

# 拾云平台实际使用案例

# 案例一 使用私有镜像部署的内部服务案例

- 大部分项目都会有众多的内部服务，支撑各业务模块的正常运行，数据库就是一个典型的例子，此处以MySQL作为案例进行演示说明。

# 项目/应用

项目/应用作为平台的基础对象，在应用页面的右上角即可找到创建入口（红框所示）






# 添加项目/应用

添加项目/应用页面，  
若是此账号第一次进行  
镜像来源选择操作，会  
自动跳转到GitHub进行  
授权绑定

5:27

< 添加主应用

图标 

名称

描述

标签

镜像来源

确定添加

# 应用详情

添加完成之后，可以在详情页面查看到具体的信息。此时因为还没进行具体配置部署，所以服务入口以及微服务列表还是处于暂无数据的状态。但是项目添加后会自动创建若干应用（项目同名应用、Ingress控制器以及默认后端应用）

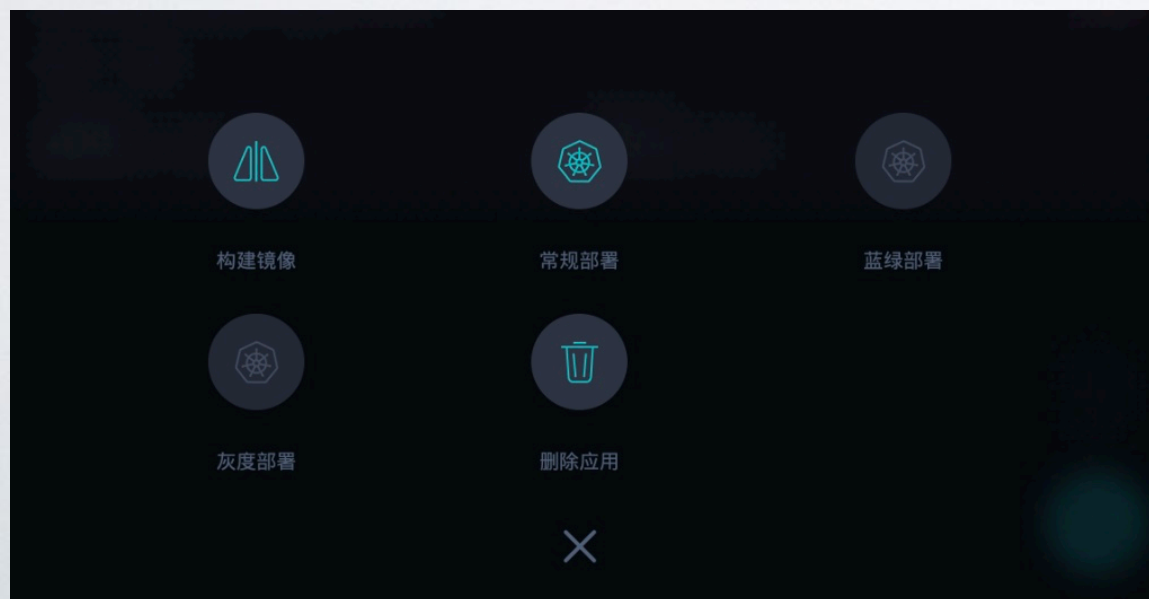


# 应用操作

如右图MySQL子应用页面可以看到应用详情。

一个常规的应用端到端通常包含构建镜像、部署实例、创建服务三个步骤。

子应用右下角的配置按钮作为操作入口之一，可以看到镜像和部署的操作都在这里





# 创建镜像

运行一个应用的第一步就是创建镜像，核心操作就是绑定好代码仓库，并且在平台上编辑好对应的Dockerfile  
Dockerfile的编写规范可以参看网络资料，此处因为使用的MySQL镜像已有成熟的官方镜像源，直接引用即可,开始构建之后页面会显示构建日志



Dockerfile



# 常规部署

创建完镜像之后，我们就可以开始进行部署任务：

- 实例数量
- 实例标签
- 容器名称
- 镜像
- 端口（MySQL的默认端口为3306）

常规部署

部署定义 [跳过，直接用YAML定义部署](#)

预设实例数量 2

实例标签  
app=mysql

容器

名称 mysql

镜像 mysql:5.7 [重新选择](#)

端口 http >

高级配置 >

添加新容器

添加端口

确认

端口

名称 http

协议 TCP >

端口号 3306

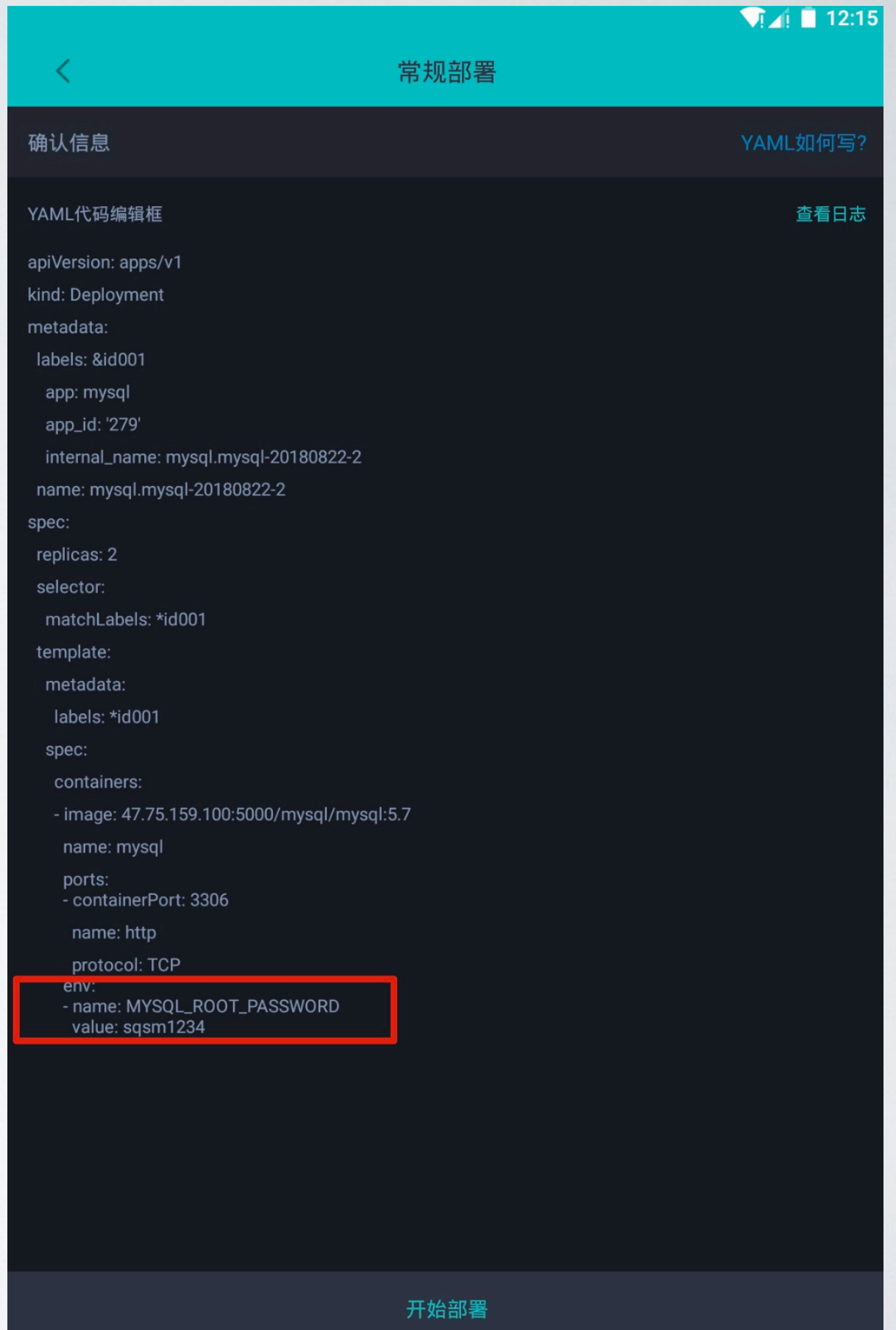
添加新端口

端口配置



# 常规部署YAML

配置选择完成之后，会生成对应的yaml内容，填写一个默认的MySQL管理员密码方便后续使用，手工在yaml中添加对应的env字段(key值MYSQL\_ROOT\_PASSWORD，value值用户自定义，注意缩进)，填写完成之后即可开始部署了



# 部署详情

部署完成之后，可以在部署详情页面查看相应信息：

- 运行总览
- 运行实例状态
- 部署定义（即yaml）
- 更新记录

18:06

部署详情

mysql-20180822-3

关联应用 应用名称 mysql

创建时间 2018-08-22 17:58:44

运行情况 

刷新

运行总览

2个

预设应用实例数量

2个

当前应用实例数量

2个

可用应用实例数量

2个

更新应用实例数量

运行时间 7m

检测状态 已完成

原因

状态产生的可能原因分析

运行实例 

状态说明

更多 >

mysql-20180822-3-89f74dd9c-rd6d2

实例ID 113

就绪情况 1/1

运行时间

创建时间 2018-08-22 17:58:50

标签 mysql=459308857

状态 Running

重启次数 0

mysql-20180822-3-89f74dd9c-rmbr8

实例ID 113

就绪情况 1/1

运行时间

创建时间 2018-08-22 17:58:50

标签 mysql=459308857

状态 Running

重启次数 0

部署定义

deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels: &id001
  app: mysql
  app_id: '279'
```

# 创建服务

实例部署完成之后，就是创建服务步骤。在主应用页面的右下角可以进入创建服务页面。

- 关键步骤1：选择匹配的实例标签（可使用逗号分隔多个筛选标签），可选择的标签可以通过右上集群实例标签链接查看；
- 关键步骤2：此处我们选择服务暴露方式为集群内访问，指定端口号（集群内访问此服务的端口），目标端口（部署时配置的实例对外暴露端口）

生成服务的yaml内容之后即可开始创建服务





# 服务详情

服务创建完成之后在详情页可以查看相关信息：

- 基本信息
- 当前状态
- 服务后端（通过ip:port就可以在内部访问使用此服务）
- 服务定义（即yaml）

至此，MySQL的内部服务已经创建完成并可以正常提供服务，后续如果用户需要变更服务内容，只要通过部署或服务页面右下角的更新按钮进入就可以修改相应配置。

18:44

服务详情

mysql

Uid49

创建时间：2018-08-22 18:40:24

基本信息

服务来源

内部服务，通过标签选择

服务标签

app\_id=278,internal\_name=mysql.mysql,

暴露类型

集群内访问

当前状态

刷新

检测状态

成功

访问服务

10.111.147.28:6789,

服务后端

地址

ip:10.244.4.53  
ip:10.244.7.20

端口

name:http, port:3306

服务定义

service.yaml

apiVersion: v1  
kind: Service  
metadata:  
  labels:  
    app\_id: '278'  
    internal\_name: mysql.mysql  
  name: mysql  
spec:  
  ports:  
    - name: http  
      port: 6789  
      protocol: TCP  
      targetPort: 3306  
  selector:  
    app: mysql  
    app\_id: '279'

## 案例二 使用云数据库作为外部服务案例

- 另外，在很多项目的实际使用过程中，并非自己搭建数据库环境，而是直接使用了现成的云产商提供的数据库服务。拾云平台同样可以整合这些云厂商的数据库功能，方便用户创建灵活的应用。

# 外部服务

差异就在于服务来源选项：

- 集群外，通过IP:端口映射服务

填写完IP地址与端口号之后，  
后续步骤与集群内服务相同。

通过yaml内容可以看出系统会自动生成对应的endpoint节点，创建完成后即可在集群内部访问管理此服务了。

<

创建服务

下一步

选择服务来源

跳过，直接用YAML定义部署

选择服务来源

集群外，通过IP:端口 映射服务

IP地址输入 54.79.191.43

端口号 3306

YAML代码编辑框

查看日志 编辑

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app_id: '278'
    internal_name: mysql.extern-mysql
  name: extern-mysql
spec:
  ports:
    - name: http
      port: 3306
      protocol: TCP
      targetPort: 3306
  type: ClusterIP
---
apiVersion: v1
kind: Endpoints
metadata:
  labels:
    app_id: '278'
    internal_name: mysql.extern-mysql
  name: extern-mysql
subsets:
  - addresses:
      - ip: 54.79.191.43
    ports:
      - port: '3306'
```



# 服务详情

外部服务创建完成之后同样可以在详情页查看相关信息：

- 基本信息
- 当前状态
- 服务后端（通过ip:port就可以在内部访问使用此服务）
- 服务定义（即yaml）

至此，即使是外部的服务也可以被我们所利用，不仅仅是数据库，各种API化的接口、服务都可以通过这个方式加入到项目中来。

<

服务详情

extern-mysql

Uid50

创建时间：2018-08-24 17:34:14

基本信息

服务来源

内部服务，通过标签选择

服务标签

app\_id=278,internal\_name=mysql.extern-mysql,

暴露类型

集群内访问

当前状态

刷新

检测状态

成功

访问服务

10.110.101.176:3306,

服务后端

服务定义

service.yaml

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app_id: '278'
    internal_name: mysql.extern-mysql
  name: extern-mysql
spec:
  ports:
    - name: http
      port: 3306
      protocol: TCP
      targetPort: 3306
    type: ClusterIP

---
apiVersion: v1
kind: Endpoints
metadata:
  labels:
    app_id: '278'
```

# 服务详情

服务创建完成之后在详情页可以查看相关信息：

- 基本信息
- 当前状态
- 服务后端（通过ip:port就可以在内部访问使用此服务）
- 服务定义（即yaml）

至此，MySQL的内部服务已经创建完成并可以正常提供服务，后续如果用户需要变更服务内容，只要通过部署或服务页面右下角的更新按钮进入就可以修改相应配置。

18:44

服务详情

mysql

Uid49

创建时间：2018-08-22 18:40:24

基本信息

服务来源

内部服务，通过标签选择

服务标签

app\_id=278,internal\_name=mysql.mysql,

暴露类型

集群内访问

当前状态

刷新

检测状态

成功

访问服务

10.111.147.28:6789,

服务后端

地址

ip:10.244.4.53  
ip:10.244.7.20

端口

name:http, port:3306

服务定义

service.yaml

apiVersion: v1  
kind: Service  
metadata:  
  labels:  
    app\_id: '278'  
    internal\_name: mysql.mysql  
  name: mysql  
spec:  
  ports:  
    - name: http  
      port: 6789  
      protocol: TCP  
      targetPort: 3306  
  selector:  
    app: mysql  
    app\_id: '279'



# 案例三：搭建并从外部访问WORDPRESS 系统案例

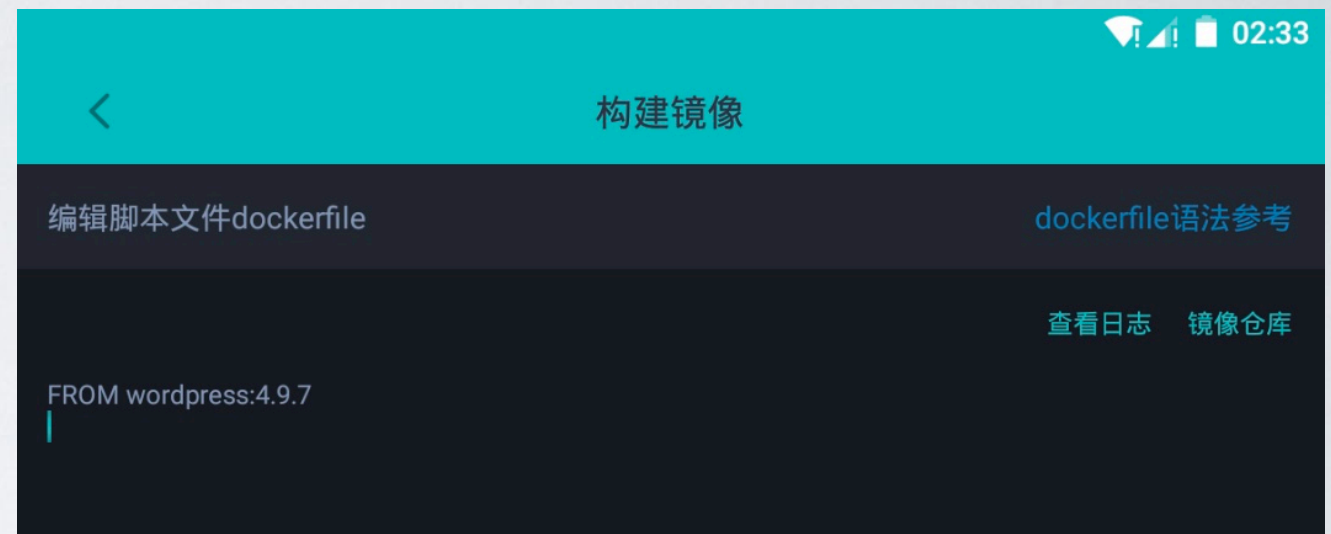
- 前面两个案例所述的都是内部服务，常用在项目的内部组件部署维护上，但是大部分应用最终还是要对外提供服务的，下面通过搭建一个WordPress博客系统，介绍一下如何在外部访问我们创建的项目，最后还会展示ingress功能如何快速地将服务绑定到域名、路由上。



# 创建镜像

建立一个WordPress项目，由数据库和服务端后端两个应用组成，数据库应用创建镜像、部署、服务可以直接参照案例一所以在此不再赘述，直接来看看服务端后端如何创建：

- 创建子应用
- 创建镜像：WordPress同样有成熟官方镜像源，我们直接在Dockerfile中引用即可（参考右图）



构建WordPress镜像

# WORDPRESS部署(I)

镜像创建完成之后我们就可以部署WordPress的程序了，填写信息如下

- 实例数量（在K8S集群中运行的实例数量，考虑目前使用的机器性能最好不要过多，否则可能导致服务器卡顿）
- 实例标签（格式xxx=yyy）
- 添加容器（选用之前创建好的WordPress镜像）
- 端口（WordPress后端使用的是80端口，具体配置可见右下图）

常规部署

部署定义 [跳过，直接用YAML定义部署](#)

预设实例数量 2

实例标签  
app=wordpress-demo-0827

容器

名称	wordpress
镜像	wordpress-demo-web:4.9.7 <a href="#">重新选择</a>
端口	<a href="#">添加端口 &gt;</a>
高级配置	<a href="#">&gt;</a>

+ 添加新容器

添加端口

端口

名称	http
协议	TCP <a href="#">&gt;</a>
端口号	80

+ 添加新端口

# WORDPRESS部署(2)

下一步之后，进入常规部署的yaml内容页面。因为WordPress后端需要访问数据库，所以这里我们要加上数据库的相关参数，注意缩进要一致：

- 数据库服务名称  
WORDPRESS\_DB\_HOST
  - 注意这个value必须和前面创建的数据库服务名称相同
- 数据库密码  
WORDPRESS\_DB\_PASSWORD
  - 注意这个value必须和前面创建的数据库密码相同

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels: &id001
  app: wordpress-demo-0827
  app_id: '266'
  internal_name: wordpress-demo-web.wordpress-demo-web-20180823-4
  name: wordpress-demo-web.wordpress-demo-web-20180823-4
spec:
  replicas: 2
  selector:
    matchLabels: *id001
  template:
    metadata:
      labels: *id001
    spec:
      containers:
        - image: 47.75.159.100:5000/wordpress-demo-web/wordpress-demo-web:4.9.7
          name: wordpress
          ports:
            - containerPort: 80
              name: http
              protocol: TCP
          env:
            - name: WORDPRESS_DB_HOST
              value: mysql
            - name: WORDPRESS_DB_PASSWORD
              value: sqsm1234
```



# 创建WORDPRESS服务

部署完成之后就可以开始创建相应的WordPress服务：

- 选择服务来源
  - 集群内应用，通过标签选择
  - 标签输入：输入之前WordPress部署时的标签
- 暴露方式
  - 开放节点（因为WordPress服务需要被外部访问到）
- 端口
  - 端口号：服务的内部访问端口号
  - 目标端口：这个需要和前面创建部署时使用的端口一致

<

创建服务

下一步

选择服务来源

跳过，直接用YAML定义部署

选择服务来源

集群内应用，通过标签选择

标签输入

查看集群内有哪些实例标签

app=wordpress-demo,app\_id=266

<

创建服务

下一步

设置端口和IP信息

选择服务暴露方式

开放节点端口，集群外部可访问

访问IP

系统指派，需要调整到下一步yaml文件...

端口

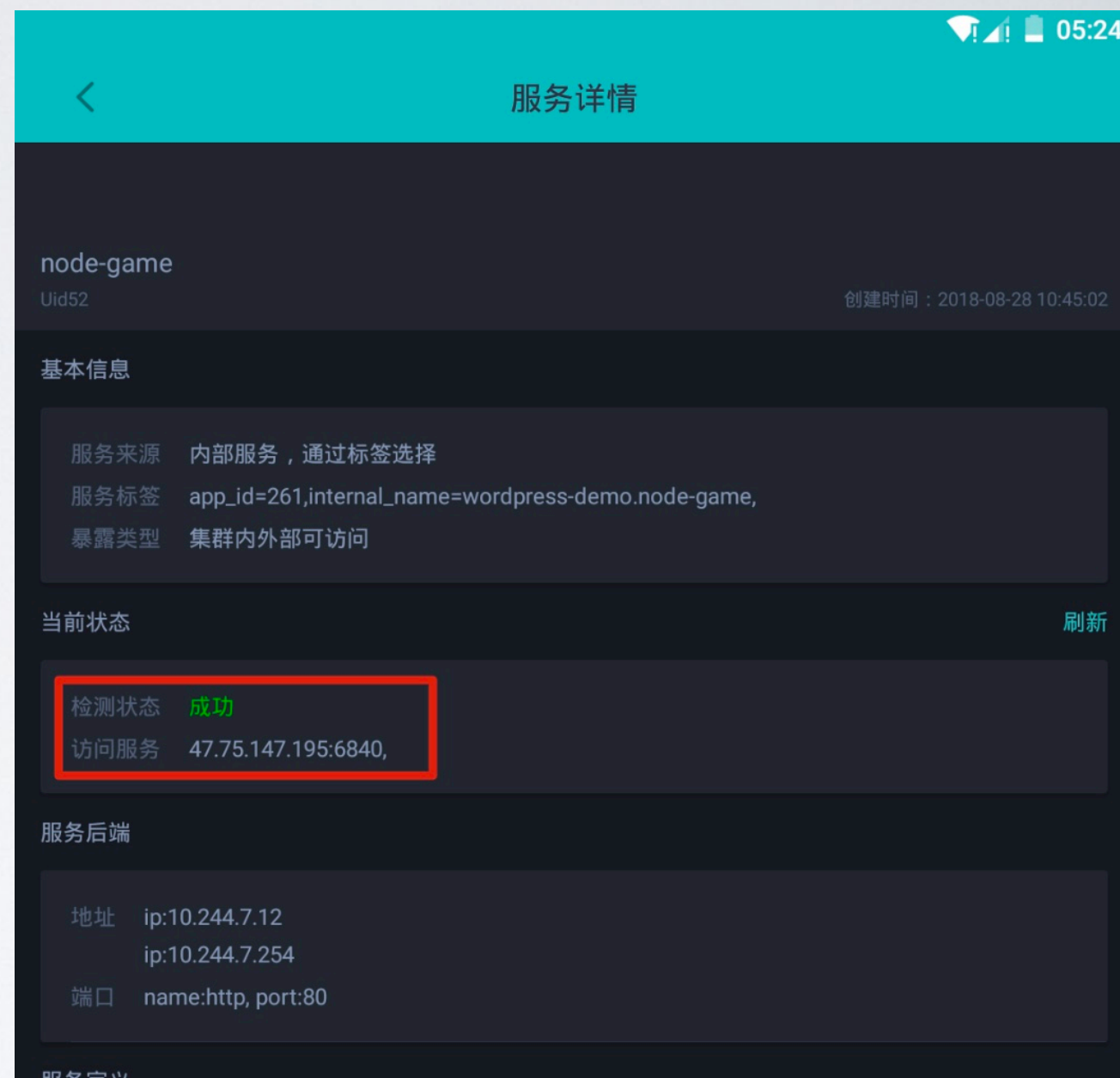
名称	http
协议	TCP
端口号	6767
目标端口	80

+ 添加新端口

# 外部访问WORDPRESS

选择开放节点模式之后，就可以在集群外访问我们刚刚创建的WordPress服务了。在服务详情页面可以看到外部访问地址，由nodeIP和nodePort组成

- nodeIP
  - 集群内部的master节点IP
- nodePort
  - 系统自动分配的访问端口



外部访问地址

# INGRESS介绍

使用IP:Port的方式从外部访问服务还是不太方便，多数用户会通过Nginx等工具自行进行域名绑定、负载代理等配置，而拾云平台提供了Ingress功能可以帮助用户节约下大量时间，通过简单地几步配置即可提供外部访问的URL。

在主应用的详情页面即可看到Ingress的入口，当用户还没配置的时候访问规则还是空的，可以通过右上角的配置按钮进入配置页面





# INGRESS配置

Ingress的配置界面简洁高效，只需要配置三个参数即可实现外部访问：

- host
  - 外部访问的域名
- path
  - 外部访问的相对路径，“/”代表根路径
- 匹配服务
  - 选择从外部访问的服务，即我们前面创建的WordPress服务

配置完成之后，即可在主应用详情页面看到生效的访问规则，通过外部URL即可方便的访问到内部服务。

