After we figured out what Git is, what it is for, and how to install it, we can create our first Git repository, i.e. the file system directory that contains files. As you see, the repository is the space in which we work with Git, so first of all we should create it.

Remember the two kinds of repositories: local and remote? Since at first everything is done locally and only then it is transferred to remote access, we should create a local repository first. In this topic, you will find out how to do it and how to add files, check the repository status, and write your first commit.

Git init

As you may remember, Git stores its files and history right in the project folder. To create a new repository, we need to open a terminal, go to our project folder, and run the

`git init`

command:

```
$ git init
```

This will include the application in that particular folder and create a hidden

`.git`

directory where the repository history and settings will be stored. As a result, you will see:

```
Initialized empty Git repository in /path to the folder with the
repository / repository_name/.git/
```

This means that your repository has been successfully created, but it is still empty. So, why not add some files?

Git add

Now you can create a file or use a program you have already written and save it in the git directory. There can be as many files as you want with different extensions. Here we have two options: you can either add files one by one using

`git add`

command or add all files by

```
git add -A
```

:

```
# adding one concrete file

$ git add my_file.txt

# adding all files from your directory


$ git add -A
```

Great, now we have the files, but that's not all.

Check repository status

To make sure we are doing everything right and also check the changes, you can track the status of the repository using the

```
git status
```

command. For example, now we will check what is happening with our newly created repository. Running

```
git status
```

should give:

```
$ git status
On branch master
Initial commit
Changes to be committed:
(use "git rm --cached ..." to unstage)

new file: my_file.txt
```

The command output shows that we have a new repository that contains a file. Note that you can use this command at any stage of your work with the repository, so you can always check if everything is correct. In the output, there are still concepts that may be incomprehensible, like *On branch master*. We will explain them in the following topics. Now, let's get to the final step of this topic, which is to **commit** the changes to the repository, i.e. adding files there. To do this, we will use the

`git commit`

command.

Git commit

A **commit** represents the state of the repository at a specific point in time. This way, each commit fixes the changes and makes it easier to track them later. To commit the changes, we need at least one staging change, for example, adding new files, as we did before. After that, we type in:

```
$ git commit -m "First commit."
```

This command will create a new commit with all the staging changes. The

`-m`

stands for "message" and the

`"First commit."`

is this message or, in other words, the comment you add to the commit. It is considered good practice to commit frequently and always write meaningful comments for this makes it easier for you and others to understand and to remember what you did.

Conclusion

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

- to create a new repository, use
- `git init`
- command
- to add files to your repository, use
- `git add <filename.extension>`
- or
- `git add -A`
- to add all files from your directory
- to commit your changes, use

- `git commit`
- and do not forget to write comments!
- to check the repository status at any time, use
- `git status`
- command