基于jenkins 构建持续集成环境

## **课程大纲**

jenkins 原理讲解,如何通过git 、maven gitlab 、tomcat 构建持续集成环境

jenkins 部署与环境安装

jenkins 项目配置与管理

## **GIT的原理**

1 必须从gitlab(源码私服) 获取我要打包的源

1.1下载指定分支最新源码(TEst)

1.2 并在本地切换过去

2 运行 mvn clean install 构建打包

3 打包之后存储至指下位置

4 将对应的包 上传到我们的nexus 私服（deploy、用户和密码）

5 基于Tomcat进行动态部署上去。

## **jenkins 部署**

**下载**

https://jenkins.io/download/

**下载对应war 包 两种启动试：**

1. 直接基于任何servlet 容器(jetty\tomcat)等即可启动
2. 基于java -jar 命令启动

java -jar jenkins.war --ajp13Port=-1 --httpPort=8888

**关于jenkins 插件安装**

1. 首先选择默认推推荐的插件安装完成
2. 进入插件管理页安装如下插件

#maven 管理插件

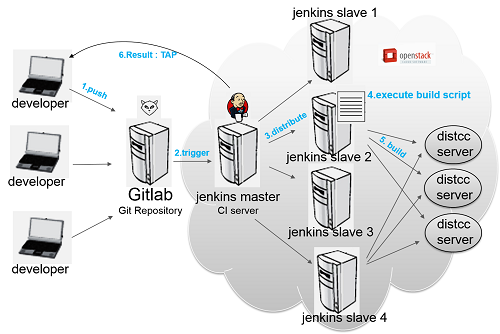
[Maven Integration plugin](https://plugins.jenkins.io/maven-plugin)

#容器部署插件

Deploy to container Plugin

**构建环境安装 ：**

1. 服务端安装maven
2. 服务端安装git 客户端



系统的工作流程大概分为以下几步:

1> 开发者将新版本push到git server (Gitlab)。

2> Gitlab随后触发jenkins master结点进行一次build。(通过web hook或者定时检测)

3> jenkins master结点将这个build任务分配给若干个注册的slave结点中的一个，这个slave结点根据一个事先设置好的脚本进行build。这个脚本可以做的事情很多，比如编译，测试，生成测试报告等等。这些原本需要手动完成的任务都可以交给jenkins来做。

4> 我们在build中要进行编译，这里使用了分布式编译器distcc来加快编译速度。

**notes**

jenkins的工作原理是先将源代码从gitlab中拷贝一份到本地，然后根据设置的脚本进行build。我们可以看出，整个系统的关键就是那个build脚本，用来告诉jenkins在一次集成中需要执行的任务。

不过我之后是用的Github作为git server。但其实差不多，先讲到这里，重点难点还是在之后jenkins的安装配置使用上。