

Gitea中文文档

书栈(BookStack.CN)

目 录

[致谢](#)

[安装](#)

[从Docker安装](#)

[Linux service](#)

[从二进制安装](#)

[选择包安装](#)

[Windows服务](#)

[从源代码安装](#)

[升级](#)

[从 Gogs 升级](#)

[特性](#)

[Comparison](#)

[认证](#)

[本地化](#)

[Webhooks](#)

[Usage](#)

[备份与恢复](#)

[Issue and Pull Request templates](#)

[Reverse Proxies](#)

[进阶](#)

[Customizing Gitea](#)

[加入 Gitea 开源](#)

[Specific variables](#)

[配置说明](#)

[Make](#)

[API Usage](#)

[帮助](#)

[Troubleshooting](#)

[需要帮助](#)

致谢

当前文档 《Gitea中文文档》 由 进击的皇虫 使用 书栈(BookStack.CN) 进行构建，生成于 2018-07-19。

书栈(BookStack.CN) 仅提供文档编写、整理、归类等功能，以及对文档内容的生成和导出工具。

文档内容由网友们编写和整理，书栈(BookStack.CN) 难以确认文档内容知识点是否错漏。如果您在阅读文档获取知识的时候，发现文档内容有不恰当的地方，请向我们反馈，让我们共同携手，将知识准确、高效且有效地传递给每一个人。

同时，如果您在日常工作、生活和学习中遇到有价值有营养的知识文档，欢迎分享到 书栈(BookStack.CN) ，为知识的传承献上您的一份力量！

如果当前文档生成时间太久，请到 书栈(BookStack.CN) 获取最新的文档，以跟上知识更新换代的步伐。

文档地址：<http://www.bookstack.cn/books/gitea-doc-zh>

书栈官网：<http://www.bookstack.cn>

书栈开源：<https://github.com/TruthHun>

分享，让知识传承更久远！ 感谢知识的创造者，感谢知识的分享者，也感谢每一位阅读到此处的读者，因为我们都将成为知识的传承者。

安装

- [从Docker安装](#)
- [Linux service](#)
- [从二进制安装](#)
- [选择包安装](#)
- [Windows服务](#)
- [从源代码安装](#)

从Docker安装

从Docker安装

阅读本章之前我们已经假设您对docker已经有了解并能够正常使用docker。

我们在 Docker Hub 的 Gitea 组织中提供了自动更新的 Docker 镜像，它会保持最新的稳定版。你也可以用其它 Docker 服务来更新。首先你需要pull镜像：

```
1. docker pull gitea/gitea:latest
```

如果要将git和其它数据持久化，你需要创建一个目录来作为数据存储的地方：

```
1. sudo mkdir -p /var/lib/gitea
```

然后就可以运行 docker 容器了，这很简单。当然你需要定义端口和数据目录：

```
1. docker run -d --name=gitea -p 10022:22 -p 10080:3000 -v /var/lib/gitea:/data gitea/gitea:latest
```

然后 容器已经运行成功，在浏览器中访问 <http://hostname:10080> 就可以看到界面了。你可以尝试在上面创建项目，clone操作 `git clone ssh://git@hostname:10022/username/repo.git` 。

注意：目前端口改为非3000时，需要修改配置文件 `LOCAL_ROOT_URL = http://localhost:3000/` 。

需要帮助？

如果从本页中没有找到你需要的内容，请访问 [帮助页面](#)

原文：<https://docs.gitea.io/zh-cn/install-with-docker/>

Linux service

Run as service in Ubuntu 16.04 LTS

Using systemd

Run the below command in a terminal:

```
1. sudo vim /etc/systemd/system/gitea.service
```

Copy the sample [gitea.service](#).

Uncomment any service that needs to be enabled on this host, such as MySQL.

Change the user, home directory, and other required startup values. Change thePORT or remove the -p flag if default port is used.

Enable and start gitea at boot:

```
1. sudo systemctl enable gitea
2. sudo systemctl start gitea
```

Using supervisor

Install supervisor by running below command in terminal:

```
1. sudo apt install supervisor
```

Create a log dir for the supervisor logs:

```
1. # assuming gitea is installed in /home/git/gitea/
2. mkdir /home/git/gitea/log/supervisor
```

Open supervisor config file in a file editor:

```
1. sudo vim /etc/supervisor/supervisord.conf
```

Append the configuration from the sample[supervisord config](#).

Change the user(git) and home(/home/git) settings to match the deploymentenvironment. Change the PORT or remove the -p flag if default port is used.

Lastly enable and start supervisor at boot:

```
1. sudo systemctl enable supervisor
```

```
2. sudo systemctl start supervisor
```

原文: <https://docs.gitea.io/zh-cn/linux-service/>

从二进制安装

从二进制安装

所有下载均包括 SQLite, MySQL 和 PostgreSQL 的支持, 同时所有资源均已嵌入到可执行程序中, 这一点和老版本有所不同。 基于二进制的安装非常简单, 只要从 [下载页面](#) 选择对应平台, 拷贝下载URL, 执行以下命令即可 (以Linux为例) :

```
1. wget -O gitea https://dl.gitea.io/gitea/1.3.2/gitea-1.3.2-linux-amd64
2. chmod +x gitea
```

测试

在执行了以上步骤之后, 你将会获得 `gitea` 的二进制文件, 在你复制到部署的机器之前可以先测试一下。在命令行执行完后, 你可以 `Ctrl + C` 关掉程序。

```
1. ./gitea web
```

需要帮助?

如果从本页中没有找到你需要的内容, 请访问 [帮助页面](#)

原文: <https://docs.gitea.io/zh-cn/install-from-binary/>

选择包安装

使用包安装

Linux

目前还没有对应的Linux安装包发布，如果我们发布了，我们将更新本页面。当前你可以查看 [从二进制安装](#)。

Windows

目前还没有对应的Windows安装包发布，如果我们发布了，我们将更新本页面。我们计划使用 MSI 安装器或者 [Chocolatey](#)来制作安装包。当前你可以查看 [从二进制安装](#)。

macOS

macOS 平台下当前我们仅支持通过 brew 来安装。如果您没有安装 [Homebrew](#)，你治可以查看 [从二进制安装](#)。在你安装了 brew 之后， 你可以执行以下命令：

```
1. brew tap go-gitea/gitea
2. brew install gitea
```

需要帮助？

如果从本页中没有找到你需要的内容，请访问 [帮助页面](#)

原文： <https://docs.gitea.io/zh-cn/install-from-package/>

Windows服务

注册为Windows服务

要注册为Windows服务，首先以Administrator身份运行 `cmd` ，然后执行以下命令：

```
1. sc create gitea start= auto binPath= "C:\gitea\gitea.exe" web --config "C:\gitea\custom\conf\app.ini"
```

别忘了将 `C:\gitea` 替换成你的 Gitea 安装目录。

之后在控制面板打开 “Windows Services”，搜索 “gitea”，右键选择 “Run”。在浏览器打开 `http://localhost:3000` 就可以访问了。（如果你修改了端口，请访问对应的端口，3000是默认端口）。

从Windows服务中删除

以Administrator身份运行 `cmd` ，然后执行以下命令：

```
1. sc delete gitea
```

原文: <https://docs.gitea.io/zh-cn/windows-service/>

从源代码安装

从源代码安装

首先你需要安装Golang，关于Golang的安装，参见官方文档 [install instructions](#)。

下载

你需要获取Gitea的源码，最方便的方式是使用 `go` 命令。执行以下命令：

```
1. go get -d -u code.gitea.io/gitea
2. cd $GOPATH/src/code.gitea.io/gitea
```

然后你可以选择编译和安装的版本，当前你有多个选择。如果你想编译 `master` 版本，你可以直接跳到 [编译](#) 部分，这是我们的开发分支，虽然也很稳定但不建议您在正式产品中使用。

如果你想编译最新稳定分支，你可以执行以下命令签出源码：

```
1. git branch -a
2. git checkout v1.0
```

最后，你也可以直接使用标签版本如 `v1.0.0`。你可以执行以下命令列出可用的版本并选择某个版本签出：

```
1. git tag -l
2. git checkout v1.0.0
```

编译

我们已经将所有的依赖项拷贝到本工程，我们提供了一些 [编译选项](#) 来让编译更简单。你可以按照你的需求来设置编译开关，可用编译选项如下：

- `bindata`：这个编译选项将会把运行Gitea所需的所有外部资源都打包到可执行文件中，这样部署将非常简单因为除了可执行程序将不再需要任何其他文件。
- `sqlite`：这个编译选项将启用SQLite3数据库的支持，建议只在少数人使用时使用这个模式。
- `tidb`：这个编译选项启用tidb嵌入式数据库的支持，他跟SQLite类似但是是用纯Go编写的。
- `pam`：这个编译选项将会启用 PAM (Linux Pluggable Authentication Modules) 认证，如果你使用这一认证模式的话需要开启这个选项。
我们支持两种方式进行编译，`Make` 工具和 `Go` 工具。不过我们推荐使用 `Make`工具，因为他将会给出更多的编译选项。

Note: We recommend the Go version 1.6 or higher because we are using vendoring and we don't set the required env variable for 1.5 anywhere.

- Make 工具

这个编译方式要求你先安装Make工具，关于Make工具的安装你可以参考Make相关资料。同样如果要使用bindata选项，你可能需要先执行make generate：

```
1. TAGS="bindata" make generate build
```

- Go 工具

使用 Go 工具编译需要你至少安装了Go 1.5以上版本并且将 govendor 的支持打开。执行命令如下：

```
1. go build
```

测试

在执行了以上步骤之后，你将会获得 `gitea` 的二进制文件，在你复制到部署的机器之前可以先测试一下。在命令行执行完后，你可以 `Ctrl + C` 关掉程序。

```
1. ./gitea web
```

需要帮助？

如果从本页中没有找到你需要的内容，请访问 [帮助页面](#)

原文：<https://docs.gitea.io/zh-cn/install-from-source/>

升级

- [从 Gogs 升级](#)

从 Gogs 升级

从 Gogs 升级

如果你正在运行Gogs 0.9.146以下版本，你可以平滑的升级到Gitea。该升级需要如下的步骤：

- 停止 Gogs 的运行
- 拷贝 Gogs 的配置文件 custom/conf/app.ini 到 Gitea 的相应位置。
- 拷贝 Gitea 的 options/ 到 Home 目录下。
- 如果你还有更多的自定义内容，比如templates和localization文件，你需要手工合并你的修改到 Gitea 的 Options 下对应目录。
- 拷贝 Gogs 的数据目录 data/ 到 Gitea 相应位置。这个目录包含附件和头像文件。
- 运行 Gitea
- 登陆 Gitea 并进入 管理面板，运行 重新生成 '.ssh/authorized_keys' 文件（警告：不是 Gitea 的密钥也会被删除）和 重新生成所有仓库的 Update 钩子（用于自定义配置文件被修改）。

原文： <https://docs.gitea.io/zh-cn/upgrade-from-gogs/>

特性

- [Comparison](#)
- [认证](#)
- [本地化](#)
- [Webhooks](#)

Comparison

Gitea compared to other Git hosting options

To help decide if Gitea is suited for your needs here is how it compares to other Git self hosted options.

Be warned that we don't regularly check for feature changes in other products so this list can be outdated. If you find anything that needs to be updated in table below please report [issue on Github](#).

Symbols used in table:

- ✓ - supported
- / - supported with limited functionality
- ✗ - unsupported

General Features

Feature	Gitea	Gogs	GitHub EE	GitLab CE	GitLab EE	BitBucket	RhodeCode CE
Open source and free	✓	✓	✗	✓	✗	✗	✓
Low resource usage (RAM/CPU)	✓	✓	✗	✗	✗	✗	✗
Multiple database support	✓	✓	✗	/	/	✓	✓
Multiple OS support	✓	✓	✗	✗	✗	✗	✓
Easy upgrade process	✓	✓	✗	✓	✓	✗	✓
Markdown support	✓	✓	✓	✓	✓	✓	✓
Static Git-powered pages	✗	✗	✓	✓	✓	✗	✗
Integrated Git-powered wiki	✓	✓	✓	✓	✓	✓	✗
Deploy Tokens	✓	✓	✓	✓	✓	✓	✓
Repository Tokens with write rights	✓	✗	✓	✓	✓	✗	✓
Built-in Container Registry	✗	✗	✗	✓	✓	✗	✗

External git mirroring	✓	✓	✗	✗	✓	✓	✓
FIDO U2F (2FA)	✓	✗	✓	✓	✓	✓	✗
Built-in CI/CD	✗	✗	✗	✓	✓	✗	✗
Subgroups: groups within groups	✗	✗	✗	✓	✓	✗	✓

Code management

Feature	Gitea	Gogs	GitHub EE	GitLab CE	GitLab EE	BitBucket	RhodeCode CE
Repository topics	✓	✗	✓	✓	✓	✗	✗
Repository code search	✓	✗	✓	✓	✓	✓	✓
Global code search	✓	✗	✓	✓	✓	✓	✓
Git LFS 2.0	✓	✗	✓	✓	✓	/	✓
Group Milestones	✗	✗	✗	✓	✓	✗	✗
Granular user roles (Code, Issues, Wiki etc)	✓	✗	✗	✓	✓	✗	✗
Verified Committer	✗	✗	?	✓	✓	✓	✗
GPG Signed Commits	✓	✗	✓	✓	✓	✓	✓
Reject unsigned commits	✗	✗	✓	✓	✓	✗	✓
Repository Activity page	✓	✗	✓	✓	✓	✓	✓
Branch manager	✓	✗	✓	✓	✓	✓	✓
Create new branches	✓	✗	✓	✓	✓	✗	✗
Web code editor	✓	✓	✓	✓	✓	✓	✓
Commit graph	✓	✗	✓	✓	✓	✓	✓

Issue Tracker

Feature	Gitea	Gogs	GitHub EE	GitLab CE	GitLab EE	BitBucket	RhodeCode CE
Issue tracker	✓	✓	✓	✓	✓	✓	✗
Issue templates	✓	✓	✓	✓	✓	✗	✗
Labels	✓	✓	✓	✓	✓	✗	✗

Time tracking	✓	✗	✓	✓	✓	✗	✗
Multiple assignees for issues	✓	✗	✓	✓	✓	✗	✗
Related issues	✗	✗	/	✗	✓	✗	✗
Confidential issues	✗	✗	✗	✓	✓	✗	✗
Comment reactions	✓	✗	✓	✓	✓	✗	✗
Lock Discussion	✗	✗	✓	✓	✓	✗	✗
Batch issue handling	✓	✗	✓	✓	✓	✗	✗
Issue Boards	✗	✗	✗	✓	✓	✗	✗
Create new branches from issues	✗	✗	✗	✓	✓	✗	✗
Issue search	✓	✗	✓	✓	✓	✓	✗
Global issue search	✗	✗	✓	✓	✓	✓	✗

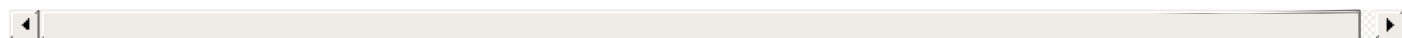
Pull/Merge requests

Feature	Gitea	Gogs	GitHub EE	GitLab CE	GitLab EE	BitBucket	RhodeCode CE
Pull/Merge requests	✓	✓	✓	✓	✓	✓	✓
Squash merging	✓	✗	✓	✗	✓	✓	✓
Rebase merging	✓	✓	✓	✗	/	✗	✓
Pull/Merge request inline comments	✗	✗	✓	✓	✓	✓	✓
Pull/Merge request approval	✗	✗	/	✓	✓	✓	✓
Merge conflict resolution	✗	✗	✓	✓	✓	✓	✗
Restrict push and merge access to certain users	✓	✗	✓	/	✓	✓	✓
Revert specific commits or a merge request	✗	✗	✓	✓	✓	✓	✗
Pull/Merge requests templates	✓	✓	✓	✓	✓	✗	✗
Cherry-picking changes	✗	✗	✗	✓	✓	✗	✗

3rd-party integrations

Feature	Gitea	Gogs	GitHub EE	GitLab CE	GitLab EE	BitBucket	RhodeCode CE
Webhook support	✓	✓	✓	✓	✓	✓	✓
Custom Git Hooks	✓	✓	✓	✓	✓	✓	✓
AD / LDAP integration	✓	✓	✓	✓	✓	✓	✓
Multiple LDAP / AD server support	✓	✓	✗	✗	✓	✓	✓
LDAP user synchronization	✓	✗	✓	✓	✓	✓	✓
OpenId Connect support	✓	✗	✓	✓	✓	?	✗
OAuth 2.0 integration (external authorization)	✓	✗	/	✓	✓	?	✓
Act as OAuth 2.0 provider	✗	✗	✓	✓	✓	✓	✗
Two factor authentication (2FA)	✓	✓	✓	✓	✓	✓	✗
Mattermost/Slack integration	✓	✓	/	✓	✓	/	✓
Discord integration	✓	✓	✓	✗	✗	✗	✗
External CI/CD status display	✓	✗	✓	✓	✓	✓	✓

原文: <https://docs.gitea.io/zh-cn/comparison/>



认证

认证

TBD

原文: <https://docs.gitea.io/zh-cn/authentication/>

本地化

本地化

TBD

原文: <https://docs.gitea.io/zh-cn/localization/>

Webhooks

Webhooks

TBD

原文: <https://docs.gitea.io/zh-cn/webhooks/>

Usage

- [备份与恢复](#)
- [Issue and Pull Request templates](#)
- [Reverse Proxies](#)

备份与恢复

备份与恢复

Gitea 已经实现了 `dump` 命令可以用来备份所有需要的文件到一个zip压缩文件。该压缩文件可以被用来进行数据恢复。

备份命令（dump）

先转到git用户的权限：`su git`。再Gitea目录运行 `./gitea dump`。一般会显示类似如下的输出：

```
1. 2016/12/27 22:32:09 Creating tmp work dir: /tmp/gitea-dump-417443001
2. 2016/12/27 22:32:09 Dumping local repositories.../home/git/gitea-repositories
3. 2016/12/27 22:32:22 Dumping database...
4. 2016/12/27 22:32:22 Packing dump files...
5. 2016/12/27 22:32:34 Removing tmp work dir: /tmp/gitea-dump-417443001
6. 2016/12/27 22:32:34 Finish dumping in file gitea-dump-1482906742.zip
```

最后生成的 `gitea-dump-1482906742.zip` 文件将会包含如下内容：

- custom - 所有保存在 custom/ 目录下的配置和自定义的文件。
- data - 数据目录下的所有内容不包含使用文件session的文件。该目录包含 attachments, avatars, lfs, indexers, 如果使用sqlite 还会包含 sqlite 数据库文件。
- gitea-db.sql - 数据库dump出来的 SQL。
- gitea-repo.zip - Git仓库压缩文件。
- log/ - Logs文件，如果用作迁移不是必须的。

中间备份文件将会在临时目录进行创建，如果您要重新指定临时目录，可以用 `-tempdir` 参数，或者用 `TMPDIR` 环境变量。

Restore Command (restore)

当前还没有恢复命令，恢复需要人工进行。主要是把文件和数据库进行恢复。

例如：

```
1. apt-get install gitea
2. unzip gitea-dump-1482906742.zip
3. cd gitea-dump-1482906742
4. mv custom/conf/app.ini /etc/gitea/conf/app.ini
5. unzip gitea-repo.zip
6. mv gitea-repo/* /var/lib/gitea/repositories/
7. chown -R gitea:gitea /etc/gitea/conf/app.ini /var/lib/gitea/repositories/
8. mysql -u$USER -p$PASS $DATABASE <gitea-db.sql
9. # or sqlite3 $DATABASE_PATH <gitea-db.sql
10. service gitea restart
```


原文: <https://docs.gitea.io/zh-cn/backup-and-restore/>

Issue and Pull Request templates

Issue and Pull Request Templates

For some projects there are a standard list of questions that users need to be asked for creating an issue, or adding a pull request. Gitea supports adding templates to the main branch of the repository so that they can autopopulate the form when users are creating issues, and pull requests. This will cut down on the initial back and forth of getting some clarifying details.

Possible file names for issue templates:

- `ISSUE_TEMPLATE.md`
- `issue_template.md`
- `.gitea/ISSUE_TEMPLATE.md`
- `.gitea/issue_template.md`
- `.github/ISSUE_TEMPLATE.md`
- `.github/issue_template.md`

Possible file names for PR templates:

- `PULL_REQUEST_TEMPLATE.md`
- `pull_request_template.md`
- `.gitea/PULL_REQUEST_TEMPLATE.md`
- `.gitea/pull_request_template.md`
- `.github/PULL_REQUEST_TEMPLATE.md`
- `.github/pull_request_template.md`

原文: <https://docs.gitea.io/zh-cn/issue-pull-request-templates/>

Reverse Proxies

Using Nginx as a reverse proxy

If you want Nginx to serve your Gitea instance you can the following `server` section inside the `http` section of `nginx.conf` :

```
1. server {
2.     listen 80;
3.     server_name git.example.com;
4.
5.     location / {
6.         proxy_pass http://localhost:3000;
7.     }
8. }
```

Using Nginx with a Sub-path as a reverse proxy

In case you already have a site, and you want Gitea to share the domain name, you can setup Nginx to serve Gitea under a sub-path by adding the following `server` section inside the `http` section of `nginx.conf` :

```
1. server {
2.     listen 80;
3.     server_name git.example.com;
4.
5.     location /git/ { # Note: Trailing slash
6.         proxy_pass http://localhost:3000/; # Note: Trailing slash
7.     }
8. }
```

Then set `[server] ROOT_URL = http://git.example.com/git/` in your configuration.

Using Apache HTTPD as a reverse proxy

If you want Apache HTTPD to serve your Gitea instance you can add the following to you Apache HTTPD configuration (usually located at `/etc/apache2/httpd.conf` in Ubuntu):

```
1. <VirtualHost *:80>
2.     ...
3.     ProxyPreserveHost On
4.     ProxyRequests off
5.     ProxyPass / http://localhost:3000/
6.     ProxyPassReverse / http://localhost:3000/
7. </VirtualHost>
```

Note: The following Apache HTTPD mods must be enabled: `proxy` , `proxy_http`

Using Apache HTTPD with a Sub-path as a reverse proxy

In case you already have a site, and you want Gitea to share the domain name, you can setup Apache HTTPD to serve Gitea under a sub-path by adding the following to you Apache HTTPD configuration (usually located at `/etc/apache2/httpd.conf` in Ubuntu):

```

1. <VirtualHost *:80>
2.     ...
3.     <Proxy *>
4.         Order allow,deny
5.         Allow from all
6.     </Proxy>
7.
8.     ProxyPass /git http://localhost:3000 # Note: no trailing slash after either /git or port
9.     ProxyPassReverse /git http://localhost:3000 # Note: no trailing slash after either /git or port
10. </VirtualHost>

```

Then set `[server] ROOT_URL = http://git.example.com/git/` in your configuration.

Note: The following Apache HTTPD mods must be enabled: `proxy` , `proxy_http`

Using Caddy with a Sub-path as a reverse proxy

If you want Caddy to serve your Gitea instance you can add the following server block to your Caddyfile:

```

1. git.example.com {
2.     proxy / http://localhost:3000
3. }

```

Using Caddy with a Sub-path as a reverse proxy

In case you already have a site, and you want Gitea to share the domain name, you can setup Caddy to serve Gitea under a sub-path by adding the following to you server block in your Caddyfile:

```

1. git.example.com {
2.     proxy /git/ http://localhost:3000 # Note: Trailing Slash after /git/
3. }

```

Then set `[server] ROOT_URL = http://git.example.com/git/` in your configuration.

原文: <https://docs.gitea.io/zh-cn/reverse-proxies/>

进阶

- [Customizing Gitea](#)
- [加入 Gitea 开源](#)
- [Specific variables](#)
- [配置说明](#)
- [Make](#)
- [API Usage](#)

Customizing Gitea

Customizing Gitea

Customizing Gitea is typically done using the `custom` folder. This is the central place to override configuration settings, templates, etc.

If Gitea is deployed from binary, all default paths will be relative to the gitea binary. If installed from a distribution, these paths will likely be modified to the Linux Filesystem Standard. Gitea will create required folders, including `custom/`. Application settings are configured in `custom/conf/app.ini`. Distributions may provide a symlink for `custom` using `/etc/gitea/`.

- [Quick Cheat Sheet](#)

- [Complete List](#)

If the `custom` folder can't be found next to the binary, check the `GITEA_CUSTOM` environment variable; this can be used to override the default path to something else. `GITEA_CUSTOM` might, for example, be set by an init script.

- [List of Environment Variables](#)

Note: Gitea must perform a full restart to see configuration changes.

Customizing /robots.txt

To make Gitea serve a custom `/robots.txt` (default: empty 404), create a file called `robots.txt` in the `custom` folder with [expected contents](#).

Serving custom public files

To make Gitea serve custom public files (like pages and images), use the folder `custom/public/` as the webroot. Symbolic links will be followed.

For example, a file `image.png` stored in `custom/public/`, can be accessed with the url `http://gitea.domain.tld/image.png`.

Changing the default avatar

Place the png image at the following path: `custom/public/img/avatar_default.png`

Customizing Gitea pages

The `custom/templates` folder allows changing every single page of Gitea. Templates to

override can be found in the `templates` directory of Gitea source. Override by making a copy of the file under `custom/templates` using a full path structure matching source.

Any statement contained inside `{{` and `}}` are Gitea's template syntax and shouldn't be touched without fully understanding these components.

Adding links and tabs

If all you want is to add extra links to the top navigation bar, or extra tabs to the repository view, you can put them in `extra_links.tpl` and `extra_tabs.tpl` inside your `custom/templates/custom/` directory.

For instance, let's say you are in Germany and must add the famously legally-required "Impressum"/about page, listing who is responsible for the site's content: just place it under your "custom/public/" directory (for instance `custom/public/impressum.html`) and put a link to it in `custom/templates/custom/extra_links.tpl`.

To match the current style, the link should have the class name "item", and you can use `{{AppSubUrl}}` to get the base URL: `Impressum`

You can add new tabs in the same way, putting them in `extra_tabs.tpl`. The exact HTML needed to match the style of other tabs is in the file `templates/repo/header.tpl` ([source in GitHub](#))

Other additions to the page

Apart from `extra_links.tpl` and `extra_tabs.tpl`, there are other useful templates you can put in your `custom/templates/custom/` directory:

- `header.tpl`, just before the end of the tag where you can add custom CSS files for instance.
- `body_outer_pre.tpl`, right after the start of .
- `body_inner_pre.tpl`, before the top navigation bar, but already inside the main container
- `body_inner_post.tpl`, before the end of the main container.
- `body_outer_post.tpl`, before the bottom element.
- `footer.tpl`, right before the end of the tag, a good place for additional Javascript.

Customizing git ignores, labels, licenses, locales, and readmes.

Place custom files in corresponding sub-folder under `custom/options`.

Customizing the look of Gitea

Gitea has two built-in themes, the default theme `gitea`, and a dark theme `arc-green`. To change the look of your Gitea install change the value of `DEFAULT_THEME` in the `ui` section of `app.ini` to another one of the available options.

原文: <https://docs.gitea.io/zh-cn/customizing-gitea/>

加入 Gitea 开源

Hacking on Gitea

首先你需要一些运行环境，这和 [从源代码安装](#) 相同，如果你还没有设置好，可以先阅读那个章节。

如果你想为 Gitea 贡献代码，你需要 Fork 这个项目并且以 `master` 为开发分支。Gitea使用Govendor来管理依赖，因此所有依赖项都被工具自动copy在vendor子目录下。用下面的命令来下载源码：

```
1. go get -d code.gitea.io/gitea
```

然后你可以在 Github 上 fork [Gitea 项目](#)，之后可以通过下面的命令进入源码目录：

```
1. cd $GOPATH/src/code.gitea.io/gitea
```

要创建 pull requests 你还需要在源码中新增一个 remote 指向你 Fork 的地址，直接推送到 origin 的话会告诉你没有写权限：

```
1. git remote rename origin upstream
2. git remote add origin git@github.com:<USERNAME>/gitea.git
3. git fetch --all --prune
```

然后你就可以开始开发了。你可以看一下 `Makefile` 的内容。`make test` 可以运行测试程序，`make build` 将生成一个 `gitea` 可运行文件在根目录。如果你的提交比较复杂，尽量多写一些单元测试代码。

好了，到这里你已经设置好了所有的开发Gitea所需的环境。欢迎成为 Gitea 的 Contributor。

原文：<https://docs.gitea.io/zh-cn/hacking-on-gitea/>

Specific variables

Specific variables

This is an inventory of Gitea environment variables. They change Gitea behaviour.

Initialize them before Gitea command to be effective, for example:

```
1. GITEA_CUSTOM=/home/gitea/custom ./gitea web
```

From Go language

As Gitea is written in Go, it uses some Go variables, such as:

- GOOS
- GOARCH
- [GOPATH](#)

For documentation about each of the variables available, refer to the [official Go documentation](#).

Gitea files

- GITEA_WORK_DIR: Absolute path of working directory.
- GITEA_CUSTOM: Gitea uses GITEA_WORK_DIR/custom folder by default. Use this variable to change _custom directory.
- GOGS_WORK_DIR: Deprecated, use GITEA_WORK_DIR
- GOGS_CUSTOM: Deprecated, use GITEA_CUSTOM

Operating system specifics

- USER: System user that Gitea will run as. Used for some repository access strings.
- USERNAME: if no USER found, Gitea will use USERNAME
- HOME: User home directory path. The USERPROFILE environment variable is used in Windows.

Only on Windows

- USERPROFILE: User home directory path. If empty, uses HOMEDRIVE + HOMEPATH

- HOMEDRIVE: Main drive path used to access the home directory (C:)
- HOMEPATH: Home relative path in the given home drive path

Macaron (framework used by Gitea)

- HOST: Host Macaron will listen on
- PORT: Port Macaron will listen on
- MACARON_ENV: global variable to provide special functionality for development environments vs. production environments. If MACARON_ENV is set to "" or "development" then templates will be recompiled on every request. For more performance, set the MACARON_ENV environment variable to "production".

Miscellaneous

- SKIP_MINWINSVC: If set to 1, do not run as a service on Windows.
- ZOOKEEPER_PATH: [Zookeeper](#) jar file path

原文: <https://docs.gitea.io/zh-cn/specific-variables/>

配置说明


配置说明

这是针对Gitea配置文件的说明，你可以了解Gitea的强大配置。需要说明的是，你的所有改变请修改

`custom/conf/app.ini`

文件而不是源文件。所有默认值可以通过 `app.ini.sample` 查看到。如果你发现

`%(X)s`

这样的内容，请查看 [ini](#) 这里的说明。标注了  的配置项表明除非你真的理解这个配置项的意义，否则最好使用默认值。

Overall (DEFAULT)

- APP_NAME: 应用名称，改成你希望的名字。
- RUN_USER: 运行Gitea的用户，推荐使用 `git`；如果你自己的个人电脑使用改成你自己的用户名。如果设置不正确，Gitea可能崩溃。
- RUN_MODE: 从性能考虑，如果在产品级的服务上改成 `prod`。如果您使用安装向导安装的那么会自动设置为 `prod`。

Repository (repository)

- ROOT: 存放git工程的根目录。这里必须填绝对路径，默认值是 `~/gitea-repositories`。
- SCRIPT_TYPE: 服务器支持的Shell类型，通常是 `bash`，但有些服务器也有可能是 `sh`。
- ANSI_CHARSET: 默认字符编码。
- FORCE_PRIVATE: 强制所有git工程必须私有。
- DEFAULT_PRIVATE: 默认创建的git工程为私有。可以是 `last`, `private` 或 `public`。默认值是 `last` 表示用户最后创建的Repo的选择。
- MAX_CREATION_LIMIT: 全局最大每个用户创建的git工程数目， `-1` 表示没限制。
- PULL_REQUEST_QUEUE_LENGTH: 小心：合并请求测试队列的长度，尽量放大。

UI (ui)

- EXPLORE_PAGING_NUM: 探索页面每页显示的仓库数量。
- ISSUE_PAGING_NUM: 工单页面每页显示的工单数量。
- FEED_MAX_COMMIT_NUM: 活动流页面显示的最大提交树木。

UI - Admin (ui.admin)

- USER_PAGING_NUM: 用户管理页面每页显示的用户数量。

- REPO_PAGING_NUM: 仓库管理页面每页显示的仓库数量。
- NOTICE_PAGING_NUM: 系统提示页面每页显示的提示数量。
- ORG_PAGING_NUM: 组织管理页面每页显示的组织数量。

Markdown (markdown)

- ENABLE_HARD_LINE_BREAK: 是否启用硬换行扩展。

Server (server)

- PROTOCOL: 可选 http 或 https。
- DOMAIN: 服务器域名。
- ROOT_URL: Gitea服务器的对外 URL。
- HTTP_ADDR: HTTP 监听地址。
- HTTP_PORT: HTTP 监听端口。
- DISABLE_SSH: 是否禁用SSH。
- START_SSH_SERVER: 是否启用内部SSH服务器。
- SSH_PORT: SSH端口, 默认为 22。
- OFFLINE_MODE: 针对静态和头像文件禁用 CDN。
- DISABLE_ROUTER_LOG: 关闭日志中的路由日志。
- CERT_FILE: 启用HTTPS的证书文件。
- KEY_FILE: 启用HTTPS的密钥文件。
- STATIC_ROOT_PATH: 存放模板和静态文件的根目录, 默认是 Gitea 的根目录。
- ENABLE_GZIP: 启用应用级别的 GZIP 压缩。
- LANDING_PAGE: 未登录用户的默认页面, 可选 home 或 explore。
- LFS_START_SERVER: 是否启用 git-lfs 支持. 可以为 true 或 false, 默认是 false。
- LFS_CONTENT_PATH: 存放 lfs 命令上传的文件的地方, 默认是 data/lfs。
- LFS_JWT_SECRET: LFS 认证密钥, 改成自己的。

Database (database)

- DB_TYPE: 数据库类型, 可选 mysql, postgres, mssql, tidb 或 sqlite3。
- HOST: 数据库服务器地址和端口。
- NAME: 数据库名称。
- USER: 数据库用户名。
- PASSWD: 数据库用户密码。
- SSL_MODE: PostgreSQL数据库是否启用SSL模式。
- PATH: Tidb 或者 SQLite3 数据文件存放路径。
- LOG_SQL: **true**: 显示生成的SQL, 默认为真。

Security (security)

- `INSTALL_LOCK`: 是否允许运行安装向导, (跟管理员账号有关, 十分重要)。
- `SECRET_KEY`: 全局服务器安全密钥 最好改成你自己的 (当你运行安装向导的时候会被设置为一个随机值)。
- `LOGIN_REMEMBER_DAYS`: Cookie 保存时间, 单位天。
- `COOKIE_USERNAME`: 保存用户名的 cookie 名称。
- `COOKIE_REMEMBER_NAME`: 保存自动登录信息的 cookie 名称。
- `REVERSE_PROXY_AUTHENTICATION_USER`: 反向代理认证的 HTTP 头名称。

Service (service)

- `ACTIVE_CODE_LIVE_MINUTES`: 登陆验证码失效时间, 单位分钟。
- `RESET_PASSWD_CODE_LIVE_MINUTES`: 重置密码失效时间, 单位分钟。
- `REGISTER_EMAIL_CONFIRM`: 启用注册邮件激活, 前提是 Mailer 已经启用。
- `DISABLE_REGISTRATION`: 禁用注册, 启用后只能用管理员添加用户。
- `SHOW_REGISTRATION_BUTTON`: 是否显示注册按钮。
- `REQUIRE_SIGNIN_VIEW`: 是否所有页面都必须登录后才可访问。
- `ENABLE_CACHE_AVATAR`: 是否缓存来自 Gravatar 的头像。
- `ENABLE_NOTIFY_MAIL`: 是否发送工单创建等提醒邮件, 需要 Mailer 被激活。
- `ENABLE_REVERSE_PROXY_AUTHENTICATION`: 允许反向代理认证, 更多细节见: <https://github.com/gogits/gogs/issues/165>
- `ENABLE_REVERSE_PROXY_AUTO_REGISTRATION`: 允许通过反向认证做自动注册。
- `ENABLE_CAPTCHA`: 注册时使用图片验证码。

Webhook (webhook)

- `QUEUE_LENGTH`: 说明: Hook 任务队列长度。
- `DELIVER_TIMEOUT`: 请求webhooks的超时时间, 单位秒。
- `SKIP_TLS_VERIFY`: 是否允许不安全的证书。
- `PAGING_NUM`: 每页显示的Webhook 历史数量。

Mailer (mailer)

- `ENABLED`: 是否启用邮件服务。
- `DISABLE_HELO`: 禁用 HELO 命令。
- `HELO_HOSTNAME`: 自定义主机名来回应 HELO 命令。
- `HOST`: SMTP 主机地址和端口 (例如: `smtp.gitea.io:587`)。
- `FROM`: 邮件发送地址, RFC 5322. 这里可以填一个邮件地址或者 “Name” `email@example.com` 格式。
- `USER`: 用户名 (通常就是邮件地址)。
- `PASSWD`: 密码。
- `SKIP_VERIFY`: 忽略证书验证。 `/email@example.com`

说明：实际上 Gitea 仅仅支持基于 STARTTLS 的 SMTP。

Cache (cache)

- ADAPTER：缓存引擎，可以为 memory, redis 或 memcache。
- INTERVAL：只对内存缓存有效，GC间隔，单位秒。
- HOST：针对redis和memcache有效，主机地址和端口。
 - Redis：
network=tcp,addr=127.0.0.1:6379,password=macaron,db=0,pool_size=100,idle_time_out=180
 - Memache：127.0.0.1:9090;127.0.0.1:9091

Session (session)

- PROVIDER：Session 内容存储方式，可选 memory, file, redis 或 mysql。
- PROVIDER_CONFIG：如果是文件，那么这里填根目录；其他的要填主机地址和端口。
- COOKIE_SECURE：强制使用 HTTPS 作为session访问。
- GC_INTERVAL_TIME：Session失效时间。

Picture (picture)

- GRAVATAR_SOURCE：头像来源，可以是 gravatar, duoshuo 或者类似 <http://cn.gravatar.com/avatar/> 的来源
- DISABLE_GRAVATAR：开启则只使用内部头像。
- ENABLE_FEDERATED_AVATAR：启用头像联盟支持（参见 <http://www.libravatar.org>）

Attachment (attachment)

- ENABLED：是否允许用户上传附件。
- PATH：附件存储路径
- ALLOWED_TYPES：允许上传的附件类型。比如：image/jpeg|image/png，用 / 表示允许任何类型。
- MAX_SIZE：附件最大限制，单位 MB，比如： 4。
- MAX_FILES：一次最多上传的附件数量，比如： 5。

Log (log)

- ROOT_PATH：日志文件根目录。

- **MODE**: 日志记录模式, 默认是为 `console`。如果要写到多个通道, 用逗号分隔
- **LEVEL**: 日志级别, 默认为`Trace`。

Cron (cron)

- **ENABLED**: 是否在后台运行定期任务。
- **RUN_AT_START**: 是否启动时自动运行。

Cron - Update Mirrors (cron.update_mirrors)

- **SCHEDULE**: 自动同步镜像仓库的Cron语法, 比如: `@every 1h`。

Cron - Repository Health Check (cron.repo_health_check)

- **SCHEDULE**: 仓库健康监测的Cron语法, 比如: `@every 24h`。
- **TIMEOUT**: 仓库健康监测的超时时间, 比如: `60s`。
- **ARGS**: 执行 `git fsck` 命令的参数, 比如: `-unreachable -tags`。

Cron - Repository Statistics Check (cron.check_repo_stats)

- **RUN_AT_START**: 是否启动时自动运行仓库统计。
- **SCHEDULE**: 藏亏统计时的Cron 语法, 比如: `@every 24h`。

Git (git)

- **MAX_GIT_DIFF_LINES**: 比较视图中, 一个文件最多显示行数。
- **MAX_GIT_DIFF_LINE_CHARACTERS**: 比较视图中一行最大字符数。
- **MAX_GIT_DIFF_FILES**: 比较视图中的最大现实文件数目。
- **GC_ARGS**: 执行 `git gc` 命令的参数, 比如: `-aggressive -auto`。

Git - 超时设置 (git.timeout)

- **MIGRATE: 600**: 迁移外部仓库时的超时时间, 单位秒
- **MIRROR: 300**: 镜像外部仓库的超时时间, 单位秒
- **CLONE: 300**: 内部仓库间克隆的超时时间, 单位秒
- **PULL: 300**: 内部仓库间拉取的超时时间, 单位秒

- GC: **60**: git仓库GC的超时时间, 单位秒

markup (markup)

外部渲染工具支持, 你可以用你熟悉的文档渲染工具. 比如一下将新增一个名字为 `asciidoc` 的渲染工具which is followed `markup.` ini section. And there are some config items below.

```
1. [markup.asciidoc]
2. ENABLED = false
3. FILE_EXTENSIONS = .adoc, .asciidoc
4. RENDER_COMMAND = "asciidoc --out-file=- -"
5. IS_INPUT_FILE = false
```

- ENABLED: 是否启用, 默认为false。
- FILE_EXTENSIONS: 关联的文档的扩展名, 多个扩展名用都好分隔。
- RENDER_COMMAND: 工具的命令行命令及参数。
- IS_INPUT_FILE: 输入方式是最后一个参数为文件路径还是从标准输入读取。

Other (other)

- SHOW_FOOTER_BRANDING: 为真则在页面底部显示Gitea的字样。
- SHOW_FOOTER_VERSION: 为真则在页面底部显示Gitea的版本。

原文: <https://docs.gitea.io/zh-cn/config-cheat-sheet/>

Make

Make

Gitea makes heavy use of Make to automate tasks and improve development. This guide covers how to install Make.

On Linux

Install with the package manager.

On Ubuntu/Debian:

```
1. sudo apt-get install make
```

On Fedora/RHEL/CentOS:

```
1. sudo yum install make
```

On Windows

One of these three distributions of Make will run on Windows:

- [Single binary build](#). Copy somewhere and add to PATH.
 - [32-bits version](#)
 - [64-bits version](#)
- [MinGW](#) includes a build.
 - The binary is called mingw32-make.exe instead of make.exe. Add the bin folder to PATH.
- [Chocolatey package](#). Run `choco install make`

原文: <https://docs.gitea.io/zh-cn/make/>

API Usage

Gitea API Usage

Enabling/configuring API access

By default, `ENABLE_SWAGGER_ENDPOINT` is true, and `MAX_RESPONSE_ITEMS` is set to 50. See [Config CheatSheet](#) for more information.

Authentication via the API

Gitea supports these methods of API authentication:

- HTTP basic authentication
- token=... parameter in URL query string
- access_token=... parameter in URL query string
- Authorization: token ... header in HTTP headers

All of these methods accept the same apiKey token type. You can better understand this by looking at the code – as of this writing, Gitea parses queries and headers to find the token in [modules/auth/auth.go](#).

You can create an apiKey token via your gitea install's web interface: [Settings | Applications | Generate New Token](#).

More on the Authorization: header

For historical reasons, Gitea needs the word `token` included before the apiKey token in an authorization header, like this:

```
1. Authorization: token 65eaa9c8ef52460d22a93307fe0aee76289dc675
```

In a `curl` command, for instance, this would look like:

```
1. curl -X POST "http://localhost:4000/api/v1/repos/test1/test1/issues" \
2.   -H "accept: application/json" \
3.   -H "Authorization: token 65eaa9c8ef52460d22a93307fe0aee76289dc675" \
4.   -H "Content-Type: application/json" -d '{"body": "testing", "title": "test 20"}' -i
```

As mentioned above, the token used is the same one you would use in the `token=` string in a GET request.

Listing your issued tokens via the API

As mentioned in[#3842](#), `/users/:name/tokens` is special and requires you to authenticate using BasicAuth, as follows:

Using basic authentication:

```
1. $ curl --request GET --url https://yourusername:yourpassword@gitea.your.host/api/v1/users/yourusername/tokens
2. [{"name":"test","sha1":"..."}, {"name":"dev","sha1":"..."}]
```

原文: <https://docs.gitea.io/zh-cn/api-usage/>

帮助

- [Troubleshooting](#)
- [需要帮助](#)

Troubleshooting

Troubleshooting

This page contains some common seen issues and their solutions.

SSH issues

For issues reaching repositories over `ssh` while the gitea web front-end, but `https` based git repository access works fine, consider looking into the following.

1. `Permission denied (publickey)`.
2. `fatal: Could not read from remote repository.`

This error signifies that the server rejected a log in attempt, check the following things:

- On the client:
 - Ensure the public and private ssh keys are added to the correct Gitea user.
 - Make sure there are no issues in the remote url, ensure the name of the git user (before the @) is spelled correctly.
 - Ensure public and private ssh keys are correct on client machine.
 - Try to connect using ssh (`ssh git@myremote.example`) to ensure a connection can be made.
- On the server:
 - Make sure the repository exists and is correctly named.
 - Check the permissions of the `.ssh` directory in the system user's home directory.
 - Verify that the correct public keys are added to `.ssh/authorized_keys`. Try to run Rewrite '`.ssh/authorized_keys`' file (for Gitea SSH keys) on the Gitea admin panel.
 - Read gitea logs.
 - Read `/var/log/auth` (or similar).
 - Check permissions of repositories.

The following is an example of a missing public SSH key where authentication succeeded, but some other setting is preventing SSH from reaching the correct repository.

1. `fatal: Could not read from remote repository.`
- 2.
3. `Please` make sure you have the correct access rights
4. `and` the repository exists.

In this case, look into the following settings:

- On the server:
 - Make sure that the git system user has a usable shell set
 - Verify this with `getent passwd git | cut -d: -f7`
 - `usermod` or `chsh` can be used to modify this.
 - Ensure that the gitea `serv` command in `.ssh/authorized_keys` uses the correct configuration file.

Missing releases after migrating repository with tags

To migrate an repository *with* all tags you need to do two things

- Push tags to the repository:

```
git push -tags
```

- (Re-)sync tags of all repositories within gitea:
`gitea admin repo-sync-releases`

原文: <https://docs.gitea.io/zh-cn/troubleshooting/>

需要帮助

需要帮助?

如果您在使用或者开发过程中遇到问题，请到以下渠道咨询：

- 到[Github issue](#)提问(因为项目维护人员来自世界各地，为保证沟通顺畅，请使用英文提问)
- 中文问题到[gocn.io](#)提问
- 访问 [Discord server - 英文](#)
- 加入 QQ群 328432459 获得进一步的支持

原文： <https://docs.gitea.io/zh-cn/seek-help/>