第一章 git系列课程

第一集 课程介绍

- 课程适用群体

  面向初级开发者、想了解并使用git、自行搭建企业内部gitlab、自行搭建企业内部持续集成平台

- 课程范围

  git的使用 gitlab的搭建与使用 持续集成平台jenkins搭建与使用 代码质量管理平台sonarqube的搭建与使用 企业私服--nexus搭建及使用 部分linux命令

- 课程大纲
- 课程效果演示

第二集 git简介

- git是什么呢？

  维基百科给出的定义： git是一个分布式版本控制软件，最初由（Linus Torvalds）创作

  什么是版本控制

  项目经理与程序员的恩怨情仇 企业真实案例：开发了a功能，之后项目所要改成b功能，开发完之后又要改c功能，最后又说还是用回a功能。 没有git等版本控制软件之前，如何做版本控制。 有了git之后，如何做版本控制。

- git的作用是什么？

  版本控制

- git的优势哪里？

  同类型软件有svn、cvs，git相比于他们最大的优势就在于git是分布式的 企业真实案例：svn服务器欠费，急需发版本。非常麻烦 Git 允许多个远程仓库存在，使得这样一种工作流成为可能：每个开发者拥有自己仓库的写权限和其他所有人仓库的读权限。 这种情形下通常会有个代表``官方''项目的权威的仓库。 要为这个项目做贡献，你需要从该项目克隆出一个自己的公开仓库，然后将自己的修改推送上去。 接着你可以请求官方仓库的维护者拉取更新合并到主项目。 维护者可以将你的仓库作为远程仓库添加进来，在本地测试你的变更，将其合并入他们的分支并推送回官方仓库。

- 初识github

  全球最大的同性交友网站：https://github.com/

第三集 window与linux双环境安装git

- windows下面安装git

  下载git 到git官网上下载，自行选择合适系统的 https://git-scm.com/ 如果是win版本的，直接按默认安装即可

- linux上安装，以centos 6.6为例

  yum命令安装

  yum install gityum install 安装的git不是最新版本，如需最新版本需要自行编译

  编译git源码安装：

  到下面的网站下载合适的版本 https://mirrors.edge.kernel.org/pub/software/scm/git/

  安装git的依赖项

  yum install curl-devel expat-devel gettext-devel openssl-devel zlib-devel

  yum install gcc perl-ExtUtils-MakeMaker

  移除已经安装的git

  yum remove git

  cd git 解压目录

  预编译git

  ./configure --prefix=/usr/local/git_2.9.5

  编译并安装git

  make && make install

  将git的脚本软连接到/usr/bin/ 目录下

  ln -s /usr/local/git_2.9.5/bin/* /usr/bin/

  git安装完成

第四集 git的入门级命令（上）

- 为什么建议使用命令行的方式操作git？

  1.命令行会了，图形界面的操作时完全没问题的，反之，则不然 2.有些地方如linux服务器，没有图形界面，如果碰到问题需要使用git，不会命令行操作啥都干不了

- git 本地操作

  git --help 调出Git的帮助文档 git +命令 --help 查看某个具体命令的帮助文档 git --version 查看git的版本 git init 生成空的本地仓库 git add 将文件添加到暂存区

- 初次commit之前，需要配置用户邮箱及用户名，使用以下命令

  git config --global user.email "you@example.com" git config --global user.name "Your Name"

- git commit

  将暂存区里的文件提交到本地仓库

第五集 git的入门级命令（下）

- git remote

  用于管理远程仓库

- git push -u origin master、

  往名字为origin的仓库的master分支上提交变更

- git fetch

  拉取远程仓库的变更到本地仓库

- git merge origin/master

  将远程的变更，合并到本地仓库的master分支

- git pull

  不建议使用 等同于fetch之后merge

第六集 git的文件状态

- git status

  用于查看git的状态

- git rm

  用于git文件的删除操作 如果只是 git rm --cache 仅删除暂存区里的文件 如果不加--cache 会删除工作区里的文件 并提交到暂存区

- git checkout

  直接加文件名 从暂存区将文件恢复到工作区，如果工作区已经有该文件，则会选择覆盖加了【分支名】 +文件名 则表示从分支名为所写的分支名中拉取文件 并覆盖工作区里的文件

  新建文件--->Untracked 使用add命令将新建的文件加入到暂存区--->Staged 使用commit命令将暂存区的文件提交到本地仓库--->Unmodified 如果对Unmodified状态的文件进行修改---> modified 如果对Unmodified状态的文件进行remove操作--->Untracked

第七集 git的图形化客户端

- 图形化客户端: sourcetree
- 下载: https://www.sourcetreeapp.com/
- 安装: 由于种种不可描述的原因，无法注册账号且无法登陆所以需要绕过登陆
- 绕过登陆

  去到 C:\Users\当前用户目录\AppData\Local\Atlassian\SourceTree 目录下 新建 accounts.json 文件 将一下内容复制进去 [{ "$id": "1", "$type": "SourceTree.Api.Host.Identity.Model.IdentityAccount, SourceTree.Api.Host.Identity", "Authenticate": true, "HostInstance": { "$id": "2", "$type": "SourceTree.Host.Atlassianaccount.AtlassianAccountInstance, SourceTree.Host.AtlassianAccount", "Host": { "$id": "3", "$type": "SourceTree.Host.Atlassianaccount.AtlassianAccountHost, SourceTree.Host.AtlassianAccount", "Id": "atlassian account" }, "BaseUrl": "https://id.atlassian.com/" }, "Credentials": { "$id": "4", "$type": "SourceTree.Model.BasicAuthCredentials, SourceTree.Api.Account", "Username": "", "Email": null }, "IsDefault": false }]

第八集 git的分支

- 什么是分支

  软件项目中启动一套单独的开发线的方法
- 为什么使用git

  可以很好的避免版本兼容开发的问题，避免不同版本之间的相互影响 封装一个开发阶段 解决bug的时候新建分支，用于对该bug的研究
- git中跟分支相关的命令

  git branch 分支名 git branch 不加任何参数，列出所有的分支，分支前面有*号，代表该分支为当前所在分支
  - 创建分支的时候，分支名不用使用特殊符号 git branch -d 分支名
    *不能删除当前所在的分支 git branch -m 旧分支名 新分支名

    git checkout 分支名 切换分支 如果在分支上面对文件进行修改之后，没有commit就切换到另外一个分支b， 这个时候会报错，因为没有commit的文件在切换分支之后会不覆盖。所以Git 报错提示。

    git checkout -f 分支名 强制切换到分支，如果当前有为提交的变更，会直接丢弃 -f 参数一定一定要非常非常小心使用，一般情况下不建议使用，除非真的要强制去执行

第九集 log命令

- log命令的作用

  用于查看git的提交历史
- git log命令显示的信息的具体含义

commit 4a70ceb24b6849ad830d6af5126c9227b333d2d1 --SHA-1 校验和 commit id Author: wiggin wiggin@gmail.com --作者跟邮箱概要信息 Date: Wed May 16 23:51:02 2018 +0800 --提交时间

v2 --commit的时候，使用-m选项说写一段概要说明 日常在使用commit的时候，-m选项所写得内容一定不能随便写 “修改了登陆的bug”--》“新增用户管理中心”

- git log -数字 表示查看最近几次的提交
- git log -p -2 显示最近两次提交的不同点
- git log --author 查看具体某个作者的提交
- git log --online 输出简要的信息
- git log --graph 以一个简单的线串联起整个提交历史
- git log 输出信息的定制

第十集 文件对比利器--git diff

- diff -->difference的缩写，用于比较差异
- 使用场景

  解决冲突 制作补丁
- git diff 不加任何参数 用于比较当前工作区跟暂存区的差异
- git diff --cached 或者--staged
- git diff HEAD
- git diff 分支名 查看当前分支跟指定的分支的差异
- git diff 分支名1 分支名2 查看两个指定分支(已提交的)的差异，分支2 跟分支1的差别
- git diff 文件名 查看指定文件的差异
- git diff commitid1 commitid2 用于列出两个历史提交的差异
- git diff --stat 用于罗列有变更的文件

第十一集 git更改状态

- 将不必要的文件add
- 上次提交觉得是错的
- 不想改变暂存区内容，只是想调整提交的信息
- 版本回滚
- git reset HEAD 文件名 移除不必要的添加到暂存区的文件

- git reset HEAD^ 或者 commitid 去掉上一次的提交
- git reset --soft HEAD^ 修改上次提交的信息吧即commit -m "修改这里的内容"
- git reset --soft 只是将HEAD引用指向指定的提交，工作区跟暂存区的内容不会改变
- git reset --mixed （默认选项）将HEAD指向指定的提交，暂存区的内容随之改变，工作区内容不变
- git reset --hard 将HEAD指向指定的提交，暂存区跟工作区都会改变

第十二集 分支合并及冲突解决

- git merge 分支名

  表示：拿指定的分支名与当前分支进行合并
- git diff --name-only --diff-filter=U 用于查看产生冲突的文件

第十三集 git的标签

- git tag 不加任何参数 表示显示标签（按字母序） 非按时间
- git tag 标签名 默认是给最近一次提交打上标签
- git tag 标签名 commitId 给响应的提交打上标签
- git show 标签名 显示该标签相关的那次提价的相关信息
- git tag -d 标签名 删除该标签
- git push 远程分支名 标签名 把某个标签（必须是本地已存在的，否则推动失败） 推送到远程分支
- 删除远程标签的步骤

  删除本地标签 git tag -d 标签名

  在删除远程的 git push origin :refs/tags/标签名

第十四集 gitignore

- 为什么要使用.gitignore 文件

  大量与项目无关的文件全推到远程仓库上，同步的时候会非常慢，且跟编辑器相关的一些配置推上去之后，别人更新也会受其影响。所以，我们使用该文件，对不必要的文件进行忽略，使其不被git追踪

  一把情况下，.gitignore文件，在项目一开始创建的时候就创建，并推送到远程服务器上。这样大家初次同步项目的时候，就是用到该文件，避免以后，团队成员把与项目无关的文件，传到远程服务器上

.log 表示忽略项目中所有以.log结尾的文件 123?.log 表示忽略项目中所有以123加任意字符的文件 /error.log 表示忽略项目中根目录中的error.log 这个文件 src/main/test/ 表示忽略/src/main/test/目录下的所有文件 *.class **/java/ 匹配所有java目录下的所有文件 !/error.log 表示在之前的匹配规则下，被命中的文件，可以使用!对前面的规则进行否定

- 对于已经提交到远程或本地仓库的文件，.gitignore配置之后不会生效。我们必须先删除本地暂存区里的文件，之后在加上.gitignore 文件，最后再把变更提交到远程仓库上。
- git rm --cached 文件名 从暂存区删除某个文件
- git rm -rf --cached 文件夹 表示递归删除暂存区该文件夹的所有东西

第二章 gitlab系列课程

第一集 虚拟机的安装

- 安装virtualbox

  https://www.virtualbox.org/

- 安装centos6.6
- 配置网络

  右键-->网络-->网卡2-->host-only cd /etc/sysconfig/network-scripts/ vi ifcfg-eth0 将此处改为yes ONBOOT=yes 此时可ping外网 cp ifcfg-eth0 ifcfg-eth1 加入以下内容 DEVICE=eth1 ONBOOT=yes BOOTPROTO=static IPADDR=192.168.56.1

  NETMASK=255.255.255.0

- 使用工具连接centos

  https://winscp.net/eng/download.php

  https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html

第二集 gitlab的简介

- gitlab是什么

  是一个用于仓库管理系统的开源项目，使用Git作为代码管理工具，并在此基础上搭建起来的web服务。 基础功能免费，高级功能收费

- 为什么要使用gitlab

基础功能开源，可自行搭建 可以进行权限控制，使得代码对部分人可见 gitlab使用方便，非常适合企业内部使用

第三集 gitlab的安装

- 在防火墙里开放http跟ssh端口

  yum install lokkit yum install curl openssh-server openssh-clients postfix cronie -y service postfix start chkconfig postfix on lokkit -s http -s ssh

- 添加gitlab仓库，并安装

  curl -sS http://packages.gitlab.cc/install/gitlab-ce/script.rpm.sh | sudo bash sudo yum install gitlab-ce

- 启动gitlab

  gitlab-ctl reconfigure vim /etc/gitlab/gitlab.rb 修改external_url为gitlab机子的ip+要使用的端口 如：http://192.168.56.101:8888 修改nginx['listen_port'] = 8888 重新配置gitlab并重启 gitlab-ctl reconfigure gitlab-ctl restart

- 配置防火墙

  vim /etc/sysconfig/iptables -A INPUT -m state --state NEW -m tcp -p tcp --dport 8888 -j ACCEPT service iptables restart

第四集 配置gitlab邮件服务

- 配置邮箱服务的用途

  有合并请求时，邮件通知 账号注册时，邮件验证 修改密码时，通过邮件修改

- 配置步骤
  - 开启QQ邮箱的smtp服务(不建议使用163邮箱，发几次之后，就不能发送)

    设置--》账户--》smtp--》密保验证--》验证成功返回一串字符串，形状如（ausdixersybgcgid）

    保存返回的字符串

  - 修改gitlab配置

    ```
    vim /etc/gitlab/gitlab.rb
    按/后输入smtp_enable，找到下面这一串文本，进行修改
    gitlab_rails['smtp_enable'] = true
    gitlab_rails['smtp_address'] = "smtp.qq.com"
    gitlab_rails['smtp_port'] = 465
    gitlab_rails['smtp_user_name'] = "1403780990@qq.com"
    gitlab_rails['smtp_password'] = "开通smtp时返回的字符"
    gitlab_rails['smtp_domain'] = "qq.com"
    gitlab_rails['smtp_authentication'] = "login"
    gitlab_rails['smtp_enable_starttls_auto'] = true
    gitlab_rails['smtp_tls'] = true

    user['git_user_email'] = "1403780990@qq.com"
    gitlab_rails['gitlab_email_from'] = '1403780990@qq.com'
    ```

```
按esc退出到命令行模式
之后:wq 保存并退出
gitlab-ctl reconfigure
```

- 测试邮件服务是否正常

  gitlab-rails console

  Notify.test_email('接收方邮件地址','邮件标题','邮件内容').deliver_now

  按回车，测试发送。

第五集 gitlab的账号注册及分组

- 开启注册邮箱验证

  admin area --》setting--》Sign-up restrictions--》勾选Send confirmation email on sign-up
- 当前用户

  root、123、wiggin
- 创建组

  首页->create a group(http://192.168.56.101:8888/dashboard/groups)

  访问级别

  Private：只有组成员才能看到

  Internal：只要登录的用户就能看到

  Public：所有人都能看到
- Guest：可以创建issue、发表评论，不能读写版本库
- Reporter：可以克隆代码，不能提交
- Developer：可以克隆代码、开发、提交、push
- Master：可以创建项目、添加tag、保护分支、添加项目成员、编辑项目
- Owner：可以设置项目访问权限 - Visibility Level、删除项目、迁移项目、管理组成员
- 键入命令：ssh-keygen -t rsa
- 提醒你输入key的名称，输入如id_rsa
- 在C:\Users\用户.ssh下产生两个文件：id_rsa和id_rsa.pub
- 用记事本打开id_rsa.pub文件，复制内容，在gitlab.com的网站上到ssh密钥管理页面，添加新公钥，随便取个名字，内容粘贴刚才复制的内容。

第六集 gitlab分支及标签的保护

- 为什么要保护分支？

  保护特定的分支不被随便合并，以免影响相应的分支
- 进入项目--> repository-->branches-->project setting

- 注意 能push 就能merge ，相应的权限把握好（ master分支设置只能masters可以合并 ）

第三章 敏捷持续集成

第一集 敏捷持续集成简介

- 什么是持续集成？

  持续集成是一种软件开发实践，即团队开发成员经常集成他们的工作，通过每个成员每天至少集成一次，也就意味着每天可能会发生多次集成。每次集成都通过自动化的构建（包括编译，发布，自动化测试）来验证，从而尽早地发现集成错误。

- 好处

  节省人力成本 加快软件开发进度 实时交付

- 整体流程

  成员通过git提交代码--》持续集成引擎来去代码并使用构建工具自动构建--》发布项目

- 重要组成部分

  git gitlab jenkins 持续集成引擎 maven 构建工具 sonarqube 代码质量管理 junit 自动执行单元测试 JDK Tomcat

  几个war--》微服务架构

第二集 jdk及maven的安装

- 版本说明

  jdk1.8（目前大部分公司内部使用的还是JDK8，大部分依赖java的工具或框架，对JDK8的支持度是最好的）
  maven3.5.3

- 安装步骤

```
vim /etc/profile
     在最下面，按i进入insert模式，添加一下内容
        JAVA_HOME=/usr/local/jdk1.8.0_91
        export JAVA_HOME
        CLASSPATH=.:$JAVA_HOME/lib
        export CLASSPATH
        PATH=$PATH:$JAVA_HOME/bin:$CLASSPATH
        export PATH
     按esc进入命令行模式，再按:wq保存退出
     激活配置
        source /etc/profile
```

```
5.解压maven
     tar -zxvf apache-maven-3.5.3-bin.tar.gz -C /usr/local/
6.配置maven环境变量
     vim /etc/profile
     在最下面，按i进入insert模式，添加一下内容
        MAVEN_HOME=/usr/local/apache-maven-3.5.3
        export MAVEN_HOME
        PATH=$PATH:$MAVEN_HOME/bin
        export PATH
    按esc进入命令行模式，再按:wq保存退出
     激活配置
        source /etc/profile
```

第三集 nexus的安装

- 下载nexus

  https://www.sonatype.com/download-oss-sonatype

- 上传到服务器/root/
- 解压

  tar -zxvf nexus-3.12.1-01-unix.tar.gz -C /usr/local/

- 修改配置文件

  vim /usr/local/nexus-3.12.1-01/etc/nexus-default.properties 修改对应的端口 修改防火墙 vim /etc/sysconfig/iptables -A INPUT -m state --state NEW -m tcp -p tcp --dport 8081 -j ACCEPT

- 浏览器打开并登陆

  http://192.168.56.102:8081/

  账号admin 密码 admin123 System Requirement: max file descriptors [4096] likely too low, increase to at least [65536].

- 修改ulimit

  vim /etc/security/limits.conf

  新增

  soft nofile 65535

  hard nofile 65535

- 配置开机自启动

  su - nexus -c '/usr/local/nexus-3.12.1-01/bin/nexus start'

第四集 nexus的使用

- 仓库类型

  proxy:代理仓库，用于代理远程仓库 group:仓库组，通常包含了多个代理仓库和宿主仓库，在项目中只要引入仓库组就可以下载到代理仓库和宿主仓库中的包 hosted:宿主仓库，内部项目、付费jar releases 发布内部release版本的仓库 snapshots 发布内部snapshots版本的仓库 third 自建第三方jar

- 配置代理

  选择阿里云http://maven.aliyun.com/nexus/content/groups/public/

- 本地maven配置

  修改maven目录下的conf/setting.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0
http://maven.apache.org/xsd/settings-1.0.0.xsd">
  <pluginGroups/>
  <proxies/>
  <servers>
    <server>
      <id>xdclass-releases</id>
      <username>admin</username>
      <password>admin123</password>
    </server>
    <server>
      <id>xdclass-snapshots</id>
      <username>admin</username>
      <password>admin123</password>
    </server>
  </servers>
  <mirrors/>
  <profiles>
    <profile>
      <id>xdclass</id>
      <activation>
        <activeByDefault>false</activeByDefault>
      </activation>
      <!-- 私有库地址-->
      <repositories>
        <repository>
          <id>xdclass</id>
          <url>http://192.168.56.101:8081/repository/maven-public/</url>
          <releases>
            <enabled>true</enabled>
          </releases>
          <snapshots>
            <enabled>true</enabled>
          </snapshots>
        </repository>
      </repositories>
      <!--插件库地址-->
      <pluginRepositories>
        <pluginRepository>
          <id>xdclass</id>
          <url>http://192.168.56.101:8081/repository/maven-public/</url>
```

```
            <releases>
              <enabled>true</enabled>
            </releases>
            <snapshots>
              <enabled>true</enabled>
            </snapshots>
          </pluginRepository>
        </pluginRepositories>
      </profile>
    </profiles>
    <activeProfiles>
      <activeProfile>xdclass</activeProfile>
    </activeProfiles>
  </settings>
```

- 修改编辑器中maven的配置，将配置指向setting.xml
- 修改pom

```
<!--pom.xml 远程仓库的配置  id要跟本地maven的setting.xml相同 -->
  <distributionManagement>
        <repository>
            <id>xdclass-releases</id>
            <name>Ruizhi Release Repository</name>
            <url>http://192.168.56.101:8081/repository/maven-releases/</url>
        </repository>

        <snapshotRepository>
            <id>xdclass-snapshots</id>
            <name>Ruizhi Snapshot Repository</name>
            <url>http://192.168.56.101:8081/repository/maven-snapshots/</url>
        </snapshotRepository>
    </distributionManagement>
```

- 测试是否nexus搭建成功

  pom添加本地没有的依赖，看nexus会不会代理 mvn deploy 看是否成功推送至nexus

第五集 编译安装mysql

- linux下安装mysql的方式

  yum 安装简单 自行编译安装 自由

- mysql下载

  http://mirrors.sohu.com/mysql/MySQL-5.7/mysql-5.7.17.tar.gz

- 编译安装mysql
  - 解压

    tar -zxvf mysql-5.7.17.tar.gz

  - 安装相应的依赖

    yum install make cmake gcc gcc-c++ bison bison-devel ncurses ncurses-devel autoconf automake wget

  - 下载boost

    mkdir /usr/local/boost

wget http://www.sourceforge.net/projects/boost/files/boost/1.59.0/boost_1_59_0.tar.gz --no-check-certificate

- 添加用户并创建相应目录存放数据

  useradd mysql cd /home/mysql/ mkdir data logs temp chown -R mysql:mysql data logs temp

- 执行cmake

```
cmake \
-DCMAKE_INSTALL_PREFIX=/usr/local/mysql \
-DMYSQL_UNIX_ADDR=/usr/local/mysql/mysql.sock \
-DDEFAULT_CHARSET=utf8 \
-DDEFAULT_COLLATION=utf8_general_ci \
-DWITH_MYISAM_STORAGE_ENGINE=1 \
-DWITH_INNOBASE_STORAGE_ENGINE=1 \
-DWITH_ARCHIVE_STORAGE_ENGINE=1 \
-DWITH_BLACKHOLE_STORAGE_ENGINE=1 \
-DWITH_MEMORY_STORAGE_ENGINE=1 \
-DWITH_READLINE=1 \
-DENABLED_LOCAL_INFILE=1 \
-DMYSQL_DATADIR=/home/mysql/data \
-DMYSQL_USER=mysql \
-DMYSQL_TCP_PORT=3306 \
-DWITH_BOOST=/usr/local/boost
```

- 编译安装

  make 进行编译 make install 安装

- 修改mysql安装目录权限

  chown -R mysql:mysql /usr/local/mysql

- 初始化mysql

  mysqld --initialize --user=mysql --basedir=/usr/local/mysql --datadir=/home/mysql/data

  产生密码 eqK:iH;+S6dC

- 删除/etc下的my.cnf

  rm /etc/my.cnf

- 复制服务启动脚本

  cp /usr/local/mysql/support-files/mysql.server /etc/init.d/mysql

- 启动 MySQL 服务

  service mysql start

- 设置mysql服务开机自启动

  chkconfig mysql on

- 登陆mysql并设置可远程登陆

  mysql -u root -p 回车 粘贴初始化时产生的临时密码 eqK:iH;+S6dC GRANT ALL PRIVILEGES ON . TO 'root'@'%' IDENTIFIED BY 'xdclass' WITH GRANT OPTION;此时不让改，提示要先设置下密码 SET PASSWORD = PASSWORD('xdclass'); ALTER USER 'root'@'localhost' PASSWORD EXPIRE NEVER; flush privileges; exit； 重新登陆，使用刚刚设置的密码 mysql -u root -p 登陆完成之后，即可设置允许远程登陆 GRANT ALL PRIVILEGES ON . TO 'root'@'%' IDENTIFIED BY 'xdclass' WITH GRANT OPTION;

- 开启防火墙端口

  vim /etc/sysconfig/iptables

  加入 -A INPUT -m state --state NEW -m tcp -p tcp --dport 3306 -j ACCEPT

- 使用连接工具测试远程连接

第六集 代码质量管理平台--sonarqube安装

- 前置依赖

  mysql 5.6 or 5.7 jdk 1.8

- 下载

  https://www.sonarqube.org/

- 安装unzip并解压sonarqube并移动到/usr/local

  yum install unzip unzip sonarqube-6.7.4.zip mv sonarqube-6.7.4 /usr/local/

- mysql里新增数据库

  CREATE DATABASE sonar DEFAULT CHARACTER SET utf8;

- 修改sonarqube相应的配置

  vim /usr/local/sonarqube-6.7.4/conf/sonar.properties sonar.jdbc.username=root sonar.jdbc.password=wiggin sonar.jdbc.url=改成步骤4创建的库名 sonar.web.context=/sonar sonar.web.host=0.0.0.0

- 新增用户，并将目录所属权赋予该用户

  useradd sonar chown -R sonar:sonar sonarqube-6.7.4/

- 启动

  su soanr /usr/local/sonarqube-6.7.4/bin/linux-x86-64/sonar.sh start

- 界面访问

  http://192.168.56.101:9000/sonar 开启防火墙 vim /etc/sysconfig/iptables 新增9000端口 在界面产生令牌，并将项目代码使用以下命令进行提交由sonarqube进行分析，完成后，查看相应的分析结果

- mvn sonar:sonar \ -Dsonar.host.url=http://192.168.56.101:9000/sonar \ -Dsonar.login=830edadfcb2c6326b1c6e2110f43c9f74d008450

第七集 Jenkins安装课堂笔记及常见问题分析定位

- 前置条件

  JDK、tomcat

- 安装tomcat

  下载地址： https://tomcat.apache.org/download-90.cgi useradd tomcat --新增一个名为tomcat的用户 passwd tomcat --给tomcat用户设置密码 tar -zxvf apache-tomcat-9.0.8.tar.gz -C /usr/local/ --将tomcat解压到相应目录 chown -R tomcat:tomcat /usr/local/apache-tomcat-9.0.8 --将整个目录的所属权转移给tomcat用户、tomcat组

- 安装Jenkins

  下载地址： https://jenkins.io/download/ 将Jenkins上传到tomcat的webapp目录 chown tomcat:tomcat Jenkins.war 修改Jenkinswar包为tomcat用户所有 启动tomcat --通过浏览器无法访问tomcat 1.看tomcat是否存活 ps aux | grep tomcat 2.看端口 netstat -tlun 看到8080端口已经使用了 3.检查防火墙 vim /etc/sysconfig/iptables 加入8080 4.查看tomcat日志 --》出现异常，地址已经被使用 5.关闭tomcat --》 查看端口（步骤2）--》发现8080依旧存在 6.断定8080被其他程序占用 --》netstat -tlunp | grep 8080 -->看到被

gitlab相关的程序使用了 7.修改tomcat端口 vim conf/server.xml ,找到8080 --》将8080改成不被占用的9999端口 8.防火墙开启9999端口 9.可以正常访问tomcat

浏览器打开http://192.168.56.101:9999/jenkins more /home/tomcat/.jenkins/secrets/initialAdminPassword 将里面的内容复制粘贴 此时发现提示Jenkins离线 访问 http://192.168.56.101:9999/jenkins/pluginManager/advanced 拉到最底下，将https--》改成http，之后提交 重启tomcat 浏览器打开http://192.168.56.101:9999/jenkins more /home/tomcat/.jenkins/secrets/initialAdminPassword 选择默认安装

第八集 Jenkins插件安装及配置课堂笔记

- 插件安装

  系统管理--》插件管理 1.安装Maven Integration plugin 2.安装SonarQube Scanner for Jenkins 3.Publish Over SSH --发布到远程服务器

- 系统配置

  系统管理--》全局工具配置 1.配置jdk 2.配置maven 3.配置sonar 4.邮件配置 系统管理--》系统设置--》邮件通知--》 smtp服务器 smtp.qq.com 用户默认邮件后缀 @qq.com 勾选ssl Reply-To Address发件者邮箱 之后测试一下配置，无误即可

- 配置gitlab授权

  Credentials--》system--》Global credentials

- 配置免密登陆

  yum -y install openssh-clients ssh-keygen -t rsa -- 产生私钥 配置git登陆 将Jenkins所在机子的公钥 more ~/.ssh/id_rsa.pub 的内容拷贝到gitlab项目上

第九集 Jenkins仪表盘简介

- 用户 --显示Jenkins里的用户
- 构建历史 --以时间轴的形式，显示项目的构建历史
- 系统管理 --跟Jenkins相关的配置都在里面

  3.1 系统设置 全局设置相关的都在里面(maven、邮件、ssh服务器等都在里面配置) 3.2 全局安全配置 用户权限、是否允许用户登录等配置 3.3 configure credentials 配置证书相关的 3.4 全局工具配置 JDK Git Maven 等都在里面配置 3.5 读取配置 放弃当前配置，而读取配置文件 3.6 管理插件 所有的插件都是在此处管理的，安装，升级 3.7 系统信息 系统相关的信息 3.8 系统日志 系统日志，帮助定位问题 3.9 负载统计 3.10 Jenkins cli 3.11 脚本命令行 3.12 管理节点 3.13 关于Jenkins 3.14 manage old data 3.15 管理用户

  Jenkins用户的管理

- 我的视图 --我们配置的要构建的项目
- Credentials --证书相关，授权相关

第十集 持续集成--手动集成

- 详见视频

第四章 持续集成实战

第一、二集--Jenkins本地持续集成

- nohup 的用途就是让提交的命令忽略 hangup 信号，那什么叫做hangup信号？这里给出了答案

  0：标准输入 1：标准输出，2：标准错误

- --本地手动构建
  - 新建job并配置

    General --可不配 源码管理 --按项目所使用的源码管理选择，课程使用git 填写项目地址，Credentials 选择配置好的认证 选择分支 可以是项目中的任意分支 构建触发器 触发远程构建 (例如,使用脚本) 其他工程构建后触发 -- 在Jenkins中其他项目构建之后，触发本项目构建，一般用于项目间有依赖关系，一方修改，另一方需实时感知 定时构建 --定时进行构建，无论是否有变更 （类似cron表达式） GitHub hook trigger for GITScm polling --github的hook触发构建,一般不使用 轮询 SCM --设置定时检查源码变更，有更新就构建（类似cron表达式）

    ```
    定时表达式含义
        * * * * * --五个字段
        分 时 天 月 周
    构建环境
        Delete workspace before build starts：在构建之前清空工作空间
        Abort the build if it's stuck：如果构建出现问题则终止构建
        Add timestamps to the Console Output：给控制台输出增加时间戳
        Use secret text(s) or file(s)：使用加密文件或者文本
    执行shell
    #!/bin/bash
    mv target/*.jar /root/demo/
    cd /root/demo
    BUILD_ID=
    java -jar springboot-demo.jar >log 2>&1 &
    ```

- 本地gitlab触发构建
- 配置gitlab webhook

  系统管理员登陆 http://192.168.56.101:8888/admin/application_settings settings Outbound requests 勾选 Allow requests to the local network from hooks and services

- sonarqube整合

  required metadata

  ```
  #projectKey项目的唯一标识，不能重复
  ```

```
sonar.projectKey=xdclass
sonar.projectName=xdclass

sonar.projectVersion=1.0
sonar.sourceEncoding=UTF-8
sonar.modules=java-module

# Java module
java-module.sonar.projectName=test
java-module.sonar.language=java
# .表示projectBaseDir指定的目录
java-module.sonar.sources=src
java-module.sonar.projectBaseDir=.
java-module.sonar.java.binaries=target/
```

第三集 jenkins blue ocean 与 pipeline

- 详见视频

第四集 分布式构建

- 详见视频