

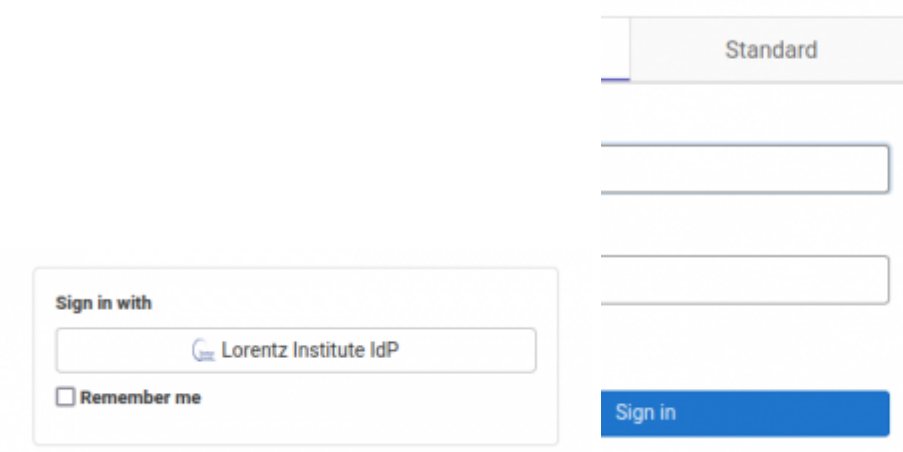
Gitlab Notes

IMPORTANT 	Gitlab logs in via the Lorentz Institute IdP might assign you a gitlab username different than expected. Please check https://gitlab.lorentz.leidenuniv.nl/-/profile/account and do not change it ssh -T git@gitlab.lorentz.leidenuniv.nl will also return your gitlab username
---	---

Lorentz Institute runs a private instance of Gitlab accessible to all institute members at <https://gitlab.lorentz.leidenuniv.nl> regardless of their role within the Institute.

You are strongly encouraged to use it to collaborate with your colleagues and fellow scientists anywhere in the world. Accounts for external users can be requested via support@lorentz.institute.nl.

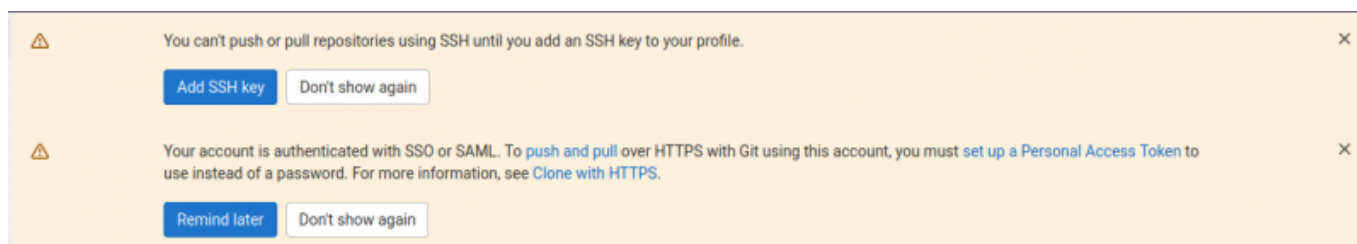
Lorentz Institute members **must use** the button *Sign in with Lorentz Institute IdP* to login via the institute Identity Provider Server, where as external users must use the *Standard* tab in the authentication form.



SSH Keys and Personal Access Tokens

Because access to the Institute Gitlab instance occurs via Single-Sign On¹⁾ for all IL members, git *push* and *pull* operations must be performed either using SSH key-pairs (SSH protocol) or using Personal Access Tokens (HTTPS protocol). Failing to set up SSH key-pairs or Access Tokens will prevent IL members to interact via git with our Gitlab server.

IL users that do not set up SSH key-pairs or Access Tokens will likely receive upon login a warning message like the one in the figure below



Generate an SSH key-pair and add it to Gitlab

On the workstation/laptop from which you would like to interact with the IL gitlab server generate a private/public key pair `$HOME/.ssh/id_ed255192)` and `$HOME/.ssh/id_ed25519.pub` using the command

```
ssh-keygen -t ed25519 -C "bongo@gitlab"
```

Visit <https://gitlab.lorentz.leidenuniv.nl/-/profile/keys>, copy the contents of the newly created `$HOME/.ssh/id_ed25519.pub` and paste it in the Key area like in the figure below. Click on *Add key*.

SSH Keys
SSH keys allow you to establish a secure connection between your computer and GitLab.

Add an SSH key
Add an SSH key for secure access to GitLab. [Learn more.](#)

Key

```
ssh-ed25519  
AAAAC3NzaC1lZDI1NTE5AAAAIHc3yo/1xv+dNTwuH51Vpl5UAsragwBEcpCcSIQWboG  
bongo@lorentz
```

Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com'.

Title
bongo@lorentz
Give your individual key a title. This will be publicly visible.

Expiration date
dd / mm / yyyy
Key can still be used after expiration.

Add key

You can non perform git operations over the *SSH* protocol. Test that your set up is correct

```
ssh -T git@gitlab.lorentz.leidenuniv.nl  
Welcome to GitLab, @bongo!
```



NOTE

In some exceptional cases³⁾ your gitlab username might differ from your Institute username. This is normal and no action is required on your side. Nonetheless, you are encouraged to take note of it because you will need it for all git operations.

Generate a Gitlab Personal Access Token

Visit https://gitlab.lorentz.leidenuniv.nl/-/profile/personal_access_tokens, choose a token name, assign the necessary scopes⁴⁾, and click on *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

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

Expiration date

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_api**
Grants read 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_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).

Create personal access token

Your personal access token will look something like xyzop-ybVxByb1aUBLA7RUAGAs (this is a fake one). Please take note of it and store it in a very safe place. Disclosing this token is equivalent to disclosing your credentials! Should a token be compromised at any time, just visit https://gitlab.lorentz.leidenuniv.nl/-/profile/personal_access_tokens to revoke any compromised tokens.

Your IL account is now enabled to perform git operations over HTTPS.

Examples

In the examples that follow it is shown how you can synchronise for the first time your local git repo to Gitlab and viceversa. The examples below assume you are typing all commands in a BASH GNU/Linux terminal and that you are familiar with the basics of git by reading carefully the following documents

- <https://gitlab.lorentz.leidenuniv.nl/help/gitlab-basics/start-using-git>
- <https://git-scm.com/docs/user-manual>

Local repo to gitlab

```
# create test repo
mkdir test
cd test/
git init .
echo "Hello World" >> README.md
```

```
git add *
git commit . -m "Test"
git push --set-upstream https://gitlab.lorentz.leidenuniv.nl/bongol/test.git
master
Username for 'https://gitlab.lorentz.leidenuniv.nl': bongol
Password for 'https://bongol@gitlab.lorentz.leidenuniv.nl': <Personal Access
Token>

Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 231 bytes | 115.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote:
remote: The private project bongol/test was successfully created.
remote:
remote: To configure the remote, run:
remote:   git remote add origin
https://gitlab.strw.leidenuniv.nl/bongol/test.git
remote:
remote: To view the project, visit:
remote:   https://gitlab.strw.leidenuniv.nl/bongol/test
remote:
remote:
remote:
To https://gitlab.lorentz.leidenuniv.nl/bongol/test.git
* [new branch]      master -> master
branch 'master' set up to track
'https://gitlab.lorentz.leidenuniv.nl/bongol/test.git/master'.
```

Remote repo to local file system

```
git clone https://gitlab.strw.leidenuniv.nl/bongol/test.git
Cloning into 'test'...
Username for 'https://gitlab.strw.leidenuniv.nl': bongol
Password for 'https://bongol@gitlab.strw.leidenuniv.nl': <Personal Access
Token>
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

1)

In other words it uses access and refresh tokens rather than usernames and passwords

2)

This is your private key which must remain secret otherwise your account is compromised.

3)

Think of users with the same private email address or same username but different domain.

4)

Read also as permissions

From:

<https://helpdesk.physics.leidenuniv.nl/wiki/> - **Computer Documentation Wiki**

Permanent link:

https://helpdesk.physics.leidenuniv.nl/wiki/doku.php?id=gitlab_notes&rev=1648455380

Last update: **2022/03/28 08:16**

