Jenkins使用简介

by dmp-team 田志鹏

2017/09/12

What is Jenkins?

- building, deploying and
- 持续集成continuous integration
- 持续交付continuous delivery

Jenkins前身Hudson是sun公司时的一个开源项目, oracle收购后
管理产生争议, 最终oracle将Hudson注册为商标, 想要商业化.
创始人和开源社区以Jenkins为名的分支继续, 而后Jenkins分支进
oracle最终将Hudson转移给Eclipse基金会.

Jenkins BuildBot TeamCity Travis-Ci GoCD Bamboo !!!!!!

一张图看一下代码写完之后有哪些事

编译打包/部署/自动化工具 `. 一则关于oracle的笑话
我这里谢了两个词一个事持续集成 一个是持续交付.

其实关于ci/cd工具, 有太多的轮子了BuildBot TeamCity Travis-Ci等等, 我已经花了一些时间比较哪个更好了, 其实只要拿一个合适的用起来就行了, jenkins使用用户最多,能满足我们的需求就够了

//看图片
这个图是我对于 代码写完之后有哪些事 的理解 欢迎补充:
首先是代码管理, 比如我们用gitlab,
配置管理, 现在我们项目比较简单, 所有配置都和代码在一起, 更合理的做法是有集中的配置管理, 这个配置包括程序的配置,和像密码这样的资质配置,和服务器节点这样的配置.
下面是持续集成, 集成我理解就是编译打包等过程, 我们的项目比较简单, 所有代码在一个项目中, 打一个jar包就可以, 更大的项目往往是多个子组件组成, 那么就需要把他们组合起来.
在下面是交付, 就是将编译好的程序放到服务器上跑起来.
而服务器就涉及到了服务器节点管理, 环境搭建, 这部分主要是自动化运维,在我们公司是idc来负责.
而更进一步的自动化运维, 这张图的所有功能都应该是运维来做, 做成paas, 业务部门可以不用做了.

1000+ Plugins

Pipeline as Code

刚才我们讲到jenkins要和上下游很多的东西进行交互，其实他是靠着1000+的社区贡献的插件来实现的.
gitlab/maven/junit/docker/ecs/jira/dingding
这是jenkins的第一个优点

第二个优点就是pipeline as code. pipeline不是我们说的linux管道, 是jenkins 2.0出现的概念, 就是将部署的步骤写成一个叫做pipeline的groovy脚本文件, pipeline as code的意义在于, 这个描述部署步骤的脚本, 是随着代码进行托管的. 这个的好处首先是易于管理不会丢, 可重用性强, 省的点点点配置, 可视化效果好等等

在出现pipeline以前, 一个部署是什么样的呢, 在jenkins的页面上创建项目, 在页面上通过点点点配置项目的部署全过程.
有了jenkins之后, 只需要在项目上配置好pipeline文件在哪, jenkins通过文件来执行构建过程.

所以jenkins就有了这两种玩法. 前者是以前的玩法, 好处是1000+插件可用, 后者写pipeline脚本, 坏处是脚本里只能用那些支持pipeline的插件定义的功能了.

调研之后决定用pipeline

Jenkins功能

- 源码管理
- 构建触发器
- 构建环境
- 构建步骤
- 构建后操作

//看页面
这个页面是一个基本的jenkins项目的配置页面. 可以看到有这几项配置,
而我们使用pipeline方式的项目,就只剩下触发器和pipeline文件可以配置了, 其余所有操作都写到pipeline文件里了

这里我们顺便看一下创建项目的过程, 有这几种项目
1. 自由风格
2. 文件夹
3. 普通pipeline
4. 多分支pipeline
5. 多配置?

源码管理各种git/svn都支持这里就不讲了.

构建触发器

- 远程触发 (发一个请求)
- Build periodically (cron表达式)
- Build after other projects are built
- Poll SCM (poll changes)
- gitlab事件发生 (push,merge)(由gitlab向jenkins发请求)(需插件)
- 手动

这个gitlab和上面拉不同, 下面这个是推的逻辑.
这两个触发器很重要. 比如我们想在某分支发生变化时构建,
再比如说我想在有人发起merge request的时候构建, 配合gitlab做到, 只有build通过才能合代码.

- 脚本分两种:
- Declarative Pipeline Vs Scripted Pipeline

构建步骤
//先看 插件式的构建步骤
下面只说pipeline相关的了.
首先jenkins的pipeline脚本有两种写法, 一种是声明式的, 一种是脚本式的,
声明式的就是一种DSL, 使用给定的一些语法写, 比较简单.
脚本式的就是写groovy代码, 比较灵活自由可控,
两种都可以, 但我只看了脚本式的怎么写.
talk is cheap
//todo看代码

构建步骤

钉子	锤子
1. git checkout 2. configuration 3. maven build 4. scp file 5. sql patch 6. opt tomcat 7. opt nginx 8. **测试** 9. 通知 10. 生成report	1. node 2. stage 3. tool 4. checkout scm 5. sh 6. echo / error / input 7. configFileProvider 8. withCredentials

左侧是我todo收集的一些常见的构建步骤.
这里我加粗了 测试 这步, 虽然我们不做测试, 但对于很多公司来讲, 使用CI的一个主要目的就是运行测试, 据说某公司每天n次代码提交, 每次代码提交运行成百上千的测试用例, 每天交付n次, 具体多少次不知道, 总之很夸张.
右侧是jenkins提供的可用的功能和steps
我只列了简单的几个, 其实有很多可用的
//看页面

小功能:用户权限管理

- 安全域
  - Jenkins专有用户数据库
  - LDAP
  - Unix用户/用户组
  - Gitlab
- 授权策略
  - 任何用户/登录用户 可以做任何事
  - 安全矩阵
  - 项目矩阵授权策略
  - 基于角色

接下来将三个功能点, 这三个功能点解决了我开始用jenkins时的三个需求
首先安全问题
安全分为各个方面, 我这里讲一下用户权限管理
安全域, 就是用户的来源或者存在哪
授权策略, 就是说用户登录后如何决定他能做哪些事
//看页面 安全配置
先定义角色, 分三种
然后给用户分配角色,
这个就解决了我当时考虑的不同用户操作不同项目的需求

小功能:Credentials

- 种类
  - username password
  - username private key
  - secret file

```
stage('ssh with specific key') {
    withCredentials([sshUserPrivateKey(credentialsId: '6f57e346-4c98-483f-a711-e80d6dab6b2c', keyFileVariable: 'keyfile')]) {
        def out = sh([returnStdout: true, script: """
            ssh -i ${keyfile} root@h31.mzhen.cn 'source ~/.bash_profile;ls'
        """])
        echo "out ${out}"
    }
}
```

Credentials即资质, 就是我们将的提供认证的一些东西, 比如账号密码, 比如密钥等.

//看页面 credentials

在jenkins中提供了Credentials管理的功能.

首先它是分级存放的, 有global的, 有项目自己的, 分domain的.

这个就解决了我当时考虑的部署到不同服务器怎么访问的需求

小功能:配置管理功能

- 配置文件管理
  - 不应该出现在gitlab, 而是在部署的时候在这里加上去

```
stage('config file') {
    configFileProvider([configFile(fileId: 'dmp-token-test', variable: 'token_file')]) {
        sh 'cp ${token_file} tokenfile'
    }
}
```

可以在构建的时候拿来用

这个就解决了我当时考虑的构建时配置文件管理的需求

更多

- 主从多机部署问题?
- 断点?
- gitlab?
- jira?
- docker?
- blue ocean?

安全问题+1

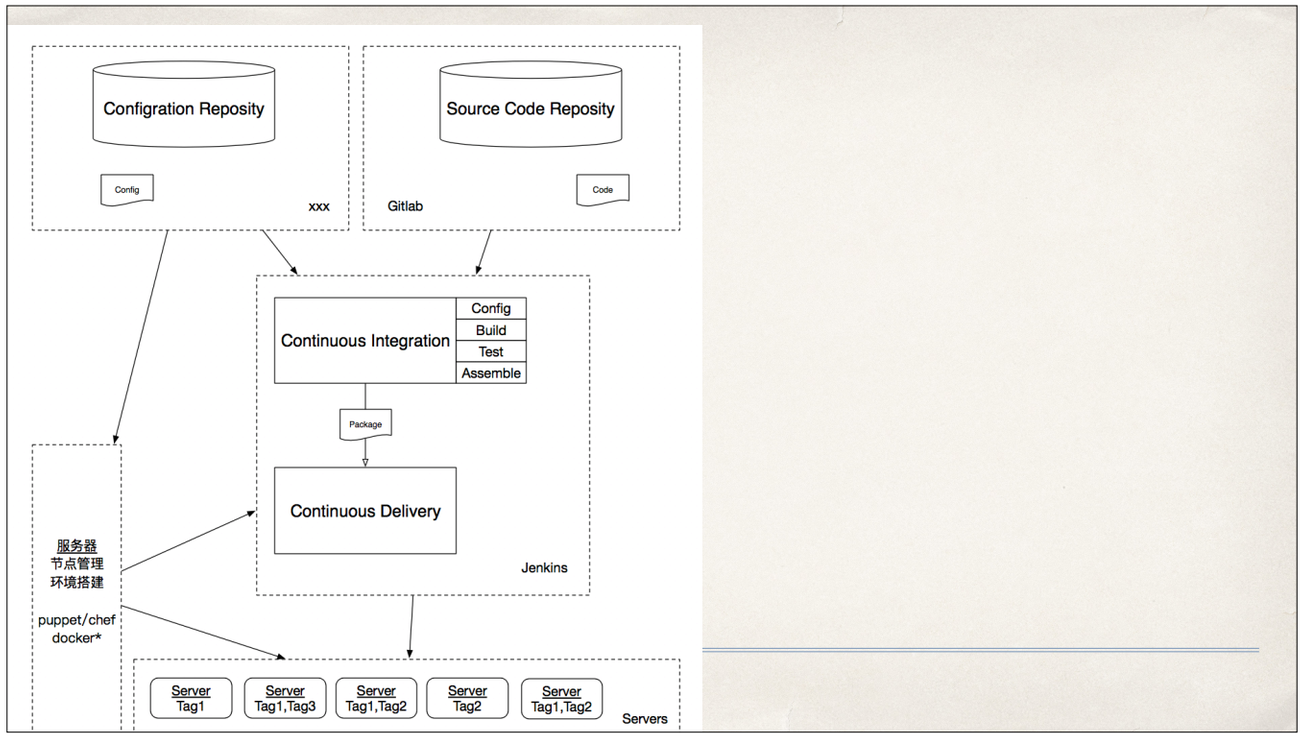
- ssh登录和执行命令
- 我来执行还是我调用服务器上的脚本?

参考资料

- jenkins.io wiki.jenkins.io
- github
- step ref
- editor和语法帮助 (原生 / blue ocean)
- 很强的教程
- 十个最佳实践
- groovy in y minutes

1 jenkins官网,必须吐槽, 文档非常之差, 可能跟社区贡献的有关吧
2 github
3 这个是所有可用的step的索引, 就属于我说的非常差的文档
4 作为上面的替代, jenkins实例上有自带的editor和语法帮助

Thanks


Configration Reposity
Config
xxx
Source Code Reposity
Code
Gitlab
Continuous Integration
Config
Build
Test
Assemble
Package
Continuous Delivery
Jenkins
服务器
节点管理
环境搭建
puppet/chef
docker*
Server
Tag1
Server
Tag1,Tag3
Server
Tag1,Tag2
Server
Tag2
Server
Tag1,Tag2
Servers