In this topic, we will start exploring Git, the essential tool to control and merge your code versions. We will learn what Git is, why we need to use it, how to install it, and some important configuration settings.

**What is Git?**

**Git** is a distributed version control system that helps developers track and record changes in files. These files can include anything you want to work with, but in most cases, Git is primarily used for managing **source code** in software development projects. With Git, you can:

* Track every modification made to files;
* Roll back if an error is introduced — you can easily revert your project to a previous version;
* Compare different versions of your files helping you analyze what has changed;
* Collaborate on the same project simultaneously with others making it easier to coordinate changes.
* Merge changes made by different contributors into a single version of the project.

This is known as **version control.** The files are stored in folders on your hard drive. This is called a **repository**(often abbreviated as repo). There are two types of repositories in Git:

1. **Local repo:**
   * It exists only on your computer's local storage;
   * It allows you to work independently without needing an internet connection;
   * All changes and version history are stored locally, providing a fast and efficient workflow.
2. **Remote repo:**
   * It is an online copy of your local repository, usually hosted on platforms such as GitHub, GitLab, or Bitbucket.
   * It allows you to collaborate with others by sharing your code and changes over the internet.
   * The remote repository acts as a backup, ensuring that your work is safe and accessible from anywhere.

The ability to synchronize changes between local and remote repositories makes Git an excellent tool for teamwork and distributed development. Git is*distributed*, that is, every user has a complete copy of the repository and can be used to restore it in case of failure.

A number of services provide hosting for Git repositories, among the most famous are [GitHub](https://github.com/), [Bitbucket](https://bitbucket.org/" \t "_blank), [GitLab](https://gitlab.com/gitlab-com" \t "_blank), [Codebase](https://www.codebasehq.com/), [SourceForge](https://sourceforge.net/index/" \t "_blank), and [SourceHut](https://sourcehut.org/" \t "_blank). We will discuss only GitHub as the most popular among these resources. However, if you want, you may use the others listed above instead.

Git is a free and open-source software primarily developed for Linux,but it also supports most major operating systems, including macOS and Windows. Next you will learn how to install Git on these systems.



**Installing Git**

**1. Windows**

There are several ways to install Git on Windows. The easiest way is to use the [graphical installer](http://git-scm.com/download/win) which you can download from the official site. After installation, you need to select the options you nee. For unattended installation, you can also use the [Git Chocolatey package](https://chocolatey.org/packages/git" \t "_blank).

Another easy way to install Git is to install GitHub for Windows. This installer includes command-line utilities and GUI Git. You can download GitHub for Windows from the [GitHub Desktop website](https://desktop.github.com/).

**2. Linux**

For Linux, you can just open a terminal and install the application using your distribution package manager. If you have Fedora (or another similar distribution such as RHEL or CentOS), you can use dnf:

$ sudo dnf install git-all

If you have a Debian-based distribution like Ubuntu, try apt:

$ sudo apt install git

**3. macOS**

There are several ways to install Git on Mac.

You may install the Xcode Command Line Tools by typing the following command in your terminal:

xcode-select --install

Then you can simply check if Git is already installed:

$ git --version

If Git is not installed, you will be prompted to install it.

The other way is to use homebrew, which can be installed from the terminal using the command below:

/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install.sh)"

More information on homebrew can be found on the [official homebrew website](https://brew.sh/). After installing it, run in the terminal:

brew install git

You can also install Git when installing GitHub for Mac. Their GUI Git tool has the option to install command line tools as well. You can download the [GitHub client for Mac](https://desktop.github.com/).

Usually, the installation is successful, but just in case, we recommend you check that the latest version of Git has actually been installed by running the git --version command in the terminal. The result will be something like the following:

git version 2.33.1

Versions are updated, so you may have a different one. In our example, this is the*2.33.1*version.

It means that Git is installed, and you can start working with it.

**Git settings**

Now that we have successfully installed Git, it’s time to configure some settings to personalize your Git environment. Let's start by setting up two of the most critical ones: your username and email address. These identifiers will be attached to every commit you make, helping to keep track of who made changes in a collaborative environment. This is essential to know who contributed what which promotes transparency and maintains accountability.

Open the terminal and run the commands:

git config --global user.name "My Name"

git config --global user.email myEmail@example.com

Now your actions will be marked with your name and e-mail address. Thus, users will always know who is responsible for certain changes. By adding the --global flag, you’re setting these configurations for all repositories on your system. However, if you want to set a different username or email for a specific project, you can do so by omitting the --global flag and running the commands from within that project’s directory:

git config user.name "My Name"

git config user.email projectEmail@example.com

To confirm that your settings have been saved correctly, you can use the following command:

git config --global --list # you can omit the --global flag

This will display all global configurations currently set for Git. Look for lines like:

user.name=My Name

user.email=myEmail@example.com

If everything looks correct, you're good to go!

**Additional options**

Let's see some additional settings you can configure.

* Set your preferred editor for writing commit messages:

git config --global core.editor "code" # VS Code example

* Change the default branch name from master to main (or any preferred name):

git config --global init.defaultBranch main

* Prevent Git from tracking permission changes in files:

git config --global core.fileMode false

* Configure line endings to avoid conflicts across different operating systems:
* git config --global core.autocrlf true # Windows

git config --global core.autocrlf input # macOS/Linux

* Create shortcuts for frequently used Git commands:
* git config --global alias.co checkout
* git config --global alias.br branch

git config --global alias.st status

There are many more settings that you can configure for git. You can view all of them [in the documentation.](https://git-scm.com/book/ms/v2/Customizing-Git-Git-Configuration)

**Conclusion**

Let's sum up what you've learned on this topic:

* Git is a set of console utilities to store, control, change, and merge your code.
* You can easily install it on Windows, Linux, or macOS.
* With Git, you can keep your projects either locally on your hard drive or use an online service.

Now, since you are familiar with Git, you might be interested to discover more about it in the next topics. But first, let's get to some tasks!

Read more on this topic in [Bridging the Gap by Understanding DevOps](https://hyperskill.org/blog/post/bridging-the-gap-between-development-and-operations-by-understanding-devops) on Hyperskill Blog.

You are working in a development environment and need to configure Git with your user details. Your task is to write the Git commands that will set up the user name and email globally.

Write the necessary Git commands below line by line to achieve the following objectives:

* Set the global Git user name to "hyper".
* Set the global Git user email to "hyper@hyperskill.org".

# Write your solution below

# Ensure to type the commands line by line

git config --global user.name "hyper"

git config --global user.email [hyper@hyperskill.org](mailto:hyper@hyperskill.org)

What two kinds of repositories does git have? Remote and local

Git allows storing data

The correct answers are:

✅ **on a special online service**  
✅ **on your hard drive**

### Explanation:

* ✅ **On your hard drive**: Git is a **distributed version control system**, so a full copy of the repository is stored locally.
* ✅ **On a special online service**: Services like **GitHub**, **GitLab**, or **Bitbucket** allow you to store and collaborate on Git repositories online.

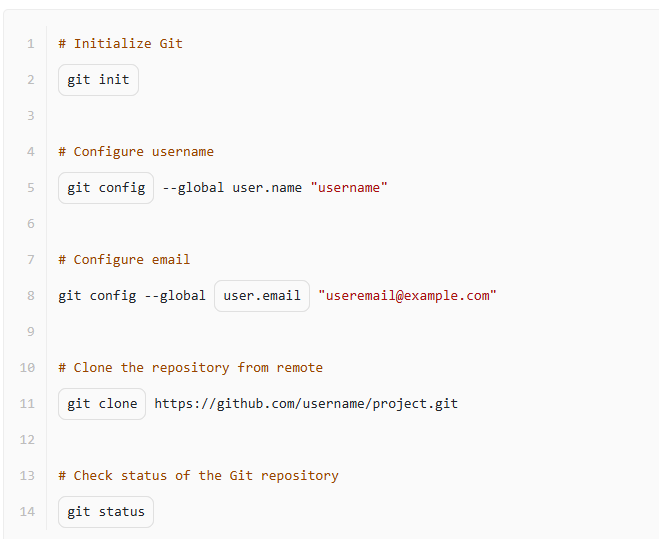
❌ **Only on a single central server**: Incorrect — Git is distributed, not centralized.

❌ **Only on GitHub**: Incorrect — Git works with many platforms, not just GitHub

## Setting up a new project with Git

 Report a typo

You've just started a new software project within a team at your company. Your team is using Git for version control. The project repository link is 'https://github.com/username/project.git'. Your task is to set up your local environment to work with this repository. Fill in the blanks in the given code sequence to initialize Git, set your username and email globally, clone the repository, and check the status of your Git repository.



**What is Git?**

What can you do with Git?

add files to GitHub

track changes in your code

record changes in files

**What is Git?**

Git is

a set of console utilities to store, control, change, and merge your files

## Configuring aliases for git commands

You are working in a development environment and need to configure Git with specific settings. Your task is to write the Git commands that will set up useful aliases.

Write the necessary Git commands below line by line to set aliases for the following commands:

* Create an alias "re" for the "remote" command;
* Create an alias "a" for the "add" command;
* Create an alias "cm" for the "commit" command;
* Create an alias "pu" for the "push" command;
* Create an alias "co" for the "checkout" command;
* Create an alias "br" for the "branch" command.

# Write your solution below

# Ensure to type the commands line by line

git config --global alias.re remote

git config --global alias.a add

git config --global alias.cm commit

git config --global alias.pu push

git config --global alias.co checkout

git config --global alias.br branch

Let's check if you've successfully installed git. Run the git --version command in terminal and write the output.

Write the answer in the **following manner**: git version <version number>

For example, git version 10.28.0

git version 2.47.1.windows.2

Find mistakes in these commands for setting name and email for git.

git --config --global user-name "Kaitukino"

git --config --global user-email kaitukino@mera.whenua

### ✅ Correct Answers:

* ✅ **user-name and user-email instead of user.name and user.email**
* ✅ **--config instead of config in both lines**

### ❌ Not correct:

* **user-name and user-email instead of --user-name and --user-email**  
  🔹 Git config keys are not preceded by double dashes (they’re not flags).
* **--global instead of global in both lines**  
  🔹 --global is correct and used properly in the original command.

## Configuring git options

 Report a typo

You are working in a development environment and need to configure Git with specific settings. Your task is to write the Git commands that will set up the core editor, initial branch name, and line endings globally. Write the necessary Git commands below line by line to achieve the following objectives:

* Set the global Git core editor to "vim";
* Set the global initial branch name to "development";
* Set the global line endings for a Windows system;

Write a program

# Write your solution below

# Ensure to type the commands line by line

git config --global core.editor "vim"

git config --global init.defaultBranch development

git config --global core.autocrlf true

**Setting up a new project with version control**

You have the task to initialize a new project using version control. Your team uses Git for these purposes. However, you've gotten mixed up and the important commands are randomly ordered in your notes. Your task is to correctly order the commands for initializing a new Git repository, adding all files to the staging area, committing those changes with a message of 'Initial commit', and checking the status of your Git repository.

