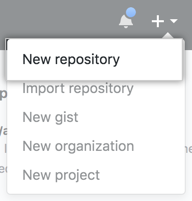
1. **Why do we need a version control system.**

Version control systems **allow multiple developers, designers, and team members to work together on the same project**. It helps them work smarter and faster! A version control system is critical to ensure everyone has access to the latest code and modifications are tracked.

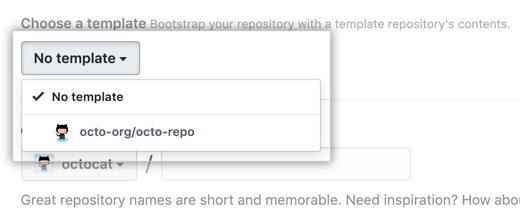
1. **Fundamentals of git.**

* Manage projects with **Repositories**
* **Clone** a project to work on a local copy
* Control and track changes with **Staging** and **Committing**
* **Branch** and **Merge** to allow for work on different parts and versions of a project
* **Pull** the latest version of the project to a local copy
* **Push** local updates to the main project

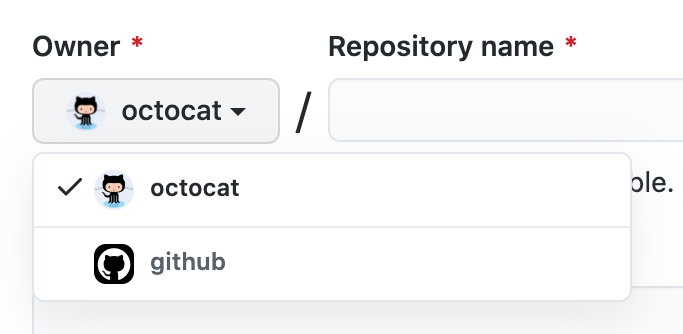
1. **Git client installation and setup.**
2. Browse to the official Git website: <https://git-scm.com/downloads>
3. Click the download link for Windows and allow the download to complete.
4. Browse to the download location (or use the download shortcut in your browser). Double-click the file to extract and launch the installer.
5. Allow the app to make changes to your device by clicking **Yes** on the User Account Control dialog that opens.
6. Review the GNU General Public License, and when you’re ready to install, click **Next**.
7. The installer will ask you for an installation location. Leave the default, unless you have reason to change it, and click **Next**.
8. A component selection screen will appear. Leave the defaults unless you have a specific need to change them and click **Next**.
9. The installer will offer to create a start menu folder. Simply click **Next**.
10. Select a text editor you’d like to use with Git. Use the drop-down menu to select Notepad++ (or whichever text editor you prefer) and click **Next**.
11. The next step allows you to choose a different name for your initial branch. The default is 'master.' Unless you're working in a team that requires a different name, leave the default option and click **Next.**
12. This installation step allows you to change the **PATH environment**. The **PATH**is the default set of directories included when you run a command from the command line. Leave this on the middle (recommended) selection and click **Next**.
13. The installer now asks which SSH client you want Git to use. Git already comes with its own SSH client, so if you don't need a specific one, leave the default option and click **Next.**
14. The next option relates to server certificates. Most users should use the default. If you’re working in an Active Directory environment, you may need to switch to Windows Store certificates. Click **Next**.
15. The next selection converts line endings. It is recommended that you leave the default selection. This relates to the way data is formatted and changing this option may cause problems. Click **Next**.
16. Choose the [terminal emulator](https://phoenixnap.com/glossary/terminal-emulation) you want to use. The default MinTTY is recommended, for its features. Click **Next**.
17. The installer now asks what the **git pull** command should do. The default option is recommended unless you specifically need to change its behavior. Click **Next**to continue with the installation.
18. Next you should choose which credential helper to use. Git uses credential helpers to fetch or save credentials. Leave the default option as it is the most stable one, and click **Next**.
19. The default options are recommended, however this step allows you to decide which extra option you would like to enable. If you use symbolic links, which are like shortcuts for the command line, tick the box. Click **Next**.
20. Depending on the version of Git you’re installing, it may offer to install experimental features. At the time this article was written, the options to include support for pseudo controls and a built-in file system monitor were offered. Unless you are feeling adventurous, leave them unchecked and click **Install**.
21. Once the installation is complete, tick the boxes to view the Release Notes or Launch Git Bash, then click **Finish**.
22. **Basic local git operations.**
23. **Creating a repository.**
24. In the upper-right corner of any page, use the  drop-down menu, and select **New repository**.



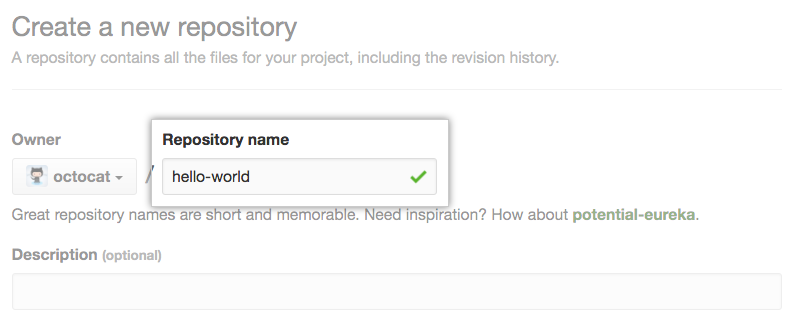
1. Optionally, to create a repository with the directory structure and files of an existing repository, use the **Choose a template** drop-down and select a template repository. You'll see template repositories that are owned by you and organizations you're a member of or that you've used before. For more information, see "[Creating a repository from a template](https://docs.github.com/en/articles/creating-a-repository-from-a-template)."



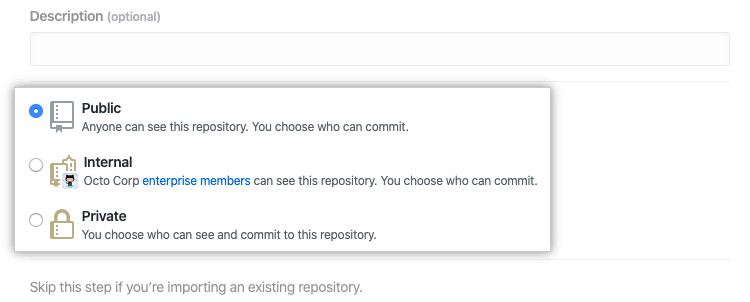
1. Optionally, if you chose to use a template, to include the directory structure and files from all branches in the template, and not just the default branch, select **Include all branches**
2. In the Owner drop-down, select the account you wish to create the repository on.



1. Type a name for your repository, and an optional description.



1. Choose a repository visibility. For more information, see "[About repositories](https://docs.github.com/en/repositories/creating-and-managing-repositories/about-repositories#about-repository-visibility)."



1. If you're not using a template, there are a number of optional items you can pre-populate your repository with. If you're importing an existing repository to GitHub, don't choose any of these options, as you may introduce a merge conflict. You can add or create new files using the user interface or choose to add new files using the command line later. For more information, see "[Importing a Git repository using the command line](https://docs.github.com/en/articles/importing-a-git-repository-using-the-command-line)," "[Adding a file to a repository](https://docs.github.com/en/repositories/working-with-files/managing-files/adding-a-file-to-a-repository#adding-a-file-to-a-repository-using-the-command-line)," and "[Addressing merge conflicts](https://docs.github.com/en/articles/addressing-merge-conflicts)."
   1. You can create a README, which is a document describing your project. For more information, see "[About READMEs](https://docs.github.com/en/articles/about-readmes)."
   2. You can create a *.gitignore* file, which is a set of ignore rules. For more information, see "[Ignoring files](https://docs.github.com/en/github/getting-started-with-github/ignoring-files)."
   3. You can choose to add a software license for your project. For more information, see "[Licensing a repository](https://docs.github.com/en/articles/licensing-a-repository)."
2. Optionally, if the personal account or organization in which you're creating uses any GitHub Apps from GitHub Marketplace, select any apps you'd like to use in the repository.
3. Click **Create repository**.
4. At the bottom of the resulting Quick Setup page, under "Import code from an old repository", you can choose to import a project to your new repository. To do so, click **Import code**.
5. **Cloning a repository**

**Step 1:** Open GitHub and navigate to the main page of the repository.

**Step 2:** Under the repository name, click on **Clone or download**.

**Step 3:** Select the **Clone with HTTPs section** and **copy the clone URL** for the repository. For the empty repository, you can copy the repository page URL from your browser and skip to next step.

**Step 4:** Open Git Bash and change the current working directory to your desired location where you want to create the local copy of the repository.

**Step 5:** Use the git clone command with repository URL to make a copy of the remote repository.

1. **Making and recording changes**