Authenticating over HTTPS against git.wur.nl with two-factor authentication enabled

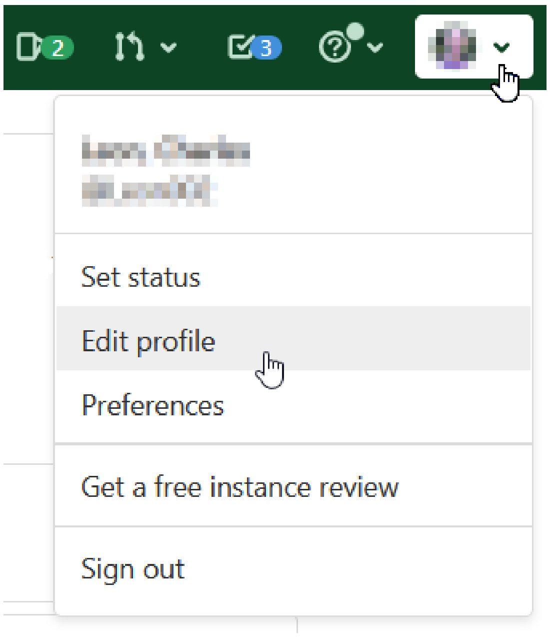
We recommend to access your repository using a SSH key instead of username and password over https. SSH keys are harder to guess than a password and can easily be revoked if they become compromised.

⚠ If you have stored your personal access token in the remote url in the .git/config on **a shared filesystem**, for example on the HPC or OneDrive shares, another user has access to your repository with your credentials (spoofing).
Solution: Revoke this personal access token in WUR GitLab, remove the personal access token from the remote url, create a new personal token and use the Git Credential Helper (see below).

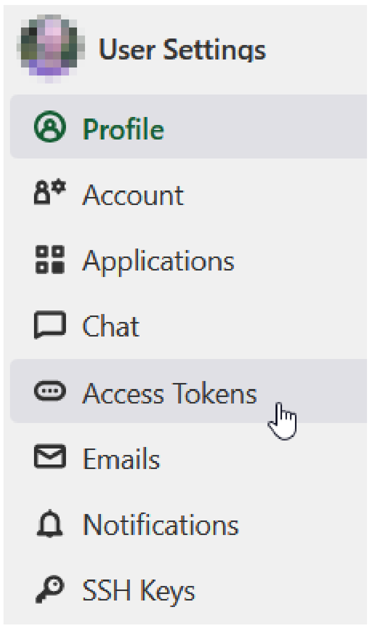
If you want to authenticate over HTTPS, you have to authenticate with a personal access token in place of your password when two-factor authentication is enabled.

Creating a personal access token

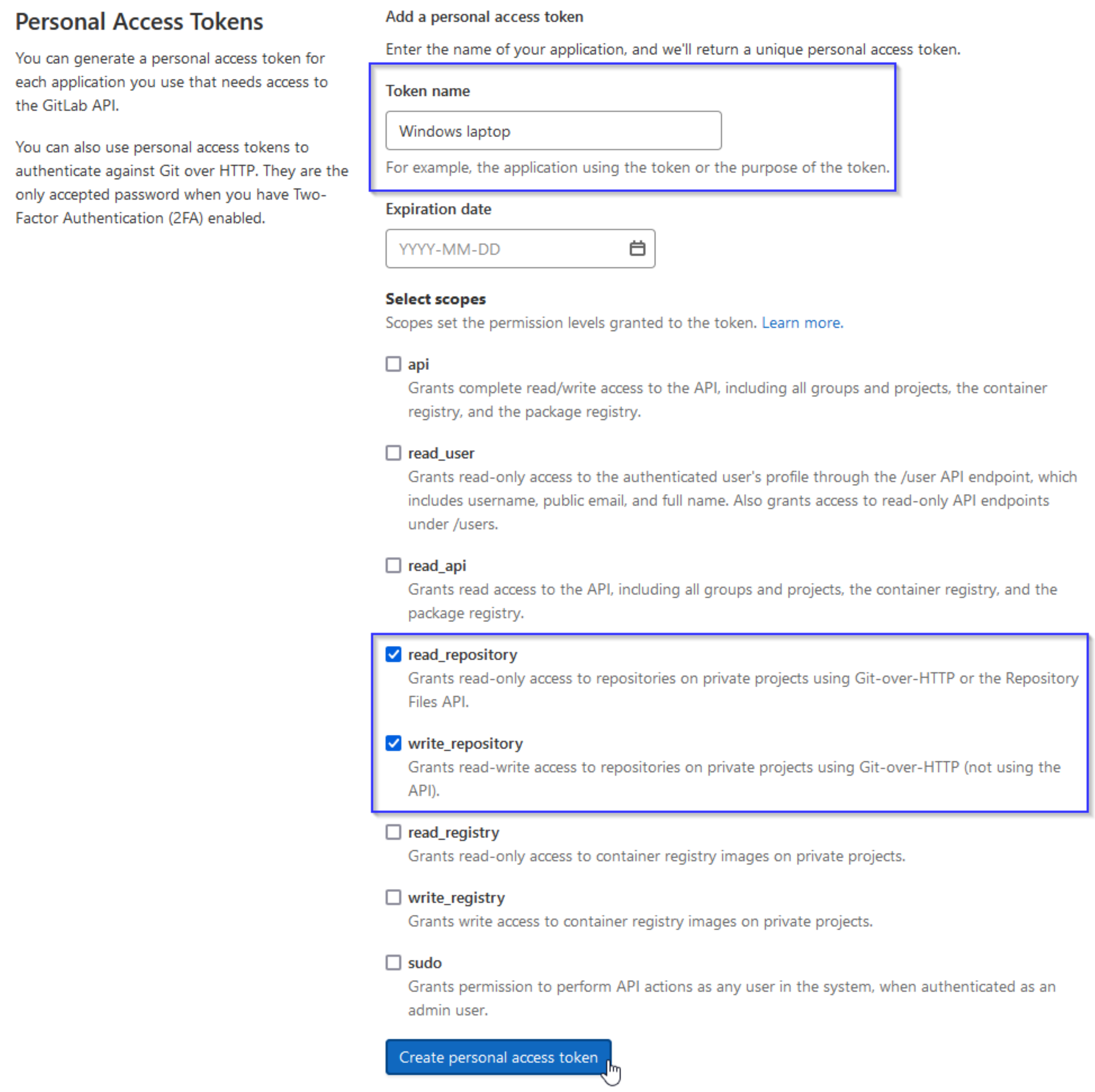
1. Navigate directly to User Settings > Access Tokens or by selecting the option Edit profile below your avatar



2. Click Access Tokens at the left side of the screen



3. In the form Personal Access Tokens, fill in a token name and tick the boxes read_repository and write_repository. Press the button Create personal access token.



Personal Access Tokens

You can generate a personal access token for each application you use that needs access to the GitLab API.

You can also use personal access tokens to authenticate against Git over HTTP. They are the only accepted password when you have Two-Factor Authentication (2FA) enabled.

Add a personal access token

Enter the name of your application, and we'll return a unique personal access token.

Token name

Windows laptop

For example, the application using the token or the purpose of the token.

Expiration date

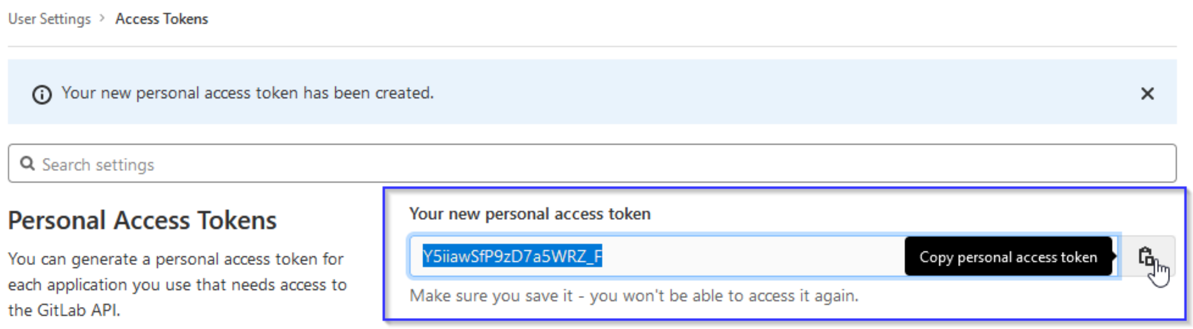
YYYY-MM-DD

Select scopes

Scopes set the permission levels granted to the token. Learn more.

- [ ] api
  Grants complete read/write access to the API, including all groups and projects, the container registry, and the package registry.
- [ ] read_user
  Grants read-only access to the authenticated user's profile through the /user API endpoint, which includes username, public email, and full name. Also grants access to read-only API endpoints under /users.
- [ ] read_api
  Grants read access to the API, including all groups and projects, the container registry, and the package registry.
- [✓] read_repository
  Grants read-only access to repositories on private projects using Git-over-HTTP or the Repository Files API.
- [✓] write_repository
  Grants read-write access to repositories on private projects using Git-over-HTTP (not using the API).
- [ ] read_registry
  Grants read-only access to container registry images on private projects.
- [ ] write_registry
  Grants write access to container registry images on private projects.
- [ ] sudo
  Grants permission to perform API actions as any user in the system, when authenticated as an admin user.

4. Save your new personal access token at a secure location like a password manager.



Your new personal access token has been created.

Personal Access Tokens

You can generate a personal access token for each application you use that needs access to the GitLab API.

Your new personal access token

Y5iiawSfP9zD7a5WRZ_F

Make sure you save it - you won't be able to access it again.

Using the Personal Access Token and Git Credential Helper at the CLI

By default git credentials are not cached at all. Every connection will prompt you for your username and password (your personal access token when two-factor authentication is enabled).

Inputting the same credentials over and over again is annoying. This is where Git Credential helper comes in place. Git credentials helper can cache credentials in memory (default for 15 minutes) or store credentials in a file.

Store credentials in memory

When you execute the following command, your credentials are stored in memory for 15 minutes:

```
$ git config --global credential.helper cache
```

Or for 8 hours

```
$ git config --global credential.helper 'cache --timeout= 28800'
```

Store credentials in a file

When you execute the following command, your credentials will be stored unencrypted indefinitely on disk, protected only by filesystem permissions. By default, the git credentials will be stored in the ".git-credentials" file in the user's home directory (~/.git-credentials).

```
$ git config --global credential.helper store
```

When you access the remote repository now, your credentials will be asked once:

```
$ git push http://example.com/repo.git
Username: <type your username>
Password: <type your personal access token>
```

```
[several days later]
$ git push http://example.com/repo.git
[your credentials are used automatically]
```