服务管理设计文档

作者：Lance

修订：2020.09.27

# 背景

Magic-box日益庞大，服务的打包部署较为繁琐，需要一个自动化服务来完成打包、部署、上线等一系列操作，此外，还需承担持续集成，日构建等功能。

# 总体架构

## 整体TOPO

用户请求从外部路由负载均衡到达cd服务器的tomcat web服务器，服务管理开始处理服务打包请求，将打包任务发送给消息中间件并刷新任务状态，打包服务接受到任务后，进行代码拉取，制作镜像，刷新数据库并将镜像上传至文件服务器。



## 基本功能及技术栈

# 模块设计

## 用户权限管理

权限管理由user\_mgt模块承担，包括用户创建，权限赋权，权限设计。

## 基础镜像管理

基础镜像作为服务打包所依赖镜像，与服务镜像分开管理，但本质还是属于镜像，可由服务镜像设置而来，支持增删改查等操作。

## 服务镜像管理

由基础镜像和服务的功能文件打包而成，可对服务镜像进行增删改查等操作。

## 服务管理

管理已运行的服务，从已发布的服务镜像运行新的服务。

## 打包服务

接收打包任务，基于基础镜像以及dockerfile制作新的服务镜像，并进行镜像数据存储。

## 负载均衡

暂只针对打包服务做负载均衡以及HA。

# 表结构设计

1. t\_docker\_image（镜像数据表）

CREATE TABLE IF NOT EXISTS `t\_docker\_image` (

`id` INT UNSIGNED NOT NULL AUTO\_INCREMENT,

`owner` INT UNSIGNED NOT NULL COMMENT 't\_user.id',

`image\_name` varchar(50) not null comment 'Image Name',

`image\_path` varchar(100) not null comment 'Image Path',

`is\_base\_images` TINYINT(1) UNSIGNED NOT NULL DEFAULT '0',

`is\_deleted` tinyint(1) unsigned not null default '0' comment 'logic delete 0: (false) exist, 1: (true) deleted',

`created\_time` datetime not null comment 'create time',

`modified\_time` datetime not null comment 'update time',

primary key (`id`),

unique key `uk\_path` (`image\_path`)

)ENGINE = innodb DEFAULT CHARSET = utf8mb4;

1. t\_services（服务表）

CREATE TABLE IF NOT EXISTS `t\_services` (

`id` INT UNSIGNED NOT NULL AUTO\_INCREMENT,

`owner` INT UNSIGNED NOT NULL COMMENT 't\_user.id',

`service\_name` varchar(50) not null,

`image\_id` INT UNSIGNED NOT NULL COMMENT 't\_docker\_image.id',

`is\_deleted` tinyint(1) unsigned not null default '0' comment 'logic delete 0: (false) exist, 1: (true) deleted',

`created\_time` datetime not null comment 'create time',

`modified\_time` datetime not null comment 'update time',

primary key (`id`),

unique key `uk\_path` (`service\_name`)

)ENGINE = innodb DEFAULT CHARSET = utf8mb4;

# 中间件选型及设计

服务管理需要一个消息队列中间件来将web后端与打包服务做解耦，而消息主要是打包任务，发送量比较小，消息体也比较小，故无需kafka这种大吞吐量的消息队列服务，而rabbitmq就相对较为合适，能够满足点对点的少量消息精准推送，故而选用rabbitmq作为消息中间件。

# 可行性分析

整体架构设计无技术盲点，实现简易，能较为快速的交付功能。

# 性能分析

Cd的service只是做任务接受以及web数据提供，压力较小，单server也能很好的承载，但打包服务因为镜像的build过程需要耗费大量的cpu，所以容易造成任务积压，于是设计了可扩展的打包服务，由nginx做负载均衡，避免单台server任务积压。