

Upgrade and migration

- [Upgrade guide](#)
- [Gitea migration](#)

Upgrade guide

This guide helps Forgejo admins perform upgrades safely and provides guidance to troubleshoot problems. It also covers upgrades from Gitea back to version 1.2.0.

Release life cycle

Each Forgejo release undergoes the following states:

- **Stable** (latest release): receives full support, bugfixes and security fixes for three months, plus a two week window after the next stable release is made available.
- **Long term support (LTS)** (the version published the first quarter of every year): receives only critical bugfixes and security support for one year and three months.
- **Experimental** (current development version): receives new features, should not be used in production.

To be notified in advance of security releases, watch or subscribe to the RSS feed of the [security-announcements repository](#). The details of the vulnerability will not be revealed, only the expected release date, for administrators to plan ahead and better secure their instance.

Semantic version compliance

Forgejo is compliant with [semantic versioning](#) as of 7.0.0. In a nutshell it means that there is no breaking change unless the first number changes (e.g. when 8.0.0 is published it will contain breaking changes compared to 7.0.0). The release notes document those breaking changes and they may require manual intervention depending on the Forgejo installation.

In versions prior to 7.0.0, the releases 1.19, 1.20 and 1.21 all contained breaking changes and the versioning scheme was not compliant with semantic versioning.

Backup

To be safe, make sure to perform a full backup before upgrading. It is a requirement when upgrading to a new stable release (going from v1.20 to v1.21 for instance) but is also a good precaution when installing a minor or patch release (going from v7.0.0 to v7.0.1 for instance).

The reliable way to perform a backup is with a synchronized point-in-time snapshot of all the storage used by Forgejo.

For instance, if everything (SQLite database + repositories etc.) is on a single QCOW2 disk attached to a virtual machine, a [gemu snapshot](#) guarantees the backup is consistent and can be done while the Forgejo server is running.

However, if Forgejo uses a S3 storage for attachments with a PostgreSQL database, stores Git repositories on a remote file system and queues in a Redis server, the only way to ensure a consistent state is to shutdown Forgejo during the backup.

In the simplest case where everything is on a single file system and if the instance is not busy (no mirrors, no users), the backup can be done with:

- `forgejo dump` to collect everything into one zip file
- `psql/mysql dump`. Although the zip file created by `forgejo dump` contains a copy of the database it has serious long standing open bugs that may introduce problems when re-injecting the SQL dump in a new database. Note that there is no need to dump SQLite because the database itself is included in the zip file already.

Verify Forgejo works

It is **critical** to verify that Forgejo works very carefully. Restoring the backup done before the upgrade is easy and does not lose any information. But if a problem is discovered days or weeks after the upgrade, it will not be an option and fixing it on a live Forgejo instance will be much more challenging.

- Run `forgejo doctor check --all --log-file /tmp/doctor.log` and make sure it does not report any problem.
- Manually verify via the web interface. Making a checklist of a typical use case is a good way to not miss anything.
- If there is a problem of any kind, increase the log level and take a look at the logs. If you cannot figure out what the problem is, ask for help [in the Forgejo chatroom](#) before trying to fix it.

Preparing an upgrade from Gitea

Upgrades to Forgejo are transparently supported [up to Gitea v1.22 included](#), in two steps:

- Upgrade from any Gitea version up to and including v1.22.x to Forgejo v10.0.x

- Upgrade from Forgejo v10.0.x to any Forgejo version greater than v10.

Preparing the Forgejo upgrade

- Manually analyze (reading the patches to the sources of the template directory) and update the customized CSS / content.
- Do not use `forgejo help` to figure out the location of `CustomPath`, look at the configuration tab of the `Site administration` panel when logged in as an admin.
- `forgejo manager flush-queues`. If it times out, run it again with a more generous `--timeout` argument. It is important because the queues contain serialized data that is not guaranteed to be backward compatible between versions.

Note: Forgejo installations using Docker requires [docker >= 20.10.6](#) otherwise mysterious problems will happen (mysterious in the sense that the problem will be about something unrelated to the Docker version").

Performing the upgrade

- Read the [list of things to know when upgrading from a specific version](#) and either verify that your current version is not affected; or apply the notes appropriately
- Otherwise, upgrade straight to the latest released Forgejo version by replacing the binary or container image (the upgrade procedure will take care of migrations)
- Verify Forgejo works

Troubleshooting

- Increase the log level. By default, the logs are outputted to console. If you need to collect logs from files, you could copy the following config into your `app.ini` file (remove all other [log] sections), then you can find the *.log files in Forgejo's log directory.

```
; To show all SQL logs, you can also set LOG_SQL=true in the [database] section
[log]
LEVEL=debug
MODE=console,file
ROUTER=console,file
XORM=console,file
ENABLE_XORM_LOG=true
```

```
FILE_NAME=forgejo.log
[log.file.router]
FILE_NAME=router.log
[log.file.xorm]
FILE_NAME=xorm.log
```

- If the upgrade from version x.y to version x.y+2 fails and there is a need to narrow down the problem, try upgrading to the latest version of each series and verify it works. It will show which series causes the issue and help debug the problem. For instance, if upgrading from Forgejo 1.19.3-0 to Forgejo 1.21.6-0 does not work:
 - Upgrade from Forgejo 1.19.3-0 to Forgejo 1.19.4-0 (the last version of the 1.19 series) and verify Forgejo works.
 - Upgrade to Forgejo 1.20.6-0 (the last version of the 1.20 series) and verify Forgejo works.

Unexpected database version

The database version is stored in the database and used to prevent an accidental downgrade. For instance, if a running `Forgejo v1.20` instance is downgraded to `Forgejo v1.19`, it will refuse to start because it would damage the content of the database.

Unexplained upgrade failures & workarounds

- All versions
 - Symptom: [blank / 500 page after login when running SQLite](#)
 - Workaround: upgrade to [Forgejo 1.19.3-0 or greater](#) and run `gitea doctor check --all --fix`

When upgrading from ... (known problematic versions or upgrade paths)

A list of things to be aware depending on which version you have installed. The list is sorted by Forgejo and Gitea versions. Upgrades from Gogs have [an extra section below](#). Please read them if

you originally installed Gogs, even if you already migrated to Gitea.

- Any version before Forgejo v1.20.3-0
 - verify the `app.ini` file does not contain problematic `[storage*]` sections [as explained in the v1.20.3-0 blog post](#)
- From v1.19.x to a version greater or equal to than [1.20](#)
 - The [tokens](#) were refactored in a way that does not guarantee their scope will be preserved. They may be larger or narrower and the only way to be sure that the intended scope is preserved is to re-create the token.
- Any version before [Gitea 1.17](#)
 - preserve a custom gitconfig: [episode 1](#), [episode 2](#)
- From Gitea 1.15.0: [Also see the release notes of 1.15.2](#)
 - Unfortunately following the release of 1.15.1 it has become apparent that there is an issue with upgrading from 1.15.0 to 1.15.1 due to problem with a table constraint that was unexpectedly automatically dropped. Users who upgraded straight from 1.14.x to 1.15.1 are not affected and users who upgraded from 1.15.0 to 1.15.1 can fix the problem using `gitea doctor recreate-table issue_index` or upgrade to 1.15.2.
- Between [1.13.0](#) and [1.13.3](#)
 - Password hashing algorithm default has changed back to pbkdf2 from argon2. ([#14673](#))
- [Gitea 1.13.0](#)
 - The Webhook shared secret inside the webhook payload has been deprecated and will be removed in 1.14.0: <https://github.com/go-gitea/gitea/issues/11755> please use the secret header that uses an hmac signature to validate the webhook payload.
 - Git hooks now default to `off`! ([#13058](#)) In your config, you can check the security section for `DISABLE_GIT_HOOKS`. To enable them again, you must set the setting to `false`.
- Do not use / skip on upgrade: Gitea v1.11.2 (now replaced by 1.11.3) was mistakenly compiled with Go 1.14, which Gitea is not currently fully tested with and it's known to cause [a few issues](#).
- Do not use / skip on upgrade: Gitea v1.10.5 (replaced by 1.10.6 if you need to keep using the 1.10 branch) was incorrectly tagged, and was in fact a snapshot of the development branch (1.12-dev). It was also compiled with Go 1.14.
- Do not use / skip on upgrade: Gitea v1.10.0, v1.10.1, v1.10.2, v1.10.3, v1.10.4, v1.11.0, and v1.11.1 **do not use** any of these versions, as a bug in the upgrade process will delete attachments from the releases on your repositories.
- From a Gitea version [lower than 1.6](#) and greater or equal to 1.2.0, proceed as follows:
 - Upgrade to 1.2.3 and manually verify it runs
 - Upgrade to 1.4.3 and manually verify it runs
 - Upgrade to 1.5.3 and manually verify it runs
 - Upgrade to 1.6.4 and manually verify it runs

Known problems when upgrading from Gogs:

- From an instance originally installed with Gogs, even if migrated later to Gitea or Forgejo because database columns are not automatically removed by migrations.
 - Drop the `created` column from the `issue` table while Forgejo is not running. It will be re-created:
 - SQLite `ALTER TABLE `issue` DROP `created``
 - MySQL/MariaDB/PostgreSQL `ALTER TABLE `issue` DROP COLUMN `created``
- Gogs from before September 2015 migrated to Forgejo v1.18 may have a [dangling pull_repo table](#) and the corresponding pull requests will be removed by `gitea doctor check --fix --all`.

Gitea migration

This guide describes how to go from Gitea to Forgejo.

Upgrades are supported [up to Gitea v1.22 included](#), in two steps:

- Upgrade from any Gitea version up to and including v1.22.x to Forgejo v10.0.x.
- Upgrade from Forgejo v10.0.x to any Forgejo version greater than v10.

Here's what to expect, and some helpful tips:

- First of all, ensure you have good backups. Ideally, shut down Gitea and do a backup of your database and files (Git repositories, attachments etc), or ideally of the whole server. This will make it a lot more relaxed to fiddle with the setup.
- Forgejo will run automatic migrations in the database to convert some structures.
- There should be no automatic modification of the app.ini related to this upgrade, but it is generally recommended to let Forgejo write to the config file because it is used to store some secrets.

Native (Package Manager) installation

If Gitea is installed via a package manager:

Arch Linux

Arch Linux provides package for both Gitea and Forgejo, but they install to different locations and run as different users. Therefore, a small amount of work is needed to transition from one to the other.

First step is to install the Forgejo package, and stop the Gitea service.

```
$ pacman -S forgejo
$ systemctl stop gitea
```

Configuration migration

Now, we can copy the Gitea configuration over the default Forgejo one. We need to set the group owner and group write permission so Forgejo can write to it.

```
$ cp /etc/gitea/app.ini /etc/forgejo/app.ini
$ chown root:forgejo /etc/forgejo/app.ini
$ chmod g+w /etc/forgejo/app.ini
```

Data migration

If we want to leave a copy of the Gitea data untouched (recommend in case of mistakes) the next step is to copy the existing data to the Forgejo user's home directory. We need to change the ownership of the files to the `forgejo` user and group too:

```
$ cp -r /var/lib/gitea/* /var/lib/forgejo
$ chown -R forgejo: /var/lib/forgejo/*
```

Finally, we need to update the configuration, which currently points to `/var/lib/gitea`, to point to the new location. Edit `/etc/forgejo/app.ini` and replace all instances of `/var/lib/gitea` with `/var/lib/forgejo`.

Data migration (no copy)

If you're not concerned about being able to roll back to your Gitea installation, you can simply change the owner of the files in `/var/lib/gitea`. There's then no need to change the configuration.

```
$ chown -R forgejo: /var/lib/gitea/*
```

Starting Forgejo

Whichever data migration you've chosen, it's now time to start the Forgejo service.

```
$ systemctl start forgejo
```

Check to see that the service has started with no issues.

```
$ systemctl status forgejo
```

...and set the service to start automatically.

```
$ sytemctl enable forgejo
```

Finally, if you're happy with everything, you can uninstall Gitea.

```
$ pacman -R gitea
```

Containerized (docker, podman, etc) installation

If gitea was running as a container, it makes sense to run forgejo the same way. If all goes well, forgejo 10.x should act as a drop-in replacement for gitea 1.22.x. Pay special attention to paths and environment variables.

If you were running a rootless gitea container, a rootless forgejo container should work similarly. The data should be owned by user 1000 and group 1000.

Configuring paths

The main configuration file is `app.ini`. This file controls the locations of all other data. If your `app.ini` specifies the locations of things, those things should continue to work. For unspecified things, pay close attention to the [configuration cheat-sheet](#) as the defaults may have changed.

Environment variables

The location of `app.ini` is currently specified by the `GITEA_APP_INI` environment variable. If unset, the default is `/var/lib/gitea/custom/conf/app.ini`. If the file is not present, `/etc/gitea/app.ini` is used as a fallback.

These variables are named with a "GITEA*" prefix for compatibility reasons. They may change in the future. It is recommended to pass two copies of these variables to your container: one with the prefix "GITEA*", and again with the prefix "FORGEJO_", to remain compatible with future versions of Forgejo.

Troubleshooting

Here are some common problems and how to solve them. Contributions are welcome.

Missing favicon or logo

Files from `public/assets/img/` must live in `custom/` now; otherwise, they will have no effect.

See [here](#) for details of where these things should live. Also see [here](#) for the necessary warnings and disclaimers.

The browser caches images, so after changing it, it may take some refreshes or cache-clears to see the change.

404 error when downloading a package

Check (and double-check) your configured paths. Refer to the [storage documentation](#), and double-check what the configured path is relative to.

Action fails to run with "repository not found"

If the "Set up job" section fails with "repository not found", it's probably trying to pull the action from the wrong place.

Workflow tasks defined with `uses:` and a relative URL are pulled from a mirror. Gitea and Forgejo have their own mirrors. Gitea mirrors some things that Forgejo does not, and vice versa.

To solve this, you can change the default mirror, as described [here](#), or you can change the `uses:` field to specify a full URL.

For example, if you trust that the upstream github project will remain compatible and sane:

```
uses: sammcj/dotenv-output-action@main
```

might become:

```
uses: https://github.com/sammcj/dotenv-output-action@main
```