

gogs 简单快速搭建 git

gogs 简单快速搭建 git.....	1
什么是 Gogs?.....	2
开发目的.....	3
开源组件.....	3
功能特性.....	3
环境要求.....	4
安装数据库.....	4
安装其它要求.....	4
安装 Gogs.....	5
二进制安装.....	5
备注.....	5
如何使用下载好的压缩包?	5
如何通过二进制升级?	5
v0.9.13 @ 2016-03-19.....	6
源码安装.....	6
安装依赖.....	6
基本依赖.....	6
第三方包.....	6
安装 Go 语言.....	6
下载.....	6
设置环境.....	7
安装 Gogs.....	7
构建 develop 分支版本.....	7
测试安装.....	8
使用标签构建.....	8
包管理安装.....	8
Packager.io.....	8
Arch Linux.....	9
注册为 Windows 服务运行.....	9
前提要求.....	9
配置与运行.....	14
配置文件.....	14
默认配置文件.....	14
自定义配置文件.....	14
为什么要这么做?	14
运行 Gogs 服务.....	14
开发者模式.....	14
部署模式.....	15
从二进制升级.....	15
从源码升级.....	16
Web 钩子.....	16

事件信息.....	17
授权认证.....	21
LDAP.....	21
本地化 Gogs 应用.....	25
贡献翻译.....	25
自定义本地化文件.....	26
配置文件手册.....	26
概览.....	26
Repository (repository).....	26
UI (ui).....	27
UI - Admin (ui.admin).....	27
Markdown (markdown).....	27
Server (server).....	27
Database (database).....	27
Security (security).....	28
Service (service).....	28
Webhook (webhook).....	28
Mailer (mailer).....	29
Cache (cache).....	29
Session (session).....	29
Picture (picture).....	29
Attachment (attachment).....	30
Log (log).....	30
Cron (cron).....	30
Cron - Update Mirrors (cron.update_mirrors).....	30
Cron - Repository Health Check (cron.repo_health_check).....	30
Cron - Repository Statistics Check (cron.check_repo_stats).....	30
Git (git).....	31
Other (other).....	31
常见问题.....	31
部署应用.....	31
管理权限.....	34
仓库管理.....	34
其它.....	34
说明.....	35

什么是 Gogs?

Gogs (Go Git Service) 是一款极易搭建的自助 Git 服务。

开发目的

Gogs 的目标是打造一个最简单、最快速和最轻松的方式搭建自助 Git 服务。原生支持中文。使用 Go 语言开发使得 Gogs 能够通过独立的二进制分发，并且支持 Go 语言支持的 **所有平台**，包括 Linux、Mac OS X、Windows 以及 ARM 平台。

开源组件

Web 框架: Macaron

UI 组件:

- Semantic UI
- GitHub Octicons
- Font Awesome

前端插件:

- DropzoneJS
- highlight.js
- clipboard.js
- emojify.js
- jQuery Date Time Picker
- jQuery MiniColors

ORM: Xorm

数据库驱动:

- github.com/go-sql-driver/mysql
- github.com/lib/pq
- github.com/mattn/go-sqlite3

以及其它所有 Go 语言的第三方包依赖。

功能特性

支持活动时间线

支持 SSH 以及 HTTP/HTTPS 协议

支持 SMTP、LDAP 和反向代理的用户认证

支持反向代理子路径

支持用户、组织和仓库管理系统

支持仓库和组织级别 Web 钩子（包括 Slack 集成）

支持仓库 Git 钩子和部署密钥

支持仓库工单（Issue）、合并请求（Pull Request）和 Wiki

支持添加和删除仓库协作者

支持 Gravatar 以及自定义源

支持邮件服务

支持后台管理面板

支持 MySQL、PostgreSQL、SQLite3 和 TiDB（实验性支持） 数据库

支持多语言本地化（15 种语言）

环境要求

数据库（选择以下一项）：

- MySQL：版本 $\geq 5.5.3$
- PostgreSQL
- 或者 **什么都不安装** 直接使用 SQLite3 或 TiDB

git (bash)：

- 服务端和客户端均需版本 $\geq 1.7.1$
- Windows 系统建议使用最新版

SSH 服务器：

- 如果您只使用 HTTP/HTTPS 或者内置 SSH 服务器的话请忽略此项
- 推荐 Windows 系统使用 Cygwin OpenSSH 或 Copssh

安装数据库

Gogs 支持 MySQL、PostgreSQL、SQLite3 和 TiDB（实验性支持），请根据您的选择进行安装：

MySQL（引擎：INNODB）

PostgreSQL

注意事项 您可以使用 `etc/mysql.sql` 来自动创建名为 `gogs` 的数据库。如果您选择手动创建，请务必将编码设置为 `utf8mb4`。

安装其它要求

Mac OS X

假设您已经安装 Homebrew：

```
$ brew update
$ brew install git
```

Debian/Ubuntu

```
$ sudo apt-get update  
$ sudo apt-get install git
```

Windows

下载并安装 Git

安装 Gogs

二进制安装
源码安装

二进制安装

目前只提供最近 5 次版本发布的二进制下载，更多版本下载请前往 [GitHub](#) 查看。

所有的版本都支持 MySQL 和 PostgreSQL 作为数据库，并且均使用构建标签 (build tags) `cert` 进行构建。需要注意的是，不同的版本的支持状态有所不同，请根据实际的 Gogs 提示进行操作。

备注

`mws` 表示提供内置 Windows 服务支持，如果您使用 NSSM 请使用另外一个版本。

如何使用下载好的压缩包？

解压压缩包。
使用命令 `cd` 进入到刚刚创建的目录。
执行命令 `./gogs web`，然后，就没有然后了。

如何通过二进制升级？

下载最新版的压缩包。
删除当前的 `templates` 目录。

解压压缩包并将所有内容复制粘贴到相应（当前）的位置。

v0.9.13 @ 2016-03-19

系统名称	系统类型	SQLite	TiDB	PAM	下载 (GitHub)
Linux	386	✓	✗	✓	HTTPS: ZIP TAR.GZ - CDN: ZIP TAR.GZ
Linux	amd64	✓	✗	✓	HTTPS: ZIP TAR.GZ - CDN: ZIP TAR.GZ
Linux	arm	✗	✗	✗	HTTPS: ZIP - CDN: ZIP
Raspberry Pi	v2	✓	✗	✓	HTTPS: ZIP - CDN: ZIP
Windows	386	✓	✗	✗	HTTPS: ZIP ZIP w/ mws - CDN: ZIP ZIP w/mws
Windows	amd64	✓	✗	✗	HTTPS: ZIP ZIP w/ mws - CDN: ZIP ZIP w/mws
Mac OS	amd64	✓	✗	✗	HTTPS: ZIP - CDN: ZIP

源码安装

安装依赖

基本依赖

Go 语言：版本 \geq 1.4

第三方包

请通过 `gopmfile` 查看完整的第三方包依赖列表。一般情况下只有在您为 Gogs 制作构建包的时候才会用到。

安装 Go 语言

如果您的系统已经安装要求版本的 Go 语言，可以跳过此小节。

下载

您可以通过以下方式安装 Go 语言到 `/home/git/local/go` 目录:

```
sudo su - gitcd ~# create a folder to install 'go'
mkdir local# Download go (change go$VERSION.$OS-$ARCH.tar.gz to the latest
realse)
wget https://storage.googleapis.com/golang/go$VERSION.$OS-$ARCH.tar.gz# expand
it to ~/local
tar -C /home/git/local -xzf go$VERSION.$OS-$ARCH.tar.gz
```

设置环境

请设置和您系统环境对应的路径:

```
sudo su - gitcd ~echo 'export GOROOT=$HOME/local/go' >> $HOME/.bashrc
echo 'export GOPATH=$HOME/go' >> $HOME/.bashrc
echo 'export PATH=$PATH:$GOROOT/bin:$GOPATH/bin' >> $HOME/.bashrc
source $HOME/.bashrc
```

安装 Gogs

常用的安装方式:

```
# 下载并安装依赖
$ go get -u github.com/gogits/gogs
# 构建主程序
$ cd $GOPATH/src/github.com/gogits/gogs
$ go build
```

如果您已经安装 `gopm`, 则可以尝试使用以下方式安装:

```
# 检查更新 Gopm
$ gopm update -v
# 下载并构建二进制
$ gopm bin -u -v gogs -d path/to/anywhere
```

构建 develop 分支版本

如果您想要安装 `develop` (或其它) 分支版本, 则可以通过以下命令:

```
$ mkdir -p $GOPATH/src/github.com/gogits
$ cd $GOPATH/src/github.com/gogits
# 请确保没有使用 “https://github.com/gogits/gogs.git”
$ git clone --depth=1 -b develop https://github.com/gogits/gogs
```

```
$ cd gogs
$ go get -u ./...
$ go build
```

测试安装

您可以通过以下方式检查 Gogs 是否可以正常工作：

```
cd $GOPATH/src/github.com/gogits/gogs
./gogs web
```

如果您没有发现任何错误信息，则可以使用 Ctrl-C 来终止运行。

使用标签构建

Gogs 默认并没有支持一些功能，这些功能需要在构建时明确使用构建标签（build tags）来支持。

目前使用标签构建的功能如下：

- sqlite3/tidb: SQLite3/TiDB 数据库支持
- pam: PAM 授权认证支持
- cert: 生成自定义证书支持
- miniwinsvc: Windows 服务内置支持(或者您可以使用 NSSM 来创建服务)

备注 使用 TiDB 需要按照 说明 进行额外安装

例如，您需要支持以上所有功能，则需要先删除 \$GOPATH/pkg/{GOOS_GOARCH}/github.com/gogits/gogs 目录，然后执行以下命令：

```
$ go get -u -tags "sqlite tidb pam cert" github.com/gogits/gogs
$ cd $GOPATH/src/github.com/gogits/gogs
$ go build -tags "sqlite tidb pam cert"
```

安装完成后可继续参照 配置与运行。

包管理安装

Packager.io

当前支持 Ubuntu 12.04 + 14.04、CentOS 6 + 7 和 Debian 7 + 8 版本。
可以从 packager.io 获取到相应包管理资源。

Arch Linux

- 基于源码构建，作者 @fanningert:
 - master 分支：AUR GitHub
 - dev 分支：AUR GitHub
 - 版本：AUR GitHub

能可以到 [Arch Linux Wiki entry](#) 查看完整说明。

注册为 Windows 服务运行

前提要求

想要使 Gogs 通过 Windows 服务的方式运行，必须满足以下两个条件：

使用 `miniwinSvc` 构建标签获得内置 Windows 服务支持。

不使用 `miniwinSvc` 构建标签并通过 `NSSM` 注册为服务。

First of all, you may choose to lock down file and directory permissions. Just keep in mind the paths to which Gogs requires write access, which includes the Gogs repository ROOT.

The following changes are made in `C:\Gogs\custom\conf\app.ini`:

```
RUN_USER = COMPUTERNAMES$
```

Sets Gogs to run as the local system user.

COMPUTERNAME is whatever the response is from `echo %COMPUTERNAME%` on the command line. If the response is `USER-PC` then `RUN_USER = USER-PC$`:

```
[server]DOMAIN = gogsPROTOCOL = httpHTTP_ADDR = 127.0.1.1HTTP_PORT = 80OFFLINE_MODE = trueROOT_URL = http://gogs/
```

Moves Gogs' listen port to 80 on the local interface subnet, and tells the Gogs HTTPd that its virtual host is the “gogs” domain. This lets you forgo including a port number when browsing, and prevents collision with other localhost services. The IP address can be anything in the range `127.0.0.2 - 127.254.254.254`, so long as it's unique to Gogs.

To complete that network route, open `Notepad.exe` as administrator and include the following in `C:\Windows\System32\drivers\etc\hosts`:

```
# Go Git Service local HTTPd127.0.1.1 gogs
```

The hosts file entry will guarantee that all requests to the “gogs” domain are captured by your localhost interface. In a web browser, generally, `gogs/` in the

address bar is sufficient to trigger that route.

Use Builtin Functionality

Open a command prompt (cmd.exe) as an Administrator. Run the following command:

```
C:\> sc create gogs start= auto binPath= ""C:\gogs\gogs.exe" web --config "C:\gogs\conf\app.ini""
```

Ensure there is a space after each =. You can choose to add additional Arguments to further modify your service, or modify it manually in the service management console.

To start the service run following command:

```
C:\> net start gogs
```

You should see following output:

```
The gogs service is starting.The gogs service was started successfully.
```

Use NSSM

Get the `nssm.exe` needed for your machine (32 or 64 bit; they' re packaged together in Iain' s zip file), and place it in a directory that is in (or will be added to) your `%PATH%` environment variable.

Open a command line as administrator and do following command to configure Gogs as a service:

```
C:\>nssm install gogs
```

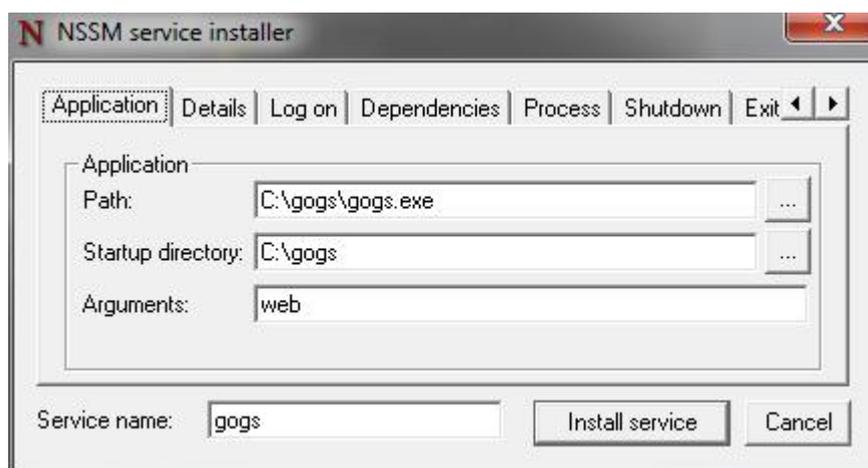
“NSSM service installer” will appear. Configure it as follows:

Application tab:

Path: C:\gogs\gogs.exe

Startup directory: C:\gogs

Arguments: web



Details tab:

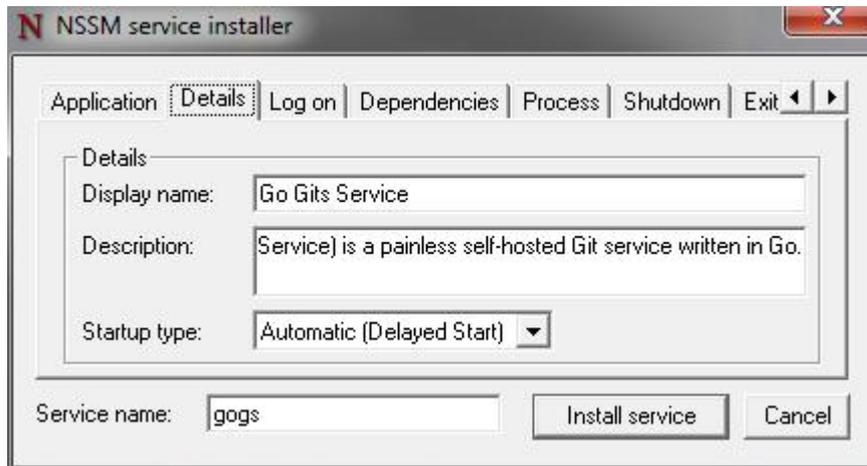
Display name: Go Gits Service

Description: Gogs (Go Git Service) is a painless self-hosted Git service.

Startup type: Automatic (Delayed Start)

Note that we' ve chosen **delayed start**, so that the service will not impact

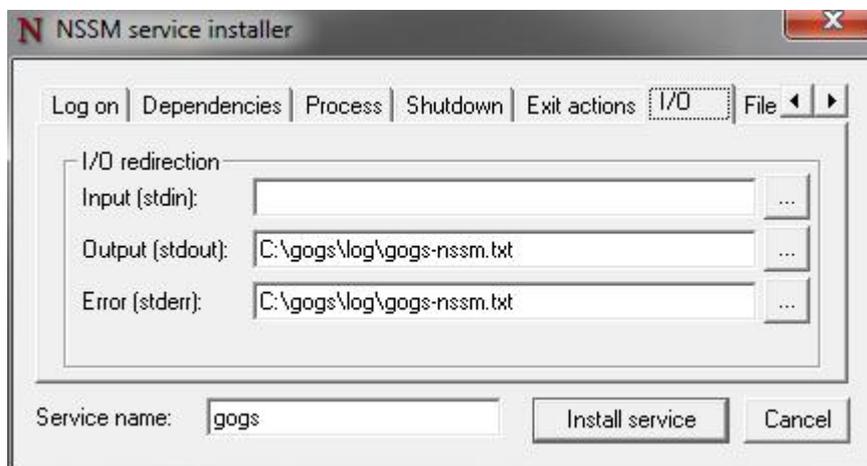
the early boot time. Gogs will start two minutes after the non-delayed services.



I/O tab:

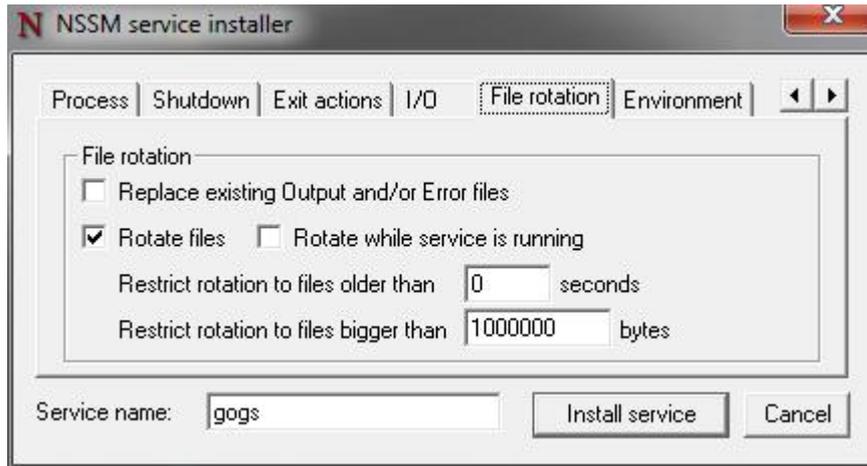
- Output (stdout): C:\gogs\log\gogs-nssm.txt
- Error (stderr): C:\gogs\log\gogs-nssm.txt

That will capture all text output that you would normally receive from Gogs on the command line console, and log it to that file instead.



File rotation tab:

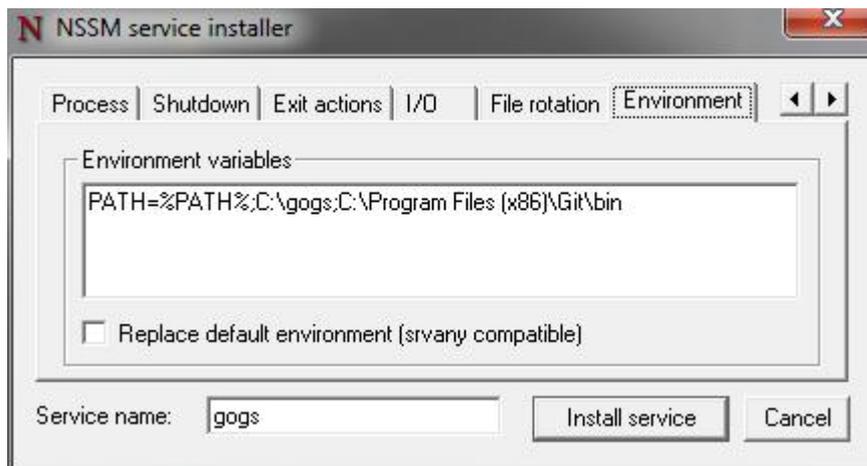
- Check: Rotate files
- Restrict rotation to files bigger than: 1000000 bytes



Environment tab:

- Environment variables: `PATH=%PATH%;C:\gogs;C:\Program Files (x86)\Git\bin`

That is a guarantee that both `gogs.exe` and `git.exe` will be on the Gogs service's path variable during runtime.



Click “Install service”, and you should be confirmed that it succeeded.

If it failed, refer back to the command line console you started, for the error message. When it succeeds, go to command line and do:

```
nssm start gogs
```

You should see:

```
gogs: START: The operation completed successfully.
```

Check that this is true by opening `C:\gogs\log\gogs-nssm.txt`. You should see Gogs' start up procedures, ending with:

```
timestamp [I] Run Mode: Production
timestamp [I] Listen: http://127.0.1.1:80
```

Now point your web browser at `http://gogs/` and log in.

Gogs is running as a service, and will not need to be run manually, unless something goes wrong. NSSM will attempt to restart the service for you, if the Gogs server crashes.

When you need to restart it after `app.ini` changes, go to an administrator command line after the changes, and do:

```
nssm restart gogs
```

See the **configuration cheat sheet** for reference of the `custom\conf\app.ini` settings.

An example of the logging and error handling in action:

```
[log]ROOT_PATH = C:\gogs\log
```

At the time of writing this line, it will cause the following to print in `C:\gogs\log\gogs-nssm.txt`:

```
timestamp [T] Custom path: C:/gogs/custom
timestamp [T] Log path: C:\gogs\log
timestamp [I] Gogs: Go Git Service x.y.z
timestamp [log.go:294 Error()] [E] Fail to set logger(file): invalid character 'g'
in string escape code
```

This is what was expected in `C:\custom\conf\app.ini`:

```
ROOT_PATH = C:/gogs/log
```

And this was the nssm interaction while solving it:

```
C:\>nssm restart gogsgogs: STOP: The operation completed successfully.gogs:
Unexpected status SERVICE_PAUSED in response to START control.
C:\>nssm start gogsgogs: START: An instance of the service is already running.
C:\>nssm stop gogsgogs: STOP: The operation completed successfully.
C:\>nssm start gogsgogs: Unexpected status SERVICE_PAUSED in response to START control.
C:\>nssm restart gogsgogs: STOP: The operation completed successfully.gogs: START:
The operation completed successfully.
```

配置与运行

配置文件

默认配置文件

默认配置都保存在 `conf/app.ini`，您 **永远不需要** 编辑它。该文件从 `v0.6.0` 版本开始被嵌入到二进制中。

自定义配置文件

那么，在不允许修改默认配置文件 `conf/app.ini` 的情况下，怎么才能自定义配置呢？很简单，只要创建 `custom/conf/app.ini` 就可以！在 `custom/conf/app.ini` 文件中修改相应选项的值即可。

例如，需要改变仓库根目录的路径：

```
[repository]ROOT = /home/jiahuachen/gogs-repositories
```

当然，您也可以修改数据库配置：

```
[database]PASSWD = root
```

为什么要这么做？

乍一看，这么做有些复杂，但是这么做可以有效地保护您的自定义配置不被破坏：

从二进制安装的用户，可以直接替换二进制及其它文件而不至于重新编写自定义配置。

从源码安装的用户，可以避免由于版本管理系统导致的文件修改冲突。

运行 Gogs 服务

开发者模式

- 您需要在 `custom/conf/app.ini` 文件中将选项 `security -> INSTALL_LOCK` 的值设置为 `true`。
-

您可以使用超能的 `make` 命令：

```
$ make
$ ./gogs web
```

您可以在 `Gogs` 源码目录使用命令 `bra run`：

安装 `bra` 工具：`go get -u github.com/Unknwon/bra`

部署模式

脚本均放置在 `scripts` 目录，但请在仓库根目录执行它们

- `Gogs` 支持多种方式的启动：
 - 普通：只需执行 `./gogs web`
 - 守护进程：详见 `scripts` 文件夹
- 然后访问 `/install` 来完成首次运行的配置工作

从二进制升级

相关下载可以从 [二进制安装](#) 页面查看。

首先，确认当前安装的位置：

```
# 默认位置在 git 用户下的家目录
```

```
$ sudo su - git
```

```
$ cd ~
```

```
$ pwd
```

```
/home/git
```

```
$ ls
```

```
gogs gogs-repositories
```

然后将当前目录移动到另一个临时的位置，但不是删除！

```
$ mv gogs gogs_old
```

下载并解压新的二进制:

```
# 请根据系统和类型获取相应的二进制版本
$ wget https://dl.gogs.io/gogs_v$VERSION_$OS_$ARCH.tar.gz
$ tar -zxvf gogs_v$VERSION_$OS_$ARCH.tar.gz
$ ls
gogs gogs_old gogs-repositories gogs_v$VERSION_$OS_$ARCH.tar.gz
复制 custom、data 和 log 目录到新解压的目录中:
$ cp -R gogs_old/custom gogs
$ cp -R gogs_old/data gogs
$ cp -R gogs_old/log gogs
最后, 运行并打开浏览器进行测试:
$ cd gogs
$ ./gogs web
```

从源码升级

升级 Gogs 的一般步骤:

```
# 更新源码以及依赖
$ go get -u github.com/gogits/gogs

$ cd $GOPATH/src/github.com/gogits/gogs
# 移除旧的二进制
$ rm gogs
# 或将旧的二进制进行备份
$ mv gogs gogs.$(date +%Y-%m-%d).old
# 重新构建 Gogs
$ go build
```

其它操作:

- 构建 develop 分支版本
- 测试安装
- 使用标签构建

Web 钩子

Gogs 支持针对仓库事件的 Web 钩子服务,您可以在仓库的设置相关页面中找到 (/:username/:reponame/settings/hooks)。所有的事件推送均为 POST 请求,目前支持 Gogs 和 Slack 两种格式的内容。

事件信息

以下为 Gogs 向 Payload URL 发送的事件信息示例:

```
{
  "secret": "",
  "ref": "refs/heads/develop",
  "before": "47df562ceddfaab3471e635e59039c03f47808e2",
  "after": "2cee0f84c0c62fa85382258705353c7d24eb7fee",
  "compare_url":
  "https://try.gogs.io/gogs/gogs/compare/47df562ceddfaab3471e635e59039c03f47808e2...2cee0f84c0c62fa85382258705353c7d24eb7fee",
  "commits": [
    {
      "id": "2cee0f84c0c62fa85382258705353c7d24eb7fee",
      "message": "Revert \"fix CI...\"\n\nThis reverts commit 94b2816446d1d700d1af0ec166e63375da6612f3.\n",
      "url":
      "https://try.gogs.io/gogs/gogs/commit/2cee0f84c0c62fa85382258705353c7d24eb7fee",
      "author": {
        "name": "Unknwon",
        "email": "u@gogs.io",
        "username": "unknwon"
      }
    }
  ]
}
```

```
},  
  
{  
  
  "id": "94b2816446d1d700d1af0ec166e63375da6612f3",  
  
  "message": "fix CI...\n",  
  
  "url":  
"https://try.gogs.io/gogs/gogs/commit/94b2816446d1d700d1af0ec166e6337  
5da6612f3",  
  
  "author": {  
  
    "name": "Unknwon",  
  
    "email": "u@gogs.io",  
  
    "username": "unknwon"  
  
  }  
  
},  
  
{  
  
  "id": "8411b50f5d4e3b30d7d601612ee2aa5e4921c968",  
  
  "message": "work on #1882\n",  
  
  "url":  
"https://try.gogs.io/gogs/gogs/commit/8411b50f5d4e3b30d7d601612ee2aa5  
e4921c968",  
  
  "author": {  
  
    "name": "Unknwon",  
  
    "email": "u@gogs.io",  
  
    "username": "unknwon"  
  
  }  
  
},
```

```
{  
  "id": "8a87bee4346968e280e9b9a6e56373c1d2e1c357",  
  "message": "what's wrong with go tip?\n",  
  "url":  
"https://try.gogs.io/gogs/gogs/commit/8a87bee4346968e280e9b9a6e56373c1d2e1c357",  
  "author": {  
    "name": "Unknwon",  
    "email": "u@gogs.io",  
    "username": "unknwon"  
  }  
},  
{  
  "id": "1dfa693a5cd221fa43f10df3a9dc216753f82547",  
  "message": "fix CI!!\n",  
  "url":  
"https://try.gogs.io/gogs/gogs/commit/1dfa693a5cd221fa43f10df3a9dc216753f82547",  
  "author": {  
    "name": "Unknwon",  
    "email": "u@gogs.io",  
    "username": "unknwon"  
  }  
}  
],
```

```
"repository": {  
  "id": 140,  
  "name": "gogs",  
  "url": "https://try.gogs.io/gogs/gogs",  
  "description": "Gogs(Go Git Service) is a painless self-hosted Git  
Service written in Go.",  
  "website": "",  
  "watchers": 6,  
  "owner": {  
    "name": "gogs",  
    "email": "u@gogs.io",  
    "username": "gogs"  
  },  
  "private": false  
},  
"pusher": {  
  "name": "unknwon",  
  "email": "u@gogs.io",  
  "username": "unknwon"  
},  
"sender": {  
  "login": "unknwon",  
  "id": 1,
```

```
"avatar_url":  
"https://try.gogs.io///1.gravatar.com/avatar/d8b2871cdac01b57bbda2371  
6cc03b96"  
  
}}
```

授权认证

LDAP

基于 BindDN 和 simple auth 的 LDAP 授权方式共享以下字段：

认证名称（必填）

A name to assign to the new method of authorization.

主机地址（必填）

The address where the LDAP server can be reached.

Example: mydomain.com

主机端口（必填）

The port to use when connecting to the server.

Example: 636

启用 TLS 加密 (可选)

Whether to use TLS when connecting to the LDAP server.

管理员过滤规则 (可选)

An LDAP filter specifying if a user should be given administrator privileges. If a user accounts passes the filter, the user will be privileged as an administrator.

Example: (objectClass=adminAccount)

用户名属性 (可选)

The attribute of the user's LDAP record containing the user name. Given attribute value will be used for new Gogs account user name after first successful sign-in. Leave empty to use login name given on sign-in form.

This is useful when supplied login name is matched against multiple attributes, but only single specific attribute should be used for Gogs account name, see "User Filter".

例如: uid

名字属性 (可选)

The attribute of the user' s LDAP record containing the user' s first name. This will be used to populate their account information.

Example: givenName

姓氏属性（可选）

The attribute of the user' s LDAP record containing the user' s surname This will be used to populate their account information.

Example: sn

邮箱属性（必填）

The attribute of the user' s LDAP record containing the user' s email address. This will be used to populate their account information.

Example: mail

基于 BindDN 需要填充以下字段:

绑定 DN（可选）

The DN to bind to the LDAP server with when searching for the user. This may be left blank to perform an anonymous search.

Example: cn=Search,dc=mydomain,dc=com

绑定密码（可选）

The password for the Bind DN specified above, if any. Note: The password is stored in plaintext at the server. As such, ensure that your Bind DN has as few privileges as possible.

用户搜索基准 (必填)

The LDAP base at which user accounts will be searched for.

Example: ou=Users,dc=mydomain,dc=com

用户过滤规则 (必填)

An LDAP filter declaring how to find the user record that is attempting to authenticate. The %s matching parameter will be substituted with login name given on sign-in form.

Example: (&(objectClass=posixAccount)(uid=%s))

To substitute more than once %[1]s should be used instead, eg. when matching supplied login name against multiple attributes such as user identifier, email or even phone number.

Example:

(&(objectClass=Person)(|(uid=%[1]s)(mail=%[1]s)(mobile=%[1]s)))

基于 simple auth 需要填充以下字段:

User DN (必填)

A template to use as the user's DN. The %s matching parameter will be substituted with login name given on sign-in form.

Example: cn=%s,ou=Users,dc=mydomain,dc=com

Example: uid=%s, ou=Users, dc=mydomain, dc=com

用户过滤规则（必填）

•

An LDAP filter declaring when a user should be allowed to log in. The %s matching parameter will be substituted with login name given on sign-in form.

Example: (&(objectClass=posixAccount)(cn=%s))

Example: (&(objectClass=posixAccount)(uid=%s))

本地化 Gogs 应用

Gogs 项目从 v0.5.0 版本开始支持应用的本地化，而您只需要鼠标单击就可实现即时语言切换。

在 custom/conf/app/ini 文件中启用它们（默认启用全部）：

```
[i18n]LANGS =  
en-US, zh-CN, zh-HK, de-DE, fr-FR, nl-NL, lv-LV, ru-RU, ja-JP, es-ES, pt-BR, pl-  
PL, bg-BG, it-IT  
ITNAMES = English, 简体中文, 繁體中  
文, Deutsch, Français, Nederlands, Latviešu, Р у с с к и й, 日本  
語, Español, Português do Brasil, Polski, б ъ л г а р с к и, Italiano
```

贡献翻译

到 [Crowdin](#) 注册一个用户然后填补未翻译的字段。

有时候您会发现您无法将一个句子准确地从英文翻译到您的语言，没关系，这很正常！只需要使用您的语言将中心思想表达出来就可以，而不需要斤斤计较。

如果您想要测试翻译好的本地化文件，但又不想涉及到 Git 历史的变动，您可以将本地化文件放至 custom/conf/locale/<file> 然后重启 Gogs。

自定义本地化文件

如果您对官方的本地化翻译不够满意，可以修改其中的部分字段并保存到 `custom/conf/locale/locale_<lang>.ini` 中，然后重启 Gogs。

配置文件手册

- 本手册会详尽地描述有关 Gogs 配置文件的选项，帮助您更好地理解和使用 Gogs。
- 请记住，任何修改都是发生在 `custom/conf/app.ini` 自定义配置文件中，该文件的具体位置与您的设置有关。
- 完整的默认设置可以通过 `app.ini` 文件查看。如果您看到类似 `%(X)s` 字符，这是由 `ini` 提供的递归取值的特性。
- 任何带有 **!** 标记的配置选项表示除非您完全了解修改后的影响，否则请保持默认值。

概览

`APP_NAME`: 应用名称，可以改成您的组织或公司名称

`RUN_USER`: 运行应用的用户名称，我们建议您使用 `git`，但如果您在个人计算机上运行 Gogs，请修改为您的系统用户名称。如果没有正确设置这个值，很可能导致您的应用崩溃

`RUN_MODE`: 鉴于性能和其它考虑，建议在部署环境下修改为 `prod` 模式。在您完成安装操作时，该值也会被设置为 `prod`

Repository (`repository`)

`ROOT`: 用户仓库存储根目录，必须为绝对路径，默认为 `~/<user name>/gogs-repositories`

`SCRIPT_TYPE`: 系统脚本类型，一般情况下均为 `bash`，但有些用户反应只能使用 `sh`

`ANSI_CHARSET`: 当遇到无法识别的字符集时使用的默认字符集

`FORCE_PRIVATE`: 强制要求所有新建的仓库都是私有的

`MAX_CREATION_LIMIT`: 全局默认的用户可创建仓库上限，`-1` 表示无限制

`PULL_REQUEST_QUEUE_LENGTH` **!**: 测试合并请求 (Pull Request) 的任务队列长度，该值越大越好

UI (ui)

EXPLORE_PAGING_NUM: 探索页面每页显示仓库的数量

ISSUE_PAGING_NUM: 每页显示工单 (Issue) 的数量 (应用到所有以列表形式显示工单的页面)

FEED_MAX_COMMIT_NUM: 一条最新活动中显示代码提交 (Commit) 的最大数量

UI - Admin (ui.admin)

- USER_PAGING_NUM: 用户管理页面每页显示记录条数
- REPO_PAGING_NUM: 仓库管理页面每页显示记录条数
- NOTICE_PAGING_NUM: 系统提示管理页面每页显示记录条数
- ORG_PAGING_NUM: 组织管理页面每页显示记录条数

Markdown (markdown)

- ENABLE_HARD_LINE_BREAK: 指示是否启动硬性换行扩展

Server (server)

- PROTOCOL: http 或 https
- DOMAIN: 服务器域名
- ROOT_URL: 公开的完整 URL 路径
- HTTP_ADDR: 应用 HTTP 监听地址
- HTTP_PORT: 应用 HTTP 监听端口号
- DISABLE_SSH: 当 SSH 功能不可用时可以禁用
- START_SSH_SERVER: 启用该选项来启动内置 SSH 服务器
- SSH_PORT: SSH 端口号, 如果不为 22 的话可以在此修改
- OFFLINE_MODE: 激活该选项来禁止从 CDN 获取静态资源, 同时 Gravatar 服务也将被自动禁用
- DISABLE_ROUTER_LOG: 激活该选项来禁止打印路由日志
- CERT_FILE: HTTPS 授权文件路径
- KEY_FILE: HTTPS 的密钥文件路径
- STATIC_ROOT_PATH: 模板文件和静态文件的上级目录, 默认为应用二进制所在的位置
- ENABLE_GZIP: 激活该选项来启用应用级别 GZIP 支持
- LANDING_PAGE: 未登录用户的默认首页, 可以是 home 或 explore (探索页)

Database (database)

- DB_TYPE: 数据库类型, 可以是 mysql、postgres 或 sqlite3
- HOST: 数据库主机地址与端口
- NAME: 数据库名称
- USER: 数据库用户名
- PASSWD: 数据库用户密码
- SSL_MODE: 仅限 PostgreSQL 使用
- PATH: 仅限 SQLite3 使用, 数据库文件路径

Security (security)

- INSTALL_LOCK: 用于指示是否允许访问安装页面 (该页面可以设置管理员帐号, 因此该选项非常重要)
- SECRET_KEY: 全局的加密密钥, **务必修改该值以确保您的服务器安全** (会在每次安装时自动生成随机字符串)
- LOGIN_REMEMBER_DAYS: 记住登录的天数
- COOKIE_USERNAME: 记录用户名的 Cookie 名称
- COOKIE_REMEMBER_NAME: 记录用户自动登录信息的 Cookie 名称
- REVERSE_PROXY_AUTHENTICATION_USER: 反向代理认证用户的 Header 字段名

Service (service)

- ACTIVE_CODE_LIVE_MINUTES: 激活码的有效期, 单位为分钟
- RESET_PASSWD_CODE_LIVE_MINUTES: 重置密码的有效期, 单位为分钟
- REGISTER_EMAIL_CONFIRM: 激活该选项来要求注册用户必须验证邮箱, 要求已启用 Mailer
- DISABLE_REGISTRATION: 激活该选项来禁止用户注册功能, 只能由管理员创建帐号
- SHOW_REGISTRATION_BUTTON: 用于指示是否显示注册按钮
- REQUIRE_SIGNIN_VIEW: 激活该选项来要求用户必须登录才能浏览任何页面
- ENABLE_CACHE_AVATAR: 激活该选项来缓存 Gravatar 的头像
- ENABLE_NOTIFY_MAIL: 激活该选项来发送通知邮件给关注者, 例如创建 issue 时, 要求已启用 Mailer
- ENABLE_REVERSE_PROXY_AUTHENTICATION: 激活该选项来开启反向代理用户认证, 请从 <https://github.com/gogits/gogs/issues/165> 了解更多信息
- ENABLE_REVERSE_PROXY_AUTO_REGISTRATION: 激活该选项来开启反向代理用户认证的自动注册功能
- DISABLE_MINIMUM_KEY_SIZE_CHECK: 激活该选项来禁止检查响应类型的密钥最小长度
- ENABLE_CAPTCHA: 激活该选项以在用户注册时要求输入验证码

Webhook (webhook)

- `QUEUE_LENGTH` **!** : 发送通知的队列长度
- `DELIVER_TIMEOUT`: 发送通知的超时时间, 以秒为单位
- `SKIP_TLS_VERIFY`: 指示是否允许向具有非信任证书的地址发送通知
- `PAGING_NUM`: Web 钩子历史页面每页显示记录条数

Mailer (`mailer`)

- `ENABLED`: 启用该选项以激活邮件服务
- `DISABLE_HELO`: 禁用 HELO 操作
- `HELO_HOSTNAME`: HELO 操作的自定义主机名
- `HOST`: SMTP 主机地址与端口
- `FROM`: 邮箱的来自地址, 遵循 RFC 5322 规范, 可以是一个单纯的邮箱地址或者“名字” `<email@example.com>` 的形式
- `USER`: 邮箱用户名
- `PASSWD`: 邮箱密码
- `SKIP_VERIFY`: 不验证自签发证书的有效性

Cache (`cache`)

- `ADAPTER`: 缓存引擎适配器, 可以为 `memory`、`redis` 或 `memcache`。如果您使用 `redis` 或 `memcache`, 请确保使用 `-tags` 选项重新构建所有依赖, 例如:
`go build -tags='redis'`
- `INTERVAL`: 仅限内存缓存使用, GC 周期, 单位为秒
- `HOST`: 仅限 `redis` 和 `memcache` 使用, 主机地址和端口号
 - `Redis` :
`network=tcp, addr=127.0.0.1:6379, password=macaron, db=0, pool_size=100, idle_timeout=180`
 - `Memache`: `127.0.0.1:9090;127.0.0.1:9091`

Session (`session`)

- `PROVIDER`: Session 引擎提供者, 可以是 `memory`、`file`、`redis` 或 `mysql`
- `PROVIDER_CONFIG`: 如果提供者为 `file`, 则为文件根目录; 如果为其它提供者, 则为主机地址和端口号
- `COOKIE_SECURE`: 激活该选项以要求所有 `session` 操作均通过 HTTPS
- `GC_INTERVAL_TIME`: GC 周期, 单位为秒

Picture (`picture`)

- `GRAVATAR_SOURCE`: 可以是 `gravatar`、`duoshuo` 或任何 URL, 例如:
`http://cn.gravatar.com/avatar/`
- `DISABLE_GRAVATAR`: 激活该选项来仅使用本地头像

Attachment (`attachment`)

- `ENABLED`: 启用该选项以允许用户上传附件
- `PATH`: 存放附件的路径
- `ALLOWED_TYPES`: 允许上传的 MIME 类型, 例如 `image/jpeg|image/png`, 使用 `/**` 允许所有类型的文件
- `MAX_SIZE`: 最大允许上传的附件体积, 单位为 MB, 例如 4
- `MAX_FILES`: 最大允许一次性上传的附件个数, 例如 5

Log (`log`)

- `ROOT_PATH`: 日志文件的根目录
- `MODE`: 日志记录模式, 默认为 `console`。如果想要开启多模式, 请使用逗号分割
- `LEVEL`: 基本日志级别, 默认为 `Trace`

Cron (`cron`)

- `ENABLED`: 激活该选项以允许周期性运行 Cron 任务
- `RUN_AT_START`: 激活该选项以允许在启动时执行 Cron 任务

Cron - Update Mirrors (`cron.update_mirrors`)

- `SCHEDULE`: 定时更新仓库镜像的 Cron 语法, 例如: `@every 1h`

Cron - Repository Health Check (`cron.repo_health_check`)

- `SCHEDULE`: 定时进行仓库健康检查的 Cron 语法, 例如: `@every 24h`
- `TIMEOUT`: 仓库健康检查超时的定义语法, 例如: `60s`
- `ARGS`: `git fsck` 命令的参数, 例如: `--unreachable --tags`

Cron - Repository Statistics Check (`cron.check_repo_stats`)

- `RUN_AT_START`: 激活该选项以在启动时执行仓库统计检查
- `SCHEDULE`: 定时进行仓库统计检查的 Cron 语法, 例如: `@every 24h`

Git (`git`)

- `MAX_GIT_DIFF_LINES`: 对比页面显示的最大行数
- `GC_ARGS`: `git gc` 命令的参数, 例如: `--aggressive --auto`

Other (`other`)

- `SHOW_FOOTER_BRANDING`: 激活该选项以在页脚显示 Gogs 推广信息
- `SHOW_FOOTER_VERSION`: 激活该选项以在页脚显示 Gogs 版本信息

常见问题

部署应用

如果我已经其它应用使用端口 3000 了怎么办?

您可以在您第一次启动 Gogs 的时候通过以下方式修改默认的监听端口:

```
./gogs web -port 3001
```

该方法同样会在您填写安装页面时生效, 因此请选择一个您希望以后一直给 Gogs 使用的端口号。

如何使用 NGINX 的反向代理?

在 `nginx.conf` 文件中, 将下面的 `server` 部分增加至 `http` 分区内并重载配置:

```
server {  
  
    listen 80;  
  
    server_name git.crystalnetwork.us;  
  
    location / {  
  
        proxy_pass http://localhost:3000;
```

```
    }  
}
```

配置子路径

如果您想要通过域名的子路径来访问 Gogs 实例，可以将 NGINX 的配置修改为以下形式（特别注意后缀 /）：

```
server {  
    listen 80;  
    server_name git.crystalnetwork.us;  
  
    location /gogs/ {  
        proxy_pass http://localhost:3000/;  
    }  
}
```

然后在配置文件中设置 [server] `ROOT_URL = http://git.crystalnetwork.us/gogs/`。

为什么上传大文件时总是发生错误？

想要了解如何让 NGINX 支持大文件上传的讨论，可以查看 [此贴](#)。一般 NGINX 会返回 413 错误，在配置文件中加入以下内容可以解决该问题：

```
client_max_body_size 50m;
```

如果我用的是 Apache 2 怎么配置子路径？

可以尝试使用下面的配置模板：

```
<VirtualHost *:443>  
  
    ...  
  
    <Proxy *>  
  
        Order allow,deny  
  
        Allow from all  
  
    </Proxy>  
  
    ProxyPass /git http://127.0.0.1:3000/
```

```
ProxyPassReverse /git http://127.0.0.1:3000/
```

```
</VirtualHost>
```

有没有用 `lighttpd` 配置子路径的例子?

可以尝试使用下面的配置模板:

```
server.modules += ( "mod_proxy_backend_http" )$HTTP["url"] =~ "^/gogs"
{
    proxy-core.protocol = "http"

    proxy-core.backends = ( "localhost:3000" )

    proxy-core.rewrite-request = (
        "_uri" => ( "^/gogs/?(.*)" => "/$1" ),
        "Host" => ( ".*" => "localhost:3000" ),
    )
}
```

如何使用 HTTPS?

在 `custom/conf/app.ini` 文件中修改下列配置选项 (以下仅为示例):

```
[server]PROTOCOL = httpsROOT_URL = https://try.gogs.io/CERT_FILE =
custom/https/cert.pemKEY_FILE = custom/https/key.pem
```

如果您想要使用自签名的 HTTPS, 则可以使用下列命令来生成所需文件 (需要使用构建标签 `cert` 或直接从官方下载二进制):

```
$ ./gogs cert -ca=true -duration=8760h0m0s -host=myhost.example.com
```

如何使用离线模式?

如果您需要将 Gogs 运行于内网环境下, 只需将 `custom/conf/app.ini` 文件中的配置选项 `server -> OFFLINE_MODE` 修改为 `true` 即可。

如何使用自定义 robots.txt?

在 `custom` 目录下创建 `robots.txt` 文件即可。

如何以守护进程形式运行？

Gogs 拥有一些由第三方提供的脚本来支持以守护进程形式运行：

```
init.d/centos  
init.d/debian
```

下小节中的 Systemd 服务

Systemd 服务

在 GitHub 上的 Gogs 仓库有一个 `systemd` 服务模版文件，您需要做出一定的修改才能够使用它：

1. 更新 `User`、`Group`、`WorkingDirectory`、`ExecStart` 和 `Environment` 为相对应的值。其中 `WorkingDirectory` 为您的 Gogs 实际安装路径根目录。
2. [可选] 如果您 Gogs 安装示例使用 MySQL/MariaDB、PostgreSQL、Redis 或 memcached，请去掉相应 `After` 属性的注释。

当您完成修改后，请将文件保存至 `/etc/systemd/system/gogs.service`，然后通过 `sudo systemctl enable gogs` 命令激活，最后执行 `sudo systemctl start gogs`。

您可以通过 `sudo systemctl status gogs -l` 或 `sudo journalctl -b -u gogs` 命令检查 Gogs 的运行状态。

管理权限

如果成为管理员？

1. 当用户注册时 `ID = 1` 则会自动成为管理员，无需邮箱验证。
2. 默认管理员登录 `Admin` -> `Users` 面板并设置其它人员为管理员。
3. 通过安装页面注册的用户会自动成为管理员。

仓库管理

如何给予用户 Git 钩子权限？

由于这是一个 **可能损坏您系统的高级权限**，您必须在用户管理面板 (`/admin/users/:userid`) 为您完全信任的用户单独开启。

其它

如何获取 Gogs 当前版本？

纯文本形式的 Gogs 版本存储在文件 `templates/.VERSION` 中。

`gogs serv` 命令是做什么用的？

用户完全没有理由和必要去手动执行该命令，当新的 SSH 推送发生时，它会被 Git 的 Update Hook 自动调用。

说明

本文档来自 gogs.io 官网 新变更内容请关注官网