部署文档

变更记录

修改人员	日期	变更原因	版本号
刘育麟	2021.6.18	创建文档	v1.0

部署过程

流程

1. push到release分支
2. Jenkins侦测GitLab release分支改变，在改变时自动构建。
3. 构建经过JenkinsFile写的流水线，这个流水线主要分为几步
    1. Jenkins在的服务器clone项目并且进行maven的打包
    2. 使用maven的插件jacoco进行测试并生成测试报告
    3. Jenkins在的服务器将项目用docker打包成镜像并push到harbor上面
    4. 另一台服务器对这个服务器的harbor进行pull
    5. 另一台服务器运行该镜像

Dockerfile

```
FROM openjdk:8-jre-alpine

RUN ln -sf /usr/share/zoneinfo/Asia/Shanghai /etc/localtime
RUN echo 'Asia/Shanghai' >/etc/timezone

ENV JAVA_OPTS ''

WORKDIR /app
ADD target/IRBL-0.0.1-SNAPSHOT.jar .

ENTRYPOINT ["sh", "-c", "set -e && java -XX:+PrintFlagsFinal \
                                        -XX:+HeapDumpOnOutOfMemoryError \
                                        -XX:HeapDumpPath=/heapdump/heapdump.hprof \
                                        -XX:+UnlockExperimentalVMOptions \
                                        -XX:+UseCGroupMemoryLimitForHeap \
                                        $JAVA_OPTS -jar IRBL-0.0.1-SNAPSHOT.jar"]

EXPOSE 40000
```

使用jdk8版本，并且将40000端口作为后端的端口暴露

JenkinsFile

```
def GetRemoteServer(ip){
    def remote = [:]
    remote.name = 'remoteServer'
    remote.host = ip
    remote.port = 22
    remote.user = 'root'
    remote.password = 'chenganchun.0811'
    remote.allowAnyHosts = true
    return remote
}

pipeline {
    agent any
    stages {
        stage('Clone to master') {
            agent {
                label 'master'
            }
            steps {
                echo "1. Git Clone Stage"
                git credentialsId: '79d2c1e6-d63c-4284-bd1f-6b4f1dfe7f56', url: "http://212.129.149.40/181250010_irbl/backend.git", branch: "release"
            }
        }
        /*stage('change yml file properties'){
            steps{
                sh """
                   rm -f src/target/IRBL-0.0.1-SNAPSHOT.jar
                   sed -i 's/root/visitor/g' ${WORKSPACE}/src/main/resources/application.yml
                   sed -i 's/#password: fill it in application-dev.yml/password: mysql@irbl/g' ${WORKSPACE}/src/main/resources/application.yml
                   sed -i 's/com.mysql.cj.jdbc.Driver/com.mysql.jdbc.Driver/g' ${WORKSPACE}/src/main/resources/application.yml
                   sed -i 's/localhost:3306/101.132.253.222/g' ${WORKSPACE}/src/main/resources/application.yml
                   """
            }
        }
        stage('change path'){
            steps{
                sh """
                   sed -i 's#ROOT_PATH.*#ROOT_PATH = "~/data/";#g' ${WORKSPACE}/src/main/java/team/software/irbl/util/SavePath.java
                   """
            }
        }*/
        stage('Maven Build') {
            agent {
                docker {
                    image 'maven:latest'
                    args '-v /root/.m2:/root/.m2 -v /report:/report'
                }
```

```
            }
            steps {
                echo "2. Maven Build Stage and Unit Test"
                sh 'mvn clean package -Dmaven.test.skip=true'
                sh 'mvn test jacoco:report'
                //sh 'apt-get install sshpass'
                //sh 'sshpass -p chenganchun.0811 scp -r target/site/jacoco root@101.132.253.222:~'
                sh 'mkdir -p /report'
                sh 'cp -r target/site /report/ && rm -rf target/site'
            }
        }
        stage('Image Build') {
            agent {
                label 'master'
            }
            steps {
                echo "3. Image Build Stage"
                sh 'docker build -f Dockerfile --build-arg jar_name=target/IRBL-0.0.1-SNAPSHOT.jar -t irbl:${BUILD_ID} . '
                sh 'docker tag irbl:${BUILD_ID} 101.132.148.43/backend/irbl:${BUILD_ID}'

            }

        }
        stage('Push') {
            agent {
                label 'master'
            }
            steps {
                echo "4. Push Docker Image Stage"
                sh "docker login -u admin -p Harbor12345@irbl 101.132.148.43"
                sh "docker push 101.132.148.43/backend/irbl:${BUILD_ID}"
            }
        }
        stage('Pull and Run'){
            agent {
                label 'master'
            }
            steps {
                echo "5. Login Docker Image Stage"
                script{
                    remoteServer = GetRemoteServer('101.132.253.222')
                    sshCommand remote: remoteServer, command: "docker login -u admin -p Harbor12345@irbl 101.132.148.43"
                }
                echo "6. Push Docker Image Stage"
                script{
                    sshCommand remote: remoteServer, command: "docker pull 101.132.148.43/backend/irbl:${BUILD_ID}"
                }
                echo "7. Run Docker Image Stage"
                script{
                    sshCommand remote: remoteServer, command: "docker stop irbl-backend", failOnError: false
```

```
98                      sshCommand remote: remoteServer, command: "docker run --rm -it -p 40000:40000 --link db:mysql --name irbl-backend -v /root/resource:/app/resource -d 101.132.148.43/backend/irbl:${BUILD_ID}"
99                  }
100             }
101         }
102     }
103 }
104
105
```

使用harbor进行镜像仓库的管理，并且使用sshCommand连接远端的服务器进行远端指令操作。

部署考量

分服务器

因为我们都是学生机，一台学生机跑Jenkins就需要1.7GB的内存，总内存就只有2GB，所以为了避免内存溢出，采用分布式的部署方法。

数据集

我们将数据集放在服务器上面，并且在最后容器运行时将本地的数据集挂载进去，git里面只有swt的数据集作为测试使用。

测试报告

测试报告原本是存在Jenkins的docker里面，使用cp指令将其挂载到本机的目录上，并使用nginx进行访问代理。