git add remote [remote name][remote url] - kachva na dr miasto a ne na github

git push [remote name] master

git init – pravi prazno hranilishte (predpolagam se polzva ako ne polzvat kato men git bash here, a ako se polzva gitbash here to avtomatichno samo si pravi tuy hranilishte)

# How To Github

In this tutorial, you will learn:

* What is a version control system
* What is Github
  + Create a github account
    - Create and manage a repository
  + Create an organization
    - Invite other users
    - Assign and resolve issues
* What is Git Bash
  + Installation and setup
  + Clone, commit, push and pull commands

## What is a version control system

Creating a large project alone is practically impossible. Large applications are developed by hundreds of programmers over many years. All those people are working on the same code, which leads to many problems.

1. Which is the latest version?

2. Which version is still tested, and which is ready for release?

3. After I implement a new functionality/fix a bug, how can I send the new source code to everyone else? Do you think sending a thousand rar files each day is a practical solution?

To tackle with problems like these, multi-developed projects use **version control systems.** Normally, those systems provide a number of useful features:

* Code repository – everyone works on the same code and commits changes in real time.
* Version history – if something fails, you can always rollback to a stable version
* Cloning – creates a copy of the repository on your local file system
* Commit && push – sends your latest changes to the online repository
* Pull/Update – gets the latest changes done by others to your local repo

More information: [Version control](https://en.wikipedia.org/wiki/Version_control), [Git](https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control)

## What is Github

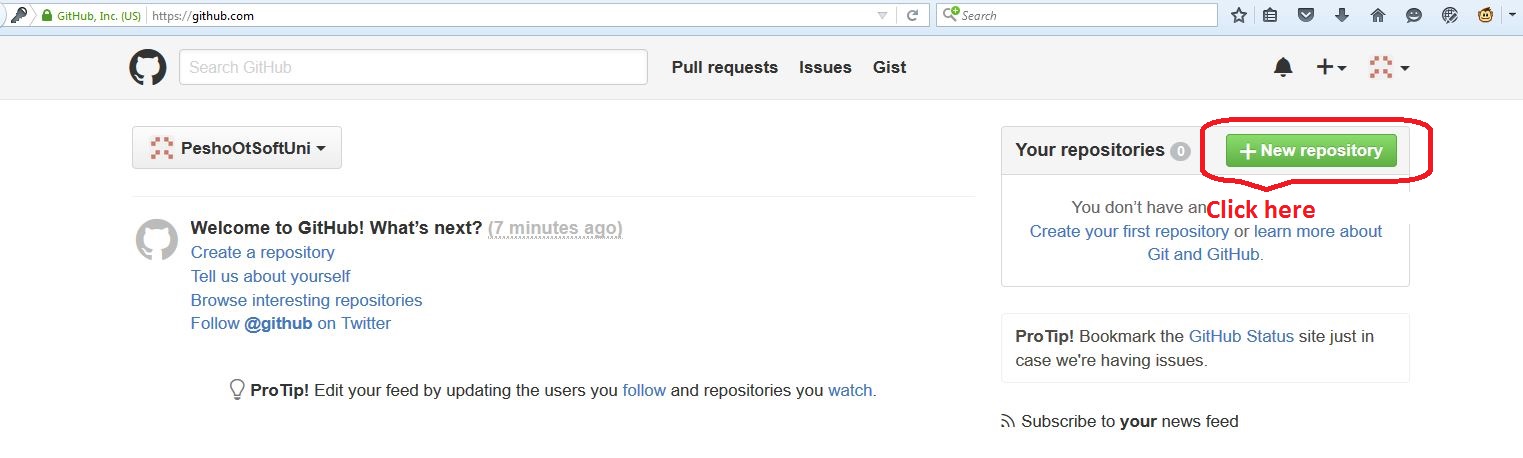
In a few words, this is the most popular repository hosting service. It provides all the functionality described above, on top of a number of other useful features. In addition, public repositories are completely free of charge – you can have as many as you want.

### Create a Github account

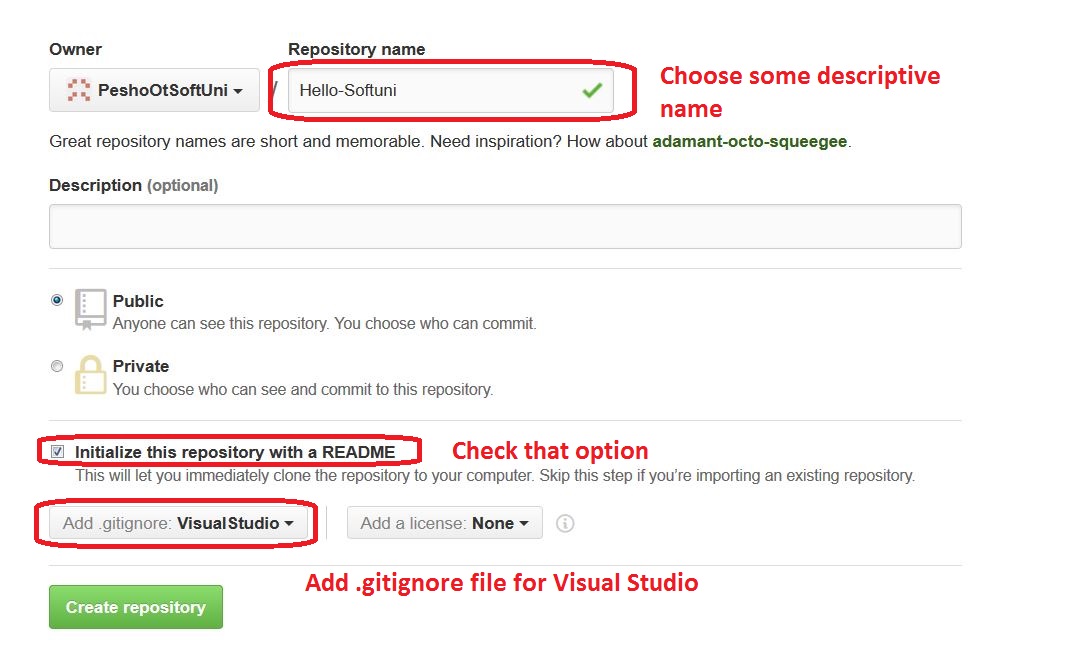
Go to [github](https://github.com/) and create an account. It’s trivial. After filling up username, email and password, you will be asked for a payment plan. The ***free*** option is checked by default – there is absolutely no reason to choose any other option.

### Create a repository

A repository is like an online directory where you can store your project along with its source code, configuration, subdirectories, etc. Creating a repository is quite easy:



You will be prompted to verify your email address. Go to your inbox, find the mail from github, and click on the verification link.



Adding a .gitignore file is important. There is no need to commit libraries, executables, or user-specific files. Github provides a visual studio .gitignore file out of the box, with all the necessary configurations done. There are other options as well – all of the popular IDEs and languages are covered.

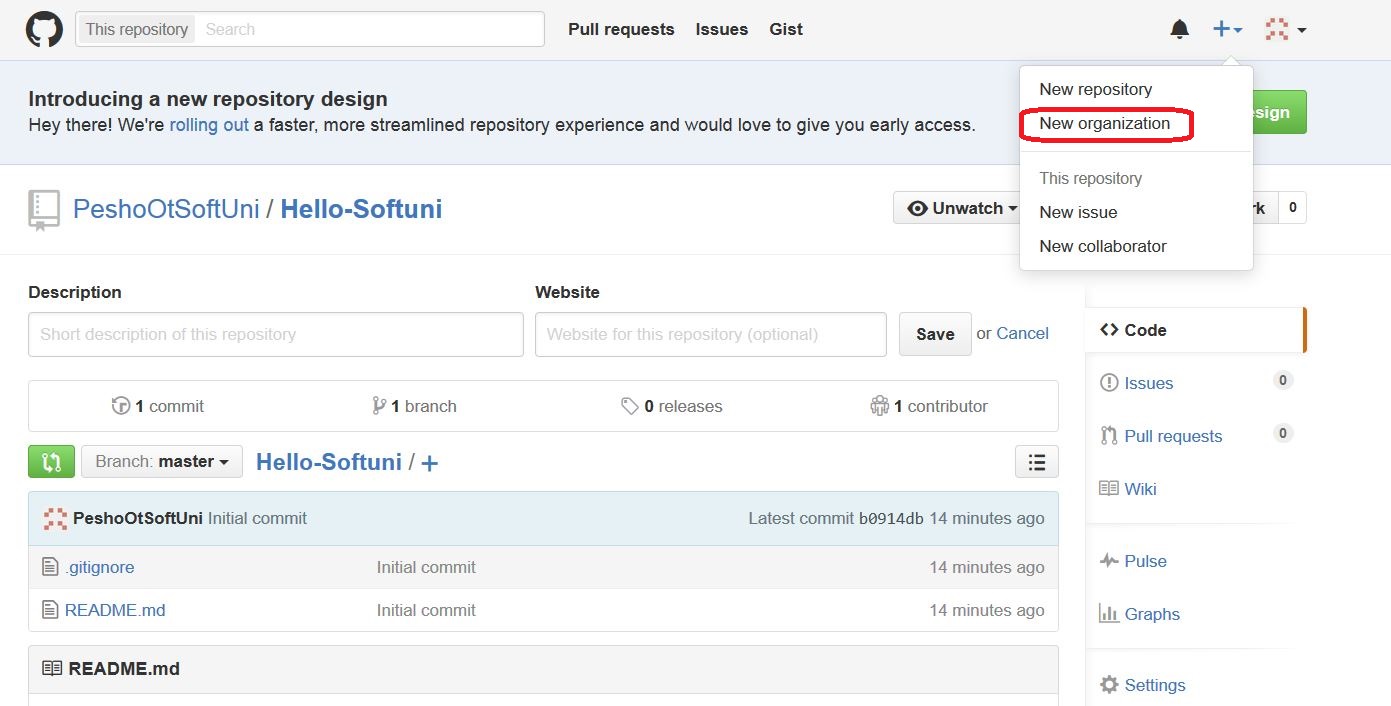
Click on **Create repository.** That’s all the work required. You can find your repo at

**https://github.com/{your-username}/{your-repo-name}**

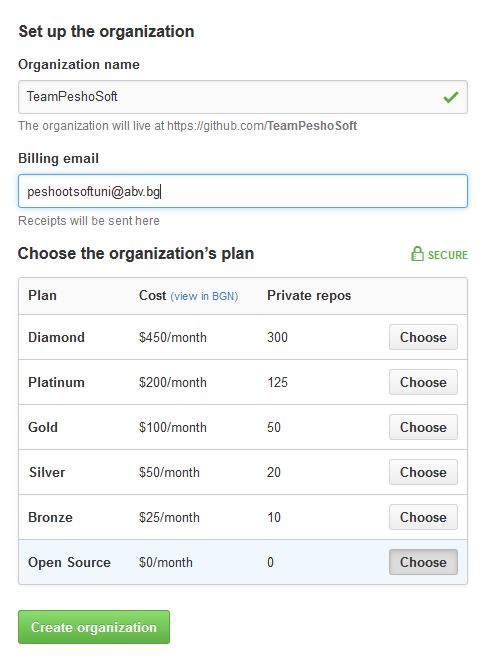
You can also navigate to it from your github profile.

### Create an Organization

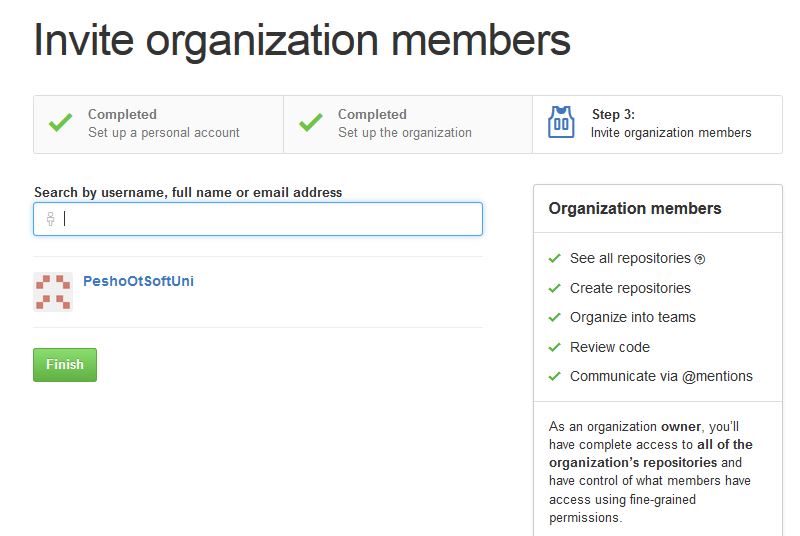
Organizations are very useful when working on a teamwork project – they allow you to share a repository with your teammates. Setting-up an organization is really simple – it’s done in a few clicks.



Choose some descriptive name. In the email field, you normally put the email address you registered in github. It’s not the important since we will be creating an Open Source organization.



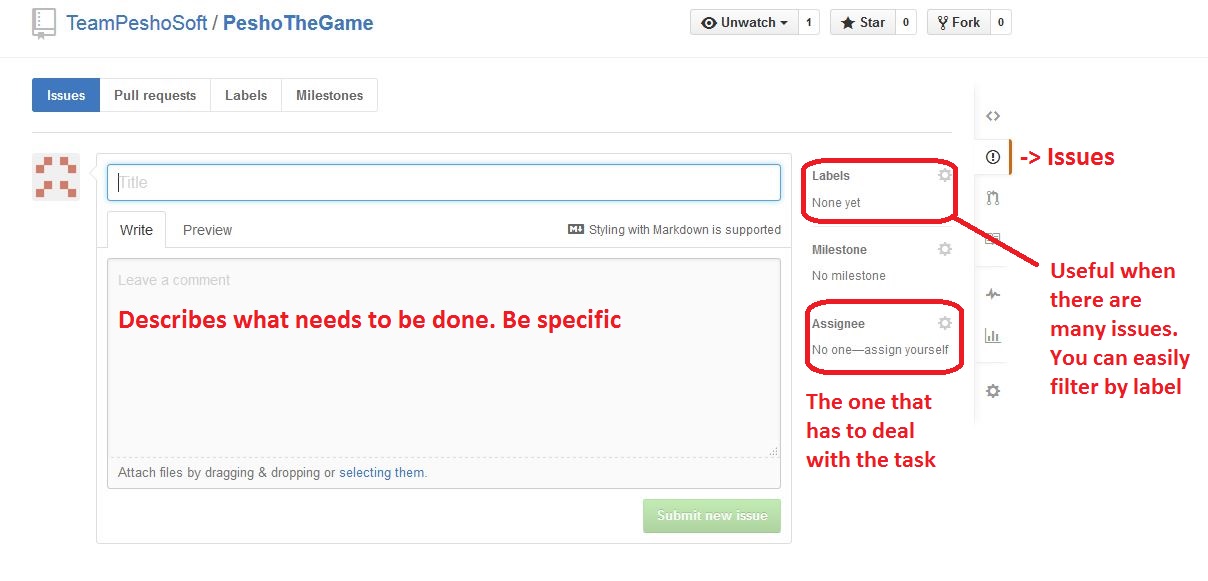
Next you have to invite the other members. They will receive an **email** prompting them to accept the invitation.



Clicking on **Finish,** the setup process is completed. You can now create the organization’s repository – the steps are the same as the ones described above.

#### Issues

Issues are a handy feature that helps you organize your team’s tasks. Essentially, they describe what needs to be done and have an ***assignee*** – the one who must do the task. Issues can also describe a bug that needs to be fixed, or whatever – it’s up to you to decide who has to do what.



Once an issue is submitted, everyone in the organization will receive a notification. Once an issue is resolved, you can close it. Normally, as a comment you put the ID of the commit that fixed the bug/implemented the functionality.

## Git Bash

Git Bash is software that allows you to upload and download (***commit/pull***) code from your computer to your repositories and vice-versa. It’s fast and easy to use. Go to [get git bash](https://git-scm.com/downloads) and download the tool.



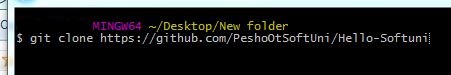
Once downloaded, install the client. **Go with all the default options and just click Next until installed** :)

Go to the folder you want to turn into a local repository, right-click anywhere in it and choose **Git Bash here.(ey tuy smotano neshto triabva da e tochno v papkata deto na sledvashtata stupka ti kopira failovete, inache dava greshka ako ne sa v edna I sushta papka!!!)**

Let’s say we want to have a copy of Pesho’s Hello-Softuni repo. The command to do this is **git clone +** the URL of the repo -git clone ti opravia putishtata inache se bugva ako ne se napravi taka!!!!

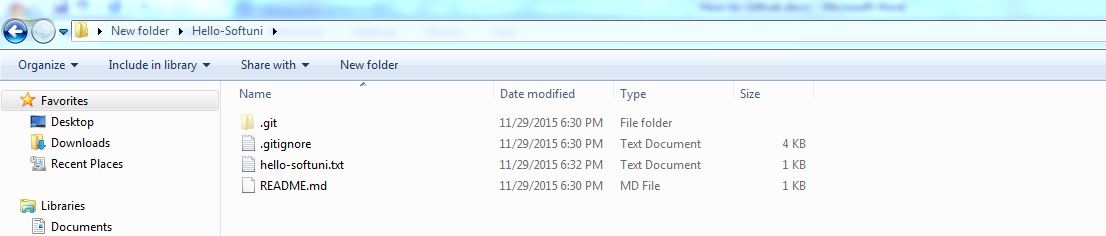
//ako iskame samo branch-a da klonirame

git clone -b imeto-na-brancha-ni I adresa na mastera ( napr. <https://git@github.com/username/myproject.git> )

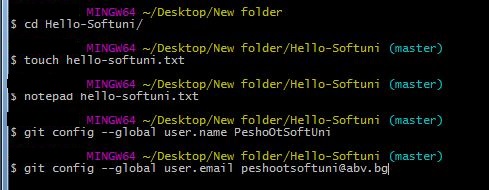


Eto tuk si pishe che triabva I newfolder imeto!!!

Go to the folder and create a .txt file with some text in it.



We will commit that file to the repo. But before that, Git Bash should know who is doing the commits. There are two simple configuration commands that we need to run.



You may skip the first three steps and create the hello-softuni.txt file the normal way. The important commands to run are

**git config --global user.name <username>**

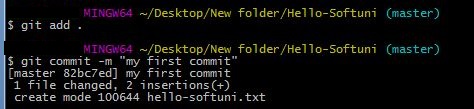
**git config --global user.email <email>**

Now we can commit. The command to add all the newly created files to the repo is

**//triabva da si smenim direktoriata s ~Desktop/nashatapapka/ MNOGO e VAJNO (stava s cd)**

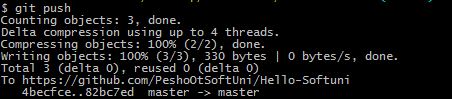
**git add .** (the dot “.” Means “all”). If you want to delete a file, just delete it and run **git add –A**

**git commit –m “my first commit”** uploads everything to the local repo. **–m <message>** is the message that will be displayed in github.



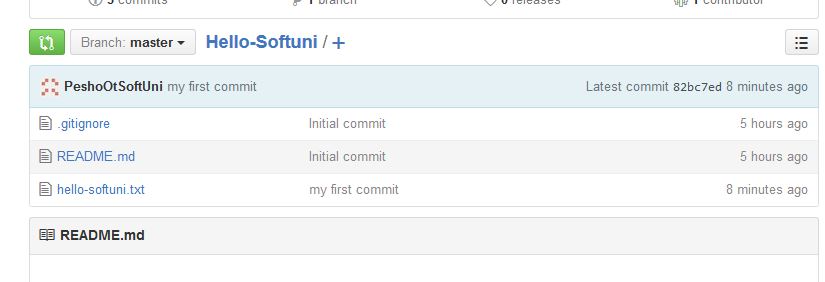
Now everything is in the local repo, but we need to send it to github’s repo. The command for that is

**git push**

****

You will be prompted to enter your username and password in github. You can only commit to your own repos, your organization’s repos, or if someone gave you access rights, to their repos.

Let’s check Pesho’s repository.



As you can see, the file is there and the message is there. All is going well.

When working in a team, the command to get the latest commits done by your teammates is **git pull.** Always run **git commit** before **git pull**, else you might lose your local changes (git bash won’t allow you anyway).

# Exercise

For this exercise, you will need a partner. The guy sitting next to you will do perfectly.

1. Create an organization and invite him to join it.
2. Create a repository for your organization.
3. Clone the repo to your computer, create a file and commit it.
4. Let your partner pull the changes. He/she should get the file you created.
5. Let him change something in the file and commit the changes.
6. Now it’s your turn to pull.
7. Create an issue and assign your partner to it. It may be something simple like “Create file “yolotrololol.txt”
8. After your partner creates the file, let him resolve the issue and post the commit id. The message can be something simple like **Closed via #82bc7ed**

