GitLab MCP Server with Gemini CLI

(With a repository creation and commit example)

Quick Guide (CLI)

⚠️THE OFFICIAL DOCUMENTATION ON THIS TOPIC IS REALLY SLIM (NONEXISTENT)⚠️

1. Installation

To use the GitLab MCP Server you need the following:

- a stable internet connection,
- Docker,
- Local GitLab Server,
- Gemini CLI.

If you don't have the Gemini CLI installed, refer back to my previous guide about Installing Gemini CLI on your device.

First to be able to do anything you have to install Docker on your device. Here's a tutorial on how to do it:

ON LINUX:

> *sudo apt update*
> *sudo apt install -y docker.io docker-compose*
> *sudo systemctl enable --now docker*

ON MAC:

> Go to this website: DOCKER INSTALL
> Download the correct version of the software and install it on your device (REMEMBER TO CHOOSE THE CORRECT CPU!)
> Run the Docker Desktop Application

ON WINDOWS:

> Go to this website: DOCKER INSTALL
> Download the correct version of the software and install it on your device
> Reboot your device
> Run the Docker Desktop Application

To check if Docker installed correctly on your device type the following commands into your terminal:
> *docker -v*
> *docker-compose -v*

If you have installed Docker correctly the next step is to start the GitLab Local Server on your device.

2. Configuration

To do that you need to create a directory (preferably named along the lines of “docker_gitlab”) and create a **docker-compose.yml** file inside that directory.
That file allows you to use Docker to create a Docker container, download needed files and run the server using one command.

Here is the *docker-compose.yml file:*

```
version: '3.8'

services:
  gitlab:
    image: 'gitlab/gitlab-ce:latest'
    platform: linux/amd64
    # This line is required for Macs with Apple Silicon processors
(M1/M2/M3).
    # On Intel systems (Windows/Linux) it will also work without issues
(it will be ignored or used natively).

    container_name: gitlab-local
    restart: always
    hostname: 'localhost'

    environment:
      GITLAB_OMNIBUS_CONFIG: |
        external_url 'http://localhost:8080'
        gitlab_rails['initial_root_password'] = 'YOUR_SECURE_PASSWORD'
        gitlab_rails['gitlab_shell_ssh_port'] = 10022

    ports:
```

```
    - '8080:8080'
    - '10022:22'

  volumes:
    - ./gitlab/config:/etc/gitlab
    - ./gitlab/logs:/var/log/gitlab
    - ./gitlab/data:/var/opt/gitlab

  shm_size: '256m'
```

After creating that file open the directory of that file in terminal and type in the following command:

ON WINDOWS AND MAC:

> docker compose up -d

ON LINUX:

> sudo docker compose up -d

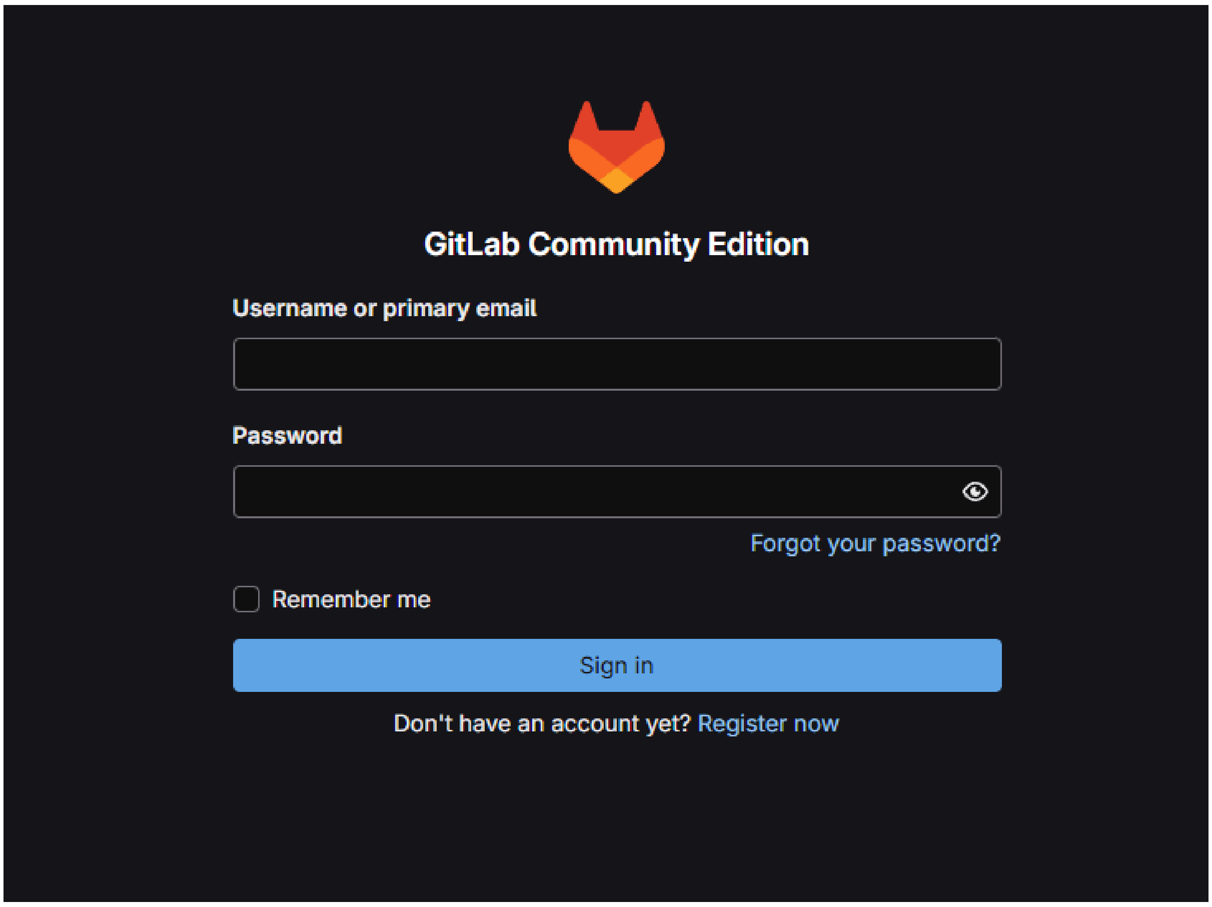
This command will create a Docker container, pull the server image and run it on your device. Using the -d flag allows us to free the terminal immediately after running the command.
Now you need to wait about 5 - 10 minutes so the server fully starts. You can check its progress by going to the localhost link.

You can also optionally check the logs by using the following command:

> docker compose logs -f

To be able to use the local GitLab Server you need to log in.

After the server fully starts you should see this on your screen when visiting the [*http://localhost*](http://localhost) link:



Here to log in you need to use the username ***root*** and for the password anything you used in your docker-compose.yml file as the ***gitlab_rails['initial_root_password']***

Now to be able to authenticate the GitLab MCP Server for the Gemini CLI you need to turn on two feature flags ("mcp_server" and "oauth_dynamic_client_registration") on your local GitLab Server.

To do that first you need to use the following command to run the GitLab Rails Console:

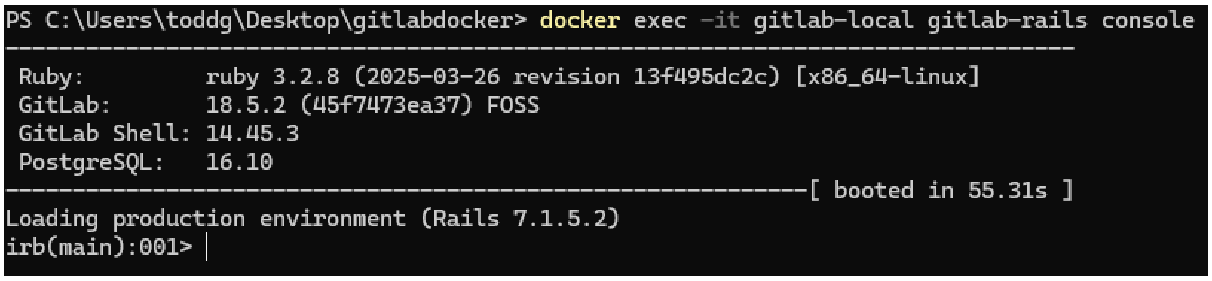
ON WINDOWS AND MAC:

> *docker compose exec -it gitlab gitlab-rails console*

ON LINUX:

> *sudo docker compose exec -it gitlab gitlab-rails console*

After running it you should see this interface pop-up in your terminal:



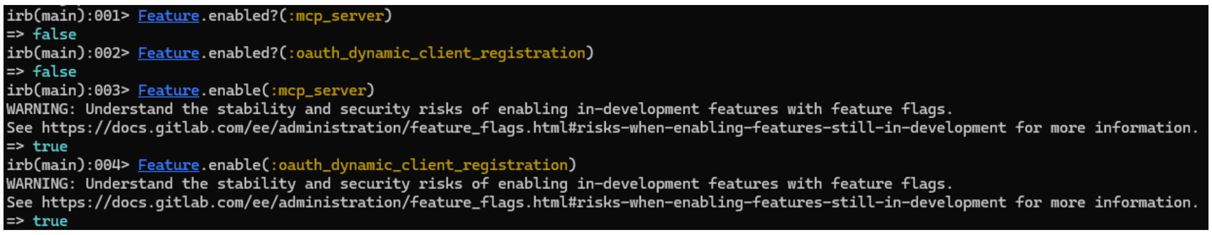
```
PS C:\Users\toddg\Desktop\gitlabdocker> docker exec -it gitlab-local gitlab-rails console
--------------------------------------------------------------------------------
 Ruby:         ruby 3.2.8 (2025-03-26 revision 13f495dc2c) [x86_64-linux]
 GitLab:       18.5.2 (45f7473ea37) FOSS
 GitLab Shell: 14.45.3
 PostgreSQL:   16.10
------------------------------------------------------------[ booted in 55.31s ]
Loading production environment (Rails 7.1.5.2)
irb(main):001>
```

In here you need to use two types of commands:

> *Feature.enable?(<FEATURE_NAME>)* ← This one to check if the feature is enabled

> *Feature.enable(<FEATURE_NAME>)* ← This one to enable the feature

Now here is how to use those commands in the GitLab Rails Console:



```
irb(main):001> Feature.enabled?(:mcp_server)
=> false
irb(main):002> Feature.enabled?(:oauth_dynamic_client_registration)
=> false
irb(main):003> Feature.enable(:mcp_server)
WARNING: Understand the stability and security risks of enabling in-development features with feature flags.
See https://docs.gitlab.com/ee/administration/feature_flags.html#risks-when-enabling-features-still-in-development for more information.
=> true
irb(main):004> Feature.enable(:oauth_dynamic_client_registration)
WARNING: Understand the stability and security risks of enabling in-development features with feature flags.
See https://docs.gitlab.com/ee/administration/feature_flags.html#risks-when-enabling-features-still-in-development for more information.
=> true
```

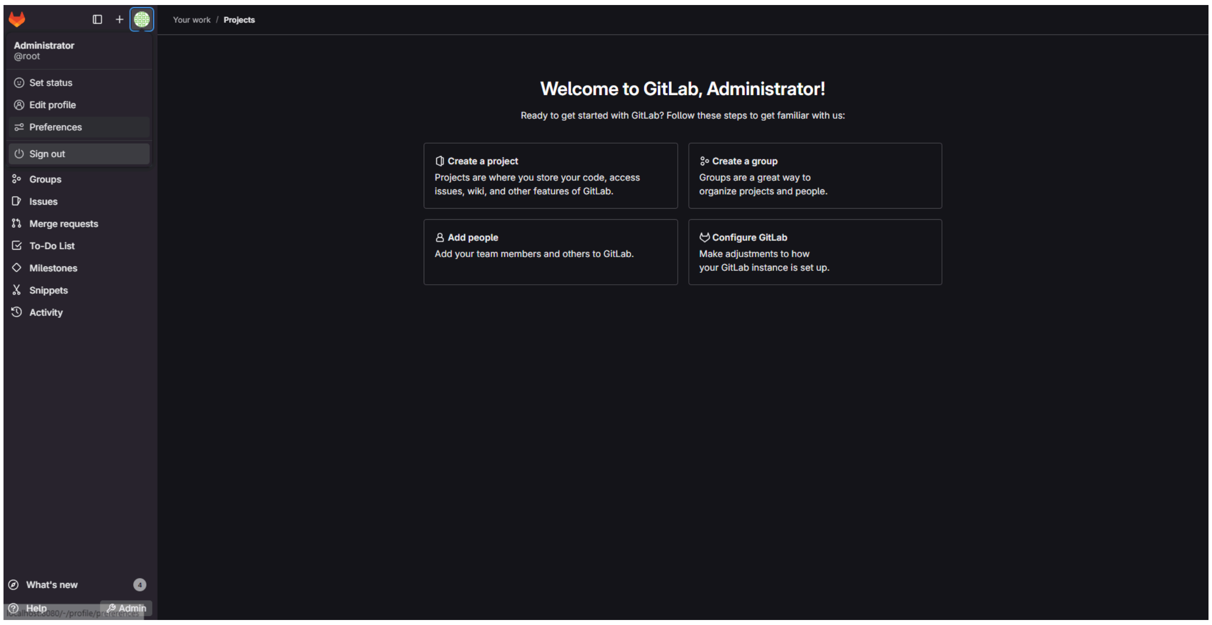
3. GitLab MCP Server installation and authentication

After you successfully enable the feature flags we can install and authenticate the GitLab MCP Server.

First you need to insert the following JSON code into the settings.json file (you can find the file in the following directory *~/.gemini*):

```
"mcpServers": {
    "gitlab": {
      "command": "npx",
      "args": [
        "-y",
        "@zereight/mcp-gitlab"
      ],
      "env": {
        "GITLAB_API_URL": "http://localhost/api/v4",
        "GITLAB_PERSONAL_ACCESS_TOKEN": "<YOUR_PERSONAL_TOKEN>",
        "NODE_TLS_REJECT_UNAUTHORIZED": "0"
      }
    }
  }
```

Now to be able to fully authenticate your MCP Server you need to go back to http://localhost and log in. Next you need to click here (check screenshot below)



Your work / Projects

Administrator
@root

Set status
Edit profile
Preferences
Sign out
Groups
Issues
Merge requests
To-Do List
Milestones
Snippets
Activity

What's new 4
Help
Admin
localhost:8080/-/profile/preferences

Welcome to GitLab, Administrator!

Ready to get started with GitLab? Follow these steps to get familiar with us:

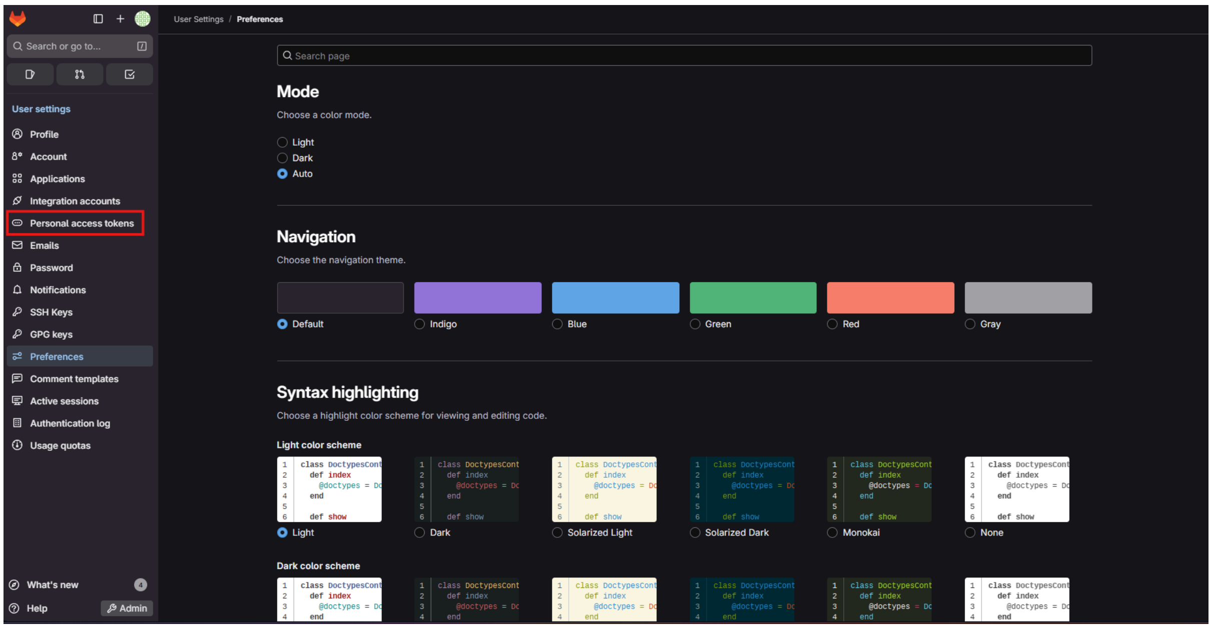
Create a project
Projects are where you store your code, access issues, wiki, and other features of GitLab.

Create a group
Groups are a great way to organize projects and people.

Add people
Add your team members and others to GitLab.

Configure GitLab
Make adjustments to how your GitLab instance is set up.

And go to “Preferences”
From here go to “Personal access tokens” (check screenshot below)



Mode

Choose a color mode.

Light [✓ Auto] Dark

Navigation

Choose the navigation theme.

[✓ Default] Indigo Blue Green Red Gray

Syntax highlighting

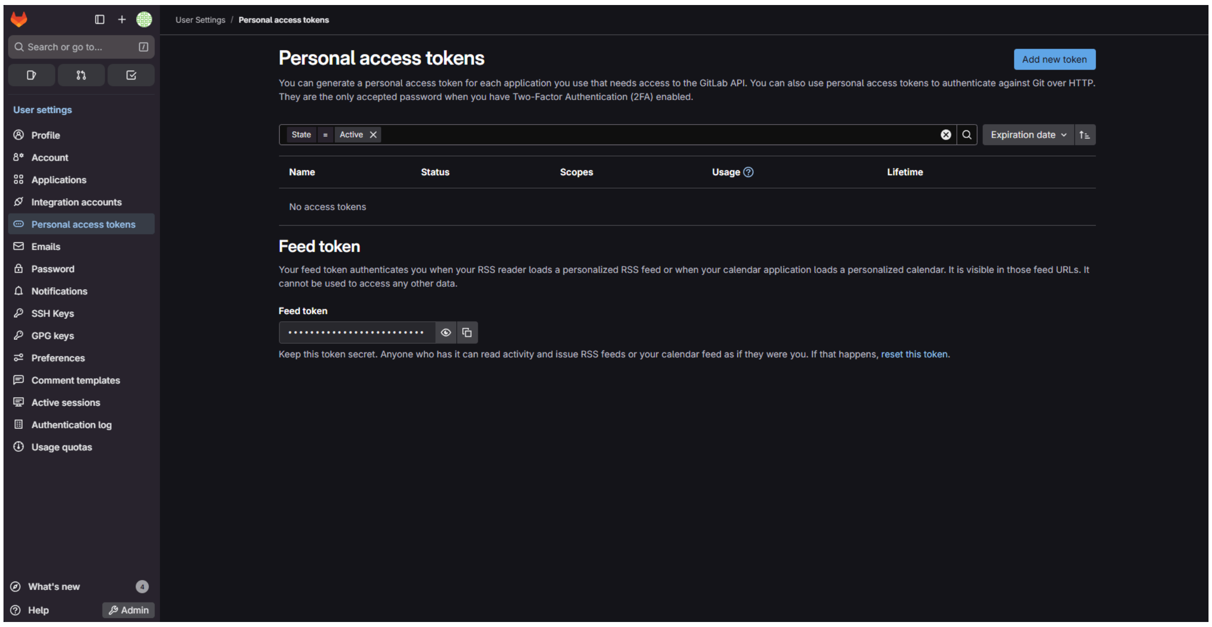
Choose a highlight color scheme for viewing and editing code.

Light color scheme

[✓ Light] Dark Solarized Light Solarized Dark Monokai None

Dark color scheme

Now you are ready to create your 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.

Name	Status	Scopes	Usage	Lifetime
No access tokens				

Feed token

Your feed token authenticates you when your RSS reader loads a personalized RSS feed or when your calendar application loads a personalized calendar. It is visible in those feed URLs. It cannot be used to access any other data.

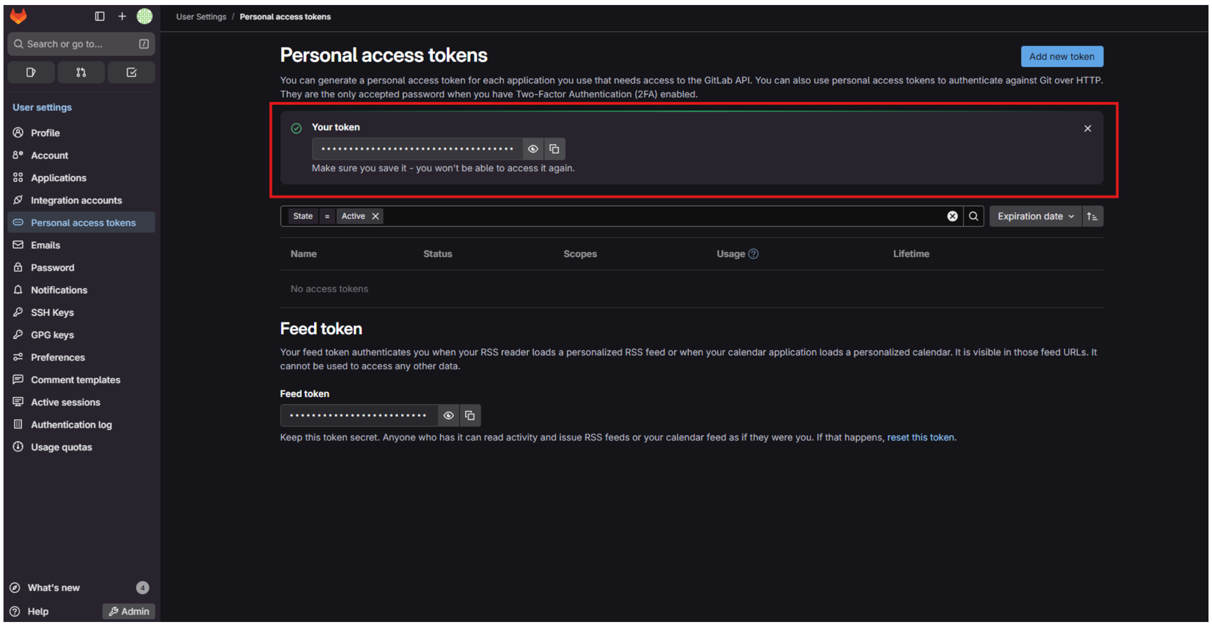
Feed token

••••••••••••••••••••••••••

Keep this token secret. Anyone who has it can read activity and issue RSS feeds or your calendar feed as if they were you. If that happens, reset this token.

In here choose “Add new token” and go through the process of creating one. **While selecting scopes remember to select the “api” one.** Add a name, description and expiration date and scroll to the bottom and click “Create token”.

After that you should see this on your screen:



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.

Your token

••••••••••••••••••••••••••••••••••••

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

Name	Status	Scopes	Usage	Lifetime
No access tokens				

Feed token

Your feed token authenticates you when your RSS reader loads a personalized RSS feed or when your calendar application loads a personalized calendar. It is visible in those feed URLs. It cannot be used to access any other data.

Feed token

••••••••••••••••••••••••••

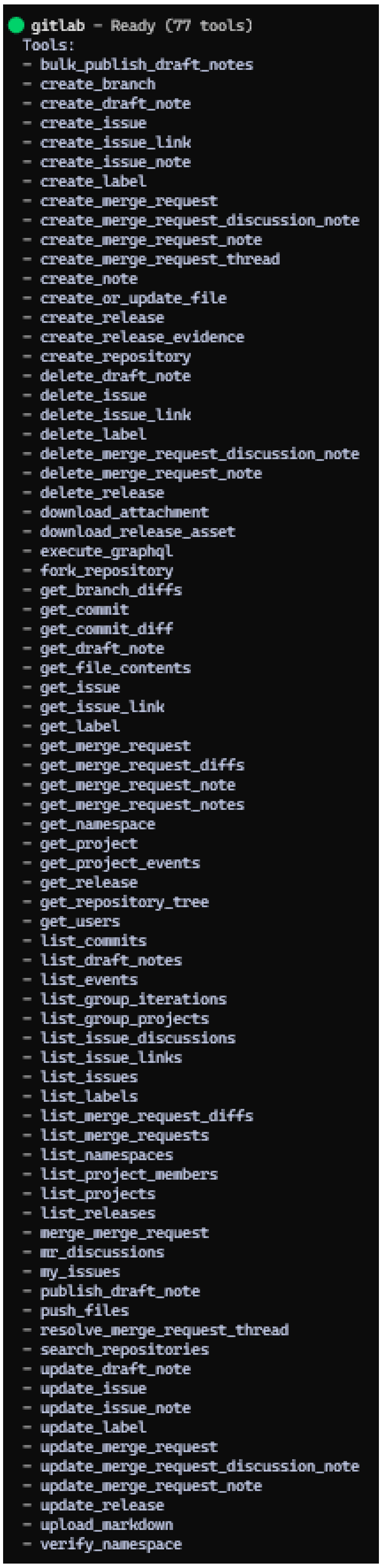
Keep this token secret. Anyone who has it can read activity and issue RSS feeds or your calendar feed as if they were you. If that happens, reset this token.

Now copy the newly created token and paste it into the settings.json file.

Now you can run the Gemini CLI and the GitLab MCP Server will automatically authenticate you and let you fully take advantage of it!
After running the Gemini CLI use the following command to check if the MCP server authenticated:

```
> /mcp list
```

If everything went according to plan you should see this in your terminal



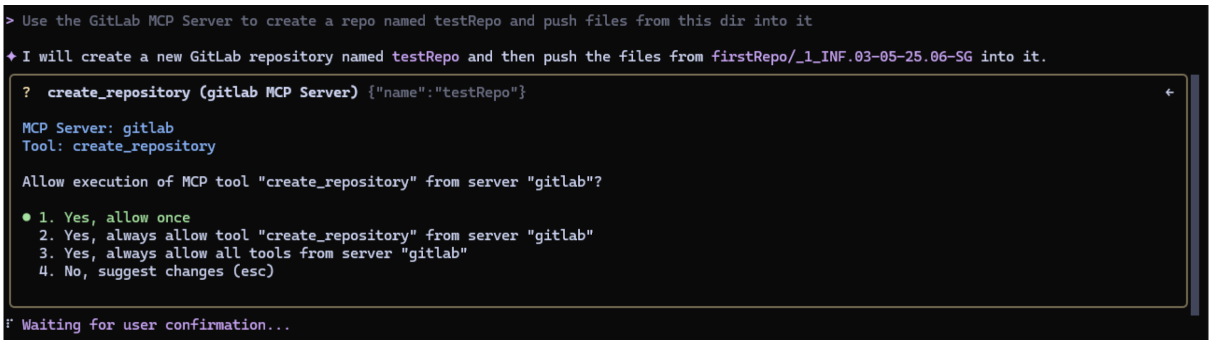
```
● gitlab - Ready (77 tools)
  Tools:
  - bulk_publish_draft_notes
  - create_branch
  - create_draft_note
  - create_issue
  - create_issue_link
  - create_issue_note
  - create_label
  - create_merge_request
  - create_merge_request_discussion_note
  - create_merge_request_note
  - create_merge_request_thread
  - create_note
  - create_or_update_file
  - create_release
  - create_release_evidence
  - create_repository
  - delete_draft_note
  - delete_issue
  - delete_issue_link
  - delete_label
  - delete_merge_request_discussion_note
  - delete_merge_request_note
  - delete_release
  - download_attachment
  - download_release_asset
  - execute_graphql
  - fork_repository
  - get_branch_diffs
  - get_commit
  - get_commit_diff
  - get_draft_note
  - get_file_contents
  - get_issue
  - get_issue_link
  - get_label
  - get_merge_request
  - get_merge_request_diffs
  - get_merge_request_note
  - get_merge_request_notes
  - get_namespace
  - get_project
  - get_project_events
  - get_release
  - get_repository_tree
  - get_users
  - list_commits
  - list_draft_notes
  - list_events
  - list_group_iterations
  - list_group_projects
  - list_issue_discussions
  - list_issue_links
  - list_issues
  - list_labels
  - list_merge_request_diffs
  - list_merge_requests
  - list_namespaces
  - list_project_members
  - list_projects
  - list_releases
  - merge_merge_request
  - mr_discussions
  - my_issues
  - publish_draft_note
  - push_files
  - resolve_merge_request_thread
  - search_repositories
  - update_draft_note
  - update_issue
  - update_issue_note
  - update_label
  - update_merge_request
  - update_merge_request_discussion_note
  - update_merge_request_note
  - update_release
  - upload_markdown
  - verify_namespace
```

CONGRATULATIONS! Now you can “fully” take advantage of the GitLab MCP Server!

4. Use case example

For the use case example I will show you how the Gemini CLI can create repositories using this MCP Server.
First run your gemini in your desired directory.
Next formulate a prompt that tells the gemini LLM what it should use the MCP Server for.
EXAMPLE : *Use the GitLab MCP Server to create a repo named testRepo.*

The Gemini CLI will ask you next for permission to use the tools included in the MCP Server.



```
> Use the GitLab MCP Server to create a repo named testRepo and push files from this dir into it

✦ I will create a new GitLab repository named testRepo and then push the files from firstRepo/_1_INF.03-05-25.06-SG into it.
 ┌──────────────────────────────────────────────────────────────────────────────────────────────────┐
 │ ?  create_repository (gitlab MCP Server) {"name":"testRepo"}                                  ← │
 │                                                                                                  │
 │ MCP Server: gitlab                                                                               │
 │ Tool: create_repository                                                                          │
 │                                                                                                  │
 │ Allow execution of MCP tool "create_repository" from server "gitlab"?                            │
 │                                                                                                  │
 │ ● 1. Yes, allow once                                                                             │
 │   2. Yes, always allow tool "create_repository" from server "gitlab"                             │
 │   3. Yes, always allow all tools from server "gitlab"                                            │
 │   4. No, suggest changes (esc)                                                                   │
 │                                                                                                  │
 └──────────────────────────────────────────────────────────────────────────────────────────────────┘
⠏ Waiting for user confirmation...
```

Choose one of the following options and watch the magic happen!