# gitLab简单实用教程

gitLab简单实用教程 1

第一章、GitLab.com 1

第二章、创建项目 1

第四章、配置ssh(可选) 1

第五章、上传项目 2

第六章、配置CI(持续集成) 2

6.1 Pipelines 2

6.2 Jobs 2

6.3 stages 2

6.4 gitlab-ci.yml 2

6.5 总结 4

第七章、实用技巧 4

## 第一章、GitLab.com

gitlab是开源项目，官网也提供了社区版安装包

1.有自己服务器的话可以私有化部署一个，安装教程参考官网<https://about.gitlab.com/installation/>

个人开发者来说要求的服务器配置有点高，1核1G的服务器也能勉强跑起来。

2.另一种选择就是使用GitLab.com，这是官方提供的免费平台，功能和社区版一样，有人在维护、更新，新功能会比较多，缺点就是偶尔抽风。地址<https://gitlab.com/users/sign_in> > GitLab.com offers free unlimited (private) repositories and unlimited collaborators.注册流程比较简单，填一下username和email就差不多了，这也将是git空间的标记，名字不要取得太随意就行了。

## 第二章、创建项目

登录后点击右上角的加号(New project)。

Project path：如果你属于group的话可以选group名字，这样项目就会放在对应group下，一般团队项目比较好用。

Project name：你的项目名字

Import project from：可以从多个github、bitbucket等主流托管平台导入项目。

Project description：项目描述，可选

Visibility Level：项目可见级别

Private：私有项目，需要授权才能访问，适合个人、团队开发。

Internal：内部项目，注意只要登录账号就能访问，适合开源贡献代码。

Public：公开项目，不用登录就能访问，适合分享项目。

点击create，创建项目，进入空项目，会出现初始化步骤，可以用ssh和https方式来上传代码，推荐ssh，比较安全。

## 第四章、配置ssh(可选)

如果本地没有ssh key，用ssh-keygen初始化一个，可以参考我之前写的github教程git初始化那部分<https://wuyuans.com/2012/05/github-simple-tutorial/#toc-3>

有ssh key后添加到后台，点击右侧头像，下拉菜单里选settings，在顶部的tab里点击SSH Keys，或者直接访问[https://gitlab.com/profile/keys。](https://gitlab.com/profile/keys%E3%80%82)

Key就是.ssh/id\_rsa.pub文件内容，title填自己知道的就行，尽量语义化点。

Add key，完成。

## 第五章、上传项目

回到我们创建的空项目页面，在项目名称下面选择传输协议，ssh或者https，下面教程里的url会跟着变。

之后按照下面的教程来做就行了，最后push

git push -u origin master

完成，这时项目页面应该就有东西了，后面就可以用正常的git命令来维护代码仓库了。

## 第六章、配置CI(持续集成)

如果只是要一个git代码托管的话上面几步已经足够了，现在开始介绍gitlab提供的持续集成功能，这对于需要打包、发布的人来说非常方便。

### 6.1 Pipelines

一个pipeline就是一次持续集成任务，一般由一次push触发，在网页上对项目的修改、merge也会触发pipline。pipline由Runner执行，Runner有两种：

Specific Runners：私有runner，部署和执行在自己服务器上，优点是安全、速度快，缺点是需要提供服务器，部署教程<https://docs.gitlab.com/runner/install/linux-repository.html>

Shared Runners：共享runner，官方提供的runner，优点是免费，缺点是会偶尔抽风、或者速度慢

### 6.2 Jobs

pipline由多个job组成，一个job会发给一个runner来执行，所以各个job之间的数据不是共享的，除非使用cache。所以尽量把一些有依赖的步骤放到一个job里，或者把一些通用步骤放到before\_script里，这个后面会提到。

### 6.3 stages

stage是对job的分组，同一个stage里的job是并行的，两个stage之间是串行的

### 6.4 gitlab-ci.yml

要想配置上面说的这些，需要在项目根目录新建.gitlab-ci.yml文件，文件格式为yaml，教程[https://docs.gitlab.com/ee/ci/yaml/README.html。举个例子，这是我www工程配置文件的简化版，使用golang编译：](https://docs.gitlab.com/ee/ci/yaml/README.html%E3%80%82%E4%B8%BE%E4%B8%AA%E4%BE%8B%E5%AD%90%EF%BC%8C%E8%BF%99%E6%98%AF%E6%88%91www%E5%B7%A5%E7%A8%8B%E9%85%8D%E7%BD%AE%E6%96%87%E4%BB%B6%E7%9A%84%E7%AE%80%E5%8C%96%E7%89%88%EF%BC%8C%E4%BD%BF%E7%94%A8golang%E7%BC%96%E8%AF%91%EF%BC%9A)

image: golang:latest

before\_script:

- ln -s /builds/wuyuans/www /go/src/www

- cd /go/src/www

- mkdir bin

stages:

- build

- deploy

build\_web:

stage: build

script:

- go build -v -o bin/web www/web

except:

- release

build\_service:

stage: build

script:

- go build -v -o bin/service www/service

except:

- release

deploy\_web:

stage: deploy

script:

- go build -v -o bin/web www/web

- scp bin/web root@${HOST\_1}:/bin/

environment:

name: www/web

url: http://$CI\_ENVIRONMENT\_SLUG.wuyuans.com

when: manual

only:

- release

#### 6.4.1 image

编译使用的docker镜像，如果是golang的话可以用golang:latest，使用最新版golang，其他可以在docker hub查<https://hub.docker.com/_/golang/>

#### 6.4.2 before\_script

每个job执行前都会执行before\_script里的步骤，主要是做一些环节初始化，比如我这里把工程目录链到了GOPATH下，这样方便使用go命令。也可以在这里做一些go get工作

#### 6.4.3 stages

我分了两个stage，build和deploy。build里有build\_web、build\_service，deploy里的是deploy\_web，名字可以随便，主要是job里的stage字段需要和stages里定义的对应上。

#### 6.4.4 build\_web、build\_service

script里的是执行的命令，做go build的工作，except表示这个job不能在release分支执行。

#### 6.4.5 deploy\_web

script和前面一样。environment用来标记发布的名字，我们可以用environment来管理发布版本、回滚等。when表示执行时间，默认是always每次都会执行，manual表示需要在后台手动执行，这样在不需要所有deploy job都执行的时候手动deploy项目。only表示只在release分支执行。

#### 6.4.6 Environments

在.gitlab-ci.yml里配置了environment后，job执行完后会在项目页面里的Pipelines->Environments下看到这次job，他会按照配置里的name来合并，每次job都可以重做，也就是可以用来做项目的重发和回滚，右边有相对于的按钮，很方便。

### 6.5 总结

gitlab有很多功能非常实用，比如上面讲到的Pipelines、Environments等，还有像Graph(以前叫network)可以显示所有分支的树状结构，这对于在多个分支里来回切换、分不清在哪个分支提交的人来说很直观。而且gitlab对于权限控制提供了很多的选项，很适合团队合作。然而缺点也还是有的，因为是官方托管的平台，日常维护、偶尔抽风什么的，还有被墙的风险什么的。。。如果是个人用户应该关系不大，如果是团队的话还是自建gitlab社区版，功能应该差不多，毕竟安全和稳定对团队来说是比较重要的。

## 第七章、实用技巧

vscode拉gitLab项目的方法：

查看-命令面板-搜索Git-Git:克隆中输入对应http地址-选择对应存储位置即可。

如何删除GitHub或者GitLab上的文件夹

情景：把本地仓库的一个文件夹A推送到了远程GIT服务器上，此时想删除远程仓库的文件夹A,但是本地又不想删除。

问题：远程操作只能操作单个文件，无法删除文件夹。

解决方案：

1.以删除 .setting 文件夹为案例

git rm -r --cached .setting #--cached不会把本地的.setting删除

git commit -m 'delete .setting dir'

git push -u origin master

2.当误提交的文件夹比较多，直接修改.gitignore文件,将不需要的文件过滤掉，然后执行命令

git rm -r --cached .

git add .

git commit

git push -u origin master