微服务平台解决方案

- 服务中心
- docker镜像管理
- 配置中心
- API网关
- 部署发布
- 监控
- 版本控制

系统环境

```
  linux:centos7
  docker:1.10.3+swarm+docker-proxy
```

系统环境描述

```
swarm：进行集群管理
docker-proxy 对外提供基于Rest的远程API调用
```

服务中心

```
spring cloud 支持：consul，eureka，zookeeper。

选择 consul用于服务注册以及发现
```

consul服务中心

- 提供 REST API 接口
    - k,v 存储
    - 服务注册
- 基于docker可以非常容易的实现集群的部署
- 提供多数据中心
- 主动发起对服务的健康检查

服务注册

spring cloud：程序启动注册服务

缺点

1. 暂时无法实现版本控制
2. 需要提供注册中心的配置信息
3. 必须保证启动的容器端口和程序启动端口保持一致
4. 扩展和维护性不强

swarm部署服务： 通过监听swarm部署服务，实现服务注册。

优点

1. 实现容器既是服务
2. 不需要对程序提供注册中心配置
3. 不需要达到端口一致，端口可以钉死
4. 可以在容器里部署任何web项目，对于java不局限于spring cloud。
5. 可扩展性比较强，可以实现版本上的控制
6. 研究spring cloud 源码发现，程序会定时获取服务配置信息，并切检查。

选择

基于此我们选择通过对swarm启动服务监听来自动注册服务

docker镜像管理

平台基于docker，因此我们通过apphouse来实现docker镜像的管理。

服务镜像上传

通过脚本打包镜像并上传

缺点

1. 需要镜像库的用户的登录名和密码
2. 需要提供镜像名字以及版本

通过gitlba-ci实现镜像自动化打包上传

优点

1. 无需知道镜像库登录名以及密码
2. 只需授权runner所在的服务器
3. 无需提供镜像名字以及版本，实现代码提交可以自动化打包以及上传镜像
4. 可以通过分支或tag来实现镜像版本控制
5. 通过runner的权限管理，可以实现镜像库的限制。

选择

暂时用脚本实现镜像打包上传

管理

AppHouse提供web方式来管理镜像

配置中心

基于spring cloud 配置中心实现
通过gitlab用于配置文件的存储
通过spring cloud config server 来对外提供Rest Api的配置服务

API网关

zuul

优点

1. 提供路由配置，可以方便路由配置
2. 提供过滤，可以对进入网关请求设置过滤规则

缺点

1. 服务扩展性不强，进行服务扩展都需要重新编码修改配置，操作过于麻烦。
2. 每次变化都要重新启动zuul。
3. 负载均衡算法太过简单

基于nginx网关

优点

1. 自动化监听注册中心服务变化来加载网关配置
2. 无需重启网关，只需加载配置
3. 可以随意使用nginx的任何负载规则

缺点

```
不可以设置过滤器
```

注册 swarm 集群部署的service的ip

用nginx对外提供网关，通过swarm来负载。

缺点

```
1. 负载可维护性不强
2. consul注册的是服务集群对外网关，通过consul的配置无法监控具体的容器中程序。
```

注册 swarm集群部署服务下的所有容器的ip

通过nginx实现对外网关以及负载

优点

```
consul注册是具体容器的ip，所以可以监控具体的程序
```

部署发布

通过docker swarm来实现服务的部署

监控

```
1. 对于spirng boot 提供：配置，虚拟机，日志，线程，健康，断路器，容器性能等接口。
2. 通过 docker remote api 可以实现对docker容器的监控以及管理。
```

版本控制

方案

```
1. 通过分支，tag可以实现镜像版本控制
2. 通过启动服务时指定tags，来实现配置nginx时做版本号控制
```

例子

1. swarm启动服务通过serviceName_version来指定服务名字。
2. tags里面添加需要合并到的版本号以及是否合并。
3. 刷新nginx服务网关配置时，根据规则来加载网关配置。