GitLab 使用介绍

目录

1.	我们的 Git 开发版本管理服务器	L
2.	在 GitLab 界面查看看项目的相关信息	L
3.	在 GitLab 界面创建自己项目	2
4.	在 Window 安装 Git 客户端软件	3
5.	生成 SSH 密匙来验证身份	1
6.	初始化 GitLab 上的项目存储库	5
7.	克隆 Git 服务器上的项目到本地10)
8.	提交与推送项目文件12	L

1. 我们的开发版本管理服务器: 192.168.1.16,用 <u>http://192.168.1.16</u> 访问即为 GitLab 界面

GITLAB	Dashboard		۵۵ -	+ 🔺 🔍	Search
Â	Issues 1	Merge Requests	0	Search	Help
● 微信项目基本 时对中文翻译也做了一	uted on wall at / weixin ^比 己完成。 ··		3 days ago. ,同	Projects (1)	+ New Project
opened	issue Test Issue 测试开bug at/ w	veixin	3 days ago.	Last activity: 3 days	s ago \rightarrow
joined pro	ject at / weixin		3 days ago.	News Feed	
joined p	roject at / weixin		3 days ago.	Homepage Blog	g @gitlabhq

2. 登录 GitLab 界面后,点击一个项目,进入项目空间,可以看到项目的相关信息。 在这里你可以查阅项目的文件内容、提交历史、Bug 列表 (Issues)、讨论墙、维基页 面等等。

GITLAE	eix weix	in			¢ĉ	+	*	Q Search	
Â	Files	Commits	Network	Issues 1		Merge	Requests	Wall	Wiki
A Show	🛔 Team	Attachments	Snippets		🗷 Edit	t	Services	Hooks	Deploy Keys
SSH HTTP	git@localhost	::ali/weixin.git					40	Download Merge F	Request Issue
又 ● 微信码	ommented on wa <i>顾目基本已完成。</i>	ll at / weixin			, <i>[</i> 2	司时对中,	文翻译也做	7	3 days ago.

3. 登录 GitLab 界面后,你可以创建自己的项目,别的开发人员也可以加入你的项目。 在右侧的 Projects 框里可以看到当前所有的项目、自己的项目和自己加入的项目。点击 +New Project 按钮可以创建新项目(顶部的+号也一样可以)。

- a) 只需要填写一个项目名称,点击 Create project 按钮就可以创建项目。新项目建 立在你的用户名之下(例如这里是 ali),SSH 访问的 URL 格式为: git@192.168.1.16:ali/example.git。
- b) 新项目创建后,需要通过Git客户端进行初始化以后才能够使用,在新项目页面给 出了通过Git命令行工具初始化的方法。因为我们这里介绍的是使用TortoiseGit 客户端,所以在这里我们先不初始化项目存储库,等到做完第4、5步,设定好用 户密匙后再介绍用TortoiseGit来初始化项目存储库。



Git global setup:

```
git config --global user.name "----"
git config --global user.email "-----"
```

Create Repository



Existing Git Repo?

```
cd existing_git_repo
git remote add origin git@192.168.1.16:ali/example.git
git push -u origin master
```

- 4. 在你的 Windows 客户端,为了能够方便的检入检出项目,需要安装 Git 客户端软件:
 - a) 到此处下载最新版 msysgit 并安装 (基本上一路 Next 即可,因为我们还要安装 tortoisegit,所以没必要选择 "Windows explorer integration")
 - b) 到<u>此处</u>下载最新版 tortoisegit 并安装(基本上一路 Next 即可。如果是在 WindowsXP 上安装, 需要先安装 Windows Installer 4.5); 如果你需要中文包, 可以到<u>这里</u>下载, 直接安装就可以了。

- 5. 因为我们使用 SSH 协议来与 Git 服务器通讯,所以需要生成 SSH 密匙来验证身份:
 - a) 从 Windows 开始菜单选择 TortoiseGit , 点击其下的 Puttygen 工具 , 该工具用 于生成 Putty 支持的 SSH 密匙对。

📗 TortoiseGit		
👷 Help		
🚊 Pageant		
😰 Puttygen		
🔊 Settings (64 bit)		
🗟 TortoiseGitBlame		
🔄 TortoiseIDiff		
🔉 TortoiseMerge		
Website		

b) 点击 Generate 按钮,然后在窗口的空白区域晃动鼠标,一个新的密匙对就生成出来了。将 Public key (公匙)和 Private key (私匙)都保存好,其中 Private key 是你的 TortoiseGit 用来验证身份用的,所以一定不要丢失了。

PuTTY Key Generator	X
File Key Conversions Help	
Key No key.	
Actions	
Generate a public/private key pair	Generate
Load an existing private key file	Load
Save the generated key	Save public key Save private key
Parameters	
Type of key to generate: SSH-1 (RSA)	H-2 RSA 💿 SSH-2 DSA
Number of bits in a generated key:	1024

🕈 PuTTY Key Generat	tor	<u> </u>			
File Key Conversio	ons Help				
Key					
Public key for pasting into OpenSSH authorized_keys file:					
ssh-rsa	ssh-rsa				
AAAAB3NzaC1yc2EA rC9rd1H3+mU5x6Ut3v	AAABJQAAAIEAI/u9sHcXzi ABFhr25KabHpZr4rBI3S5u	KqyMIT3IYCnX9/etNbyCX/UBCWtu/ KNiltEc804o3zQDSbPJGnxf8vRHRT			
mb/QdE+6ENNqaqZJ	hyG47wIAEKo4lfAEict0IHG	GuRvfmqdXT3q90PpF0ThPgbW0=			
rsa-key-20130114		T			
Key fingerprint:	ssh-rsa 1024 fd:26:77:5c:3	30:37:eb:87:b2:39:de.fd:42:b7:b5:b1			
Key comment:	rsa-key-20130114				
Key passphrase:					
Confirm passphrase:					
Actions			ĥ		
Generate a public/priva	ate key pair	Generate			
Load an existing private	e key file	Load			
Save the generated key Save public key Save private key					
Parameters					
Type of key to generate SSH-1 (RSA)	e:	SSH-2 DSA			
Number of bits in a gen	erated key:	1024			

c) 将生成的 Public key 粘贴到 GitLab 站点里你的帐号里。登录 <u>http://192.168.1.6</u>, 点击顶部的头像进入你的 Profile,在 SSH Keys 选单下点击 Add new,把上面我 们生成的 Public key 内容粘贴进来并保存(title 即名字随便起)。完成之后你的身 份就可以被你的 Private key 验证了。

🧐 GITLAB	Profile		+ 🔺	۹ Search
ñ	Account	SSH Keys 0	Design	History
SSH Keys				Add new
SSH key allows yo	u to establish a secure connect	ion between your computer	and GitLab	
Name		Added		
	There are	no SSH keys with access to	o your account.	

d) 从 Windows 开始菜单选择 TortoiseGit,点击其下的 Settings 工具,在对话框里选择 Git 项,填写你的帐号信息和你刚刚生成的 Private key:

양 设置 - TortoiseGit		×
▲ 🔩 常规设置	🔦 Git	
小小 扩展菜单	名字(187):	
₩ 药每1	Rmail.	
₩ 5562	Similar house Divisions' A set	
₩ 颜色3	Signing Key D. tosers (. SSR) ppk	
(編辑器		
🔺 痴 图标覆盖	□ 自动换行: 检查回车 false ▼	
- 100 覆盖程序	││ 在提交日志中缺少Signed-Off-By时给出警告	
	☑ 保存为全局 (G)	
○○○○○○○○○○○○○○○○○○○○○○○○○○○○○○○○○○○○	編辑本地.git/config(l) 編辑全局.git/config(o) 🐉编辑	系统gitconfig(y)
	编辑系	系统gitconfig(w)
Git		
▲ ⑦ Hook 脚本		
🧼 问题跟踪器集成		
		(A) 帮助

- 6. 设定完用户验证密匙后,我们就可以初始化 GitLab 上的项目存储库了:
 - a) 在Windows 文件管理器任意目录下(例如 D:\Dev 新建一个目录,例如 example;
 - b) 进入 D:\Dev\example 目录 鼠标右键点击空白区域 选择"Git Create repository here...",在弹出对话框中点确认即可,这样一个空的本地 Git 存储库就建立了;

	*=	
	亘⁄व(V)	•
	排序方式(O)	•
	分组依据(P)	+
	刷新(E)	
	自定义文件夹(F)	
	粘贴(P)	
	粘贴快捷方式(S)	
	撤消 重命名(U)	Ctrl+Z
	Git Init Here	
	Git Gui	
	Git Bash	
	共享(H)	۲
N	Git Clone	
ľ	Git Create repository here	
R	TortoiseGit	۰.
\$	WinMerge	
s	共享文件夹同步	+
	新建(W)	۲
	扂性(R)	

c) 在该文件夹下创建新文件或者把已有的项目文件拷贝进来,例如我们创建一个空的

README.txt 文本文件。这时 Git 能侦测到存储库被改变了。点击鼠标右键,选择 Git Commit -> "master"…,在弹出对话框里输入提交注释,选择要加入版本 控制的文件(这里就是 README.txt),点确定即可;

	查看(V)	۱.
	排序方式(O)	•
	分组依据(P)	•
	刷新(E)	
	自定义文件夹(F)	
	粘贴(P)	
	粘贴快捷方式(S)	
	撤消 重命名(U)	Ctrl+Z
	Git Add all files now	
	Git Commit Tool	
	Git Gui	
	Git Branch	•
	Git Bash	
	共享(H)	۲
eq.	Git Sync	
7	Git Commit -> "master"	
*	TortoiseGit	•
2	WinMerge	
s	共享文件夹同步	•
	新建(W)	×
	属性(R)	



 d) 这时候你只是在本地实现了使用 Git 管理项目,在上一步点击确定后,会弹出另外 一个对话框指示本地提交成功,点击其左下角的 Push...按钮,把我们的改动递交 到 Git 服务器上去。

😪 D:\Dev\example - Git Command Progress - TortoiseGit
[master (root-commit) 759a4e3] initial commitment 0 files changed create mode 100644 README.txt
Success (172 ms @ 2013/1/17 15:19:37)
Push Close Abort

此时会弹出 Push 对话框,在 Ref - Remote 栏里设定当前分支名为 master;

D:\Dev\example	- Push - TortoiseGit 📃
Ref	25
Local:	master 🔹 🕨
Remote:	master 🔹 🛄
Destination	
Remote:	origin Manage
O Arbitrary URL:	git@192.168.1.16:ali/example.git v
Options Force Overwrite Use Thin Pack (Include Tags Autoload Putty Set upstream/to Always push to Always push to	Existing Branch (May discard changes) For slow network connections) Key rack remote branch the selected remote archive for this local branch the selected remote branch for this local branch
	OK Cancel Help

然后点击 Destination – Remote 栏的 Manage 按钮,在弹出的设置对话框中设

定 Remote 为 origin, URL 为我们从第 3 步创建新项目中得到的 URL (git@192.168.1.16:ali/example.git),Putty则是我们在第5步中创建的Private key(后缀为.ppk),点击 "Add New/Save" 按钮保存这一设置,然后点确定退

👷 Settings - TortoiseGit				x
General General Context M Set Extend Colors 1 Colors 2 Colors 3 Colors 3 Colors 3 Colors 3 Colors 3 Colors 3 Colors 4 Colors 4 Colors 4 Colors 4 Colors 5 Colors 5 Colors 4 Colors 5 Colors 5 Colors 4 Colors 5 Colors 5 Colors 5 Colors 7 Colors 7 Color	Remote:	Remote: URL: Putty Add New Remo	origin git@192.168.1.16:ali/example. D:\Users\ \.ssh\ .p /Save we	git opk
		确定	取消 应用(A)	帮助

出这个对话框。

确认提交 Push 操作之后,你会得到一个 Success 的结果。如果不是这样的蓝色字样,你就得好好检查你的设置了。到此,我们的新项目存储库就初始化完成了。



- 7. 如何克隆 Git 服务器上的项目到本地:
 - a) 在 Windows 文件管理器任意目录下 (例如 D:\Dev), 鼠标右键点击空白区域, 可 以看到 Git Clone 选项:

	查看(V)	•
	排序方式(O)	•
	分组依据(P)	•
	刷新(E)	
	自定义文件夹(F)	
	粘贴(P)	
	粘贴快捷方式(S)	
	撤消 重命名(U)	Ctrl+Z
	Git Init Here	
	Git Gui	
	Git Bash	
	共享(H)	+
8	Git 克隆	
ť	Git 在这里创建版本库(y)	
R	TortoiseGit	•
2	WinMerge	
S	共享文件夹同步	•
	新建(W)	•
	属性(R)	

b) 在弹出的对话框里,填写你要使用的项目地址,该地址格式可以从 GitLab 的项目 信息里看到。



👷 Git克隆 - 1	TortoiseGit				
克隆存在的版本库					
URL:	git@192.168.1.16:weixin.git				
目录(D)	D:\Dev\weixin				
	□ 深度 0 □ 递归 □ 克隆成纯版本库				
☑ 加载p	utty密钥(🏦 D:\Users\\.ssh'ppk 🔻				
─从SVN版本£	从SVN版本库				
□ 从SVN	版本库				
	主干(T) trunk □标签(c tags □分支(h) branches				
	以(F) 0 日户名(n):				
	确定 取消 帮助				

- c) 点击确认按钮, TortoiseGit 就会把该项目的文件下载到本地。你就可以在本地进 行开发了。
- 8. 新的项目一般来说只有一个目录,我们需要在该目录创建文件或者把文件拷贝到该 目录。这时候,新的文件是没有被 Git 管理起来的,我们需要提交我们的更改。提交分 两步,首先是本地的提交,然后是推送到Git服务器上。实际上你可以在多次本地提交 之后,再推送到服务器上。
- a) 本地提交,需要在项目目录下,点击鼠标右键,在弹出菜单上选择 Git Commit (Git 提交),然后在弹出对话框里输入此次改动描述、时间、署名等等,点确定 即可。

排序方式(O) > 分组依据(P) > 刷新(E) 自定义文件夹(F) 粘贴(P) 粘贴快速方式(S) 撤消重命名(U) Ctrl+Z Git Add all files now Git Commit Tool Git Branch > Git Bash 共享(H) > Git 同步 Git 同步 Yange 新建(W) > 新建(W) > 属性(R)		查看(V)	+
分組依据(P) ・ 刷新(E) 自定义文件夹(F) 粘贴(P) 粘贴快捷方式(S) 撤消重命名(U) Ctrl+Z Git Add all files now Git Add all files now Git Add all files now Git Bash 技事(H) ・ ダ Git 周步 Git 見支(C) -> "master" び TortoiseGit ・ 新建(W) ・ 編載(W)		排序方式(O)	+
刷新(E) 自定义文件夹(F) 粘贴(P) 粘贴快速方式(S) 撤消 重命名(U) Ctrl+Z Git Add all files now Git Commit Tool Git Branch Git Bash 共享(H) Git 同步 Git 現交(C) -> "master" Git 現交(C) -> "master" WinMerge 共享文件夹同步 新建(W) 属性(R)		分组依据(P)	•
自定义文件夹(F) 粘贴(P) 粘贴快捷方式(S) 撤消重命名(U) Ctrl+Z Git Add all files now Git Commit Tool Git Gui Git Branch jtp:(H) Git 同步 Git 建交(C) -> "master" VinMerge 共享文件夹同步 新建(W) 属性(R)		刷新(E)	
粘贴(P) 粘助快捷方式(S) 撤消重命名(U) Ctrl+Z Git Add all files now Git Commit Tool Git Commit Tool Git Gui Git Branch Git Branch Git Bash 共享(H) (Git 同步 Git 同步 Git 最支(C) -> "master" (MinMerge 共享文件共同步 新建(W) 属性(R)		自定义文件夹(F)	
粘贴快速方式(S) 撤消重命名(U) Ctrl+Z Git Add all files now Git Commit Tool Git Gui Git Bistory Git Branch Git Bash 共享(H) Git 同步 Git 現交(C) -> "master" YortoiseGit WinMerge 共享文件共同步 新建(W) 属性(R)		粘贴(P)	
 撤消重命名(U) Ctrl+Z Git Add all files now Git Commit Tool Git Commit Tool Git History Git Gui Git Branch 其享(H) Git 昆ash 其享(H) Git 昆交(C) -> "master" Git 昆交(C) -> "master" TortoiseGit WinMerge 共享文件共同步 新建(W) 属性(R) 		粘贴快捷方式(S)	
Git Add all files now Git Commit Tool Git History Git Gui Git Branch Cit Bash 共享(H) Git 同步 Git 同步 Git 最交(C) -> "master" Cit 提交(C) -> "master" Kit 操交(C) -> "master" Kit 操文(C) -> "master" Kit 操文(C) -> "master" Common Common Commo		撤消 重命名(U)	Ctrl+Z
Git Commit Tool Git History Git Gui Git Branch Git Bash 共享(H) Git 同步 Git 最交(C) -> "master" Cit 提交(C) -> "master" VinMerge 共享文件共同步 新建(W) 属性(R)		Git Add all files now	
Git History Git Gui Git Branch Cit Bash 注享(H) Git 同步 Git 提交(C) -> "master" Cit 提交(C) -> "master" VinMerge 共享文件共同步 新建(W) 属性(R)		Git Commit Tool	
Git Gui Git Branch Git Bash 共享(H) Git 同步 Git 最交(C) -> "master" Git 提交(C) -> "master" VinMerge 共享文件夹同步 新建(W) 属性(R)		Git History	
Git Branch > Git Bash 共享(H) > Git 同步 Git 現交(C) -> "master" TortoiseGit > WinMerge 共享文件夹同步 新建(W) > 属性(R)		Git Gui	
Git Bash 共享(H) Git 同步 Git 提交(C) -> "master" TortoiseGit WinMerge 共享文件共同步 新建(W) 属性(R)		Git Branch	•
共享(H) > Git 同步 Git 提交(C) -> "master" TortoiseGit > WinMerge 共享文件夹同步 > 新建(W) > 属性(R)		Git Bash	
Git 同步 Git 提交(C) -> "master" TortoiseGit WinMerge 共享文件共同步 新建(W) 属性(R)		共享(H)	×
 Git 提交(C) -> "master" TortoiseGit WinMerge 共享文件夹同步 新建(W) 属性(R) 	7	Git 同步	
 TortoiseGit WinMerge 共享文件夹同步 新建(W) 属性(R) 	2	Git 提交(C) -> "master"	
WinMerge 共享文件夹同步 ▶ 新建(W) ▶ 属性(R)	2	TortoiseGit	•
 Ⅰ 共享文件夹同步 ▶ 新建(W) ▶ 属性(R) 	Σ	WinMerge	
新建(₩) ▶ 属性(R)	S	共享文件夹同步	•
属性(R)		新建(W)	۱.
		属性(R)	

☆* D:\Dev\weixin - 提交 - TortoiseGit					
提交至: master 日志信息(M):		🔲 新建	₿分支		
修改了自述文件内容	ŧ				
Signed-off-by:	• • • • • • • • • • • • • • • • • • •				
同族たちを担うか					4/1
□ 16 成上次提父(L)		204.01.44	. C		
☑ 设置提父口期(0)		2013/ 1/1	L4 (18:12:33 🐨 添加会者通过(5)
○ 变更列表(双击文件查 核查 全部(A)无(N	昏差异)) 未販調	: 本控制 ē	已经版	反本技	计制已增加已删除已修改文件子模块
路径	扩	状态	増.	刪.	
README.txt	.txt	已修改	1	0	G
☑ 显示无版本控制的文	[件 <mark>(U)</mark>	1			选择了1个文件,共有1个文件
🔲 不自动选择子模块					查看补丁>>
☑整个工程(₩) □ 仅仅消息(y)				(确定(O) 取消 帮助

b) 推送到 Git 服务器的动作,一般是在你认为你的改动达到一定要求,或者当天下班 之前需要保存工作内容,才进行的。在项目目录下,点击鼠标右键,选择 TortoiseGit



-> Push (推送); 然后在弹出的对话框里选择合适选项, 点确定即可。

St D:\Dev\weixin -	推送 - TortoiseGit	×		
引用 ■ 推送所有分支	:(P)			
本地(L)	master	• •		
远端 (R)	master			
目标				
◎ 远端(m)	origin 👻	管理(g)		
◎ 其他URL		-		
选项				
 一 强制覆盖已存 使用小包(为情) 	在的分支(可能丢失改动)(F) 漫速网络连接)(P)			
🔲 包含标签(T)				
 ✓ 自动加载putty密钥(A) □ 设置上游/跟踪远程分支(S) □ 分見4000000000000000000000000000000000000				
□ 忌走推达这千年地方又到这样印码辆 □ 总是推送这个本地分支到选择的远端分支				
	确定 取消	帮助		