OpenShift

STAGES OF A DEPLOYMENT

PUBUDU WELAGEDARA

All Stages

1. Build
2. Dockerize
3. Publish
4. Deploy

1. Build

Use Gradle to build the JAR file

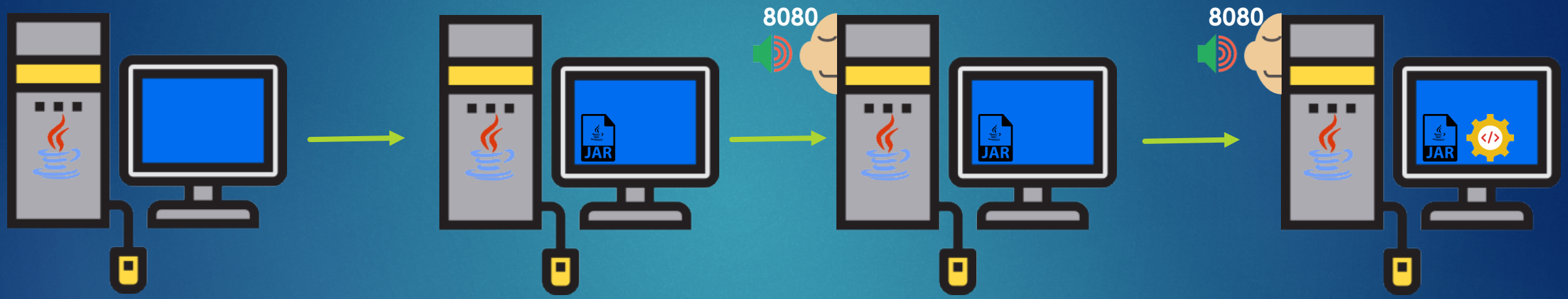
Spring Boot Application

$ ./gradlew clean build⏎

oc-0.0.1-SNAPSHOT.jar

2. Dockerize | Steps

Think about how you deploy your JAR file on a Linux Box



Your Linux PC has Java

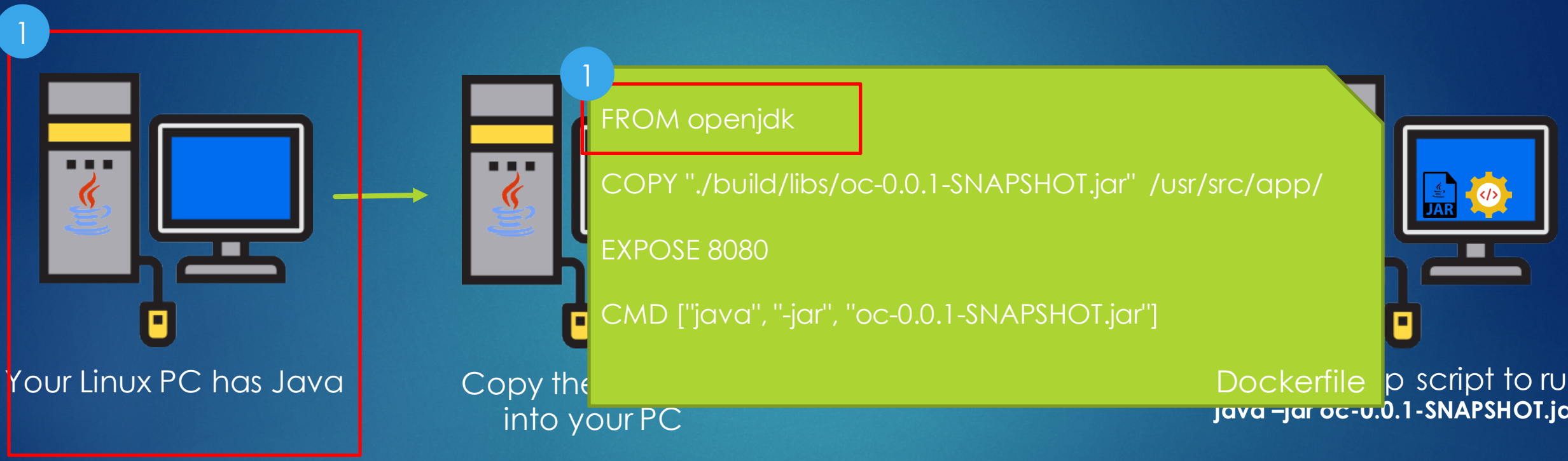
Copy the JAR file into your PC

Open port **8080**

Add a startup script to run **java –jar oc-0.0.1-SNAPSHOT.jar**

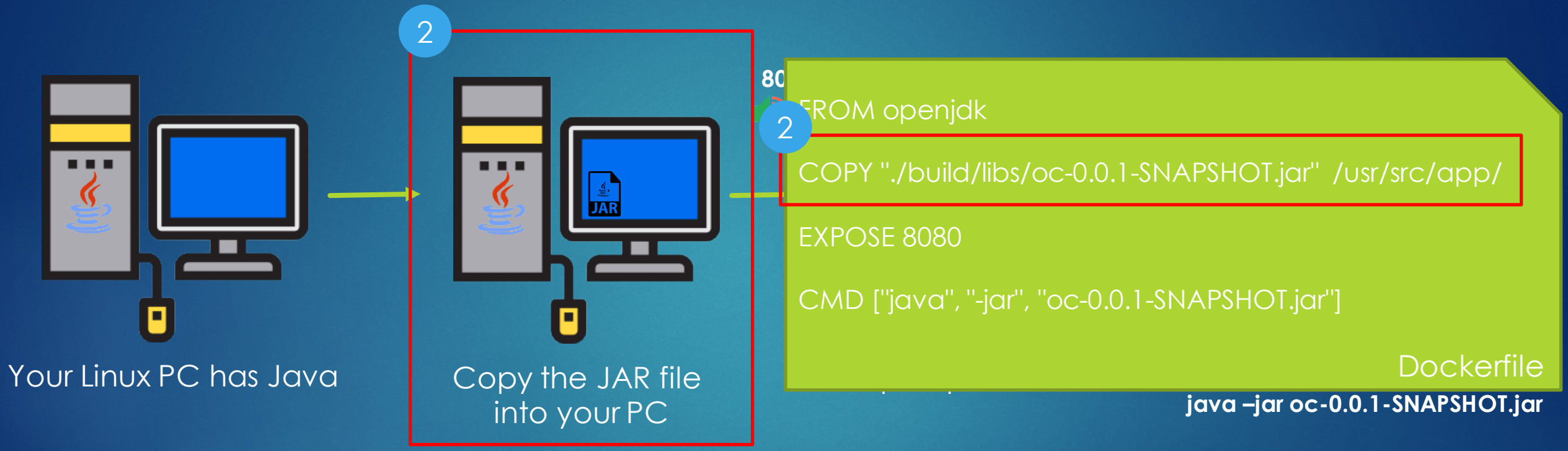
2. Dockerize Cont… | Dockerfile

Dockerfile has a set of instructions to build the Docker Image



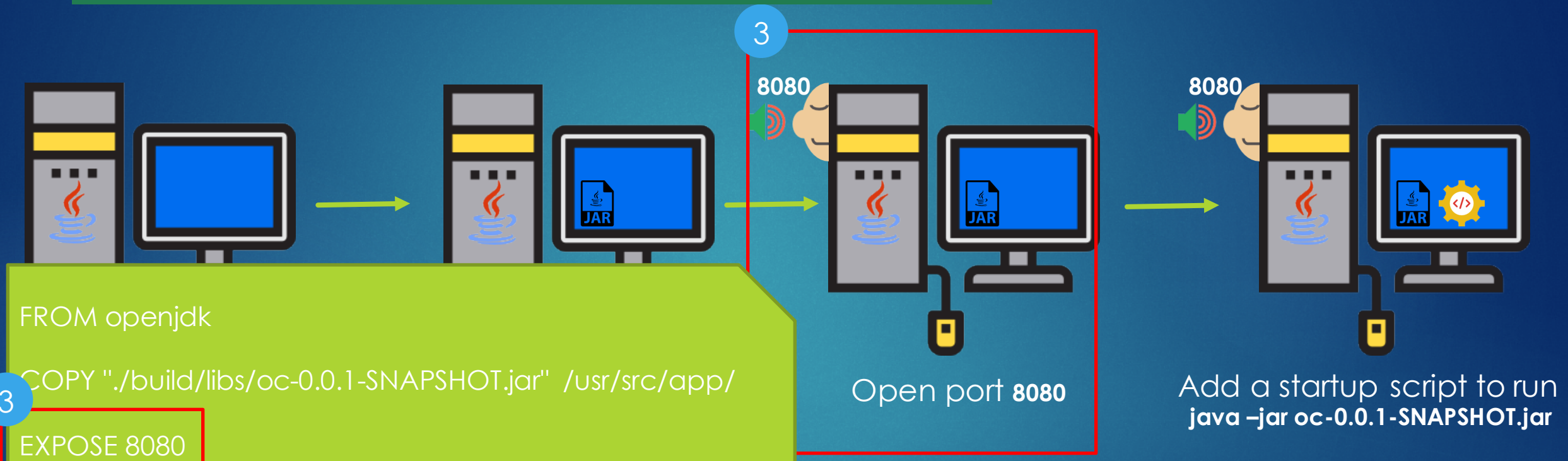
2. Dockerize Cont... | Dockerfile

Dockerfile has a set of instructions to build the Docker Image



2. Dockerize Cont... | Dockerfile

Dockerfile has a set of instructions to build the Docker Image



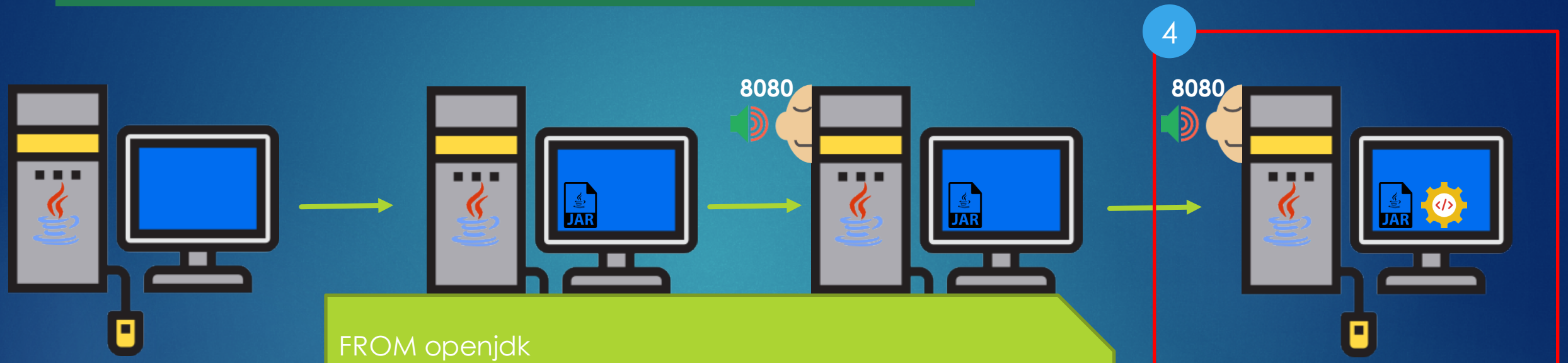

```
FROM openjdk

COPY "./build/libs/oc-0.0.1-SNAPSHOT.jar"  /usr/src/app/

EXPOSE 8080

CMD ["java", "-jar", "oc-0.0.1-SNAPSHOT.jar"]
```

3

Dockerfile

2. Dockerize Cont… | Dockerfile

Dockerfile has a set of instructions to build the Docker Image



Your Linux PC has Java

```
FROM openjdk

COPY "./build/libs/oc-0.0.1-SNAPSHOT.jar"  /usr/src/app/

EXPOSE 8080

CMD ["java", "-jar", "oc-0.0.1-SNAPSHOT.jar"]
```

4

Dockerfile

4

Add a startup script to run
java –jar oc-0.0.1-SNAPSHOT.jar

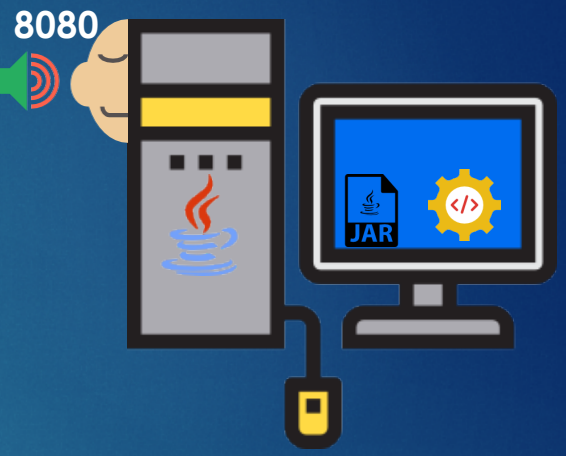
2. Dockerize Cont… | Docker

Docker build command creates the Docker Image.
Docker Image has a name and a tag(version).

```
FROM openjdk

COPY "./build/libs/oc-0.0.1-SNAPSHOT.jar"  /usr/src/app/

EXPOSE 8080

CMD ["java", "-jar", "oc-0.0.1-SNAPSHOT.jar"]
```

Dockerfile

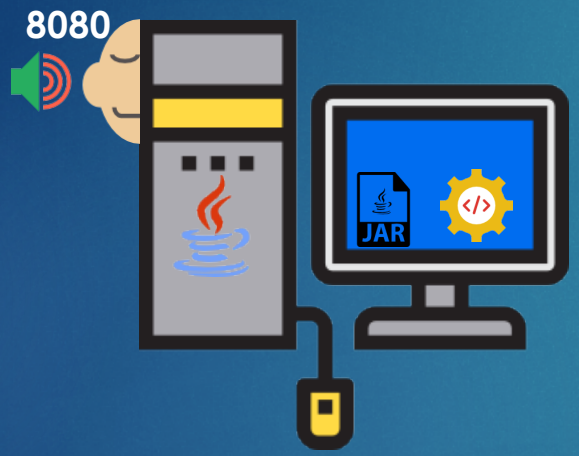
$ docker build⏎



Docker Image
oc:1.0.3

3. Publish

Docker push command pushes the Docker Image to an Artifact Repository



Docker Image
oc:1.0.3

$ docker push⏎

…

…

…

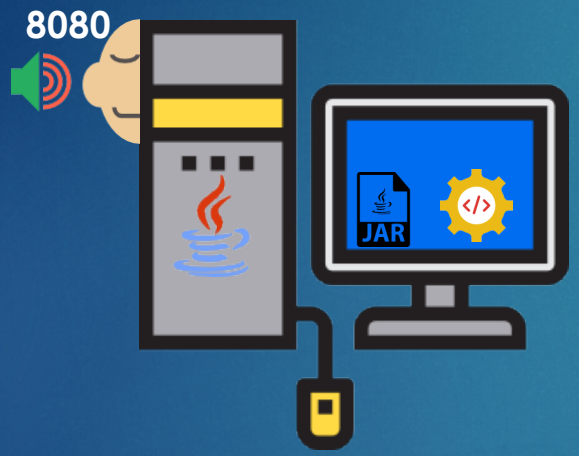
…

JFrog | docker | …

Artifact Repository

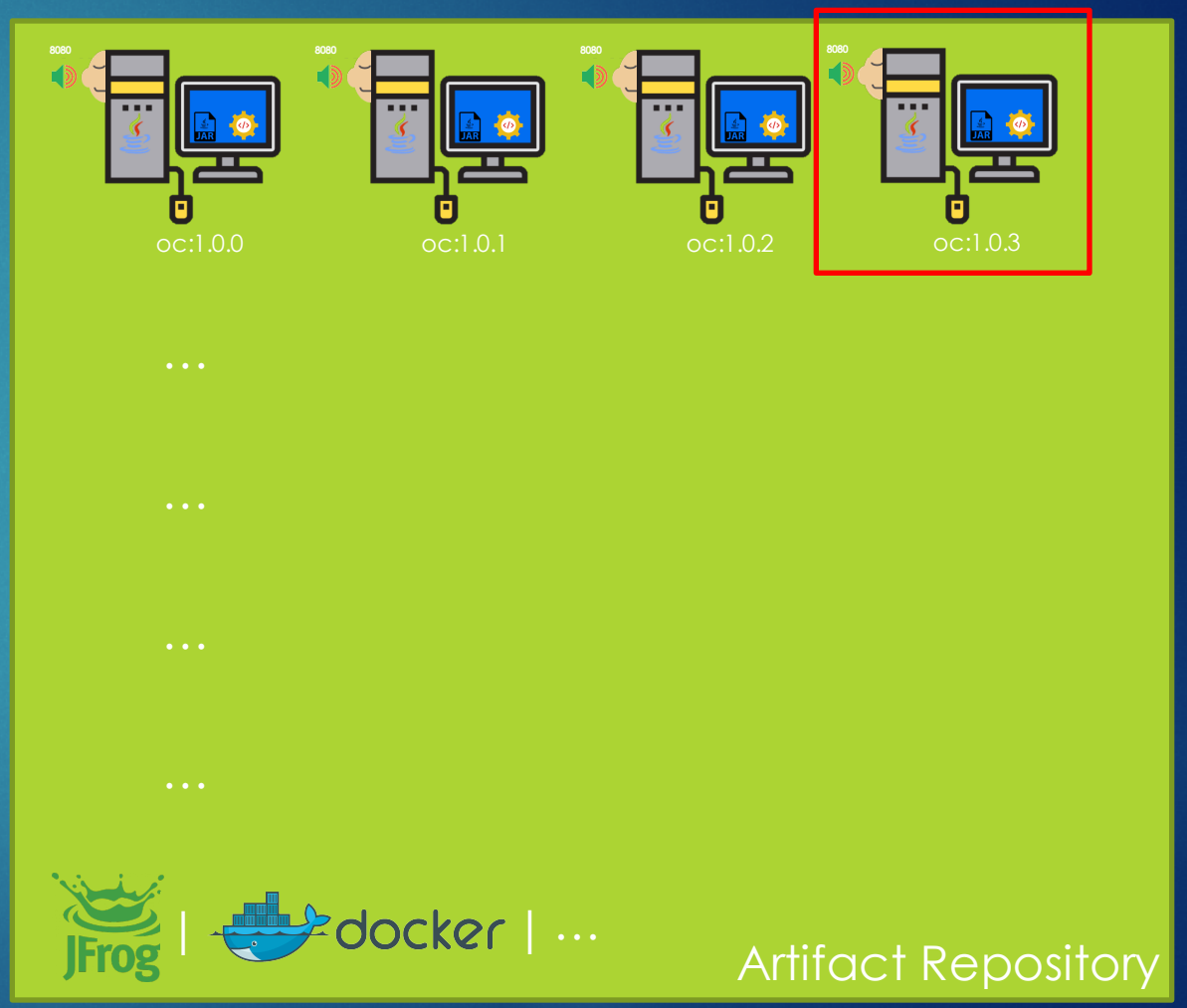
3. Publish Cont…

Docker push command pushes the Docker Image to an Artifact Repository



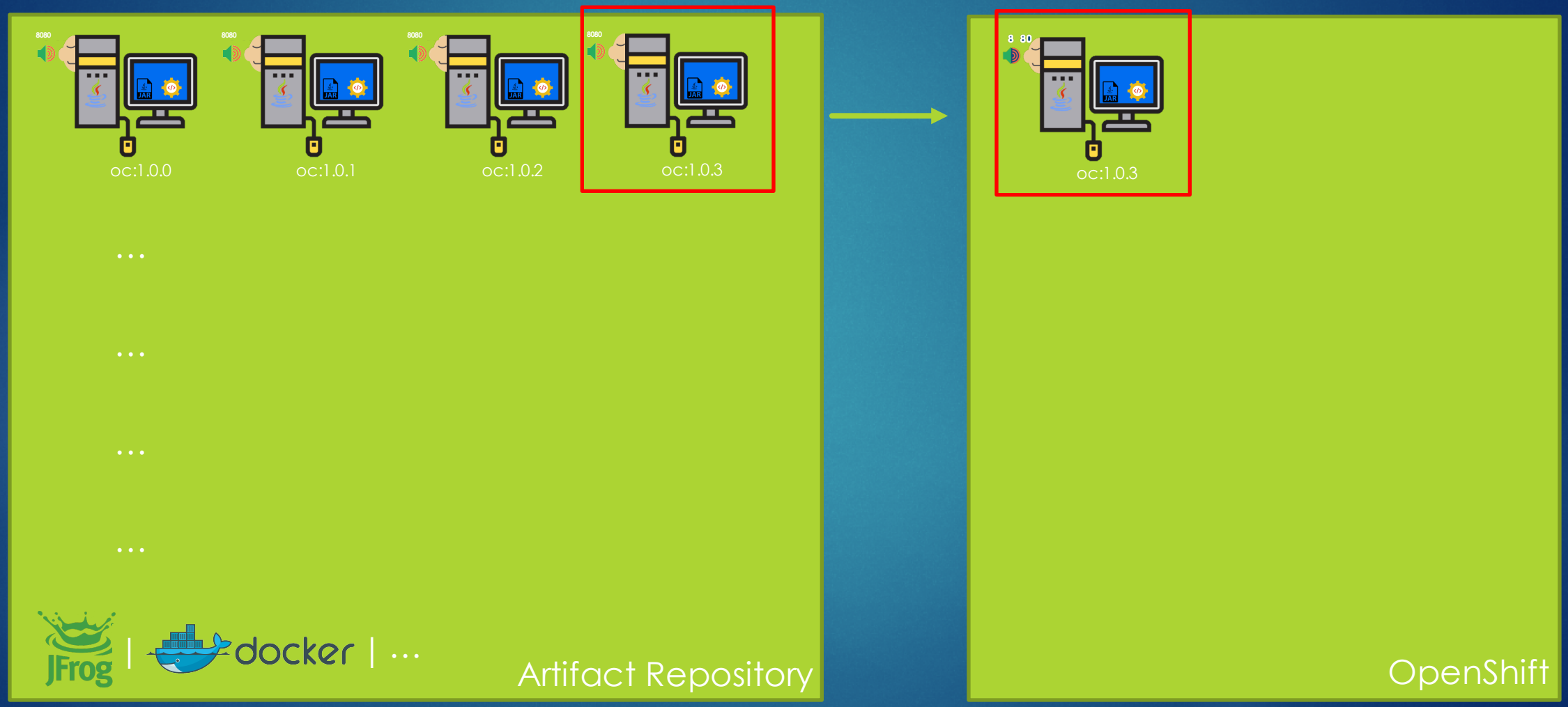
Docker Image
oc:1.0.3

$ docker push⏎



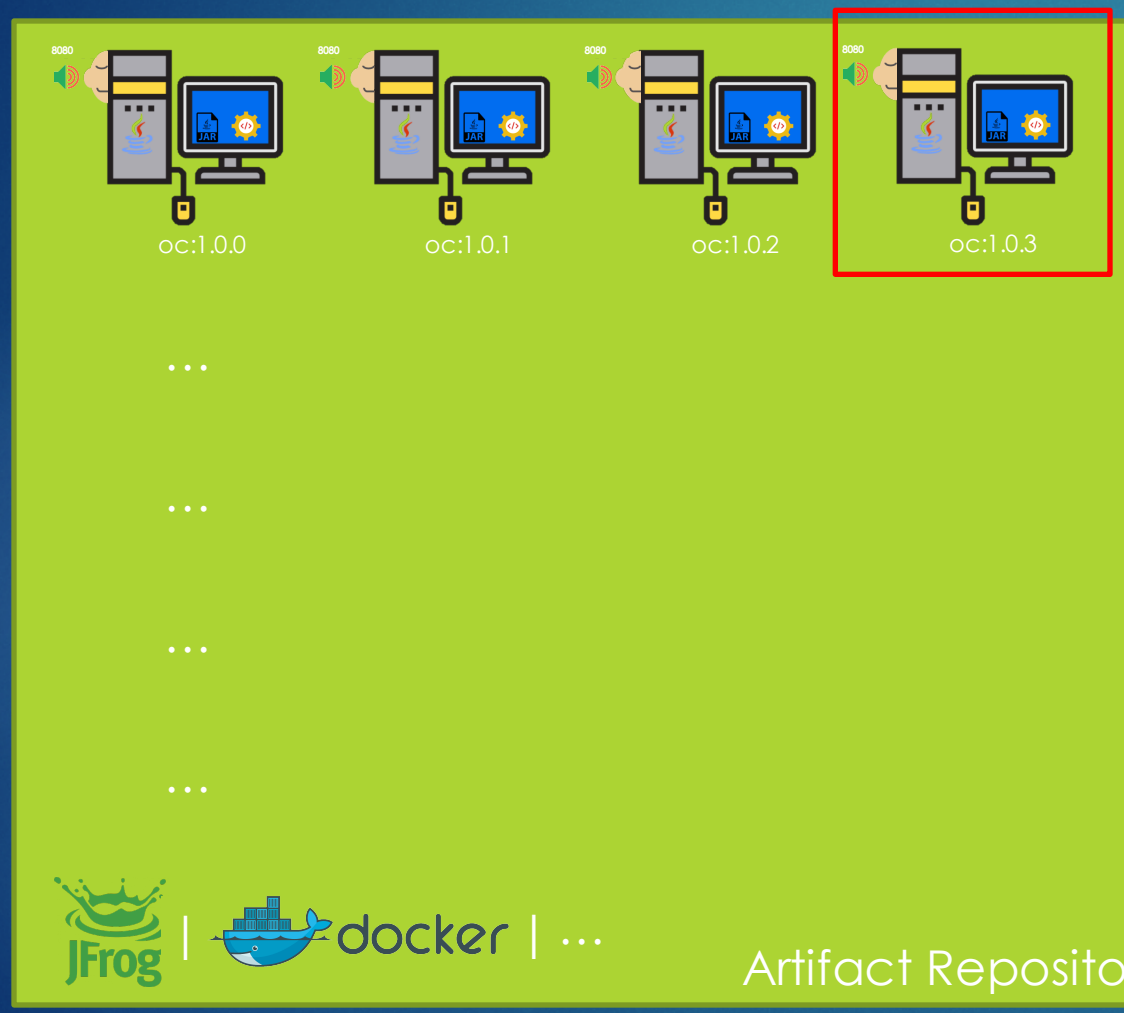
4. Deploy | Step 1

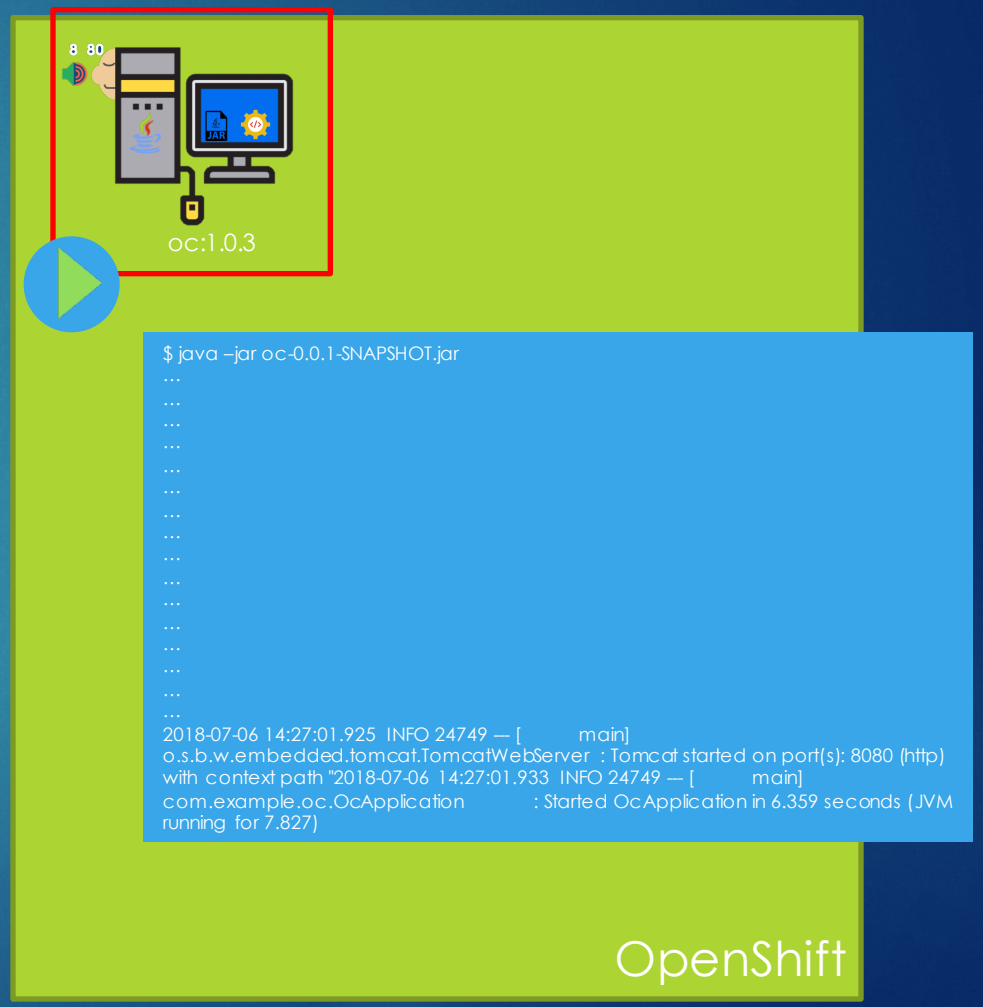
Pull the Docker Image from the Artifact Repository to OpenShift



4. Deploy | Step 2

Run the Docker Image(Running will execute the startup script)

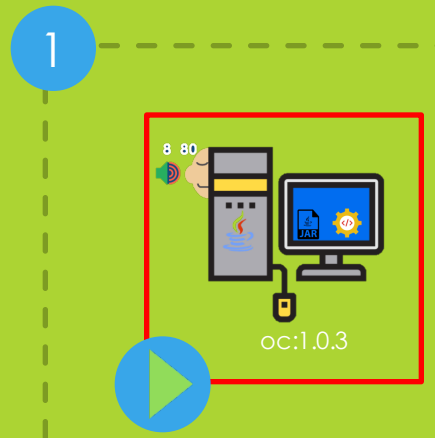




4. Deploy | Deployment

Defines the desired state of the app(what Image to use, how many instances should run etc.).

$ oc apply –f deployment.yaml⏎




```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: oc
spec:
  selector:
    matchLabels:
      app: oc
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: oc
    spec:
      containers:
      - image: pubuduwelagedara/oc:1.0.0
        name: oc
        imagePullPolicy: Always
        env:
        - name: SPRING_PROFILES_ACTIVE
          value: deployment
        ports:
        - containerPort: 8080
          name: oc
```

deployment.yaml

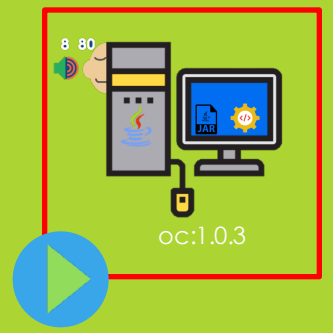
OpenShift

4. Deploy | Service

Makes the Deployment accessible within the cluster.

$ oc apply –f service.yaml⏎

2




```
apiVersion: v1
kind: Service
metadata:
  name: oc-service
spec:
  ports:
  - port: 8080
    name: oc
    targetPort: oc
  selector:
    app: oc
```

service.yaml

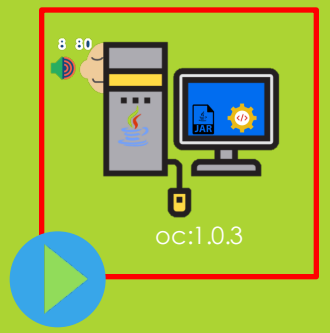
oc-service

OpenShift

4. Deploy | Ingress*

Makes the Service accessible outside the cluster(L7 Load Balancing) from internet.

3



oc.yourdomain.com

OpenShift

*An Ingress is usually created manually without a yaml file.

“Thank you”

Icons made by Freepik from www.flaticon.com