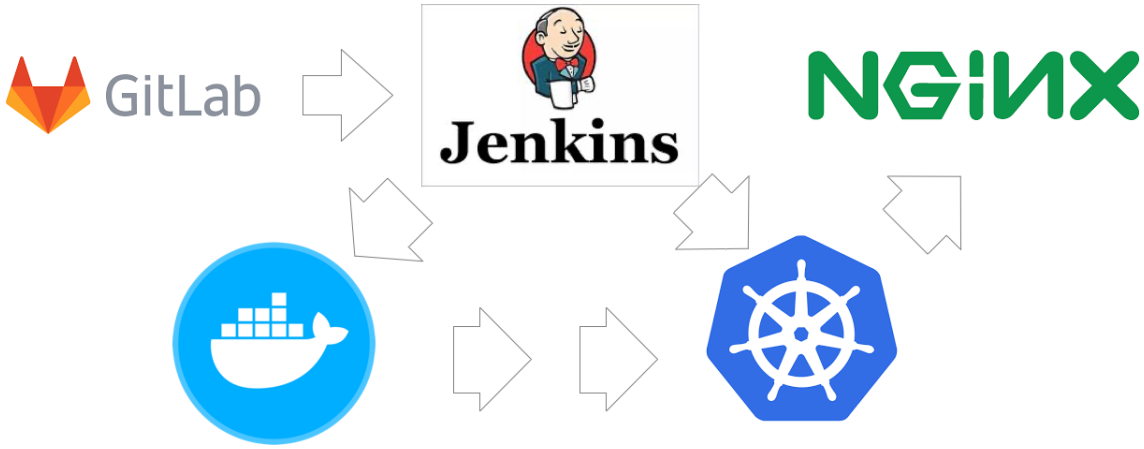
讲师(老司机)

容器虚拟化技术和自动化部署



jenkins-实战篇

软件开发生命周期

软件开发生命周期又叫做SDLC（Software Development Life Cycle），它是集合了需求分析、设计、开发、测试
和部署等过程的集合。

软件生命周期又称为软件生存周期或系统开发生命周期，是软件的产生直到报废的生命周期，周期内有问题定义、可行性分析、总体描述、系统设计、编码、调试和测试、验收与运行、维护升级到废弃等阶段，这种按时间分程的思想方法是软件工程中的一种思想原则，即按部就班、逐步推进，每个阶段都要有定义、工作、审查、形成文档以供交流或备查，以提高软件的质量。但随着新的面向对象的设计方法和技术的成熟，软件生命周期设计方法的指导意义正在逐步减少。

生命周期的每一个周期都有确定的任务，并产生一定规格的文档（资料），提交给下一个周期作为继续工作的依据。按照软件的生命周期，软件的开发不再只单单强调“编码”，而是概括了软件开发的全过程。软件工程要求每一周期工作的开始只能必须是建立在前一个周期结果“正确”前提上的延续；因此，每一周期都是按“活动-结果-审核-再活动-直至结果正确”循环往复进展的。

大致分为如下各个阶段：

- 需求分析

这是生命周期的第一阶段，根据项目需求，团队执行一个可行性计划的分析。项目需求可能是公司内部或者客户提出的。这阶段主要是对信息的收集，也有可能是对现有项目的改善和重新做一个新的项目。还要分析项目的预算多长，可以从哪方面受益及布局，这也是项目创建的目标。

- 设计

第二阶段就是设计阶段，系统架构和满意状态（就是要做成什么样子，有什么功能），和创建一个项目计划。计划可以使用图表，布局设计或者文者的方式呈现。

- 实现

第三阶段就是实现阶段，项目经理创建和分配工作给开者，开发者根据任务和在设计阶段定义的目标进行开发代码。依据项目的大小和复杂程度，可以需要数月或更长时间才能完成。

- 测试

测试人员进行代码测试，包括功能测试、代码测试、压力测试等。 在软件设计完成后要经过严密的测试，以发现软件在整个设计过程中存在的问题并加以纠正。

- 运行及维护

最后进阶段就是对产品不断的进化改进和维护阶段，根据用户的使用情况，可能需要对某功能进行修改，bug修复，功能增加等。

为什么需要jenkins

服务器规划

生成至少三台虚拟机

主机名	主机IP
k8s-master01	192.168.198.156
k8s-node01	192.168.198.157
k8s-node02	192.168.198.158
k8s-node03	192.168.198.159
harbor-155	192.168.198.155
jenkins-153	192.168.198.153
gitlab-152	192.168.198.152
jenkinsagent-151	192.168.198.151

微服项目

编写一个简单的springboo项目模拟开发环境要部署的微服项目。

创建项目

springboot/jenkinsdemo

控制器

```java
@RestController
public class JenkinsTestController {

    @GetMapping("/")
    public String login() {
        return "hello jenkins!!!";
    }
}
```

打包插件

```xml
    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
```

本地运行微服项目

```
cmd
mvn clean package -Dmaven.test.skip=true

java -jar jenkinsdemo-0.0.1-SNAPSHOT.jar

将jar上传到linux服务器
```

自定义jar包名称

在pom.xml文件中增加配置finalName。打包时可以自定义jar包名称。

```xml
    <build>
        <finalName>k8sdemo</finalName>
        <plugins>
            ....
        </plugins>
    </build>
```

dockerfile

基础镜像

```
docker pull openjdk:8-alpine3.9
```

安装docker插件

idea安装docker插件。Dockerfile、docker-compose.yml文件大部分内容会有提示信息。方便开发人员编写配置文件。

```
官网地址：
https://plugins.jetbrains.com/plugin/7724-docker/versions
```

制作镜像-dockerfile

jenkins/Dockerfile

```
FROM openjdk:8-alpine3.9
# 作者信息
MAINTAINER laosiji Docker springboot "laosiji@lagou.com"
# 修改源
RUN echo "http://mirrors.aliyun.com/alpine/latest-stable/main/" > /etc/apk/repositories && \
    echo "http://mirrors.aliyun.com/alpine/latest-stable/community/" >> /etc/apk/repositories

# 安装需要的软件，解决时区问题
RUN apk --update add curl bash tzdata && \
    rm -rf /var/cache/apk/*

#修改镜像为东八区时间
ENV TZ Asia/Shanghai
ARG JAR_FILE
COPY ${JAR_FILE} app.jar
EXPOSE 8080
ENTRYPOINT ["java","-jar","/app.jar"]
```

生成测试镜像

```
docker build --rm -t lagou/jenkinsdemo:v1 --build-arg JAR_FILE=jenkinsdemo.jar .
```

测试、删除镜像

```
docker run -itd --name=jenkinsdemo -p 8080:8080 lagou/jenkinsdemo:v1

docker ps | grep jenkins


docker logs -f jenkinsdemo

http://192.168.198.154:8080

docker stop jenkinsdemo

docker rm jenkinsdemo
```

harobor私服

推送镜像

```
登录私服
vi /etc/docker/daemon.json

"insecure-registries":["192.168.198.155:5000"]
systemctl daemon-reload
systemctl restart docker

并上传镜像
docker tag lagou/jenkinsdemo:v1
192.168.198.155:5000/lagouedu/jenkinsdemo:v1

docker push 192.168.198.155:5000/lagouedu/jenkinsdemo:v1

docker rmi -f lagou/jenkinsdemo:v1
```

部署项目

K8S部署镜像

jenkins/jenkinsdemo-deployment.yml清单

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: jenkinsdemo-deploy
  labels:
    app: jenkinsdemo-deploy
spec:
  replicas: 1
  template:
    metadata:
      name: jenkinsdemo-deploy
      labels:
        app: jenkinsdemo-deploy
```

```yaml
    spec:
      containers:
        - name: jenkinsdemo-deploy
          image: 192.168.198.155:5000/lagouedu/jenkinsdemo:v1
          imagePullPolicy: IfNotPresent
          ports:
            - containerPort: 8080
      restartPolicy: Always
  selector:
    matchLabels:
      app: jenkinsdemo-deploy

---
apiVersion: v1
kind: Service
metadata:
  name: jenkinsdemo-svc
spec:
  selector:
    app: jenkinsdemo-deploy
  ports:
    - port: 8888
      protocol: TCP
      targetPort: 8080
      nodePort: 30099
  type: NodePort
```

执行命令

```
kubectl apply -f jenkinsdemo-deployment.yml
kubectl get pods

进入容器，查看容器时间是否是东八区时间
kubectl exec -it jenkinsdemo-deploy-68785fcf7d-stjbz sh
date
exit

kubectl delete -f jenkinsdemo-deployment.yml
```

jenkins安装

jenkins官网

```
https://www.jenkins.io/zh/

官方文档
https://www.jenkins.io/zh/doc/
```

window系统

自动部署

下载官网提供的msi文件即可

```
https://www.jenkins.io/download/thank-you-downloading-windows-installer-stable/
```

手动部署一

```
java -jar jenkins.war
```

手动部署二

```
1.部署tomcat9

2.在官网下载jenkins.war
http://mirrors.jenkins.io/war-stable/2.235.5/jenkins.war

3.将war包放入apache-tomcat-9.0.34\webapps\目录中

4.启动tomcat
cmd
cd apache-tomcat-9.0.34\bin\

执行 startup.bat 启动tomcat

5.访问jenkins
http://localhost:8080/jenkins
```

linux系统

安装jenkins

```
1.安装JDK8
yum install java-1.8.0-openjdk* -y

2.yum方式安装
wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo

rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key

yum install jenkins
```

```

3.或者采用rpm方式安装：在清华大学镜像源下载jenkins的rpm文件:jenkins-2.253-1.1.noarch.rpm
https://mirrors.tuna.tsinghua.edu.cn/jenkins/redhat/

上传centos服务器进行安装：
rpm -ivh jenkins-2.190.3-1.1.noarch.rpm
```

配置jenkins

```
1.修改Jenkins配置
vi /etc/syscofig/jenkins

修改内容如下：
JENKINS_USER="root"
JENKINS_PORT="8888"


2.启动Jenkins服务
systemctl start jenkins
systemctl enable jenkins

3.打开浏览器访问
http://192.168.198.153:8888
```

获取密码

```
获取并输入admin账户密码
cat /var/lib/jenkins/secrets/initialAdminPassword
```

容器化安装

基础镜像最新版

```
docker pull jenkins/jenkins:2.235.5-alpine  355MB

docker pull jenkins/jenkins:2.253-centos7   599MB
```

推荐使用镜像版本

```
docker pull jenkins/jenkins:2.204.5         617MB

docker pull jenkins/jenkins:2.204.5-alpine  224MB
```

运行镜像

```
docker run -itd --name jenkins -p 8080:8080 -p 50000:50000 -u root --restart=always  jenkins/jenkins:2.204.5-alpine

查看容器启动日志并找到jenkins初始化密码
docker logs -f jenkins
```

浏览器测试

```
http://192.168.198.153:8080
```

部署映射目录

```
给映射目录授权
mkdir -p /data/jenkins && chown -R 1000:1000 /data/jenkins

运行容器
docker run -itd --name jenkins -p 8080:8080 -p 50000:50000 -u root -e JAVA_OPTS=-Duser.timezone=Asia/Shanghai --restart=always  -v /data/jenkins/:/var/jenkins_home/  jenkins/jenkins:2.204.5-alpine


查看容器启动日志并找到jenkins初始化密码
docker logs -f jenkins
```

中文社区

官网地址

```
官网地址：
https://bintray.com/jenkins-zh/generic/jenkins

docker官网地址：
https://hub.docker.com/r/jenkinszh/jenkins-zh

github官网地址：
https://github.com/jenkins-zh/jenkins-formulas
```

window系统部署

```
java -jar jenkins-zh.war
```

基础镜像

中文社区提供的镜像体积很庞大。

```
最新版本：
docker pull jenkinszh/jenkins-zh:2.235     718MB

推荐版本：
docker pull jenkinszh/jenkins-zh:2.204.5   682MB
```

安装jenkins

```
给映射目录授权
mkdir -p /data/jenkins && chown -R 1000:1000 /data/jenkins

docker run -itd --name jenkins -p 8080:8080 -p 50000:50000 -u root --restart=always  jenkinszh/jenkins-zh:2.204.5


docker run -itd --name jenkins -p 8080:8080 -p 50000:50000 -u root -e JAVA_OPTS=-Duser.timezone=Asia/Shanghai --restart=always -v /data/jenkins:/var/jenkins_home  jenkinszh/jenkins-zh:2.235
```

制作jenkins镜像

制作步骤

```
制作步骤：
1.编写Dockerfile文件。
2.从下一步骤开始。所有文件都在jenkins中文社区寻找。但是不能在windows系统创建。因为系统文件格式不同。jenkins启动时会报错。
3.创建init.groovy文件。
4.创建hudson.model.UpdateCenter.xml
5.创建mirror-adapter.crt
6.制作镜像
7.启动容器，试运行镜像
8.jenkins工作台操作，检验插件下载速度
9.停止容器
10.删除容器
备份镜像
```

基础镜像

```
docker pull jenkins/jenkins:2.204.5-alpine

```

Dockerfile

在基础镜像中增加时区修正。安装常用软件。更新jenkins插件为中文社区地址

```
FROM jenkins/jenkins:2.204.5-alpine
#自定义jenkins镜像
# 作者信息
MAINTAINER war kubernete jenkins "admin@lagou.com"
# 修改源
USER root
RUN echo "http://mirrors.aliyun.com/alpine/latest-stable/main/" >> /etc/apk/repositories && \
    echo "http://mirrors.aliyun.com/alpine/latest-stable/community/" >> /etc/apk/repositories
# 安装需要的软件，解决时区问题
RUN apk --update add  curl bash tzdata && \
    rm -rf /var/cache/apk/*


#修改镜像为东八区时间
ENV TZ Asia/Shanghai
ENV JENKINS_UC https://updates.jenkins-zh.cn
ENV JENKINS_UC_DOWNLOAD https://mirrors.tuna.tsinghua.edu.cn/jenkins
ENV JENKINS_OPTS="-Dhudson.model.UpdateCenter.updateCenterUrl=https://updates.jenkins-zh.cn/update-center.json"
ENV JENKINS_OPTS="-Djenkins.install.runSetupWizard=false"
COPY init.groovy /usr/share/jenkins/ref/init.groovy.d/init.groovy
COPY hudson.model.UpdateCenter.xml /usr/share/jenkins/ref/hudson.model.UpdateCenter.xml
COPY mirror-adapter.crt /usr/share/jenkins/ref/mirror-adapter.crt
```

init.groovy

在github的jenkins中文社区中找到文件，复制文件内容

```
import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.net.URL;
import hudson.init.InitMilestone;
import jenkins.model.Jenkins;

Thread.start {
  while(true) {
    Jenkins instance = Jenkins.getInstance();
    InitMilestone initLevel = instance.getInitLevel();
    Thread.sleep(1500L);
    println "Jenkins not ready when handle init config..."
    if (initLevel >= InitMilestone.PLUGINS_STARTED) {
      InputStream input = new FileInputStream("/usr/share/jenkins/ref/mirror-adapter.crt")
```

```
      FileOutputStream out = new
FileOutputStream(System.getenv("JENKINS_HOME") +  "/war/WEB-INF/update-
center-rootCAs/mirror-adapter.crt");
      byte[] buf = new byte[1024];
      int count = -1;

      while((count = input.read(buf)) > 0) {
        out.write(buf, 0, count);
      }
      println "Jenkins init ready..."
      break
    }
  }
}
```

hudson.model.UpdateCenter.xml

在github的jenkins中文社区中找到文件，复制文件内容。更换插件安装地址为国内插件地址

```
<?xml version='1.1' encoding='UTF-8'?>
<sites>
  <site>
    <id>default</id>
    <url>https://updates.jenkins-zh.cn/update-center.json</url>
  </site>
</sites>
```

mirror-adapter.crt

在github的jenkins中文社区中找到文件，复制文件内容。jenkins官方配置2对秘钥，一对是jenkins官网使用。这一对是jenkins中文社区使用。更新秘钥后。下载地址更新为国内地址。

```
-----BEGIN CERTIFICATE-----
MIICcTCCAdoCCQD/jZ7AgrzJKTANBgkqhkiG9w0BAQsFADB9MQswCQYDVQQGEwJD
TjELMAkGA1UECAwCR0QxCzAJBgNVBAcMAlNaMQ4wDAYDVQQKDAV2aWhvbzEMMAoG
A1UECwwDZGV2MREwDwYDVQQDDAhkZW1vLmNvbTEjMCEGCSqGSIb3DQEJARYUYWRt
aW5AamVua2lucy16aC5jb20wHhcNMTkxMTA5MTA0MDA5WhcNMjIxMTA4MTA0MDA5
WjB9MQswCQYDVQQGEwJDTjELMAkGA1UECAwCR0QxCzAJBgNVBAcMAlNaMQ4wDAYD
VQQKDAV2aWhvbzEMMAoGA1UECwwDZGV2MREwDwYDVQQDDAhkZW1vLmNvbTEjMCEG
CSqGSIb3DQEJARYUYWRtaW5AamVua2lucy16aC5jb20wgZ8wDQYJKoZIhvcNAQEB
BQADgY0AMIGJAoGBAN+6jN8rCIjVkQ0Q7ZbJLk4IdcHor2WdskOQMhlbR0gOyb4g
RX+CorjDRjDm6mj2OohqlrtRxLGYxBnXFeQGU7wWjQHyfKDghtP51G/672lXFtzB
KXukHByHjtzrDxAutKTdolyBCuIDDGJmRk+LavIBY3/Lxh6f0ZQSeCSJYiyxAgMB
AAEwDQYJKoZIhvcNAQELBQADgYEAD92l26PoJcbl9GojK2L3pyOQjeeDm/vV9e3R
EgwGmoIQzlubM0mjxpCz1J73nesoAcuplTEps/46L7yoMjptCA3TU9FZAHNQ8dbz
a0vm4CF9841/FIk8tsLtwCT6ivkAi0lXGwhX0FK7FaAyU0nNeo/EPvDwzTim4XDK
9j1WGpE=
-----END CERTIFICATE-----
```

制作镜像

```
docker build --rm -t lagou/jenkins:2.204.5-alpine .
```

启动容器

```
创建映射目录
mkdir -p /data/jenkins && chown -R 1000:1000 /data/jenkins


运行容器
docker run -itd --name jenkins -p 8080:8080 -p 50000:50000 -u root -e
JAVA_OPTS=-Duser.timezone=Asia/Shanghai --restart=always  -v
/data/jenkins/:/var/jenkins_home/ lagou/jenkins:2.204.6-alpine


查看容器启动日志并找到jenkins初始化密码
docker logs -f jenkins
```

测试镜像

```
http://192.168.198.153:8080

安装插件:

```

删除镜像

```
停止容器
docker stop jenkins

删除容器
docker rm jenkins

备份镜像
docker save lagou/jenkins:2.204.5-alpine -o lagou.jenkins:2.204.5-
alpine.tar

或者将镜像上传harbor私服
```

jenkins插件离线安装

Jenkins 社区的网络基础设施都是架设在国外的服务器上，而且，并没有在国内有 CDN 或者负载均衡的配置。 对所有的 Jenkins 用户而言，1500+的插件可以帮助他们解决很多问题。然而，我相信，对于国内的很多用户来说， 可能有过一些不太愉快的经历——插件下载速度很慢，甚至会超时。难道遇到这种情况下，我们就只能等吗?

程序员，作为天生懒惰的人，总是希望能通过手中的键盘来解决各种个样的问题。凭什么？下载一个插件， 我还的苦苦地等待来自美国的数据包呢？数数你手里的 Jenkins 都安装了多少个插件。30个算少的吧。 经过一番搜索，发现果然已经有前人帮忙把大树种好了。让我们一起感谢“清华大学开源软件镜像站”提供的镜像服务

```
https://mirrors.tuna.tsinghua.edu.cn/jenkins/
```