第十四章　GitLab 与 Jenkins 结合构建持续集成(CI)环境

本节所讲内容：

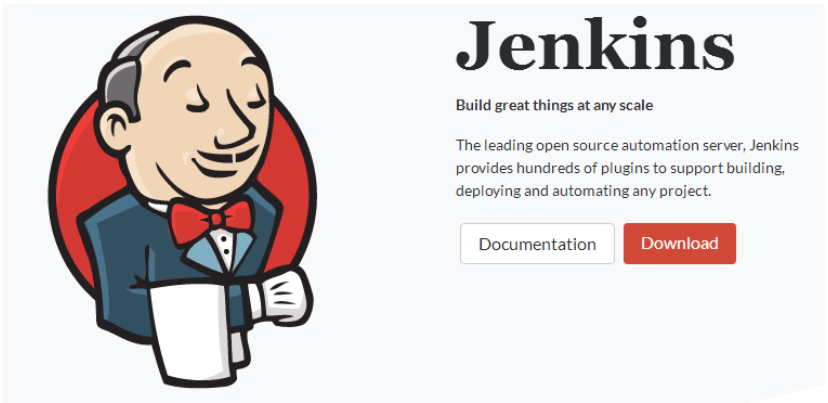
14.1　持续集成概述及运行流程

14.2　搭建 GitLab 平台及使用方法

14.3　安装 git 客户端使用 gitlab

14.4　搭建 Jenkins 实现持续集成

实战：GitLab 与 Jenkins 结合构建持续集成(CI)环境



14.1　持续集成概述及运行流程

14.1.1　持续集成概述

持续集成概述：持续集成（Continuous integration）持续集成是指开发者在代码的开发过程中，可以频繁的将代码部署集成到主干，并进程自动化测试。

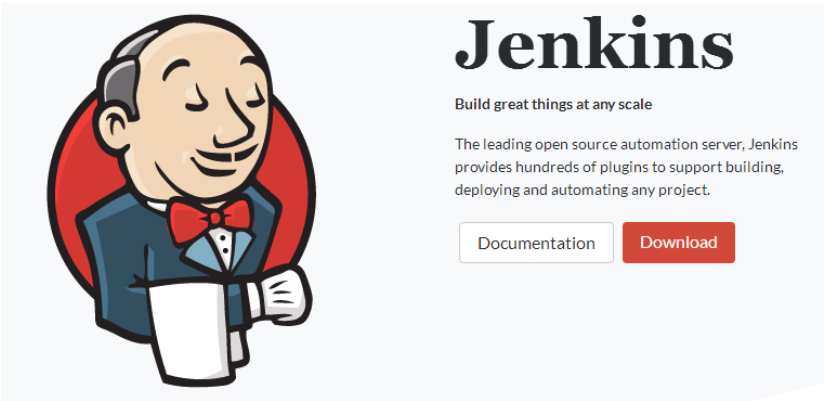
持续交付：持续交付指的是在持续集成的环境基础之上，将代码部署到预生产环境。

持续部署：在持续交付的基础上，把部署到生产环境的过程自动化。

14.1.2　jenkins 和 Gitlab 概述

Jenkins 概述：是一个开源软件项目，是基于 Java 开发的一种持续集成工具，用于监控持续重复的工作，旨在提供一个开放易用的软件平台，使软件的持续集成变成可能。

网方网站：https://jenkins.io/



GitLab 概述：

是一个利用 Ruby on Rails 开发的开源应用程序，实现一个自托管的 Git 项目仓库，可通过 Web 界面进行访问公开的或者私人项目。Ruby on Rails 是一个可以使你开发、部署、维护 web 应用程序变得简单的框架。

GitLab 拥有与 Github 类似的功能，能够浏览源代码，管理缺陷和注释。可以管理团队对仓库的访问，它非常易于浏览提交过的版本并提供一个文件历史库。它还提供一个代码片段收集功能可以轻松实现代码复用，便于日后有需要的时候进行查找。

GitLab 中文网：https://www.gitlab.cc/installation/#centos-7

14.1.3 GitLab 和 GitHub 的区别

GitHub 和 GitLab 的区别：

相同点: 二者都是基于 web 的 Git 仓库，在很大程度上 GitLab 是仿照 GitHub 来做的，它们都提供了分享开源项目的平台，为开发团队提供了存储、分享、发布和合作开发项目的中心化云存储的场所。

不同点：

1、GitHub 如果要使用私有仓库，是需要付费的。GitLab 可以在上面创建私人的免费仓库。

2、GitLab 让开发团队对他们的代码仓库拥有更多的控制，相比于 GitHub，它有不少的特色：允许免费设置仓库权限；允许用户选择分享一个 project 的部分代码；允许用户设置 project 的获取权限，进一步的提升安全性；可以设置获取到团队整体的改进进度；通过 innersourcing 让不在权限范围内的人访问不到该资源。

总结：从代码私有性方面来看，有时公司并不希望员工获取到全部的代码，这个时候 GitLab 无疑是更好的选择。但对于开源项目而言，GitHub 依然是代码托管的首选。

git 相关概念：

git	是一种版本控制系统，是一个命令，是一种工具
gitlib	是用于实现 git 功能的开发库
github	是一个基于 git 实现的在线代码托管仓库，包含一个网站界面，向互联网开放
gitlab	是一个基于 git 实现的在线代码仓库托管软件，一般用于在企业内部网络搭建 git 私服

注： gitlab-ce 社区版 ； gitlab-ee 是企业版，收费

持续集成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 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。

14.2 搭建 GitLab 平台

实验环境： centos7.4 虚拟机需要 6G，不然后期运行时，内存不够用，直接报错。

14.2.1 安装 Gitlab 需要的组件：

[root@xuegod63 ~]#yum install curl policycoreutils openssh-server openssh-clients postfix -y

默认，使用 Postfix 发送邮件

[root@xuegod63 ~]#systemctl enable sshd

[root@xuegod63 ~]#systemctl start sshd

[root@xuegod63 ~]#systemctl enable postfix

[root@xuegod63 ~]#systemctl start postfix

[root@xuegod63 ~]# iptables -F #清空规则

[root@xuegod63 ~]# systemctl stop firewalld

[root@xuegod63 ~]# systemctl disable firewalld

禁止防火墙，就不用执行下面两条命令：

[root@xuegod63 ~]#firewall-cmd --permanent --add-service=http

[root@xuegod63 ~]#systemctl reload firewalld

14.2.2 安装 gitlab

下载 gitlab 的两种方法：

方法 1：使用 yum 下载太慢。直接使用迅雷下载以下链接：

https://mirrors.tuna.tsinghua.edu.cn/gitlab-ce/yum/el7/gitlab-ce-10.2.3-ce.0.el7.x86_64.rpm

[root@xuegod63 ~]# rpm -ivh gitlab-ce-10.2.3-ce.0.el7.x86_64.rpm #安装

方法 2：配置 yum 源，使用 yum 安装：

[root@xuegod63 ~]# yum install gitlab-ce -y #安装太慢。下面使用清华的源：

[root@xuegod63 yum.repos.d]# cat gitlab_gitlab-ce.repo

[gitlab-ce]

name=gitlab-ce

baseurl=http://mirrors.tuna.tsinghua.edu.cn/gitlab-ce/yum/el7

repo_gpgcheck=0

gpgcheck=0

enabled=1

gpgkey=https://packages.gitlab.com/gpg.key
[root@xuegod63 ~]# yum install gitlab-ce -y

方法 3：本地上传：我使用这种
将下载的软件包 gitlab-ce-10.2.3-ce.0.el7.x86_64.rpm 上传到 linux 系统中。
[root@xuegod63 ~]# rpm -ivh gitlab-ce-10.2.3-ce.0.el7.x86_64.rpm

配置 gitlab 域名：
[root@xuegod63 ~]# vim /etc/gitlab/gitlab.rb #修改 gitlab 外部访问地址
改：13 external_url 'http://gitlab.example.com'
为：13 external_url 'http://192.168.1.63'
注：这里必须修改，不然后后期访问时，用户到地址是：http://gitlab.example.com/xxxx ，根本不能访问。 修改后获得是： http://192.168.1.63/xxxx

应用重新配好的配置并重启 GitLab
[root@xuegod63 ~]# gitlab-ctl reconfigure #重新配置应用程序。修改了 gitlab 服务配置文件后，都需要执行一下这个命令。让各个服务的配置文件，重新加载一下配置文件。这里等个 2 分钟左右。
。。。
Running handlers:
Running handlers complete
Chef Client finished, 2/501 resources updated in 37 seconds
gitlab Reconfigured!

[root@xuegod63 ~]# gitlab-ctl status
[root@xuegod63 ~]# gitlab-ctl status #可以使用 gitlab-ctl 管理 gitlab，例如查看 gitlab 状态：
run: gitlab-workhorse: (pid 3275) 169s; run: log: (pid 3151) 280s
run: logrotate: (pid 3169) 273s; run: log: (pid 3168) 273s
run: nginx: (pid 3157) 279s; run: log: (pid 3156) 279s
run: postgresql: (pid 3009) 349s; run: log: (pid 3008) 349s
run: redis: (pid 2926) 360s; run: log: (pid 2925) 360s
run: sidekiq: (pid 3142) 287s; run: log: (pid 3141) 287s
run: unicorn: (pid 3110) 293s; run: log: (pid 3109) 293s
[root@xuegod63 config]# netstat -antup | grep :80
tcp 0 0 127.0.0.1:8080 0.0.0.0:* LISTEN 10864/unicorn maste
tcp 0 0 0.0.0.0:80 0.0.0.0:* LISTEN 10729/nginx: master
默认使用 nginx 做为 web 界面。
注：如果后期 web 界面访问时，总报 502，要把防火墙清空规则，另外内存要大于 4G，不然后内存不足，也报 502

502

Whoops, GitLab is taking too much time to respond.

Try refreshing the page, or going back and attempting the action again.

Please contact your GitLab administrator if this problem persists.

[root@xuegod63 ~]# iptables -F #清空规则

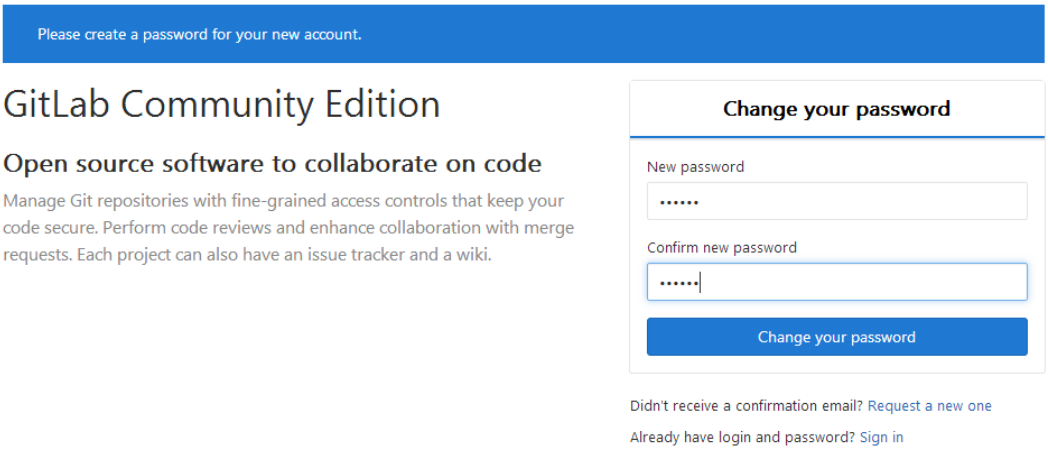
[root@xuegod63 ~]# free -m #已经使用 4G 以上内存

	total	used	free	shared	buff/cache	available
Mem:	5817	4187	133	80	1496	1198

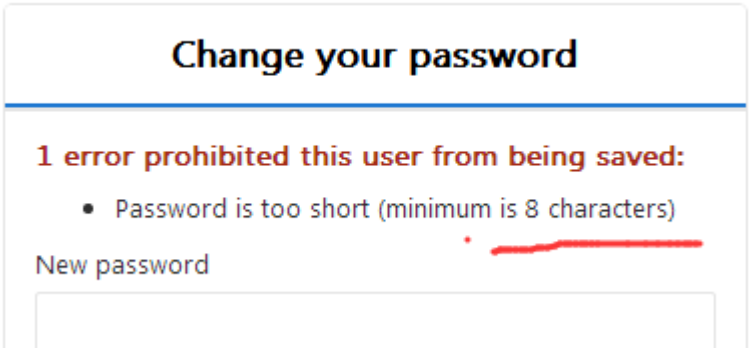
14.2.3 登录 gitlab

http://192.168.1.63/users/sign_in

第一次登录 gitlab，需要为 root 用户修改密码，root 用户也是 gitlab 的超级管理员，输入新密码：xuegod.cn



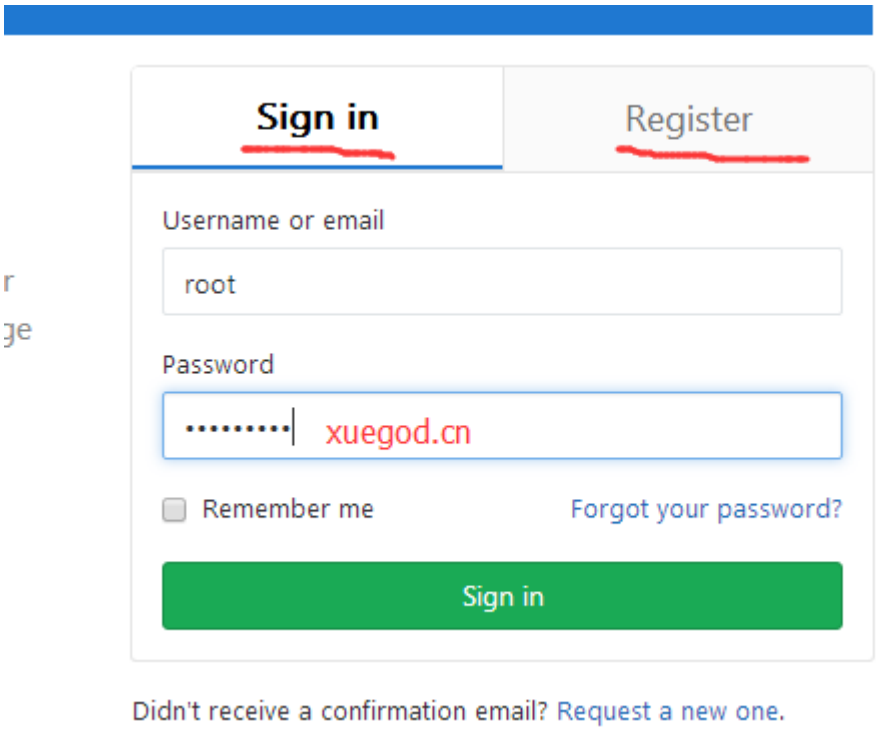
如果密码太简单，将报错：

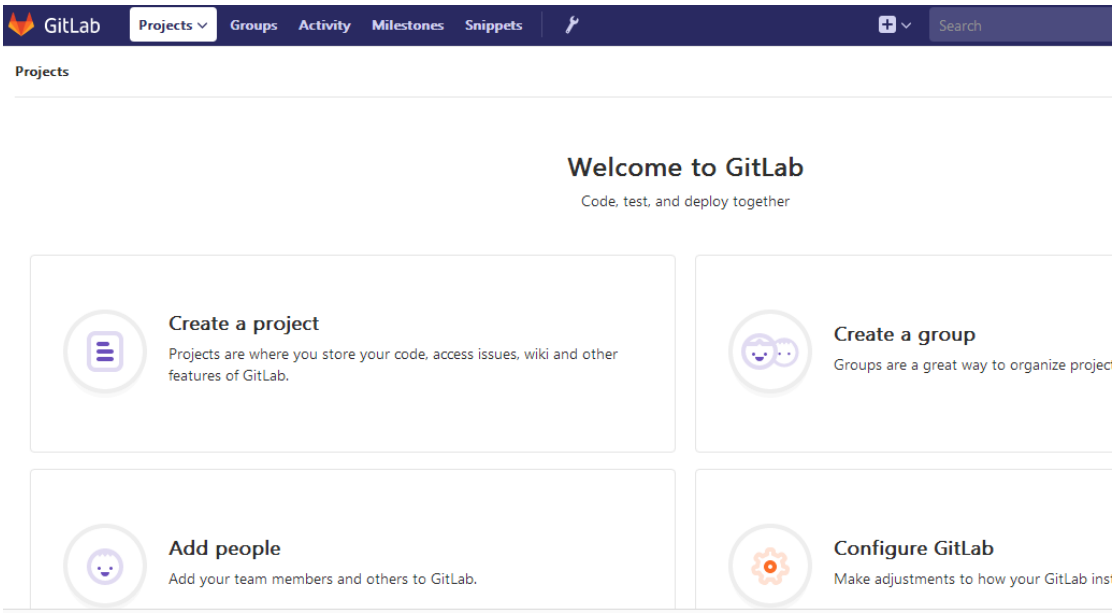


改成密码后， 登录一下：

使用 root 用户和刚才创建的密码登录 gitlab：

登录： http://192.168.1.63/ 用户名： root 密码; xuegod.cn





到此，gitlab 搭建成功。

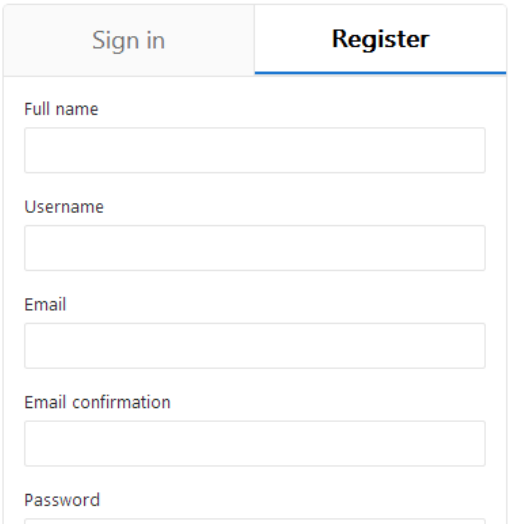
14.2.4 管理 gitlab

关闭 gitlab： # gitlab-ctl stop
启动 gitlab： # gitlab-ctl start
重启 gitlab： # gitlab-ctl restart

gitlab 主配置文件：/etc/gitlab/gitlab.rb //可以自定义一些邮件服务等
日志地址：/var/log/gitlab/ // 对应各服务
服务地址：/var/opt/gitlab/ // 对应各服务的主目录
仓库地址：/var/opt/gitlab/git-data //记录项目仓库等提交信息
重置配置：gitlab-ctl reconfigure //不要乱用，会重置为最原始的配置的
重启服务：gitlab-ctl stop/start/restart //启动命令

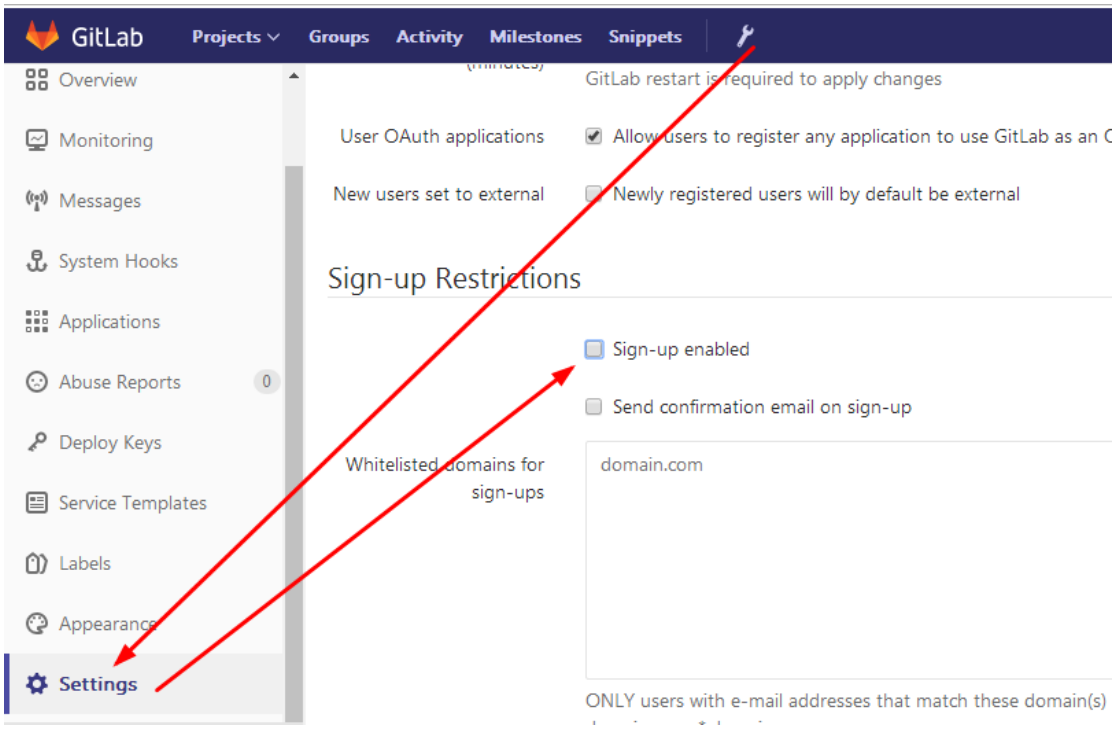
14.2.5 关闭 gitlab 注册功能

默认情况下可以直接注册账号，我里不需要注册功能，可以关闭。



以 root 用户登录：http://192.168.1.63/users/sign_in

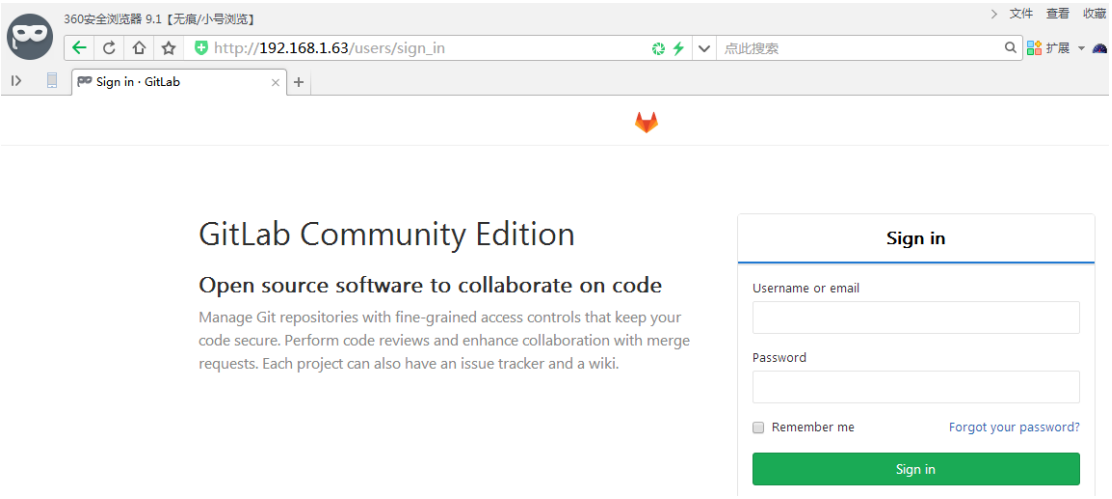
点 Admin area -》 setting -》 取消 sign-up enabled 标签前对勾



在此网页的最后，点 save ：

测试，使用无痕浏览器进行登录，发已经关闭了注册功能：

http://192.168.1.63/users/sign_in



14.2.6 centos7 部署 汉化版 gitlab 10.2.3

说明：gitlab 中文社区版的项目，v7-v8.8 是由 Larry Li 发起的“GitLab 中文社区版项目”（https://gitlab.com/larryli/gitlab），从 v8.9 之后，@xhang 开始继续该汉化项目（https://gitlab.com/xhang/gitlab）。

[root@gitlab ~]#git clone https://gitlab.com/xhang/gitlab.git #下载汉化补丁

方法 2：上传本地 gitlab-patch-zh.tat.gz 到 linux，我使用这个。

[root@xuegod63 ~]# tar zxvf gitlab-patch-zh.tat.gz

[root@xuegod63 ~]# cat /root/gitlab/VERSION #查看该汉化补丁的版本

1、停止 gitlab 服务

[root@xuegod63 ~]# gitlab-ctl stop

2、切换到 gitlab 汉化包所在的目录（即步骤二获取的汉化版 gitlab）

cd /root/gitlab

3、 比较汉化标签和原标签，导出 patch 用的 diff 文件到/root 下

[root@xuegod63 gitlab]# git diff v10.2.3 v10.2.3-zh > ../10.2.3-zh.diff

4、 将 10.2.3-zh.diff 作为补丁更新到 gitlab 中

[root@xuegod63 gitlab]# patch -d /opt/gitlab/embedded/service/gitlab-rails -p1 < /root/10.2.3-zh.diff #这个目录下存储着关于 web 前端相关的页面

[root@xuegod63 gitlab]# gitlab-ctl restart #重启服务，等 1 分钟，再去访问 web 页面。访问太快会显示 502 错误。

5、登录汉化版本：

http://192.168.1.63/profile

汉化方法 2：

gitlab-ctl stop

cp /home/local/gitlab/* /opt/gitlab/embedded/service/gitlab-rails/ -rf

复制时可能不断提示是否要覆盖，这时可能是系统每次执行 cp 命令时，其实是执行了 cp -i 命令的别名。出现这种情况可以修改~/.bashrc，在“alias cp=’cp-i’”前加#注释即可。

14.2.6 gitlab 日常使用

一、新建项目

二、创建用户

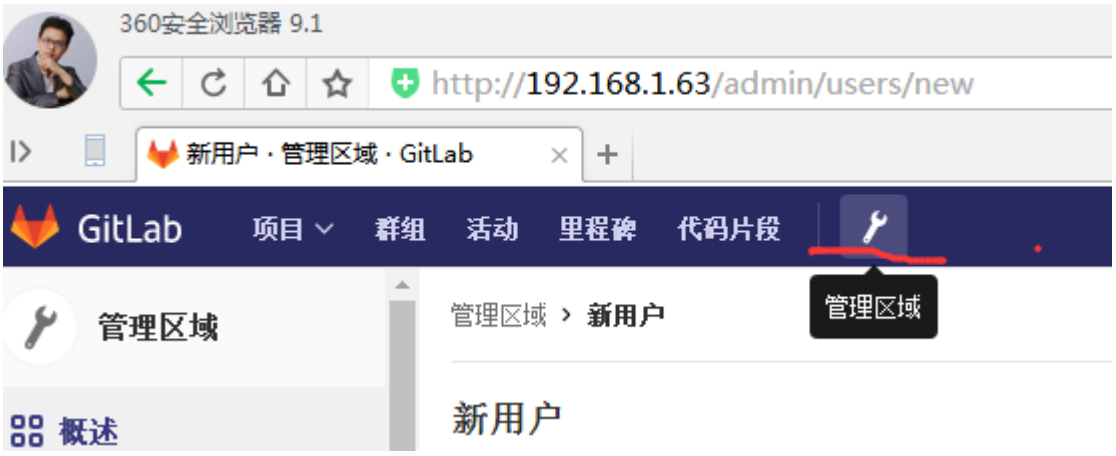
三、重置用户密码

四、删除用户

一、新建项目

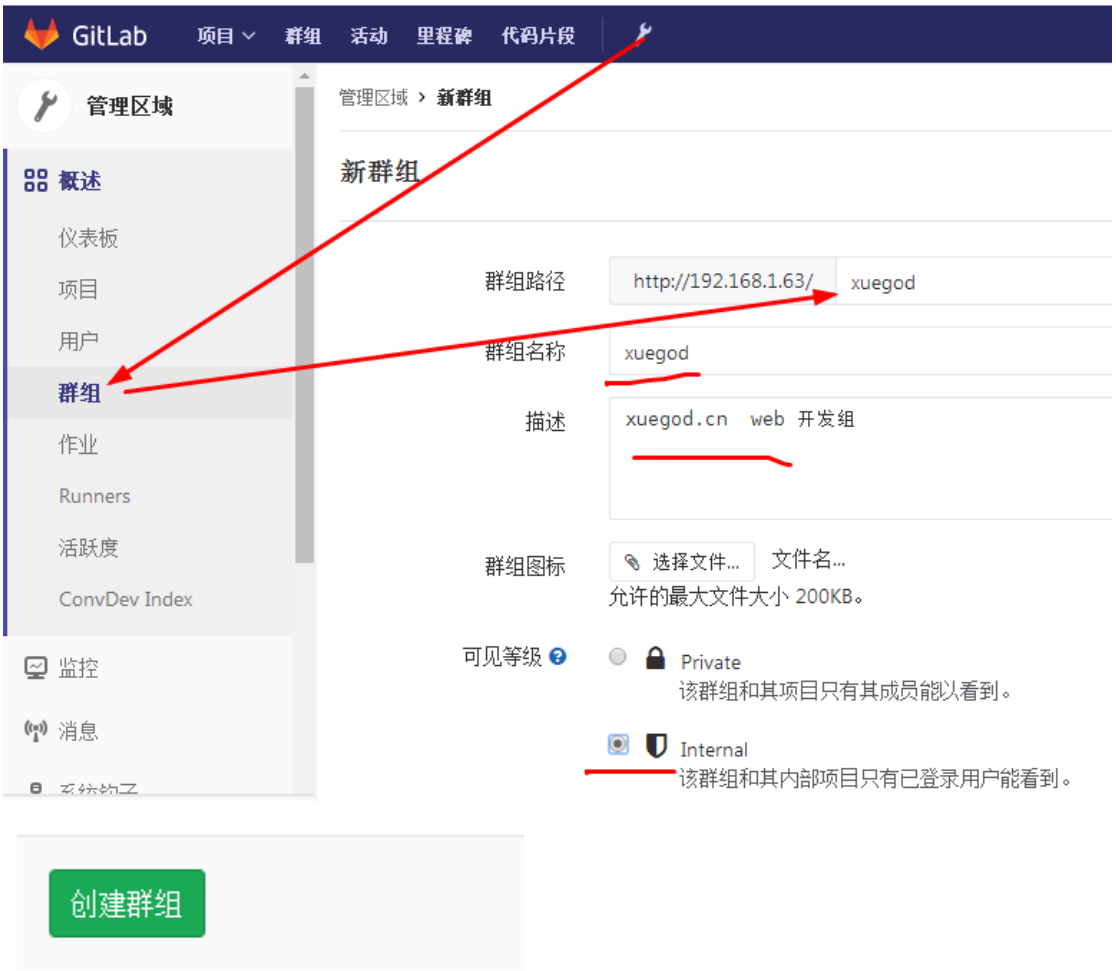
1、新建项目前，先创建项目所在的组（也就是说这个项目文件是保存在哪个组里）

选择 Admin area



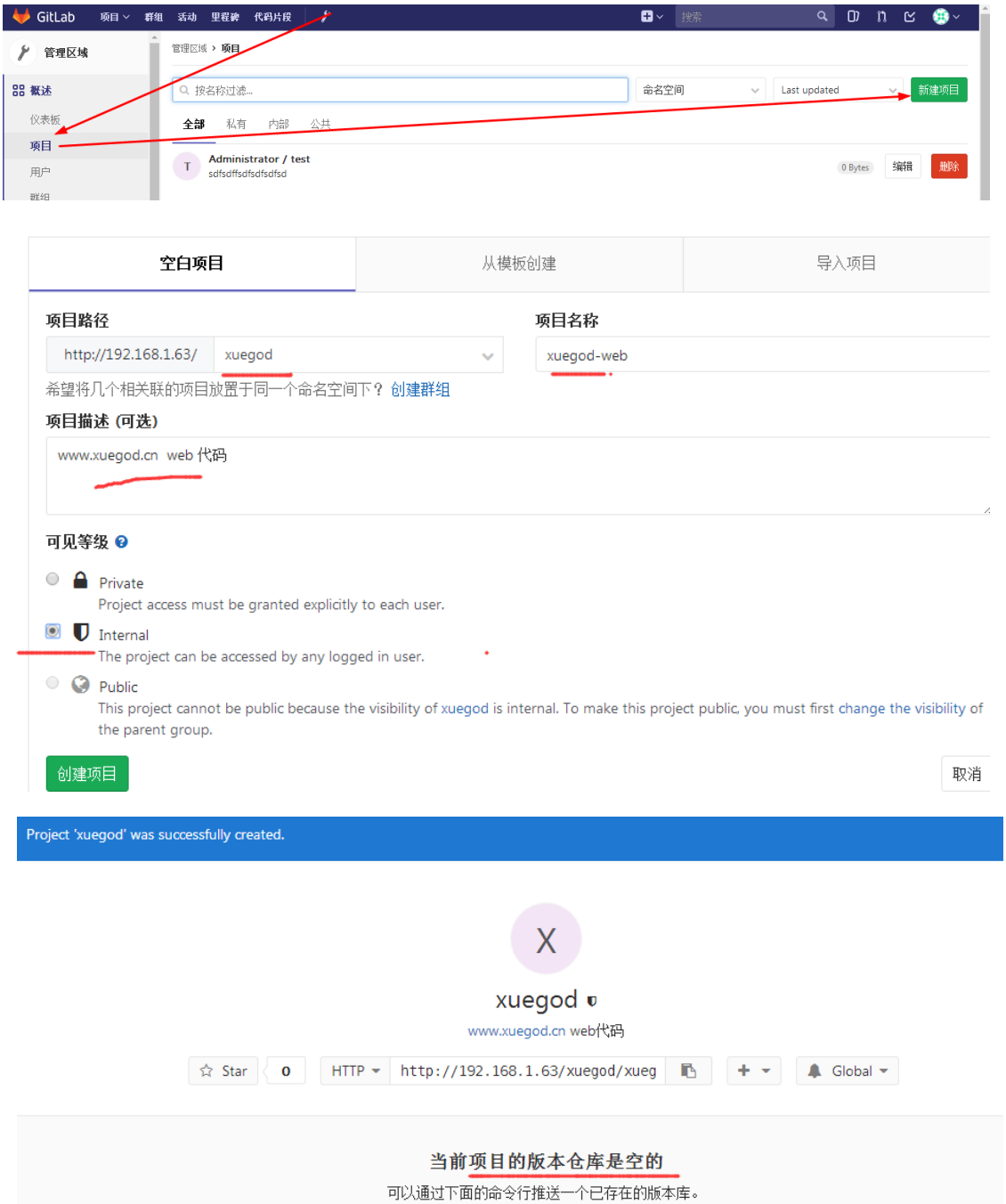
2、选择 Groups——New Group

http://192.168.1.63/projects/new



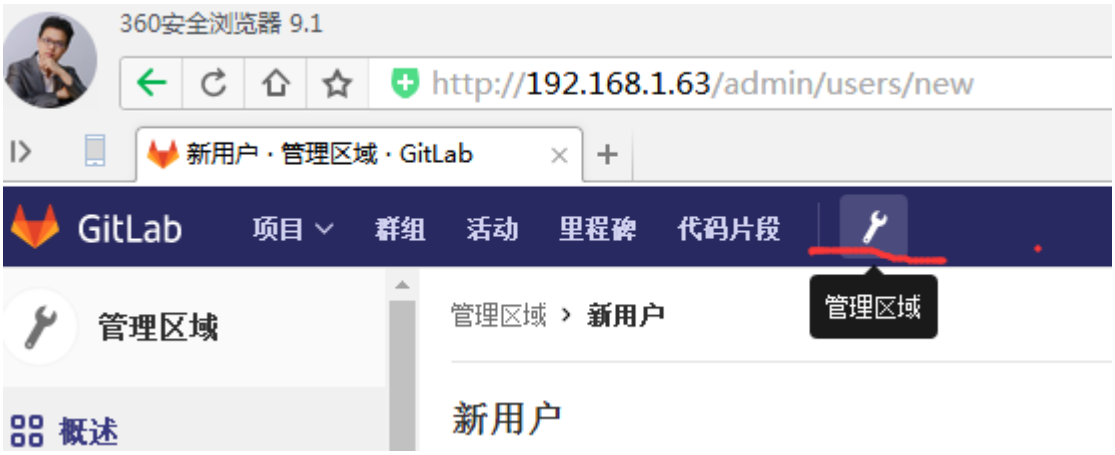
4、选择 Create New Projcet ，然后在输入项目名称，同时在 namespace 里选择刚才创建的组

www.xuegod.cn web 代码



二、创建用户

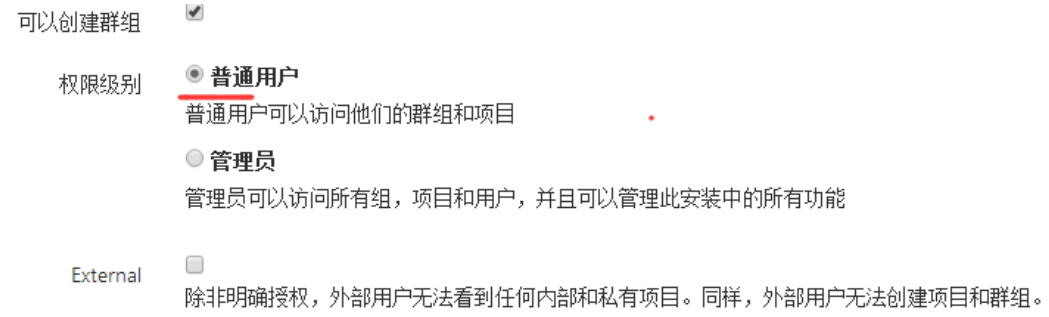
1、选择 Admin area



2、选择 Users——New User

3、其中 Name 为对方的中文名，Username 是登录用户名，一般可以设置成邮箱的前缀，Email 为公司邮箱。

5、项目现在默认即可，创建一个普通用户。



7、信息输入完成后，选择 Create user。 基本资料可以不用写。

三、重置新创建的用户 jianmingbasic 的密码

登录邮件：jianmingbasic@163.com

点开设置自己的初始密码：密码必须 8 位以一上。我这里是： xuegodlinux

生成密码后，登录：

方法 2： 修改密码

1、选择 Admin area -》用户-》选中用户-》编辑

设置新的密码是： xuegod.cn

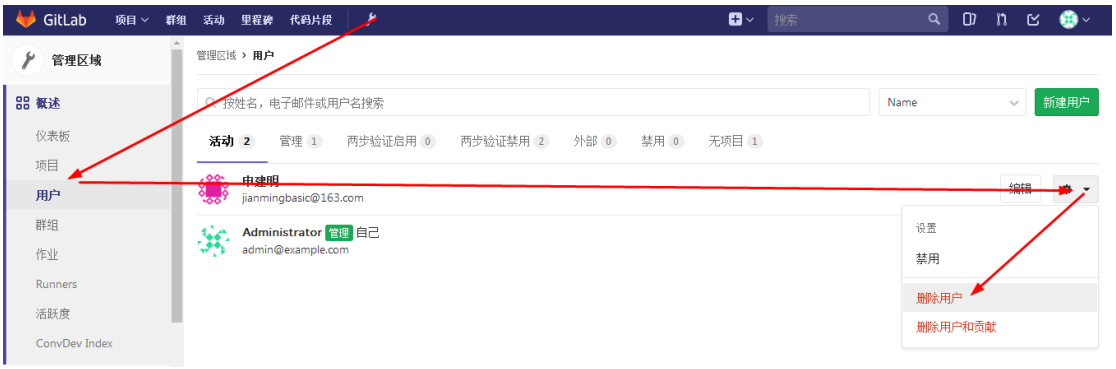
点保存后。

4、使用 jianmingbasic 登录 http://192.168.1.63/ 时， 还会弹出修改密码选项：
新密码：xuegodlinux

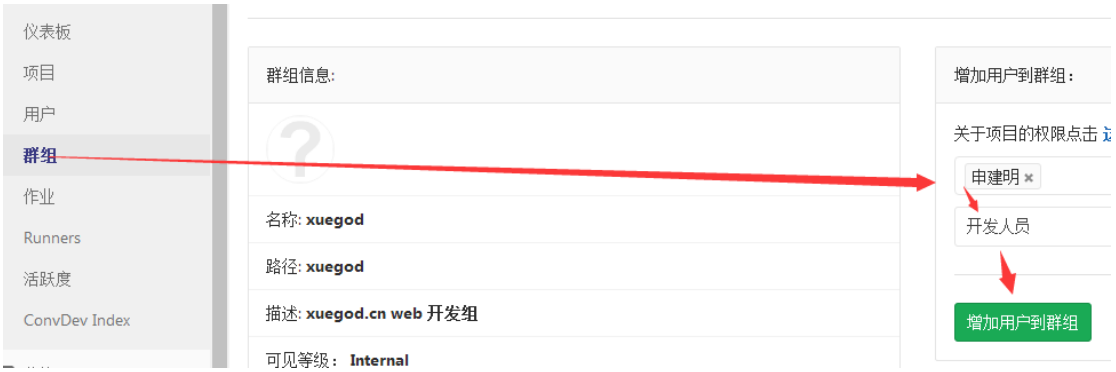
四、删除用户

当对方离职时候，为了安全起见，需要删除对方的 gitlab 权限，避免机密信息丢失，操作方法如下：

1、选择 Admin area ，选择 User，删除用户 。 我们这里先不删除，后期要用

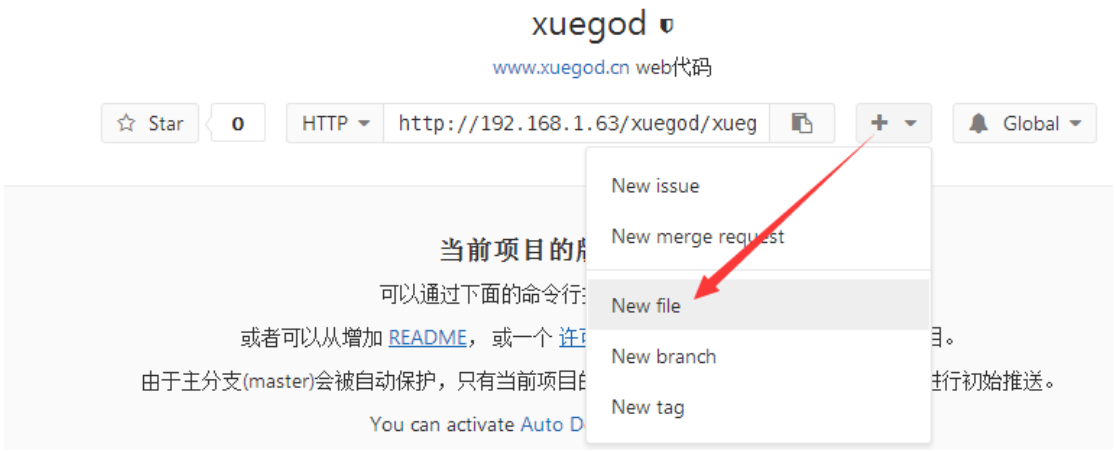


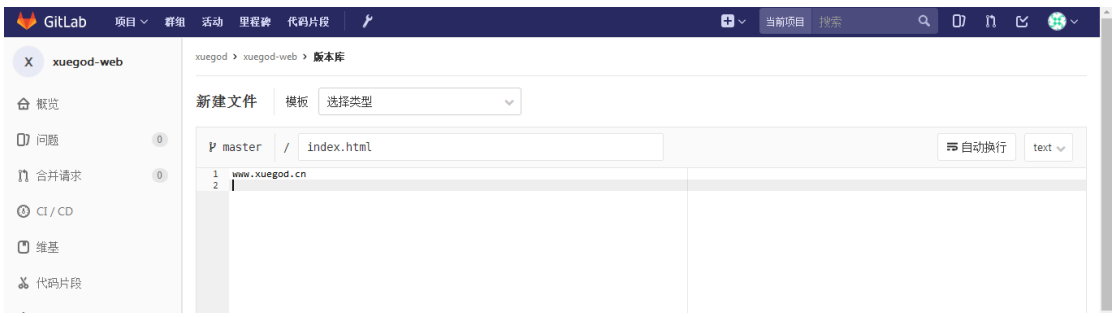
2、把用户 jianmingbasic 添加到 xuegod 组中，这样后期就可以提交这个组中项目的代码了。

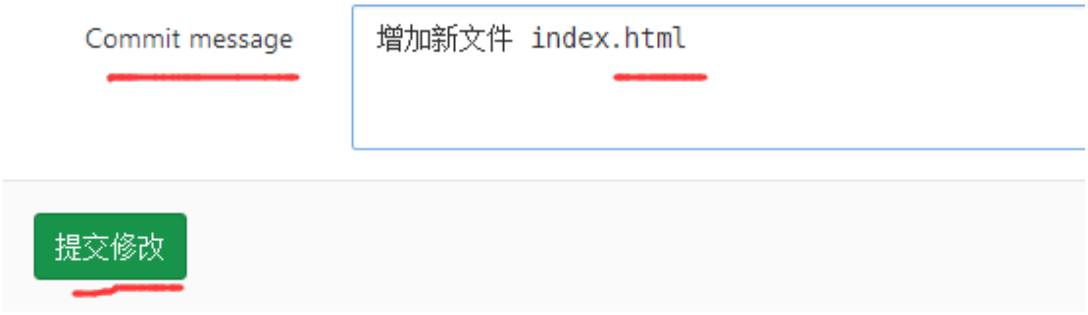


五、在项目中添加一个文件 index.html

http://192.168.1.63/xuegod/xuegod-web







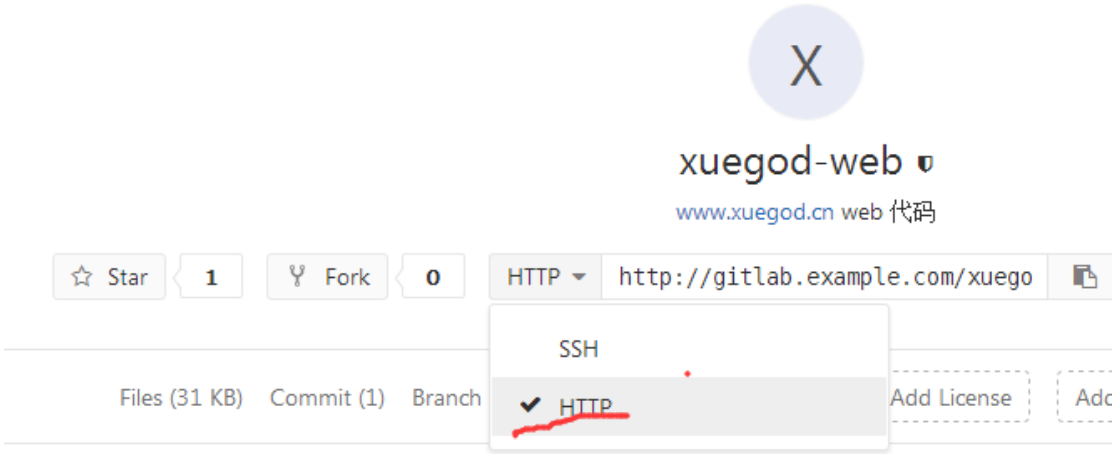
14.3 安装 git 客户端使用 gitlab

14.3.1 安装 git 并 clone 代码

 # yum install git -y

 # git clone jianmingbasic@192.168.1.63:xuegod/xuegod-web.git #下载地址，可以从这里获得

也可以使用 http 协议进行下载：



[root@xuegod63 test]# git clone http://192.168.1.63/xuegod/xuegod-web.git
正克隆到 'xuegod-web'...

Username for 'http://192.168.1.63': jianmingbasic　　#输入 gitlab 的帐号

Password for 'http://jianmingbasic@192.168.1.63': xuegodlinux　#输入密码

查看下载下来的文件：

[root@xuegod63 test]# ls xuegod-web/ -a

.　..　.git　index.html

14.3.2　初次运行 Git 前的配置

一般在新的系统上，我们都需要先配置下自己的 Git 工作环境。配置工作只需一次，以后升级时还会沿用现在的配置。

第一个要配置的是你个人的用户名称和电子邮件地址。这两条配置很重要，每次 Git 提交时都会引用这两条信息，说明是谁提交了更新，所以会随更新内容一起被永久纳入历史记录。

git 运行的环境变量有点像.bashrc，决定了 Git 在各个环节的具体工作方式和行为。这些变量可以存放在以下两个的地方：

1、~/.gitconfig 文件：用户目录下的配置文件只适用于该用户。若使用 git config 时用 --global 选项，读写的就是这个文件。

例 1：修改用户信息

[root@xuegod63 ~]# git config --global user.name "jianmingbasic"

[root@xuegod63 ~]# git config --global user.email "jianmingbasic@163.com"

[root@xuegod63 ~]# cat ~/.gitconfig　#查看

[user]

 email = jianmingbasic@163.com

 name = jianmingbasic

2、当前项目的 Git 目录中的配置文件（也就是工作目录中的 .git/config 文件）：这里的配置仅仅针对当前项目有效。每一个级别的配置都会覆盖上层的相同配置，所以 .git/config 里的配置会覆盖 ~/.gitconfig 中的同名变量。

如果要在某个特定的项目中使用其他名字或者邮件地址，先进到项目上下，然后只要去掉 --global 选项重新配置即可。 最后配置的用户和邮件地址会保存在当前项目的 .git/config 文件里。

例：修改某个 git 项目下的环境变量

[root@xuegod63 xuegod-web]#　cd xuegod-web/

[root@xuegod63 xuegod-web]# git config　user.name "jianmingbasic"

[root@xuegod63 xuegod-web]# git config　user.email "jianmingbasic@163.com"

[root@xuegod63 .git]# vim　./.git/config

...

[user]

 name = jianmingbasic

 email = jianmingbasic@163.com

14.3.3　git 常用命令：

git config --global user.name　“name“ #设置全局用户名

git config --global user.email　mail　　#设置全局邮箱

git config --global　--list　　#列出用户全局设置

git　add index.html　　#添加文件到暂存区

git commit -m　“描述内容“　　#提交文件到工作区

git status　　#查看工作区的状态
git push　　#提交代码到 git 服务器上
git pull　　#获取代码到本地
git log　　#查看操作日志
vim　.gitignore　　#定义忽略文件
git reset --hard　HEAD^　#git 版本回滚，HEAD 为当前版本，加一个^为上一个，^^为上上一个版本
git reflog　# #获取每次提交的 ID，可以使用--hard 根据提交的 ID 进行版本回退
git reset --hard 5ae4b06　　#回退到指定 id 的版本
git branch　#查看当前所处的分支
git checkout --　file　　#从服务器更新某个那文件覆盖本地的文件

例：把修改过的 index.html 文件更新主版本中
[root@xuegod63 test]# cd xuegod-web/
[root@xuegod63 xuegod-web]# echo "bbs.xuegod.cn" >> index.html
[root@xuegod63 xuegod-web]#　git add index.html
[root@xuegod63 xuegod-web]# git commit -m "add bbs.xuegod.cn"
[root@xuegod63 xuegod-web]# git push -u origin　master　#上传到 master 主干下
origin [ˈɒrɪdʒɪn] 起源，根

[root@xuegod63 xuegod-web]# rm -rf index.html　　#删除一些代码
[root@xuegod63 xuegod-web]# git reset --hard　HEAD　#回滚到最新版本
[root@xuegod63 xuegod-web]# ls

查看 git 当前的版本：
[root@xuegod63 ~]# git --version
git version 1.8.3.1

[root@xuegod63 xuegod-web]# git reflog　　#获取每次提交的 ID
9c1e21a HEAD@{0}: commit: aaa
cd9d1d5 HEAD@{1}: commit: add bbs
b2866fd HEAD@{2}: clone: from http://192.168.1.63/xuegod/xuegod-web.git

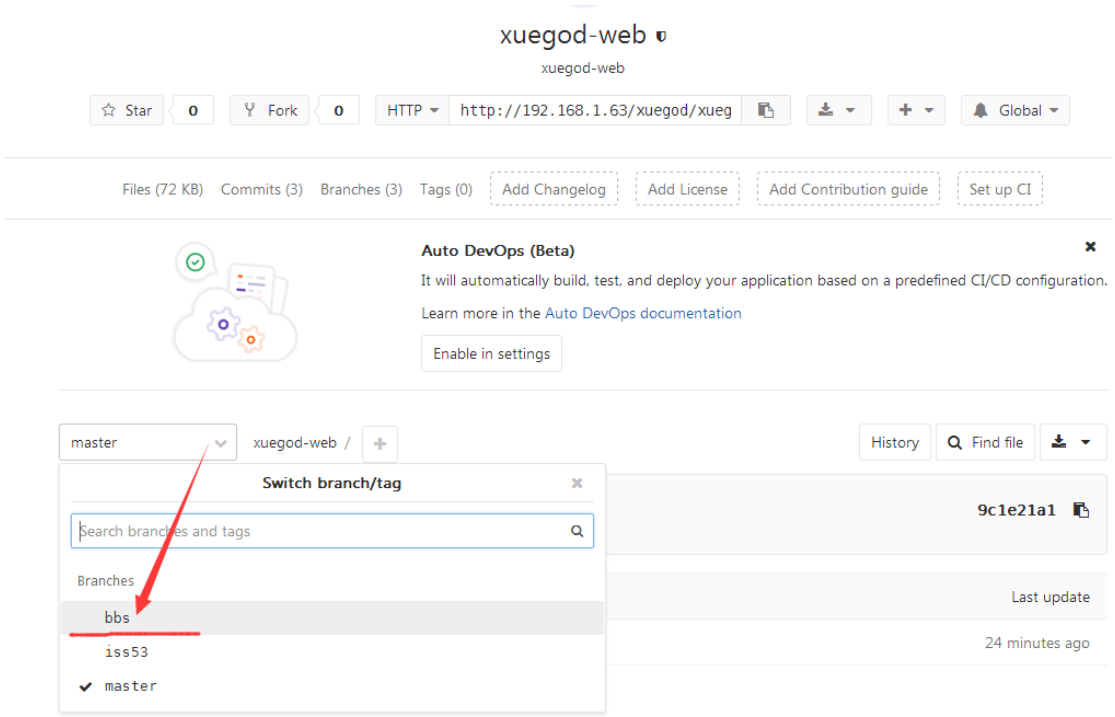
14.3.4　工作区和暂存区及分支概述

1、工作区就是编辑文件的目录区域，需要将工作区的修改好的文件 add 到暂存区才能提交到 git 服务器，在工作区有多个文件的时候可以将一个或多个文件添加至暂存区，再提交到 git 服务器即可。

2、在服务器创建分支
[root@xuegod63 xuegod-web]#　git branch bbs　#创建一个分支
[root@xuegod63 xuegod-web]#　git checkout bbs　#切换到分支 bbs
[root@xuegod63 xuegod-web]#　git branch　　#查看当前所处的分支

[root@xuegod63 xuegod-web]#　vim a.txt　#随意在里面写一些内容
[root@xuegod63 xuegod-web]#　git add a.txt

[root@xuegod63 xuegod-web]# git commit -m "add a.txt " #提交到暂存区中
[root@xuegod63 xuegod-web]# git push -u origin bbs #上传到分支 bbs 分支上



关于 git push.default 设置的知识。

默认配置下，当使用 git push 命令而没有明确的指名本地分支和远程参考分支的情况下，会有如上的提示。如果 git push 命令没有明确指定引用规格(refspec),也就是没有指定推送的源分支和目标分支，那么 git 会采用 push.default 定义的动作。不同的值适用于不同的工作流程模式。

显而易见，主要是因为之前没有进行设置引用规格才出现的这种问题，现在我把 push.default 的可用值与配置方法贴在下面。push.default 可用的值如下：

1.nothing　不推送任何东西并有错误提示，除非明确指定分支引用规格。强制使用分支引用规格来避免可能潜在的错误。

2.current　推送当前分支到接收端名字相同的分支。

3.upstream　推送当前分支到上游@{upstream}。这个模式只适用于推送到与拉取数据相同的仓库，比如中央工作仓库流程模式。

4.simple　在中央仓库工作流程模式下，拒绝推送到上游与本地分支名字不同的分支。也就是只有本地分支名和上游分支名字一致才可以推送，　就算是推送到不是拉取数据的远程仓库，只要名字相同也是可以的。在 GIT 2.0 中，simple 将会是 push.default 的默认值。　simple 只会推送本地当前分支。

5.matching　推送本地仓库和远程仓库所有名字相同的分支。这是 git 当前版本的缺省值。

配置 push.default 的命令如下：　　git config --global push.default simple

14.4　搭建 Jenkins 实现持续集成

14.4.1　安装 JDK1.8

Jenkins 是 Java 编写的，所以需要先安装 JDK，这里采用 yum 安装，如果对版本有需求，可以直接在 Oracle 官网下载 JDK。

[root@xuegod63 ~]#　yum install -y java-1.8.0　　#光盘镜像中有

14.4.2　安装 jenkins

[root@xuegod63 ~]# cd /etc/yum.repos.d/

[root@xuegod63 yum.repos.d]# wget http://pkg.jenkins.io/redhat/jenkins.repo

[root@xuegod63 yum.repos.d]# rpm --import http://pkg.jenkins.io/redhat/jenkins.io.key

[root@xuegod63 yum.repos.d]# yum install -y　jenkins　　#默认安装最新版本。或者直接安装 jenkins-2.93-1.1.noarch.rpm 包

注：新版 GitLab 的服务端口为 8080，为了不和 GitLab 的服务端口相冲突，修改 Jenkins 的默认端口 8080 为 198

[root@xuegod63 yum.repos.d]# vim　/etc/sysconfig/jenkins

改：56 JENKINS_PORT="8080"

为：56 JENKINS_PORT=" 198 "

10 JENKINS_HOME="/var/lib/jenkins"　　#数据目录，建议用固态磁盘来存数据，可以自己定义

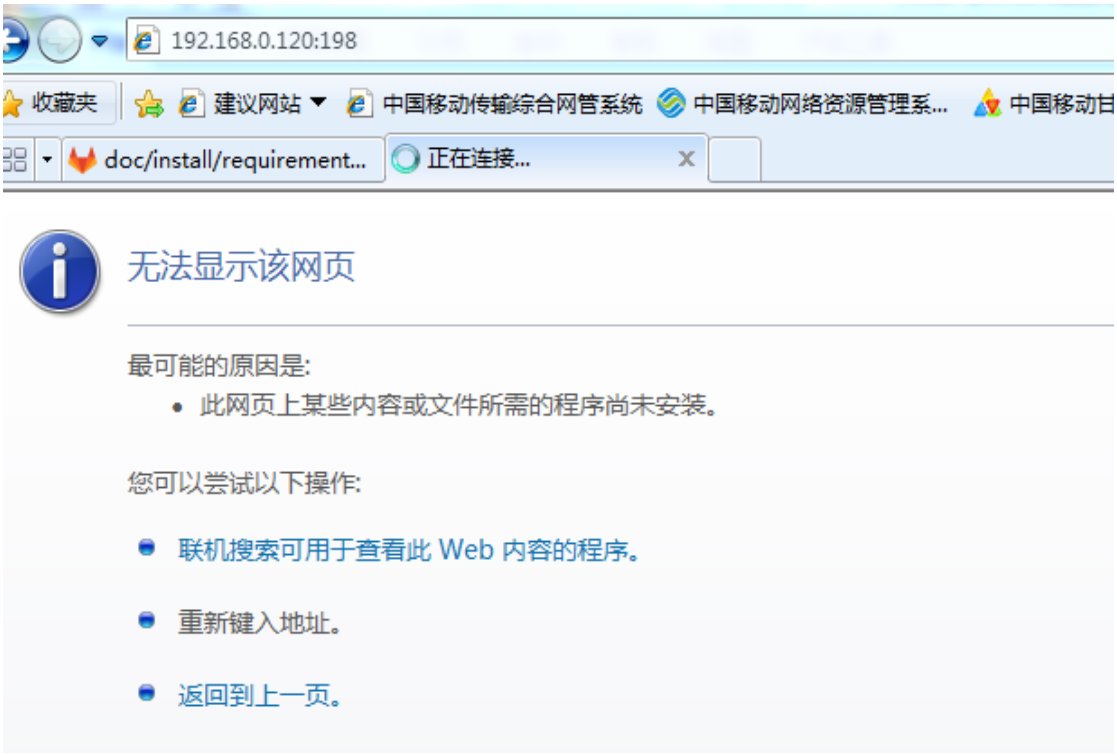
[root@xuegod63 ~]# /etc/init.d/jenkins　start　　#启动

[root@xuegod63 ~]# chkconfig jenkins on　　#设置开机启动

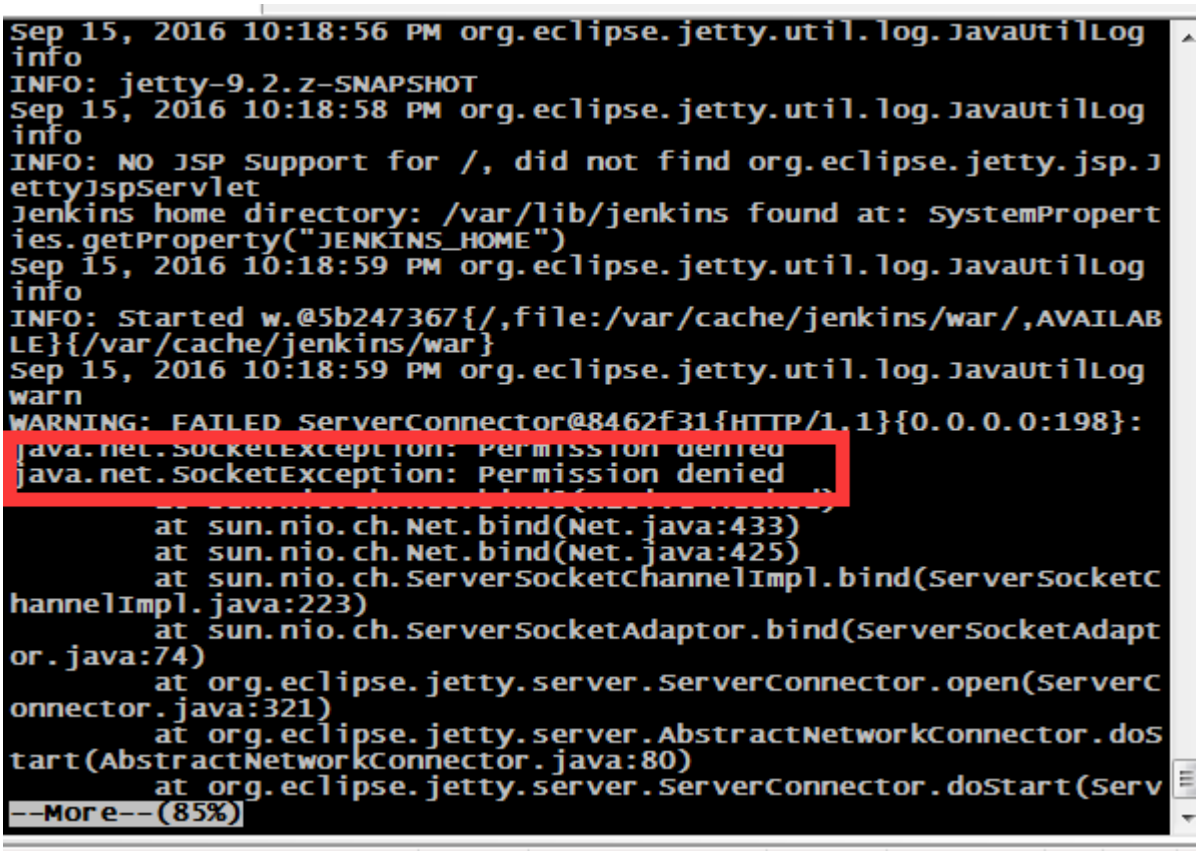
[root@xuegod63 ~]# chkconfig --list jenkins

14.4.3　访问 Jenkins 并安装相关插件

在浏览器输入 http://192.168.1.63:198 访问 jenkins。



无法访问，此时查看 jenkins 进程，已经意外关闭。再次尝试启动 Jenkins,结果还是意外关闭，经过一翻查找资料与分析日志： java.net.socket exception permission denied



得出 GitLab 默认使用的是 root 用户，而 Jenkins 默认使用的是 jenkins 用户，因此也就出现日志中的权限问题了。修改 Jenkins 的默认用户为 root。

[root@xuegod63 ~]# vim /etc/sysconfig/jenkins

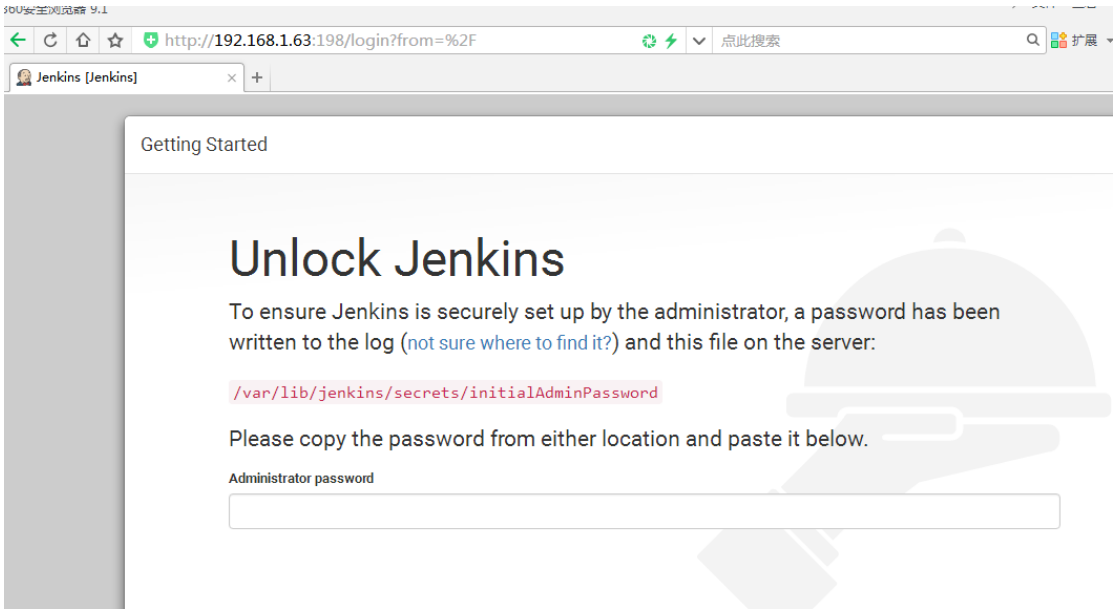
改：29 JENKINS_USER="jenkins"

为：29 JENKINS_USER="root"

[root@xuegod63 ~]# /etc/init.d/jenkins restart

访问：

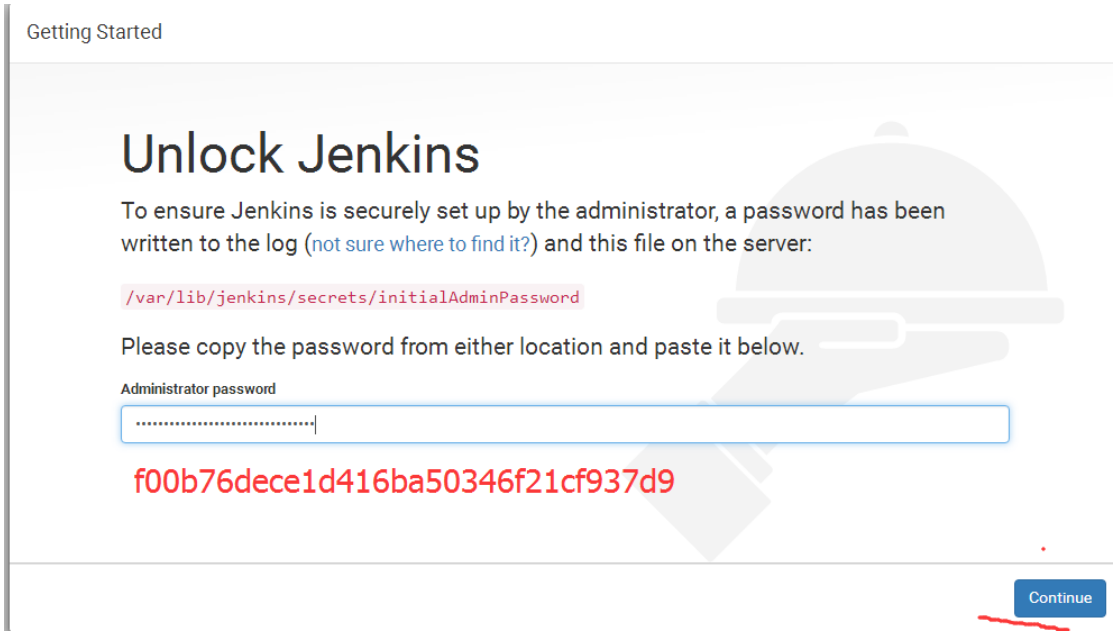
http://192.168.1.63:198



为了安全考虑，首先需要解锁 Jenkins

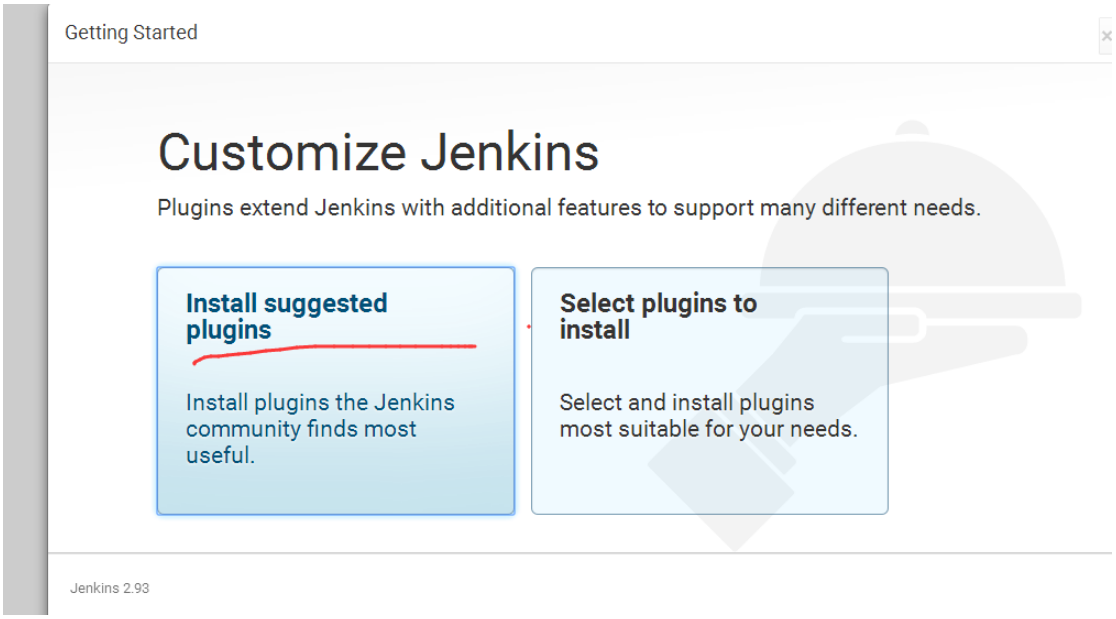
[root@xuegod63 ~]# cat /var/lib/jenkins/secrets/initialAdminPassword #查看初始化密码文件。

f00b76dece1d416ba50346f21cf937d9 #把密码输入以一下页面，点 continue 。

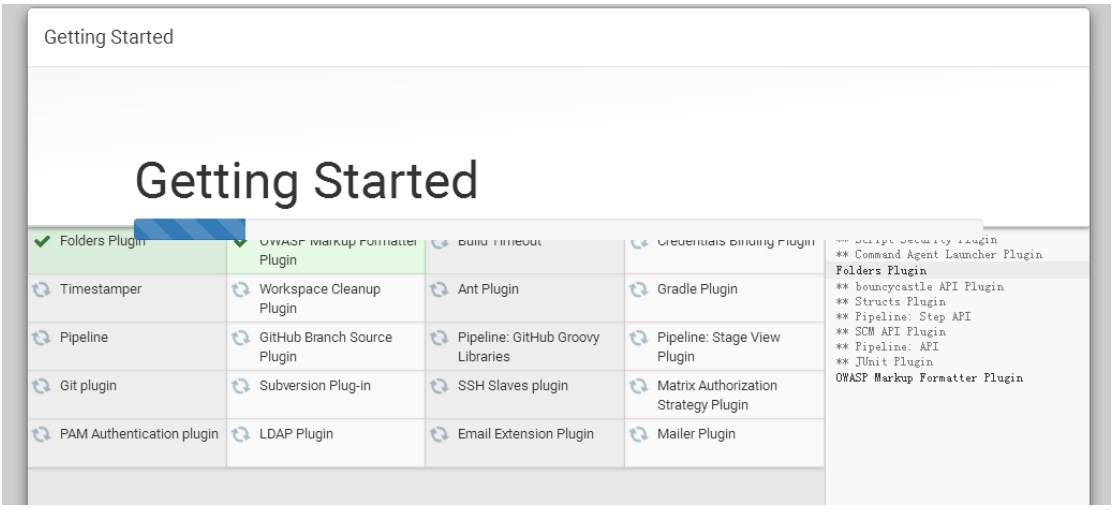


选择需要安装的插件：

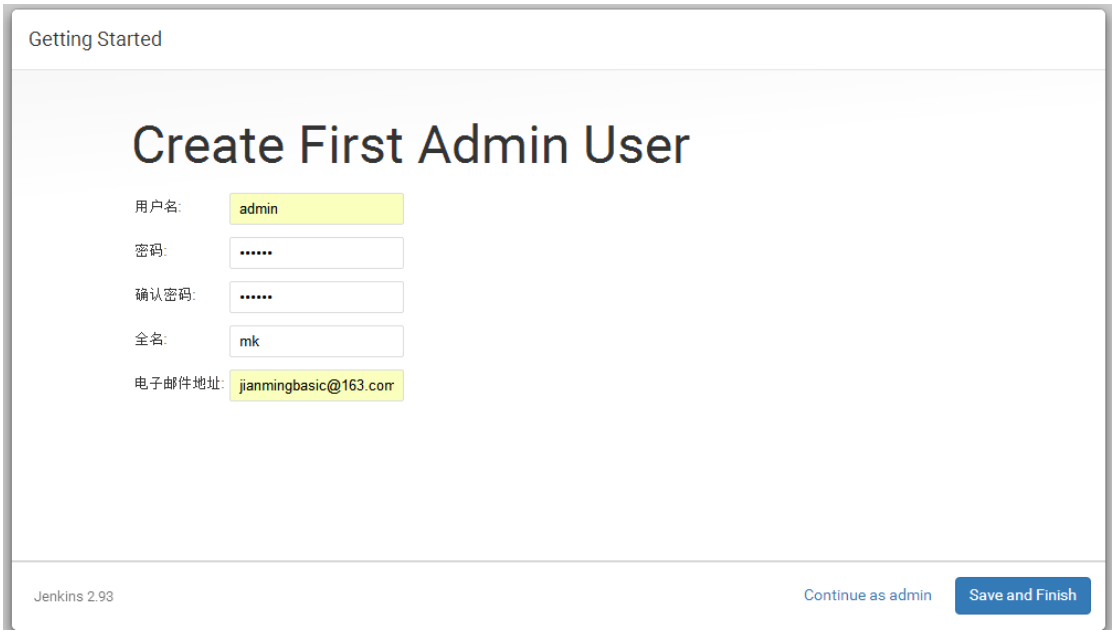
选择默认即可，会安装通用的社区插件，剩下的可以在使用的时候再进行安装。

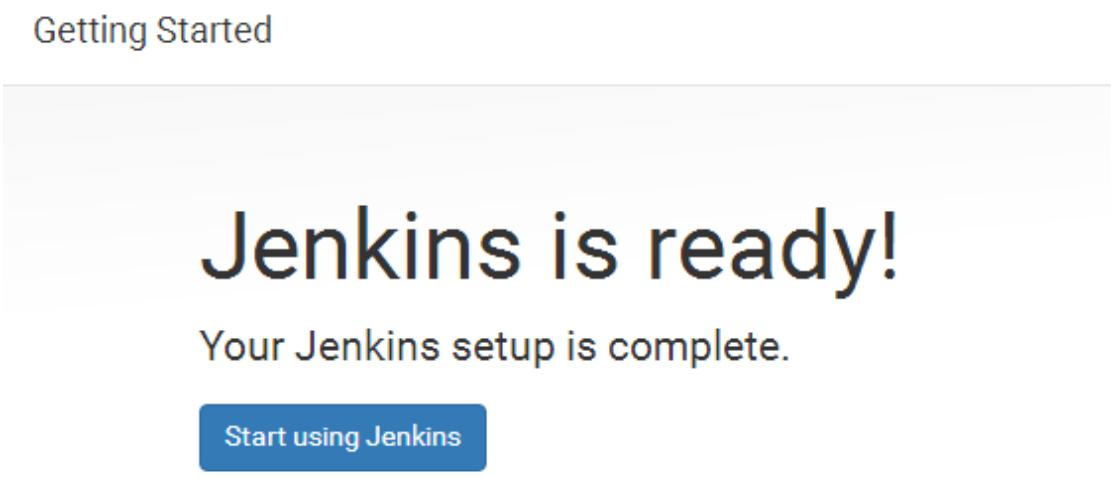


确保推荐安装的插件都安装成功。



创建管理员用户：admin 密码： 123456 全名： admin

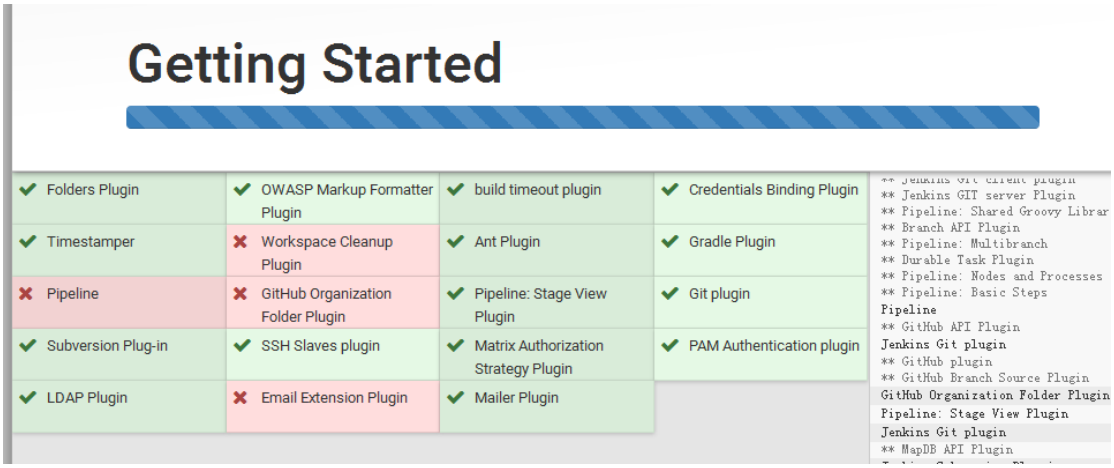




到些 jenkins 安装成功。

14.4.4 手动安装相关插件

如果在下线安装插件失败了，或是无网环境下想安装插件，可以选择手动安装。



这里不用管，等安装成功后，我们再手动安装插件。安装成功后，登录系统，选择：
系统管理->插件管理->高级

插件下载地址：

http://updates.jenkins-ci.org/download/plugins/ #在有网的环境下，把自己需要的插件下载好，然后再从本地上传。

方法 2： 也可以直接把一台安装好 jenkins 插件服务器的/var/lib/jenkins/plugins 目录下的文件复制到新的 jenkins 中。

把准备好的插件解压一下：

[root@xuegod63 jenkins]# tar czvf plugins.tar.gz plugins/

#cd /var/lib/jenkins/

#rm -rf /var/lib/jenkins/plugins

#tar -zxvf plugins.tar.gz #上传 plugins.tar.gz 到 linux 系统上，解压缩

#chown jenkins.jenkins ./* -R

#/etc/init.d/jenkins start

注：记得重启 jenkins，这个非常重要，因为不重启，插件不会生效

到此 jenkins 安装成功。

登录 gitlab http://192.168.1.63/ 用户名： root 密码： xuegod.cn

登录 jenkins http://192.168.1.63:198/ 用户名： root 密码： 123456

总结：

14.1 持续集成概述及运行流程

14.2 搭建 GitLab 平台及使用方法

14.3 安装 git 客户端使用 gitlab

14.4 搭建 Jenkins 实现持续集成