[Review] Working with GitLab Personal Access Tokens (PAT) in devNext

Last edited by SMOHANA1 1 year ago

Overview

This document explains what a GitLab Personal Access Token (PAT) is and provides guidance on how to configure and use PATs with devNext. **It is applicable for both Continuous Integration (CI) workflows using jenkins and general developer workflows**. The document also includes the necessary commands to enable and disable the PAT configuration.

Why GitLab Personal Access Token (PAT)?

A GitLab Personal Access Token (PAT) is a secure way to authenticate with GitLab and access API. **It acts as a replacement for username-passwords and OAuth tokens**, offering a higher level of security and control. PATs are used for authenticating API requests and accessing GitLab repositories.

Advantages of Using PATs

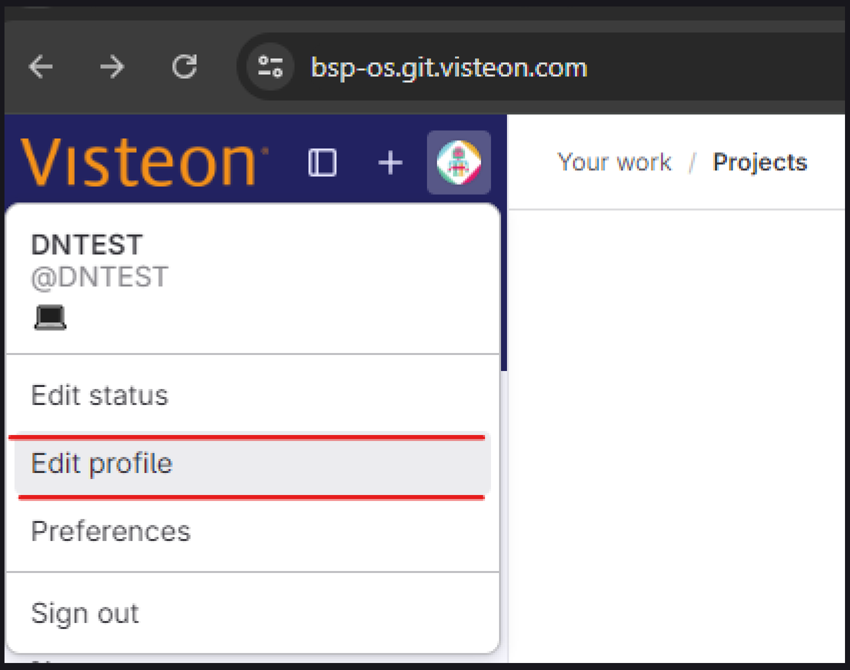
- **Enhanced Security**: PATs can be scoped to limit access to specific resources.
- **Revocability**: PATs can be easily revoked if compromised.
- **Ease of Use**: They simplify the processes of automation and integration by providing an easy way to authenticate without the need for username and password.

Using PAT with devNext

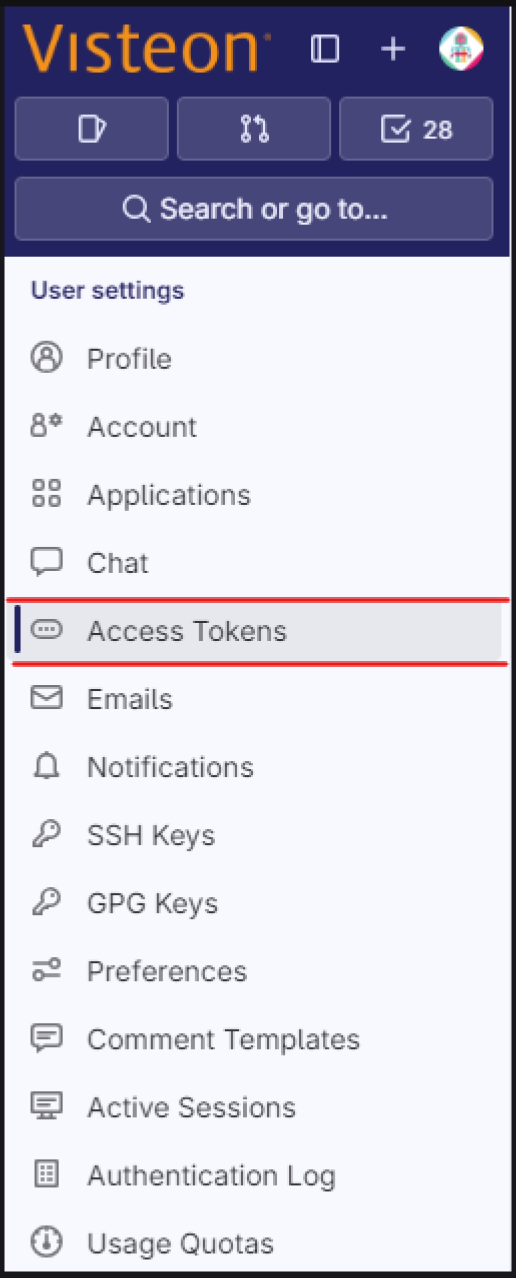
When working with devNext, having a PAT allows you to authenticate and interact with GitLab repositories securely. Follow the steps below to enable and configure PAT for devNext.

Enabling PAT for devNext

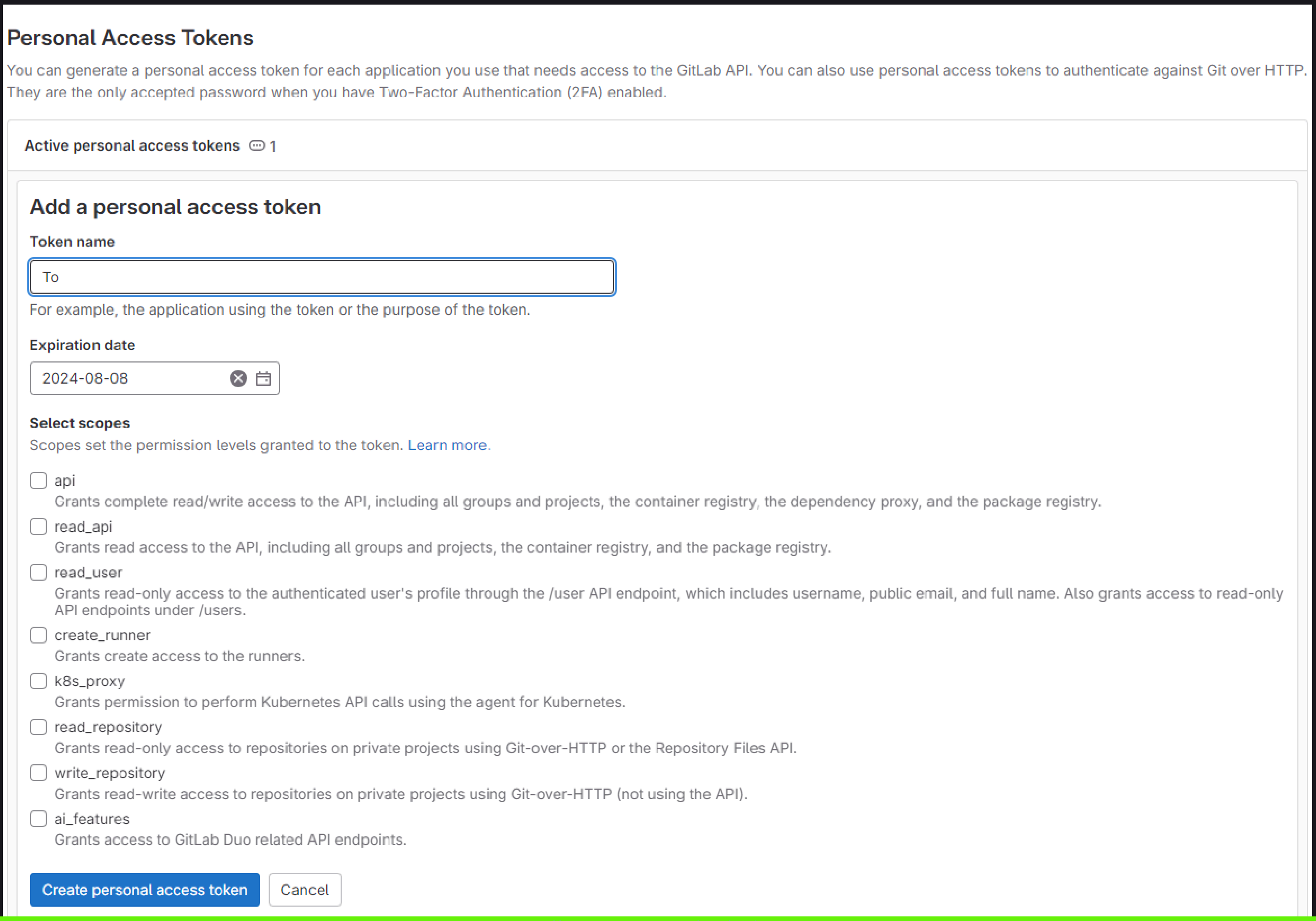
1. **Generate a PAT**: First, generate a PAT in GitLab with the necessary scopes. Follow the GitLab documentation to create a PAT: [Creating a personal access token](#).

   - Login to the server
   - on the left sidebar, select your avatar
   - select `Edit Profile`
   - 

   - On the left sidebar, select Access Tokens.

- 

- Select Add new token.
- Enter a name and expiry date for the token.
    - The token expires on that date at midnight UTC.
    - If you do not enter an expiry date, the expiry date is automatically set to 365 days later than the current date.
    - By default, this date can be a maximum of 365 days later than the current date.



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.

Active personal access tokens 1

Add a personal access token

Token name: To

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

Expiration date: 2024-08-08

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, the dependency proxy, 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.
- [ ] create_runner
  Grants create access to the runners.
- [ ] k8s_proxy
  Grants permission to perform Kubernetes API calls using the agent for Kubernetes.
- [ ] 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).
- [ ] ai_features
  Grants access to GitLab Duo related API endpoints.

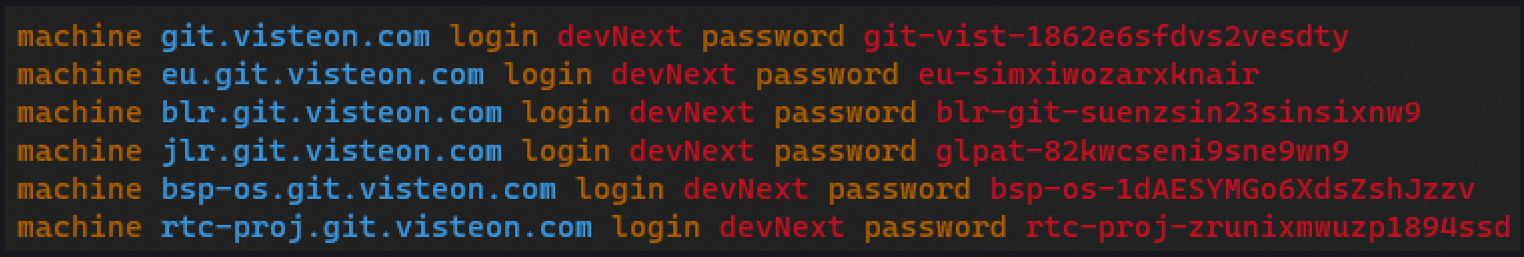
FAQs at https://q2a.visteon.com/tag/git/

- Select the desired scopes.
- Select Create personal access token.
- Save the personal access token safe. After you leave the page, you no longer have access to the token.

2. **Configure netrc to Use PAT**: Once you have your PAT (Personal Access Token), You need to write it as your machine credential in the `netrc` file. Refer to the document for detailed instructions on how to write machine credentials in `netrc` and this will be used by devNext to access the GitLab resources.

   Example Here, the user `devNext` and their PAT configuration for the servers are shown below:

   - Syntax : `machine {hostname} login {username} password {your-password}`
   - Example my server url https://bsp-os.git.visteon.com and user cdsid is **devNext**
   - `machine bsp-os.git.visteon.com login devNext password <PAT_HERE>`



```
machine git.visteon.com login devNext password git-vist-1862e6sfdvs2vesdty
machine eu.git.visteon.com login devNext password eu-simxiwozarxknair
machine blr.git.visteon.com login devNext password blr-git-suenzsin23sinsixnw9
machine jlr.git.visteon.com login devNext password glpat-82kwcseni9sne9wn9
machine bsp-os.git.visteon.com login devNext password bsp-os-1dAESYMGo6XdsZshJzzv
machine rtc-proj.git.visteon.com login devNext password rtc-proj-zrunixmwuzp1894ssd
```

Command to Enable PAT for devNext

```
git config --global devnext.auth PAT
```

This command sets the `devNext.auth` configuration to use your PAT, allowing devNext to access the GitLab resources using the PAT for operations like configure workspace, gitfetch and gitupdate.

Command to Unset PAT Configuration for devNext

```
git config --global --unset devnext.auth
```

This command removes the `devnext.auth` configuration, stopping devNext from using the PAT for authentication.

Q2A Forum: Q&A platform helps find answers and ask your questions, learn and share knowledge https://q2a.visteon.com

Support: If you've any Queries w.r.t above mentioned steps, please reach out to devNext.support@visteon.com