.157.139:8080

Updating the frontend with the correct JAVA URL
(http://15.206.157.139:8080)

CODE → BUILD → STORE → DEPLOY

BUILD SERVER

STORAGE SERVER

ON THE BUILD SERVER

1) Change the code
nano microserviceapp/demo-frontend/src/App.js

2) Build the code into a build
cd microserviceapp/demo-frontend
npm run build

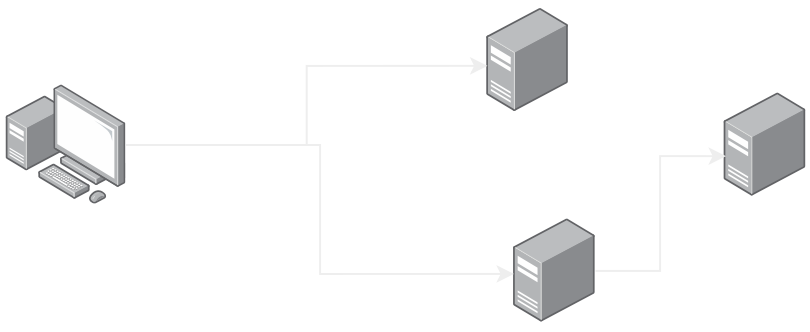
3) Store and Distribute
tar -czvf demo-frontend-1.0.1.tar.gz build
cp demo-frontend-1.0.1.tar.gz /var/www/html
(downloadable url http://13.127.234.6/demo-frontend-1.0.1.tar.gz)

BUILD SERVER

ON THE DEPLOY SERVER

1) Download and extract the file to /var/www/html
wget "http://13.127.234.6/demo-frontend-1.0.1.tar.gz"
rm -rf build
tar -xzvf demo-frontend-1.0.1.tar.gz
cp -prf build/* /var/www/html/

DEPLOY SERVER



Developing a Container Image in Steps

Nginx → Nginx build

Dockerfile

Content:

```
FROM nginx
COPY build /usr/share/nginx/html
```

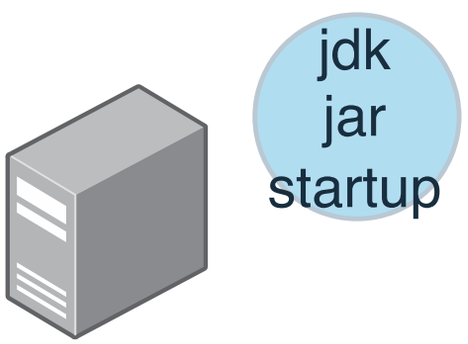
Docker Installed
Dockerfile
build

COMMAND:

```
cd microserviceapp/demo-frontend
docker build -t basilvarghese/demo-frontend:latest  .  -f Dockerfile
```

Develop the Java Application to a container

```
FROM openjdk:8-jdk-alpine

ENV SA_LOGIC_API_URL http://localhost:5000

EXPOSE 8080

WORKDIR /usr/app

CMD ["java", "-jar", "/usr/app/sentiment-analysis-web-0.0.2-SNAPSHOT.jar",
"--sa.logic.api.url=${SA_LOGIC_API_URL}"]

COPY target/sentiment-analysis-web-0.0.2-SNAPSHOT.jar /usr/app
```

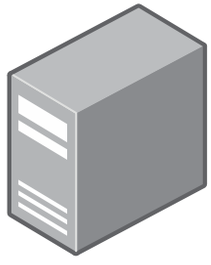


BUILD SERVER
1) Docker Installed
2) Jar file

```
docker build -t basilvarghese/demo-backend1:latest . -f Dockerfile
```

Develop the Python Application to a container

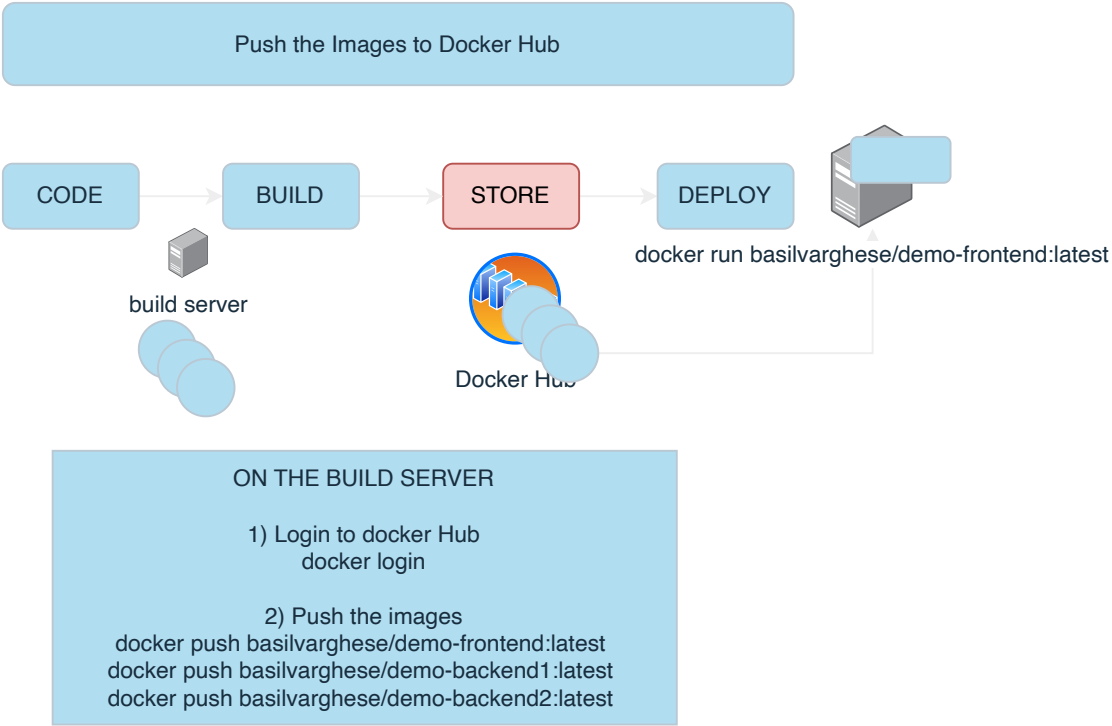
```
FROM python:3.6-slim

WORKDIR /app

EXPOSE 5000

CMD ["python3", "sentiment_analysis.py"]

COPY sa /app

RUN pip3 install -r requirements.txt && \
  python3 -m textblob.download_corpora
```



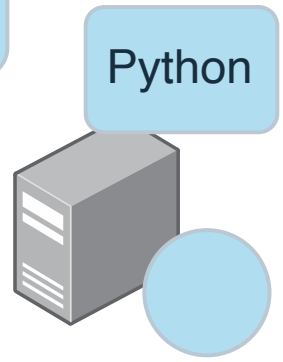
python
app dep
program
startup

BUILD SERVER
1) Docker Installed
2) requirements.txt and sentiment_analysis.py

cd demo-backend2
docker build -t basilvarghese/demo-backend2:latest . -f Dockerfile


Push the Images to Docker Hub
CODE
BUILD
STORE
DEPLOY
build server
Docker Hub
docker run basilvarghese/demo-frontend:latest
ON THE BUILD SERVER
1) Login to docker Hub
docker login
2) Push the images
docker push basilvarghese/demo-frontend:latest
docker push basilvarghese/demo-backend1:latest
docker push basilvarghese/demo-backend2:latest

DEPLOYING THE PYTHON APPLICATIONS



CODE → BUILD → STORE (Docker Hub) → DEPLOY

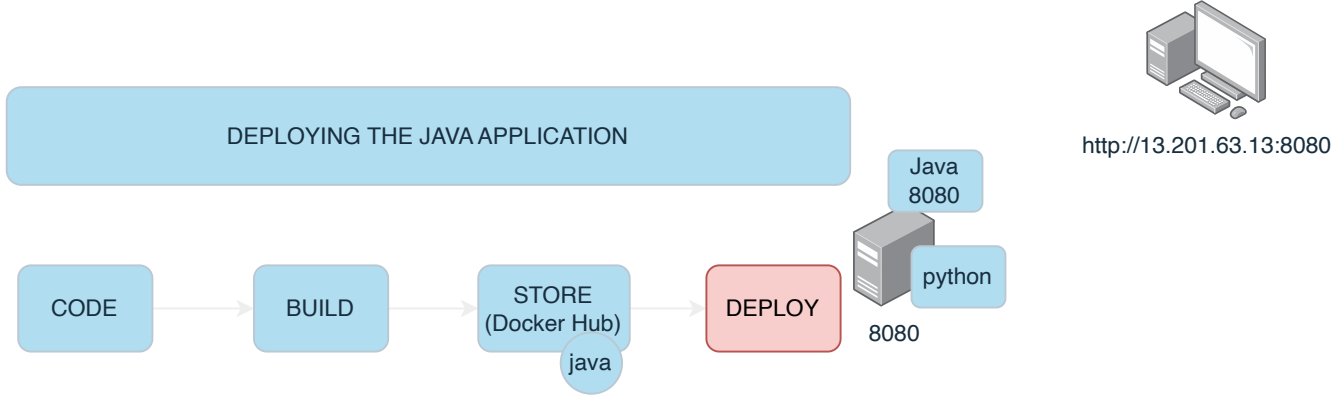
ON DEPLOY SERVER (Docker Host)
PreSet:
1) Docker Must be installed
2) Login to the registry (docker login)

Take care of
1) Run your Containers (docker run)

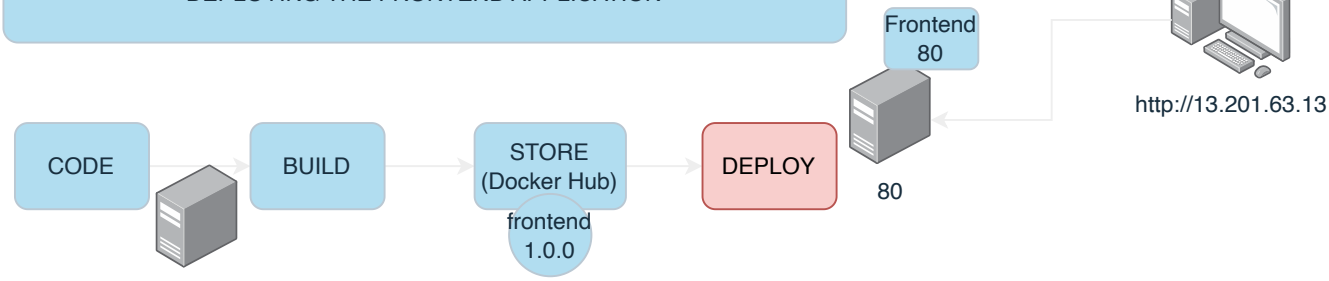
2) Network settings (172.17.0.2)

3) Configure the containers

Command:
docker run -d basilvarghese/demo-backend2:latest
docker inspect <Container ID> # Give the private IP


DEPLOYING THE JAVA APPLICATION
CODE
BUILD
STORE
(Docker Hub)
java
DEPLOY
Java
8080
python
8080
http://13.201.63.13:8080
ON DEPLOY SERVER (Docker Host)
PreSet:
1) Docker Must be installed
2) Login to the registry (docker login)
Take care of
1) Run your Containers (docker run)
2) Network settings (http://13.201.63.13:8080)
3) Configure the containers (SA_LOGIC_API_URL=http://172.17.0.2)
Command:
docker run -d -p 8080:8080 basilvarghese/demo-backend1:latest

DEPLOYING THE FRONTEND APPLICATION



ON DEPLOY SERVER (Docker Host)

PreSet:
1) Docker Must be installed
2) Login to the registry (docker login)

Take care of
1) Run your Containers (docker run)

2) Network settings (http://13.201.63.13)

3) Configure the containers (Require a Code Change, JAVA URL to be hardcoded in src/App.js 23rd line)

Command:
docker run -d -p 80:80 basilvarghese/demo-frontend:latest

FOUR STEP PROCESS TO CONFIGURE FRONTEND APP WITH JAVA URL

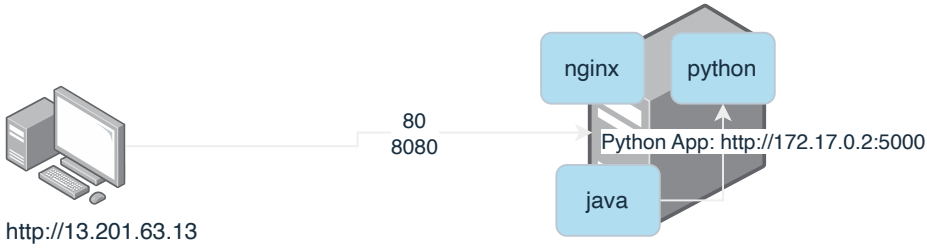
ON BUILD SERVER
1) Change the Code => 23rd line of src/App.js and put JAVA URL
http://13.201.63.13:8080

2) Build the container
cd demo-frontend
npm run build
docker build -t basilvarghese/demo-frontend:1.0.0 . -f Dockerfile

3) Publish the image to registry
docker login # already there
docker push basilvarghese/demo-frontend:1.0.0

ON DEPLOY SERVER
4) Deploy the application
docker run -d -p 80:80 basilvarghese/demo-frontend:1.0.0

DEPLOYING THE FRONTEND APPLICATION



FOUR STEP PROCESS TO CONFIGURE FRONTEND APP WITH JAVA URL

ON BUILD SERVER

1) Change the Code => 23rd line of src/App.js and put JAVA URL
http://13.201.63.13:8080

2) Build the container
cd demo-frontend
npm run build
docker build -t basilvarghese/demo-frontend:1.0.0 . -f Dockerfile

3) Publish the image to registry
docker login # already there
docker push basilvarghese/demo-frontend:1.0.0

ON DEPLOY SERVER

4) Deploy the application
docker run -d -p 80:80 basilvarghese/demo-frontend:1.0.0