

# Basic Interaction with GitHub Cheat-Sheet

There are various remote repository hosting sites:

- [GitHub](#)
- [BitBucket](#)
- [Gitlab](#).

Follow the workflow at <https://github.com/join> to set up a free account, username, and password. After that, [these steps](#) will help you create a brand new repository on GitHub.

Some useful commands for getting started:

Command	Explanation & Link
git clone URL	<a href="#">Git clone is used to clone a remote repository into a local workspace</a>
git push	<a href="#">Git push is used to push commits from your local repo to a remote repo</a>
git pull	<a href="#">Git pull is used to fetch the newest updates from a remote repository</a>

This can be useful for keeping your local workspace up to date.

- <https://help.github.com/en/articles/caching-your-github-password-in-git>
  - <https://help.github.com/en/articles/generating-an-ssh-key>
- 

## Git Remotes Cheat-Sheet

Command	Explanation & Links
git remote	<a href="#">Lists remote repos</a>
git remote -v	<a href="#">List remote repos verbosely</a>
git remote show <name>	<a href="#">Describes a single remote repo</a>
git remote update	<a href="#">Fetches the most up-to-date objects</a>
git fetch	<a href="#">Downloads specific objects</a>
git branch -r	<a href="#">Lists remote branches</a> ; can be combined with other branch arguments to manage remote br

You can also see more in the video [Cryptography in Action](#) from the course [IT Security: Defense against the digital dark arts](#).

---

## Conflict Resolution Cheat Sheet

Merge conflicts are not uncommon when working in a team of developers, or on Open Source Software. Fortunately, GitHub has some good documentation on how to handle them when they happen:

- <https://help.github.com/en/github/collaborating-with-issues-and-pull-requests/about-merge-conflicts>
- <https://help.github.com/en/github/collaborating-with-issues-and-pull-requests/resolving-a-merge-conflict-using-the-command-line>

You can also use [git rebase branchname](#) to change the base of the current branch to be branchname

The git rebase command is a lot more powerful. Check out [this link](#) for more information.

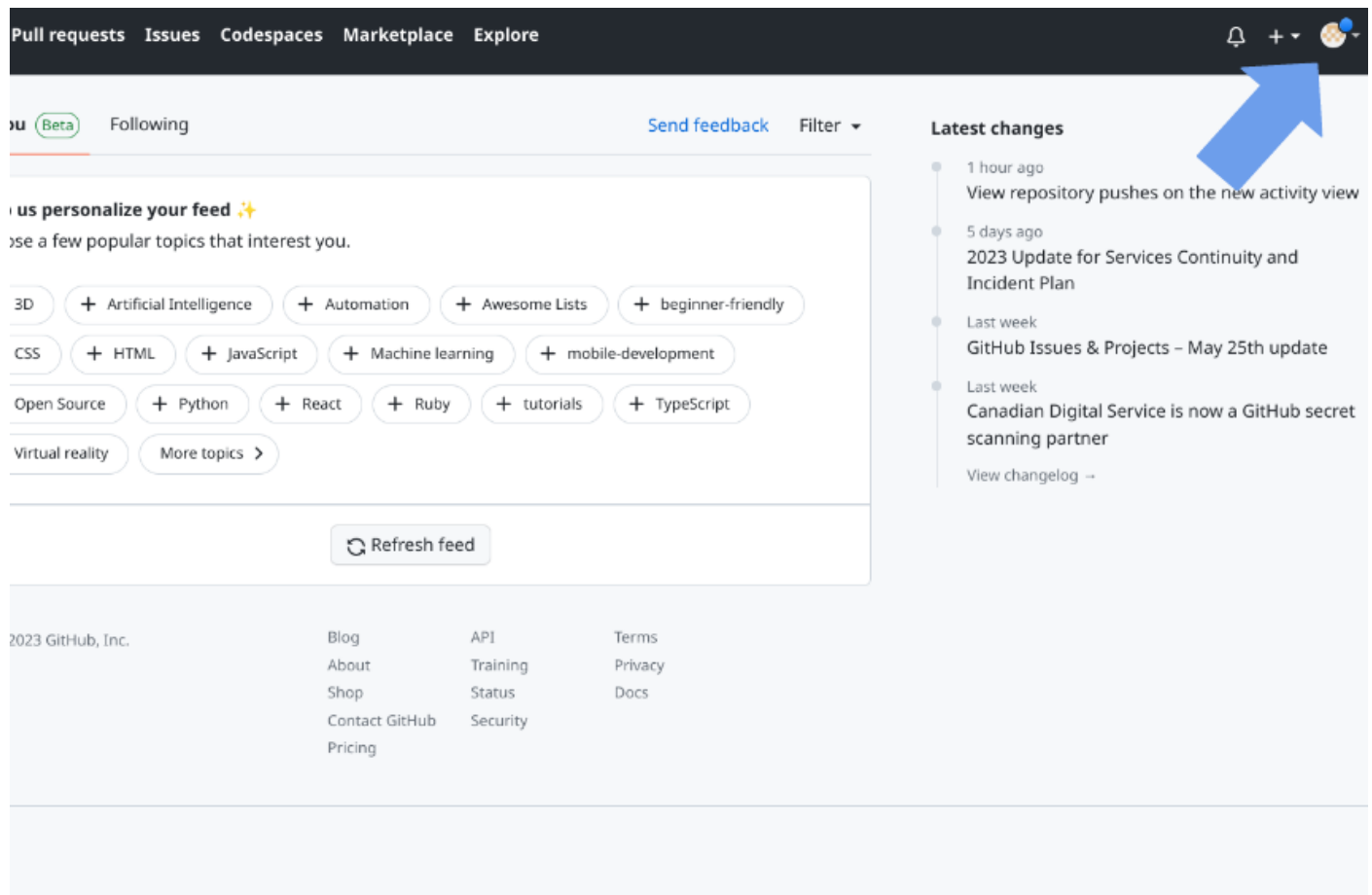
---

## Creating a Personal Access Token in GitHub

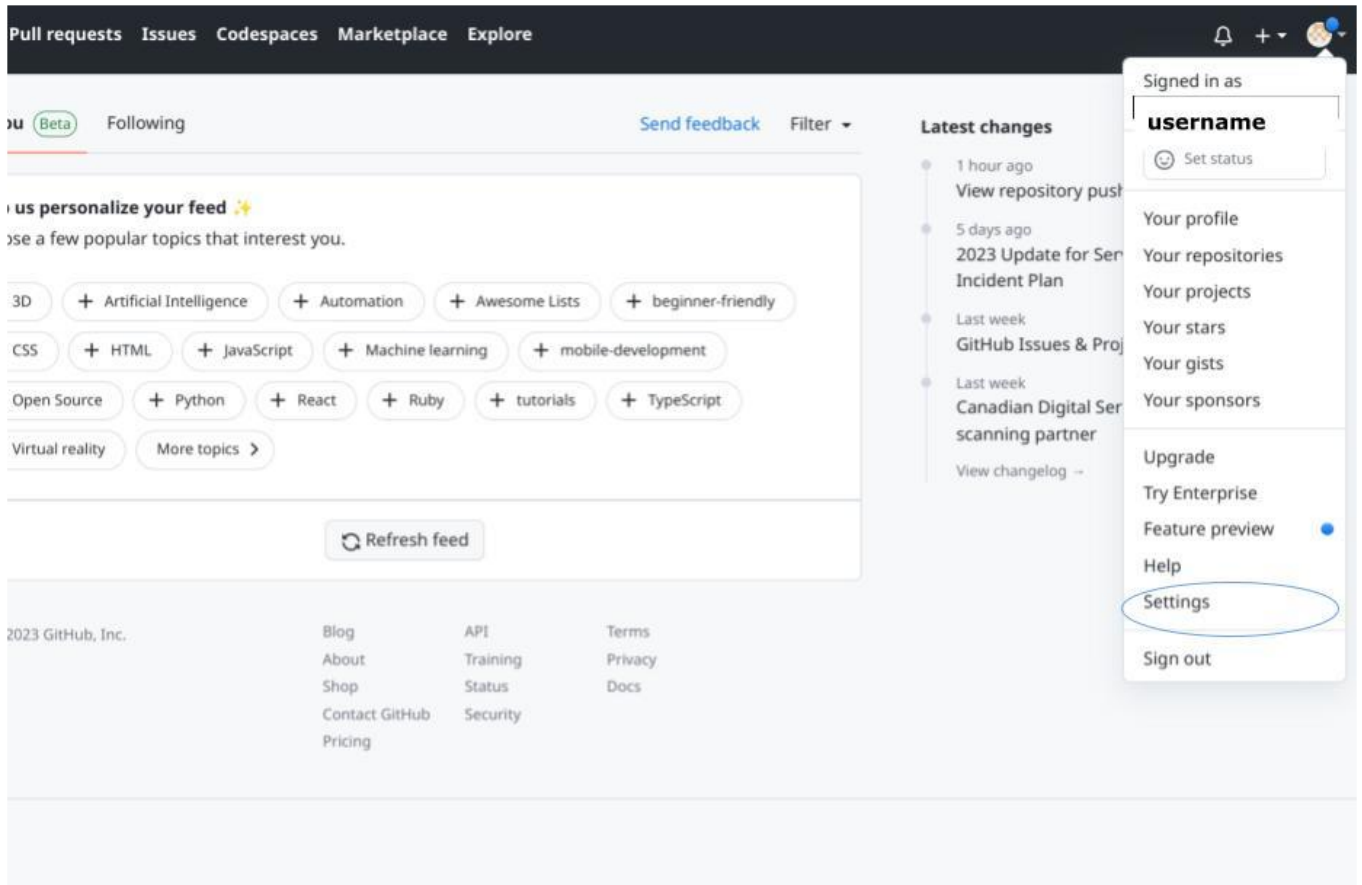
Personal access tokens are used in place of your GitHub password at the command-line. To use a personal access token, you must first create one. The following is a step-by-step guide on how to create a personal access token in GitHub which will be used in the next lab, [Qwiklabs Assessment: Introduction to GitHub](#).

# Steps to creating a personal access token in GitHub

1. Log into your GitHub account with your username and password.
2. In the upper right hand corner click on your profile picture.



3. Use the drop-down menu, and click on Settings.

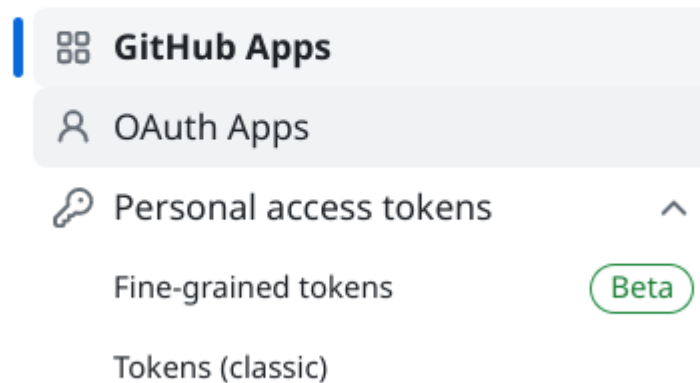


4. On the left sidebar, click on Developer Settings. It looks like this:

<> Developer settings

5. On the left sidebar, click on **Personal access tokens**. Choose **tokens (classic)**.

Settings / Developer settings



6. Click **Generate new token**. Choose **Generate new token (classic)**.

## Personal access tokens (classic)

Generate new token ▾

Revoke all

Tokens you have generated that can be used to access the

**Generate new token** Beta

Fine-grained, repo-scoped

**Generate new token (classic)**

For general use

Personal access tokens (classic) function like ordinary OAuth access tokens or can be used to [authenticate to the API over Basic Authentication](#).

Git over HTTPS,

7. In the “Note” field, give your token a name.

8. Set when you want your token to expire.

9. Select the scopes you want to grant this token. For the lab that follows make sure you select **repo** so that you can access repositories from the command-line.

## New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

### Note

TokenName

What's this token for?

### Expiration \*

30 days ▾

The token will expire on Fri, Jun 30 2023

### Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

<input checked="" type="checkbox"/> <b>repo</b>	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo_deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input checked="" type="checkbox"/> repo:invite	Access repository invitations
<input checked="" type="checkbox"/> security_events	Read and write security events
<input type="checkbox"/> <b>workflow</b>	Update GitHub Action workflows
<input type="checkbox"/> <b>write:packages</b>	Upload packages to GitHub Package Registry
<input type="checkbox"/> read:packages	Download packages from GitHub Package Registry
<input type="checkbox"/> <b>delete:packages</b>	Delete packages from GitHub Package Registry

10. Click **Generate token**.

11. Copy your new token to your clipboard. You may want to paste it in a document or note that you will delete after you have completed the lab. You will not be able to see the token again.

12. Finally, when you go to login to your GitHub account at the command-line use your GitHub username and the token you just generated for your password.

Here is the [official documentation](#) for creating personal access tokens from GitHub.