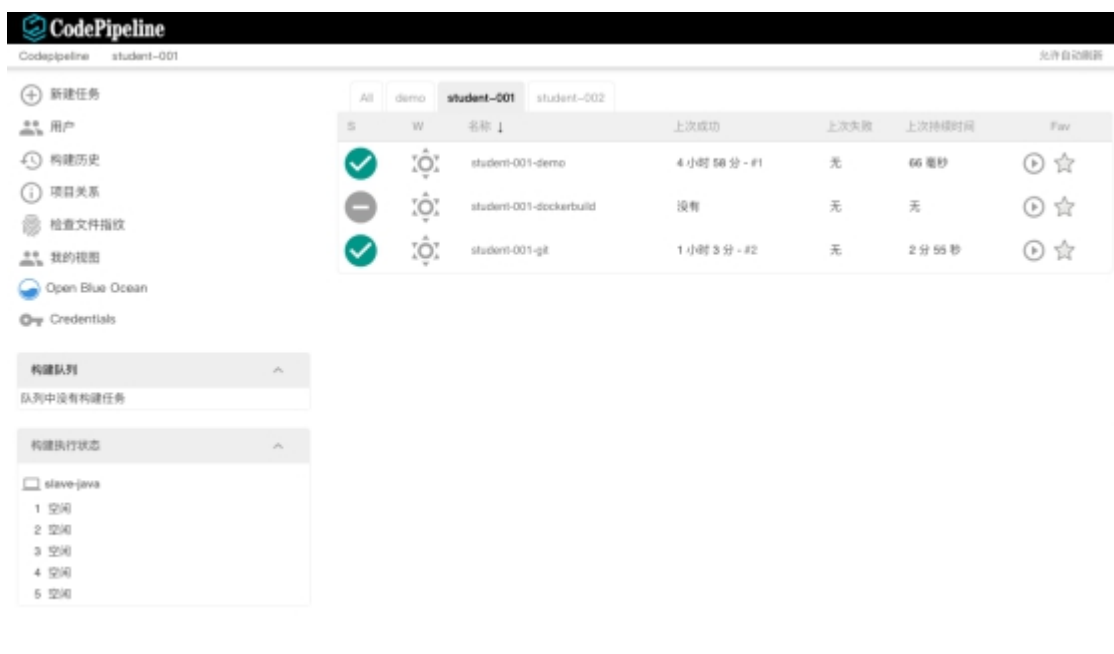


## 实验课程7-3 使用CodePipeline构建docker镜像并推送

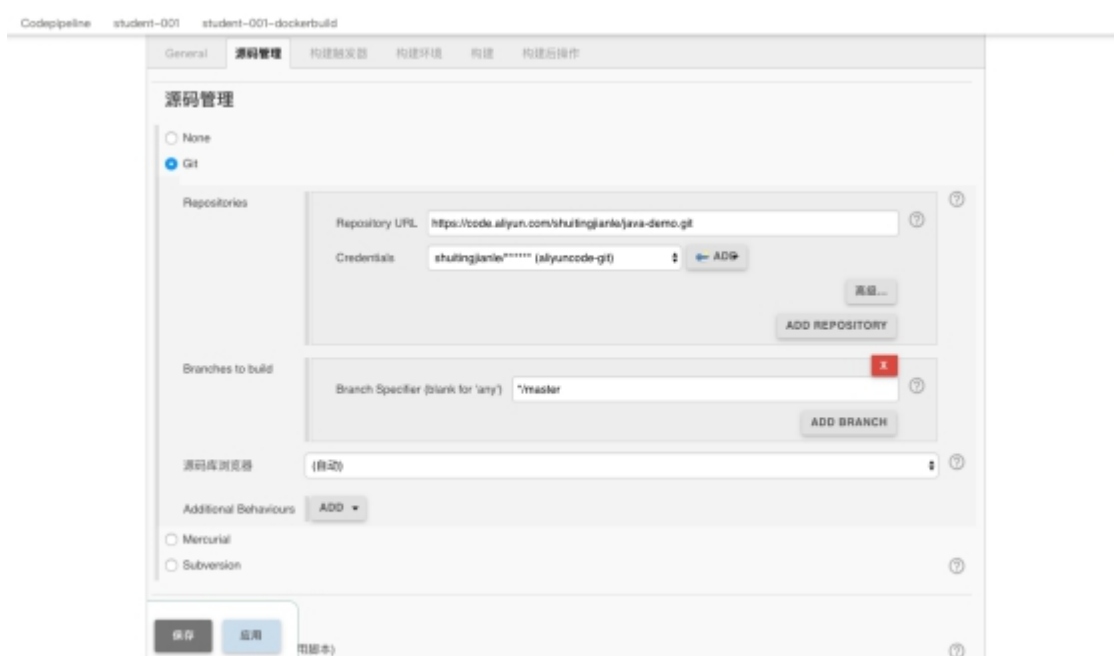
本节课程会演示如何把7-2节课程中生成的war包并打包进docker镜像，最后推送至阿里云容器镜像服务。

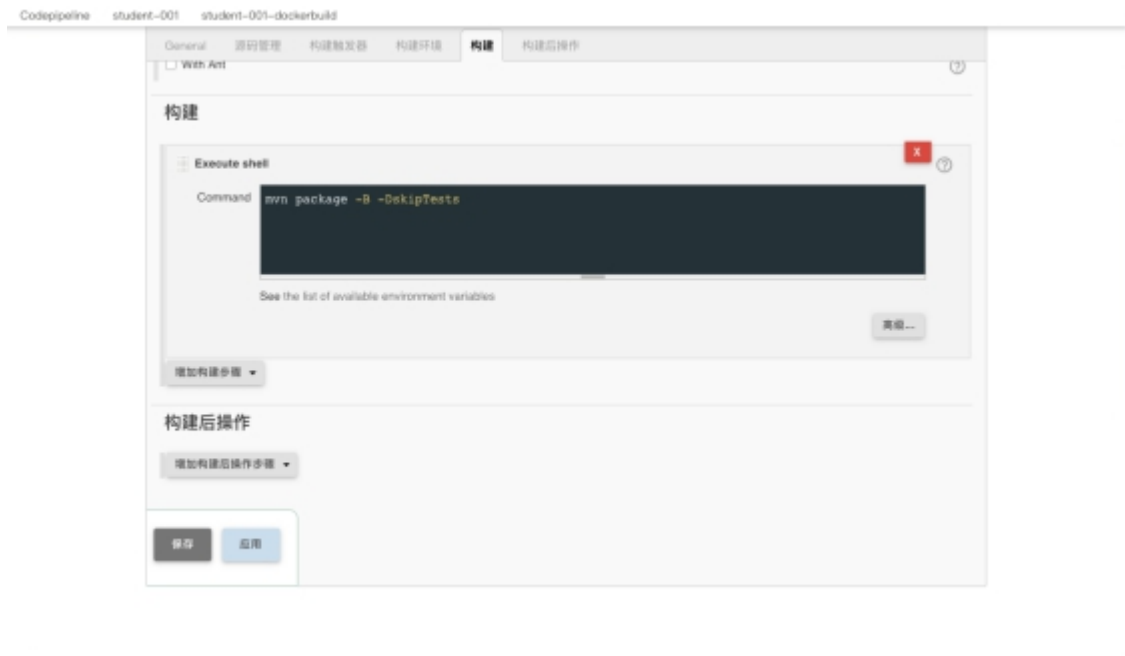
### 一、新建项目student-001-dockerbuild

#### 1、选择“新建任务”

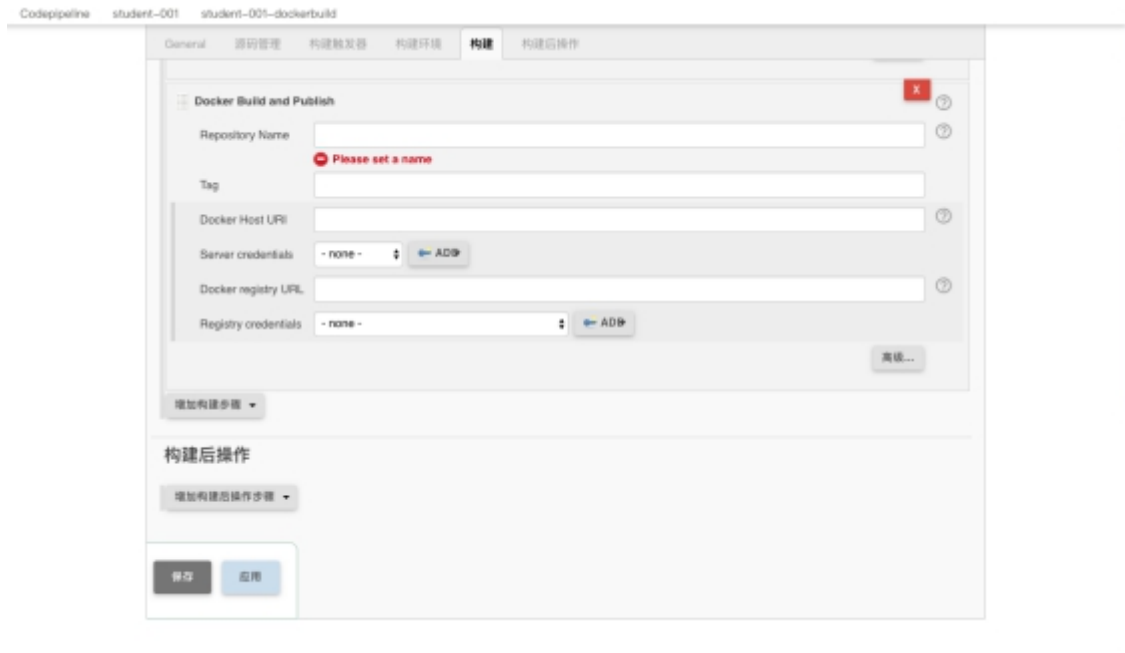


2、进入项目配置页面，配置“限制项目的运行节点”，配置拉取的源码仓库及凭证，添加mvn构建命令后：（这部分与student-001-git项目配置相同）



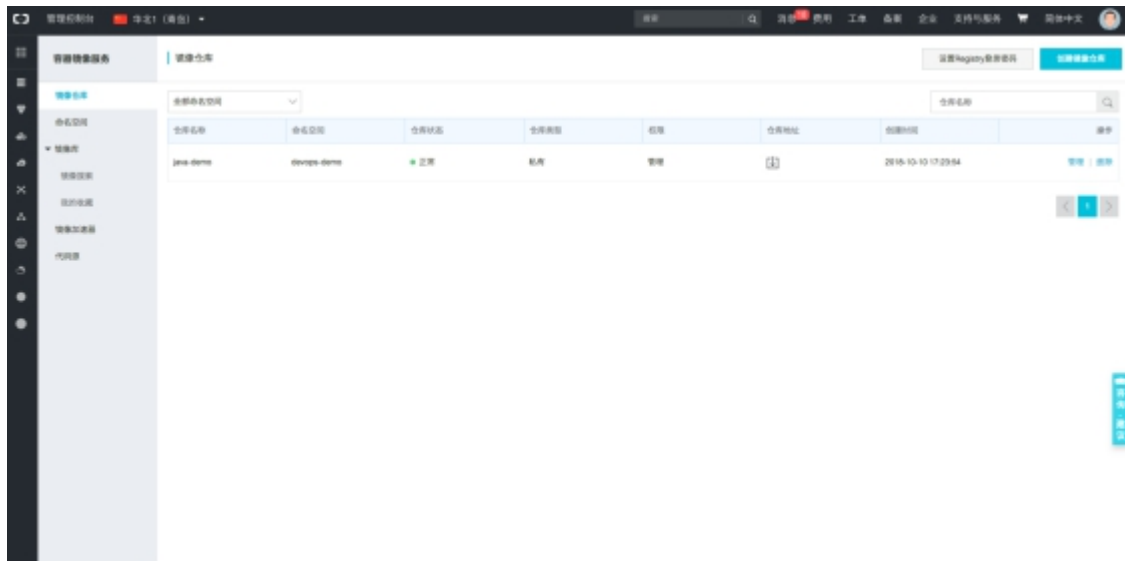


### 3、新增加构建步骤 Docker Build and Publish:



## 二、配置 Docker Build and Publish

### 1、查看容器镜像服务中我们创建过的镜像仓库信息:



## 2、配置

本示例中，我们要推送的docker镜像url为：registry.cn-qingdao.aliyuncs.com/devops-demo/java-demo:v1，则 student-001-docker build的项目配置为：

Repository Name -> devops-demo/java-demo Tag -> v1 Docker registry URL -> <https://registry.cn-qingdao.aliyuncs.com/v2/> Registry credentials -> 新建凭证

## 三、新建Docker Registry Auth类型证书

1、依次点击 ADD -> CodePipeline: Kind -> Docker Registry Auth Email -> 可选填 Username -> docker镜像仓库的用户名 Password -> docker镜像仓库的密码 ID -> 凭证ID，自定义名称

本示例为docker-registry Description -> 凭证描述，本示例为docker-registry

**Jenkins Credentials Provider: Codepipeline**

**Add Credentials**

Domain: Global credentials (unrestricted)

Kind: Docker Registry Auth

Email: [Empty field]

Username: [Redacted field]

Password: [Masked field]

ID: docker-registry

Description: docker-registry

Buttons: ADD, CANCEL

最终配置如图：

Codepipeline student-001 student-001-dockerbuild

General 源站管理 构建触发器 构建环境 构建 构建后操作

Docker Build and Publish

Repository Name

devops-demo/java-demo

Tag

v1

Docker Host URI

Server credentials

- none -

Docker registry URL

https://registry.cn-qingdao.aliyuncs.com/v2/

Registry credentials

\*\*\*\*\* (docker-registry)

高级...

增加构建步骤

构建后操作

增加构建后操作步骤

保存 应用

2、保存。

## 四、构建并查看构建日志

Codepipeline student-001 student-001-dockerbuild #1 registry.cn-qingdao.aliyuncs.com/devops-demo/java-demo:v1 registry.cn-qingdao.aliyuncs.com/devops-demo/java-demo:latest

```
0a743e17e0d0: Preparing
20448c041c0a: Preparing
185c62a48a71: Preparing
d683471ab65a: Preparing
0f25831f224d: Preparing
08a01612ffca: Preparing
8bb25f9cdc41: Preparing
f715ed19c28b: Preparing
185c62a48a71: Waiting
08a01612ffca: Waiting
d683471ab65a: Waiting
0f25831f224d: Waiting
8bb25f9cdc41: Waiting
f715ed19c28b: Waiting
0a743e17e0d0: Waiting
20448c041c0a: Waiting
18bbfbcc62ab: Layer already exists
9881f52bcab: Layer already exists
199f39ef28fb: Layer already exists
d277f245ccfb: Layer already exists
ea8ed57aee48: Layer already exists
185c62a48a71: Layer already exists
0a743e17e0d0: Layer already exists
0f25831f224d: Layer already exists
d683471ab65a: Layer already exists
20448c041c0a: Layer already exists
f715ed19c28b: Layer already exists
8bb25f9cdc41: Layer already exists
08a01612ffca: Layer already exists
latest: digest: sha256:c3b70c7fc709e86f80d3a4784e772889d87034ab61923b47d2ac2957820962d size: 3044
Finished: SUCCESS
```

## 五、查看镜像仓库中镜像版本

<

java-demo

第 1 页 | 包名 | 本地仓库 | 正常

部署包

基本信息

仓库地址

Webhook

部署版本

部署同步

部署版本

版本	镜像ID	状态	Digest	镜像大小	最后更新时间	操作
latest	4b9e1a030e...	正常	c3811a761780b48f80d3a0764a772889487234a481823b47c8ec2951f920963d	185.719 MB	2018-10-31 17:50:15	安全扫描   镜像源   同步   删除
v1	4b9e1a030e...	正常	c3811a761780b48f80d3a0764a772889487234a481823b47c8ec2951f920963d	185.719 MB	2018-10-31 17:50:14	安全扫描   镜像源   同步   删除

< 1 >

部署包

####