# Docker 在摩拜的实践

谢恩龙 2018/10

# 目录

# 构建

# base image

https://mobike.io/common/alpine-oraclejdk8

其实就是在 oraclejdk8 镜像上加上了 base、curl 和「听云」。

# Dockerfile

问题：我们知道每个镜像构建都需要一个 Dockerfile 文件，为什么我们的应用源码中没有这个文件?

# docker-maven-plugin

将 Docker 的构建流程和 Maven 的生命周期绑定

- `mvn package` -> `docker build`
- `mvn deploy` -> `docker push`

# docker-maven-plugin

https://mobike.io/common/parent

```
<plugin>
    <groupId>com.spotify</groupId>
    <artifactId>docker-maven-plugin</artifactId>
    <version>1.0.0</version>
    <configuration>
        <imageName>${repo.docker}/${app.imagePath}</imageName>
        <baseImage>${repo.docker}/infra/java:${java.version}</baseImage>
        <workdir>/app</workdir>
        <env>
            <TZ>Asia/Shanghai</TZ>
            <SPRING_APPLICATION_NAME>${project.artifactId}</SPRING_APPLICATION_NAME>
        </env>
        <entryPoint>["boot.sh", "${project.build.finalName}.${project.packaging}"]</entryPoint>
        <resources>
            <resource>
                <targetPath>/app</targetPath>
                <directory>${project.build.directory}</directory>
                <include>${project.build.finalName}.${project.packaging}</include>
            </resource>
        </resources>
        <runs>
            <run>ln -snf /usr/share/zoneinfo/$TZ /etc/localtime</run>
            <run>echo $TZ &gt; /etc/timezone</run>
        </runs>
        <forceTags>true</forceTags>
        <imageTags>
            <imageTag>${project.version}</imageTag>
        </imageTags>
    </configuration>
</plugin>
```

# docker-maven-plugin

转换成的 Dockerfile

```
FROM docker.mobike.io/common/alpine-oraclejdk8:1.2

ENV APP_HOME /opt
ENV SPRING_APPLICATION_NAME api
ENV TZ Asia/Shanghai

WORKDIR /opt
ADD /opt/api.jar /opt/
RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime
RUN echo $TZ > /etc/timezone

ENTRYPOINT ["/opt/boot.sh", "/opt/api.jar"]
```

# docker-maven-plugin

优点：

- 组件化，可继承复用
- 版本控制
- 可以使用 pom 中的变量

缺点：

- 稳定性
- 依赖 Maven 构建环境

# 纯 Docker 的多阶段

```
FROM maven:3.5-jdk-8 AS build-env

ADD . /src
WORKDIR /src
RUN mvn clean package

FROM docker.mobike.io/common/alpine-oraclejdk8:1.2

ENV APP_HOME /opt
ENV SPRING_APPLICATION_NAME api
ENV TZ Asia/Shanghai

WORKDIR /opt
COPY --from=build-env /src/target/ares-blade.jar /opt/ares
RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime
RUN echo $TZ > /etc/timezone

ENTRYPOINT ["/opt/boot.sh", "/opt/ares-blade.jar"]
```

# 纯 Docker 的多阶段

为什么我更喜欢这种方式?

- 只依赖 Docker
- 不可变构建环境

# 运行

# 资源限制

问题：

1. 资源限制和什么有关?
2. 我们做了资源限制么?
3. 我们为什么做了/没做一些资源限制?

# 资源限制

**资源限制和什么有关?**

在单机容器运行时（Docker/Docker Compose）：

- 一个容器最多只能使用限制的 CPU 数量
- 当容器内存申请超过限制的内存时，容器会被杀掉（OOM Kill）

在集群模式下运行时（Swarm/Kubernetes）：

- 容器只会被分配到有足够资源的节点中
- 集群会使用不同的算法，根据每个节点的当前资源和容器申请的资源，找到最合适的节点

# 资源限制

## 我们做了资源限制么?

以 https://mobike.io/user/user 为例

**docker-compose.prod.yml** 617 Bytes

```yaml
version: '3.2'
services:
  api:
    deploy:
      replicas: 18
      resources:
        reservations:
          cpus: '2'
  region:
    deploy:
      replicas: 8
      resources:
        reservations:
          cpus: '2'
  mix:
    deploy:
      replicas: 8
      resources:
        reservations:
          cpus: '2'
```

**user-prod.yaml** 7.63 KB

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: user-api
spec:
  replicas: 18
  selector:
    matchLabels:
      app: user-api
  template:
    metadata:
      labels:
        app: user-api
    spec:
      containers:
      - name: user-api
        image: docker.mobike.io/user/user-api:1.7.35
        imagePullPolicy: Always
        resources:
          requests:
            cpu: "2000m"
        readinessProbe:
          httpGet:
            path: /health
            port: 9900
          initialDelaySeconds: 35
          periodSeconds: 5
        livenessProbe:
          httpGet:
            path: /health
            port: 9900
          initialDelaySeconds: 60
          periodSeconds: 5
        ports:
        - name: http
          containerPort: 9900
```

# 资源限制

**为什么我们没有限制内存?**

通过 JVM 参数限制内存

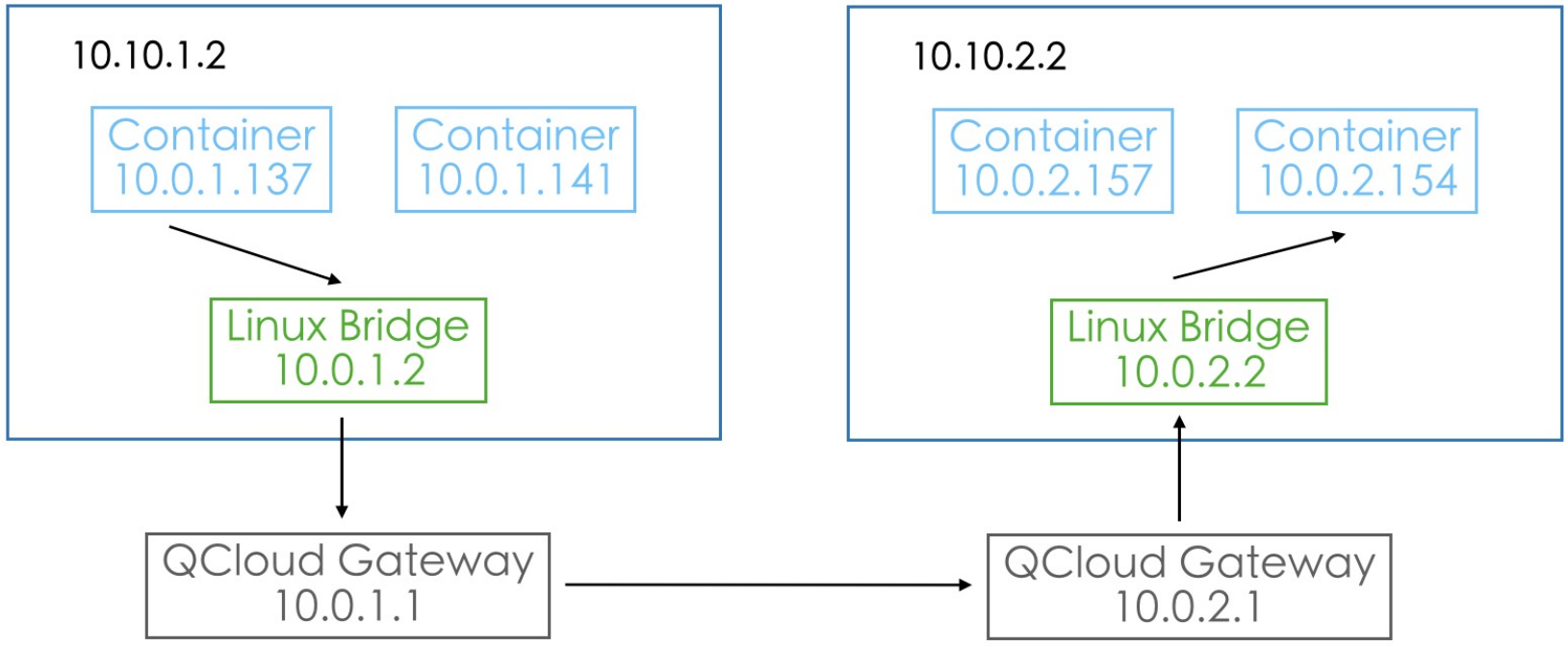
好处:

- 不用担心在发生 GC 之前，容器就被 OOM Kill 杀掉

坏处:

- JVM 的内存限制可能会遗漏一些特殊的堆外内存，导致内存泄露
- 集群分配算法将会忽略内存因素（通过 reservations/requests 避免）

# 网络



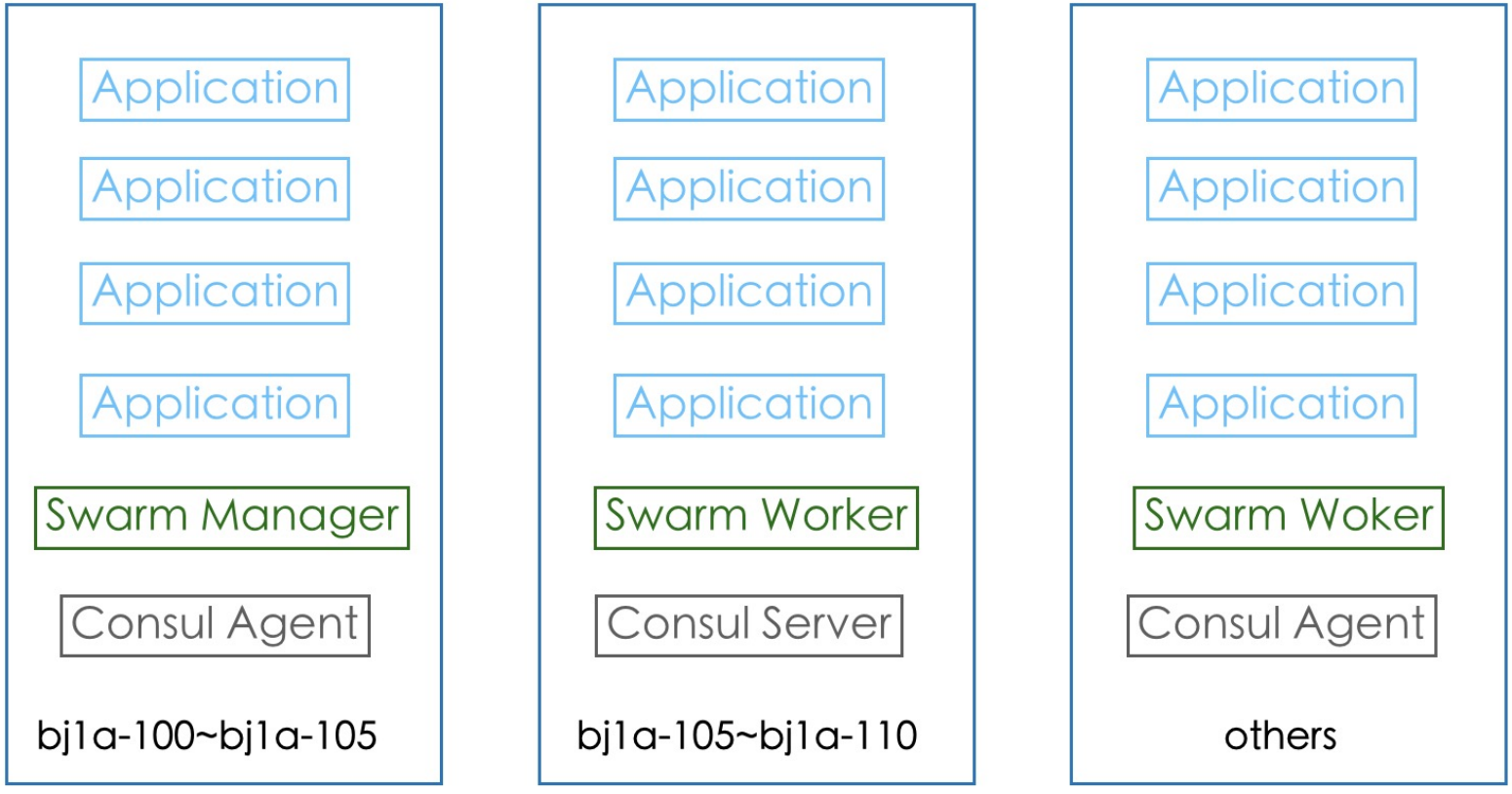

https://wiki.mobike.com/pages/viewpage.action?pageId=18669586

https://wiki.mobike.com/pages/viewpage.action?pageId=22347559

https://wiki.mobike.com/pages/viewpage.action?pageId=29278470

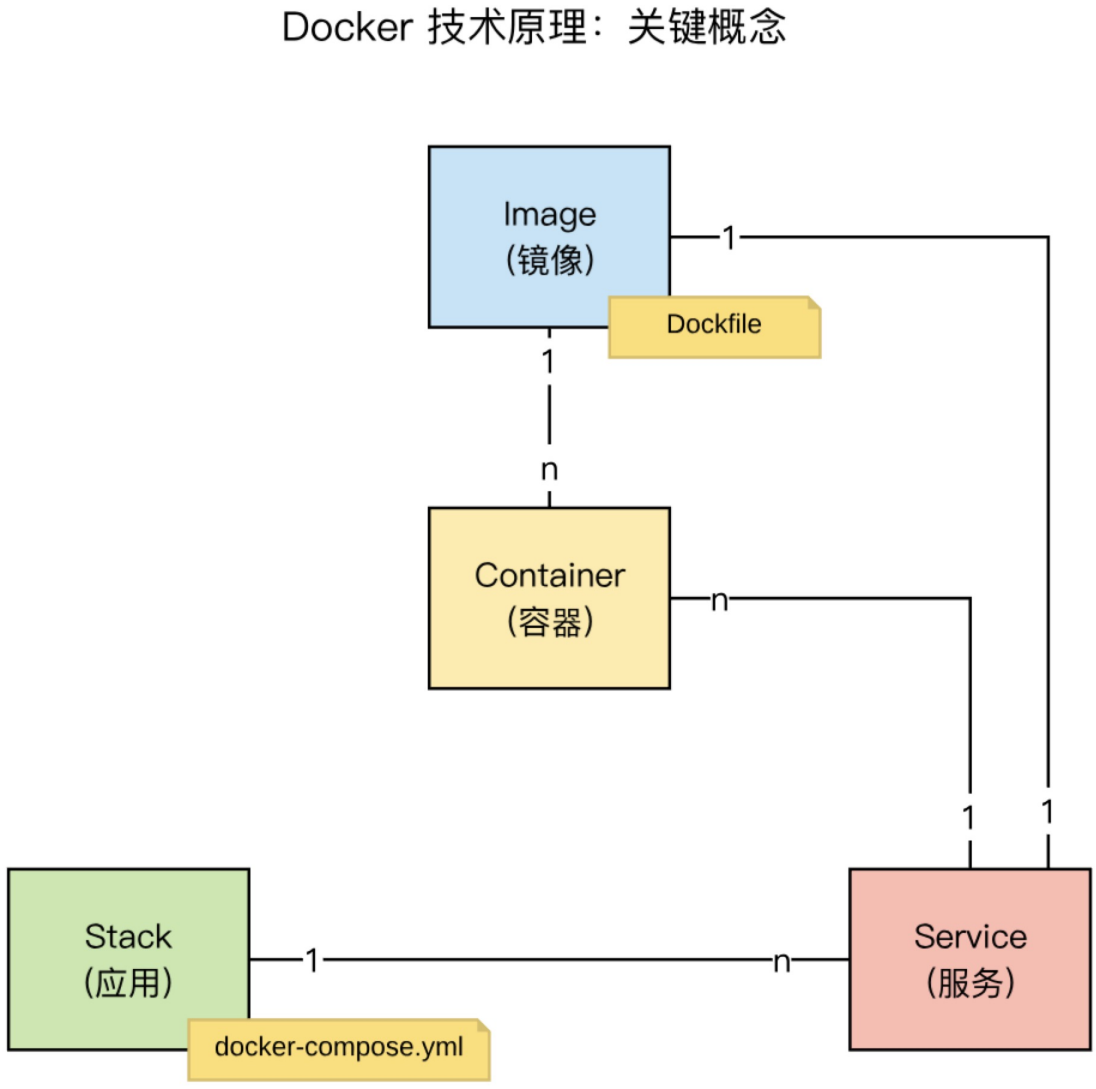
# 集群

# Swarm 物理架构




混部 Swarm Manager 的问题：部署时 Swarm Manager 压力增大影响到同一台机器上的容器，容器崩溃后重启进一步增加 Swarm Manager 的压力。

# Swarm 逻辑架构


Docker 技术原理：关键概念
Image
(镜像)
Dockfile
1
1
n
Container
(容器)
n
1
1
Stack
(应用)
1
n
Service
(服务)
docker-compose.yml

# Kubernetes

## 为什么要从 Swarm 切换到 Kubernetes?

从容器化走向云原生

# Kubernetes 新功能

- Sidecar
- 服务注册和发现
- 配置中心
- 定时任务
- 调度亲和性
- Service Mesh

# Q & A