

jenkins与k8s的集成

前提环境：已经存在有一个k8s集群，再另一台虚拟机上操作jenkins.

操作jenkins的这台虚拟机，要安装好docker，版本要k8s集群里的docker版本一致，这里是1.14.3。

具体操作可以参考网址：<https://www.kubernetes.org.cn/5462.html>，步骤到安装完docker结束。

docker的方式部署jenkins

- 下载jenkins镜像

```
docker pull jenkinsci/blueocean
```

- 启动镜像，注意/home/jenkins的路径是宿主机映射的路径，jenkins文件夹需要手动创建。具体可以看官网：<https://jenkins.io/zh/doc/book/installing/>

```
docker run -u root -d -p 8080:8080 -p 50000:50000 -v /home/jenkins:/var/jenkins_home -v /var/run/docker.sock:/var/run/docker.sock jenkinsci/blueocean
```

- 网页打开jenkins,虚拟机ip加端口，例如：<http://192.168.233.131:8080>



()

获取密码，再刚创建的宿主机的jenkins文件夹下查看：

```
cat jenkins/secrets/initialAdminPassword
```

解锁之后，再页面选择“安装建议插件”

最后设置一下管理员，注意填上邮箱。

- 在进入jenkins管理页面，系统管理-》插件管理-》高级，将底部的升级站点换成国内的,提交保存

`https://mirrors.tuna.tsinghua.edu.cn/jenkins/updates/update-center.json`

升级站点

URL `https://mirrors.tuna.tsinghua.edu.cn/jenkins/updates/update-center.json`

提交

23 小时 之前获取了更新信息

立即获取

- 在可选插件的地方搜索kubernetes进行自动安装。

创建一个boot项目发到github上

注意这里项目的端口与后面容器暴露出去的端口一致。

添加Dockerfile，用来构建docker镜像

- 在项目的根目录下添加Dockerfile:

```
FROM harbor.olavoice.com/library/oracle/serverjre:8
ARG JAR_FILE
ARG WORK_PATH="/opt/demo"
ENV JAVA_OPTS="" \
    JAR_FILE=${JAR_FILE}
#设置时区
RUN ln -sf /usr/share/zoneinfo/Asia/Shanghai /etc/localtime && \
    echo "Asia/Shanghai" > /etc/timezone
COPY target/$JAR_FILE $WORK_PATH/
WORKDIR $WORK_PATH
ENTRYPOINT exec java $JAVA_OPTS -jar $JAR_FILE
```

FROM 后面是基础镜像，来源的是私有镜像仓库的jdk镜像。

其中\${JAR_FILE}参数在pipeline执行docker build时，通过build-arg参数传入。

添加k8s-deployment.tpl 和 jenkinsfile

- 两个文件均创建在目录下，k8s-deployment.tpl文件，此文件用来作为k8s的yaml文件模板，k8s-deployment.tpl内容如下：

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {APP_NAME}-deployment
  labels:
    app: {APP_NAME}
spec:
  replicas: 1
  selector:
    matchLabels:
```

```

    app: {APP_NAME}
template:
  metadata:
    labels:
      app: {APP_NAME}
  spec:
    imagePullSecrets:
      - name: regcred
    containers:
      - name: {APP_NAME}
        image: {IMAGE_URL}:{IMAGE_TAG}
        ports:
          - containerPort: 8082

```

注意containerPort的端口要与项目的端口一致。

- jenkinsfile文件内容如下：

```

pipeline {
    agent any
    # 在jenkins中配置的凭证
    environment {
        HARBOR_CREDS = credentials('jenkins-harbor')
        K8S_CONFIG = credentials('jenkins-k8s-config')
    }

    parameters {
        string(name: 'HARBOR_HOST', defaultValue: 'harbor.olavoice.com',
description: 'harbor仓库地址')
        string(name: 'DOCKER_IMAGE', defaultValue: 'demo-master', description:
'docker镜像名')
        string(name: 'APP_NAME', defaultValue: 'demo-master', description: 'k8s
中标签名')
        string(name: 'K8S_NAMESPACE', defaultValue: 'default', description:
'k8s的namespace名称')
        string(name: 'DOCKER_IMAGE_TAG', defaultValue: '1.0', description: '镜像
版本')
    }

    stages {
        # 拉取代码，打成jar包
        stage('Maven Build') {
            agent {
                docker {
                    image 'maven:3-jdk-8-alpine'
                    args '-v $HOME/.m2:/root/.m2'
                }
            }
            steps {
                sh 'mvn clean package -Dfile.encoding=UTF-8 -DskipTests=true'
                stash includes: 'target/*.jar', name: 'demo'
            }
        }
    }
}

```

```

    }
    # 将jar包打包成镜像推送到私有镜像仓库
    stage('Docker Build') {
        agent any
        steps {
            unstash 'demo'
            sh "docker login -u ${HARBOR_CREDS_USR} -p ${HARBOR_CREDS_PSW}
${params.HARBOR_HOST}"
            sh "docker build --build-arg JAR_FILE=`ls target/*.jar |cut -d '/' -
f2` -t ${params.HARBOR_HOST}/test/${params.DOCKER_IMAGE}:${params.DOCKER_IMAGE_TAG} ."
            sh "docker push
${params.HARBOR_HOST}/test/${params.DOCKER_IMAGE}:${params.DOCKER_IMAGE_TAG}"
            sh "docker rmi
${params.HARBOR_HOST}/test/${params.DOCKER_IMAGE}:${params.DOCKER_IMAGE_TAG}"
        }
    }

    #拉取kubect1镜像，将在jenkins中添加凭证的内容，base64解码一下，复制到kubect1镜像容器中
    #将k8s-deployment.tpl写入k8s-deployment.yml，将变量替换
    stage('Deploy') {
        agent {
            docker {
                image 'lwolf/helm-kubect1-docker'
            }
        }
        steps {
            sh "mkdir -p ~/.kube"
            sh "echo ${K8S_CONFIG} | base64 -d > ~/.kube/config"
            sh "sed -e 's#
${IMAGE_URL}##${params.HARBOR_HOST}/test/${params.DOCKER_IMAGE}#g;s#
${IMAGE_TAG}##${params.DOCKER_IMAGE_TAG}#g;s#${APP_NAME}##${params.APP_NAME}#g' k8s-
deployment.tpl > k8s-deployment.yml"
            sh "kubect1 apply -f k8s-deployment.yml --
namespace=${params.K8S_NAMESPACE}"
        }
    }
}
}
}

```

- stages说明：

- Maven Build：使用docker的方式执行maven命令，args参数中将.m2目录映射出来
- Docker Build：通过sh依次执行docker命令登录harbor、构建镜像、上传镜像、移除本地镜像。构建镜像时，会获取jar文件名传入JAR_FILE参数。
- Deploy：使用docker的方式执行kubect1命令。先pull kubect1的镜像，将jenkins的jenkins-k8s-config凭证中的加密内容base64解密并存储为~/.kube/config配置文件，然后将k8s-deployment.tpl文件中形式参数替换为实际的参数值，最后kubect1命令部署至k8s。

- 凭证-》系统-》全局凭证-》添加凭证 分别添加github上的用户名密码，harbor上的用户名密码，最后是jenkins到k8s的凭证。

- 填上你自己的github上的用户名密码

类型	Username with password
范围	全局 (Jenkins, nodes, items, all child items, etc)
用户名	
密码	
ID	
描述	

确定

- 配置私有镜像仓库的用户名密码

注意这里起一个ID,这里的ID是与jenkinsfile里的environment设置的“jenkins-harbor”保持一致，为了能引用到。

- 配置jenkins到k8s的凭证

添加的类型是secret text

类型	Username with password
	Username with password
	Docker Host Certificate Authentication
	OpenShift OAuth token
	OpenShift Username and Password
	SSH Username with private key
	Secret file
	Secret text
	Certificate
ID	

注意Secret填写的是k8s master节点下/root/.kube/config文件内容，需要base64加密一下，因为config文件是一个yaml格式的文件，如果不加密，会在复制的时候将空格默认去除。


新建jenkins任务


- 点击新建任务，填写任务名称，选择流水线，之后按确定。


输入一个任务名称


demo3


» 必填项


 **构建一个自由风格的软件项目**
这是Jenkins的主要功能。Jenkins将会结合任何SCM和任何构建系统来构建你的项目，甚至可以构建软件以外的系统。


 **流水线**
精心地组织一个可以长期运行在多个节点上的任务。适用于构建流水线（更加正式地应当称为工作流），增加或者组织难以采用自由风格的任务类型。

 **构建一个多配置项目**
适用于多配置项目，例如多环境测试，平台指定构建，等等。


 **Bitbucket Team/Project**
Scans a Bitbucket Cloud Team (or Bitbucket Server Project) for all repositories matching some defined markers.

 **文件夹**
创建一个可以嵌套存储的容器。利用它可以进行分组。视图仅仅是一个过滤器，而文件夹则是一个独立的命名空间，因此你可以有多个相同名称的内容，只要它们在不同的文

 **GitHub 组织**
扫描一个 GitHub 组织（或者个人账户）的所有仓库来匹配已定义的标记。

 **多分支流水线**
根据一个SCM仓库中检测到的分支创建一系列流水线。

如果你想根据一个已经存在的任务创建，可以使用这个选项

 复制

确定

- 在流水线定义选项中选择Pipeline script from SCM, SCM选项选择Git, Repository URL填写代码仓库的地址, Credentials选择之前设置的github仓库凭证。

流水线

定义 Pipeline script from SCM

SCM Git

Repositories

Repository URL

Please enter Git repository.

Credentials - 无 - 添加

高级...

Add Repository

Branches to build

Branch Specifier (blank for 'any') */master

Add Branch

源代码浏览器 (自动)

Additional Behaviours 新增

脚本路径 Jenkinsfile

轻量级检出 ☒

[流水线语法](#)

保存 应用

点击保存。

打开Blue Ocean,运行刚才创建的流水线任务。

部分地方还可以参考：<https://gitee.com/tinyk/pipeline-demo>