centos下快速部署gitlab

<http://yijiebuyi.com/blog/49aa7d3793aeafeb77da67a4159ec1aa.html>

在程序开发过程中git已经成为最受欢迎的代码托管工具，从此工具上衍生出来了大名鼎鼎的开源平台 github . 它给我们代码管理带来方便快捷,但是在 github 上,只有开源的公开项目才可以免费试用,如果公司的私有项目,必须付费购买,而且价格还不低.所以今天这篇文章和大家分享下在 centOS 6.5 环境下,傻瓜式,简洁快速部署安装 gitlab 服务.

什么是gitlab

GitLab是利用 Ruby on Rails 一个开源的版本管理系统，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。它拥有与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。团队成员可以利用内置的简单聊天程序(Wall)进行交流。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

安装说明

GitLab 有2个版本 GitLab Community Edition (CE) 及 GitLab Enterprise Edition (EE)，CE 是免费版,我们今天安装的就是 CE 版本.

GitLab 官网有提供 CentOS / Debian / Ubuntu 系统的安装套件,今天我们要在 CentOS 6.5 上安装GitLab.

安装准备

我们直接采用官方提供的 rpm 安装方式,[GitLab官方地址](https://about.gitlab.com/downloads/) 你可以通过这个地址找到你对应的安装OS 系统, 直接就能看到对应此系统的安装提示方法.

中途不需安裝 git , ruby 或 web server 了，因為官方提供的有整合了大部分的套件(nginx,ruby & rails , git , redis , postgresql …)

下载 GitLab 并安装

* 下载

wget https://downloads-packages.s3.amazonaws.com/centos-6.5/gitlab-7.2.1\_omnibus-1.el6.x86\_64.rpm

* 下载完成后

yum install openssh-server sudo yum install postfix sudo service postfix start sudo chkconfig postfix on

依次安装上面这几部,然后接着安装

rpm -i gitlab-7.2.1\_omnibus-1.el6.x86\_64.rpm

修改对外开放域名或ip

vi /etc/gitlab/gitlab.rb

打开后有一行 external\_url 的設定改成要對外開放 web 的 url ， 例如我可以指定 git.yijiebuyi.com 只想內部使用也可以改成 http://192.168.0.11 这样的内部IP地址.

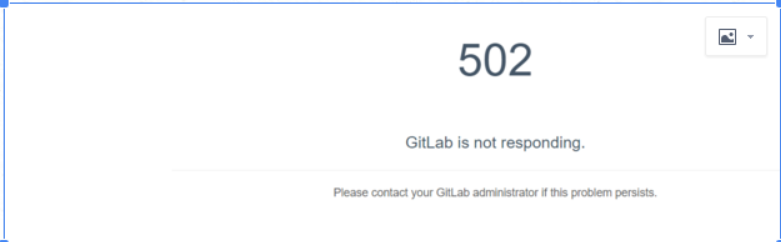
执行 GitLab

sudo gitlab-ctl reconfigure sudo lokkit -s http -s ssh

如果运行上面的命令,发现没有安装 lokkit ,那么需要你手动 yum install lokkit lokkit 可以帮助我们设定 iptables 打开 http 和 ssh

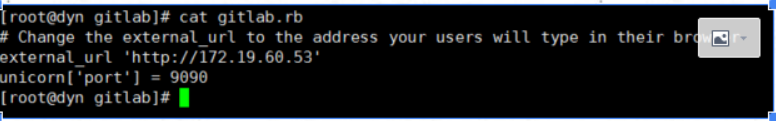
通过web访问

刚才上面设置的对外开放域名或者 ip ,这时你在浏览器可以访问了,这样就算正式安装成功了.



原来是在服务器上还开启了一个tomcat服务，占用了8080端口,使GitLab的unicorn服务不能开启。

最后在/etc/gitlab/gitlab.rb 中做出如下修改



添加 unicorn['port'] = 9090

然后 gitlab-ctl reconfigure

初始密码：

Username: root

Password: 5iveL!fe

刚开始登陆需要更新密码

密码：lanmao123

配置完成后，登陆报错，原因是密码忘记了

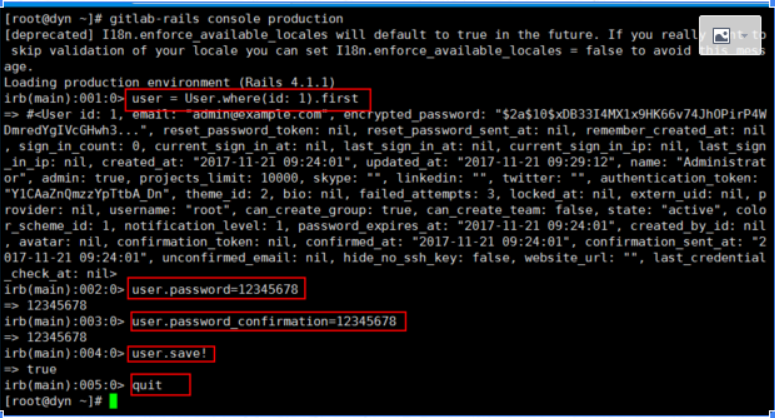
解决方法：

 在root用户下，执行

gitlab-rails console production

* 1

1. 获得用户数据，修改用户密码



注意：**密码没有使用引号**，奇怪的是使用单引号或双引号，密码就无效，估计是包含了这个字符，不包含，就没有问题。

3. 保存用户数据

user.save!

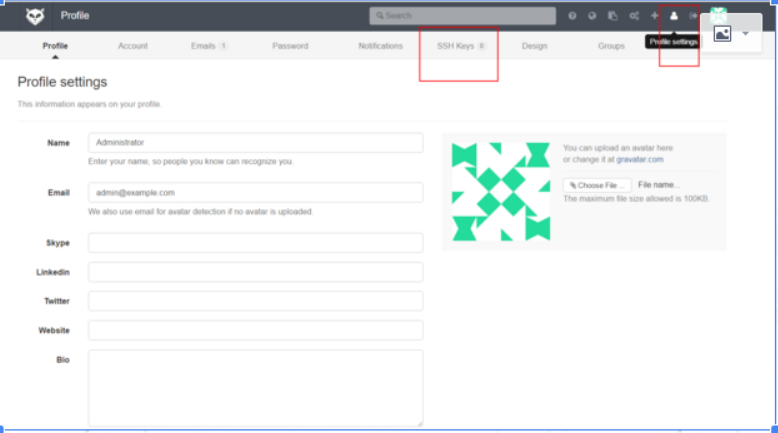
* 1

注意需要使用后面的**感叹号!**

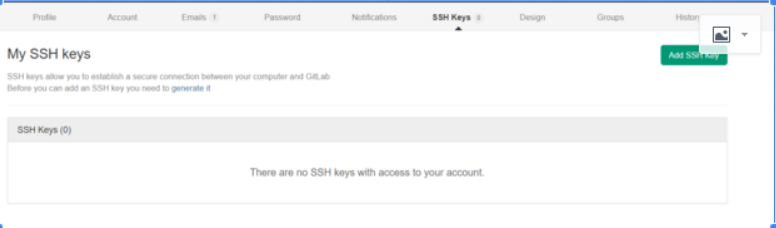
**我这里用户：root**

**密码：lanmao123**

**登陆成功后显示如下页面**



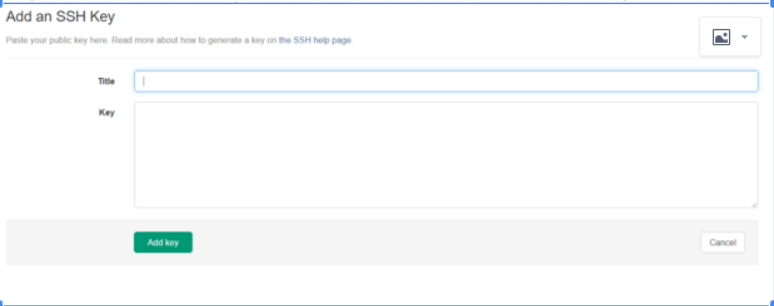
**点击SSH Key**



**Add SSH Key**

**注意：title——写一个简易一点的名称**

**key：在服务器/root/.ssh/id\_rsa.pub**



如果本地没有ssh key，用ssh-keygen初始化一个，可以参考如下部分：

################################配置Git################################

首先在本地创建ssh key

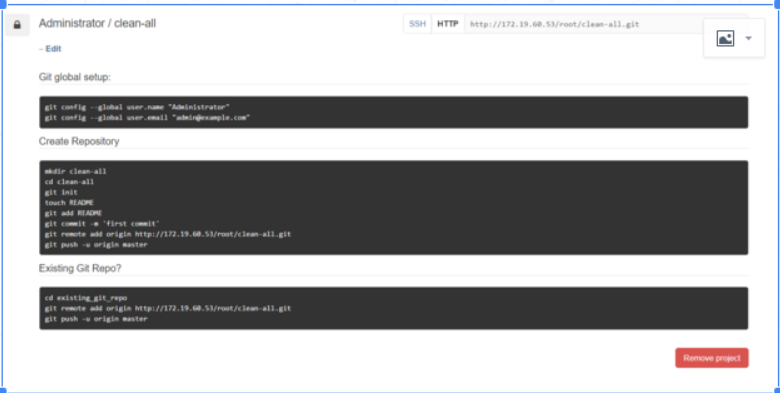
# ssh-keygen -t rsa -C "yanan.duan@lanmaoly.com"

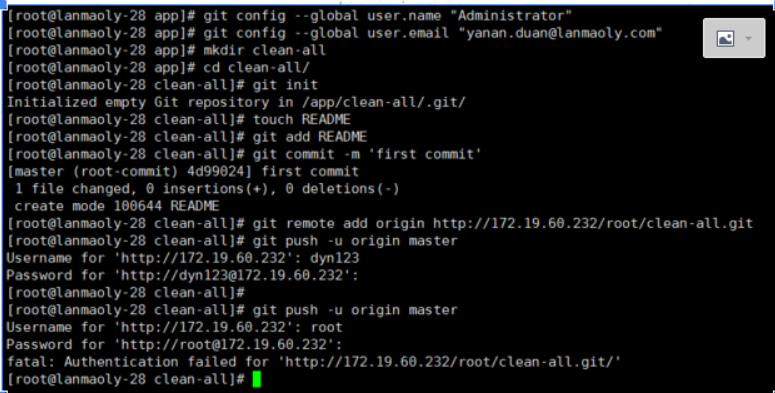
后面的your\_email@youremail.com改为你的邮箱，之后会要求确认路径和输入密码，我们这里使用默认的一路回车就行，成功的话会在~/下生成.ssh文件夹，进去，打开id\_rsa.pub，复制里卖弄的key

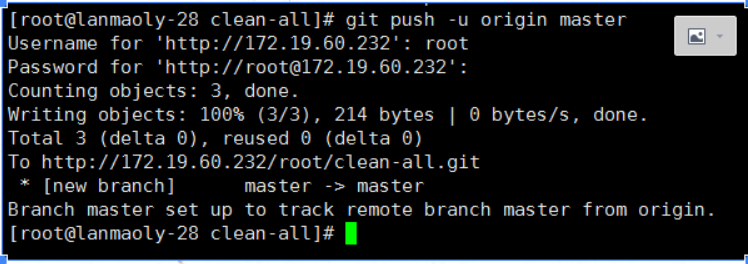
回到gitlab，进入settings，左边选择ssh key，add SSH Key，粘贴key。

添加完key之后流程如下：

在git服务器上做以下内容







username:gitlab用户名

密码：gitlab密码

如果是第一次的会提示是否continue，输入yes就会看到：You’ve successfully authenticated, but GitHub does not provide shell access 。这就表示已成功连上github。

接下来我们要做的就是把本地仓库传到github上去，在此之前还需要设置username和email，因为github每次commit都会记录他们。

1. $ git config --global user.name "your name"
2. $ git config --global user.email "your\_email@youremail.com"

进入要上传的仓库，右键git bash，添加远程地址：

1. $ git remote add origin git@github.com:yourName/yourRepo.git

后面的yourName和yourRepo表示你再github的用户名和刚才新建的仓库，加完之后进入.git，打开config，这里会多出一个remote “origin”内容，这就是刚才添加的远程地址，也可以直接修改config来配置远程地址。

**4.提交、上传**

接下来在本地仓库里添加一些文件，比如README，

1. $ git add README
2. $ git commit -m "first commit"

上传到github：

1. $ git push origin master

git push命令会将本地仓库推送到远程服务器。

git pull命令则相反。

修改完代码后，使用git status可以查看文件的差别，使用git add 添加要commit的文件，也可以用git add -i来智能添加文件。之后git commit提交本次修改，git push上传到github。

**5.gitignore文件**

.gitignore顾名思义就是告诉git需要忽略的文件，这是一个很重要并且很实用的文件。一般我们写完代码后会执行编译、调试等操作，这期间会产生很多中间文件和可执行文件，这些都不是代码文件，是不需要git来管理的。我们在git status的时候会看到很多这样的文件，如果用git add -A来添加的话会把他们都加进去，而手动一个个添加的话也太麻烦了。这时我们就需要.gitignore了。比如一般c#的项目我的.gitignore是这样写的：

1. bin
2. \*.suo
3. obj

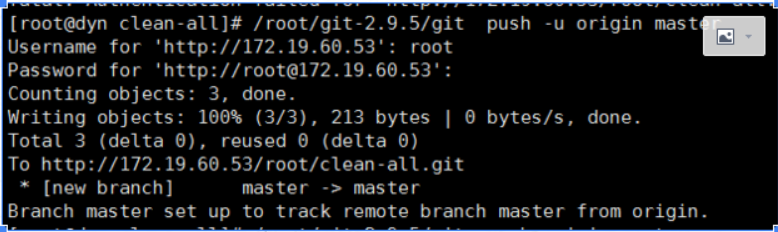
bin和obj是编译目录，里面都不是源代码，忽略；suo文件是vs2010的配置文件，不需要。这样你在git status的时候就只会看到源代码文件了，就可以放心的git add -A了。

git status ——查看文件的差别

git add ——添加要commit的文件

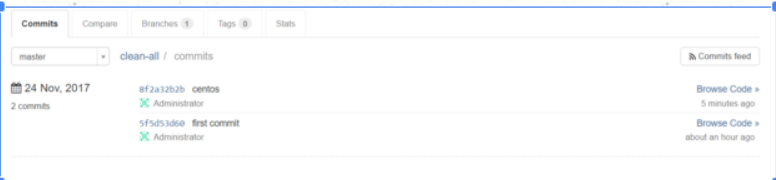
git commit -m ——提交本次修改

git push ——上传到gitlab



上图是提交到gitlab成功

在gitlab上可以看到所提交的内容



报错信息：

（1）

[root@dyn clean-all]# /root/git-2.9.5/git push -u origin master

fatal: Unable to find remote helper for 'http'

解决：

因为/opt/gitlab/embedded/libexec/git-core/没在PATH环境变量中

我们查看以下ls /opt/gitlab/embedded/libexec/git-core/

这导致里面的git-remote-https,git-remote-http这些得不到执行，所以git所表现出来的功能不全

将/opt/gitlab/embedded/libexec/git-core/纳入PATH，至少在使用git之前，设置一下PATH

#PATH=￥PATH:/opt/gitlab/embedded/libexec/git-core/

或直接在/etc/profile中修改

6.tag

我们可以创建一个tag来指向软件开发中的一个关键时期，比如版本号更新的时候可以建一个“v2.0”、“v3.1”之类的标签，这样在以后回顾的时候会比较方便。tag的使用很简单，主要操作有：查看tag、创建tag、验证tag以及共享tag。

6.1查看tag

列出所有tag：

1. git tag

这样列出的tag是按字母排序的，和创建时间没关系。如果只是想查看某些tag的话，可以加限定：

1. git tag -l v1.\*

这样就只会列出1.几的版本。

6.2创建tag

创建轻量级tag：

1. git tag v1.0

这样创建的tag没有附带其他信息，与之相应的是带信息的tag：

1. git tag -a v1.0 -m 'first version'

-m后面带的就是注释信息，这样在日后查看的时候会很有用，这种是普通tag，还有一种有签名的tag：

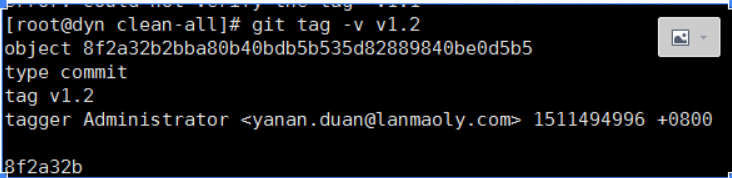
1. git tag -s v1.0 -m 'first version'

前提是你有GPG私钥，把上面的a换成s就行了。除了可以为当前的进度添加tag，我们还可以为以前的commit添加tag：

1. #首先查看以前的commit
2. git log --oneline
3. #假如有这样一个commit：8a5cbc2 updated readme
4. #这样为他添加tag
5. [root@dyn clean-all]# git tag -a v1.2 8f2a32b

会打开一个文件，然后将“8f2a32b”这个git版本号添加到打开文件的末尾

6.验证tag：git tag -v v1.0



6.3删除tag

很简单，知道tag名称后：

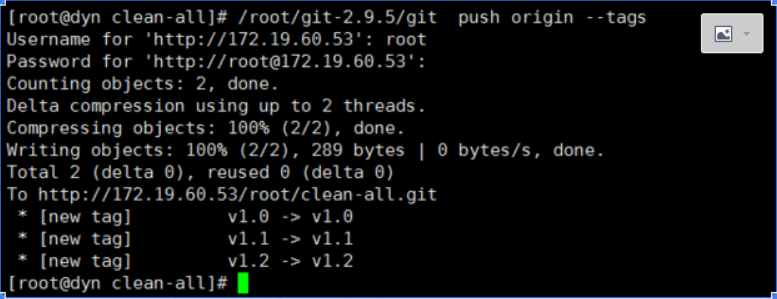
1. git tag -d v1.0

6.4.共享tag

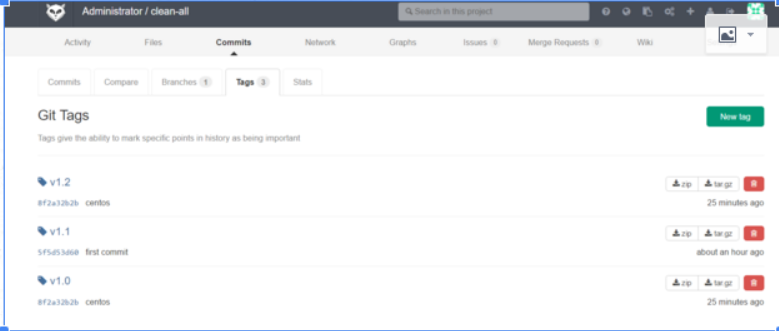
我们在执行git push的时候，tag是不会上传到服务器的，比如现在的gitlab，创建tag后git push，在gitlabb网页上是看不到tag的，为了共享这些tag，你必须这样：

1. git push origin --tags

服务器上提交tag



gilab网页上查看从服务器上所提交的tag



想要查看关于key与tag相关的可以看如下链接：

[https://wuyuans.com/2012/05/github-simple-tutorial/#toc-3](https://wuyuans.com/2012/05/github-simple-tutorial/" \l "toc-3)