# ljhui GIT使用手册

## 激活gitlab账号

运维人员已给各位注册了git帐号，各位收下邮件，点击链接，重置密码激活帐号就可以使用。Git仓库是公司内网服务器，所以需要登录openvpn，才能访问。

地址： http://git.cm-pro.cn/cm/ljsh

## 安装git工具

到<http://www.git-scm.com/download/>上下载最新版的git工具然后安装

安装完成以后，打开git bash，配置一下信息（用户名和邮箱请改成个人自己的）

$ git config --global user.name "张涛"

$ git config --global user.email " zhangtao@cm-inv.com"

$ git config --global core.autocrlf input

$ git config --global core.safecrlf warn

## 从gitlab上下载项目源码

打开git bash命令行，创建ssh key

$ ssh-keygen -t rsa -C "youremail@example.com"

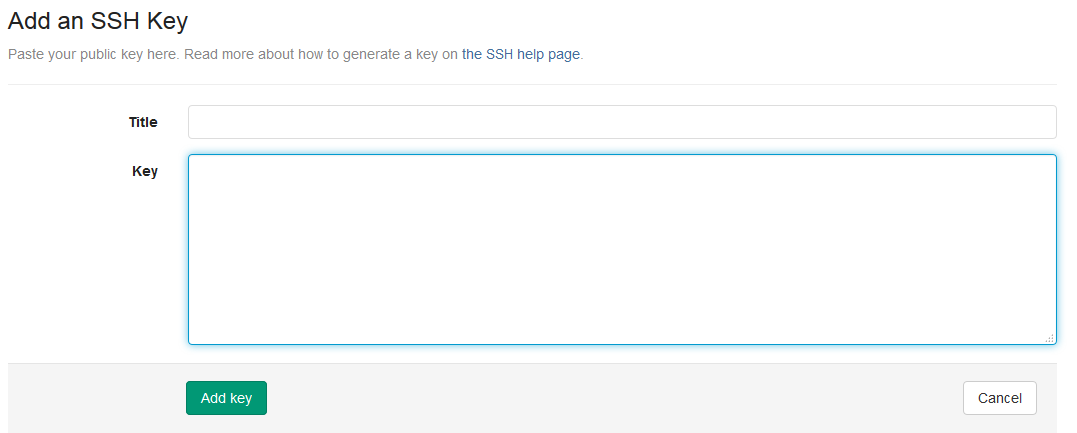
创建好的ssh key在用户个人目录下，例如

C:/users/zhangtao/.ssh/

打开gitlab配置ssh key的网址

http://git.cm-pro.cn/profile/keys/new

选择”Add SSH Key”，将刚刚创建好的ssh目录里面的id\_rsa.pub文件内容粘贴到“key”输入框里面，再点击“Add key”即可



打开git bash，切换到你打算放置本地项目的目录，比如c:/users/ zhangtao /git

$ cd /c/Users/zhangtao/git/

从git仓库中下载源代码

$ git clone git@git.cm-pro.cn:cm/ljsh.git

项目代码已经被下载到 c:\users\zhangtao\git\ljsh 目录中

切换到dev分支

$ cd ljsh

$ git checkout dev

## 使用maven构建项目

安装jdk和maven，并在windows环境变量里面设置以下变量

JAVA\_HOME C:\Program Files\Java\jdk1.7.0\_79

M2\_HOME C:\work\maven3

并在Path中加入

%M2\_HOME%\bin;

在命令行窗口执行

mvn –version

显示maven 和 jdk 信息则表示配置正确



## 使用eclipse IDE打开项目

使用命令行进入根目录

在命令行输入如下指令，编译并生产eclipse项目：

$ mvn install

$ mvn eclipse:eclipse

编译后进入eclipse，将项目import进来

项目中点击右键，选择 Configure -> Convert to maven project 等待完成

若需要本地测试该工程，只需要右键 Run as -> Run on server

## 使用Intellij IDE打开项目

## 如何开发自己的功能

当开发自己的新功能或者修复bug的时候，需要创建一个自己的分支branch，在分支里修改代码。



### user1修复bug 2577

从dev创建分支，修改并提交本地

git checkout dev

git pull origin dev

git checkout -b bug-2577

修改文件

vi a.txt

git add a.txt

git commit -m "fix bug 2577"

把自己的代码合并进dev

git pull origin dev

git merge --no-ff bug-2577 (合并时--no-ff选项避免fast forward)

从本地dev提交本地dev到远端

git push origin dev

删除本地分支

git branch -d bug-2577

### user1和user2 开发feature 1101

**feather分支用于某个新feather的开发，源自develop，并最终merge到develop。**

user1从dev创建分支并上传到remote

git checkout dev

git pull origin dev

git checkout -b feature-1101

git push origin feature-1101

user2下载分支到本地

git checkout -b feature-1101 origin/feature-1101

user1(user2)在分支feature修改文件

vi b.txt

git -a -m "update b.txt for feature 1101"

user1(user2)提交本地修改到远程feature分支

git push origin feature-1101

feature开发完毕后，合并分支feature到dev即可

git pull origin dev

git merge --no-ff feature-1101

删除远程feature分支

git push origin --delete feature-1101

### user1提交hotfix到master

**Hotfix分支用于紧急bug修复，源自master，merge到develop和master。**

从master创建hotfix分支，修改并提交本地

git checkout master

git pull origin master

git checkout -b hotfix-4830

vi a.txt

git commit -a -m "hotfix 4830"

合并hotfix进master

git checkout master

git merge --no-ff hotfix-4830

git push origin master

合并hotfix进dev

git fetch origin dev:dev

git checkout dev

git merge --no-ff hotfix-4830

git push origin dev

删除本地hotfix分支

git branch -d hotfix-4830

## 准备新版本的发布

**release分支用于准备新版本的发布。源自develop，merge到develop和master。**

release分支仅修复小的bug，完成准备版本号，build date等工作。该分支上修复的bug需要merge到develop，并在该分支完成时merge到master。此时需要给master打上tag，标记这个新的release。

创建release branch：

# git checkout –b release-x.y develop

完成release branch

# git checkout master

# git merge --no-ff release-x.y

# git tag –a x.y

# git checkout develop

# git merge --no-ff release-x.y

# git branch –d release-1.2