## **Introduction**

Imagine you and your dream team are developing a large and complex project. You decided to divide it into parts so that everyone could concentrate on doing one of them. In the end, you plan to join forces. What is the best way to do it? The simplest approach is to set up a remote repository to which everyone will upload their versions. This way, you won't be confused by the new code versions and will not lose important features.

In other words, when someone completes one part of the project, the new data is uploaded to the remote repository. After you've updated the data, the other team member may download it and continue working on it.

There are useful commands: **git clone**, **git push**, and **git pull**,to upload and download data from there. Let's take a look at how they all work.

## **Git clone**

You can start working after you've created a remote repository, for example, on GitHub. For this, you should clone the repository to your computer. You can do it with the

git clone command. The syntax is really simple:

$ git clone <remote url>

For example, imagine there is a fictional GitHub account for team called

dream\_team, where they collaborate on all their projects. It contains a new repository: project. Here is how you'd clone the repository:

$ git clone https://github.com/dream\_team/project.git

The syntax is as follows: first, we type the command git clone and then add the address of the remote GitHub-hosted Git repository. Once you clone it, you will get the whole repository with all its content. The example abouve used HTTPS. You could also use SSH or the GitHub CLI to clone the repo. Also, you can also download the source code as a .zip

file and extract it locally. Here is how the URLs/command would look like in each case:

# SSH

$ git@github.com:dream\_team/project.git

# HTTPS

$ https://github.com/dream\_team/project.git

# GitHub CLI command

$ gh repo clone dream\_team/project

For SSH, you need to [create](https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent) SSH keys and [add the public key](https://docs.github.com/en/authentication/connecting-to-github-with-ssh/adding-a-new-ssh-key-to-your-github-account) to your GitHub account before you can use it. Here is how you generate SSH keys:

$ ssh-keygen -t ed25519 -C "youremail@example.com"

You will be prompted for a location to store the keys and an optional secure passphrase. Once you generate the keys, retrieve the public key from the folder you stored your keys in. On Linux, you can use the cat command:

$ cat /home/username/.ssh/id\_ed25519.pub

ssh-ed25519 JJJKC3NzaC1lBDI1NTE5LKMJIHGbatN+FN3zbbpFOyNFKnSxW1ezasNSFKPLCDAS1b6 youremail@example.com

Copy this key and add it in your GitHub [account settings](https://github.com/settings/keys). For the GitHub CLI, you need to [install and authenticate](https://github.com/cli/cli?tab=readme-ov-file#github-cli) it with your GitHub account before you can use the command. HTTPS is the easiest to use since no additional setup is required here. However, in some cases, other options are more suitable.

When cloning, you don't have to clone the entire repository. For instance, with the

--depth option, you can specify that you only need a certain number of commits. With the

--branch option, you can specify that you want to clone a specific branch. Check out the [official documentation](https://git-scm.com/docs/git-clone#_options) to see all the options you can use.

Cool, now you have a working area, you can now start/continue coding!

## **Git push**

When you use Git push, you send commits and branches to a remote repository on the server. That is, every time you make some changes to the code and commit them, you then upload the updated data to the server. This way, your colleagues will always see the current version of the code. Also, it is essential to have such a backup in case of data loss on your computer.

When pushing to the remote repository, on GitHub in our case, authentication and authorization is required. The authentication method depends on whether you are using HTTPS or SSH. For HTTPS, Git will ask for a username and password. Instead of your password, you will need to enter a **personal access token**.

Review the topic on [Personal Access Tokens](https://hyperskill.org/learn/step/40645) for more information on setting them up.

If you used SSH, good news, no additional configuration is required. Your SSH key pairs (public and private keys) will be used for authentication.

The command syntax is as follows:

git push origin <branch\_name>. This means you push the data to the remote server (origin) and push it to a concrete branch. In the example below, this is the main branch:

$ git push origin main

If you want, you can push to another branch. If there is no such branch on the server, then it will be created. If you have already pushed to it before, then new commits will simply be sent there, and in this case, you can use short versions of the command: just git push.

If you have not previously set an upstream branch, Git will prompt you to do so. To explicitly set an upstream branch, use:

$ git push --set-upstream origin <branch\_name>

Remember, before pushing changes, you must first commit them.

Okay, now you've pushed your changes to the server. But what about the opposite situation when you need to download them from the server? You've probably guessed it: you may clone the repository one more time by

git clone, but it will copy all the changes ever committed. What if you are interested only in the latest changes that are more current? Then you should use the git pull command.

## **Git pull**

Git pull is downloading data from the server and only the most recent ones. This way, you won't get confused about the commits and will be sure you only copied the freshest and most current of them. So, you just do:

git pull origin <branch\_name>, i.e., you download the latest changes from the remote repository branch on the server. In the example, we download data from the main branch.

$ git pull origin main

You can choose all the branches you need and use a short version of the command. Just write a

git pull if the local branch already belongs to some remote branch from which you are downloading the changes.

Awesome, now with all the latest changes, you can start making your own and push them again. And so on in a circle, until the project is finished. In a large project with a lot of constant changes, there may arise some problems. Let's discuss one of the most frequent ones below to understand why it happens and how it can be solved.

Sometimes, you will make a new commit, try to push it, and git throws you a

git push rejected error in response. This happens to you if someone has made changes you didn't pull yet. Before you send your commits to the server, you need to pull new commits from it. The ones that your colleagues did. That is, do a git pull, and the problem is solved!

## **Conclusion**

Now you know how to clone a remote repository from the server using the

git clone command, how to upload changes to the server using the git push command, and how to download the most recent updates using the git pull command.

George wants to send his work to the server and put it in the current\_tasks branch. What command should he use in this case?

git push origin current\_tasks

## 

## **Share your changes**

Nika made some new changes to the code and committed them. What command should she use now to upload these changes to the server?

git push

## **Git push error**

Marina made a git push to send her changes to the server. But Katie had already made many new commits before her. As a result, Marina got an error message git push rejected. Which command will help Marina fix everything and repeat the submission later?

git pull

Maria, Pierre, and Denis have created a remote repository. Maria needs to copy it in order to leave the first commit there. What command can she use to download the repository from the server?

git clone

Why is it convenient to use a remote repository?

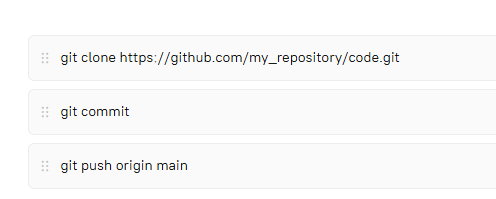
It stores all the project data in one place

All team members have access to the current version of the project

Andrew created his own GitHub repository https://github.com/my\_repository/code.git. He wants to perform the following sequence of actions:

* download the repository to his local computer,
* make changes to it and commit them,
* add changes to GitHub in the main branch.

In what order should he execute the following commands?



Complete the command below to correctly download and apply the latest changes from a remote repository for a specific branch called main. Fill in the missing parts to complete the command.

Git pull origin main