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

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

项目中新建Dockerfile文件

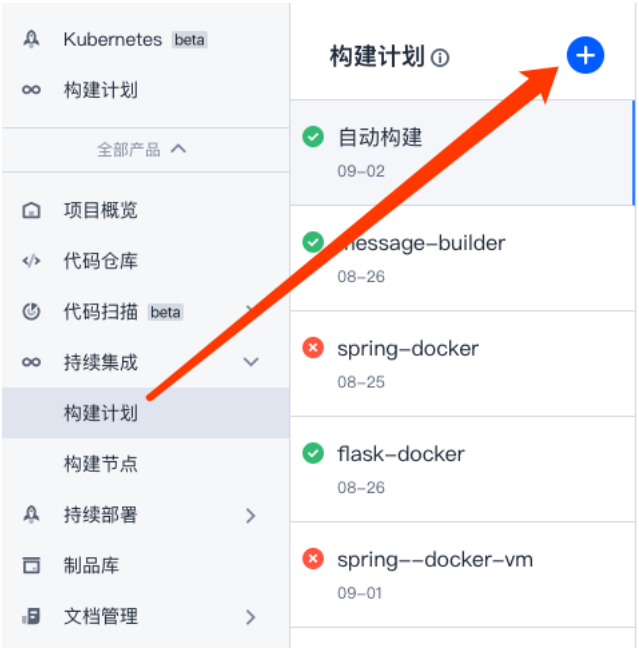
dockerfile文件用于构建项目镜像

C端小程序项目 示例如下：

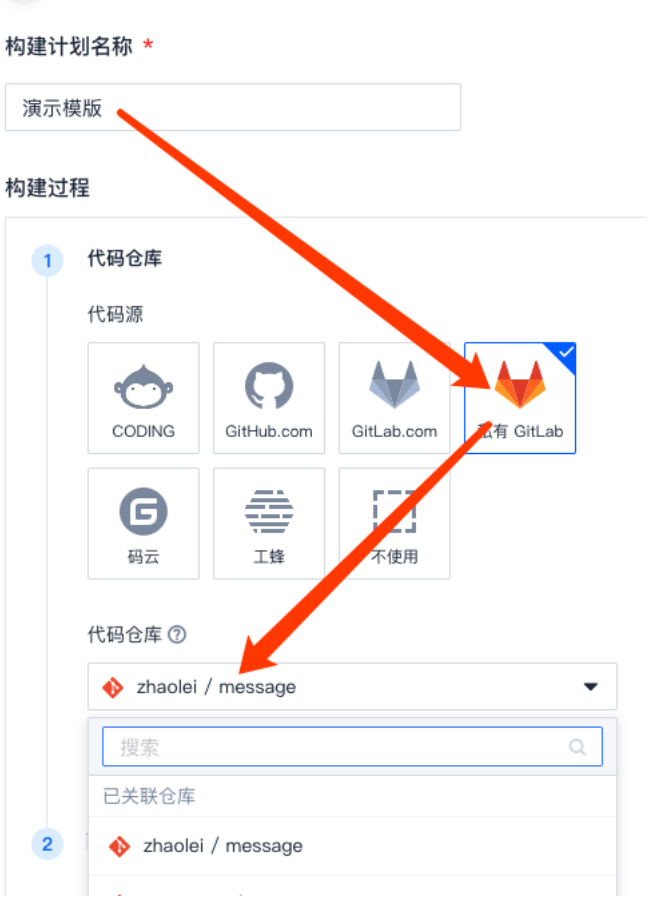
```
# 该镜像需要依赖的基础镜像
FROM java:8
# 将当前目录下的jar包复制到docker容器的/目录下
ADD cgp-app/target/cgp-app-1.0-SNAPSHOT.jar ~/mall-tiny-docker-file.jar
# 运行过程中创建一个mall-tiny-docker-file.jar文件
RUN bash -c 'touch ~/mall-tiny-docker-file.jar'
# 声明服务运行在8080端口
EXPOSE 8080
# 指定docker容器启动时运行jar包
ENTRYPOINT ["java", "-jar","~/mall-tiny-docker-file.jar","--spring.profiles.active=fat"]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

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**

REG_HOST at 61:5

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

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的名
            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

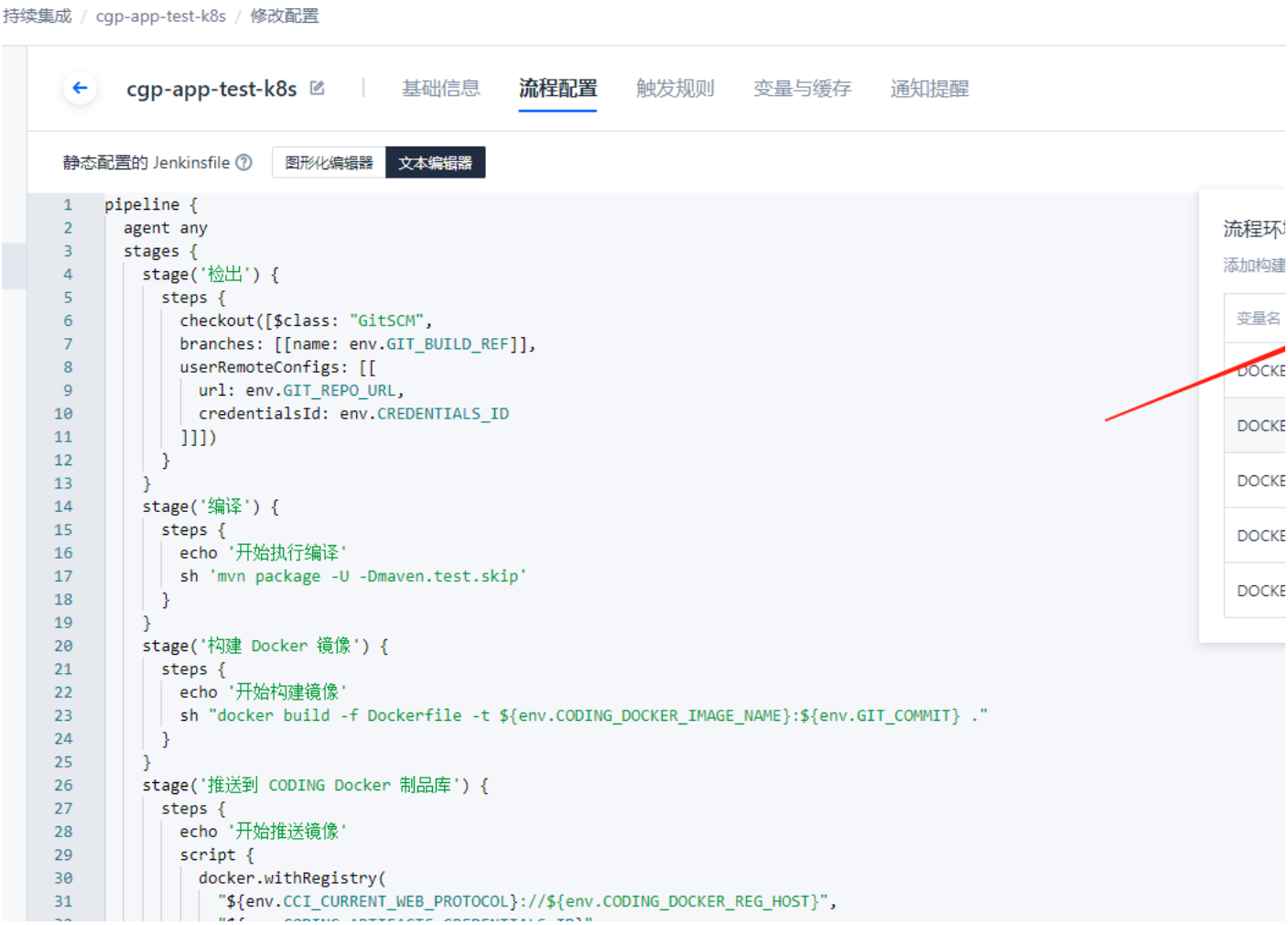
构建过程 | **构建快照** | 改动记录 | 测试报告 | 通用报告 | 构建产物

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




```
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Kerbernetes凭据相关说明，将Kubeconfig录入到CODING对应项目的凭据管理系统

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'])
```

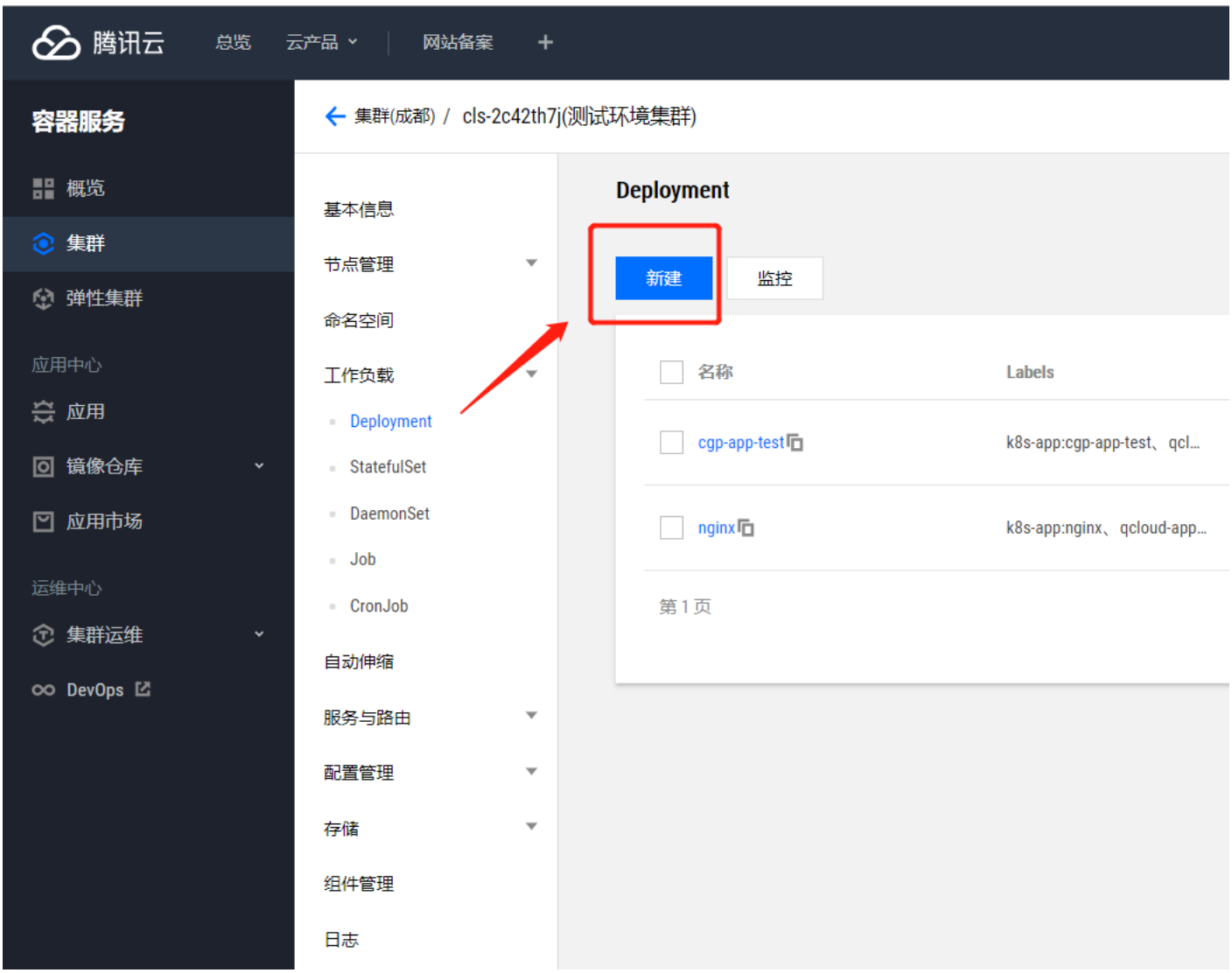
在TKE中配置目标服务

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



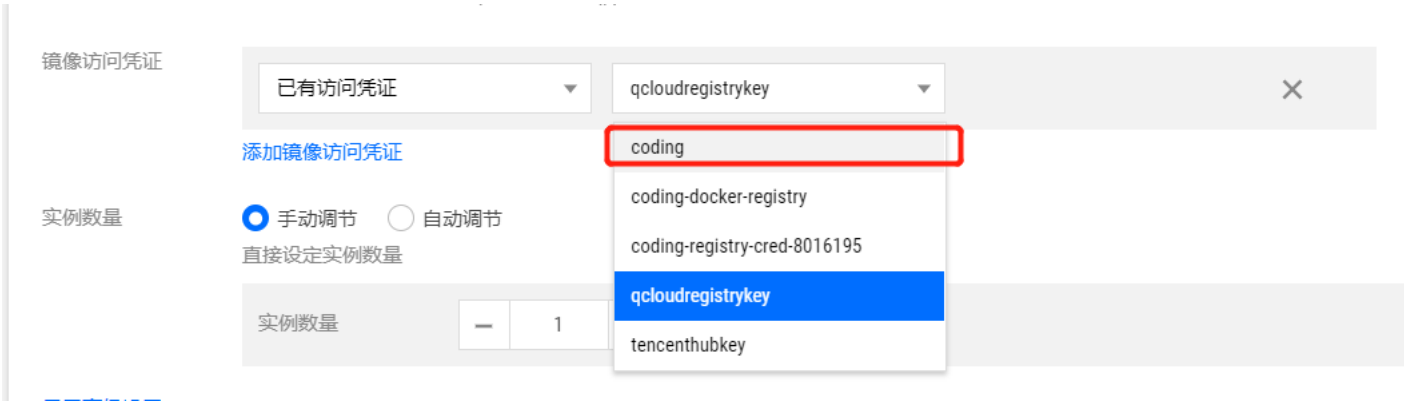
新建workload

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

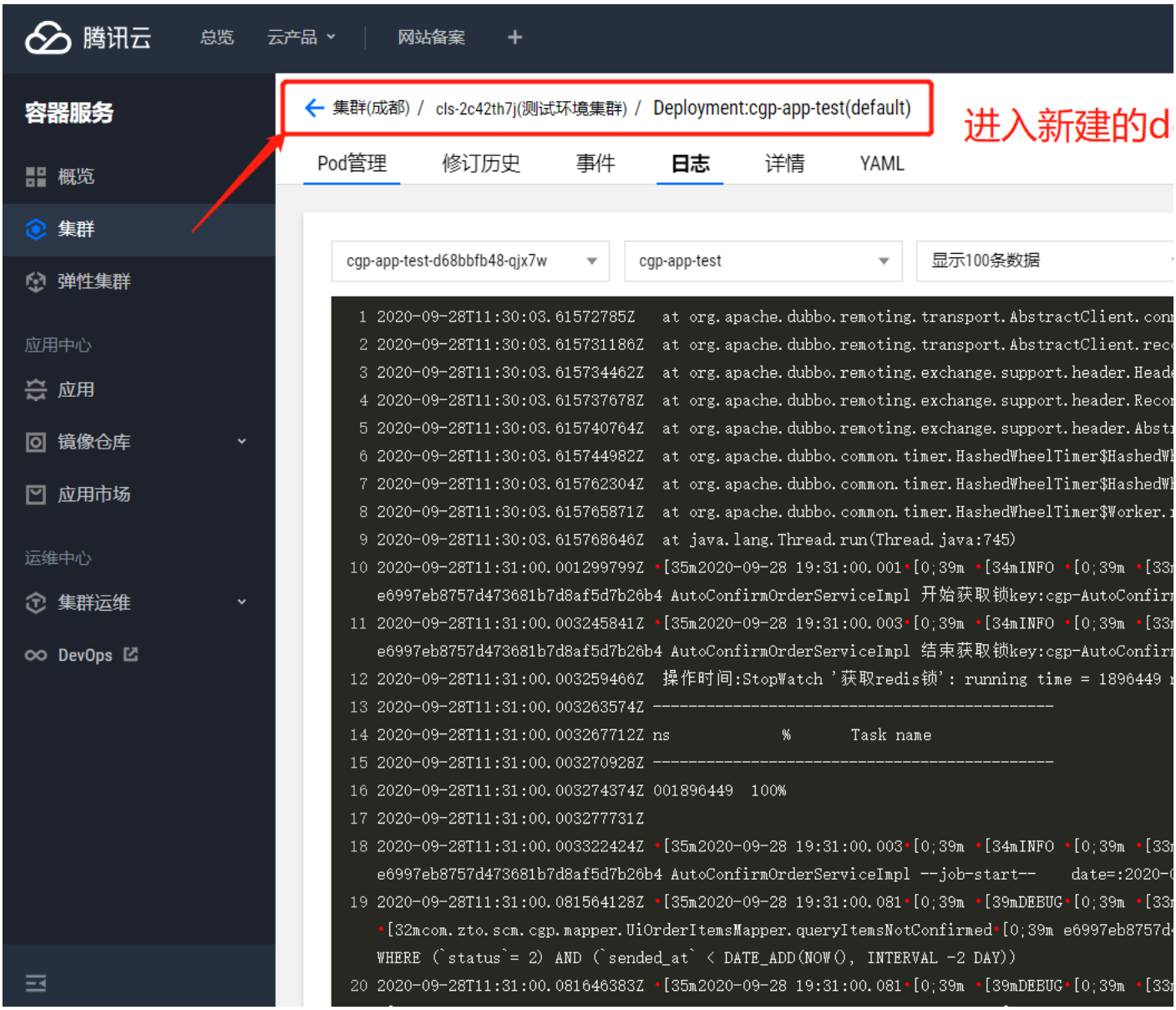
选择镜像拉取的访问凭证



设置应用在k8s内部到容器外部的端口映射

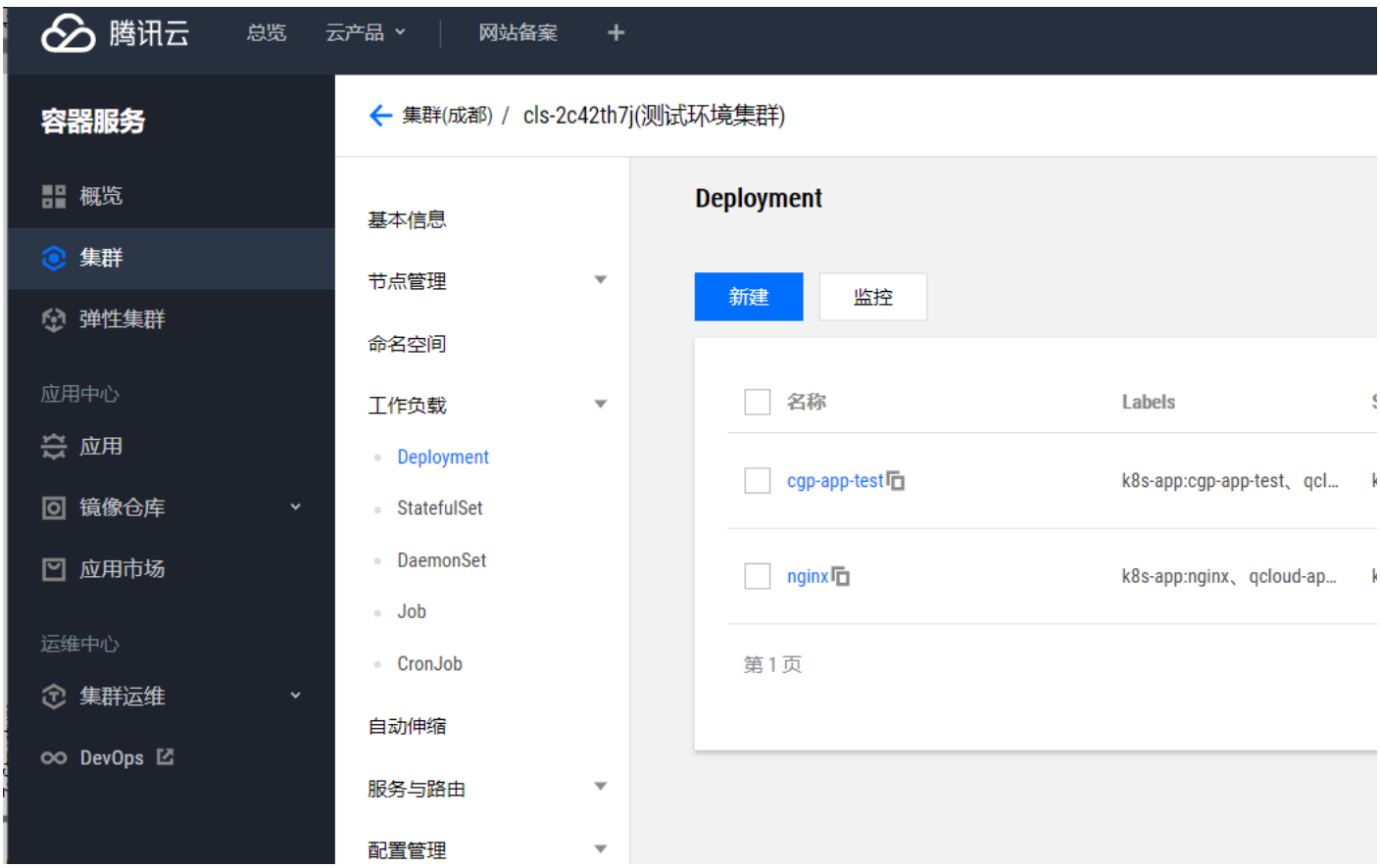
新建完成workload后，即可启动构建

查看应用日志



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

该文档没有评论