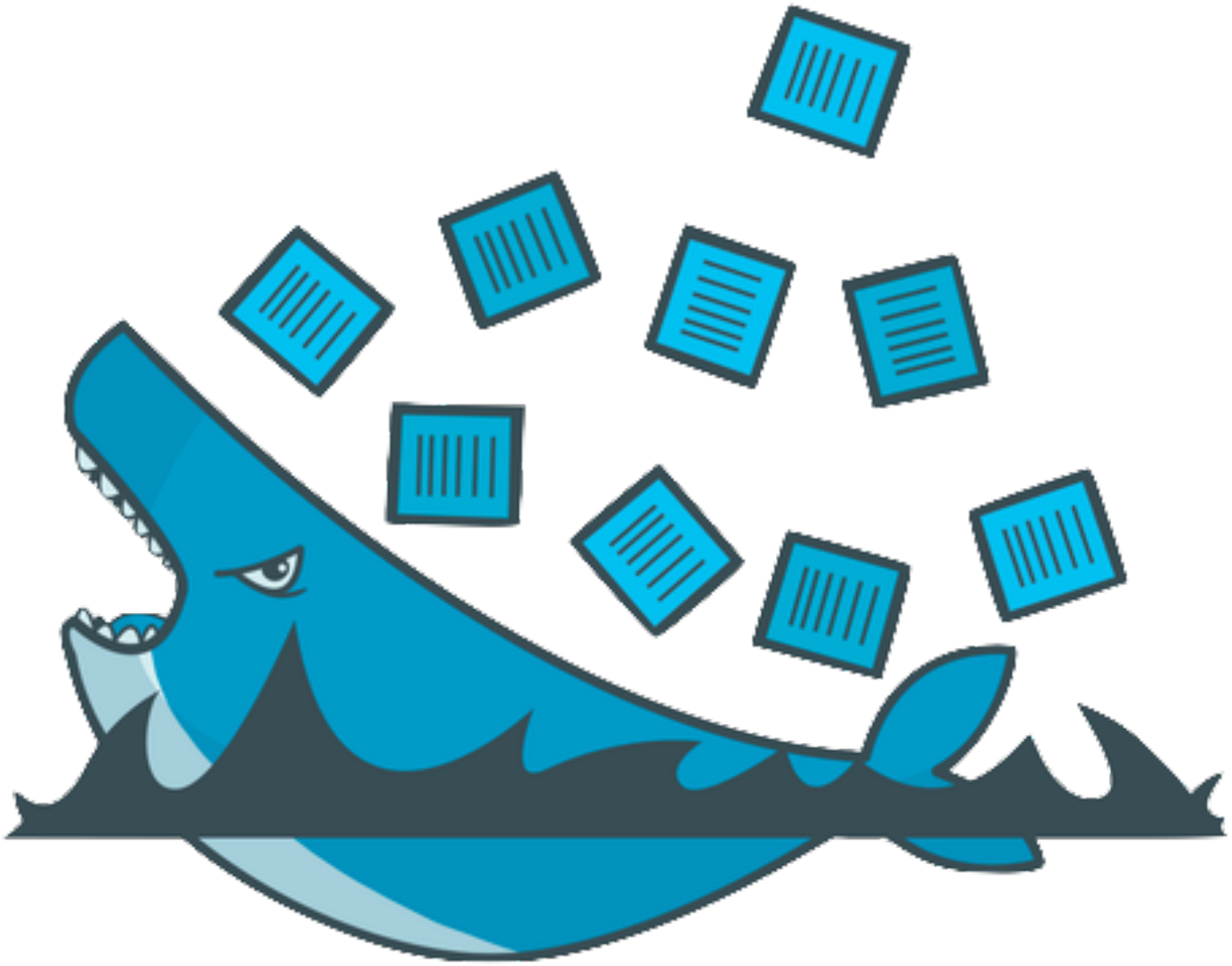
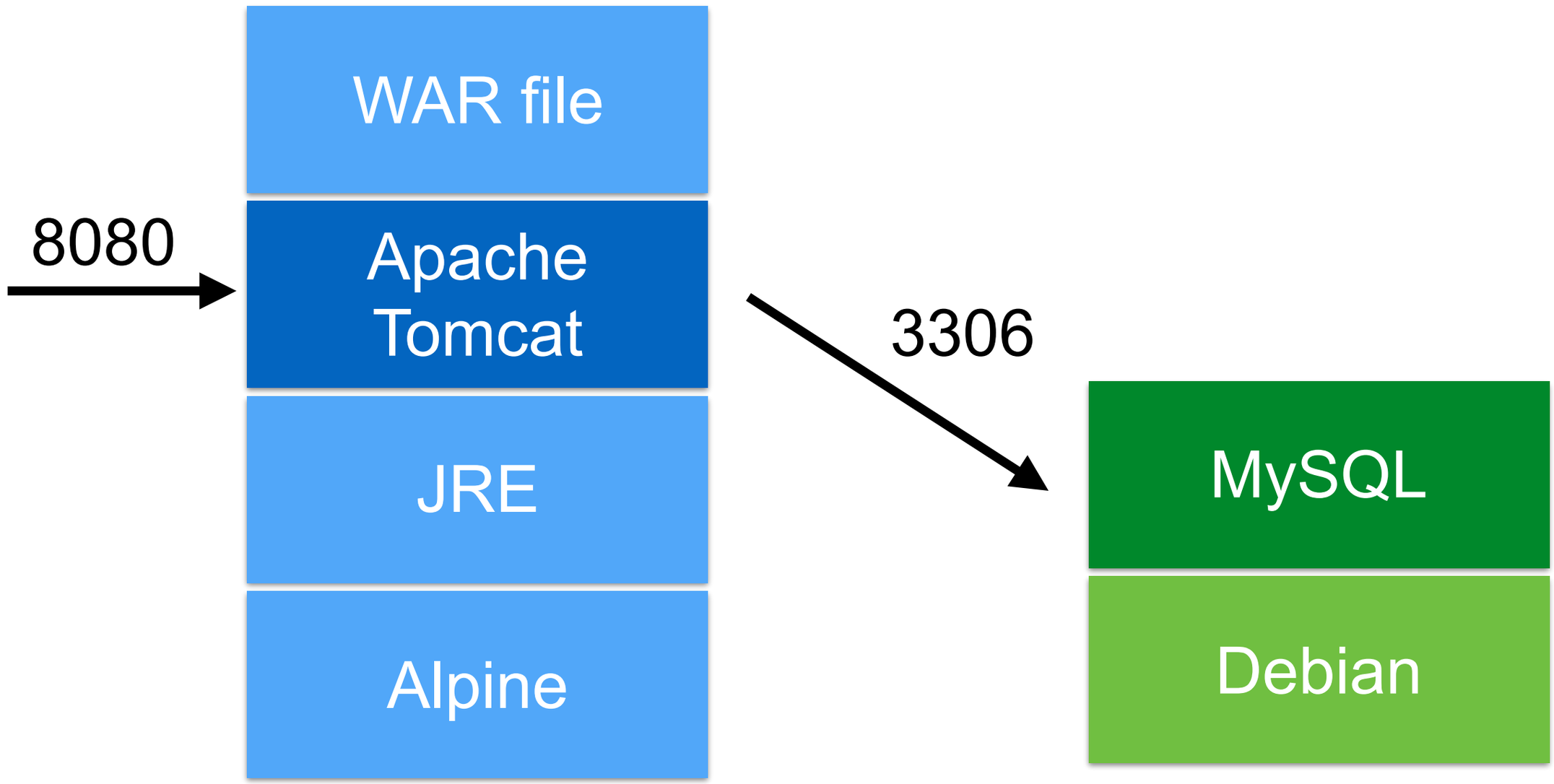


Docker workshop

Topics

Dockerfile
Multi-stage of Dockerfile
Working with environment variable
Working with volume
Working with Docker-compose
Working with Docker-swarm

Architecture



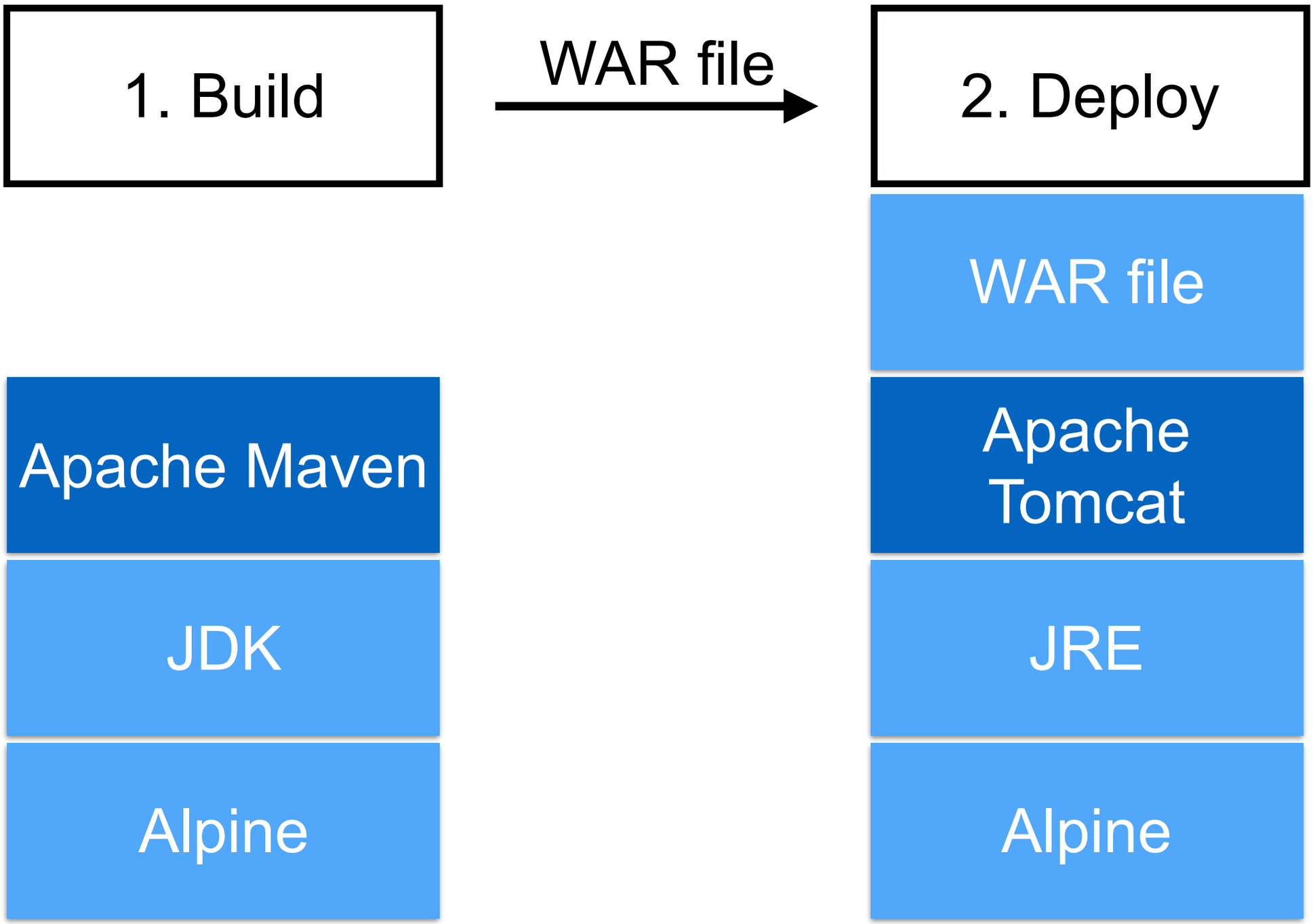
Source code

https://github.com/up1/docker-workshop-java-mysql

Web with Apache Tomcat

Step to deploy

Step to deploy



Build

Dockerfile_build

```
FROM maven:3.5.2-jdk-8-alpine
WORKDIR /src
COPY  . /src

ENTRYPOINT ["mvn"]
CMD ["clean", "package"]
```

Build image of Build process

```
$docker image build -t web_build:0.1
-f Dockerfile_build .
```

Create build container

```
$docker container run --rm
  -v "$HOME/.m2":/root/.m2
  -v $(pwd):/src
  web_build:0.1
```

Deploy

Dockerfile_deploy

```
FROM tomcat:9.0.1-jre8-alpine
COPY  ./target/api.war /usr/local/tomcat/
webapps/api.war
```

Build image of Deploy process

```
$docker image build -t web_deploy:0.1
-f Dockerfile_deploy .
```

Create deploy container

```
$docker container run -d
  -p 8080:8080
  web_deploy:0.1
```

Many step to build !!

Using multi-stage build

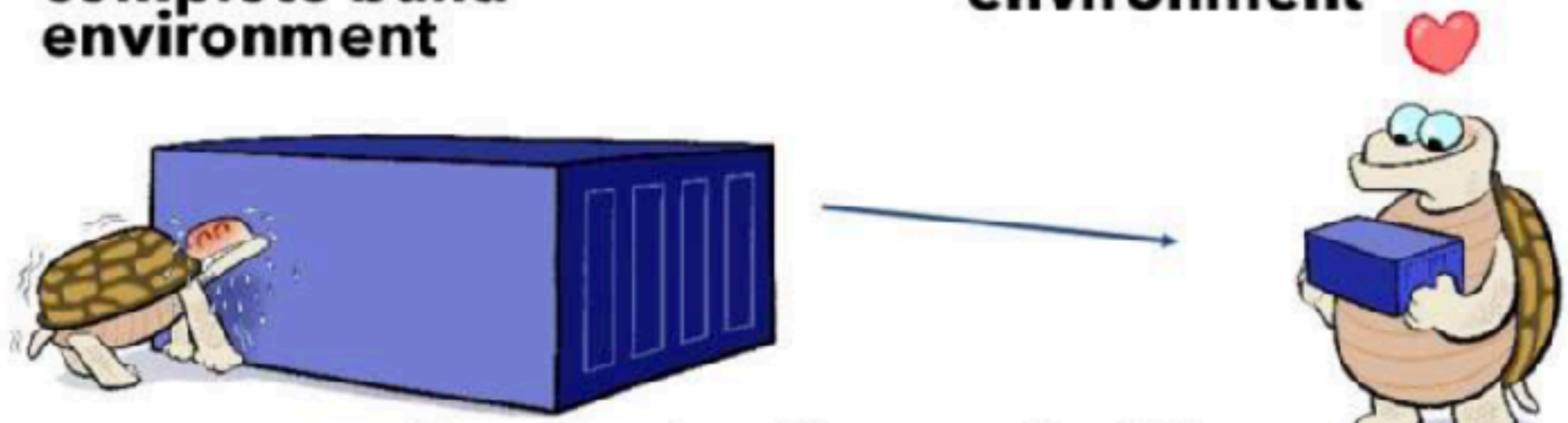
(Docker 17.05+)

https://docs.docker.com/engine/userguide/eng-image/multistage-build/

Build smaller images with multi-stage builds

First stage: complete build environment

Second stage: minimal runtime environment



One Dockerfile, one build

Dockerfile_api

```
FROM maven:3.5.2-jdk-8-alpine as builder
WORKDIR /src
COPY  . /src
RUN mvn clean package

FROM tomcat:9.0.1-jre8-alpine
COPY --from=builder /src/target/api.war /usr/
local/tomcat/webapps/
```

Build image with multi-stage

```
$docker image build -t web_api:0.1
-f Dockerfile_api .
```

Downloading .....

How to improve ?

Local maven server !!

JFrog Artifactory

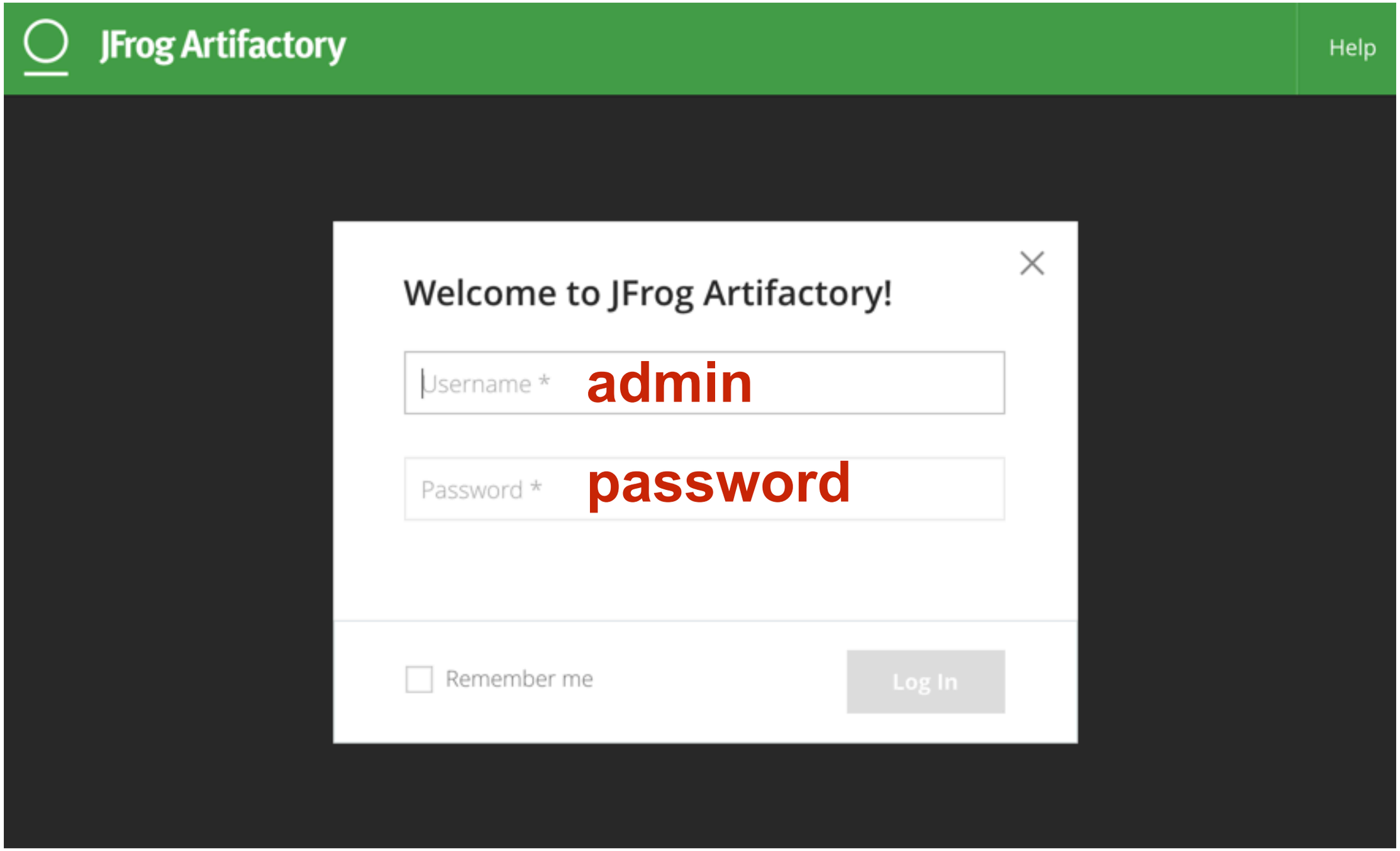
Sonatype Nexus

Frog Artifactory

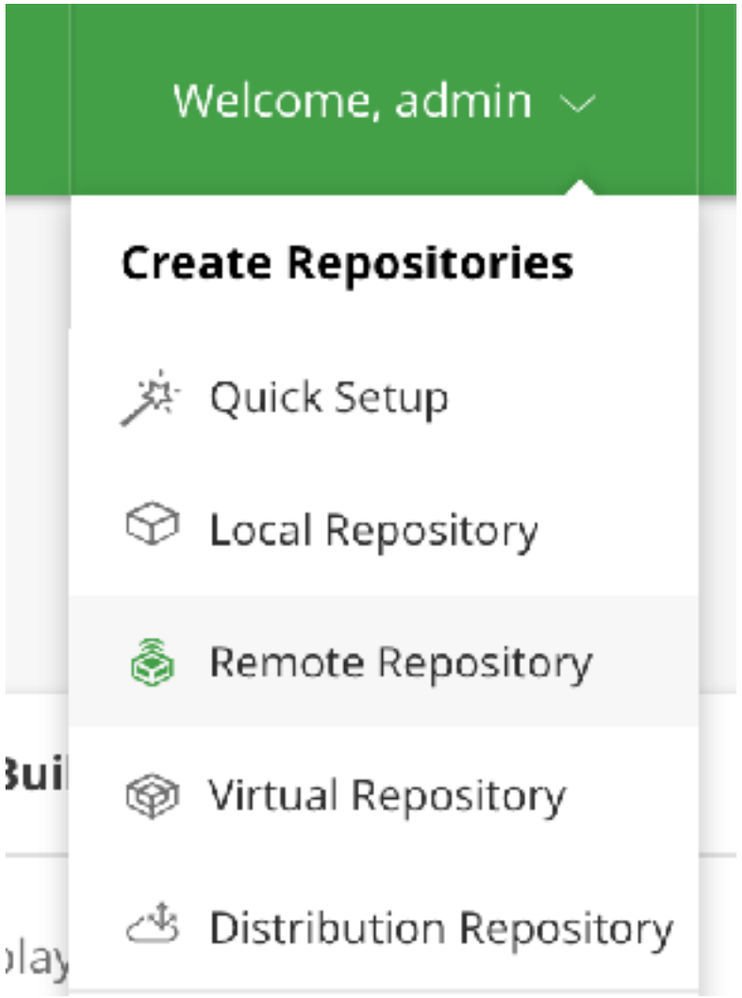
```
$docker run --name artifactory -d
  -p 8081:8081
  docker.bintray.io/jfrog/artifactory-oss:latest
```

https://www.jfrog.com/confluence/display/RTF/Installing+with+Docker

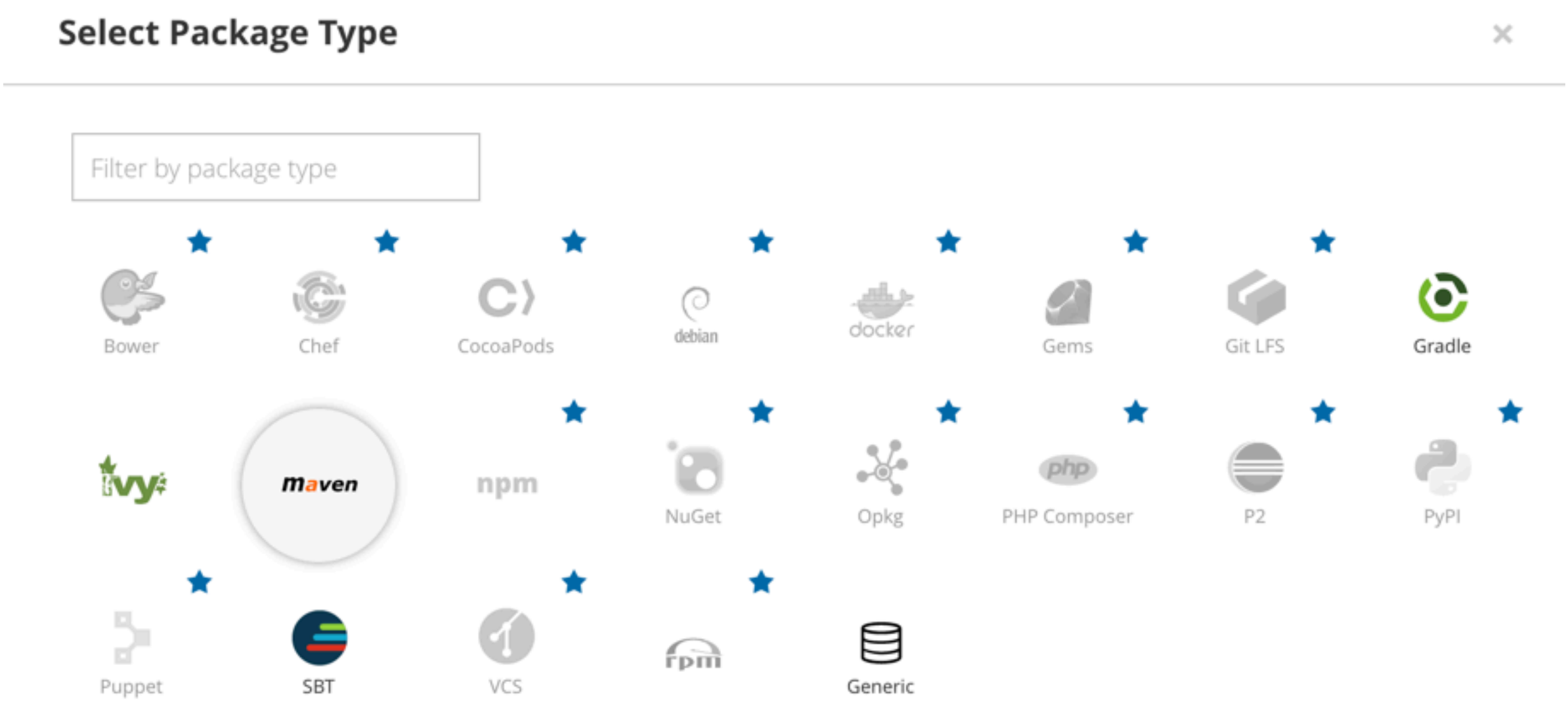
Config Frog Artifactory



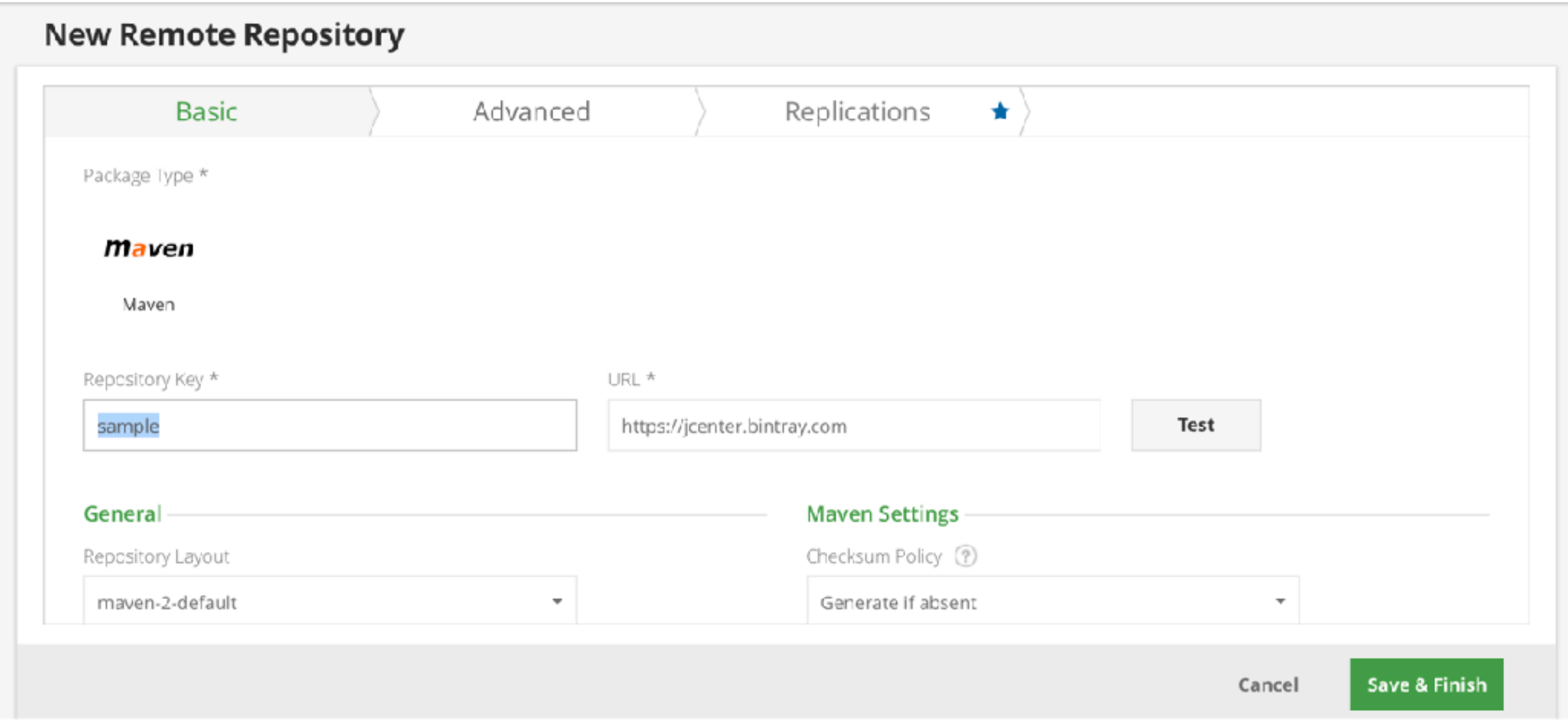
Create remote repository



Choose Maven



Create remote repository



Config maven

File /root/.m2/settings.xml

```
<mirrors>
    <mirror>
        <id>central</id>
        <name>central</name>
        <url>http://172.17.0.2:8081/artifactory/sample</url>
        <mirrorOf>*</mirrorOf>
    </mirror>
</mirrors>
```

Let’s try again !!

Dockerfile_api

```
FROM maven:3.5.2-jdk-8-alpine as builder
WORKDIR /src
COPY settings.xml /root/.m2/settings.xml
COPY  . /src
RUN mvn clean package

FROM tomcat:9.0.1-jre8-alpine
COPY --from=builder /src/target/api.war /usr/
local/tomcat/webapps/
```

Build image with multi-stage

```
$docker image build -t web_api:0.1
-f Dockerfile_api .
```

Create container

```
$docker container run -d
  -p 8080:8080
  web_api:0.1
```

Database with MySQL

Docker image of MySQL

OFFICIAL REPOSITORY

mysql ☆

Last pushed: 8 days ago

Repo Info Tags

Short Description

MySQL is a widely used, open-source relational database management system (RDBMS).

Docker Pull Command

```
docker pull mysql
```

Full Description

Supported tags and respective `Dockerfile` links

- `8.0.3`, `8.0`, `8` (*8.0/Dockerfile*)
- `5.7.20`, `5.7`, `5`, `latest` (*5.7/Dockerfile*)
- `5.6.38`, `5.6` (*5.6/Dockerfile*)
- `5.5.58`, `5.5` (*5.5/Dockerfile*)

https://hub.docker.com/_/mysql/

Create container

```
$docker container run \
 -d \
 --name=my_database \
 -e "MYSQL_ROOT_PASSWORD=mypassword" \
 -e "MYSQL_DATABASE=sample" \
 -e "MYSQL_USER=user01" \
 -e "MYSQL_PASSWORD=password" \
 mysql:5.7.20
```

Insert data into mysql

Create file **import.sql**

```
USE sample;

CREATE TABLE USER (
  id INT(11),
  name char(60)
) ENGINE=INNODB;

INSERT INTO USER VALUES(1, 'Sample name 01');
INSERT INTO USER VALUES(2, 'Sample name 02');
INSERT INTO USER VALUES(3, 'Sample name 03');
```

Create Dockerfile_data

```
FROM mysql:5.7.20
COPY import.sql /docker-entrypoint-initdb.d/
```

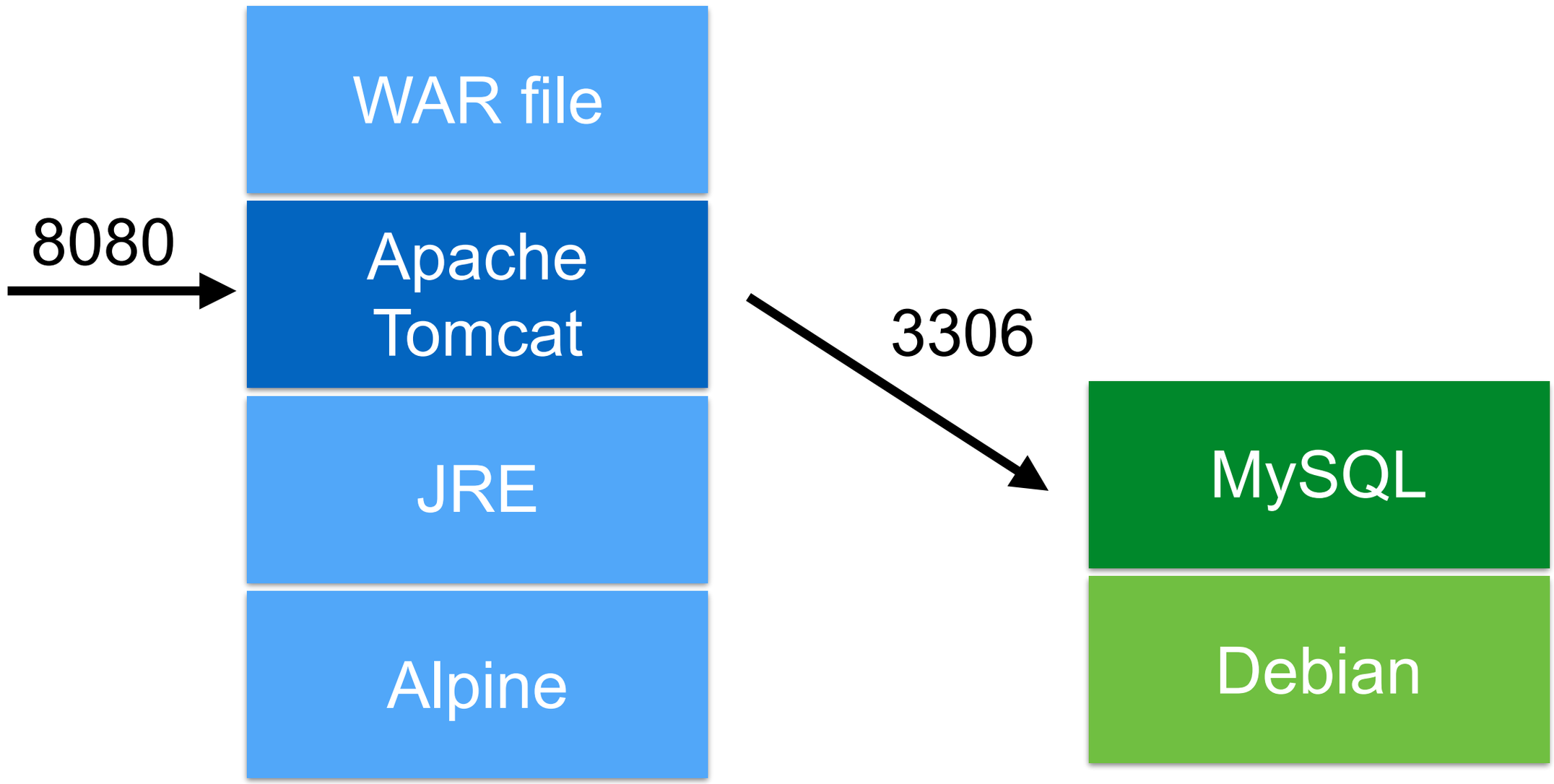
Build image of mysql with data

```
$docker image build -t mysql_data
  -f Dockerfile_data .
```

Create container again !!

```
$docker container run \
 -d \
 --name=my_database \
 -e "MYSQL_ROOT_PASSWORD=mypassword" \
 -e "MYSQL_DATABASE=sample" \
 -e "MYSQL_USER=user01" \
 -e "MYSQL_PASSWORD=password" \
 mysql_data
```

Architecture



Linking container

Create api container !!

```
$docker container run
  -d
  -p 8080:8080
  --link my_database
  web_api:0.1
```

Testing with database

http://localhost:8080/api/hello/db

Working with Environment variable

Using ENV in Dockerfile

```
FROM tomcat:9.0.1-jre8-alpine
ENV DATABASE_URL="jdbc:mysql://my_database/
sample?user=user01&password=password"
COPY --from=builder /src/target/api.war /usr/
local/tomcat/webapps/
```

Get ENV from Java

```
System.getenv("DATABASE_URL")
```

Working with Volume

Volume

-v $PWD:/data

HOST Container

Named Volume

-v **my_volume**:/data

Docker volume

Container

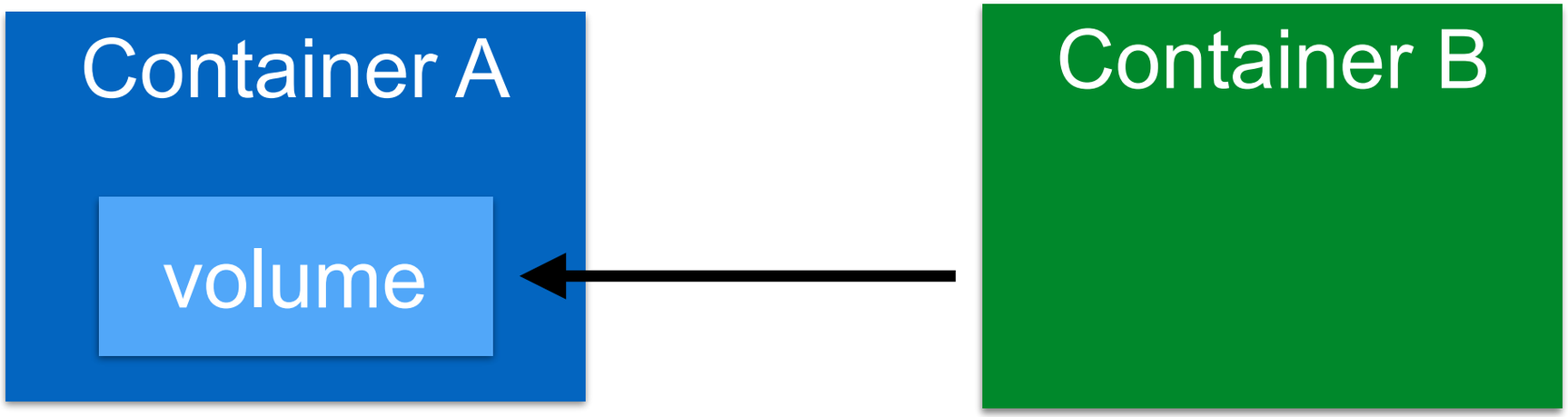
Working with volume

```
$docker volume ls
$docker volume create my_volume
$docker volume inspect my_volume
```

Remove volume

```
$docker volume prune
$docker volume rm my_volume
```

Volume from container



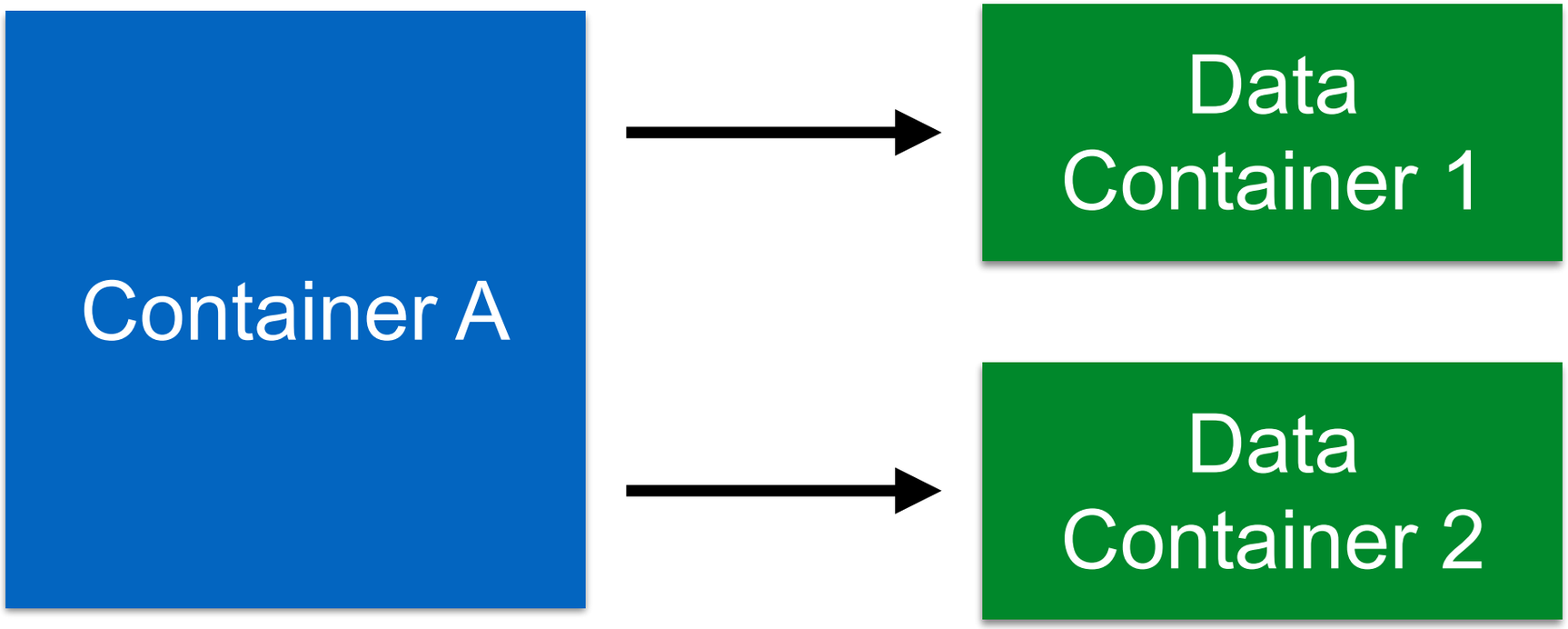
Create container A

$docker run … **–v /var/log** --name A

Create container B

$docker run … -v /var/log --name A
$docker run … **--volumes-from A**

Volume/Data container



Move data out from mysql container ?

Inspect Image

$docker inspect mysql:5.7.20

```
"Volumes": {
    "/var/lib/mysql": {}
},
"WorkingDir": "",
"Entrypoint": [
    "docker-entrypoint.sh"
],
```

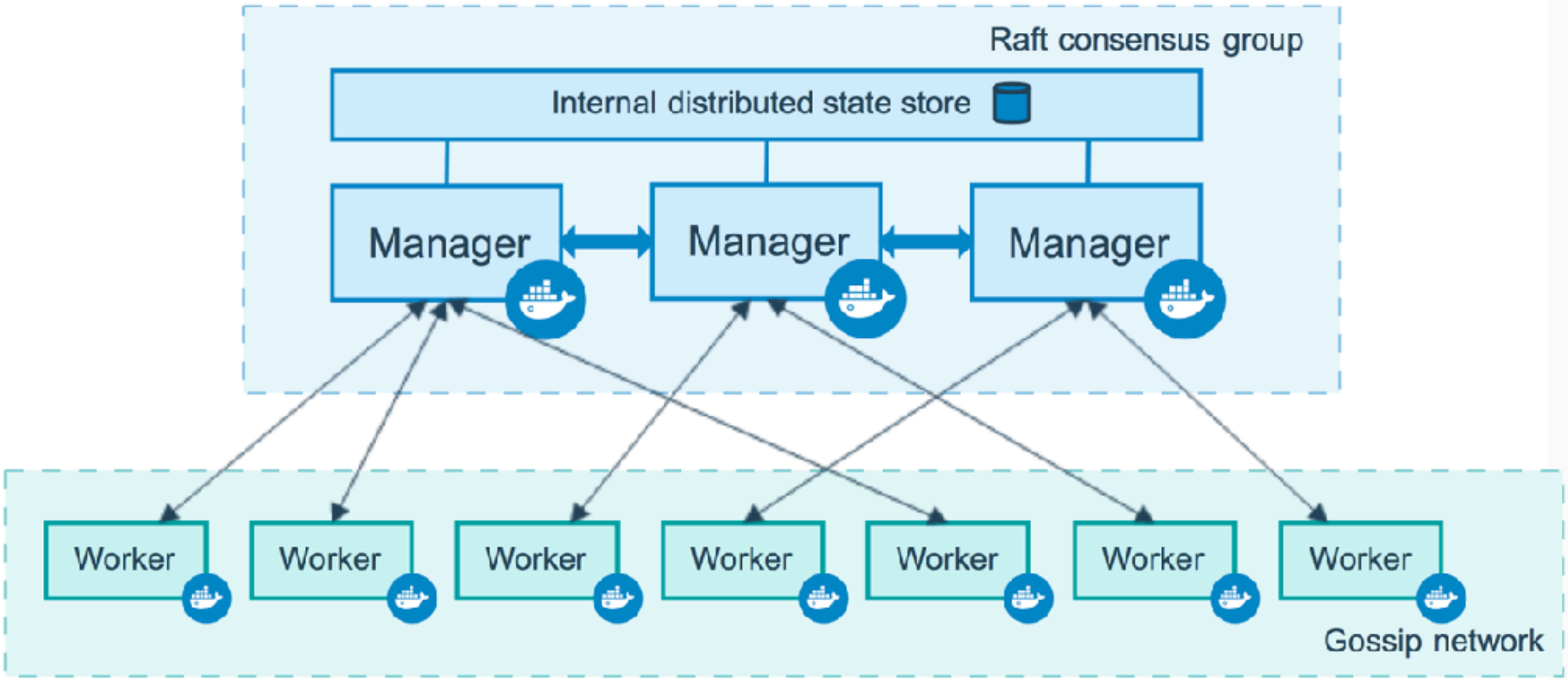
Working with Docker-compose

Running

```
$docker-compose up -d
$docker-compose ps
$docker-compose down
$docker-compose top
```

Working with Docker swarm

Swarm



https://docs.docker.com/engine/swarm/

2 Types of Node

Manager node
Worker node

Step to Swarm

Docker-compose v3
Initial swarm cluster
Add manager node
Add worker node
Deploy and Scaling

Initial Swarm

```
$docker swarm init
$docker swarm join-token manager
```

Deploy stack to Swarm

```
$docker stack deploy
  --compose-file=docker-compose-v2.yml
  my_demo
```

Scaling service

```
$docker service ls
$docker service scale my_demo_web=2
```

Goodbye Swarm

```
$docker stack rm my_demo
$docker swarm leave
```

Thank you & Question