

CODING持续集成-k8s

在2020/10/10 19:44上被fu bin (/xwiki/bin/view/XWiki/fubin)修改

- 持续集成
 - 项目中新建Dockerfile文件
 - 创建构建计划
 - Jenkinsfile模版
 - 环境变量
- 腾讯TKE
 - CODING访问k8s集群权限
- 在TKE中配置目标服务
 - 新建Deployment
 - 新建workload
 - 设置镜像拉取的地址
 - 选择镜像拉取的访问凭证
 - 设置应用在k8s内部到容器外部的端口映射
- 查看应用日志
- 如何停止K8S部署的应用
- 参考文档

持续集成

持续集成基于Jenkinsfile描述构建过程，可以指定使用项目内的Jenkinsfile，也可以在CODING持续集成界面手写Jenkinsfile，还支持通过图形化组件构建整个流程（推荐使用

项目中新建Dockerfile文件

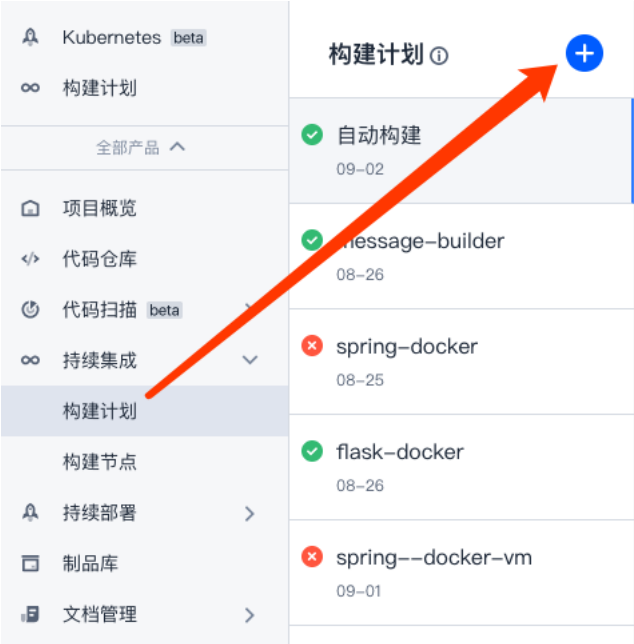
dockerfile文件用于构建项目镜像

C端小程序项目 示例如下:

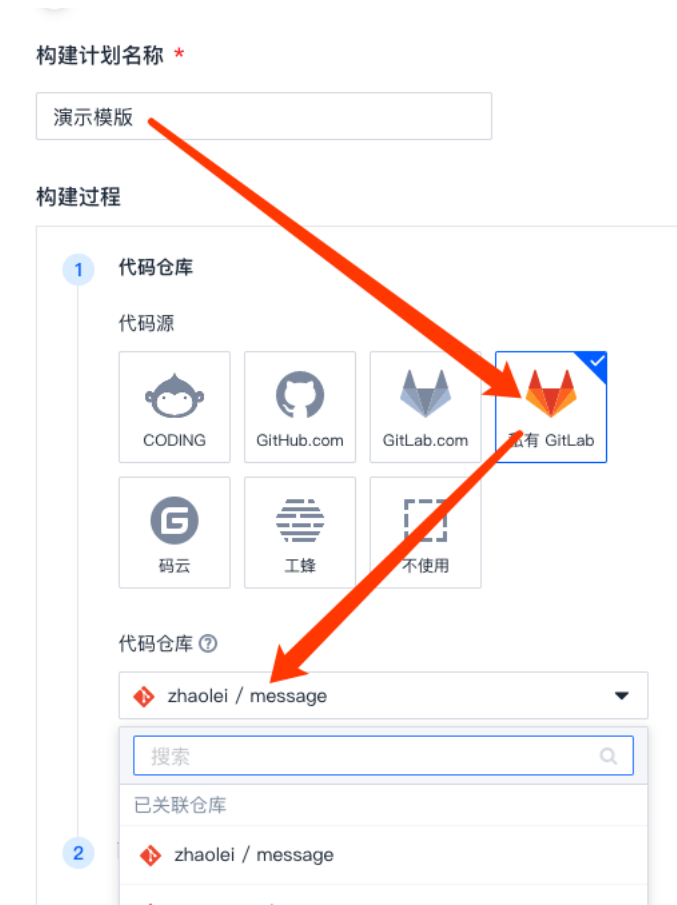
```
# 该镜像需要依赖的基础镜像
FROM java:8
# 将当前目录下的jar包复制到docker容器的/目录下
ADD cgp-app/target/cgp-app-1.0-SNAPSHOT.jar ~/mall-tiny-docker-file.jar
# 运行过程中创建一个mall-tiny-docker-file.jar文件
RUN bash -c 'touch ~/mall-tiny-docker-file.jar'
# 声明服务运行在8080端口
EXPOSE 8080
# 指定docker容器启动时运行jar包
ENTRYPOINT ["java", "-jar", "~/mall-tiny-docker-file.jar", "--spring.profiles.active=fat"]
```

上述例子中 `cgp-app/target/cgp-app-1.0-SNAPSHOT.jar` 为maven构建生成的jar包路径，项目中启动的jar包在子模块target目录下的需要注意。

创建构建计划



点击「+」号后，直接拉到最底下，选择「自定义构建过程」



「代码源」选则「私有GitLab」，代码仓库下拉选择需要发布的项目，注意：需要在Gitlab中将Administrator用户添加为Developer



再根据项目根目录是否有Jenkinsfile自行选择



上图是已经创建好的构建流程，也可以选择文本编辑器将自行编写的Jenkinsfile复制进去

←

spring-boot-demo-build

基础信息

流程配置

触发规则

静态配置的 Jenkinsfile

图形化编辑器

文本编辑器

```
1 pipeline {
2   agent any
3   stages {
4     stage('检出') {
5       steps {
6         checkout([$class: 'GitSCM',
7           branches: [[name: env.GIT_BUILD_REF]],
8           userRemoteConfigs: [[
9             url: env.GIT_REPO_URL,
10            credentialsId: env.CREDENTIALS_ID
11          ]]])
12       }
13     }
14     stage('编译') {
15       steps {
16         echo '开始执行编译'
17         sh 'mvn package -Dmaven.test.skip'
18       }
19     }
20     stage('构建 Docker 镜像') {
21       steps {
```

Jenkinsfile模版

```
//pipeline表示定义整个流程
pipeline {
  agent any
  //stages表示定义流程中的各个操作阶段
  stages {
    //代码检出阶段，指定项目的git地址，包括验证时使用的credential
    //env.GIT_REPO_URL、env.CREDENTIALS_ID等都为系统环境变量，相关信息在关联私有Gitlab到CODING系统时已添加，无需自行配置
    stage('检出') {
      steps {
        checkout([$class: 'GitSCM',
          branches: [[name: env.GIT_BUILD_REF]],
          userRemoteConfigs: [[
            url: env.GIT_REPO_URL,
            credentialsId: env.CREDENTIALS_ID
          ]]])
      }
    }
    //代码编译阶段，直接执行mvn编译命令
    stage('编译') {
      steps {
        echo '开始执行编译'
```

```
sh 'mvn package -Dmaven.test.skip'
}
}
//Docker镜像构建阶段，使用docker build命令构建镜像
//使用environments里的CODING_DOCKER_IMAGE_NAME变量作为镜像名，使用系统变量作为镜像版本，此处可自行定义，例如镜像版本处可加上分支信息
stage('构建 Docker 镜像') {
    steps {
        echo '开始构建镜像'
        sh "docker build -f Dockerfile -t ${env.CODING_DOCKER_IMAGE_NAME}:${env.GIT_COMMIT} ."
    }
}
//推送镜像阶段，将镜像推送到指定镜像仓库
//CODING_DOCKER_REG_HOST也是在environments里定义的变量，使用的是CODING系统制品库对应的镜像仓库，taohuaxiaoxiao-docker.pkg.coding
stage('推送到 CODING Docker 制品库') {
    steps {
        echo '开始推送镜像'
        script {
            docker.withRegistry(
                "${env.CCI_CURRENT_WEB_PROTOCOL}://${env.CODING_DOCKER_REG_HOST}",
                "${env.CODING_ARTIFACTS_CREDENTIALS_ID}"
            ) {
                imageName="${env.CODING_DOCKER_IMAGE_NAME}:${env.GIT_COMMIT}"
                imageFullPath="${env.CODING_DOCKER_REG_HOST}/${env.CODING_DOCKER_IMAGE_NAME}:${env.GIT_COMMIT}"
                docker.image(imageName).push()
            }
        }
    }

    echo '镜像推送完成'
}
}
//部署阶段，credentialsId为需要部署的k8s集群的凭据id，凭据创建在前文已提到
stage('部署到 K8s') {
    steps {
        echo '部署到k8s中...'
        script {
            //请修改 credentialsId: 填入 k8s 凭据ID
            withKubeConfig([credentialsId: 'e05aa0b8-a146-40d0-b19f-4324eefdb196']) {
                //使用 kubectl 创建 K8s 密钥：用于让k8s集群可以从镜像仓库拉取镜像然后部署，其中 DOCKER_USER 和 DOCKER_PASSWORD 为镜像仓库的
                sh(script: "kubectl create secret docker-registry coding --docker-server=${env.CODING_DOCKER_REG_HOST} --docke

                //使用 kubectl 修改 K8s deployment：指定 Docker 镜像链接和密钥，需注意imageFullPath为包含host的完整路径，如taohuaxiaoxia
                // 请修改下面命令中出现的两处 spring-boot-demo 值，第一处为已创建好的Deployment的名称，第二处为本项目部署后对应的container的
                sh "kubectl patch deployment spring-boot-demo --patch '{\\\"spec\\\": {\\\"template\\\": {\\\"spec\\\": {\\\"containers\\\": [\"

            }
        }
    }
}
}
environment {
    CODING_DOCKER_REG_HOST = "${env.CCI_CURRENT_TEAM}-docker.pkg.${env.CCI_CURRENT_DOMAIN}"
    CODING_DOCKER_IMAGE_NAME = "${env.PROJECT_NAME.toLowerCase()}/${env.DOCKER_REPO_NAME}/${env.DOCKER_IMAGE_NAME}"
}
}
```

环境变量

构建过程

构建快照

改动记录

测试报告

通用报告

构建产物

启动参数

环境变量

Jenkinsfile

构建节点

序号	变量名	变量值
1	DOCKER_PASSWORD	*****
2	DOCKER_USER	spring-test-1599565564003
3	DOCKER_IMAGE_NAME	spring-boot-demo
4	DOCKER_REPO_NAME	spring-test
5	DOCKER_IMAGE_VERSION	\${GIT_LOCAL_BRANCH:-branch}-\${GIT_COMMIT}

除系统环境变量外，还需要添加如上图的自定义环境变量，其中前三项需要根据镜像仓库、镜像名称等实际情况填写
可以一次性以键值对的形式复制进去

cgp-app-test-k8s

基础信息

流程配置

触发规则

变量与缓存

通知提醒

静态配置的 Jenkinsfile

图形化编辑器

文本编辑器

```
1 pipeline {
2   agent any
3   stages {
4     stage('检出') {
5       steps {
6         checkout([$class: "GitSCM",
7           branches: [[name: env.GIT_BUILD_REF]],
8           userRemoteConfigs: [[
9             url: env.GIT_REPO_URL,
10            credentialsId: env.CREDENTIALS_ID
11          ]]])
12       }
13     }
14     stage('编译') {
15       steps {
16         echo '开始执行编译'
17         sh 'mvn package -U -Dmaven.test.skip'
18       }
19     }
20     stage('构建 Docker 镜像') {
21       steps {
22         echo '开始构建镜像'
23         sh "docker build -f Dockerfile -t ${env.CODING_DOCKER_IMAGE_NAME}:${env.GIT_COMMIT} ."
24       }
25     }
26     stage('推送到 CODING Docker 制品库') {
27       steps {
28         echo '开始推送镜像'
29         script {
30           docker.withRegistry(
31             "${env.CCI_CURRENT_WEB_PROTOCOL}://${env.CODING_DOCKER_REG_HOST}",
32             "e05aa0b8-a146-40d0-b19f-4324eefdb196") {
33             docker.push(env.CODING_DOCKER_IMAGE_NAME, env.GIT_COMMIT)
34           }
35         }
36       }
37     }
38   }
39 }
```

流程环境

添加构建

变量名

DOCKER

DOCKER

DOCKER

DOCKER

DOCKER

```
DOCKER_IMAGE_VERSION=${GIT_LOCAL_BRANCH:-branch}-${GIT_COMMIT}
DOCKER_IMAGE_NAME=cgp-app-test
DOCKER_REPO_NAME=spring-test
DOCKER_USER=spring-test-1599565564003
DOCKER_PASSWORD=密码（密码可联系加瓦或赵磊获取）
```

镜像仓库用户名和密码的获取方法见参考文档

腾讯TKE

全名Tencent Kubernetes Engine，是腾讯基于k8s包装的容器服务

CODING访问k8s集群权限

见参考文档中录入Kerbertenes凭据相关说明，将Kubeconfig录入到CODING对应项目的凭据管理系统

项目设置

项目与成员

项目公告

开发者选项

开发者选项

接口与事件

外部仓库管理

项目令牌

WebHook

凭据管理

凭据管理 (10)

将密码、私钥、证书等信息存储到凭据管理中，可最大程度的提高凭据的安全性和管控使用权限。在并

凭据名称	已授权数	凭据 ID
	24	
k8s_test	24	e0

需要根据需求将凭据ID复制填入上文Jenkinsfile模版中，替换模版里的内容（例如测试环境和生产环境会是两个k8s集群，则会有两个凭据）

```
withKubeConfig([credentialsId: 'e05aa0b8-a146-40d0-b19f-4324eefdb196'])
```

在TKE中配置目标服务

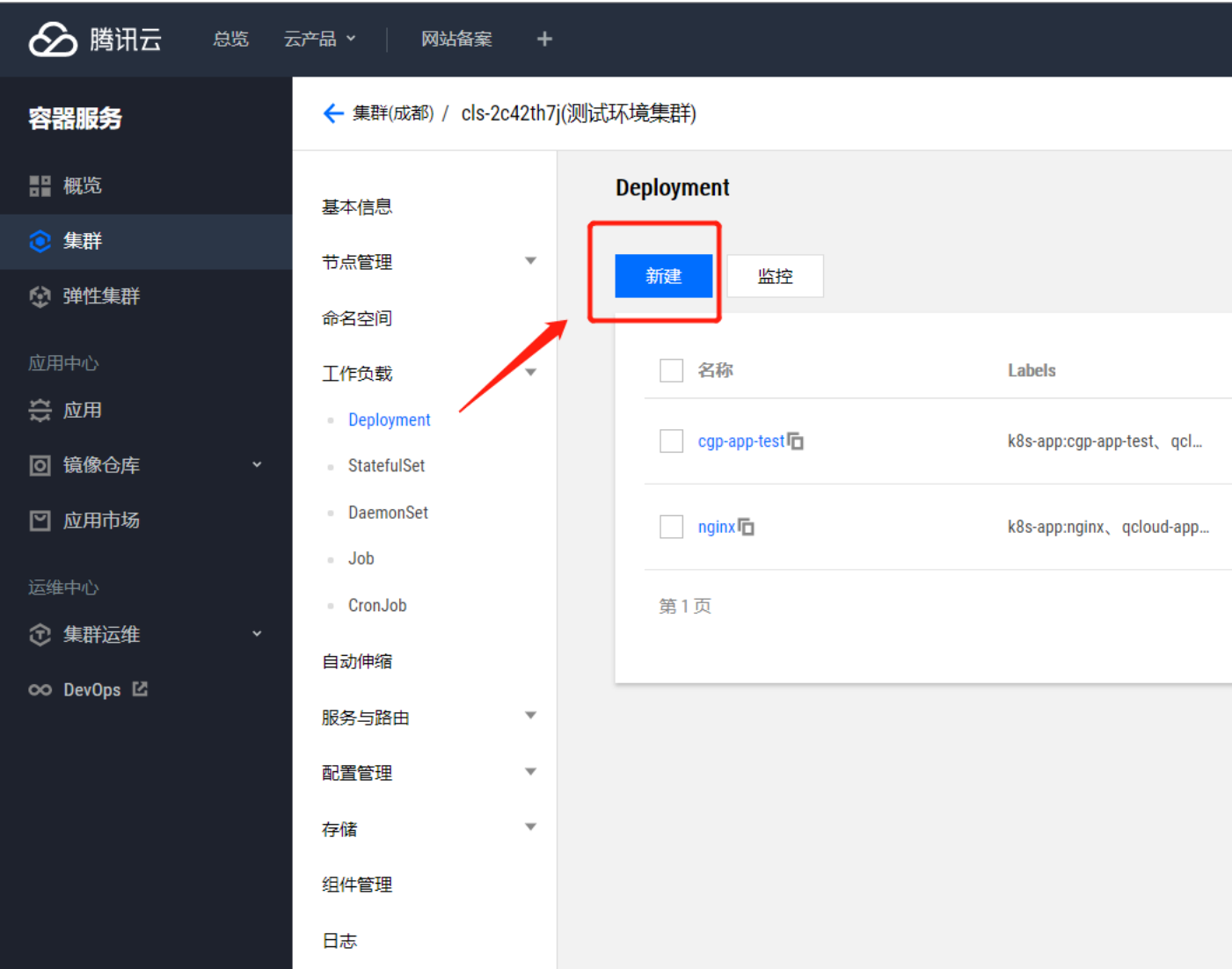
打开 腾讯云的容器服务：<https://console.cloud.tencent.com/tke2> (<https://console.cloud.tencent.com/tke2>)

找到左侧集群菜单栏，进入目标集群



新建Deployment

deployment用于设置部署应用的配置资源，端口映射等



新建workload

填写的工作负载名就是jenkinsfile指定的名称

```
// 请修改 laravel-demo、web 为你的 deployment 中的值
sh "kubectl patch deployment cgp-app-test --patch '{\"spec\": {\"template\": {\"spec\": {\"containers\": [{\"name\": \"cgp-"
```

设置镜像拉取的地址

类型

☒ Deployment (可扩展的部署Pod)

☐ DaemonSet (在每个主机上运行Pod)

☐ StatefulSet (有状态集的运行Pod)

☐ CronJob (按照Cron的计划定时运行)

☐ Job (单次任务)

数据卷 (选填)

[添加数据卷](#)

为容器提供存储, 目前支持临时路径、主机路径、云硬盘数据卷、文件存储NFS、配置文件、PVC, 还需挂载到容器的指定路径中。

实例内容

名称

最长63个字符, 只能包含小写字母、数字及分隔符("-"), 且不能以分隔符开头或结尾

镜像

选择镜像

直接填写spring-tes

镜像版本 (Tag)

一般不填写, 确保拉取最

镜像拉取策略

Always

IfNotPresent

Never

若不设置镜像拉取策略, 当镜像版本为空或latest时, 使用Always策略, 否则使用IfNotPresent策略

CPU/内存限制

CPU限制

request

0.25

-

limit

0.5

核

内存限制

request

256

-

limit

1024

M

Request用于预分配资源,当集群中的节点没有request所要求的资源数量时,容器会创建失败。
Limit用于设置容器使用资源的最大上限,避免异常情况下节点资源消耗过多。

创建Workload

取消

选择镜像拉取的访问凭证

镜像访问凭证

已有访问凭证

qcloudregistrykey

×

添加镜像访问凭证

coding

coding-docker-registry

coding-registry-cred-8016195

qcloudregistrykey

tencenthubkey

实例数量

☒ 手动调节

☐ 自动调节

直接设定实例数量

实例数量

-

1

+

设置应用在k8s内部到容器外部的端口映射

访问设置 (Service)

Service

☒ 启用

服务访问方式

☒ 提供公网访问

☐ 仅在集群内访问

☐ VPC内网访问

☐ 主机端口访问

如何选择 [🔗](#)

自动创建公网CLB (0.02元/小时) 以提供Internet访问入口, 支持TCP/UDP协议, 如web前台类服务可以选择公网访问。
如您需要公网通过HTTP/HTTPS协议或根据URL转发, 您可以在Ingress页面使用Ingress进行路由转发, [查看详情](#) [🔗](#)

IP版本

IPv4

IPv6 NAT64

IP版本在后续更新过程中不支持变更

负载均衡器

自动创建

使用已有

自动创建CLB用于公网/内网访问Service, 请勿手动修改由TKE创建的CLB监听器, [查看更多说明](#) [🔗](#)

端口映射

协议 📘	容器端口 📘	服务端口 📘	
TCP ▼	<div>容器内应用程序监听的端口</div>	<div>建议与容器端口一致</div>	<div>✕</div>

添加端口映射

新建完成workload后, 即可启动构建

查看应用日志

腾讯云

总览

云产品

网站备案

+

容器服务

概览

集群

弹性集群

应用中心

应用

镜像仓库

应用市场

运维中心

集群运维

DevOps

← 集群(成都) / cls-2c42th7j(测试环境集群) / Deployment:cgp-app-test(default)

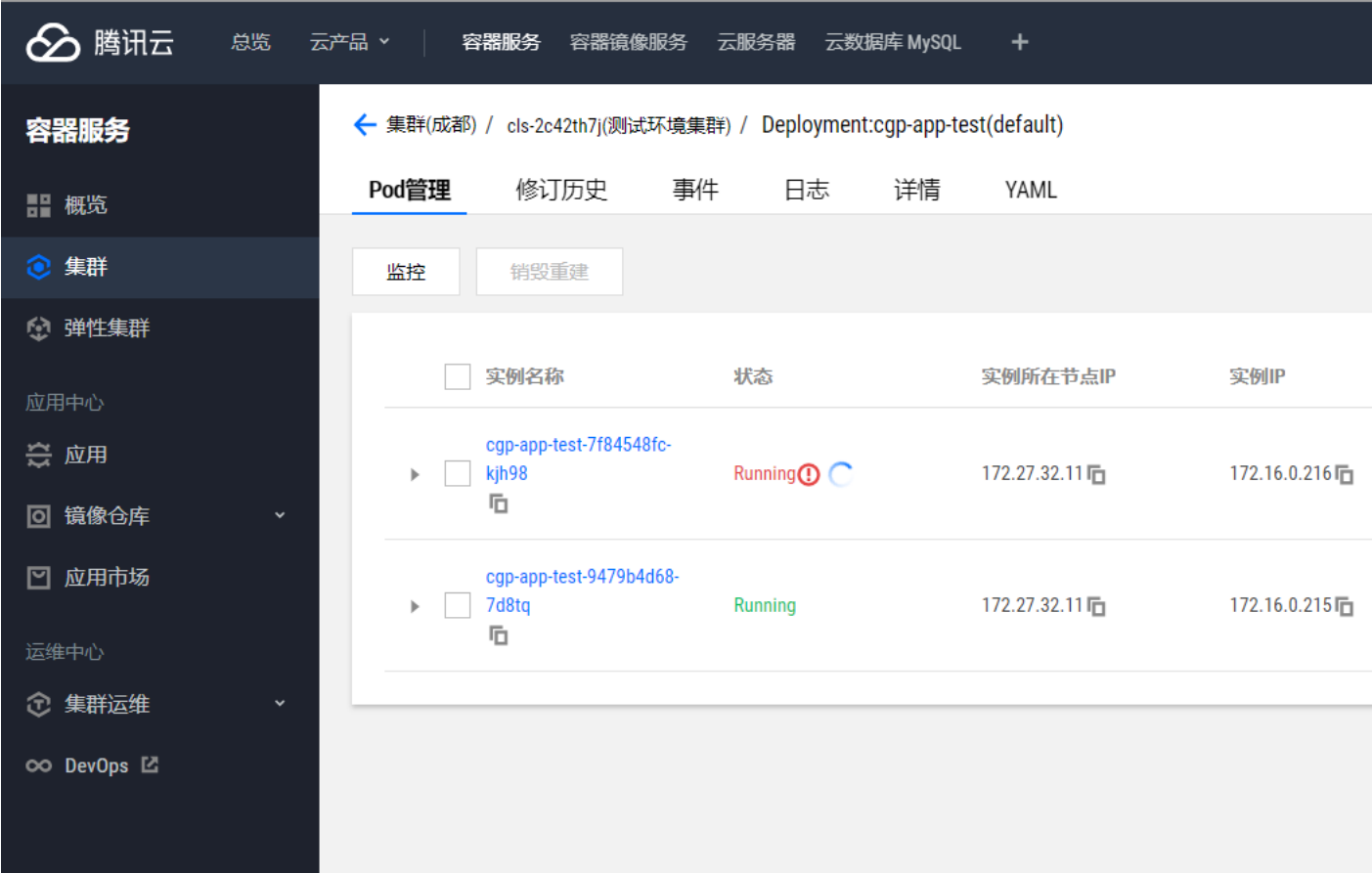
Pod管理 修订历史 事件 日志 详情 YAML

cgp-app-test-d68bbfb48-qjx7w cgp-app-test 显示100条数据

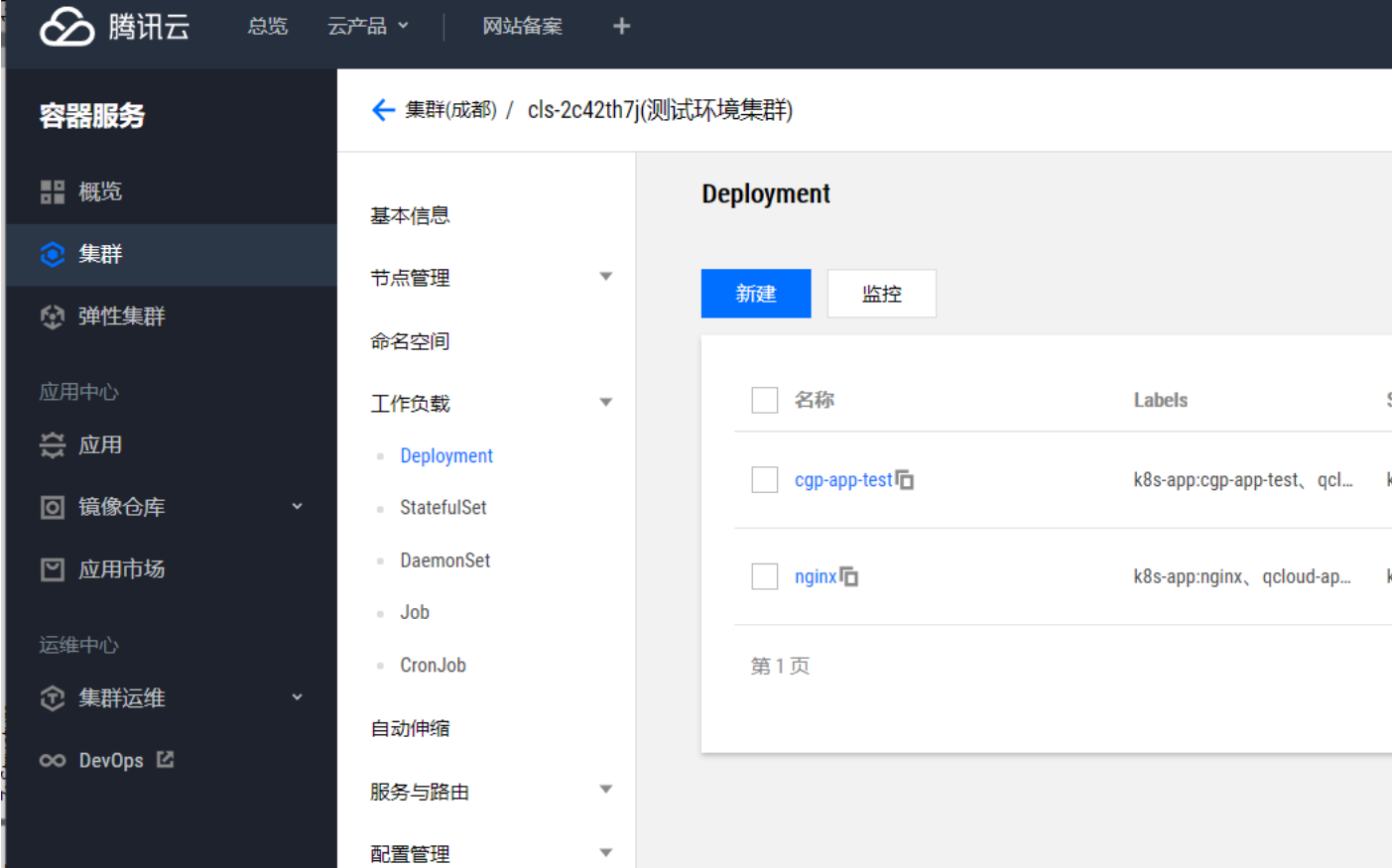
```
1 2020-09-28T11:30:03.61572785Z at org.apache.dubbo.remoting.transport.AbstractClient.com
2 2020-09-28T11:30:03.615731186Z at org.apache.dubbo.remoting.transport.AbstractClient.rec
3 2020-09-28T11:30:03.615734462Z at org.apache.dubbo.remoting.exchange.support.header.Heade
4 2020-09-28T11:30:03.615737678Z at org.apache.dubbo.remoting.exchange.support.header.Reco
5 2020-09-28T11:30:03.615740764Z at org.apache.dubbo.remoting.exchange.support.header.Abst
6 2020-09-28T11:30:03.615744982Z at org.apache.dubbo.common.timer.HashedWheelTimer$HashedW
7 2020-09-28T11:30:03.615762304Z at org.apache.dubbo.common.timer.HashedWheelTimer$HashedW
8 2020-09-28T11:30:03.615765871Z at org.apache.dubbo.common.timer.HashedWheelTimer$Worker.
9 2020-09-28T11:30:03.615768646Z at java.lang.Thread.run(Thread.java:745)
10 2020-09-28T11:31:00.001299799Z [35m2020-09-28 19:31:00.001[0;39m [34mINFO [0;39m [33m
e6997eb8757d473681b7d8af5d7b26b4 AutoConfirmOrderServiceImpl 开始获取锁key:cgp-AutoConfirm
11 2020-09-28T11:31:00.003245841Z [35m2020-09-28 19:31:00.003[0;39m [34mINFO [0;39m [33m
e6997eb8757d473681b7d8af5d7b26b4 AutoConfirmOrderServiceImpl 结束获取锁key:cgp-AutoConfirm
12 2020-09-28T11:31:00.003259466Z 操作时间:StopWatch '获取redis锁': running time = 1896449
13 2020-09-28T11:31:00.003263574Z -----
14 2020-09-28T11:31:00.003267712Z ns % Task name
15 2020-09-28T11:31:00.003270928Z -----
16 2020-09-28T11:31:00.003274374Z 001896449 100%
17 2020-09-28T11:31:00.003277731Z
18 2020-09-28T11:31:00.003322424Z [35m2020-09-28 19:31:00.003[0;39m [34mINFO [0;39m [33m
e6997eb8757d473681b7d8af5d7b26b4 AutoConfirmOrderServiceImpl --job-start-- date=:2020-0
19 2020-09-28T11:31:00.081564128Z [35m2020-09-28 19:31:00.081[0;39m [39mDEBUG[0;39m [33m
[32mcom.zto.scm.cgp.mapper.UiOrderItemsMapper.queryItemsNotConfirmed[0;39m e6997eb8757d
WHERE (`status`= 2) AND (`send_at` < DATE_ADD(NOW(), INTERVAL -2 DAY))
20 2020-09-28T11:31:00.081646383Z [35m2020-09-28 19:31:00.081[0;39m [39mDEBUG[0;39m [33m
```

进入新建的d

也可以登录容器内部查看应用日志



如何停止K8S部署的应用



参考文档

自动部署到k8s集群 (<https://help.coding.net/docs/devops/ci/deploy/k8s.html>)

标签:

在2020/09/09 13:06上被加 瓦 (/xwiki/bin/view/XWiki/jiawa)创建

