

ch07-服务部署

公众号：锋哥聊编程

主讲：小锋

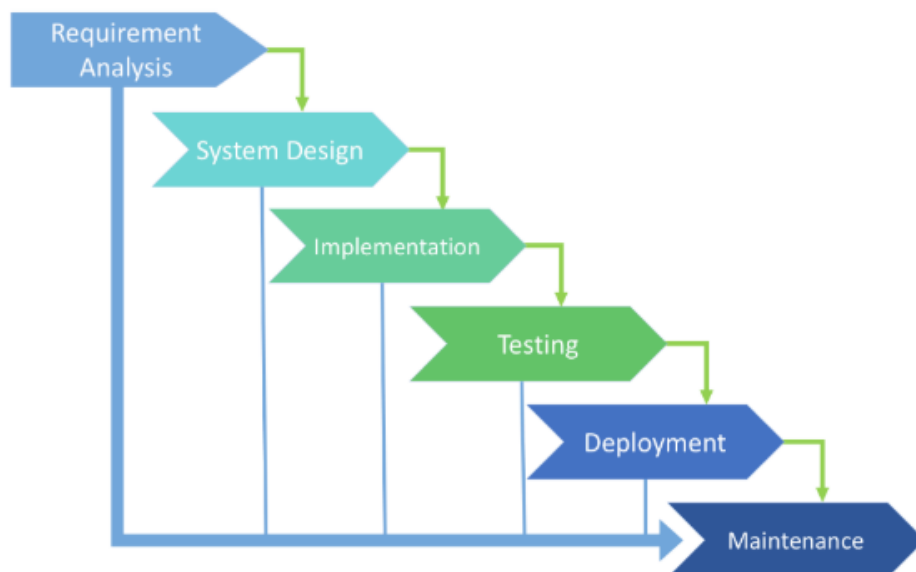


1、瀑布模型和敏捷开发

瀑布模型

瀑布模型是最著名和最常使用的软件开发模型。瀑布模型就是一系列的软件开发过程。它是由制造业繁衍出来的。一个高度化的结构流程在一个方向上流动，有点像生产线一样。在瀑布模型创建之初，没有其它开发的模型，有很多东西全靠开发人员去猜测，去开发。这样的模型仅适用于那些简单的软件开发，但是已经不适合现在的开发了。

下图对软件开发模型的一个阐述。



优势	劣势
简单易用和理解	各个阶段的划分完全固定，阶段之间产生大量的文档，极大地增加了工作量。
当前一阶段完成后，您只需要去关注后续阶段。	由于开发模型是线性的，用户只有等到整个过程的末期才能见到开发成果，从而增加了开发风险。
为项目提供了按阶段划分的检查节点，产出物	瀑布模型的突出缺点是不适应用户需求的变化。

敏捷开发

敏捷开发（Agile Development）的核心是**迭代开发**（Iterative Development）与 **增量开发**（Incremental Development）。

===何为迭代开发？===

对于大型软件项目，传统的开发方式是采用一个大周期（比如一年）进行开发，整个过程就是一次"大开发"；迭代开发的方式则不一样，它将开发过程拆分成多个小周期，即一次"大开发"变成多次"小开发"，每次小开发都是同样的流程，所以看上去就好像重复在做同样的步骤。

举例来说，SpaceX 公司想造一个大推力火箭，将人类送到火星。但是，它不是一开始就造大火箭，而是先造一个最简陋的小火箭 Falcon 1。结果，第一次发射就爆炸了，直到第四次发射，才成功进入轨道。然后，开发了中型火箭 Falcon 9，九年中发射了70次。最后，才开发 Falcon 重型火箭。如果 SpaceX 不采用迭代开发，它可能直到现在还无法上天。

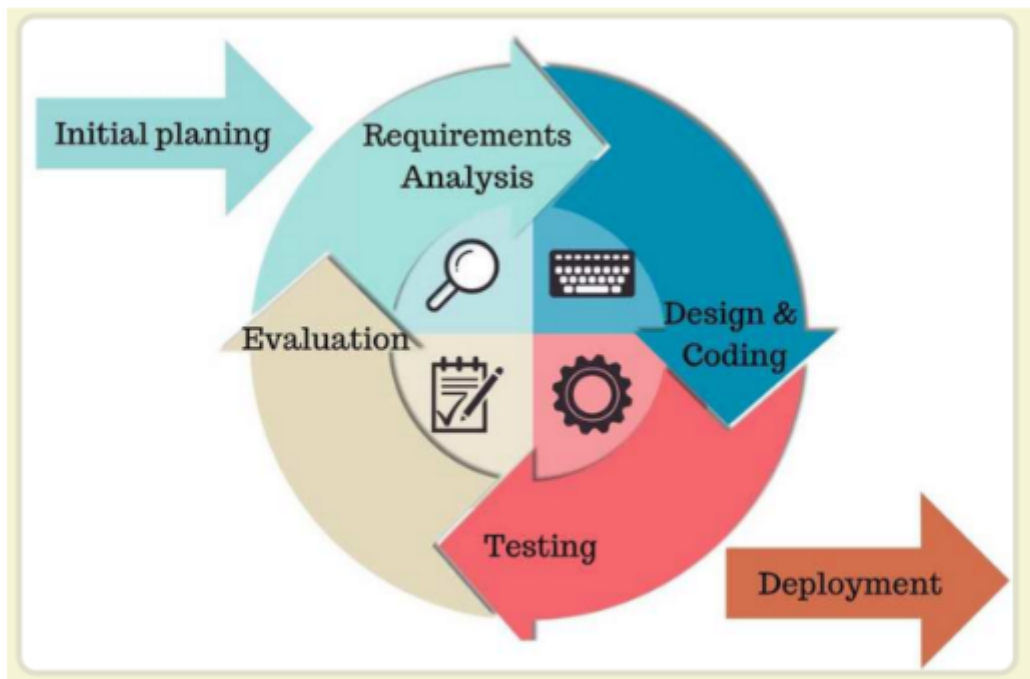
===何为增量开发？===

软件的每个版本，都会新增一个用户可以感知的完整功能。也就是说，按照新增功能来划分迭代。

举例来说，房产公司开发一个10栋楼的小区。如果采用增量开发的模式，该公司第一个迭代就是交付一号楼，第二个迭代交付二号楼.....每个迭代都是完成一栋完整的楼。而不是第一个迭代挖好10栋楼的地基，第二个迭代建好每栋楼的骨架，第三个迭代架设屋顶.....

敏捷开发如何迭代？

虽然敏捷开发将软件开发分成多个迭代，但是也要求，每次迭代都是一个完整的软件开发周期，必须按照软件工程的方法论，进行正规的流程管理。



敏捷开发有什么好处？

==早期交付==

敏捷开发的第一个好处，就是早期交付，从而大大降低成本。

还是以上一节的房产公司为例，如果按照传统的“瀑布开发模式”，先挖10栋楼的地基、再盖骨架、然后架设屋顶，每个阶段都等到前一个阶段完成后开始，可能需要两年才能一次性交付10栋楼。也就是说，如果不考虑预售，该项目必须等到两年后才能回款。

敏捷开发是六个月后交付一号楼，后面每两个月交付一栋楼。因此，半年就能回款10%，后面每个月都会有现金流，资金压力就大大减轻了。

==降低风险===

敏捷开发的第二个好处是，及时了解市场需求，降低产品不适用的风险。

请想一想，哪一种情况损失比较小：10栋楼都造好以后，才发现卖不出去，还是造好第一栋楼，就发现卖不出去，从而改进或停建后面9栋楼？

2、持续集成与持续部署

什么是持续集成/持续部署

持续集成（Continuous integration，简称 CI）

持续部署（Continuous Deployment，简称 CD）

所谓持续集成/持续部署：项目要不断**持续**上线，且**自动化**上线！！

持续集成的目的，就是让产品可以快速迭代，同时还能保持高质量。它的核心措施是，代码集成到主干之前，必须通过自动化测试。只要有一个测试用例失败，就不能集成。

通过持续集成，团队可以快速的从一个功能到另一个功能，简而言之，敏捷软件开发很大一部分都要归功于持续集成。

持续集成的流程

根据持续集成的设计，代码从提交到生产，整个过程有以下几步。

- 提交

流程的第一步，是开发者向代码仓库提交代码。所有后面的步骤都始于本地代码的一次提交（commit）。

- 测试（第一轮）

代码仓库对commit操作配置了钩子（hook），只要提交代码或者合并进主干，就会跑自动化测试。

- 构建

通过第一轮测试，代码就可以合并进主干，就算可以交付了。

交付后，就先进行构建（build），再进入第二轮测试。所谓构建，指的是将源码转换为可以运行的实际代码，比如安装依赖，配置各种资源（样式表、JS脚本、图片）等等。

- 测试（第二轮）

构建完成，就要进行第二轮测试。如果第一轮已经涵盖了所有测试内容，第二轮可以省略，当然，这时构建步骤也要移到第一轮测试前面。

- 部署

过了第二轮测试，当前代码就是一个可以直接部署的版本（artifact）。将这个版本的所有文件打包（tar filename.tar *）存档，发到生产服务器。

- 回滚

一旦当前版本发生问题，就要回滚到上一个版本的构建结果。最简单的做法就是修改一下符号链接，指向上一个版本的目录。

持续集成的组成要素

- 一个**自动化构建过程**，从检出代码、编译构建、运行测试、结果记录、测试统计等都是自动完成的，无需人工干预。
- 一个**代码存储库**，即需要版本控制软件来保障代码的可维护性，同时作为构建过程的素材库，一般使用SVN或Git。

- 一个**持续集成服务器**，**Jenkins** 就是一个配置简单和使用方便的持续集成服务器。CI/CD软件

持续集成的好处

- 1、降低风险，由于持续集成不断去构建，编译和测试，可以很早期发现问题，所以修复的代价就少；
- 2、对系统健康持续检查，减少发布风险带来的问题；
- 3、减少重复性工作；
- 4、持续部署，提供可部署单元包；
- 5、持续交付可供使用的版本；
- 6、增强团队信心；

3、Jenkins简介



Jenkins 是一款流行的Java开源持续集成（Continuous Integration）工具，广泛用于项目开发，具有自动化构建、测试和部署等功能。官网：<http://jenkins-ci.org/>。

CD/CI: Continuous Deployment (持续部署)、Continuous Integration (持续集成)

Jenkins的特征：

- 开源的Java语言开发持续集成工具，支持持续集成，持续部署。
- 易于安装部署配置：可通过yum安装,或下载war包以及通过docker容器等快速实现安装部署，可方便web界面配置管理。
- 消息通知及测试报告：集成RSS/E-mail通过RSS发布构建结果或当构建完成时通过e-mail通知，生成JUnit/TestNG测试报告。
- 利用K8S (kubernetes) 分布式构建：支持Jenkins能够让多台计算机一起构建/测试。
- 文件识别：Jenkins能够跟踪哪次构建生成哪些jar，哪次构建使用哪个版本的jar等。
- 丰富的插件支持：支持扩展插件，你可以开发适合自己团队使用的工具，如git, svn, maven, docker等。

4、配套软件安装

JDK

```
yum install java-1.8.0-openjdk* -y
```

Git

```
yum -y install git
```

Maven

先上传Maven软件到192.168.66.133

```
tar -xzf apache-maven-3.5.0-bin.tar.gz 解压
```

```
mkdir -p /opt/maven 创建目录
```

```
mv apache-maven-3.5.0/* /opt/maven 移动文件
```

并修改conf/settings.xml文件指定仓库

```
<localRepository>/opt/maven/repo</localRepository>
```

阿里云私服：

```
<mirror>
  <id>alimaven</id>
  <name>aliyun maven</name>
  <url>http://maven.aliyun.com/nexus/content/groups/public/</url>
  <mirrorOf>central</mirrorOf>
</mirror>
```

配置Maven环境变量

```
vi /etc/profile
```

```
export JAVA_HOME=/usr/java/jdk1.8.0_171-amd64
export MAVEN_HOME=/opt/maven
export PATH=$PATH:$JAVA_HOME/bin:$MAVEN_HOME/bin
```

```
source /etc/profile 配置生效
```

```
mvn -v 查找Maven版本
```

5、Jenkins安装

下载Jenkins包

下载地址: <https://mirrors.jenkins-ci.org/redhat/>

课程版本: jenkins-2.355-1.1.noarch.rpm

安装

```
rpm -ivh jenkins-2.355-1.1.noarch.rpm
```

修改Jenkins配置

修改Jenkins的账户和端口

```
vi /usr/lib/systemd/system/jenkins.service
```

这里的jenkins用户改为root

```
# TimeoutStartSec=90s

# Unix account that runs the Jenkins daemon
# Be careful when you change this, as you need to update the permissions
# $JENKINS_HOME, $JENKINS_LOG, and (if you have already run Jenkins)
# $JENKINS_WEBROOT.
User=root
Group=root

# Directory where Jenkins stores its configuration and workspaces
```

端口改为8000

```
# Port to listen on for HTTP requests. Set to -1 to disable.
# To be able to listen on privileged ports (port numbers less than 1024),
# add the CAP_NET_BIND_SERVICE capability to the AmbientCapabilities
# directive below.
Environment="JENKINS_PORT=8000"
```

启动Jenkins

```
systemctl start jenkins
```

解锁Jenkins

访问: <http://192.168.66.133:8000>

解锁 Jenkins

为了确保管理员安全地安装 Jenkins，密码已写入到日志中（[不知道在哪里？](#)）该文件在服务器上：

```
/var/jenkins_home/secrets/initialAdminPassword
```

请从本地复制密码并粘贴到下面。

管理员密码

到提示目录复制密码，进入下一步。

安装Jenkins插件

自定义Jenkins

插件通过附加特性来扩展Jenkins以满足不同的需求。

安装推荐的插件


















安装Jenkins社区推荐的插件。

选择插件来安装

选择并安装最适合的插件。

选择安装推荐的插件，这个步骤要等待一段时间。

新手入门

<input checked="" type="checkbox"/> Folders	 OWASP Markup Formatter	 Build Timeout	 Credentials Binding	<div><div>** JavaBeans Activation Framework (JAF) API</div><div>** JavaMail API</div><div>Folders</div><div>** Mina SSHD API :: Commons</div><div>** Mina SSHD API :: Core</div></div>
 Timestampers	 Workspace Cleanup	 Ant	 Gradle	
 Pipeline	 GitHub Branch Source	 Pipeline: GitHub Groovy Libraries	 Pipeline: Stage View	
 Git	 SSH Build Agents	 Matrix Authorization Strategy	 PAM Authentication	
<input type="radio"/> LDAP	 Email Extension	<input type="radio"/> Mailer	 Localization: Chinese (Simplified)	

创建管理员账户

创建第一个管理员用户

用户名:


密码:

确认密码:

全名:

电子邮件地址:

完成后，进入后台

 **Jenkins**

Dashboard >

+ 新建Item

👤 用户列表

📁 构建历史

⚙️ Manage Jenkins

👤 My Views

📁 新建视图

欢

This | build

Star

Ci

如果Jenkins安装失败，参考以下步骤，删除Jenkins再重来

```
rpm -e jenkins

rpm -ql jenkins

find / -iname jenkins | xargs -n 1000 rm -rf
```

6、Jenkins全局工具配置

Manager Jenkins->Global Tool Configuration



Global Tool Configuration

Configure tools, their locations and automatic installers.

Git

Git installations

≡ Git

Name

git

Path to Git executable ?

git

☐ Install automatically ?

Maven

Maven 安装

系统下Maven 安装列表

新增 Maven

≡

Maven

Name

maven

MAVEN_HOME

/opt/maven

7、微服务整合Docker

每个微服务使用的dockerfile的方式进行构建镜像后创建容器，需要在每个微服务中添加docker相关的配置

(1) 修改**每个微服务**的pom文件，添加dockerfile的插件

```
<build>
  <finalName>mafeng-user</finalName>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
      <executions>
        <execution>
          <goals>
            <goal>repackage</goal>
          </goals>
        </execution>
      </executions>
    </plugin>
    <plugin>
      <groupId>com.spotify</groupId>
      <artifactId>dockerfile-maven-plugin</artifactId>
      <version>1.3.6</version>
      <configuration>
        <repository>${project.artifactId}</repository>
        <buildArgs>
```

```
<JAR_FILE>target/${project.build.finalName}.jar</JAR_FILE>
    </buildArgs>
  </configuration>
</plugin>
</plugins>
</build>
```

注意：finalName要修改为当前项目名称。

(2) 在每个微服务的根目录下创建Dockerfile文件，如下：

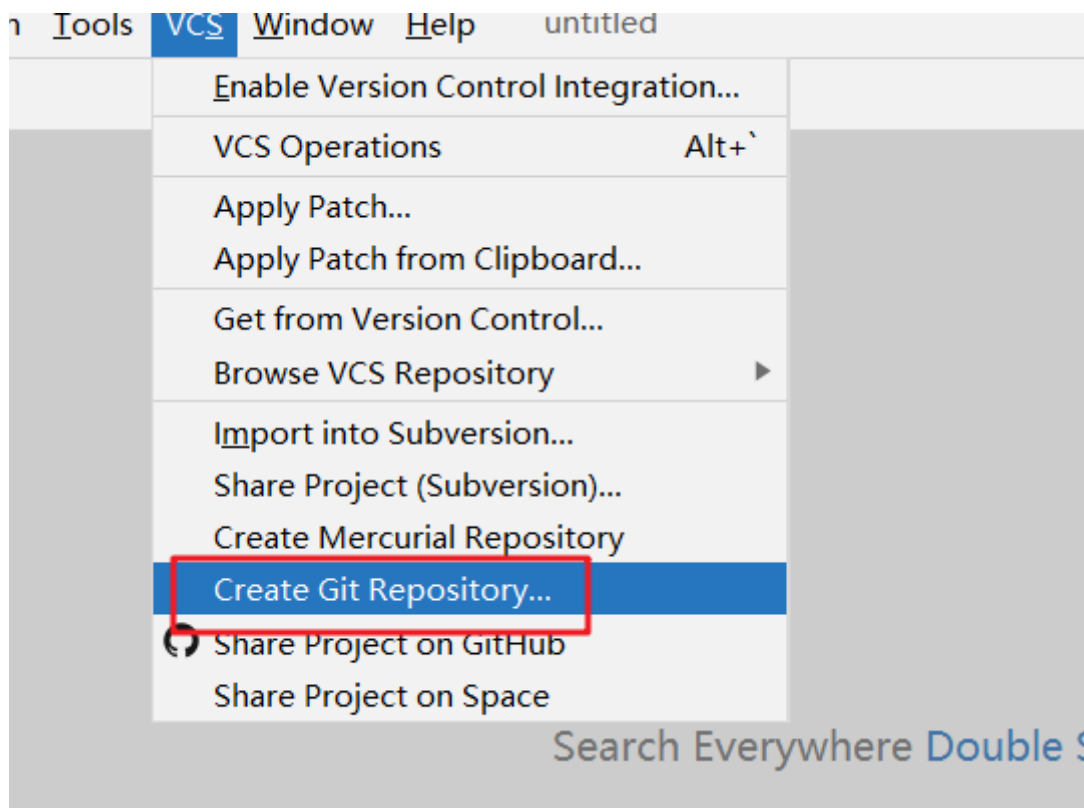
```
# 设置JAVA版本
FROM openjdk:8
# 指定存储卷，任何向/tmp写入的信息都不会记录到容器存储层
VOLUME /tmp
# 拷贝运行JAR包
ARG JAR_FILE
COPY ${JAR_FILE} app.jar
# 入口点，执行JAVA运行命令
ENV JAVA_OPTS=""
-server \
-Xms256m \
-Xmx512m \
-XX:MetaspaceSize=256m \
-XX:MaxMetaspaceSize=512m"
ENTRYPOINT java ${JAVA_OPTS} -jar /app.jar
```

8、提交代码到Git仓库

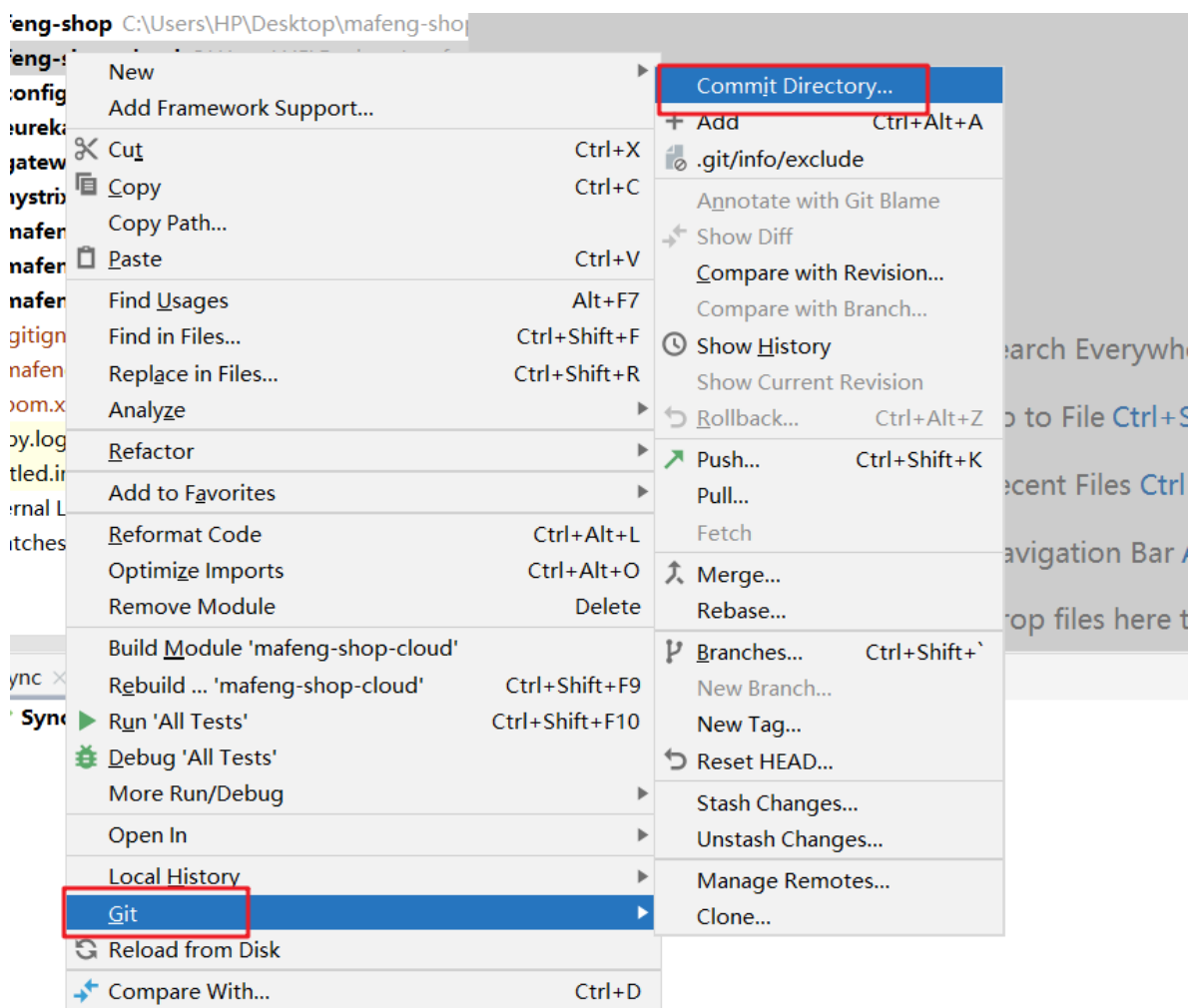
- 1) 在码云上建立新仓库
- 2) mafeng-shop-cloud根目录下建立.gitignore文件

```
.idea
target
```

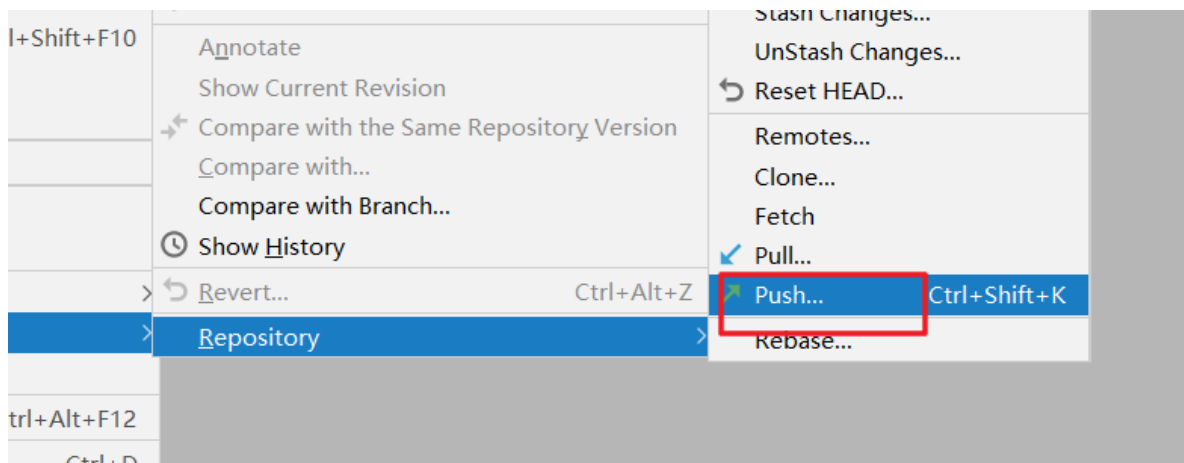
- 3) 创建git仓库



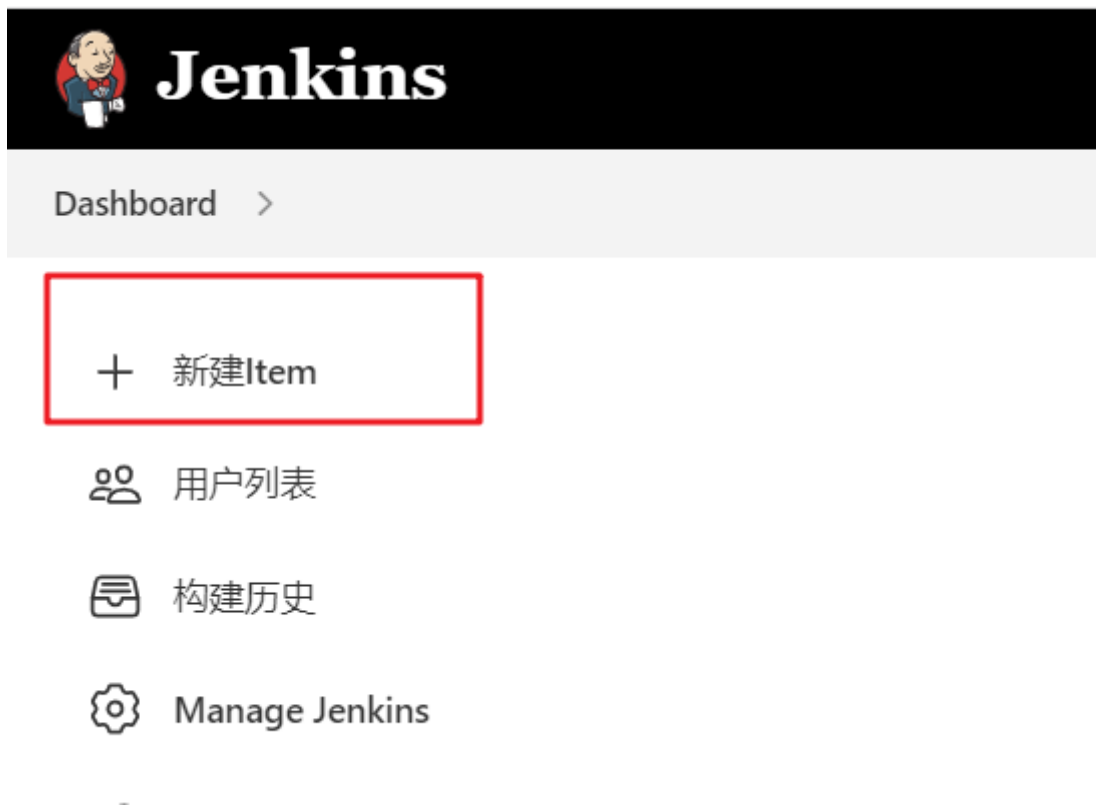
4) 提交代码



5) 推送代码到码云



9、基础依赖安装



输入一个任务名称

mafeng-shop-cloud

» 必填项



Freestyle project

This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system other than software build.



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的任务类型）。



构建一个多配置项目

源码管理

☐ 无

☒ Git ?

Repositories ?

Repository URL ?

https://gitee.com/eric123456/mafeng-cloud-jenkins.git

Credentials ?

1014671449@qq.com/*****

构建

≡ Invoke top-level Maven targets ?

Maven 版本

maven

目标

clean install -Dmaven.test.skip=true

高级...

```
clean install -Dmaven.test.skip=true
```

10、微服务持续集成

创建mafeng-user和mafeng-order项目，参考以下步骤在Jenkins配置并构建，成功后便可进行接口测试

☐ 无

☒ Git ?

Repositories ?

Repository URL ?

https://gitee.com/eric123456/mafeng-cloud-jenkins.git

Credentials ?

1014671449@qq.com/*****

+ 添加

高级...

≡ Invoke top-level Maven targets ?

Maven 版本

maven

目标

clean install -Dmaven.test.skip=true dockerfile:build -f mafeng-user/pom.xml

```
clean install -Dmaven.test.skip=true dockerfile:build -f mafeng-user/pom.xml
```

≡ Execute shell ?

命令

查看 可用的环境变量列表

```
#删除之前的容器
docker rm -f $(docker ps -a -f name=$JOB_NAME --format '{{.ID}}' )
fi
# 清理镜像
docker image prune -f
# 启动docker服务
docker run -d --net=host --name $JOB_NAME $JOB_NAME
```

```
if [ -n "$(docker ps -a -f name=$JOB_NAME --format '{{.ID}}' )" ]
then
#删除之前的容器
docker rm -f $(docker ps -a -f name=$JOB_NAME --format '{{.ID}}' )
fi
# 清理镜像
docker image prune -f
# 启动docker服务
docker run -d --net=host --name $JOB_NAME $JOB_NAME
```