# docker运行平台marathon操作指导

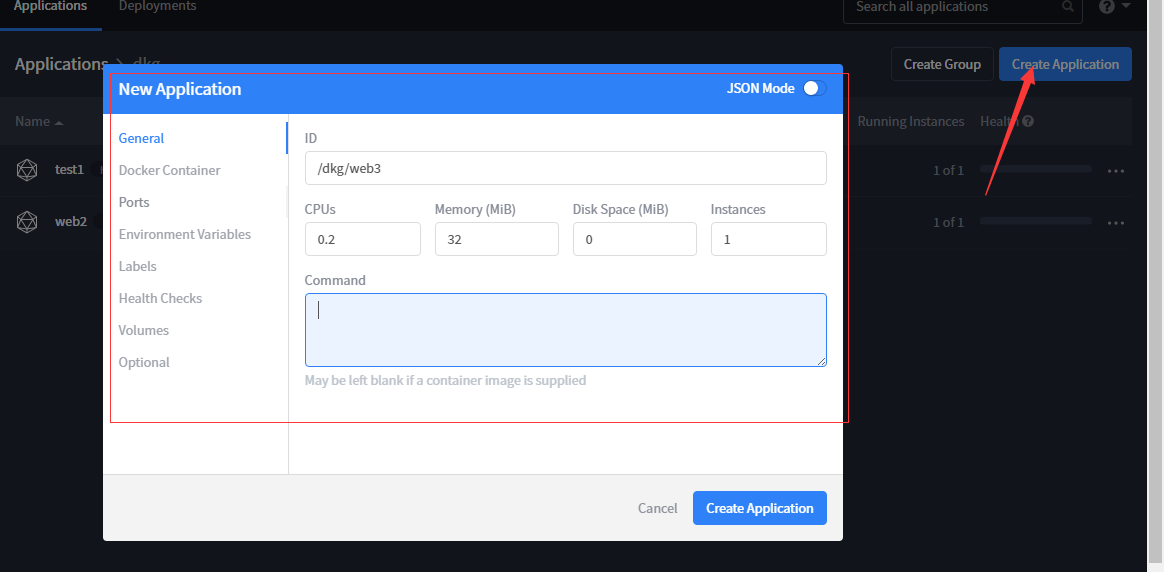
## 1. 进入控制台

<http://192.168.2.37:8080/ui/>

## 2. 发布应用

应用可进行分组管理，相当于可以放在不同的文件夹里面

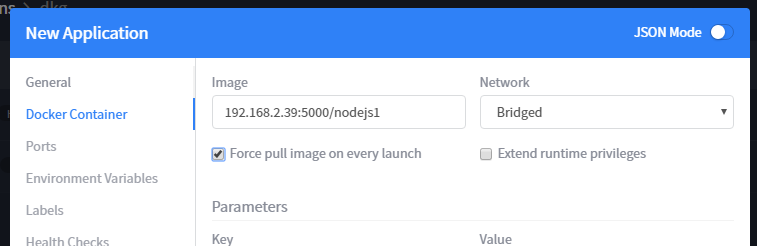
可以在某一分组下进行Create Application



## 3. 镜像及容器的设置

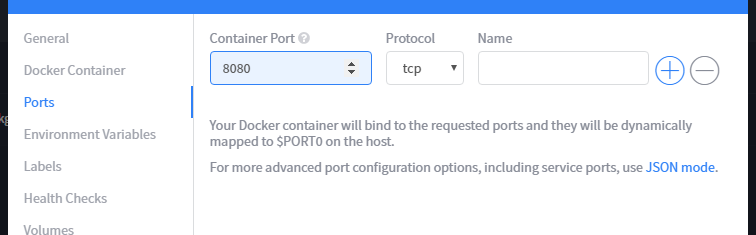
192.168.2.39:5000为镜像库地址，新镜像需要上传到镜像库地址

使用桥接网络， 每个容器使用单独的子网IP



## 4. 开放的端口设置

其中容器开放8080端口， 平台将会将容器8080映射到主机上随机端口



## 5. 应用标签设置

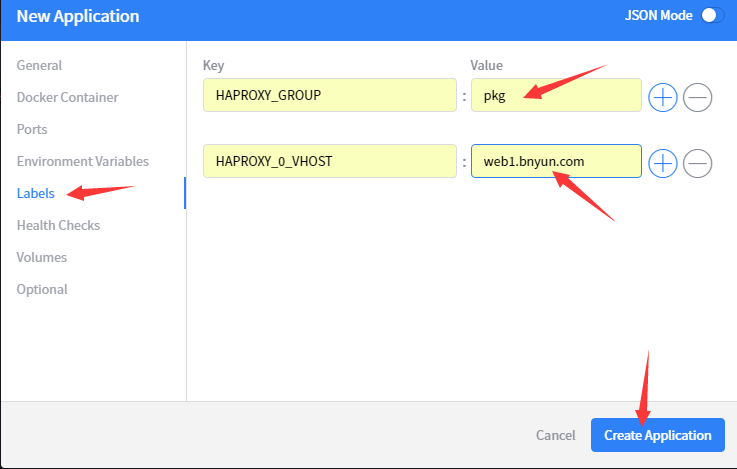
如果需要使用haproxy进行负载均衡，需要增加相关的标签并确认发布

相关的HAPROXY\_GROUP与启动的haproxy标签必须相同，才能进行反向代理，

本环境中

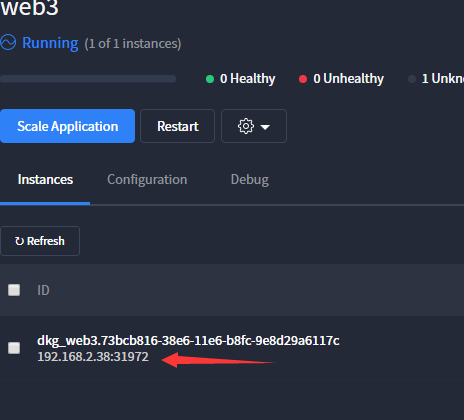
192.168.2.39 上80端口的haproxy 代理 group = pgk的

192.168.2.38 上80端口的haproxy 代理 group = host-pgk的

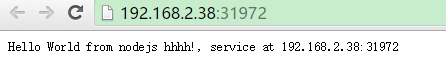


## 7. 启动后初步测试

应用进入running状态后,可以点击相关的链接访问，这边marathon平台已经进行的端口的映射管理



结果



## 8. 负载均衡处理

本例中group=dgk使用的是192.168.2.37上的haproxy进行负载的 vhost=web1.bn.com将指到对应的docker  application中，可以负载多个容器实例

因此增加hosts配置

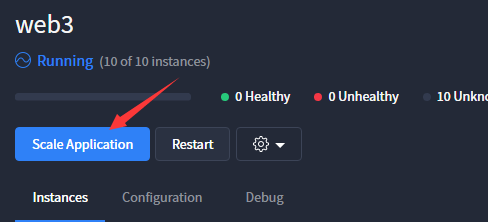
IMG_262

## 9. 单实例时负载均衡



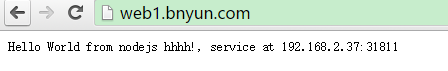
## 10 .应用扩展实例数

当前应用只有一个实例，进行应用的扩展后，扩展成10个实例，其中实例如果被后台stop或rm,平台会自动重启实例以保证实例数



## 11.多实例下负载均衡

多次刷新请求相关的负载地址，每次返回值都可能不相同，表示已经请求到不同应用实例了





## 12.应用启停任务情况查看

mesos平台可查看每次任务执行情况

界面： http://192.168.2.39:5050

可以查看每次的发布任务、资源、任务重启及相关日志情况

## 13.本用例中镜像源码

## server.js 其中HOST, PORT\_8080均是容器的环境变量

##　可以使用docker inspect <id>命令进行查看

var http = require('http');

var ppp = process.env.HOST +":"+ process.env.PORT\_8080;

var handleRequest = function(request, response) {

  response.writeHead(200);

  response.end("Hello World from nodejs hhhh!, service at " + ppp );

}

var www = http.createServer(handleRequest);

[www.listen(8080);](http://www.listen(8080);)

## Dockerfile文件

FROM index.alauda.cn/library/node

EXPOSE 8080

COPY server.js .

CMD node server.js

## 使用命令生成镜像

docker build -t 192.168.2.39:5000/nodejsx .

＃上传到到镜像库即可使用

docker push  192.168.2.39:5000/nodejsx