**Register as a Windows service**

**Prerequisites**[**​**](https://docs.gitea.com/installation/windows-service#prerequisites)

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

app.ini

RUN\_USER = COMPUTERNAME$

Sets Gitea 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$

**Use absolute paths**[**​**](https://docs.gitea.com/installation/windows-service#use-absolute-paths)

If you use SQLite3, change the PATH to include the full path:

app.ini

[database]  
PATH = c:/gitea/data/gitea.db

**Register Gitea**[**​**](https://docs.gitea.com/installation/windows-service#register-gitea)

To register Gitea as a Windows service, open a command prompt (cmd) as an Administrator, then run the following command:

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

Do not forget to replace C:\gitea with the correct Gitea directory.

Open "Windows Services", search for the service named "gitea", right-click it and click on "Run". If everything is OK, Gitea will be reachable on http://localhost:3000 (or the port that was configured).

**Service startup type**[**​**](https://docs.gitea.com/installation/windows-service#service-startup-type)

It was observed that on loaded systems during boot Gitea service may fail to start with timeout records in Windows Event Log. In that case change startup type to Automatic-Delayed. This can be done during service creation, or by running config command

sc.exe config gitea start= delayed-auto

**Adding startup dependencies**[**​**](https://docs.gitea.com/installation/windows-service#adding-startup-dependencies)

To add a startup dependency to the Gitea Windows service (eg Mysql, Mariadb), as an Administrator, then run the following command:

sc.exe config gitea depend= mariadb

This will ensure that when the Windows machine restarts, the automatic starting of Gitea is postponed until the database is ready and thus mitigate failed startups.

**Unregister Gitea**[**​**](https://docs.gitea.com/installation/windows-service#unregister-gitea)

To unregister Gitea as a Windows service, open a command prompt (cmd) as an Administrator and run:

sc.exe delete gitea

# Command Line

## Usage[​](https://docs.gitea.com/administration/command-line#usage)

gitea [global options] command [command or global options] [arguments...]

## Global options[​](https://docs.gitea.com/administration/command-line#global-options)

All global options can be placed at the command level.

* --help, -h: Show help text and exit. Optional.
* --version, -v: Show version and exit. Optional. (example: Gitea version 1.1.0+218-g7b907ed built with: bindata, sqlite).
* --work-path path, -w path: Gitea's work path. Optional. (default: the binary's path or $GITEA\_WORK\_DIR)
* --custom-path path, -C path: Gitea's custom folder path. Optional. (default: WorkPath/custom or $GITEA\_CUSTOM).
* --config path, -c path: Gitea configuration file path. Optional. (default: CustomPath/conf/app.ini).

NB: The defaults custom-path, config and work-path can also be changed at build time (if preferred).

## Commands[​](https://docs.gitea.com/administration/command-line#commands)

### web[​](https://docs.gitea.com/administration/command-line" \l "web" \o "Direct link to web)

Starts the server:

* Options:
  + --port number, -p number: Port number. Optional. (default: 3000). Overrides configuration file.
  + --install-port number: Port number to run the install page on. Optional. (default: 3000). Overrides configuration file.
  + --pid path, -P path: Pidfile path. Optional.
  + --quiet, -q: Only emit Fatal logs on the console for logs emitted before logging set up.
  + --verbose: Emit tracing logs on the console for logs emitted before logging is set-up.
* Examples:
  + gitea web
  + gitea web --port 80
  + gitea web --config /etc/gitea.ini --pid /some/custom/gitea.pid
* Notes:
  + Gitea should not be run as root. To bind to a port below 1024, you can use setcap on Linux: sudo setcap 'cap\_net\_bind\_service=+ep' /path/to/gitea. This will need to be redone every time you update Gitea.

### admin[​](https://docs.gitea.com/administration/command-line" \l "admin" \o "Direct link to admin)

Admin operations:

* Commands:
  + user:
    - list:
      * Options:
        + --admin: List only admin users. Optional.
      * Description: lists all users that exist
      * Examples:
        + gitea admin user list
    - delete:
      * Options:
        + --email: Email of the user to be deleted.
        + --username: Username of user to be deleted.
        + --id: ID of user to be deleted.
        + One of --id, --username or --email is required. If more than one is provided then all have to match.
      * Examples:
        + gitea admin user delete --id 1
    - create:
      * Options:
        + --name value: Username. Required. As of Gitea 1.9.0, use the --username flag instead.
        + --username value: Username. Required. New in Gitea 1.9.0.
        + --password value: Password. Required.
        + --email value: Email. Required.
        + --admin: If provided, this makes the user an admin. Optional.
        + --access-token: If provided, an access token will be created for the user. Optional. (default: false).
        + --must-change-password: The created user will be required to set a new password after the initial login, default: true. It could be disabled by --must-change-password=false.
        + --random-password: If provided, a randomly generated password will be used as the password of the created user. The value of --password will be discarded. Optional.
        + --random-password-length: If provided, it will be used to configure the length of the randomly generated password. Optional. (default: 12)
      * Examples:
        + gitea admin user create --username myname --password asecurepassword --email me@example.com
    - change-password:
      * Options:
        + --username value, -u value: Username. Required.
        + --password value, -p value: New password. Required.
        + --must-change-password: The user is required to set a new password after the login, default: true. It could be disabled by --must-change-password=false.
      * Examples:
        + gitea admin user change-password --username myname --password asecurepassword
    - must-change-password:
      * Args:
        + [username...]: Users that must change their passwords
      * Options:
        + --all, -A: Force a password change for all users
        + --exclude username, -e username: Exclude the given user. Can be set multiple times.
        + --unset: Revoke forced password change for the given users
    - generate-access-token:
      * Options:
        + --username value, -u value: Username. Required.
        + --token-name value, -t value: Token name. Required.
        + --scopes value: Comma-separated list of scopes. Scopes follow the format [read|write]:<block> or all where <block> is one of the available visual groups you can see when opening the API page showing the available routes (for example repo).
      * Examples:
        + gitea admin user generate-access-token --username myname --token-name mytoken
        + gitea admin user generate-access-token --help
  + regenerate
    - Options:
      * hooks: Regenerate Git Hooks for all repositories
      * keys: Regenerate authorized\_keys file
    - Examples:
      * gitea admin regenerate hooks
      * gitea admin regenerate keys
  + auth:
    - list:
      * Description: lists all external authentication sources that exist
      * Examples:
        + gitea admin auth list
    - delete:
      * Options:
        + --id: ID of source to be deleted. Required.
      * Examples:
        + gitea admin auth delete --id 1
    - add-oauth:
      * Options:
        + --name: Application Name.
        + --provider: OAuth2 Provider.
        + --key: Client ID (Key).
        + --secret: Client Secret.
        + --auto-discover-url: OpenID Connect Auto Discovery URL (only required when using OpenID Connect as provider).
        + --use-custom-urls: Use custom URLs for GitLab/GitHub OAuth endpoints.
        + --custom-tenant-id: Use custom Tenant ID for OAuth endpoints.
        + --custom-auth-url: Use a custom Authorization URL (option for GitLab/GitHub).
        + --custom-token-url: Use a custom Token URL (option for GitLab/GitHub).
        + --custom-profile-url: Use a custom Profile URL (option for GitLab/GitHub).
        + --custom-email-url: Use a custom Email URL (option for GitHub).
        + --icon-url: Custom icon URL for OAuth2 login source.
        + --skip-local-2fa: Allow source to override local 2FA. (Optional)
        + --scopes: Additional scopes to request for this OAuth2 source. (Optional)
        + --required-claim-name: Claim name that has to be set to allow users to login with this source. (Optional)
        + --required-claim-value: Claim value that has to be set to allow users to login with this source. (Optional)
        + --group-claim-name: Claim name providing group names for this source. (Optional)
        + --admin-group: Group Claim value for administrator users. (Optional)
        + --restricted-group: Group Claim value for restricted users. (Optional)
        + --group-team-map: JSON mapping between groups and org teams. (Optional)
        + --group-team-map-removal: Activate automatic team membership removal depending on groups. (Optional)
      * Examples:
        + gitea admin auth add-oauth --name external-github --provider github --key OBTAIN\_FROM\_SOURCE --secret OBTAIN\_FROM\_SOURCE
    - update-oauth:
      * Options:
        + --id: ID of source to be updated. Required.
        + --name: Application Name.
        + --provider: OAuth2 Provider.
        + --key: Client ID (Key).
        + --secret: Client Secret.
        + --auto-discover-url: OpenID Connect Auto Discovery URL (only required when using OpenID Connect as provider).
        + --use-custom-urls: Use custom URLs for GitLab/GitHub OAuth endpoints.
        + --custom-tenant-id: Use custom Tenant ID for OAuth endpoints.
        + --custom-auth-url: Use a custom Authorization URL (option for GitLab/GitHub).
        + --custom-token-url: Use a custom Token URL (option for GitLab/GitHub).
        + --custom-profile-url: Use a custom Profile URL (option for GitLab/GitHub).
        + --custom-email-url: Use a custom Email URL (option for GitHub).
        + --icon-url: Custom icon URL for OAuth2 login source.
        + --skip-local-2fa: Allow source to override local 2FA. (Optional)
        + --scopes: Additional scopes to request for this OAuth2 source.
        + --required-claim-name: Claim name that has to be set to allow users to login with this source. (Optional)
        + --required-claim-value: Claim value that has to be set to allow users to login with this source. (Optional)
        + --group-claim-name: Claim name providing group names for this source. (Optional)
        + --admin-group: Group Claim value for administrator users. (Optional)
        + --restricted-group: Group Claim value for restricted users. (Optional)
      * Examples:
        + gitea admin auth update-oauth --id 1 --name external-github-updated
    - add-smtp:
      * Options:
        + --name: Application Name. Required.
        + --auth-type: SMTP Authentication Type (PLAIN/LOGIN/CRAM-MD5). Default to PLAIN.
        + --host: SMTP host. Required.
        + --port: SMTP port. Required.
        + --force-smtps: SMTPS is always used on port 465. Set this to force SMTPS on other ports.
        + --skip-verify: Skip TLS verify.
        + --helo-hostname: Hostname sent with HELO. Leave blank to send current hostname.
        + --disable-helo: Disable SMTP helo.
        + --allowed-domains: Leave empty to allow all domains. Separate multiple domains with a comma (',').
        + --skip-local-2fa: Skip 2FA to log on.
        + --active: This Authentication Source is Activated. Remarks: --force-smtps, --skip-verify, --disable-helo, --skip-loca-2fs and --active options can be used in form:
        + --option, --option=true to enable
        + --option=false to disable If those options are not specified value would not be changed in update-smtp or would use default false value in add-smtp
      * Examples:
        + gitea admin auth add-smtp --name ldap --host smtp.mydomain.org --port 587 --skip-verify --active
    - update-smtp:
      * Options:
        + --id: ID of source to be updated. Required.
        + other options are shared with add-smtp
      * Examples:
        + gitea admin auth update-smtp --id 1 --host smtp.mydomain.org --port 587 --skip-verify=false
        + gitea admin auth update-smtp --id 1 --active=false
    - add-ldap: Add new LDAP (via Bind DN) authentication source
      * Options:
        + --name value: Authentication name. Required.
        + --not-active: Deactivate the authentication source.
        + --security-protocol value: Security protocol name. Required.
        + --skip-tls-verify: Disable TLS verification.
        + --host value: The address where the LDAP server can be reached. Required.
        + --port value: The port to use when connecting to the LDAP server. Required.
        + --user-search-base value: The LDAP base at which user accounts will be searched for. Required.
        + --user-filter value: An LDAP filter declaring how to find the user record that is attempting to authenticate. Required.
        + --admin-filter value: An LDAP filter specifying if a user should be given administrator privileges.
        + --restricted-filter value: An LDAP filter specifying if a user should be given restricted status.
        + --username-attribute value: The attribute of the user’s LDAP record containing the user name.
        + --firstname-attribute value: The attribute of the user’s LDAP record containing the user’s first name.
        + --surname-attribute value: The attribute of the user’s LDAP record containing the user’s surname.
        + --email-attribute value: The attribute of the user’s LDAP record containing the user’s email address. Required.
        + --public-ssh-key-attribute value: The attribute of the user’s LDAP record containing the user’s public ssh key.
        + --avatar-attribute value: The attribute of the user’s LDAP record containing the user’s avatar.
        + --bind-dn value: The DN to bind to the LDAP server with when searching for the user.
        + --bind-password value: The password for the Bind DN, if any.
        + --attributes-in-bind: Fetch attributes in bind DN context.
        + --synchronize-users: Enable user synchronization.
        + --page-size value: Search page size.
      * Examples:
        + gitea admin auth add-ldap --name ldap --security-protocol unencrypted --host mydomain.org --port 389 --user-search-base "ou=Users,dc=mydomain,dc=org" --user-filter "(&(objectClass=posixAccount)(|(uid=%[1]s)(mail=%[1]s)))" --email-attribute mail
    - update-ldap: Update existing LDAP (via Bind DN) authentication source
      * Options:
        + --id value: ID of authentication source. Required.
        + --name value: Authentication name.
        + --not-active: Deactivate the authentication source.
        + --security-protocol value: Security protocol name.
        + --skip-tls-verify: Disable TLS verification.
        + --host value: The address where the LDAP server can be reached.
        + --port value: The port to use when connecting to the LDAP server.
        + --user-search-base value: The LDAP base at which user accounts will be searched for.
        + --user-filter value: An LDAP filter declaring how to find the user record that is attempting to authenticate.
        + --admin-filter value: An LDAP filter specifying if a user should be given administrator privileges.
        + --restricted-filter value: An LDAP filter specifying if a user should be given restricted status.
        + --username-attribute value: The attribute of the user’s LDAP record containing the user name.
        + --firstname-attribute value: The attribute of the user’s LDAP record containing the user’s first name.
        + --surname-attribute value: The attribute of the user’s LDAP record containing the user’s surname.
        + --email-attribute value: The attribute of the user’s LDAP record containing the user’s email address.
        + --public-ssh-key-attribute value: The attribute of the user’s LDAP record containing the user’s public ssh key.
        + --avatar-attribute value: The attribute of the user’s LDAP record containing the user’s avatar.
        + --bind-dn value: The DN to bind to the LDAP server with when searching for the user.
        + --bind-password value: The password for the Bind DN, if any.
        + --attributes-in-bind: Fetch attributes in bind DN context.
        + --synchronize-users: Enable user synchronization.
        + --page-size value: Search page size.
      * Examples:
        + gitea admin auth update-ldap --id 1 --name "my ldap auth source"
        + gitea admin auth update-ldap --id 1 --username-attribute uid --firstname-attribute givenName --surname-attribute sn
    - add-ldap-simple: Add new LDAP (simple auth) authentication source
      * Options:
        + --name value: Authentication name. Required.
        + --not-active: Deactivate the authentication source.
        + --security-protocol value: Security protocol name. Required.
        + --skip-tls-verify: Disable TLS verification.
        + --host value: The address where the LDAP server can be reached. Required.
        + --port value: The port to use when connecting to the LDAP server. Required.
        + --user-search-base value: The LDAP base at which user accounts will be searched for.
        + --user-filter value: An LDAP filter declaring how to find the user record that is attempting to authenticate. Required.
        + --admin-filter value: An LDAP filter specifying if a user should be given administrator privileges.
        + --restricted-filter value: An LDAP filter specifying if a user should be given restricted status.
        + --username-attribute value: The attribute of the user’s LDAP record containing the user name.
        + --firstname-attribute value: The attribute of the user’s LDAP record containing the user’s first name.
        + --surname-attribute value: The attribute of the user’s LDAP record containing the user’s surname.
        + --email-attribute value: The attribute of the user’s LDAP record containing the user’s email address. Required.
        + --public-ssh-key-attribute value: The attribute of the user’s LDAP record containing the user’s public ssh key.
        + --avatar-attribute value: The attribute of the user’s LDAP record containing the user’s avatar.
        + --user-dn value: The user’s DN. Required.
      * Examples:
        + gitea admin auth add-ldap-simple --name ldap --security-protocol unencrypted --host mydomain.org --port 389 --user-dn "cn=%s,ou=Users,dc=mydomain,dc=org" --user-filter "(&(objectClass=posixAccount)(cn=%s))" --email-attribute mail
    - update-ldap-simple: Update existing LDAP (simple auth) authentication source
      * Options:
        + --id value: ID of authentication source. Required.
        + --name value: Authentication name.
        + --not-active: Deactivate the authentication source.
        + --security-protocol value: Security protocol name.
        + --skip-tls-verify: Disable TLS verification.
        + --host value: The address where the LDAP server can be reached.
        + --port value: The port to use when connecting to the LDAP server.
        + --user-search-base value: The LDAP base at which user accounts will be searched for.
        + --user-filter value: An LDAP filter declaring how to find the user record that is attempting to authenticate.
        + --admin-filter value: An LDAP filter specifying if a user should be given administrator privileges.
        + --restricted-filter value: An LDAP filter specifying if a user should be given restricted status.
        + --username-attribute value: The attribute of the user’s LDAP record containing the user name.
        + --firstname-attribute value: The attribute of the user’s LDAP record containing the user’s first name.
        + --surname-attribute value: The attribute of the user’s LDAP record containing the user’s surname.
        + --email-attribute value: The attribute of the user’s LDAP record containing the user’s email address.
        + --public-ssh-key-attribute value: The attribute of the user’s LDAP record containing the user’s public ssh key.
        + --avatar-attribute value: The attribute of the user’s LDAP record containing the user’s avatar.
        + --user-dn value: The user’s DN.
      * Examples:
        + gitea admin auth update-ldap-simple --id 1 --name "my ldap auth source"
        + gitea admin auth update-ldap-simple --id 1 --username-attribute uid --firstname-attribute givenName --surname-attribute sn

### cert[​](https://docs.gitea.com/administration/command-line" \l "cert" \o "Direct link to cert)

Generates a self-signed SSL certificate. Outputs to cert.pem and key.pem in the current directory and will overwrite any existing files.

* Options:
  + --host value: Comma separated hostnames and ips which this certificate is valid for. Wildcards are supported. Required.
  + --ecdsa-curve value: ECDSA curve to use to generate a key. Optional. Valid options are P224, P256, P384, P521.
  + --rsa-bits value: Size of RSA key to generate. Optional. Ignored if --ecdsa-curve is set. (default: 3072).
  + --start-date value: Creation date. Optional. (format: Jan 1 15:04:05 2011).
  + --duration value: Duration which the certificate is valid for. Optional. (default: 8760h0m0s)
  + --ca: If provided, this cert generates it's own certificate authority. Optional.
* Examples:
  + gitea cert --host git.example.com,example.com,www.example.com --ca

### dump[​](https://docs.gitea.com/administration/command-line" \l "dump" \o "Direct link to dump)

Dumps all files and databases into a zip file. Outputs into a file like gitea-dump-1482906742.zip in the current directory.

* Options:
  + --file name, -f name: Name of the dump file with will be created. Optional. (default: gitea-dump-[timestamp].zip).
  + --tempdir path, -t path: Path to the temporary directory used. Optional. (default: /tmp).
  + --skip-repository, -R: Skip the repository dumping. Optional.
  + --skip-custom-dir: Skip dumping of the custom dir. Optional.
  + --skip-lfs-data: Skip dumping of LFS data. Optional.
  + --skip-attachment-data: Skip dumping of attachment data. Optional.
  + --skip-package-data: Skip dumping of package data. Optional.
  + --skip-log: Skip dumping of log data. Optional.
  + --database, -d: Specify the database SQL syntax. Optional (supported arguments: sqlite3, mysql, mssql, postgres).
  + --verbose, -V: If provided, shows additional details. Optional.
  + --type: Set the dump output format. Optional. (formats: zip, tar, tar.sz, tar.gz, tar.xz, tar.bz2, tar.br, tar.lz4, tar.zst default: zip).
* Examples:
  + gitea dump
  + gitea dump --verbose

### generate[​](https://docs.gitea.com/administration/command-line" \l "generate" \o "Direct link to generate)

Generates random values and tokens for usage in configuration file. Useful for generating values for automatic deployments.

* Commands:
  + secret:
    - Options:
      * INTERNAL\_TOKEN: Token used for an internal API call authentication.
      * JWT\_SECRET: LFS & OAUTH2 JWT authentication secret (LFS\_JWT\_SECRET is aliased to this option for backwards compatibility).
      * SECRET\_KEY: Global secret key.
    - Examples:
      * gitea generate secret INTERNAL\_TOKEN
      * gitea generate secret JWT\_SECRET
      * gitea generate secret SECRET\_KEY

### keys[​](https://docs.gitea.com/administration/command-line" \l "keys" \o "Direct link to keys)

Provides an SSHD AuthorizedKeysCommand. Needs to be configured in the sshd config file:

...  
*# The value of -e and the AuthorizedKeysCommandUser should match the*  
*# username running Gitea*  
AuthorizedKeysCommandUser git  
AuthorizedKeysCommand /path/to/gitea keys -e git -u %u -t %t -k %k

The command will return the appropriate authorized\_keys line for the provided key. You should also set the value SSH\_CREATE\_AUTHORIZED\_KEYS\_FILE=false in the [server] section of app.ini.

NB: opensshd requires the Gitea program to be owned by root and not writable by group or others. The program must be specified by an absolute path. NB: Gitea must be running for this command to succeed.

### migrate[​](https://docs.gitea.com/administration/command-line" \l "migrate" \o "Direct link to migrate)

Migrates the database. This command can be used to run other commands before starting the server for the first time. This command is idempotent.

### doctor check[​](https://docs.gitea.com/administration/command-line#doctor-check)

Diagnose and potentially fix problems with the current Gitea instance. Several checks are run by default, but additional ones can be run:

* gitea doctor check --list - will list all the available checks
* gitea doctor check --all - will run all available checks
* gitea doctor check --default - will run the default checks
* gitea doctor check --run [check(s),]... - will run the named checks

Some problems can be automatically fixed by passing the --fix option. Extra logging can be set with --log-file=....

#### doctor recreate-table[​](https://docs.gitea.com/administration/command-line#doctor-recreate-table)

Sometimes when there are migrations the old columns and default values may be left unchanged in the database schema. This may lead to warning such as:

2020/08/02 11:32:29 ...rm/session\_schema.go:360:Sync() [W] Table user Column keep\_activity\_private db default is , struct default is 0

You can cause Gitea to recreate these tables and copy the old data into the new table with the defaults set appropriately by using:

gitea doctor recreate-table user

You can ask Gitea to recreate multiple tables using:

gitea doctor recreate-table table1 table2 ...

And if you would like Gitea to recreate all tables simply call:

gitea doctor recreate-table

It is highly recommended to back-up your database before running these commands.

### doctor convert[​](https://docs.gitea.com/administration/command-line#doctor-convert)

Converts a MySQL database from utf8 to utf8mb4 or a MSSQL database from varchar to nvarchar.

### manager[​](https://docs.gitea.com/administration/command-line" \l "manager" \o "Direct link to manager)

Manage running server operations:

* Commands:
  + shutdown: Gracefully shutdown the running process
  + restart: Gracefully restart the running process - (not implemented for windows servers)
  + flush-queues: Flush queues in the running process
    - Options:
      * --timeout value: Timeout for the flushing process (default: 1m0s)
      * --non-blocking: Set to true to not wait for flush to complete before returning
  + logging: Adjust logging commands
    - Commands:
      * pause: Pause logging
        + Notes:

The logging level will be raised to INFO temporarily if it is below this level.

Gitea will buffer logs up to a certain point and will drop them after that point.

* + - * resume: Resume logging
      * release-and-reopen: Cause Gitea to release and re-open files and connections used for logging (Equivalent to sending SIGUSR1 to Gitea.)
      * remove name: Remove the named logger
        + Options:

--group group, -g group: Set the group to remove the sublogger from. (defaults to default)

* + - * add: Add a logger
        + Commands:

console: Add a console logger

Options:

--group value, -g value: Group to add logger to - will default to "default"

--name value, -n value: Name of the new logger - will default to mode

--level value, -l value: Logging level for the new logger

--stacktrace-level value, -L value: Stacktrace logging level

--flags value, -F value: Flags for the logger

--expression value, -e value: Matching expression for the logger

--prefix value, -p value: Prefix for the logger

--color: Use color in the logs

--stderr: Output console logs to stderr - only relevant for console

file: Add a file logger

Options:

--group value, -g value: Group to add logger to - will default to "default"

--name value, -n value: Name of the new logger - will default to mode

--level value, -l value: Logging level for the new logger

--stacktrace-level value, -L value: Stacktrace logging level

--flags value, -F value: Flags for the logger

--expression value, -e value: Matching expression for the logger

--prefix value, -p value: Prefix for the logger

--color: Use color in the logs

--filename value, -f value: Filename for the logger -

--rotate, -r: Rotate logs

--max-size value, -s value: Maximum size in bytes before rotation

--daily, -d: Rotate logs daily

--max-days value, -D value: Maximum number of daily logs to keep

--compress, -z: Compress rotated logs

--compression-level value, -Z value: Compression level to use

conn: Add a network connection logger

Options:

--group value, -g value: Group to add logger to - will default to "default"

--name value, -n value: Name of the new logger - will default to mode

--level value, -l value: Logging level for the new logger

--stacktrace-level value, -L value: Stacktrace logging level

--flags value, -F value: Flags for the logger

--expression value, -e value: Matching expression for the logger

--prefix value, -p value: Prefix for the logger

--color: Use color in the logs

--reconnect-on-message, -R: Reconnect to host for every message

--reconnect, -r: Reconnect to host when connection is dropped

--protocol value, -P value: Set protocol to use: tcp, unix, or udp (defaults to tcp)

--address value, -a value: Host address and port to connect to (defaults to :7020)

smtp: Add an SMTP logger

Options:

--group value, -g value: Group to add logger to - will default to "default"

--name value, -n value: Name of the new logger - will default to mode

--level value, -l value: Logging level for the new logger

--stacktrace-level value, -L value: Stacktrace logging level

--flags value, -F value: Flags for the logger

--expression value, -e value: Matching expression for the logger

--prefix value, -p value: Prefix for the logger

--color: Use color in the logs

--username value, -u value: Mail server username

--password value, -P value: Mail server password

--host value, -H value: Mail server host (defaults to: 127.0.0.1:25)

--send-to value, -s value: Email address(es) to send to

--subject value, -S value: Subject header of sent emails

* + processes: Display Gitea processes and goroutine information
    - Options:
      * --flat: Show processes as flat table rather than as tree
      * --no-system: Do not show system processes
      * --stacktraces: Show stacktraces for goroutines associated with processes
      * --json: Output as json
      * --cancel PID: Send cancel to process with PID. (Only for non-system processes.)

### dump-repo[​](https://docs.gitea.com/administration/command-line" \l "dump-repo" \o "Direct link to dump-repo)

Dump-repo dumps repository data from Git/GitHub/Gitea/GitLab:

* Options:
  + --git\_service service : Git service, it could be git, github, gitea, gitlab, If clone\_addr could be recognized, this could be ignored.
  + --repo\_dir dir, -r dir: Repository dir path to store the data
  + --clone\_addr addr: The URL will be clone, currently could be a git/github/gitea/gitlab http/https URL. i.e. <https://github.com/lunny/tango.git>
  + --auth\_username lunny: The username to visit the clone\_addr
  + --auth\_password <password>: The password to visit the clone\_addr
  + --auth\_token <token>: The personal token to visit the clone\_addr
  + --owner\_name lunny: The data will be stored on a directory with owner name if not empty
  + --repo\_name tango: The data will be stored on a directory with repository name if not empty
  + --units <units>: Which items will be migrated, one or more units should be separated as comma. wiki, issues, labels, releases, release\_assets, milestones, pull\_requests, comments are allowed. Empty means all units.

### restore-repo[​](https://docs.gitea.com/administration/command-line" \l "restore-repo" \o "Direct link to restore-repo)

Restore-repo restore repository data from disk dir:

* Options:
  + --repo\_dir dir, -r dir: Repository dir path to restore from
  + --owner\_name lunny: Restore destination owner name
  + --repo\_name tango: Restore destination repository name
  + --units <units>: Which items will be restored, one or more units should be separated as comma. wiki, issues, labels, releases, release\_assets, milestones, pull\_requests, comments are allowed. Empty means all units.

### actions generate-runner-token[​](https://docs.gitea.com/administration/command-line#actions-generate-runner-token)

Generate a new token for a runner to use to register with the server

* Options:
  + --scope {owner}[/{repo}], -s {owner}[/{repo}]: To limit the scope of the runner, no scope means the runner can be used for all repos, but you can also limit it to a specific repo or owner

To register a global runner:

gitea actions generate-runner-token

To register a runner for a specific organization, in this case org:

gitea actions generate-runner-token -s org

To register a runner for a specific repo, in this case username/test-repo:

gitea actions generate-runner-token -s username/test-repo

**Authentication**

**LDAP (Lightweight Directory Access Protocol)**[**​**](https://docs.gitea.com/administration/authentication#ldap-lightweight-directory-access-protocol)

Both the LDAP via BindDN and the simple auth LDAP share the following fields:

* Authorization Name **(required)**
  + A name to assign to the new method of authorization.
* Host **(required)**
  + The address where the LDAP server can be reached.
  + Example: mydomain.com
* Port **(required)**
  + The port to use when connecting to the server.
  + Example: 389 for LDAP or 636 for LDAP SSL
* Enable TLS Encryption (optional)
  + Whether to use TLS when connecting to the LDAP server.
* Admin Filter (optional)
  + An LDAP filter specifying if a user should be given administrator privileges. If a user account passes the filter, the user will be privileged as an administrator.
  + Example: (objectClass=adminAccount)
  + Example for Microsoft Active Directory (AD): (memberOf=CN=admin-group,OU=example,DC=example,DC=org)
* Username attribute (optional)
  + The attribute of the user's LDAP record containing the user name. Given attribute value will be used for new Gitea 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 Gitea account name, see "User Filter".
  + Example: uid
  + Example for Microsoft Active Directory (AD): sAMAccountName
* First name attribute (optional)
  + 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
* Surname attribute (optional)
  + The attribute of the user's LDAP record containing the user's surname. This will be used to populate their account information.
  + Example: sn
* E-mail attribute **(required)**
  + 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

**LDAP via BindDN**[**​**](https://docs.gitea.com/administration/authentication#ldap-via-binddn)

Adds the following fields:

* Bind DN (optional)
  + 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
* Bind Password (optional)
  + The password for the Bind DN specified above, if any. *Note: The password is stored encrypted with the SECRET\_KEY on the server. It is still recommended to ensure that the Bind DN has as few privileges as possible.*
* User Search Base **(required)**
  + The LDAP base at which user accounts will be searched for.
  + Example: ou=Users,dc=mydomain,dc=com
* User Filter **(required)**
  + An LDAP filter declaring how to find the user record that is attempting to authenticate. The %[1]s matching parameter will be substituted with login name given on sign-in form.
  + Example: (&(objectClass=posixAccount)(|(uid=%[1]s)(mail=%[1]s)))
  + Example for Microsoft Active Directory (AD): (&(objectCategory=Person)(memberOf=CN=user-group,OU=example,DC=example,DC=org)(sAMAccountName=%s)(!(UserAccountControl:1.2.840.113556.1.4.803:=2)))
  + To substitute more than once, %[1]s should be used instead, e.g. 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)))
* Enable user synchronization
  + This option enables a periodic task that synchronizes the Gitea users with the LDAP server. The default period is every 24 hours but that can be changed in the app.ini file. See the *cron.sync\_external\_users* section in the [sample app.ini](https://github.com/go-gitea/gitea/blob/release/v1.22/custom/conf/app.example.ini) for detailed comments about that section. The *User Search Base* and *User Filter* settings described above will limit which users can use Gitea and which users will be synchronized. When initially run the task will create all LDAP users that match the given settings so take care if working with large Enterprise LDAP directories.

**LDAP using simple auth**[**​**](https://docs.gitea.com/administration/authentication#ldap-using-simple-auth)

Adds the following fields:

* User DN **(required)**
  + 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
* User Search Base (optional)
  + The LDAP base at which user accounts will be searched for.
  + Example: ou=Users,dc=mydomain,dc=com
* User Filter **(required)**
  + An LDAP filter declaring when a user should be allowed to log in. The %[1]s matching parameter will be substituted with login name given on sign-in form.
  + Example: (&(objectClass=posixAccount)(|(cn=%[1]s)(mail=%[1]s)))
  + Example: (&(objectClass=posixAccount)(|(uid=%[1]s)(mail=%[1]s)))

**Verify group membership in LDAP**[**​**](https://docs.gitea.com/administration/authentication#verify-group-membership-in-ldap)

Uses the following fields:

* Group Search Base DN (optional)
  + The LDAP DN used for groups.
  + Example: ou=group,dc=mydomain,dc=com
* Group Attribute Containing List Of Users (optional)
  + The attribute of the group object that lists/contains the group members.
  + Example: memberUid or member
* User Attribute Listed in Group (optional)
  + The user attribute that is used to reference a user in the group object.
  + Example: uid if the group objects contains a member: bender and the user object contains a uid: bender.
  + Example: dn if the group object contains a member: uid=bender,ou=users,dc=planetexpress,dc=com.
* Verify group membership in LDAP (optional)
  + An LDAP filter declaring how to find valid groups in the above DN.
  + Example: (|(cn=gitea\_users)(cn=admins))

**PAM (Pluggable Authentication Module)**[**​**](https://docs.gitea.com/administration/authentication#pam-pluggable-authentication-module)

This procedure enables PAM authentication. Users may still be added to the system manually using the user administration. PAM provides a mechanism to automatically add users to the current database by testing them against PAM authentication. To work with normal Linux passwords, the user running Gitea must also have read access to /etc/shadow in order to check the validity of the account when logging in using a public key.

**Note**: If a user has added SSH public keys into Gitea, the use of these keys *may* bypass the login check system. Therefore, if you wish to disable a user who authenticates with PAM, you *should* also manually disable the account in Gitea using the built-in user manager.

1. Configure and prepare the installation.
   * It is recommended that you create an administrative user.
   * Deselecting automatic sign-up may also be desired.
2. Once the database has been initialized, log in as the newly created administrative user.
3. Navigate to the user setting (icon in top-right corner), and select Site Administration -> Authentication Sources, and select Add Authentication Source.
4. Fill out the field as follows:
   * Authentication Type : PAM
   * Name : Any value should be valid here, use "System Authentication" if you'd like.
   * PAM Service Name : Select the appropriate file listed under /etc/pam.d/ that performs the authentication desired.[1](https://docs.gitea.com/administration/authentication#user-content-fn-1)
   * PAM Email Domain : The e-mail suffix to append to user authentication. For example, if the login system expects a user called gituser, and this field is set to mail.com, then Gitea will expect the user email field for an authenticated GIT instance to be gituser@mail.com.[2](https://docs.gitea.com/administration/authentication#user-content-fn-2)

**Note**: PAM support is added via [build-time flags](https://docs.gitea.com/installation/install-from-source#build), and the official binaries provided do not have this enabled. PAM requires that the necessary libpam dynamic library be available and the necessary PAM development headers be accessible to the compiler.

**SMTP (Simple Mail Transfer Protocol)**[**​**](https://docs.gitea.com/administration/authentication#smtp-simple-mail-transfer-protocol)

This option allows Gitea to log in to an SMTP host as a Gitea user. To configure this, set the fields below:

* Authentication Name **(required)**
  + A name to assign to the new method of authorization.
* SMTP Authentication Type **(required)**
  + Type of authentication to use to connect to SMTP host, PLAIN or LOGIN.
* Host **(required)**
  + The address where the SMTP host can be reached.
  + Example: smtp.mydomain.com
* Port **(required)**
  + The port to use when connecting to the server.
  + Example: 587
* Allowed Domains
  + Restrict what domains can log in if using a public SMTP host or SMTP host with multiple domains.
  + Example: gitea.com,mydomain.com,mydomain2.com
* Force SMTPS
  + SMTPS will be used by default for connections to port 465, if you wish to use SMTPS for other ports. Set this value.
  + Otherwise if the server provides the STARTTLS extension this will be used.
* Skip TLS Verify
  + Disable TLS verify on authentication.
* This Authentication Source is Activated
  + Enable or disable this authentication source.

**FreeIPA**[**​**](https://docs.gitea.com/administration/authentication#freeipa)

* In order to log in to Gitea using FreeIPA credentials, a bind account needs to be created for Gitea:
* On the FreeIPA server, create a gitea.ldif file, replacing dc=example,dc=com with your DN, and provide an appropriately secure password:

dn: uid=gitea,cn=sysaccounts,cn=etc,dc=example,dc=com  
changetype: add  
objectclass: account  
objectclass: simplesecurityobject  
uid: gitea  
userPassword: secure password  
passwordExpirationTime: 20380119031407Z  
nsIdleTimeout: 0

* Import the LDIF (change localhost to an IPA server if needed). A prompt for Directory Manager password will be presented:

ldapmodify -h localhost -p 389 -x -D \  
"cn=Directory Manager" -W -f gitea.ldif

* Add an IPA group for gitea\_users :

ipa group-add --desc="Gitea Users" gitea\_users

* Note: For errors about IPA credentials, run kinit admin and provide the domain admin account password.
* Log in to Gitea as an Administrator and click on "Authentication" under Admin Panel. Then click Add New Source and fill in the details, changing all where appropriate.

**SPNEGO with SSPI (Kerberos/NTLM, for Windows only)**[**​**](https://docs.gitea.com/administration/authentication#spnego-with-sspi-kerberosntlm-for-windows-only)

Gitea supports SPNEGO single sign-on authentication (the scheme defined by RFC4559) for the web part of the server via the Security Support Provider Interface (SSPI) built in Windows. SSPI works only in Windows environments - when both the server and the clients are running Windows.

Before activating SSPI single sign-on authentication (SSO) you have to prepare your environment:

* Create a separate user account in active directory, under which the gitea.exe process will be running (eg. user under domain domain.local):
* Create a service principal name for the host where gitea.exe is running with class HTTP:
  + Start Command Prompt or PowerShell as a privileged domain user (eg. Domain Administrator)
  + Run the command below, replacing host.domain.local with the fully qualified domain name (FQDN) of the server where the web application will be running, and domain\user with the name of the account created in the previous step:

setspn -A HTTP/host.domain.local domain\user

* Sign in (*sign out if you were already signed in*) with the user created
* Make sure that ROOT\_URL in the [server] section of custom/conf/app.ini is the fully qualified domain name of the server where the web application will be running - the same you used when creating the service principal name (eg. host.domain.local)
* Start the web server (gitea.exe web)
* Enable SSPI authentication by adding an SPNEGO with SSPI authentication source in Site Administration -> Authentication Sources
* Sign in to a client computer in the same domain with any domain user (client computer, different from the server running gitea.exe)
* If you are using Chrome or Edge, add the URL of the web app to the Local intranet sites (Internet Options -> Security -> Local intranet -> Sites)
* Start Chrome or Edge and navigate to the FQDN URL of Gitea (eg. http://host.domain.local:3000)
* Click the Sign In button on the dashboard and choose SSPI to be automatically logged in with the same user that is currently logged on to the computer
* If it does not work, make sure that:
  + You are not running the web browser on the same server where Gitea is running. You should be running the web browser on a domain joined computer (client) that is different from the server. If both the client and server are running on the same computer NTLM will be preferred over Kerberos.
  + There is only one HTTP/... SPN for the host
  + The SPN contains only the hostname, without the port
  + You have added the URL of the web app to the Local intranet zone
  + The clocks of the server and client should not differ with more than 5 minutes (depends on group policy)
  + Integrated Windows Authentication should be enabled in Internet Explorer (under Advanced settings)

**Reverse Proxy**[**​**](https://docs.gitea.com/administration/authentication#reverse-proxy)

Gitea supports Reverse Proxy Header authentication, it will read headers as a trusted login user name or user email address. This hasn't been enabled by default, you can enable it with

[service]  
ENABLE\_REVERSE\_PROXY\_AUTHENTICATION = true

The default login user name is in the X-WEBAUTH-USER header, you can change it via changing REVERSE\_PROXY\_AUTHENTICATION\_USER in app.ini. If the user doesn't exist, you can enable automatic registration with ENABLE\_REVERSE\_PROXY\_AUTO\_REGISTRATION=true.

The default login user email is X-WEBAUTH-EMAIL, you can change it via changing REVERSE\_PROXY\_AUTHENTICATION\_EMAIL in app.ini, this could also be disabled with ENABLE\_REVERSE\_PROXY\_EMAIL

If set ENABLE\_REVERSE\_PROXY\_FULL\_NAME=true, a user full name expected in X-WEBAUTH-FULLNAME will be assigned to the user when auto creating the user. You can also change the header name with REVERSE\_PROXY\_AUTHENTICATION\_FULL\_NAME.

You can also limit the reverse proxy's IP address range with REVERSE\_PROXY\_TRUSTED\_PROXIES which default value is 127.0.0.0/8,::1/128. By REVERSE\_PROXY\_LIMIT, you can limit trusted proxies level.

You can enable the this authentication method for the API with

[service]  
ENABLE\_REVERSE\_PROXY\_AUTHENTICATION\_API = true

note

When this method is enabled for the API, the reverse proxy is responsible for handling CSRF protection.

**Footnotes**[**​**](https://docs.gitea.com/administration/authentication#footnote-label)

1. For example, using standard Linux log-in on Debian "Bullseye" use common-session-noninteractive - this value may be valid for other flavors of Debian including Ubuntu and Mint, consult your distribution's documentation. [↩](https://docs.gitea.com/administration/authentication#user-content-fnref-1)
2. **This is a required field for PAM**. Be aware: In the above example, the user will log into the Gitea web interface as gituser and not gituser@mail.com [↩](https://docs.gitea.com/administration/authentication#user-content-fnref-2)

**Environment variables**

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

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

GITEA\_CUSTOM=/home/gitea/custom ./gitea web

**From Go language**[**​**](https://docs.gitea.com/administration/environment-variables#from-go-language)

As Gitea is written in Go, it uses some variables that influence the behaviour of Go's runtime, such as:

* GOMEMLIMIT
* GOGC
* GOMAXPROCS
* GODEBUG

For documentation about each of the variables available, refer to the [official Go documentation on runtime environment variables](https://pkg.go.dev/runtime#hdr-Environment_Variables).

**Gitea files**[**​**](https://docs.gitea.com/administration/environment-variables#gitea-files)

* GITEA\_WORK\_DIR: Absolute path of working directory.
* GITEA\_CUSTOM: Gitea uses WorkPath/custom folder by default. Use this variable to change *custom* directory.

**Operating system specifics**[**​**](https://docs.gitea.com/administration/environment-variables#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**[**​**](https://docs.gitea.com/administration/environment-variables#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

**Miscellaneous**[**​**](https://docs.gitea.com/administration/environment-variables#miscellaneous)

* SKIP\_MINWINSVC: If set to 1, do not run as a service on Windows.

**Backup and Restore**

Gitea currently has a dump command that will save the installation to a ZIP file. This file can be unpacked and used to restore an instance.

**Backup Consistency**[**​**](https://docs.gitea.com/administration/backup-and-restore#backup-consistency)

To ensure the consistency of the Gitea instance, it must be shutdown during backup.

Gitea consists of a database, files and git repositories, all of which change when it is used. For instance, when a migration is in progress, a transaction is created in the database while the git repository is being copied over. If the backup happens in the middle of the migration, the git repository may be incomplete although the database claims otherwise because it was dumped afterwards. The only way to avoid such race conditions is by stopping the Gitea instance during the backups.

**Backup Command (dump)**[**​**](https://docs.gitea.com/administration/backup-and-restore#backup-command-dump)

Switch to the user running Gitea: su git. Run ./gitea dump -c /path/to/app.ini in the Gitea installation directory. There should be some output similar to the following:

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

Inside the gitea-dump-1482906742.zip file, will be the following:

* app.ini - Optional copy of configuration file if originally stored outside the default custom/ directory
* custom/ - All config or customization files in custom/.
* data/ - Data directory (APP\_DATA\_PATH), except sessions if you are using file session. This directory includes attachments, avatars, lfs, indexers, SQLite file if you are using SQLite.
* repos/ - Complete copy of the repository directory.
* gitea-db.sql - SQL dump of database
* log/ - Various logs. They are not needed for a recovery or migration.

Intermediate backup files are created in a temporary directory specified either with the --tempdir command-line parameter or the TMPDIR environment variable.

**Backup the database**[**​**](https://docs.gitea.com/administration/backup-and-restore#backup-the-database)

The SQL dump created by gitea dump uses XORM and Gitea admins may prefer to use the native the MySQL and PostgreSQL dump tools instead. There are still open issues when using XORM for dumping the database that may cause problems when attempting to restore it.

# mysql  
mysqldump -u$USER -p$PASS --database $DATABASE > gitea-db.sql  
# postgres  
pg\_dump -U $USER $DATABASE > gitea-db.sql

**Using Docker (dump)**[**​**](https://docs.gitea.com/administration/backup-and-restore#using-docker-dump)

There are a few caveats for using the dump command with Docker.

The command has to be executed with the RUN\_USER = <OS\_USERNAME> specified in gitea/conf/app.ini; and, for the zipping of the backup folder to occur without permission error the command docker exec must be executed inside of the --tempdir.

Example:

docker exec -u <OS\_USERNAME> -it -w <--tempdir> $(docker ps -qf 'name=^<NAME\_OF\_DOCKER\_CONTAINER>$') bash -c '/usr/local/bin/gitea dump -c </path/to/app.ini>'

\*Note: --tempdir refers to the temporary directory of the docker environment used by Gitea; if you have not specified a custom --tempdir, then Gitea uses /tmp or the TMPDIR environment variable of the docker container. For --tempdir adjust your docker exec command options accordingly.

The result should be a file, stored in the --tempdir specified, along the lines of: gitea-dump-1482906742.zip

**Restore Command (restore)**[**​**](https://docs.gitea.com/administration/backup-and-restore#restore-command-restore)

There is currently no support for a recovery command. It is a manual process that mostly involves moving files to their correct locations and restoring a database dump.

Example:

unzip gitea-dump-1610949662.zip  
cd gitea-dump-1610949662  
mv app.ini /etc/gitea/conf/app.ini  
mv data/\* /var/lib/gitea/data/  
mv log/\* /var/lib/gitea/log/  
mv repos/\* /var/lib/gitea/data/gitea-repositories/  
chown -R gitea:gitea /etc/gitea/conf/app.ini /var/lib/gitea  
  
# mysql  
mysql --default-character-set=utf8mb4 -u$USER -p$PASS $DATABASE <gitea-db.sql  
# sqlite3  
sqlite3 $DATABASE\_PATH <gitea-db.sql  
# postgres  
psql -U $USER -d $DATABASE < gitea-db.sql  
  
service gitea restart

Repository Git Hooks should be regenerated if installation method is changed (eg. binary -> Docker), or if Gitea is installed to a different directory than the previous installation.

With Gitea running, and from the directory Gitea's binary is located, execute: ./gitea admin regenerate hooks

This ensures that application and configuration file paths in repository Git Hooks are consistent and applicable to the current installation. If these paths are not updated, repository push actions will fail.

If you still have issues, consider running ./gitea doctor check to inspect possible errors (or run with --fix).

**Email setup**

Gitea has mailer functionality for sending transactional emails (such as registration confirmation). It can be configured to either use Sendmail (or compatible MTAs like Postfix and msmtp) or directly use SMTP server.

**Using Sendmail**[**​**](https://docs.gitea.com/administration/email-setup#using-sendmail)

Use sendmail command as mailer.

Note: For use in the official Gitea Docker image, please configure with the SMTP version (see the following section).

Note: For Internet-facing sites consult documentation of your MTA for instructions to send emails over TLS. Also set up SPF, DMARC, and DKIM DNS records to make emails sent be accepted as legitimate by various email providers.

app.ini

[mailer]  
ENABLED = true  
FROM = gitea@mydomain.com  
PROTOCOL = sendmail  
SENDMAIL\_PATH = /usr/sbin/sendmail  
SENDMAIL\_ARGS = "--" ; most "sendmail" programs take options, "--" will prevent an email address being interpreted as an option.

**Using SMTP**[**​**](https://docs.gitea.com/administration/email-setup#using-smtp)

Directly use SMTP server as relay. This option is useful if you don't want to set up MTA on your instance but you have an account at email provider.

app.ini

[mailer]  
ENABLED = true  
FROM = gitea@mydomain.com  
PROTOCOL = smtps  
SMTP\_ADDR = mail.mydomain.com  
SMTP\_PORT = 587  
USER = gitea@mydomain.com  
PASSWD = `password`

Restart Gitea for the configuration changes to take effect.

To send a test email to validate the settings, go to Gitea > Site Administration > Configuration > SMTP Mailer Configuration.

For the full list of options check the [Config Cheat Sheet](https://docs.gitea.com/administration/config-cheat-sheet)

Please note: authentication is only supported when the SMTP server communication is encrypted with TLS or HOST=localhost. TLS encryption can be through:

* STARTTLS (also known as Opportunistic TLS) via port 587. Initial connection is done over cleartext, but then be upgraded over TLS if the server supports it.
* SMTPS connection (SMTP over TLS) via the default port 465. Connection to the server use TLS from the beginning.
* Forced SMTPS connection with PROTOCOL=smtps. (These are both known as Implicit TLS.) This is due to protections imposed by the Go internal libraries against STRIPTLS attacks.

Note that Implicit TLS is recommended by [RFC8314](https://tools.ietf.org/html/rfc8314#section-3) since 2018.

**Gmail**[**​**](https://docs.gitea.com/administration/email-setup#gmail)

The following configuration should work with GMail's SMTP server:

app.ini

[mailer]  
ENABLED = true  
HOST = smtp.gmail.com:465 ; Remove this line for Gitea >= 1.18.0  
SMTP\_ADDR = smtp.gmail.com  
SMTP\_PORT = 465  
FROM = example.user@gmail.com  
USER = example.user  
PASSWD = `\*\*\*`  
PROTOCOL = smtps

Note that you'll need to create and use an [App password](https://support.google.com/accounts/answer/185833?hl=en) by enabling 2FA on your Google account. You won't be able to use your Google account password directly.

**ProtonMail**[**​**](https://docs.gitea.com/administration/email-setup#protonmail)

This feature is currently only available for select Proton for Business customers and those with Visionary and Family plans with custom domain addresses. See [ProtonMail's SMTP documentation](https://proton.me/support/smtp-submission) for more information. This limitation can be circumvented by using the ProtonMail Bridge application.

Note that emails sent using SMTP are not [end-to-end encrypted](https://proton.me/support/proton-mail-encryption-explained). However, they’re still stored with zero-access encryption like any other emails in your Proton Mail inbox.

The following configuration should work with ProtonMail's SMTP server:

1. In your browser (or desktop application), sign in to your Proton Mail account and select **Settings → All settings → Proton Mail → IMAP/SMTP → SMTP tokens**.
2. Click **Generate token**.
3. Enter the following details to create a new SMTP token:
   * **Token name**: Select a name for your token. This is for your reference only and does not affect the token's functionality.
   * **Email address**: Select one of your active custom domain addresses to pair with your token. Copy this email address and use it for the FROM and USER configuration in app.ini.
4. Click **Generate**.
5. Enter your Proton Mail Account password.

Your SMTP username and SMTP token (password) will be generated. You can now enter them as the USER and PASSWD in your app.ini configuration.

app.ini

[mailer]  
ENABLED = true  
FROM = example.user@customdomain.tld  
PROTOCOL = smtp+starttls  
SMTP\_ADDR = smtp.protonmail.ch  
SMTP\_PORT = 587  
USER = example.user@customdomain.tld  
PASSWD = `TOKEN`

After closing the popup, you will not be able to see this SMTP token (password) again for security reasons. You can always generate more tokens if you need to rotate passwords.

Note: Your Proton Mail login or mailbox passwords will not work with SMTP

# Git LFS setup

To use Gitea's built-in LFS support, you must update the app.ini file:

[server]  
*; Enables git-lfs support. true or false, default is false.*  
LFS\_START\_SERVER = true  
  
[lfs]  
*; Where your lfs files reside, default is data/lfs.*  
PATH = /home/gitea/data/lfs

note

LFS server support needs at least Git v2.1.2 installed on the server

**HTTPS setup**

**Using the built-in server**[**​**](https://docs.gitea.com/administration/https-setup#using-the-built-in-server)

Before you enable HTTPS, make sure that you have valid SSL/TLS certificates. You could use self-generated certificates for evaluation and testing. Please run gitea cert --host [HOST] to generate a self signed certificate.

If you are using Apache or nginx on the server, it's recommended to check the [reverse proxy guide](https://docs.gitea.com/administration/reverse-proxies).

To use Gitea's built-in HTTPS support, you must change your app.ini file:

[server]  
PROTOCOL = https  
ROOT\_URL = https://git.example.com:3000/  
HTTP\_PORT = 3000  
CERT\_FILE = cert.pem  
KEY\_FILE = key.pem

Note that if your certificate is signed by a third party certificate authority (i.e. not self-signed), then cert.pem should contain the certificate chain. The server certificate must be the first entry in cert.pem, followed by the intermediaries in order (if any). The root certificate does not have to be included because the connecting client must already have it in order to establish the trust relationship. To learn more about the config values, please checkout the [Config Cheat Sheet](https://docs.gitea.com/administration/config-cheat-sheet#server-server).

For the CERT\_FILE or KEY\_FILE field, the file path is relative to the GITEA\_CUSTOM environment variable when it is a relative path. It can be an absolute path as well.

**Setting up HTTP redirection**[**​**](https://docs.gitea.com/administration/https-setup#setting-up-http-redirection)

The Gitea server is only able to listen to one port; to redirect HTTP requests to the HTTPS port, you will need to enable the HTTP redirection service:

[server]  
REDIRECT\_OTHER\_PORT = true  
*; Port the redirection service should listen on*  
PORT\_TO\_REDIRECT = 3080

If you are using Docker, make sure that this port is configured in your docker-compose.yml file.

**Using ACME (Default: Let's Encrypt)**[**​**](https://docs.gitea.com/administration/https-setup#using-acme-default-lets-encrypt)

[ACME](https://tools.ietf.org/html/rfc8555) is a Certificate Authority standard protocol that allows you to automatically request and renew SSL/TLS certificates. [Let's Encrypt](https://letsencrypt.org/) is a free publicly trusted Certificate Authority server using this standard. Only HTTP-01 and TLS-ALPN-01 challenges are implemented. In order for ACME challenges to pass and verify your domain ownership, external traffic to the gitea domain on port 80 (HTTP-01) or port 443 (TLS-ALPN-01) has to be served by the gitea instance. Setting up [HTTP redirection](https://docs.gitea.com/administration/https-setup#setting-up-http-redirection) and port-forwards might be needed for external traffic to route correctly. Normal traffic to port 80 will otherwise be automatically redirected to HTTPS. **You must consent** to the ACME provider's terms of service (default Let's Encrypt's [terms of service](https://letsencrypt.org/documents/LE-SA-v1.2-November-15-2017.pdf)).

Minimum setup using the default Let's Encrypt:

[server]  
PROTOCOL=https  
DOMAIN=git.example.com  
ENABLE\_ACME=true  
ACME\_ACCEPTTOS=true  
ACME\_DIRECTORY=https  
*;; Email can be omitted here and provided manually at first run, after which it is cached*  
ACME\_EMAIL=email@example.com

Minimum setup using a [smallstep CA](https://github.com/smallstep/certificates), refer to [their tutorial](https://smallstep.com/docs/tutorials/acme-challenge) for more information.

[server]  
PROTOCOL=https  
DOMAIN=git.example.com  
ENABLE\_ACME=true  
ACME\_ACCEPTTOS=true  
ACME\_URL=https://ca.example.com/acme/acme/directory  
*;; Can be omitted if using the system's trust is preferred*  
*;ACME\_CA\_ROOT=/path/to/root\_ca.crt*  
ACME\_DIRECTORY=https  
ACME\_EMAIL=email@example.com

To learn more about the config values, please checkout the [Config Cheat Sheet](https://docs.gitea.com/administration/config-cheat-sheet#server-server).

**Using a reverse proxy**[**​**](https://docs.gitea.com/administration/https-setup#using-a-reverse-proxy)

Setup up your reverse proxy as shown in the [reverse proxy guide](https://docs.gitea.com/administration/reverse-proxies).

After that, enable HTTPS by following one of these guides:

* [nginx](https://nginx.org/en/docs/http/configuring_https_servers.html)
* [apache2/httpd](https://httpd.apache.org/docs/2.4/ssl/ssl_howto.html)
* [caddy](https://caddyserver.com/docs/tls)

Note: Enabling HTTPS only at the proxy level is referred as [TLS Termination Proxy](https://en.wikipedia.org/wiki/TLS_termination_proxy). The proxy server accepts incoming TLS connections, decrypts the contents, and passes the now unencrypted contents to Gitea. This is normally fine as long as both the proxy and Gitea instances are either on the same machine, or on different machines within private network (with the proxy is exposed to outside network). If your Gitea instance is separated from your proxy over a public network, or if you want full end-to-end encryption, you can also [enable HTTPS support directly in Gitea using built-in server](https://docs.gitea.com/administration/https-setup#using-the-built-in-server) and forward the connections over HTTPS instead.

# Fail2ban Setup

**Remember that fail2ban is powerful and can cause lots of issues if you do it incorrectly, so make sure to test this before relying on it so you don't lock yourself out.**

Gitea returns an HTTP 200 for bad logins in the web logs, but if you have logging options on in app.ini, then you should be able to go off of log/gitea.log, which gives you something like this on a bad authentication from the web or CLI using SSH or HTTP respectively:

2018/04/26 18:15:54 [I] Failed authentication attempt for user from xxx.xxx.xxx.xxx

2020/10/15 16:05:09 modules/ssh/ssh.go:143:publicKeyHandler() [W] Failed authentication attempt from xxx.xxx.xxx.xxx

(DEPRECATED: This may be a false positive as the user may still go on to correctly authenticate.)

2020/10/15 16:05:09 modules/ssh/ssh.go:155:publicKeyHandler() [W] Failed authentication attempt from xxx.xxx.xxx.xxx

(DEPRECATED: This may be a false positive as the user may still go on to correctly authenticate.)

2020/10/15 16:05:09 modules/ssh/ssh.go:198:publicKeyHandler() [W] Failed authentication attempt from xxx.xxx.xxx.xxx

(DEPRECATED: This may be a false positive as the user may still go on to correctly authenticate.)

2020/10/15 16:05:09 modules/ssh/ssh.go:213:publicKeyHandler() [W] Failed authentication attempt from xxx.xxx.xxx.xxx

(DEPRECATED: This may be a false positive as the user may still go on to correctly authenticate.)

2020/10/15 16:05:09 modules/ssh/ssh.go:227:publicKeyHandler() [W] Failed authentication attempt from xxx.xxx.xxx.xxx

(DEPRECATED: This may be a false positive as the user may still go on to correctly authenticate.)

2020/10/15 16:05:09 modules/ssh/ssh.go:249:sshConnectionFailed() [W] Failed authentication attempt from xxx.xxx.xxx.xxx

(From 1.15 this new message will available and doesn't have any of the false positive results that above messages from publicKeyHandler do. This will only be logged if the user has completely failed authentication.)

2020/10/15 16:08:44 ...s/context/context.go:204:HandleText() [E] invalid credentials from xxx.xxx.xxx.xxx

Add our filter in /etc/fail2ban/filter.d/gitea.conf:

*# gitea.conf*  
[Definition]  
failregex = .\*(Failed authentication attempt|invalid credentials|Attempted access of unknown user).\* from <HOST>  
ignoreregex =

Add our jail in /etc/fail2ban/jail.d/gitea.conf:

[gitea]  
enabled = true  
filter = gitea  
logpath = /var/lib/gitea/log/gitea.log  
maxretry = 10  
findtime = 3600  
bantime = 900  
action = iptables-allports

If you're using Docker, you'll also need to add an additional jail to handle the **FORWARD** chain in **iptables**. Configure it in /etc/fail2ban/jail.d/gitea-docker.conf:

[gitea-docker]  
enabled = true  
filter = gitea  
logpath = /var/lib/gitea/log/gitea.log  
maxretry = 10  
findtime = 3600  
bantime = 900  
action = iptables-allports[chain="FORWARD"]

Then simply run service fail2ban restart to apply your changes. You can check to see if fail2ban has accepted your configuration using service fail2ban status.

Make sure and read up on fail2ban and configure it to your needs, this bans someone for **15 minutes** (from all ports) when they fail authentication 10 times in an hour.

If you run Gitea behind a reverse proxy with Nginx (for example with Docker), you need to add this to your Nginx configuration so that IPs don't show up as 127.0.0.1:

proxy\_set\_header X-Real-IP $remote\_addr;

The security options in app.ini need to be adjusted to allow the interpretation of the headers as well as the list of IP addresses and networks that describe trusted proxy servers (See the [configuration cheat sheet](https://docs.gitea.com/administration/config-cheat-sheet#security-security) for more information).

REVERSE\_PROXY\_LIMIT = 1  
REVERSE\_PROXY\_TRUSTED\_PROXIES = 127.0.0.1/8 ; 172.17.0.0/16 for the docker default network

**Reverse Proxies**

**General configuration**[**​**](https://docs.gitea.com/administration/reverse-proxies#general-configuration)

1. Set [server] ROOT\_URL = https://git.example.com/ in your app.ini file.
2. Make the reverse-proxy pass https://git.example.com/foo to http://gitea:3000/foo.
3. Make sure the reverse-proxy does not decode the URI. The request https://git.example.com/a%2Fb should be passed as http://gitea:3000/a%2Fb.
4. Make sure Host and X-Fowarded-Proto headers are correctly passed to Gitea to make Gitea see the real URL being visited.

**Use a sub-path**[**​**](https://docs.gitea.com/administration/reverse-proxies#use-a-sub-path)

Usually it's **not recommended** to put Gitea in a sub-path, it's not widely used and may have some issues in rare cases.

To make Gitea work with a sub-path (eg: https://common.example.com/gitea/), there are some extra requirements besides the general configuration above:

1. Use [server] ROOT\_URL = https://common.example.com/gitea/ in your app.ini file.
2. Make the reverse-proxy pass https://common.example.com/gitea/foo to http://gitea:3000/foo.
3. The container registry requires a fixed sub-path /v2 at the root level which must be configured:
   * Make the reverse-proxy pass https://common.example.com/v2 to http://gitea:3000/v2.
   * Make sure the URI and headers are also correctly passed (see the general configuration above).

**Nginx**[**​**](https://docs.gitea.com/administration/reverse-proxies#nginx)

If you want Nginx to serve your Gitea instance, add the following server section to the http section of nginx.conf.

Make sure client\_max\_body\_size is large enough, otherwise there would be "413 Request Entity Too Large" error when uploading large files.

server {  
 ...  
 location / {  
 client\_max\_body\_size 512M;  
 proxy\_pass http://localhost:3000;  
 proxy\_set\_header Connection $http\_connection;  
 proxy\_set\_header Upgrade $http\_upgrade;  
 proxy\_set\_header Host $host;  
 proxy\_set\_header X-Real-IP $remote\_addr;  
 proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;  
 proxy\_set\_header X-Forwarded-Proto $scheme;  
 }  
}

**Nginx with a sub-path**[**​**](https://docs.gitea.com/administration/reverse-proxies#nginx-with-a-sub-path)

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 into the http section of nginx.conf:

server {  
 ...  
 location ~ ^/(gitea|v2)($|/) {  
 client\_max\_body\_size 512M;  
  
 # make nginx use unescaped URI, keep "%2F" as-is, remove the "/gitea" sub-path prefix, pass "/v2" as-is.  
 rewrite ^ $request\_uri;  
 rewrite ^/(gitea($|/))?(.\*) /$3 break;  
 proxy\_pass http://127.0.0.1:3000$uri;  
  
 # other common HTTP headers, see the "Nginx" config section above  
 proxy\_set\_header Connection $http\_connection;  
 proxy\_set\_header Upgrade $http\_upgrade;  
 proxy\_set\_header Host $host;  
 proxy\_set\_header X-Real-IP $remote\_addr;  
 proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;  
 proxy\_set\_header X-Forwarded-Proto $scheme;  
 }  
}

Then you **MUST** set something like [server] ROOT\_URL = http://git.example.com/gitea/ correctly in your configuration.

**Nginx and serve static resources directly**[**​**](https://docs.gitea.com/administration/reverse-proxies#nginx-and-serve-static-resources-directly)

We can tune the performance in splitting requests into categories static and dynamic.

CSS files, JavaScript files, images and web fonts are static content. The front page, a repository view or issue list is dynamic content.

Nginx can serve static resources directly and proxy only the dynamic requests to Gitea. Nginx is optimized for serving static content, while the proxying of large responses might be the opposite of that (see <https://serverfault.com/q/587386>).

Download a snapshot of the Gitea source repository to /path/to/gitea/. After this, run make frontend in the repository directory to generate the static resources. We are only interested in the public/ directory for this task, so you can delete the rest. (You will need to have [Node with npm](https://nodejs.org/en/download/) and make installed to generate the static resources)

Depending on the scale of your user base, you might want to split the traffic to two distinct servers, or use a cdn for the static files.

**Single node and single domain**[**​**](https://docs.gitea.com/administration/reverse-proxies#single-node-and-single-domain)

Set [server] STATIC\_URL\_PREFIX = /\_/static in your configuration.

server {  
 listen 80;  
 server\_name git.example.com;  
  
 location /\_/static/assets/ {  
 alias /path/to/gitea/public/;  
 }  
  
 location / {  
 proxy\_pass http://localhost:3000;  
 }  
}

**Two nodes and two domains**[**​**](https://docs.gitea.com/administration/reverse-proxies#two-nodes-and-two-domains)

Set [server] STATIC\_URL\_PREFIX = http://cdn.example.com/gitea in your configuration.

# application server running Gitea  
server {  
 listen 80;  
 server\_name git.example.com;  
  
 location / {  
 proxy\_pass http://localhost:3000;  
 }  
}

# static content delivery server  
server {  
 listen 80;  
 server\_name cdn.example.com;  
  
 location /gitea/ {  
 alias /path/to/gitea/public/;  
 }  
  
 location / {  
 return 404;  
 }  
}

**Apache HTTPD**[**​**](https://docs.gitea.com/administration/reverse-proxies#apache-httpd)

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

<VirtualHost \*:80>  
 ...  
 ProxyPreserveHost On  
 ProxyRequests off  
 AllowEncodedSlashes NoDecode  
 ProxyPass / http://localhost:3000/ nocanon  
 RequestHeader set "X-Forwarded-Proto" expr=%{REQUEST\_SCHEME}  
</VirtualHost>

Note: The following Apache HTTPD mods must be enabled: proxy, proxy\_http.

If you wish to use Let's Encrypt with webroot validation, add the line ProxyPass /.well-known ! before ProxyPass to disable proxying these requests to Gitea.

**Apache HTTPD with a sub-path**[**​**](https://docs.gitea.com/administration/reverse-proxies#apache-httpd-with-a-sub-path)

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):

<VirtualHost \*:80>  
 ...  
 <Proxy \*>  
 Order allow,deny  
 Allow from all  
 </Proxy>  
 AllowEncodedSlashes NoDecode  
 # Note: no trailing slash after either /git or port  
 ProxyPass /git http://localhost:3000 nocanon  
 ProxyPreserveHost On  
 RequestHeader set "X-Forwarded-Proto" expr=%{REQUEST\_SCHEME}  
</VirtualHost>

Then you **MUST** set something like [server] ROOT\_URL = http://git.example.com/git/ correctly in your configuration.

Note: The following Apache HTTPD mods must be enabled: proxy, proxy\_http.

**Caddy**[**​**](https://docs.gitea.com/administration/reverse-proxies#caddy)

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

git.example.com {  
 reverse\_proxy localhost:3000  
}

**Caddy with a sub-path**[**​**](https://docs.gitea.com/administration/reverse-proxies#caddy-with-a-sub-path)

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 your server block in your Caddyfile:

git.example.com {  
 route /git/\* {  
 uri strip\_prefix /git  
 reverse\_proxy localhost:3000  
 }  
}

Then set [server] ROOT\_URL = http://git.example.com/git/ in your configuration.

**IIS**[**​**](https://docs.gitea.com/administration/reverse-proxies#iis)

If you wish to run Gitea with IIS. You will need to setup IIS with URL Rewrite as reverse proxy.

1. Setup an empty website in IIS, named let's say, Gitea Proxy.
2. Follow the first two steps in [Microsoft's Technical Community Guide to Setup IIS with URL Rewrite](https://techcommunity.microsoft.com/t5/iis-support-blog/setup-iis-with-url-rewrite-as-a-reverse-proxy-for-real-world/ba-p/846222#M343). That is:

* Install Application Request Routing (ARR for short) either by using the Microsoft Web Platform Installer 5.1 (WebPI) or downloading the extension from [IIS.net](https://www.iis.net/downloads/microsoft/application-request-routing)
* Once the module is installed in IIS, you will see a new Icon in the IIS Administration Console called URL Rewrite.
* Open the IIS Manager Console and click on the Gitea Proxy Website from the tree view on the left. Select and double click the URL Rewrite Icon from the middle pane to load the URL Rewrite interface.
* Choose the Add Rule action from the right pane of the management console and select the Reverse Proxy Rule from the Inbound and Outbound Rules category.
* In the Inbound Rules section, set the server name to be the host that Gitea is running on with its port. e.g. if you are running Gitea on the localhost with port 3000, the following should work: 127.0.0.1:3000
* Enable SSL Offloading
* In the Outbound Rules, ensure Rewrite the domain names of the links in HTTP response is set and set the From: field as above and the To: to your external hostname, say: git.example.com
* Now edit the web.config for your website to match the following: (changing 127.0.0.1:3000 and git.example.com as appropriate)

*<?xml version="1.0" encoding="UTF-8"?>*  
<configuration>  
 <system.web>  
 <httpRuntime requestPathInvalidCharacters="" />  
 </system.web>  
 <system.webServer>  
 <security>  
 <requestFiltering>  
 <hiddenSegments>  
 <clear />  
 </hiddenSegments>  
 <denyUrlSequences>  
 <clear />  
 </denyUrlSequences>  
 <fileExtensions allowUnlisted="true">  
 <clear />  
 </fileExtensions>  
 </requestFiltering>  
 </security>  
 <rewrite>  
 <rules useOriginalURLEncoding="false">  
 <rule name="ReverseProxyInboundRule1" stopProcessing="true">  
 <match url="(.\*)" />  
 <action type="Rewrite" url="http://127.0.0.1:3000{UNENCODED\_URL}" />  
 <serverVariables>  
 <set name="HTTP\_X\_ORIGINAL\_ACCEPT\_ENCODING" value="HTTP\_ACCEPT\_ENCODING" />  
 <set name="HTTP\_ACCEPT\_ENCODING" value="" />  
 </serverVariables>  
 </rule>  
 </rules>  
 <outboundRules>  
 <rule name="ReverseProxyOutboundRule1" preCondition="ResponseIsHtml1">  
 *<!-- set the pattern correctly here - if you only want to accept http or https -->*  
 *<!-- change the pattern and the action value as appropriate -->*  
 <match filterByTags="A, Form, Img" pattern="^http(s)?://127.0.0.1:3000/(.\*)" />  
 <action type="Rewrite" value="http{R:1}://git.example.com/{R:2}" />  
 </rule>  
 <rule name="RestoreAcceptEncoding" preCondition="NeedsRestoringAcceptEncoding">  
 <match serverVariable="HTTP\_ACCEPT\_ENCODING" pattern="^(.\*)" />  
 <action type="Rewrite" value="{HTTP\_X\_ORIGINAL\_ACCEPT\_ENCODING}" />  
 </rule>  
 <preConditions>  
 <preCondition name="ResponseIsHtml1">  
 <add input="{RESPONSE\_CONTENT\_TYPE}" pattern="^text/html" />  
 </preCondition>  
 <preCondition name="NeedsRestoringAcceptEncoding">  
 <add input="{HTTP\_X\_ORIGINAL\_ACCEPT\_ENCODING}" pattern=".+" />  
 </preCondition>  
 </preConditions>  
 </outboundRules>  
 </rewrite>  
 <urlCompression doDynamicCompression="true" />  
 <handlers>  
 <clear />  
 <add name="StaticFile" path="\*" verb="\*" modules="StaticFileModule,DefaultDocumentModule,DirectoryListingModule" resourceType="Either" requireAccess="Read" />  
 </handlers>  
 *<!-- Map all extensions to the same MIME type, so all files can be*  
 *downloaded. -->*  
 <staticContent>  
 <clear />  
 <mimeMap fileExtension="\*" mimeType="application/octet-stream" />  
 </staticContent>  
 </system.webServer>  
</configuration>

**HAProxy**[**​**](https://docs.gitea.com/administration/reverse-proxies#haproxy)

If you want HAProxy to serve your Gitea instance, you can add the following to your HAProxy configuration

add an acl in the frontend section to redirect calls to gitea.example.com to the correct backend

frontend http-in  
 ...  
 acl acl\_gitea hdr(host) -i gitea.example.com  
 use\_backend gitea if acl\_gitea  
 ...

add the previously defined backend section

backend gitea  
 server localhost:3000 check

If you redirect the http content to https, the configuration work the same way, just remember that the connection between HAProxy and Gitea will be done via http so you do not have to enable https in Gitea's configuration.

**HAProxy with a sub-path**[**​**](https://docs.gitea.com/administration/reverse-proxies#haproxy-with-a-sub-path)

In case you already have a site, and you want Gitea to share the domain name, you can setup HAProxy to serve Gitea under a sub-path by adding the following to you HAProxy configuration:

frontend http-in  
 ...  
 acl acl\_gitea path\_beg /gitea  
 use\_backend gitea if acl\_gitea  
 ...

With that configuration <http://example.com/gitea/> will redirect to your Gitea instance.

then for the backend section

backend gitea  
 http-request replace-path /gitea\/?(.\*) \/\1  
 server localhost:3000 check

The added http-request will automatically add a trailing slash if needed and internally remove /gitea from the path to allow it to work correctly with Gitea by setting properly <http://example.com/gitea> as the root.

Then you **MUST** set something like [server] ROOT\_URL = http://example.com/gitea/ correctly in your configuration.

**Traefik**[**​**](https://docs.gitea.com/administration/reverse-proxies#traefik)

If you want traefik to serve your Gitea instance, you can add the following label section to your docker-compose.yaml (Assuming the provider is docker).

gitea:  
 image: gitea/gitea  
 ...  
 labels:  
 - "traefik.enable=true"  
 - "traefik.http.routers.gitea.rule=Host(`example.com`)"  
 - "traefik.http.services.gitea-websecure.loadbalancer.server.port=3000"

This config assumes that you are handling HTTPS on the traefik side and using HTTP between Gitea and traefik.

**Traefik with a sub-path**[**​**](https://docs.gitea.com/administration/reverse-proxies#traefik-with-a-sub-path)

In case you already have a site, and you want Gitea to share the domain name, you can setup Traefik to serve Gitea under a sub-path by adding the following to your docker-compose.yaml (Assuming the provider is docker) :

gitea:  
 image: gitea/gitea  
 ...  
 labels:  
 - "traefik.enable=true"  
 - "traefik.http.routers.gitea.rule=Host(`example.com`) && PathPrefix(`/gitea`)"  
 - "traefik.http.services.gitea-websecure.loadbalancer.server.port=3000"  
 - "traefik.http.middlewares.gitea-stripprefix.stripprefix.prefixes=/gitea"  
 - "traefik.http.routers.gitea.middlewares=gitea-stripprefix"

This config assumes that you are handling HTTPS on the traefik side and using HTTP between Gitea and traefik.

Then you **MUST** set something like [server] ROOT\_URL = http://example.com/gitea/ correctly in your configuration.

**Embedded data extraction tool**

Gitea's executable contains all the resources required to run: templates, images, style-sheets and translations. Any of them can be overridden by placing a replacement in a matching path inside the custom directory (see [Customizing Gitea](https://docs.gitea.com/administration/customizing-gitea)).

To obtain a copy of the embedded resources ready for editing, the embedded command from the CLI can be used from the OS shell interface.

note

The embedded data extraction tool is included in Gitea versions 1.12 and above.

**Listing resources**[**​**](https://docs.gitea.com/administration/cmd-embedded#listing-resources)

To list resources embedded in Gitea's executable, use the following syntax:

gitea embedded list [--include-vendored] [patterns...]

The --include-vendored flag makes the command include vendored files, which are normally excluded; that is, files from external libraries that are required for Gitea (e.g. [octicons](https://octicons.github.com/), etc).

A list of file search patterns can be provided. Gitea uses [gobwas/glob](https://github.com/gobwas/glob) for its glob syntax. Here are some examples:

* List all template files, in any virtual directory: \*\*.tmpl
* List all mail template files: templates/mail/\*\*.tmpl
* List all files inside public/assets/img: public/assets/img/\*\*

Don't forget to use quotes for the patterns, as spaces, \* and other characters might have a special meaning for your command shell.

If no pattern is provided, all files are listed.

**Example: Listing all embedded files**[**​**](https://docs.gitea.com/administration/cmd-embedded#example-listing-all-embedded-files)

Listing all embedded files with openid in their path:

$ gitea embedded list '\*\*openid\*\*'  
public/assets/img/auth/openid\_connect.svg  
public/assets/img/openid-16x16.png  
templates/user/auth/finalize\_openid.tmpl  
templates/user/auth/signin\_openid.tmpl  
templates/user/auth/signup\_openid\_connect.tmpl  
templates/user/auth/signup\_openid\_navbar.tmpl  
templates/user/auth/signup\_openid\_register.tmpl  
templates/user/settings/security\_openid.tmpl

**Extracting resources**[**​**](https://docs.gitea.com/administration/cmd-embedded#extracting-resources)

To extract resources embedded in Gitea's executable, use the following syntax:

gitea [--config {file}] embedded extract [--destination {dir}|--custom] [--overwrite|--rename] [--include-vendored] {patterns...}

The --config option tells Gitea the location of the app.ini configuration file if it's not in its default location. This option is only used with the --custom flag.

The --destination option tells Gitea the directory where the files must be extracted to. The default is the current directory.

The --custom flag tells Gitea to extract the files directly into the custom directory. For this to work, the command needs to know the location of the app.ini configuration file (--config) and, depending of the configuration, be ran from the directory where Gitea normally starts. See [Customizing Gitea](https://docs.gitea.com/administration/customizing-gitea) for details.

The --overwrite flag allows any existing files in the destination directory to be overwritten.

The --rename flag tells Gitea to rename any existing files in the destination directory as filename.bak. Previous .bak files are overwritten.

At least one file search pattern must be provided; see list subcomand above for pattern syntax and examples.

**Important notice**[**​**](https://docs.gitea.com/administration/cmd-embedded#important-notice)

Make sure to **only extract those files that require customization**. Files that are present in the custom directory are not upgraded by Gitea's upgrade process. When Gitea is upgraded to a new version (by replacing the executable), many of the embedded files will suffer changes. Gitea will honor and use any files found in the custom directory, even if they are old and incompatible.

**Example: Extracting mail templates**[**​**](https://docs.gitea.com/administration/cmd-embedded#example-extracting-mail-templates)

Extracting mail templates to a temporary directory:

$ mkdir tempdir  
$ gitea embedded extract --destination tempdir 'templates/mail/\*\*.tmpl'  
Extracting to tempdir:  
tempdir/templates/mail/auth/activate.tmpl  
tempdir/templates/mail/auth/activate\_email.tmpl  
tempdir/templates/mail/auth/register\_notify.tmpl  
tempdir/templates/mail/auth/reset\_passwd.tmpl  
tempdir/templates/mail/issue/assigned.tmpl  
tempdir/templates/mail/issue/default.tmpl  
tempdir/templates/mail/notify/collaborator.tmpl

**What is Gitea?**

Gitea is a painless, self-hosted, all-in-one software development service. It includes Git hosting, code review, team collaboration, package registry, and CI/CD. It is similar to GitHub, Bitbucket and GitLab.

Gitea was originally forked from [Gogs](https://gogs.io/) and almost all the code has been changed. See the [Gitea Announcement](https://blog.gitea.com/welcome-to-gitea/) blog post to read about the justification for a fork.

warning

Gitea does not sent or cherry-picked commits from upstream, so there is no guarantee it will work if you upgrade from Gogs to Gitea. The recommended method is to migrate repositories from Gogs to Gitea.

**Purpose**[**​**](https://docs.gitea.com/#purpose)

The goal of this project is to provide the easiest, fastest, and most painless way of setting up a self-hosted Git service.

With Go, this can be done platform-independently across **all platforms** which Go supports, including Linux, macOS, and Windows, on x86, amd64, ARM and PowerPC architectures. You can try it out using [the online demo](https://demo.gitea.com/).

**Features**[**​**](https://docs.gitea.com/#features)

* **Code Hosting**

Gitea supports creating and managing repositories, browsing commit history and code files, reviewing and merging code submissions, managing collaborators, handling branches, and more. It also supports many common Git features such as tags, Cherry-pick, hooks, integrated collaboration tools, and more.

* **Lightweight and Fast**

One of Gitea's design goals is to be lightweight and fast in response. Unlike some large code hosting platforms, it remains lean, performs well in terms of speed, and is suitable for resource-limited server environments. Due to its lightweight design, Gitea has relatively low resource consumption and performs well in resource-constrained environments.

* **Easy Deployment and Maintenance**

It can be easily deployed on various servers without complex configurations or dependencies. This makes it convenient for individual developers or small teams to set up and manage their own Git services.

* **Security**

Gitea places a strong emphasis on security, offering features such as user permission management, access control lists, and more to ensure the security of code and data.

* **Code Review**

Code review supports both the Pull Request workflow and AGit workflow. Reviewers can browse code online and provide review comments or feedback. Submitters can receive review comments and respond or modify code online. Code reviews can help individuals and organizations enhance code quality.

* **CI/CD**

Gitea Actions supports CI/CD functionality, compatible with GitHub Actions. Users can write workflows in familiar YAML format and reuse a variety of existing Actions plugins. Actions plugins support downloading from any Git website.

* **Project Management**

Gitea tracks project requirements, features, and bugs through columns and issues. Issues support features like branches, tags, milestones, assignments, time tracking, due dates, dependencies, and more.

* **Artifact Repository**

Gitea supports over 20 different types of public or private software package management, including Cargo, Chef, Composer, Conan, Conda, Container, Helm, Maven, npm, NuGet, Pub, PyPI, RubyGems, Vagrant, and more.

* **Open Source Community Support**

Gitea is an open-source project based on the MIT license. It has an active open-source community that continuously develops and improves the platform. The project also actively welcomes community contributions, ensuring updates and innovation.

* **Multilingual Support**

Gitea provides interfaces in multiple languages, catering to users globally and promoting internationalization and localization.

For more detailed information, please refer to: <https://docs.gitea.com/installation/comparison#general-features>

**System Requirements**[**​**](https://docs.gitea.com/#system-requirements)

* A Raspberry Pi 3 is powerful enough to run Gitea for small workloads.
* 2 CPU cores and 1GB RAM is typically sufficient for small teams/projects.
* Gitea should be run with a dedicated non-root system account on UNIX-type systems.
  + Note: Gitea manages the ~/.ssh/authorized\_keys file. Running Gitea as a regular user could break that user's ability to log in.
* [Git](https://git-scm.com/) version 2.0.0 or later is required.
  + [Git Large File Storage](https://git-lfs.github.com/) will be available if enabled and if your Git version is >= 2.1.2
  + Git commit-graph rendering will be enabled automatically if your Git version is >= 2.18

**Browser Support**[**​**](https://docs.gitea.com/#browser-support)

* Last 2 versions of Chrome, Firefox, Safari and Edge
* Firefox ESR

**Components**[**​**](https://docs.gitea.com/#components)

* Web server framework: [Chi](http://github.com/go-chi/chi)
* ORM: [XORM](https://xorm.io/)
* UI frameworks:
  + [jQuery](https://jquery.com/)
  + [Fomantic UI](https://fomantic-ui.com/)
  + [Vue3](https://vuejs.org/)
  + and various components (see package.json)
* Editors:
  + [CodeMirror](https://codemirror.net/)
  + [EasyMDE](https://github.com/Ionaru/easy-markdown-editor)
  + [Monaco Editor](https://microsoft.github.io/monaco-editor)
* Database drivers:
  + [github.com/go-sql-driver/mysql](https://github.com/go-sql-driver/mysql)
  + [github.com/lib/pq](https://github.com/lib/pq)
  + [github.com/mattn/go-sqlite3](https://github.com/mattn/go-sqlite3)
  + [github.com/denisenkom/go-mssqldb](https://github.com/denisenkom/go-mssqldb)

**Integrated support**[**​**](https://docs.gitea.com/#integrated-support)

Please visit [Awesome Gitea](https://gitea.com/gitea/awesome-gitea/) to get more third-party integrated support

**Repository indexer**

**Builtin repository code search without indexer**[**​**](https://docs.gitea.com/administration/repo-indexer#builtin-repository-code-search-without-indexer)

Users could do repository-level code search without setting up a repository indexer. The builtin code search is based on the git grep command, which is fast and efficient for small repositories. Better code search support could be achieved by setting up the repository indexer.

**Setting up the repository indexer**[**​**](https://docs.gitea.com/administration/repo-indexer#setting-up-the-repository-indexer)

Gitea can search through the files of the repositories by enabling this function in your [app.ini](https://docs.gitea.com/administration/config-cheat-sheet):

[indexer]  
*; ...*  
REPO\_INDEXER\_ENABLED = true  
REPO\_INDEXER\_PATH = indexers/repos.bleve  
MAX\_FILE\_SIZE = 1048576  
REPO\_INDEXER\_INCLUDE =  
REPO\_INDEXER\_EXCLUDE = resources/bin/\*\*

Please bear in mind that indexing the contents can consume a lot of system resources, especially when the index is created for the first time or globally updated (e.g. after upgrading Gitea).

**Choosing the files for indexing by size**[**​**](https://docs.gitea.com/administration/repo-indexer#choosing-the-files-for-indexing-by-size)

The MAX\_FILE\_SIZE option will make the indexer skip all files larger than the specified value.

**Choosing the files for indexing by path**[**​**](https://docs.gitea.com/administration/repo-indexer#choosing-the-files-for-indexing-by-path)

Gitea applies glob pattern matching from the [gobwas/glob library](https://github.com/gobwas/glob) to choose which files will be included in the index.

Limiting the list of files prevents the indexes from becoming polluted with derived or irrelevant files (e.g. lss, sym, map, etc.), so the search results are more relevant. It can also help reduce the index size.

REPO\_INDEXER\_EXCLUDE\_VENDORED (default: true) excludes vendored files from index.

REPO\_INDEXER\_INCLUDE (default: empty) is a comma separated list of glob patterns to **include** in the index. An empty list means "*include all files*". REPO\_INDEXER\_EXCLUDE (default: empty) is a comma separated list of glob patterns to **exclude** from the index. Files that match this list will not be indexed. REPO\_INDEXER\_EXCLUDE takes precedence over REPO\_INDEXER\_INCLUDE.

Pattern matching works as follows:

* To match all files with a .txt extension no matter what directory, use \*\*.txt.
* To match all files with a .txt extension *only at the root level of the repository*, use \*.txt.
* To match all files inside resources/bin and below, use resources/bin/\*\*.
* To match all files *immediately inside* resources/bin, use resources/bin/\*.
* To match all files named Makefile, use \*\*Makefile.
* Matching a directory has no effect; the pattern resources/bin will not include/exclude files inside that directory; resources/bin/\*\* will.
* All files and patterns are normalized to lower case, so \*\*Makefile, \*\*makefile and \*\*MAKEFILE are equivalent.