

iOS Professional GitHub Setup

Git is the version control system most teams and companies use to store and collaborate on source code.

This document will walk you through how to set up your own git repos, so you can practice and learn the necessary git commands you will need to work professionally in the workplace.

[Create your account](#)

[Create your repository](#)

[Checkout your repository](#)

[Your first commit](#)

[Creating a personal access token](#)

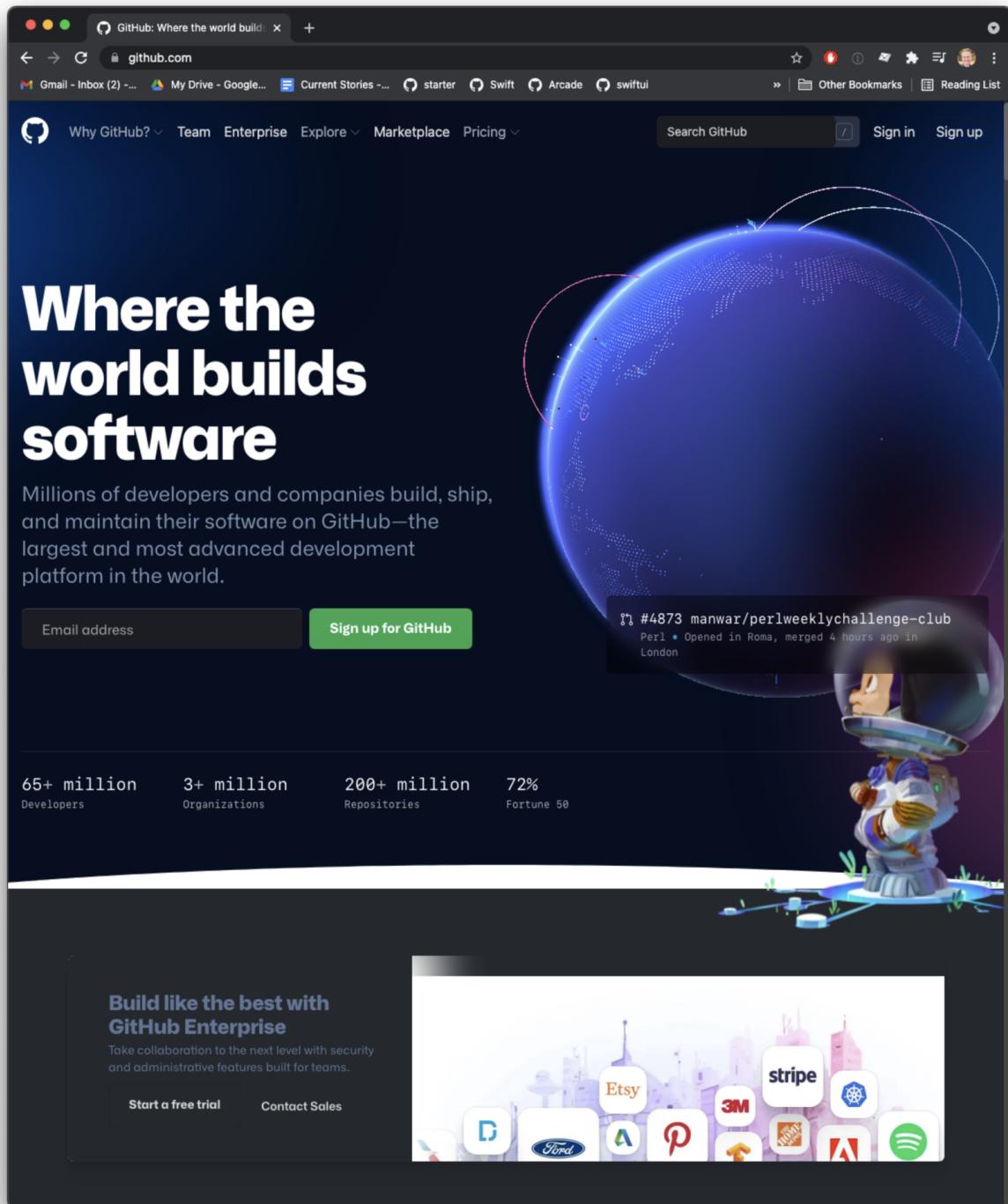
[Installing GitHubCLI](#)

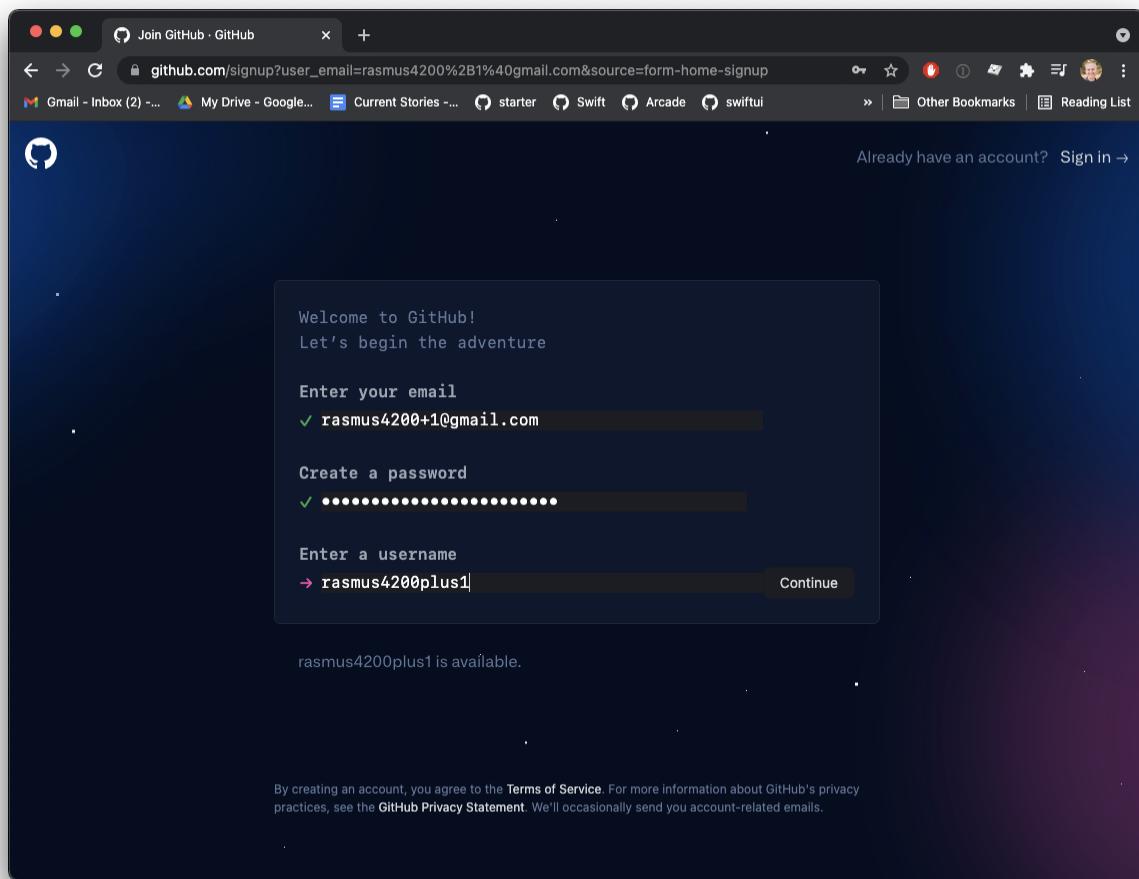
[Enter GitHub Token](#)

[What should I do when my token expires?](#)

Create your account

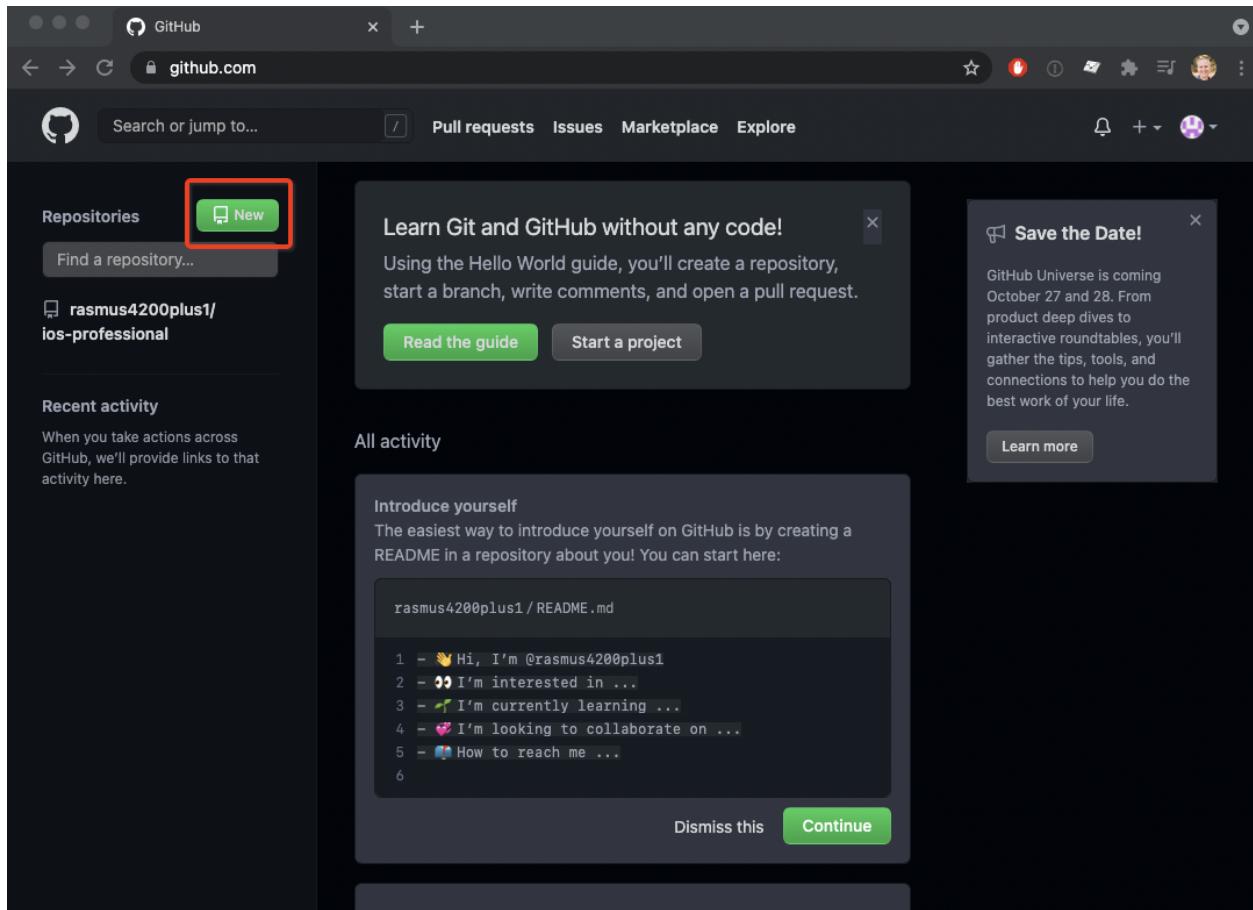
<https://github.com/signup>





Create your repository

Next let's create your first repository.



Create a New Repository

github.com/new

Search or jump to... Pull requests Issues Marketplace Explore

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

Owner * rasmus4200plus1 / ios-professional ✓

Great repository names are short and memorable. Need inspiration? How about [friendly-telegram](#)?

Description (optional)
Repository for professional iOS work

Public Anyone on the internet can see this repository. You choose who can commit.

Private You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

Add a README file This is where you can write a long description for your project. [Learn more](#).

Add .gitignore Choose which files not to track from a list of templates. [Learn more](#).
.gitignore template: Swift ▾

Choose a license A license tells others what they can and can't do with your code. [Learn more](#).
License: MIT License ▾

This will set `main` as the default branch. Change the default name in your [settings](#).

[Create repository](#)

© 2021 GitHub, Inc. Terms Privacy Security Status Docs

Contact GitHub Pricing API Training Blog About

The screenshot shows a GitHub repository page for the user 'rasmus4200plus1'. The repository name is 'ios-professional' and it is described as 'Repository for professional iOS work'. The page includes sections for 'About', 'Releases', and 'Packages'. At the bottom, there are links to GitHub's Terms, Privacy, Security, Status, Docs, Contact GitHub, Pricing, API, Training, Blog, and About pages.

About
Repository for professional iOS work

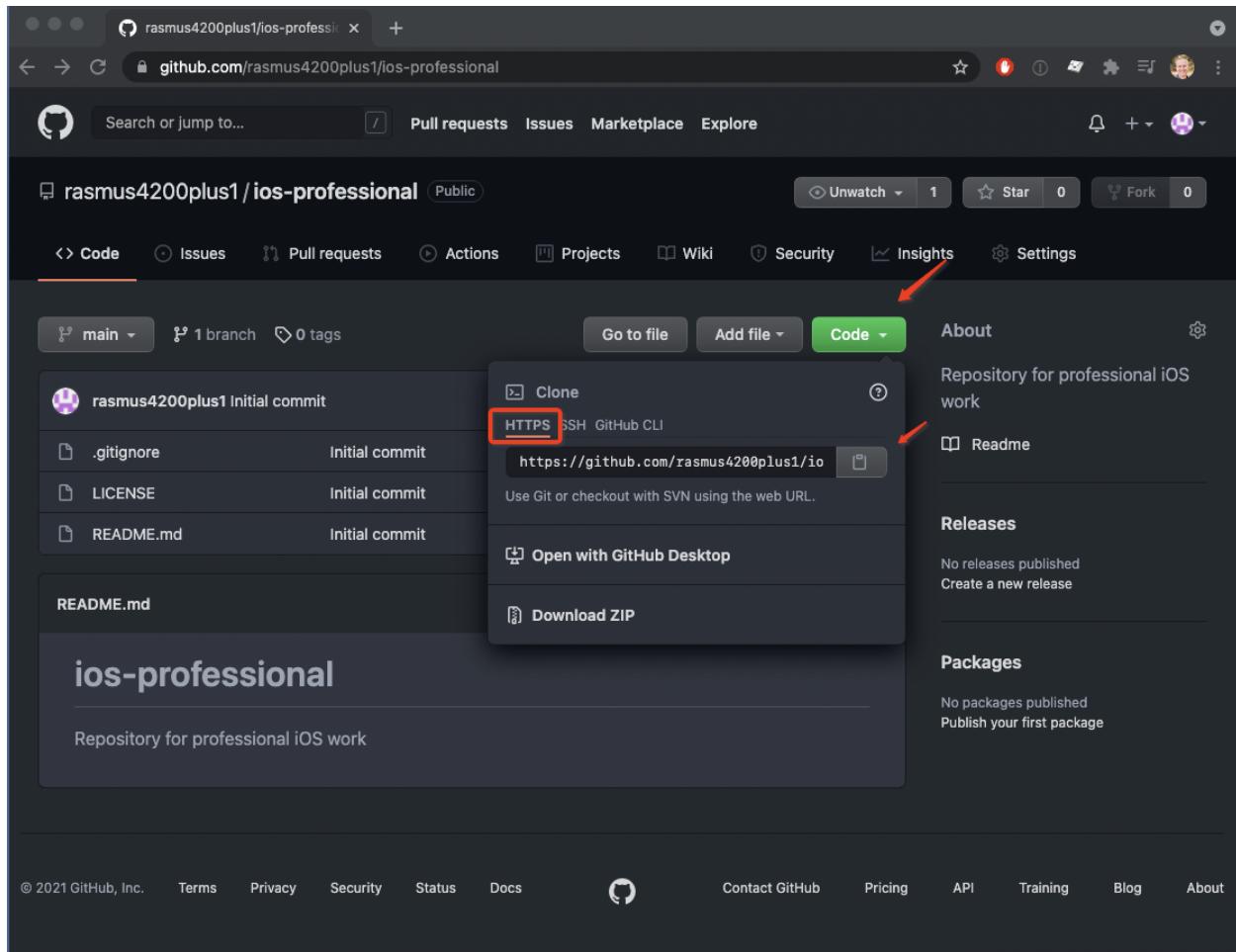
Releases
No releases published
Create a new release

Packages
No packages published
Publish your first package

© 2021 GitHub, Inc. Terms Privacy Security Status Docs Contact GitHub Pricing API Training Blog About

Checkout your repository

Now that we've created your repository, let's check it out locally to your machine.

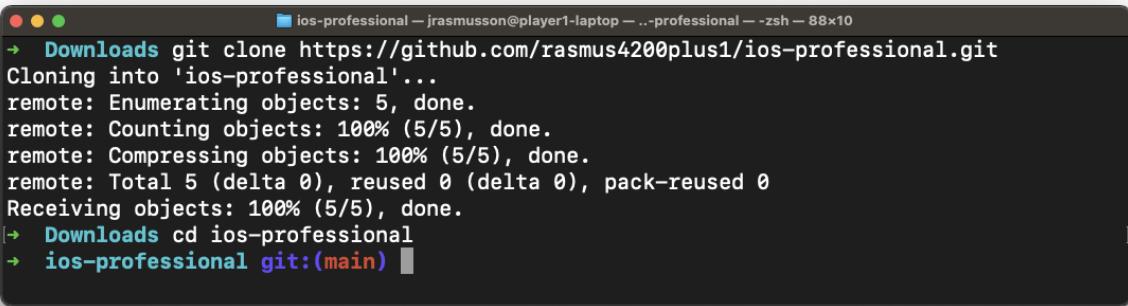


Click Code, HTTPS, and then copy the path to your repos.

Then open a terminal on your mac. Navigate to the directory where you want to checkout your repos, and clone your repos with the following command.

```
> git clone <repos path>
> cd ios-professional
```

And if all goes well you should see your repos on the branch called **main**.



A screenshot of a macOS terminal window titled "ios-professional — jrasmusson@player1-laptop — --professional — -zsh — 88x10". The window contains the following text:

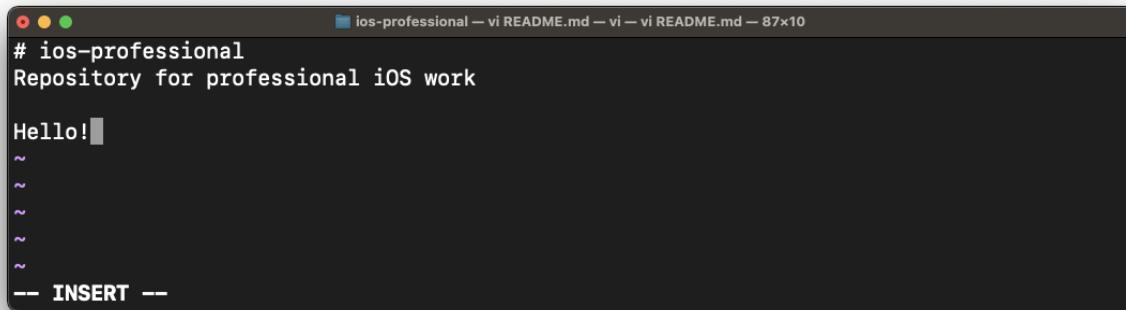
```
↳ Downloads git clone https://github.com/rasmus4200plus1/ios-professional.git
Cloning into 'ios-professional'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
[↳ Downloads cd ios-professional
↳ ios-professional git:(main) ]
```

Your first commit

To make a commit against your repository we first need to:

- Change a file
- Add it to our branch
- Commit it
- And then push it up to github

To change a file, open up the README.md file and make a change (i.e. add some text).

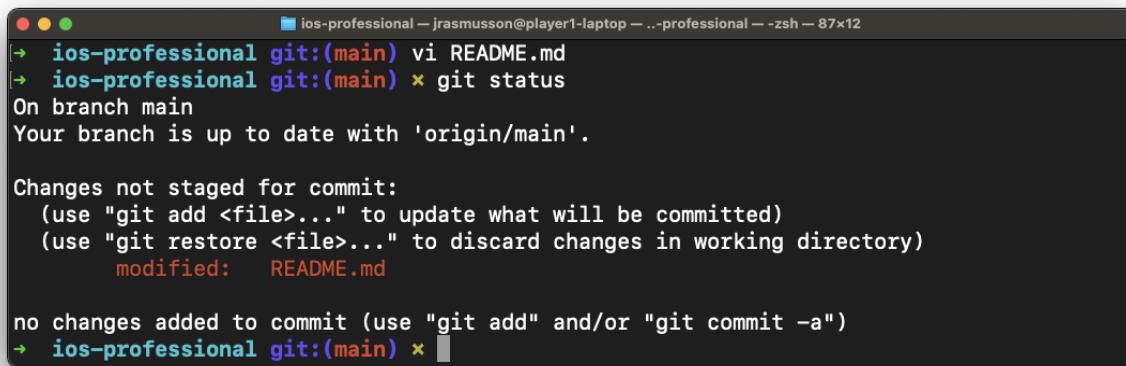


```
# ios-professional
Repository for professional iOS work

Hello!~^~^~^~^~^
-- INSERT --
```

You can now see that a file has been changed on your repos by typing

```
> git status
```



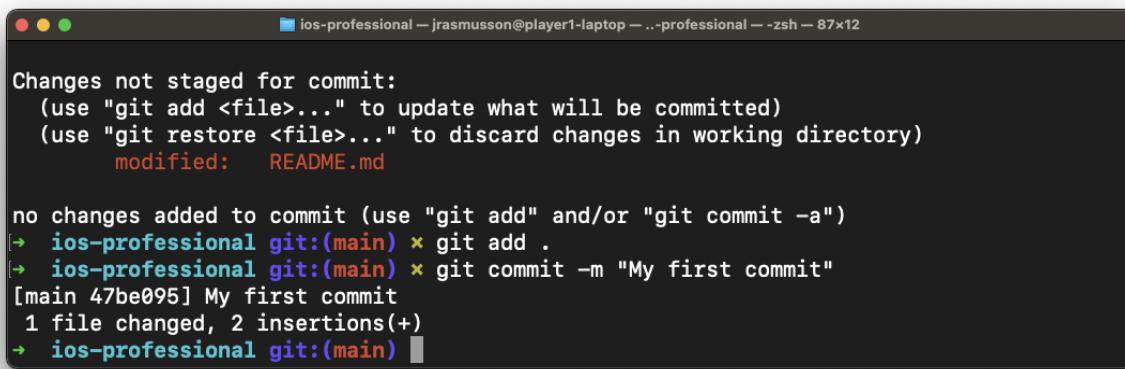
```
→ ios-professional git:(main) vi README.md
→ ios-professional git:(main) ✘ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
    (use "git restore <file>..." to discard changes in working directory)
      modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
→ ios-professional git:(main) ✘
```

To add and commit that file type

```
> git add .
> git commit -m "My first commit"
```



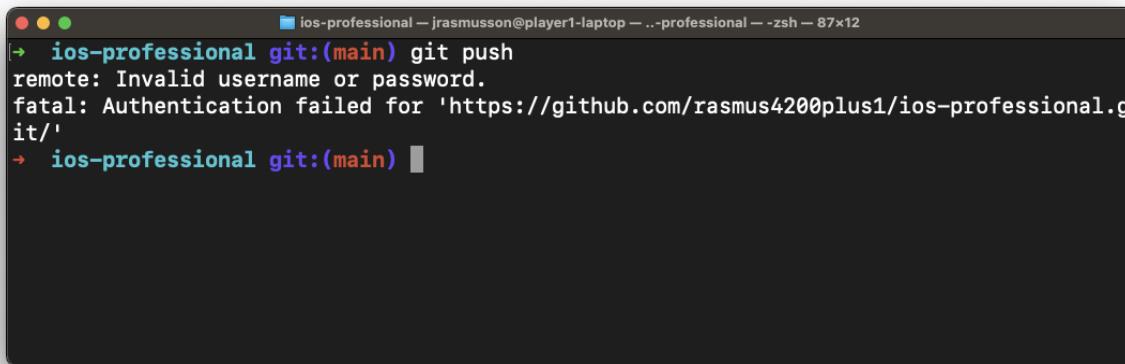
A terminal window titled "ios-professional" showing the output of a git commit command. The output indicates changes were staged for commit, including a modified README.md file. It shows the command history: "git add .", "git commit -m "My first commit\"", and the resulting commit message "[main 47be095] My first commit". The commit summary shows 1 file changed, 2 insertions(+).

```
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
[+ ios-professional git:(main) ✘ git add .
[+ ios-professional git:(main) ✘ git commit -m "My first commit"
[main 47be095] My first commit
  1 file changed, 2 insertions(+)
→ ios-professional git:(main) ]
```

With our changes committed, we are not ready to push them up to github. Try it with the following command.

```
> git push
```



A terminal window titled "ios-professional" showing the output of a git push command. The command fails with an "invalid username or password" error, indicating authentication failed for the GitHub repository.

```
[+ ios-professional git:(main) git push
remote: Invalid username or password.
fatal: Authentication failed for 'https://github.com/rasmus4200plus1/ios-professional.git'
→ ios-professional git:(main) ]
```

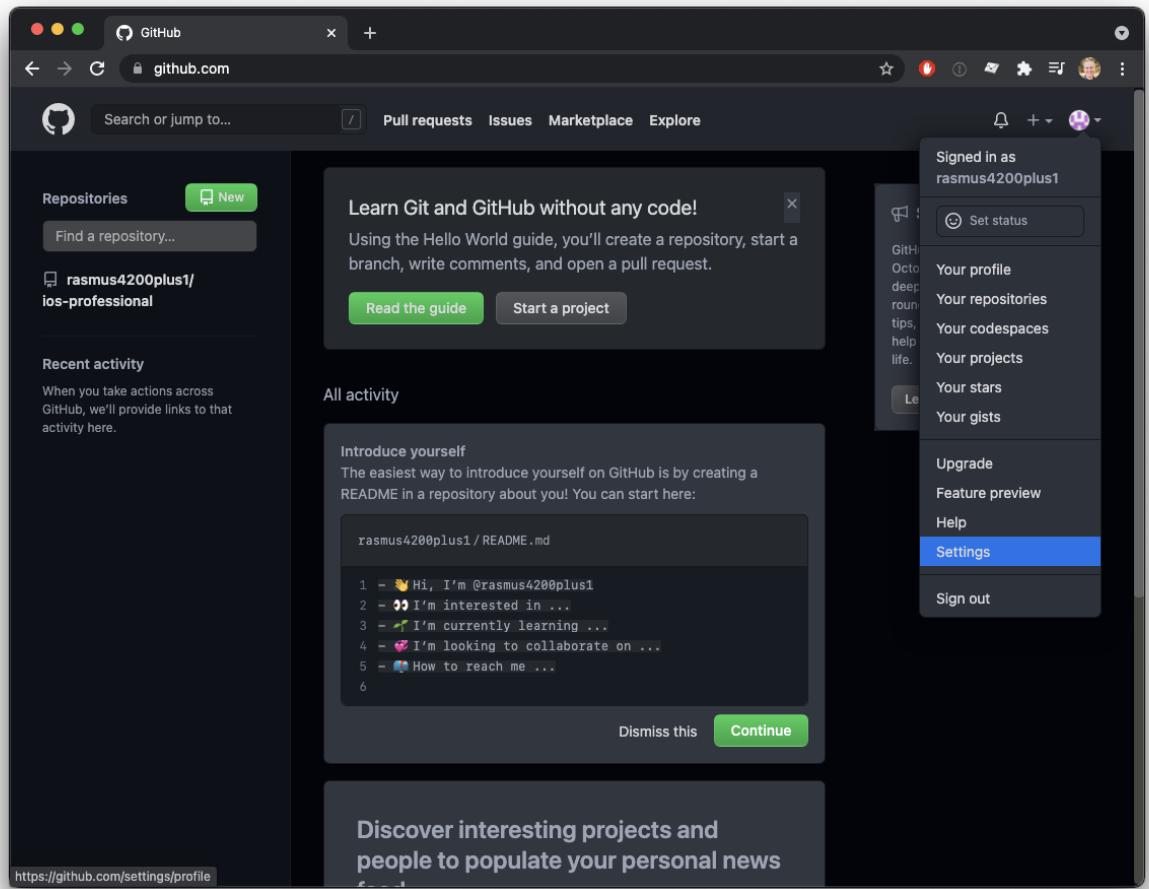
Normally this would work, but because it's our first time, GitHub doesn't know how we are. So before we can do pushes, we first need to get a token.

Creating a personal access token

The steps for creating a token are outlined here. Let's walk through them together.

First, if you haven't already done so, open up your email and verify your email address.

Then from the main github page, click Settings.

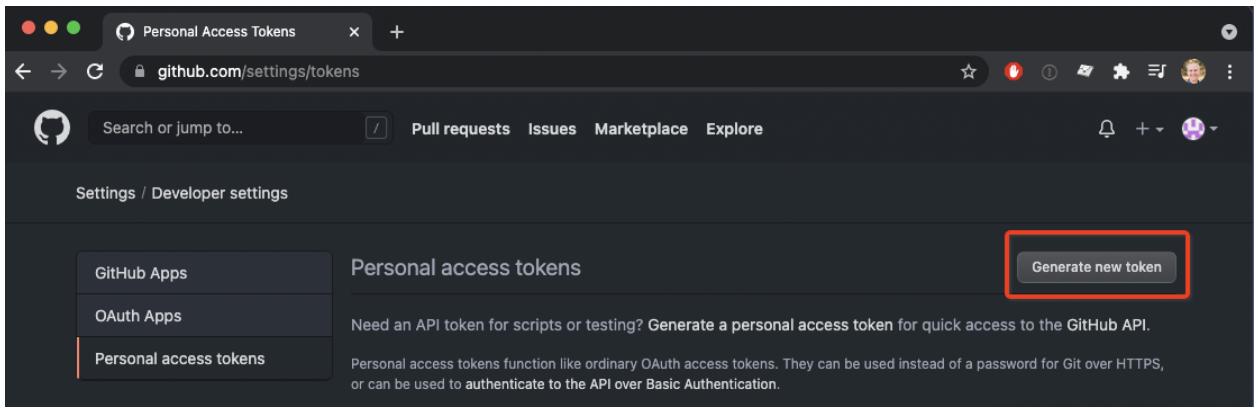


Click Developer settings

The screenshot shows the GitHub Profile settings page. On the left, there's a sidebar with various options like Account security, Billing & plans, Security log, etc. The 'Developer settings' link is highlighted with a red box. The main area contains sections for Public email, Bio, URL, Twitter username, Company, and Location. At the bottom right is a green 'Update profile' button.

Person Access token

The screenshot shows the GitHub Apps settings page. The sidebar has links for GitHub Apps, OAuth Apps, and Personal access tokens, with 'Personal access tokens' highlighted by a red box. The main content area is titled 'GitHub Apps' and contains text about building GitHub Apps. At the bottom, there are links for Contact GitHub, Pricing, API, Training, Blog, and About.



The official documentation for these steps can be found [here](#).

Now we are ready to create our token. Give your token a name, expiration date, and necessary permissions to commit against your repository.

```
> ios-professional-token  
> 90 days  
> repos, workflow, write:packages, admin:org
```

New personal access token

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

Note

ios-professional-token

Expiration •

90 days The token will expire on Wed, Dec 15 2021

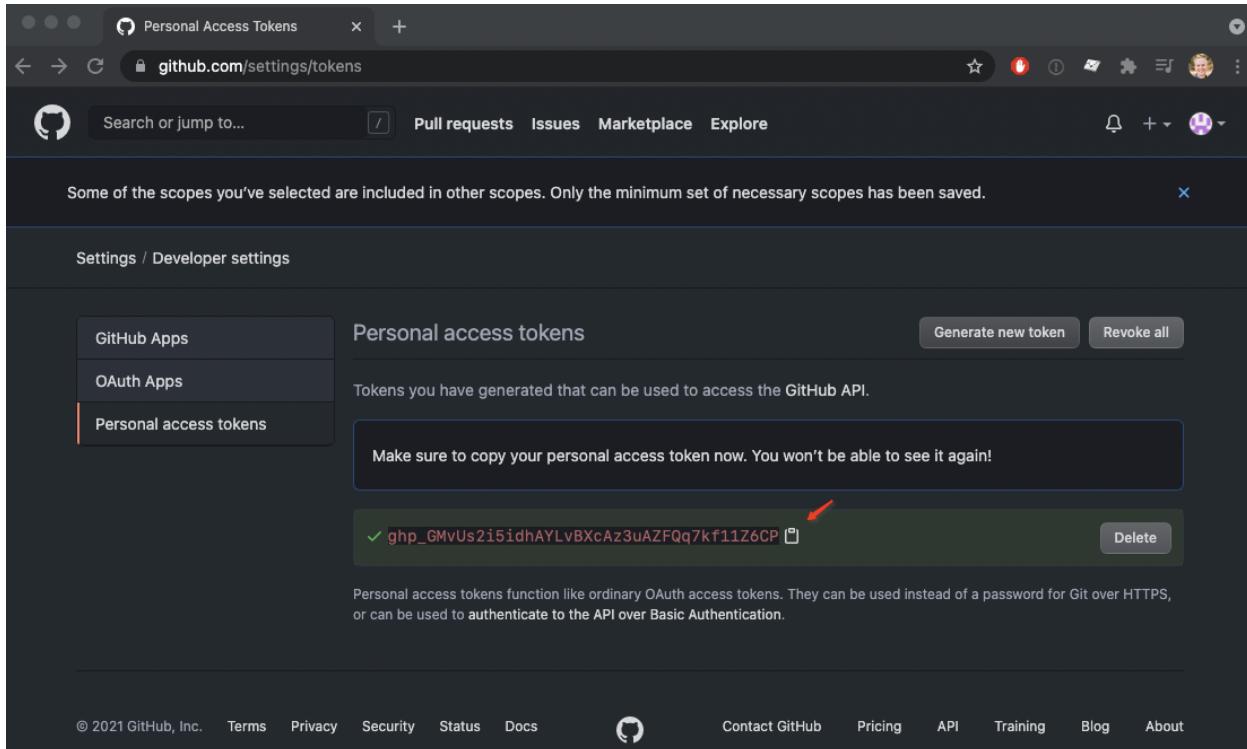
Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

<input checked="" type="checkbox"/> repo	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo_deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input checked="" type="checkbox"/> repo:invite	Access repository invitations
<input checked="" type="checkbox"/> security_events	Read and write security events
<input checked="" type="checkbox"/> workflow	Update GitHub Action workflows
<input checked="" type="checkbox"/> write:packages	Upload packages to GitHub Package Registry
<input type="checkbox"/> read:packages	Download packages from GitHub Package Registry
<input type="checkbox"/> delete:packages	Delete packages from GitHub Package Registry
<input checked="" type="checkbox"/> admin:org	Full control of orgs and teams, read and write org projects
<input checked="" type="checkbox"/> write:org	Read and write org and team membership, read and write org projects
<input checked="" type="checkbox"/> read:org	Read org and team membership, read org projects

Then scroll down to the bottom and click Generate token.





This token is what we are going to use to authenticate against github.

Copy it by clicking the copy button to the right of the code. Save it somewhere (or leave this window open).

And then to use it, we are going to install one more thing. The github command line tool or GitHubCLI.

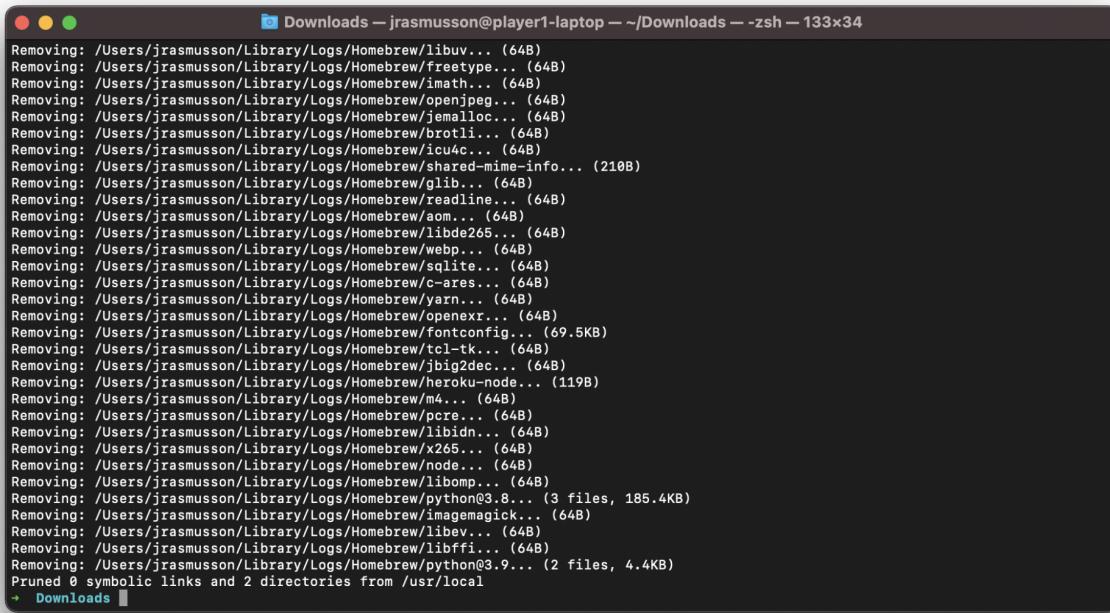
Installing GitHubCLI

In order to not have to enter our credentials every time, we need to cache our credentials somewhere. For this we are going to use GitHubCLI.

Instructions on installation options can be found [here](#).

But for us, we are simply going to use homebrew (a mac installation tool). Open up a new command line terminal and type:

```
> brew install gh
```



```
✖ Downloads — jrasmusson@player1-laptop — ~/Downloads — zsh — 133x34
Removing: /Users/jrasmusson/Library/Logs/Homebrew/libuv... (64B)
Removing: /Users/jrasmusson/Library/Logs/Homebrew/freetype... (64B)
Removing: /Users/jrasmusson/Library/Logs/Homebrew/imath... (64B)
Removing: /Users/jrasmusson/Library/Logs/Homebrew/openjpeg... (64B)
Removing: /Users/jrasmusson/Library/Logs/Homebrew/jemalloc... (64B)
Removing: /Users/jrasmusson/Library/Logs/Homebrew/brotli... (64B)
Removing: /Users/jrasmusson/Library/Logs/Homebrew/icu4c... (64B)
Removing: /Users/jrasmusson/Library/Logs/Homebrew/shared-mime-info... (210B)
Removing: /Users/jrasmusson/Library/Logs/Homebrew/glib... (64B)
Removing: /Users/jrasmusson/Library/Logs/Homebrew/readline... (64B)
Removing: /Users/jrasmusson/Library/Logs/Homebrew/aom... (64B)
Removing: /Users/jrasmusson/Library/Logs/Homebrew/libde265... (64B)
Removing: /Users/jrasmusson/Library/Logs/Homebrew/webp... (64B)
Removing: /Users/jrasmusson/Library/Logs/Homebrew/sqlite... (64B)
Removing: /Users/jrasmusson/Library/Logs/Homebrew/c-ares... (64B)
Removing: /Users/jrasmusson/Library/Logs/Homebrew/yarn... (64B)
Removing: /Users/jrasmusson/Library/Logs/Homebrew/openexr... (64B)
Removing: /Users/jrasmusson/Library/Logs/Homebrew/fontconfig... (69.5KB)
Removing: /Users/jrasmusson/Library/Logs/Homebrew/tcl-tk... (64B)
Removing: /Users/jrasmusson/Library/Logs/Homebrew/jbig2dec... (64B)
Removing: /Users/jrasmusson/Library/Logs/Homebrew/heroku-node... (119B)
Removing: /Users/jrasmusson/Library/Logs/Homebrew/m4... (64B)
Removing: /Users/jrasmusson/Library/Logs/Homebrew/pcre... (64B)
Removing: /Users/jrasmusson/Library/Logs/Homebrew/libidn... (64B)
Removing: /Users/jrasmusson/Library/Logs/Homebrew/x265... (64B)
Removing: /Users/jrasmusson/Library/Logs/Homebrew/node... (64B)
Removing: /Users/jrasmusson/Library/Logs/Homebrew/libomp... (64B)
Removing: /Users/jrasmusson/Library/Logs/Homebrew/python@3.8... (3 files, 185.4KB)
Removing: /Users/jrasmusson/Library/Logs/Homebrew/imagemagick... (64B)
Removing: /Users/jrasmusson/Library/Logs/Homebrew/libev... (64B)
Removing: /Users/jrasmusson/Library/Logs/Homebrew/libffi... (64B)
Removing: /Users/jrasmusson/Library/Logs/Homebrew/python@3.9... (2 files, 4.4KB)
Pruned 0 symbolic links and 2 directories from /usr/local
+ Downloads
```

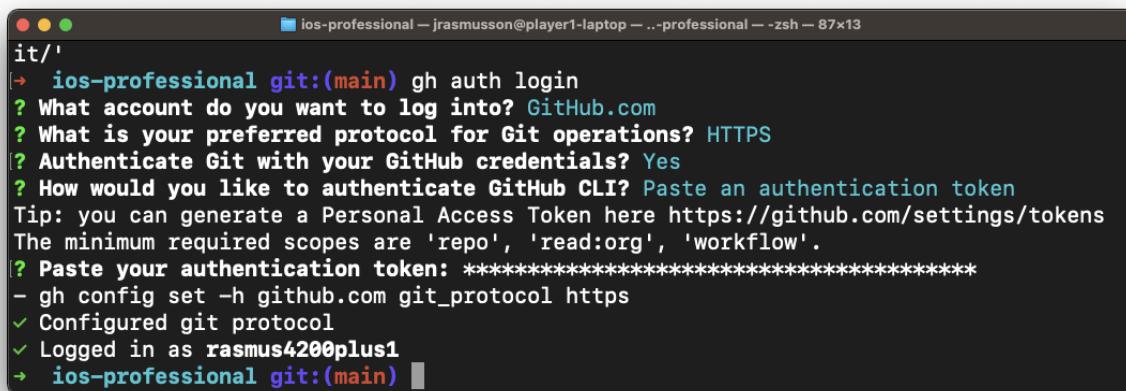
Enter GitHub Token

Then once GitHubCLI installation completes, enter your github credentials by typing:

```
> gh auth login
```

Follow the command prompts and use the following entries by pressing enter for each:

```
> GitHub.com (Enter)  
> HTTPS  
> Authenticate with credentials? Y  
> Paste an authentication token
```



The screenshot shows a terminal window titled "ios-professional" with the command "git:(main) gh auth login". The terminal displays a series of prompts from the GitHub CLI:

- ? What account do you want to log into? GitHub.com
- ? What is your preferred protocol for Git operations? HTTPS
- ? Authenticate Git with your GitHub credentials? Yes
- ? How would you like to authenticate GitHub CLI? Paste an authentication token

Below the prompts, there is a tip: "Tip: you can generate a Personal Access Token here <https://github.com/settings/tokens>". It also specifies that the minimum required scopes are 'repo', 'read:org', 'workflow'. The final step is to paste the authentication token, indicated by a series of asterisks: "? Paste your authentication token: *****". The terminal then shows the configuration command: "- gh config set -h github.com git_protocol https". Finally, it confirms the configuration: "Configured git protocol", "Logged in as rasmus4200plus1", and ends with "ios-professional git:(main)".

You should now be logged into github!

Now if we try making that push again.

```
> git push
```

```
ios-professional git:(main) vi README.md
ios-professional git:(main) x git add .
ios-professional git:(main) x git commit -m "My first commit"
[main 0c4f10a] My first commit
 1 file changed, 2 insertions(+)
ios-professional git:(main) git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 313 bytes | 313.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/rasmus4200plus1/ios-professional.git
 96ed74d..0c4f10a main -> main
ios-professional git:(main)
```

It should commit. And if you type:

```
> git log
```

```
commit 0c4f10a444f094748898719432ff8c6998ed31e9 (HEAD -> main, origin/main, origin/HEAD)
)
Author: Jonathan Rasmussen <rasmus4200@gmail.com>
Date: Thu Sep 16 06:27:20 2021 -0600

  My first commit

commit 96ed74d777da351da6758e3720f6deb595281c46
Author: rasmus4200plus1 <90771753+rasmus4200plus1@users.noreply.github.com>
Date: Thu Sep 16 05:38:57 2021 -0600

  Initial commit
(END)
```

You should see your first commit stored and pushed against your repository.

Congrats!

What should I do when my token expires?

When your token expires, GitHub will send you an email.

We noticed your personal access token "ios-professional-token" with admin:org, repo, workflow, and write:packages scopes will expire in 6 days.

If this token is still needed, visit <https://github.com/settings/tokens/698099153/regenerate> to generate an equivalent.

If you run into problems, please contact support by visiting <https://github.com/contact?tags=dotcom-accounts>

Thanks,
The GitHub Team

Simply click the link. Generate a new token. And then follow the instructions above in the previous section 'Enter GitHub Token'.