**迭代二Jenkins文档**

**注意事项**

由于迭代二增加了Python服务，对服务器造成了沉重负担（ 镜像大小4G， 服务器仅剩20G磁盘空间 ）， 再加上服务器性能本就有限， 请**不要同时构建两个项目**。

且前端和Python服务的构建非常吃资源，一次build就可能让服务器卡死，甚至Jenkins容器退出。 因此如果**遇到Jenkins卡死等情况，请联系我们组成员**， 不要认为这是我们cicd没做好qaq

**文档信息**

**小组信息**

|  |  |  |
| --- | --- | --- |
| 姓名 | 电子邮箱 | QQ |
| 李大磊 | 191250067@smail.nju.edu.cn | 1945652814 |
| 曹璐 | 191250003@smail.nju.edu.cn | 1534972955 |
| 陆昱宽 | 191820133[@smail.nju.edu.cn](mailto:191830106@smail.nju.edu.cn" \t "_blank) | 1037400635 |
| 熊智星 | 191250162@smail.nju.edu.cn | 1731393946 |

**基本信息**

|  |  |  |  |
| --- | --- | --- | --- |
| 创建者 | 文档管理者 | 文档使用者 | 文档创建日期 |
| 陆昱宽 | 陆昱宽 | lyk,李大磊，熊智星，曹璐 | 2022年3月5日 |

**修改记录**

|  |  |  |
| --- | --- | --- |
| 日期 | 修改人 | 改动 |
| 2022年3月5日 | 陆昱宽 | 初始化文档 |
| 2022年4月1日 | 陆昱宽 | 修改文档，增添迭代二部分内容 |
|  |  |  |

**简介**

我们的Jenkins有三个Item, 前后端和算法服务各一个。因此有三份Jenkinsfile.

**agent**

我们通过ssh连接了另一台服务器作为Jenkins节点，指挥该节点进行构建。 该节点已经安装了构建所需的环境，比如对应版本的maven, jdk, node, npm, vue等。 整个脚本都在该节点执行

**docker**

我们的项目全部打包成docker容器，push到镜像仓库进行管理。 要运行服务，只需要pull镜像并且启动

**执行逻辑**

前后端的执行逻辑都相同:

1. 从git仓库clone代码
2. cd进入工作目录
3. 预处理， 比如前端需要npm install下载依赖
4. 进行项目构建：

* 后端： maven clean package
* 前端： npm run build

1. 将构建后的文件打成docker镜像
2. 登录Dockerhub
3. 将镜像push到Dockerhub
4. 从Dockerhub pull镜像
5. 运行镜像

算法部分的执行逻辑略有不同， 因为Python项目要依赖各种库，打包后镜像太大了（有4.5G）， 每次CICD时，这么大的镜像要push到Dockerhub，再pull下来，是巨大的性能瓶颈。 因此算法部分的pipeline脚本，没有了push和pull镜像的操作

同时，对于Python服务，需要专门指定一个`requirement.txt`文件来描述依赖，该文件会被pip3读取（ 这一步在Dockerfile中 ），来下载依赖

**后端脚本**

node("slave1") {

def workspace = pwd()

def git\_branch = 'master'

def git\_repository = 'git@git.nju.edu.cn:191820133/backend-volatile.git'

def vm\_ip = '124.222.135.47'

def vm\_port = '22'

def vm\_user = 'lyk'

def vm\_project\_place = "/usr/local/src"

def vm\_target\_place = "/usr/local/src/target/"

def IMAGE\_NAME = 'volatile\_backend'

def IMAGE\_NAME\_WITH\_TAG = 'volatile\_backend:latest'

def IMAGE\_TO\_RUN = 'lyklove/volatile\_backend:latest'

def CONTAINER\_NAME = 'volatile\_backend'

stage('clone from gitlab into slave\'s workspace') {

echo "workspace: ${workspace}"

git branch: "${git\_branch}", url: "${git\_repository}"

}

stage('cd to build context') {

echo "the context now is:"

sh "ls -al"

sh "cd ${workspace}"

echo "cd to build context, now the context is:"

sh "ls -al"

}

stage('build jar on slave machine, jacoco file generated') {

sh 'mvn --version'

sh "mvn clean package jacoco:report -Dmaven.test.failure.ignore=true"

echo "build finish on ${vm\_ip}"

}

stage( 'testing, using jacoco' ) {

jacoco (

execPattern: '\*\*/target/jacoco.exec',

classPattern: '\*\*/classes',

sourcePattern: '\*\*/src/main/java',

exclusionPattern: '\*\*/src/test\*',

// inclusionPattern: '\*\*/com/hel/auto/service/\*.class,\*\*/com/hel/auto/controller/\*.class',

)

}

stage("build docker image"){

sh "docker build -t ${IMAGE\_NAME} ."

}

stage("login to dockerhub"){

withCredentials([usernamePassword(credentialsId: 'DOCKERHUB\_KEY', passwordVariable: 'password', usernameVariable: 'username')]) {

sh 'docker login -u $username -p $password'

}

}

stage("push to dockerhub"){

echo "begin push to dockerhub"

sh "docker image tag ${IMAGE\_NAME\_WITH\_TAG} lyklove/${IMAGE\_NAME\_WITH\_TAG}"

sh "docker image push lyklove/${IMAGE\_NAME\_WITH\_TAG}"

}

stage("clean previous image and container"){

sh "docker container rm -f ${CONTAINER\_NAME}"

sh "docker image rm ${IMAGE\_NAME\_WITH\_TAG}"

sh "docker image rm ${IMAGE\_TO\_RUN}"

}

stage( "pull image" ){

sh "docker pull lyklove/${IMAGE\_NAME\_WITH\_TAG}"

}

stage("run container") {

sh "docker image ls"

sh "docker container run --name ${CONTAINER\_NAME} --net=host -d ${IMAGE\_TO\_RUN}"

}

stage("signal gitlab: deployed"){

updateGitlabCommitStatus name: 'deployed', state: 'success'

}

}

**前端脚本**

node("slave1") {

def workspace = pwd()

def git\_branch = 'master'

def git\_repository = 'git@git.nju.edu.cn:191820133/frontend-volatile.git'

def vm\_ip = '124.222.135.47'

def vm\_port = '22'

def vm\_user = 'lyk'

def vm\_project\_place = "/usr/local/src"

def vm\_target\_place = "/usr/local/src/target/"

def IMAGE\_NAME = 'volatile\_frontend'

def IMAGE\_NAME\_WITH\_TAG = 'volatile\_frontend:latest'

def IMAGE\_TO\_RUN = 'lyklove/volatile\_frontend:latest'

def CONTAINER\_NAME = 'volatile\_frontend'

stage('clone from gitlab into slave\'s workspace') {

echo "workspace: ${workspace}"

git branch: "${git\_branch}", url: "${git\_repository}"

}

stage('cd to build context') {

echo "the context now is:"

sh "ls -al"

sh "cd ${workspace}"

echo "cd to build context, now the context is:"

sh "ls -al"

}

stage('get version info'){

sh 'node -v'

sh 'npm -v'

sh 'vue -V'

sh 'npm list vue'

}

stage('build with npm') {

// sh 'npm config set registry http://registry.cnpmjs.org'

sh 'npm install'

sh 'npm run build'

echo "build finish on ${vm\_ip}"

}

// stage('npm run serve'){

//

// echo 'not using docker yet!!'

// sh 'npm run serve'

// }

stage("build docker image"){

sh "docker build -t ${IMAGE\_NAME} --no-cache ."

// sh "imageId=`docker images | grep #{IMAGE\_NAME} | awk '{print $3}'`"

}

stage("login to dockerhub"){

withCredentials([usernamePassword(credentialsId: 'DOCKERHUB\_KEY', passwordVariable: 'password', usernameVariable: 'username')]) {

sh 'docker login -u $username -p $password'

}

}

stage("push to dockerhub"){

echo "begin push to dockerhub"

sh "docker image tag ${IMAGE\_NAME\_WITH\_TAG} lyklove/${IMAGE\_NAME\_WITH\_TAG}"

sh "docker image push lyklove/${IMAGE\_NAME\_WITH\_TAG}"

}

stage("clean previous image and container"){

sh "docker container rm -f ${CONTAINER\_NAME}"

sh "docker image rm ${IMAGE\_NAME\_WITH\_TAG}"

sh "docker image rm ${IMAGE\_TO\_RUN}"

}

stage( "pull image" ){

sh "docker pull lyklove/${IMAGE\_NAME\_WITH\_TAG}"

}

stage("run container") {

sh "docker image ls"

sh "docker container run --name ${CONTAINER\_NAME} --net=host -d ${IMAGE\_TO\_RUN}"

}

stage("signal gitlab: deployed"){

updateGitlabCommitStatus name: 'deployed', state: 'success'

}

}

**算法脚本**

node("slave1") {

def workspace = pwd()

def git\_branch = 'master'

def git\_repository = 'git@git.nju.edu.cn:191820133/ai-volatile.git'

def vm\_ip = '124.222.135.47'

def vm\_port = '22'

def vm\_user = 'lyk'

def vm\_project\_place = "/usr/local/src"

def vm\_target\_place = "/usr/local/src/target/"

def IMAGE\_NAME = 'volatile\_ai'

def IMAGE\_NAME\_WITH\_TAG = 'volatile\_ai:latest'

def IMAGE\_TO\_RUN = 'lyklove/volatile\_ai:latest'

def CONTAINER\_NAME = 'volatile\_ai'

stage('clone from gitlab into slave\'s workspace') {

echo "workspace: ${workspace}"

git branch: "${git\_branch}", url: "${git\_repository}"

}

stage('cd to build context') {

echo "the context now is:"

sh "ls -al"

sh "cd ${workspace}"

echo "cd to build context, now the context is:"

sh "ls -al"

}

stage("build docker image"){

sh "docker build -t ${IMAGE\_NAME} ."

}

// stage("login to dockerhub"){

// withCredentials([usernamePassword(credentialsId: 'DOCKERHUB\_KEY', passwordVariable: 'password', usernameVariable: 'username')]) {

// sh 'docker login -u $username -p $password'

// }

// }

//

stage("push to dockerhub"){

// echo "begin push to dockerhub"

sh "docker image tag ${IMAGE\_NAME\_WITH\_TAG} lyklove/${IMAGE\_NAME\_WITH\_TAG}"

// sh "docker image push lyklove/${IMAGE\_NAME\_WITH\_TAG}"

}

stage("clean previous image and container"){

sh "docker container rm -f ${CONTAINER\_NAME}"

// sh "docker image rm ${IMAGE\_NAME\_WITH\_TAG}"

// sh "docker image rm ${IMAGE\_TO\_RUN}"

}

// stage( "pull image" ){

// sh "docker pull lyklove/${IMAGE\_NAME\_WITH\_TAG}"

// }

stage("run container") {

sh "docker image ls"

sh "docker container run --name ${CONTAINER\_NAME} --net=host -d ${IMAGE\_TO\_RUN}"

}

stage("signal gitlab: deployed"){

updateGitlabCommitStatus name: 'deployed', state: 'success'

}

}