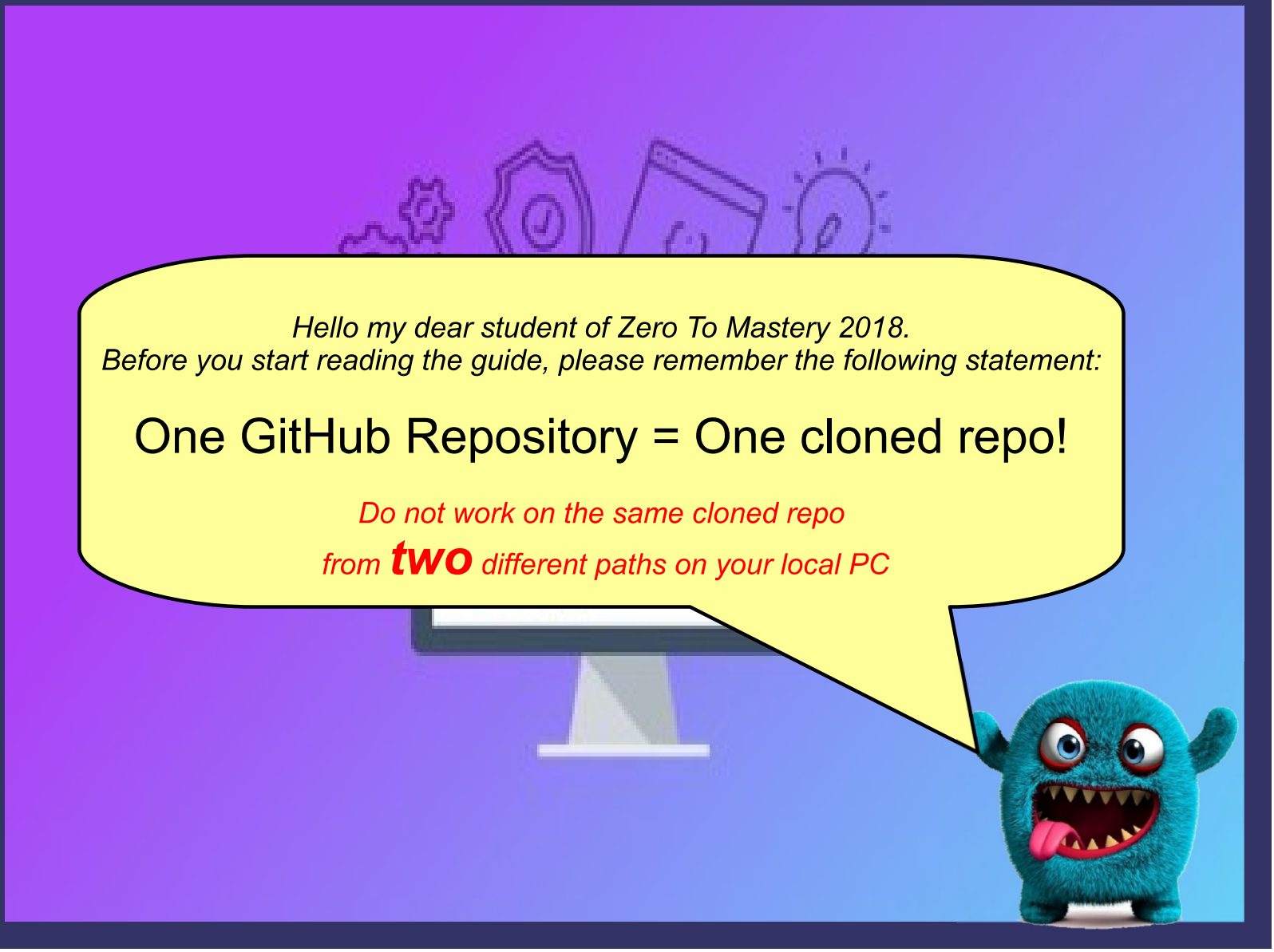
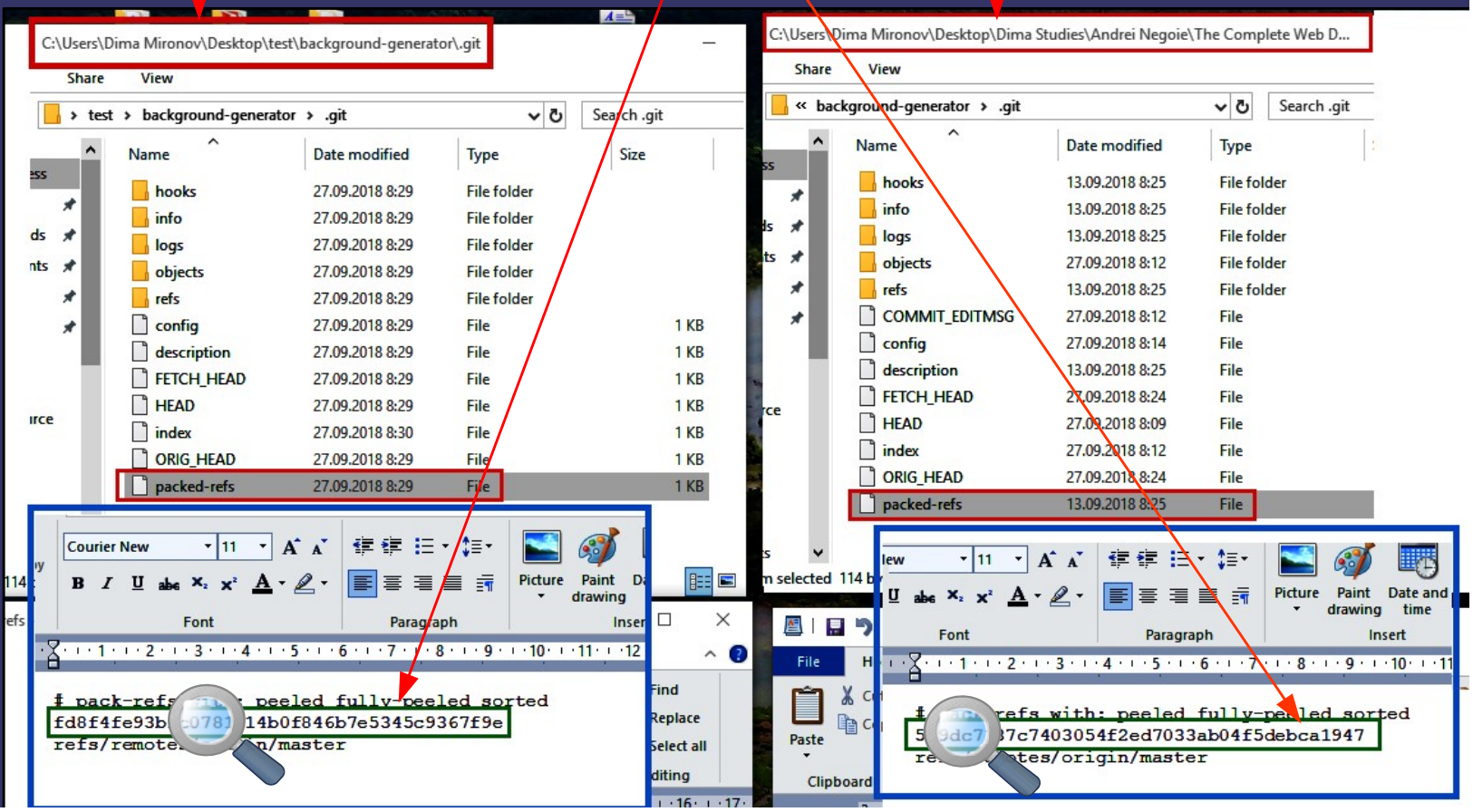

Hello my dear student of Zero To Mastery 2018.
Before you start reading the guide, please remember the following statement:
One GitHub Repository = One cloned repo!
Do not work on the same cloned repo
from two different paths on your local PC

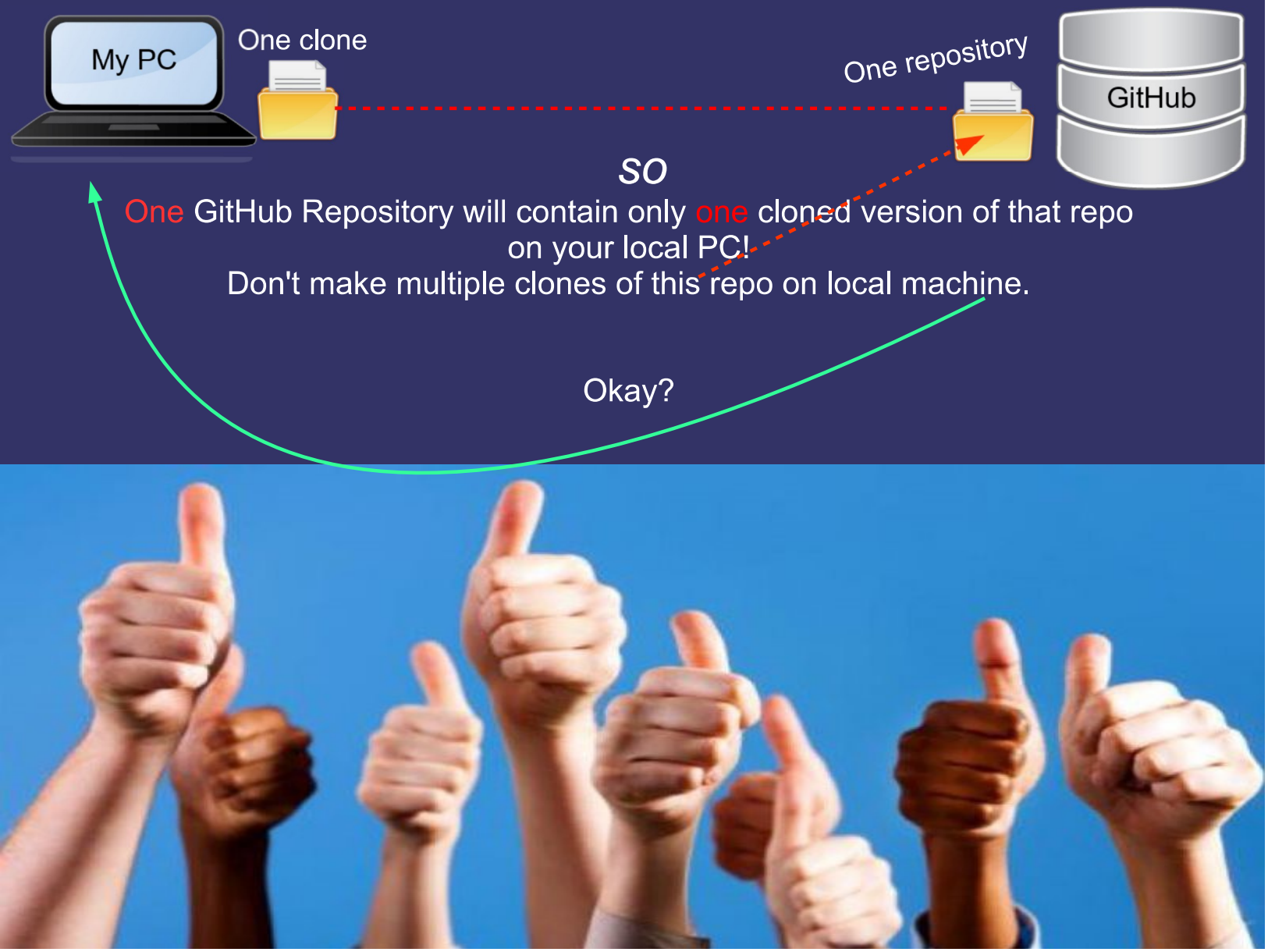
Remember this, or It might cause unexpected results...

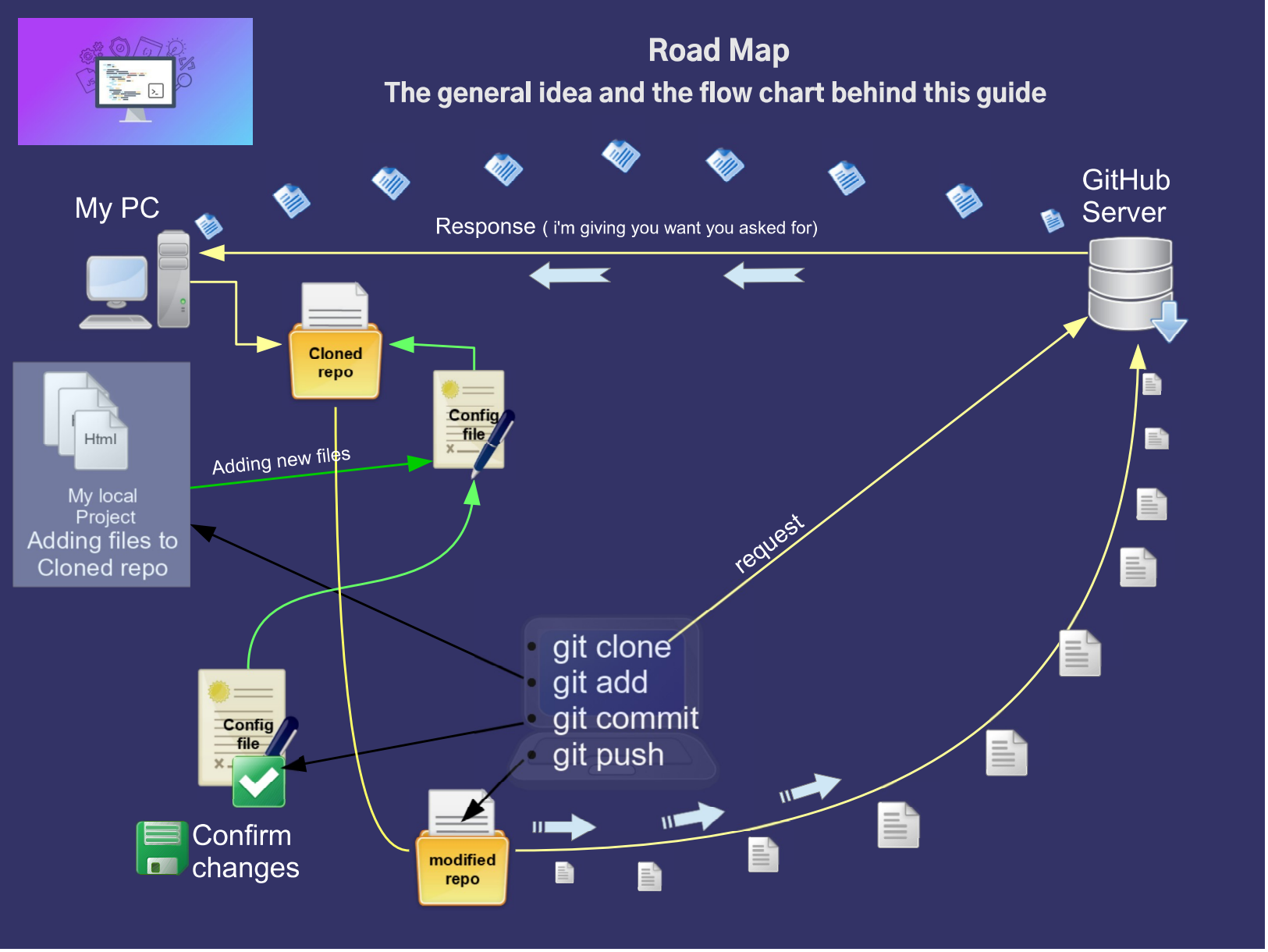
The very same cloned repo in different places on your PC will have different package references id's after you **manipulate the original repo!**

Second path =Desktop cloned repo

First path = Original cloned repo




My PC
One clone
One repository
GitHub
SO
One GitHub Repository will contain only one cloned version of that repo on your local PC!
Don't make multiple clones of this repo on local machine.
Okay?


Road Map
The general idea and the flow chart behind this guide
My PC
GitHub Server
Response (i'm giving you want you asked for)
Cloned repo
Config file
Html
My local Project
Adding files to Cloned repo
Adding new files
request
git clone
git add
git commit
git push
Config file
Confirm changes
modified repo

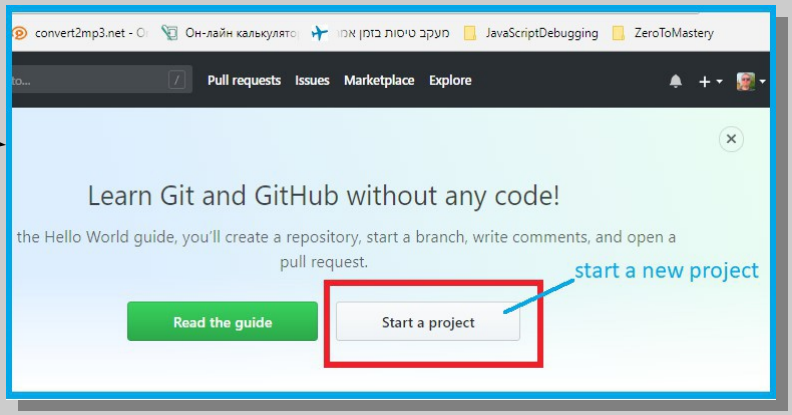
Git and GitHub Source Control

Part1 (Working on a new repository)

The Complete Web Developer in 2018: Zero to Mastery

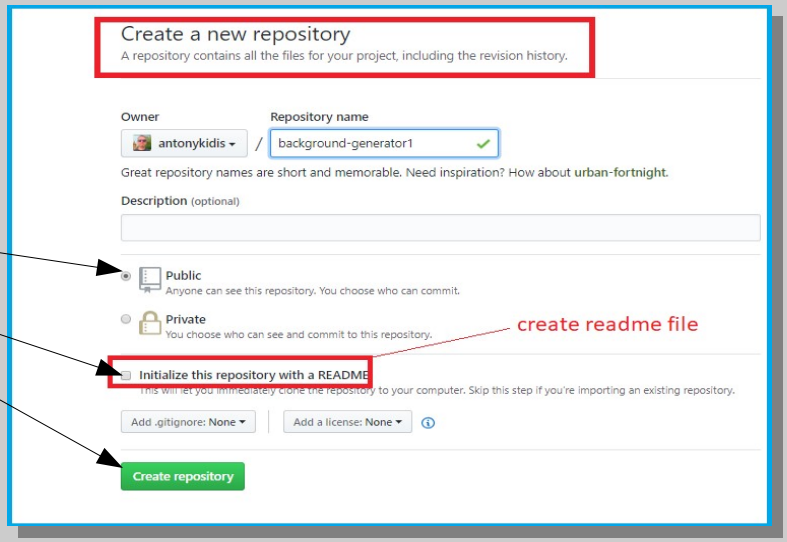
Step 1

- Navigate to https://github.com
- Start a new project



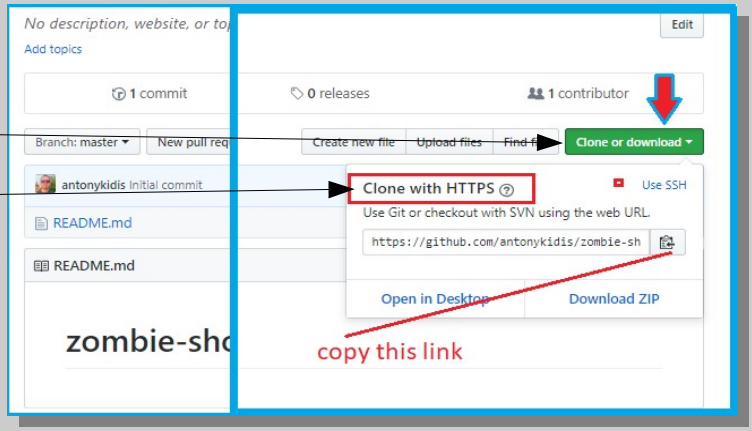
Step 2

- Name your project background-generator
- Select Public
- Check to create a README file
- Finally click create repository button



Step 3

- Click Clone or download button
- Select use HTTPS
- Copy the URL to a clipboard by clicking the Little icon



Step 4

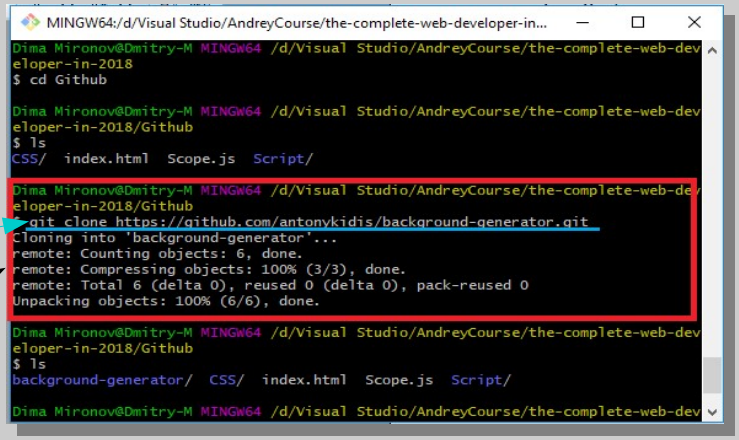
- Open terminal(bash,or powershell) or other prefered terminal.
- Navigate to a project you currently work on.
- Enter the following commad:

  git clone https://github.com/antonykidis/background-generator.git

This will clone GitHub's repository to your computer.

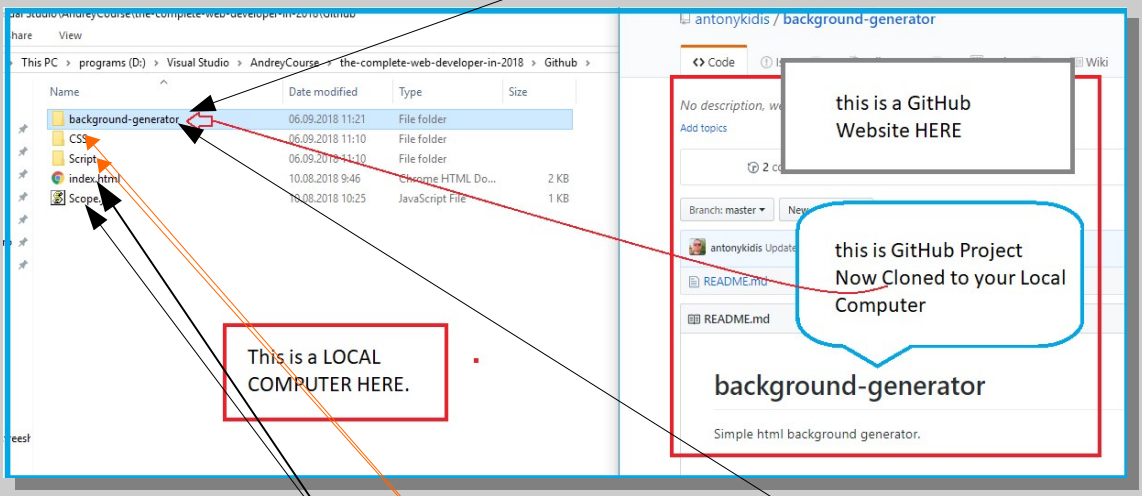
- Here is the following output

  Unpacking objects 100% (6 / 6) done.
  The background-generator cloned now to your computer



We've just copied our repo to a local computer.

You can see the background-generator appeared in the left window



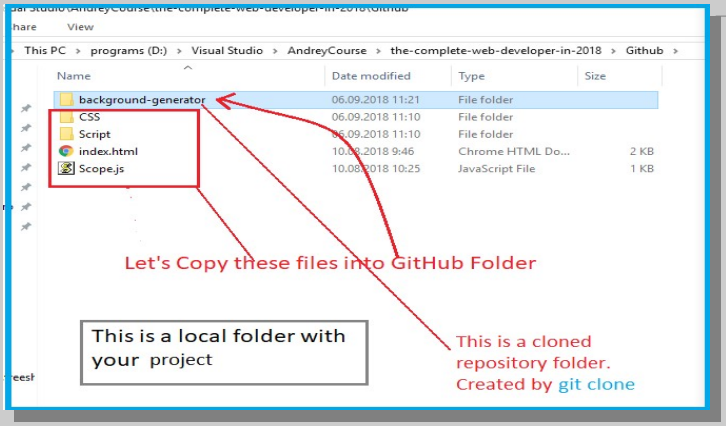
Step 5

Next Step is to take all of our local project's files and folders, and copy them into the background-generator repository folder.

You can copy these files using the IDE, or a command line (terminal)

To copy files and folders via command line please refer to the terminal section of the course

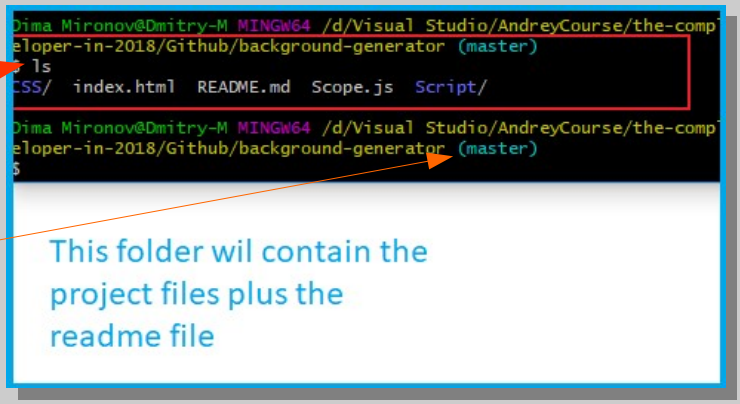
Illustration



Step 6

1. Back to a terminal window and cd (navigate) to a newly created cloned direcory named background-generator.
   Example:
   cd background-generator
2. Then type ls (dir in windows) to list the contents of a background-generator folder.
3. You should see the similar output with all the copied files from the previous step.

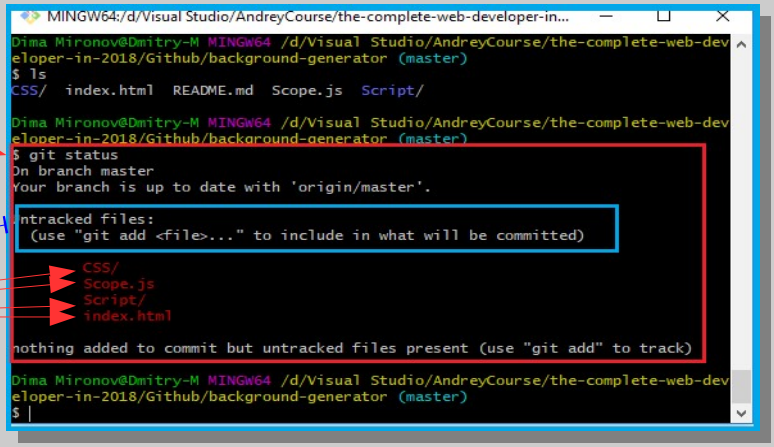
(master) means that we now working in the repository folder



Step 7

- We now can check the status of our files since we've copied our files into the repository folder(comunication folder)
- While in terminal type git status
- This will invoke the following output

You can see that we have untracked files and folders
We can now add these files to a GitHub **Configuration file** before upload it back to GitHub



Configuration file. We will explore it in the following sections.

Adding files and folders to configuration file

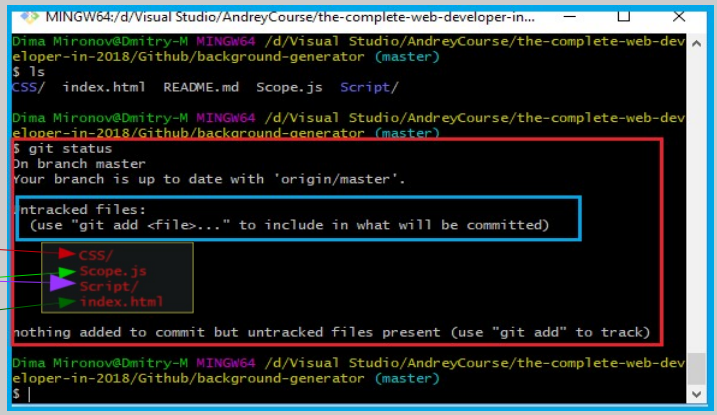
Step 8

- Let's Add utracked files to a special Configuration file

  Type the following commands:

- git add CSS (folder)
- git add Script(folder)
- git add Scope.js(file)
- git add index.html (file)

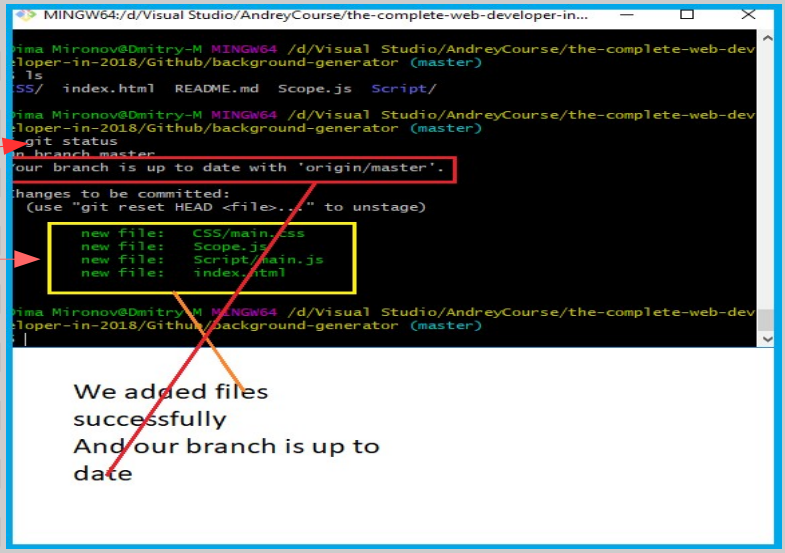
  To add all files and folders at once type **git add .**



Step 9

- After adding each file, and folder

1. Type git status
2. You should see the following output
3. Okay we've just set the configuration file. Great!
4. Next step is to commit these files into the configuration file.

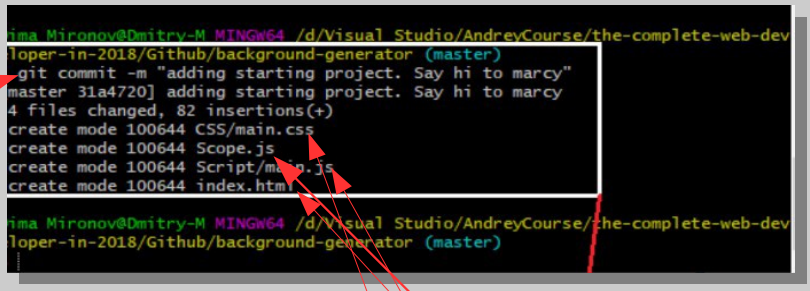


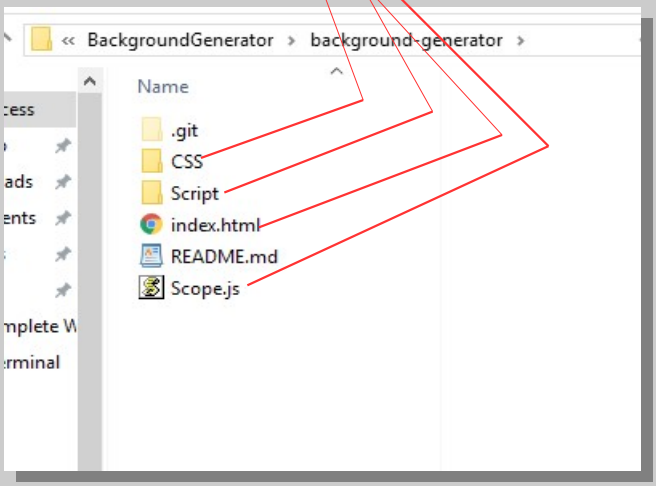
Commit the changes we've made in previous steps

Step 10

- Let's commit previously added files and folders.
- Enter the following command
- git commit -m "adding starting project"

Commit means **Save all** the changes to config.file before pushing it back to the GitHub server

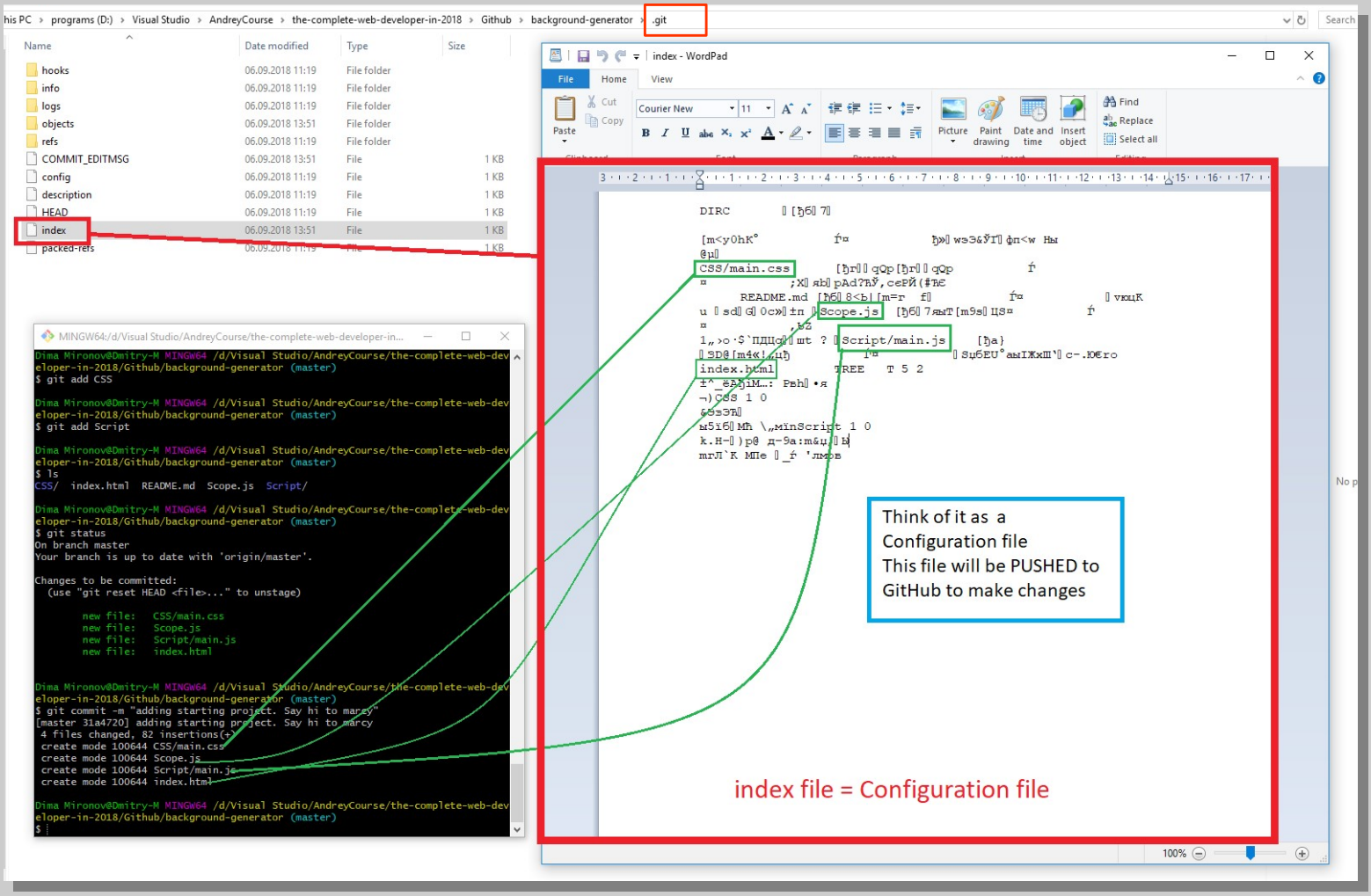




We've just set up the configuration file

- The last step is to push these changes back to the GitHubServer. But let's explore the configuration file first.

Explore the configuration file



Push changes to GitHub

Step 11

- Finally Push the changes to the server
- Use the following command
- git push
- This will get you to the following output
- We've succesfully wrote objects to http://github.com...

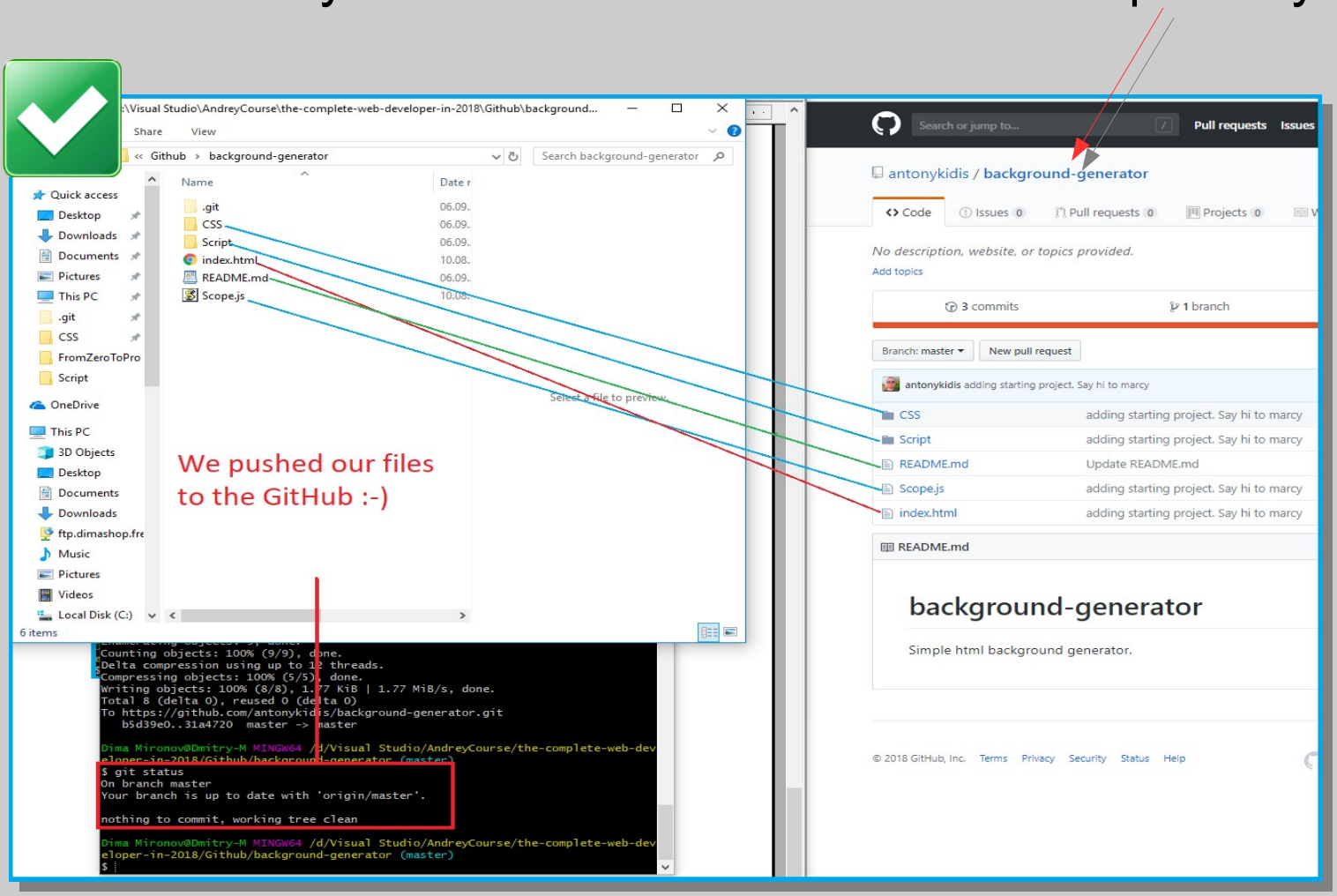
```
Dima Mironov@Dmitry-M MINGW64 /d/Visual Studio/AndreyCourse/the-complete-web-dev
eloper-in-2018/Github/background-generator (master)
$ git push
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 12 threads.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (8/8), 1.77 KiB | 1.77 MiB/s, done.
Total 8 (delta 0), reused 0 (delta 0)
To https://github.com/antonykidis/background-generator.git
   b5d39e0..31a4720  master -> master
```

Great!
Now a background-generator is online

Step 12

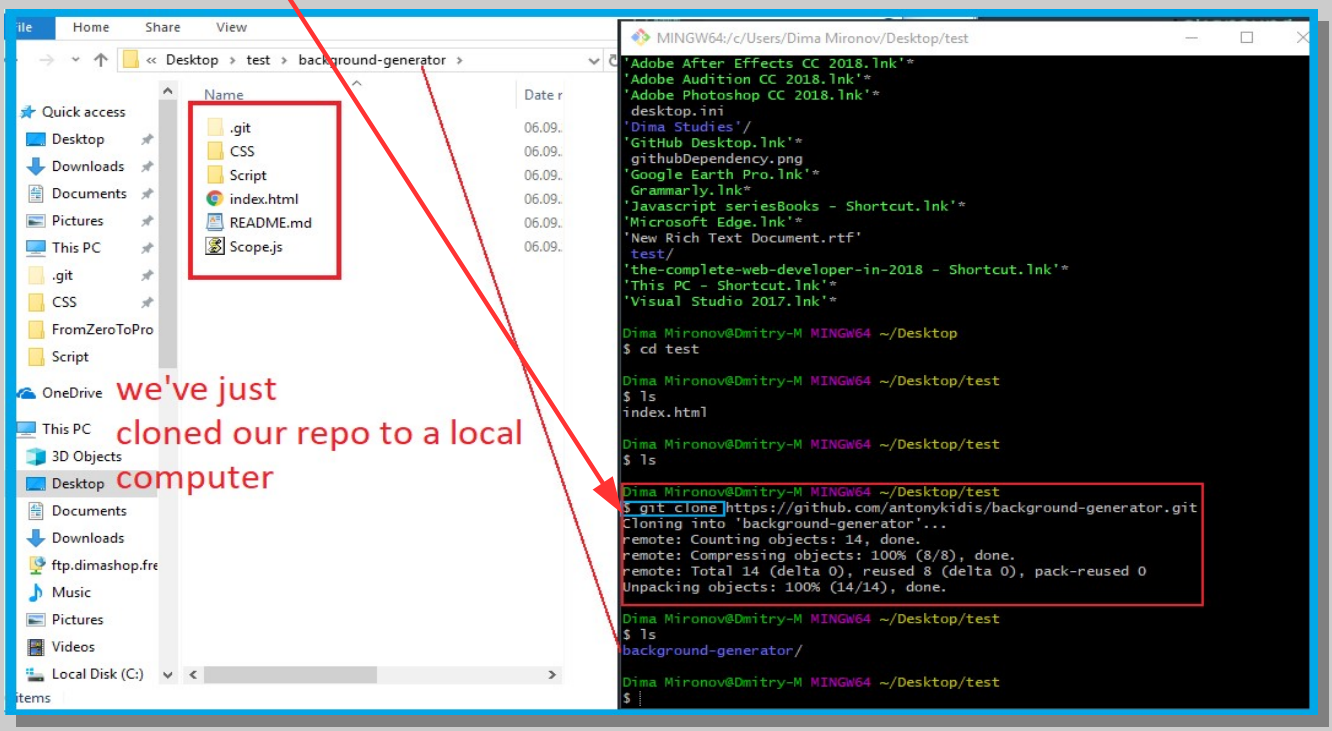
Go back to https://github.com and check
What you've done so far
(Test pushed files on GitHub)

What we've done so far
We succesfully added a new files into GitHub repository



Next Step - Work on Existing Repository

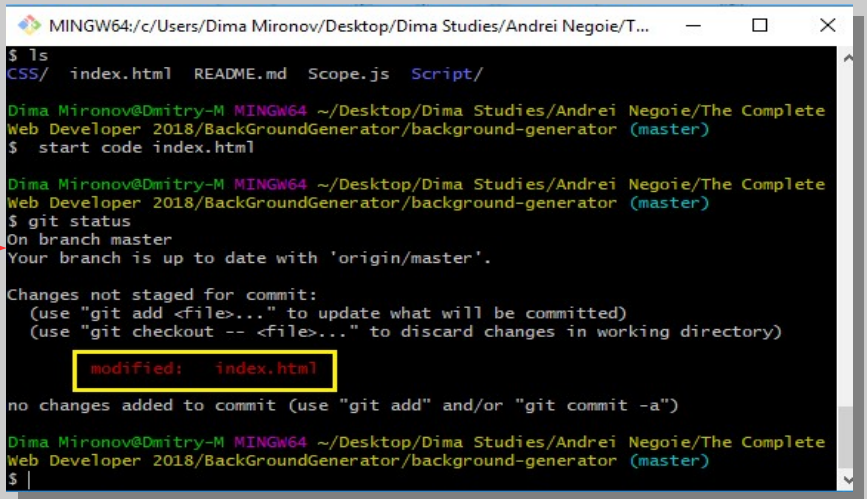
- Open Terminal(or cmd/bash/powrshell in windows)
- Navigate to the folder where you want to download a GitHub existing repository
- Copy repository's url from GitHub
- Type the following command (Example)
  git clone https://github.com/antonykidis/background-generator.git
- Finally check if a cloned repository appears in the desired folder



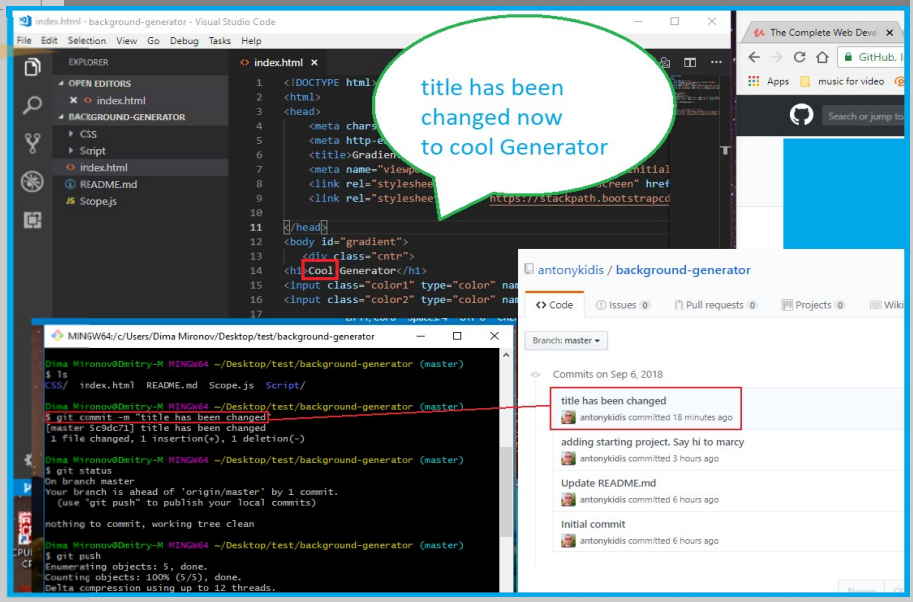
Next Step - Pushing a modified files back to a GitHub

1. For example:
   open index.html and change the title<h1>here</h1>
   to a cool generator
2. Save it
3. Comeback to a terminal window, and type
   git status

   You should see a similar output
4. Type git add (adding the index.html)
5. Type git commit -m "title has been changed"
   (save changes)
6. Type git push (pushing the changes back to
   server)

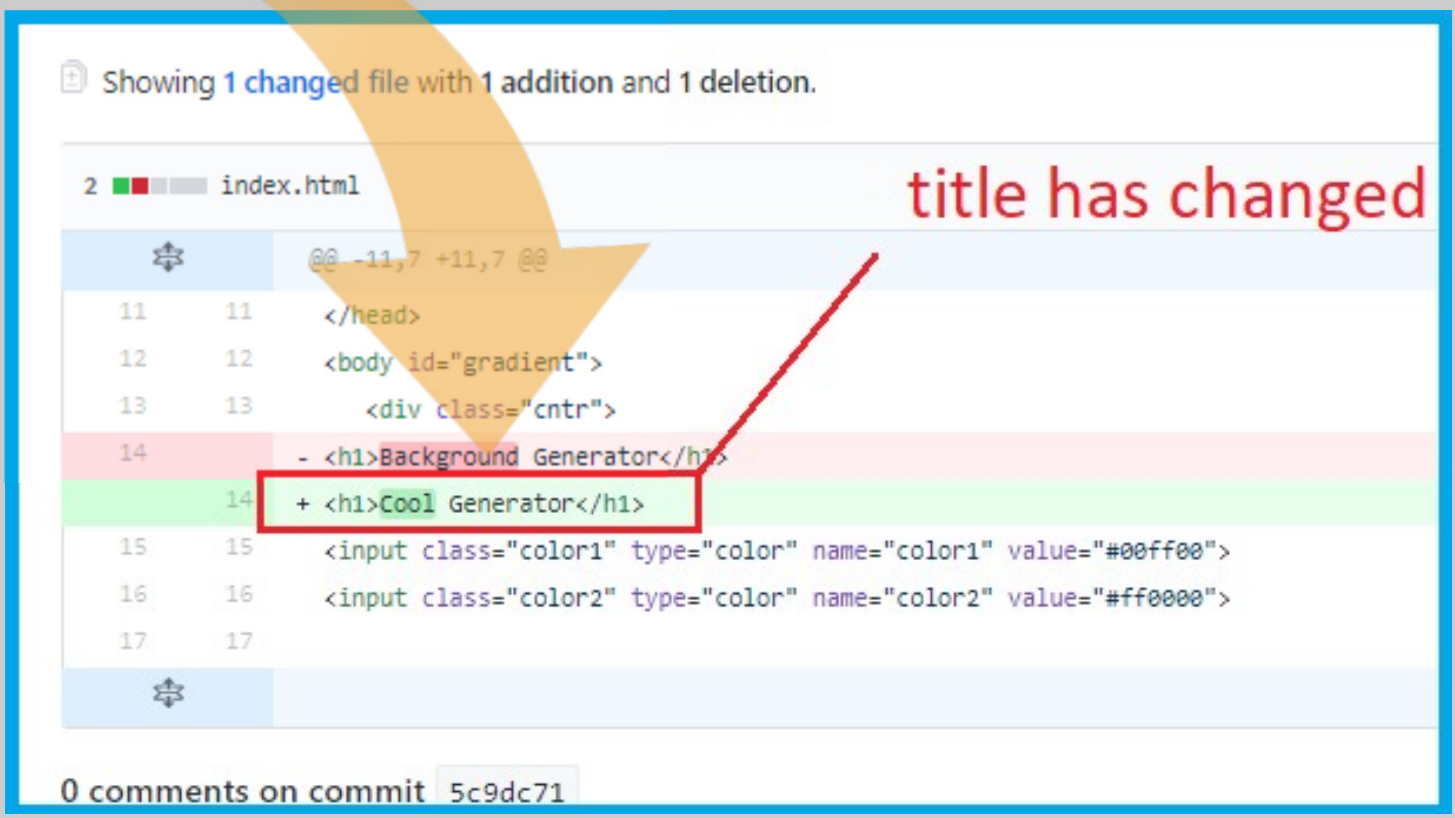


7.Go to GitHub



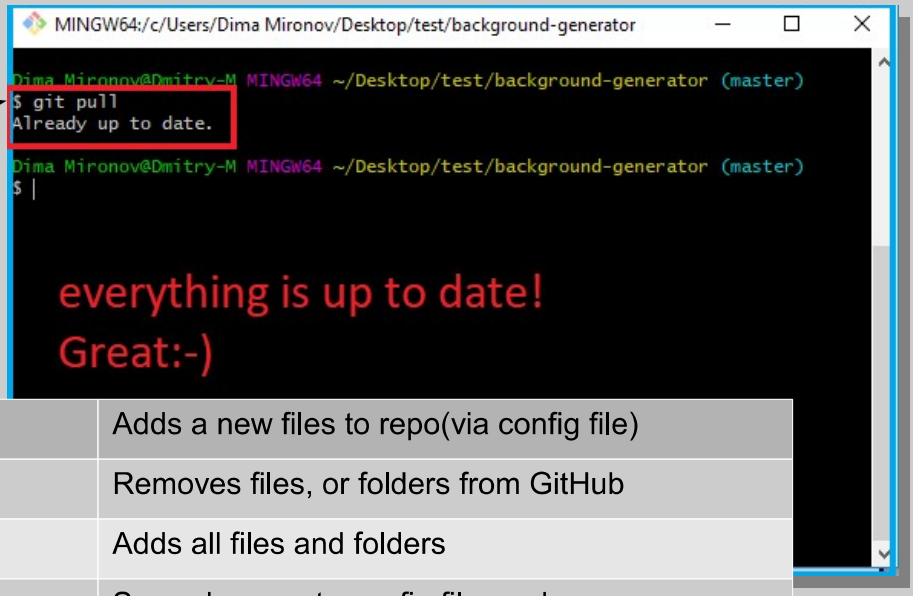
Navigate to commits link

And see the actual changes we've just made
As you can see the title has been changed to a Cool Generator



We can pull the latest version from GitHub

- Type git pull



Useful commands

git add <files, or folders>	Adds a new files to repo(via config file)
git rm <files or folders>	Removes files, or folders from GitHub
git add .	Adds all files and folders
git commit	Save changes to config file, and prepare pushing it to the server
git push	Pushes the new changes back to server, and updates the remote repository
git pull	Pulls the latest version of the repository from a GitHub
git status	Checks the overall status of repository
git branch -d branch_name or git branch -D branch_name	Removes the named branch
git revert dd61ab32	Removes the last commit mistakenly pushed to the GitHub By commit ID. So the commit id in this case is dd61ab32

git push origin --delete <branch_name>	Delete a remote GIT branch
git push <remote_name> :<branch_name> Remote name means this: git push origin :<branch_name>	Delete a remote GIT branch

The end of part 1
Please read the GitHub guide part 2

The complete Web Developer in 2018

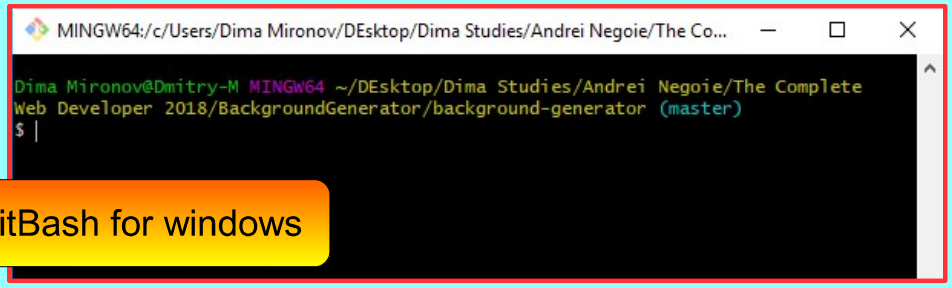
Git and GitHub Guide

Part-2

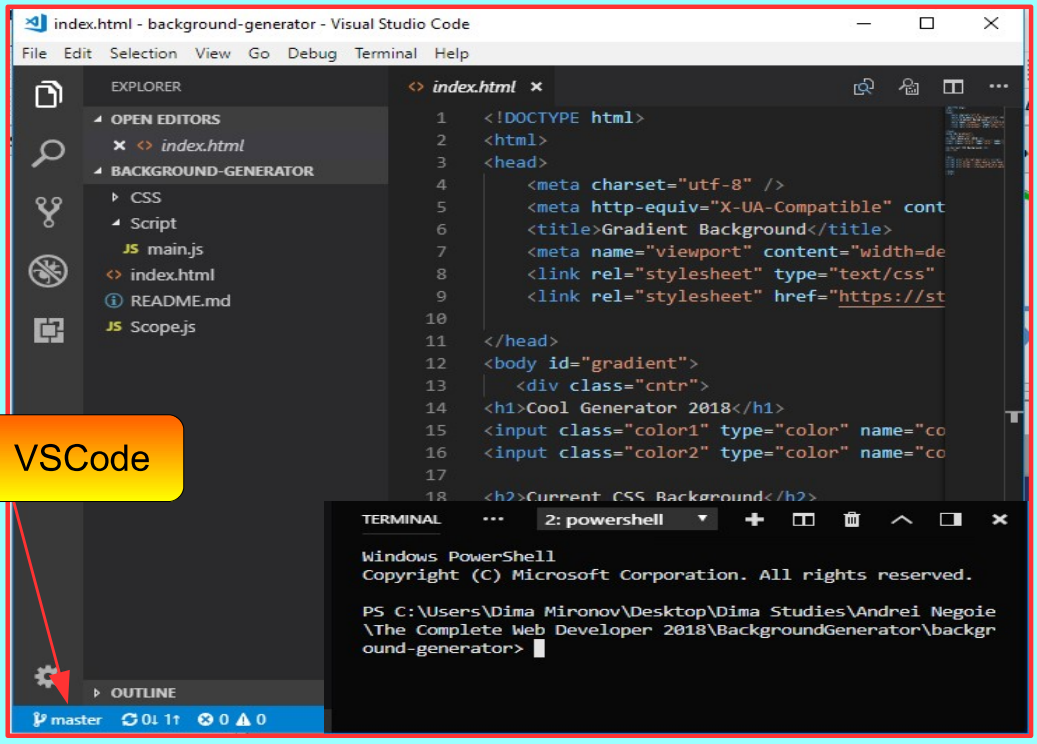
Branching

Info Page

For all the windows users.
*You can u*se Gitbash terminal, or a Visual Studio Code terminal.
Download it here https://git-scm.com/downloads



GitBash for windows



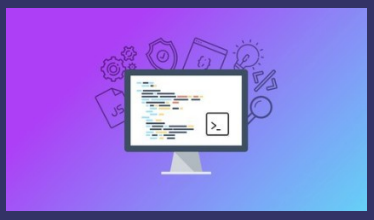
VSCode

VS Code displays a branch name. However, It doesn't display a branch name In the PowerShell terminal.

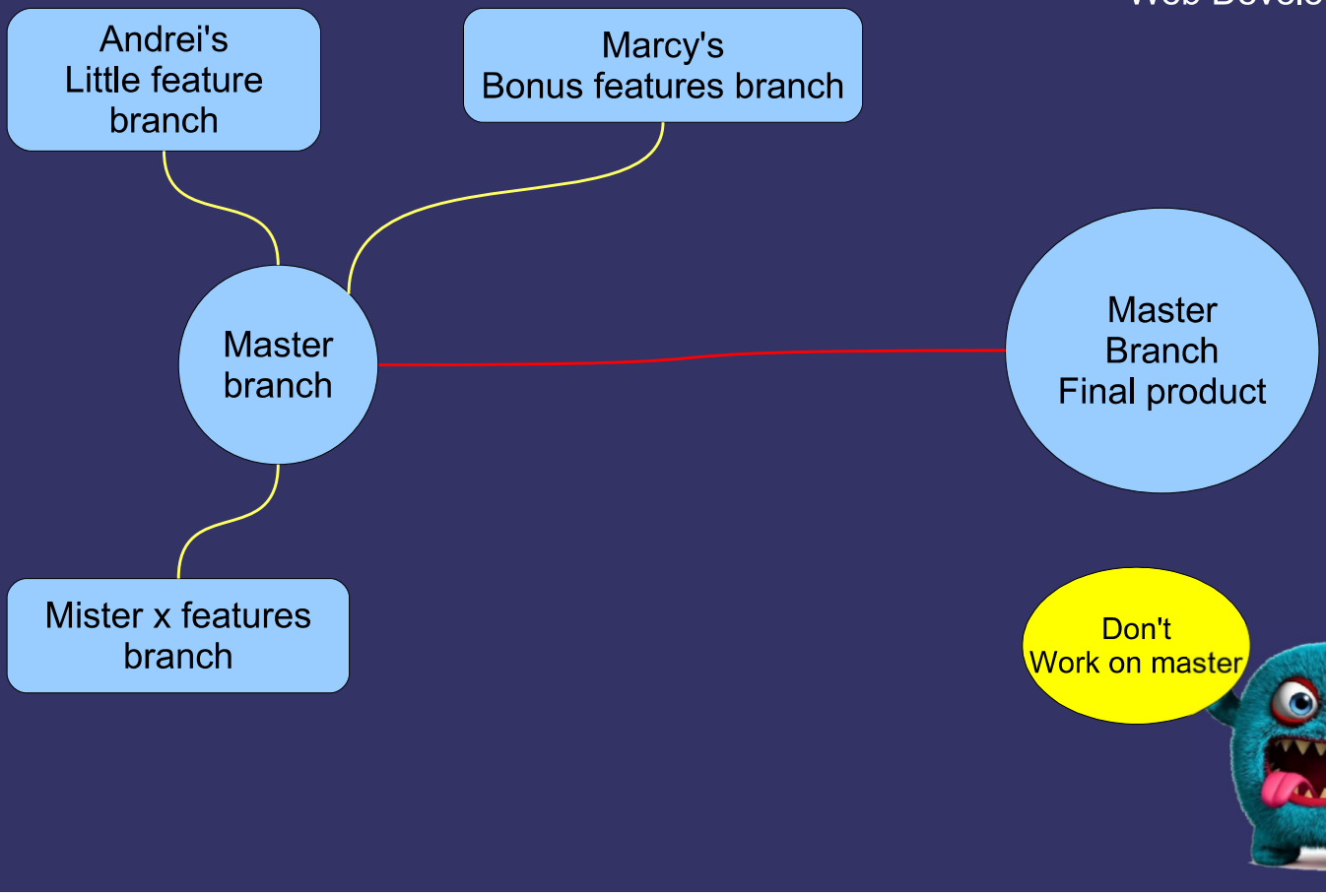
My suggestion is to work with the Git Bash terminal.

Refers to Section 15, video #145

Git and GitHub Guide Part-2 Branching



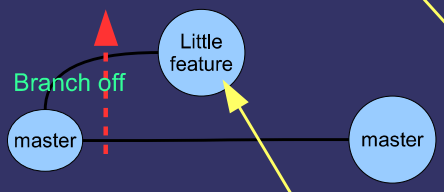
The complete
Web Developer in 2018



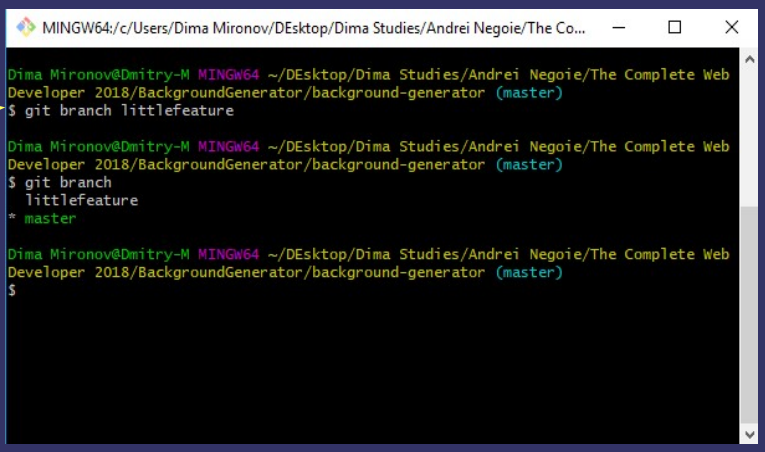
Create a new branch

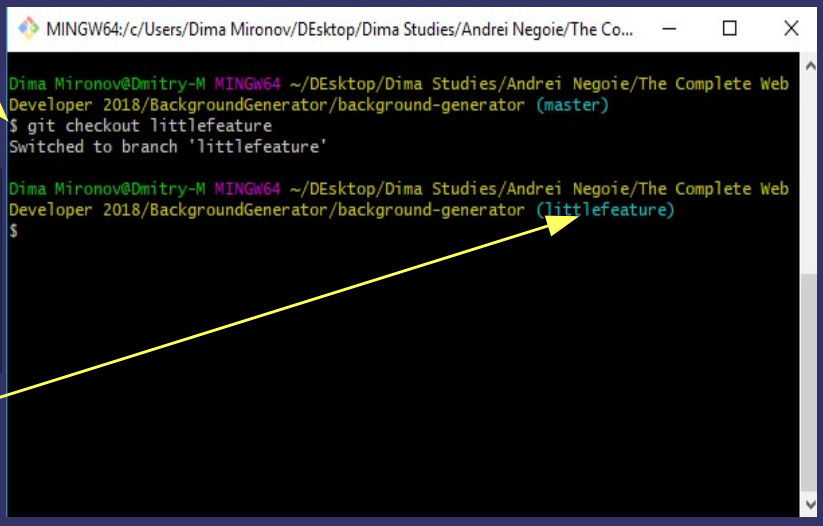
Let's create a new branch (terminal)

1. Type git branch littlefeature (this will create a new branch)
2. Type git branch (will display all branches available)
3. Type git checkout littlefeature (Switch to a new branch)



* master means we currently on master branch.





Note:
If something went wrong and you need to delete the branch, type git branch -d littlefeature
The above command will delete the mistaken branch

4. We successfully switched to a new branch

Let's modify index.html now

1. Open index html (use IDE or terminal) Type: start code index.html

2. Locate the <h2> tag and change the title to
<h2>This is thebackground</h2>

3. Save all, then do the following procedure...

4. git add index.html
5. git commit -m“changing title“
6. git push

If it doesn't work and you get an error

fatal: The current branch littlefeature has no upstream branch.
To push the current branch and set the remote as upstream,
use git push --set-upstream origin littlefeature

Fix.
Type git push –set-upstream origin littlefeature

After using the above command once,
you can then use a regular
git push command (without notifications)

Finally, you can cheat and type this
git push origin littlefeature

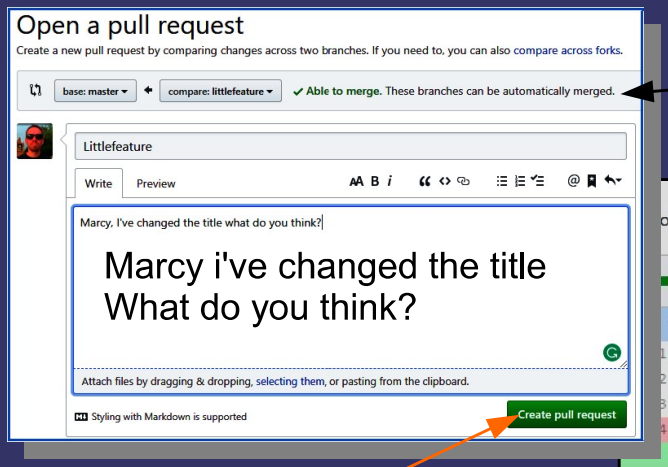
```
12  <body id="gradient">
13      <div class="cntr">
14  <h1>Cool Generator 2018</h1>
15  <input class="color1" type="color" name="color1" value="#00f
16  <input class="color2" type="color" name="color2" value="#ff0
17
18  <h2>This is the background</h2>
19  <h3></h3>
20
21
```

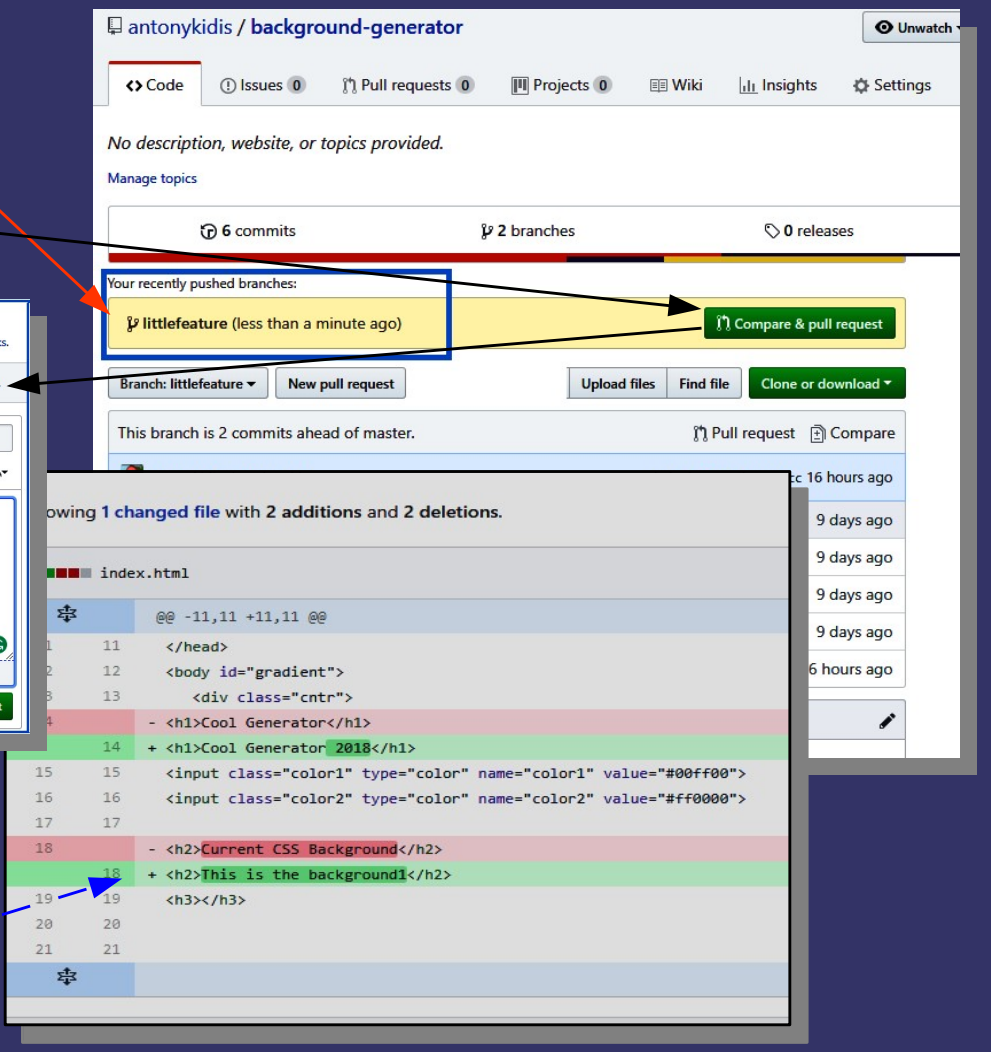
```
Dima Mironov@Dmitry-M MINGW64 ~/Desktop/Dima Studies/Andrei Negoie/The Complete
Web Developer 2018/BackgroundGenerator/background-generator (littlefeature)
$ git push
fatal: The current branch littlefeature has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin littlefeature


Dima Mironov@Dmitry-M MINGW64 ~/Desktop/Dima Studies/Andrei Negoie/The Complete
Web Developer 2018/BackgroundGenerator/background-generator (littlefeature)
$ git push origin littlefeature
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 302 bytes | 302.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0)
```

Go Back to https://github.com

1. We now see a yellow notification
   You recently pushed branches

2. If we click the green button
   Compare & pull request
   It will open the a pull request window





3. Eneter a message, check the changes
   And finally click the create pull request button

Note:
We've made a change to index.html
Marcy will decide aprove it or not.

Okay now STOP!
Do you really understand what is going on here!?
No...
Yes or Yes!?

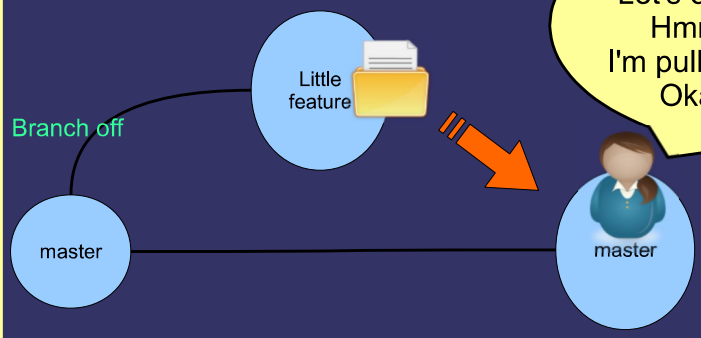
Hey marcy I made a pull request
Do something about it

Compare & pull request

Hey Marcy I made a pull request. Please look at my html file,
compare new changes to the previous one.
And if they good enough please pull them, and merge them into the master branch.

Marcy is an administrator and she will open your pushed files on a littlefeature branch.
She will compare them to the previous one, and decide whether to prove these changes or not

Hi Andrei!
Let's compare your changes
Hmm..okay looks good!
I'm pulling this into the master
Okay it's merged now!



Important
At this point you as a developer
Just need to wait for Marcy's approval.
That's it.
Don't be confused by the rest of the diagrams.

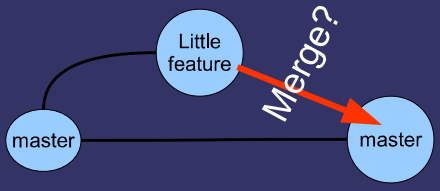
Let's see How Marcy handle my request on her side as Administrator.

With this information in mind let's Proceed to the next step.

We successfully created a pull request (PR)

As you can see antonykidis(Andrei) wants to merge 3 commits into **master** from **littlefeature**

Hey Marcy I did a little Change on little feature, is it okay for me to Merge it to master?



Let's see what Marcy will probably do..

Littlefeature #1

Open **antonykidis** wants to merge 3 commits into `master` from `littlefeature`

Conversation 0 · Commits 3 · Checks 0 · Files changed 1

antonykidis commented 4 minutes ago — Owner

Marcy, I've changed the title what do you think?

antonykidis added some commits 19 hours ago

`update the title` 122ade7
`changing title` ef61fcc
`changing title to background1` 52de4ef

1 commit
2 commit
3 commit

Add more commits by pushing to the **`littlefeature`** branch on **antonykidis/background-generator**.

Continuous integration has not been set up
Several apps are available to automatically catch bugs and enforce style.

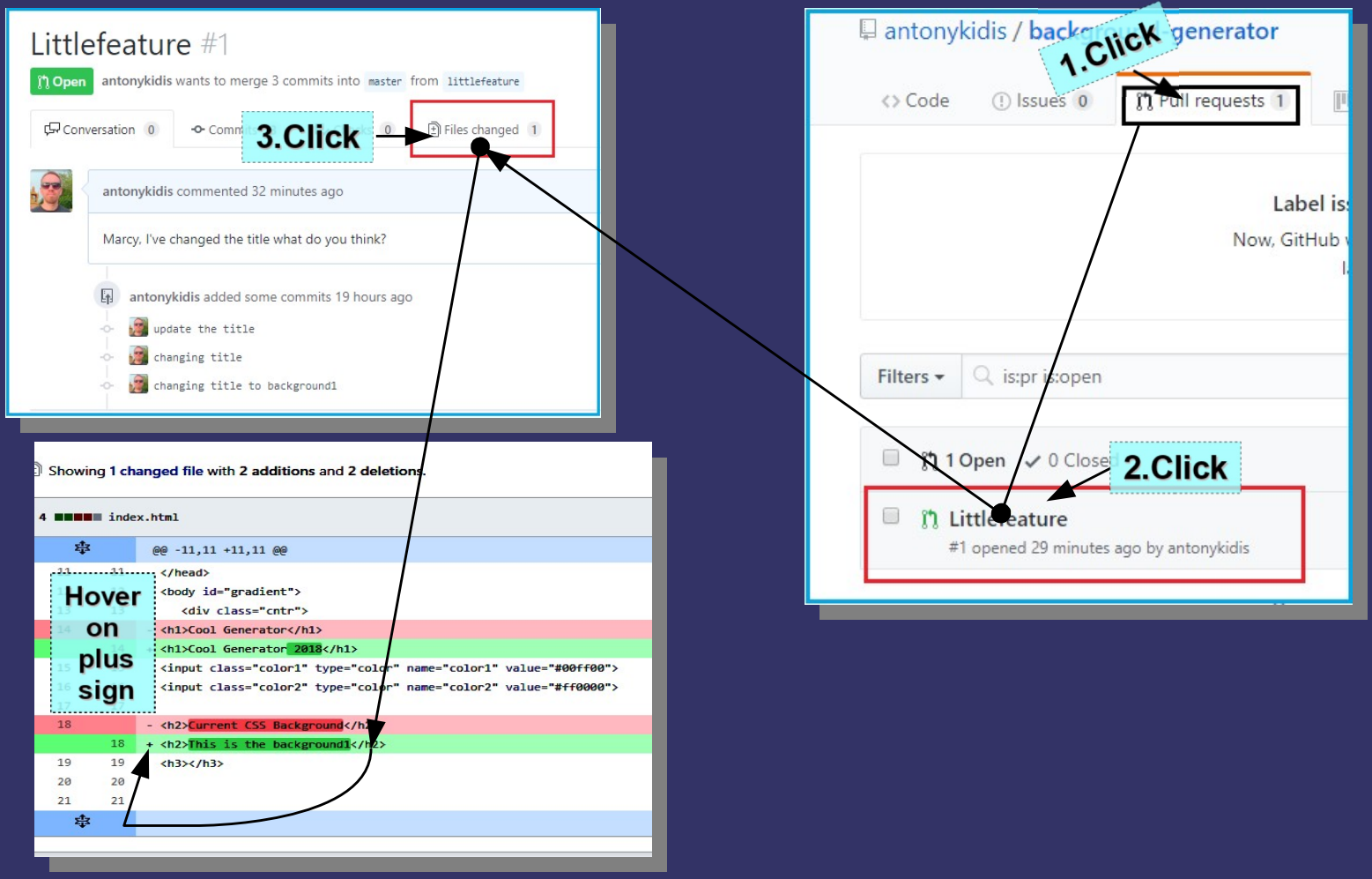
This branch has no conflicts with the base branch
Merging can be performed automatically.

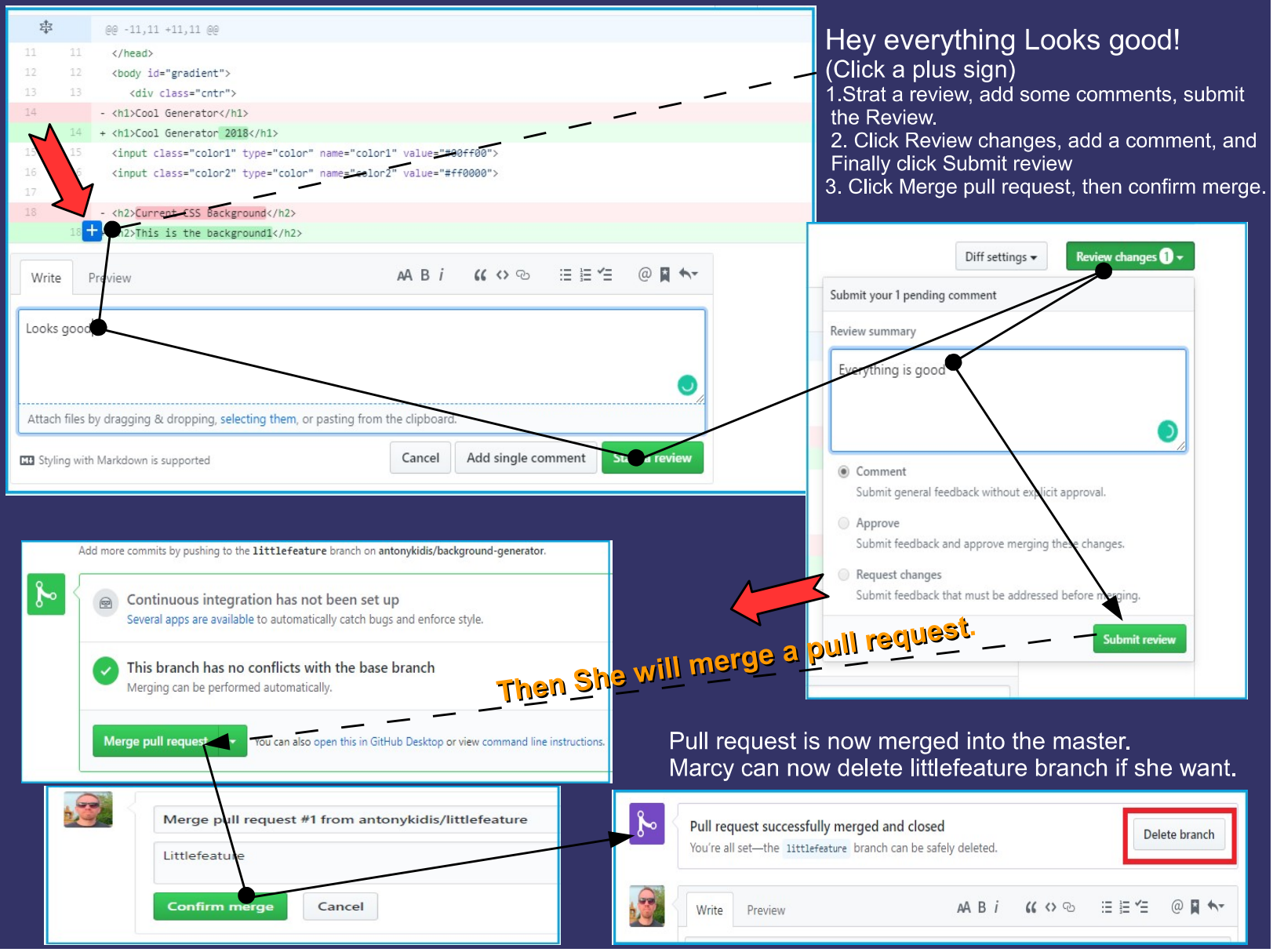
Merge pull request — You can also open this in GitHub Desktop or view command line instructions.

Write Preview

Leave a comment

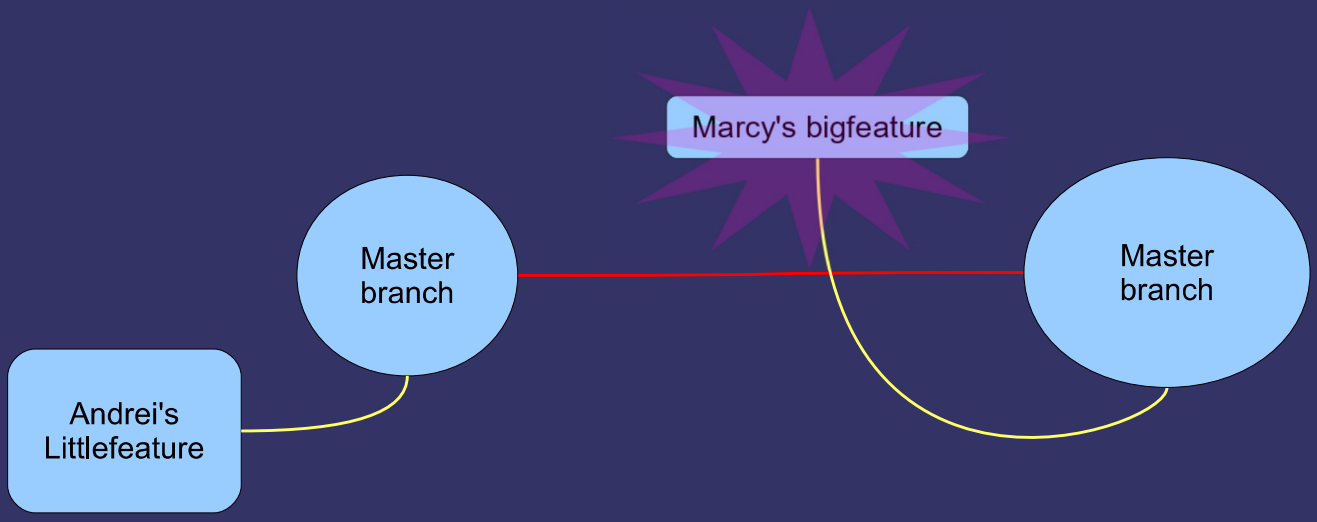
Marcy will Open-up a pull request link

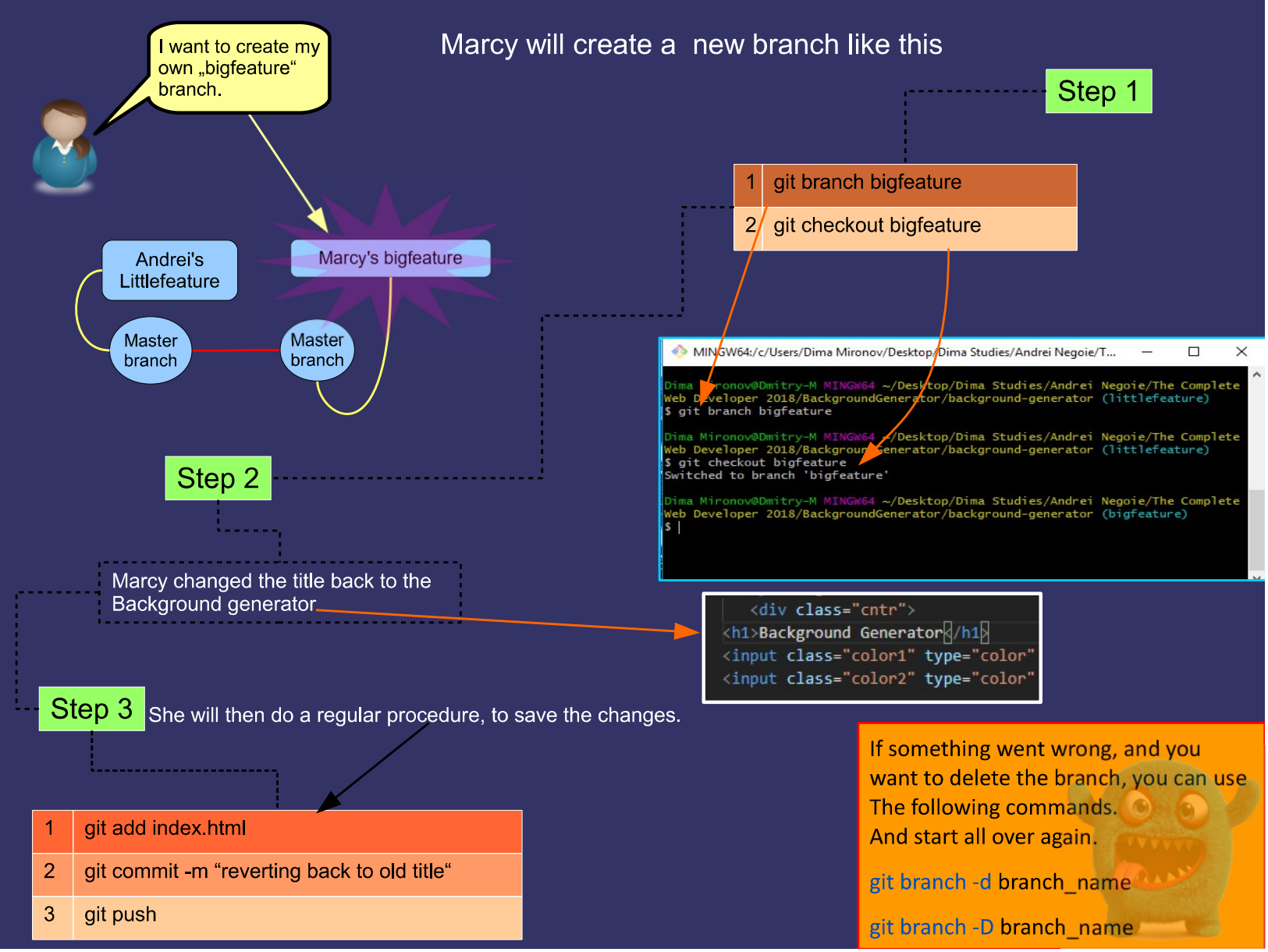



Hey everything Looks good!
(Click a plus sign)
1.Strat a review, add some comments, submit the Review.
2. Click Review changes, add a comment, and Finally click Submit review
3. Click Merge pull request, then confirm merge.
Write
Preview
Looks good
Attach files by dragging & dropping, selecting them, or pasting from the clipboard.
Styling with Markdown is supported
Cancel
Add single comment
Diff settings
Review changes
Submit your 1 pending comment
Review summary
Everything is good
Comment
Submit general feedback without explicit approval.
Approve
Submit feedback and approve merging these changes.
Request changes
Submit feedback that must be addressed before merging.
Submit review
Then She will merge a pull request.
Add more commits by pushing to the littlefeature branch on antonykidis/background-generator.
Continuous integration has not been set up
Several apps are available to automatically catch bugs and enforce style.
This branch has no conflicts with the base branch
Merging can be performed automatically.
Merge pull request
You can also open this in GitHub Desktop or view command line instructions.
Merge pull request #1 from antonykidis/littlefeature
Littlefeature
Confirm merge
Cancel
Pull request is now merged into the master.
Marcy can now delete littlefeature branch if she want.
Pull request successfully merged and closed
You're all set—the littlefeature branch can be safely deleted.
Delete branch
Write
Preview

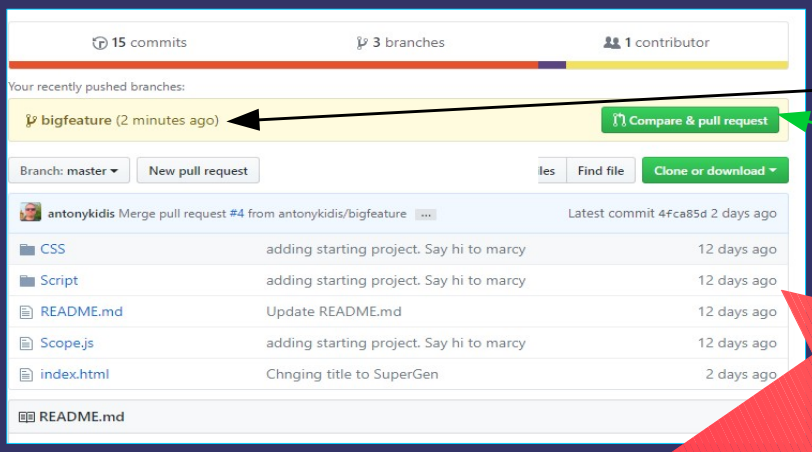
Merge Conflicts 07:45 (time)

What if Marcy wants to create her own feature!?



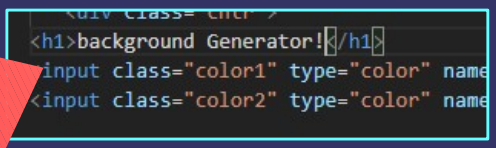

I want to create my own „bigfeature“ branch.
Marcy will create a new branch like this
Step 1
1 git branch bigfeature
2 git checkout bigfeature
Andrei's Littlefeature
Marcy's bigfeature
Master branch
Master branch
MINGW64:/c/Users/Dima Mironov/Desktop/Dima Studies/Andrei Negoie/T...
Dima Mironov@Dmitry-M MINGW64 ~/Desktop/Dima Studies/Andrei Negoie/The Complete Web Developer 2018/BackgroundGenerator/background-generator (littlefeature)
$ git branch bigfeature
Dima Mironov@Dmitry-M MINGW64 ~/Desktop/Dima Studies/Andrei Negoie/The Complete Web Developer 2018/BackgroundGenerator/background-generator (littlefeature)
$ git checkout bigfeature
Switched to branch 'bigfeature'
Dima Mironov@Dmitry-M MINGW64 ~/Desktop/Dima Studies/Andrei Negoie/The Complete Web Developer 2018/BackgroundGenerator/background-generator (bigfeature)
$
Step 2
Marcy changed the title back to the Background generator

<h1>Background Generator</h1>
<input class="color1" type="color"
<input class="color2" type="color"
Step 3
She will then do a regular procedure, to save the changes.
1 git add index.html
2 git commit -m “reverting back to old title“
3 git push
If something went wrong, and you want to delete the branch, you can use The following commands. And start all over again.
git branch -d branch_name
git branch -D branch_name



Now go back to a GitHub. We again see The pull request. **Don't compare and pull request This time!**

Before comparing and pulling a request Marcy decided to add an exclamation mark to the title



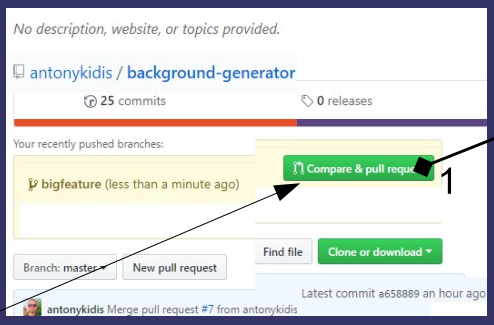
Marcy adds exclamation mark

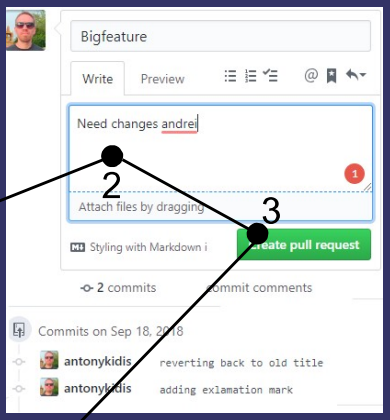
She then Save all...and follow the below steps

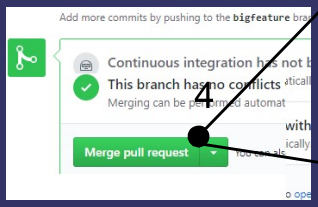
1	git add index.html
2	git commit -m “adding exclamation mark“
3	git push

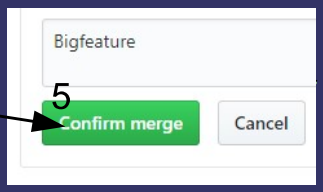
Let's go back to GitHub.
So now we have 2 commits
1. is the background generator
2. and the background generator! with the exclamation mark

1. Click Compare and pull request button,
2. Add a comment.
3. Click Create Pull Request button.
4. Merge pull request button.
5. Finally click Confirm merge button.

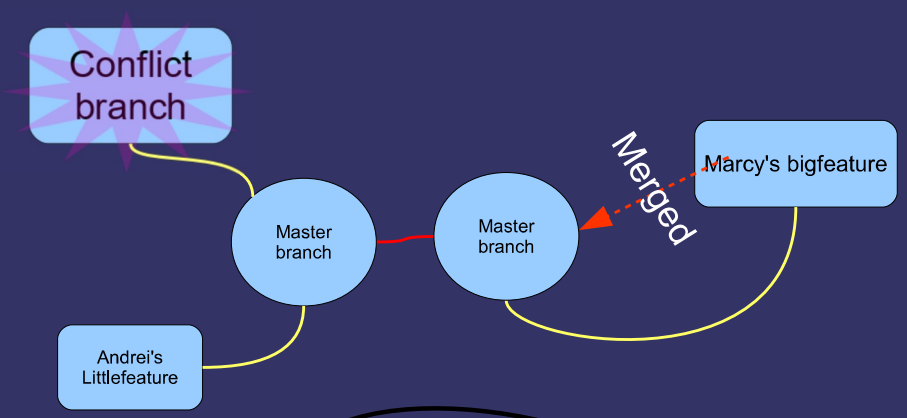








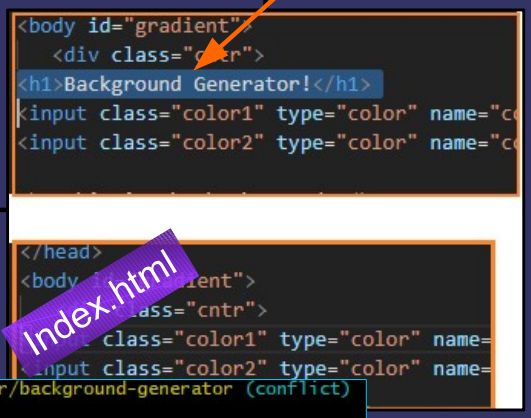
Marcy just merged a pull request from a bigfeature branch.
Meanwhile, Andrei decided to create a new branch too



In the terminal window, Andrei wil enter the following commands:

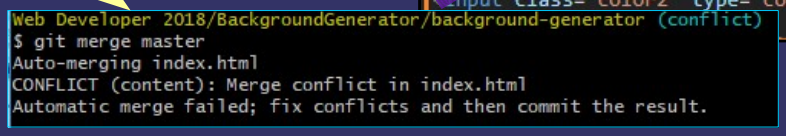
1. git branch conflict
2. git checkout conflict

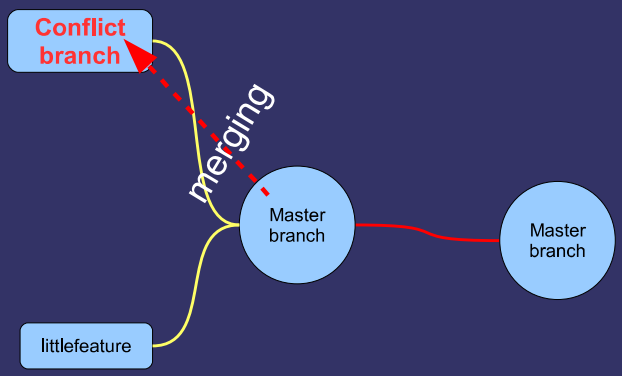
3 Andrei decided to completly remove the background generator title
4. save all



He will then go back to a terminal
And enter the following commands:

5. git add index.html
6. git commit -m“deleting title“
7. git checkout master
8. git pull
9. git checkout conflict
10 .merge master into the conflict branch.
11. type git merge master






```
Dima Mironov@Dmitry-M MINGW64 ~/Desktop/Dima Studies/Andrei Negoie/The Complete
Web Developer 2018/backgroundGenerator/background-generator (conflict)
$ git merge master
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
Automatic merge failed; fix conflicts and then commit the result.

Dima Mironov@Dmitry-M MINGW64 ~/Desktop/Dima Studies/Andrei Negoie/The Complete
Web Developer 2018/backgroundGenerator/background-generator (conflict|MERGING)
$
```

Important to read!

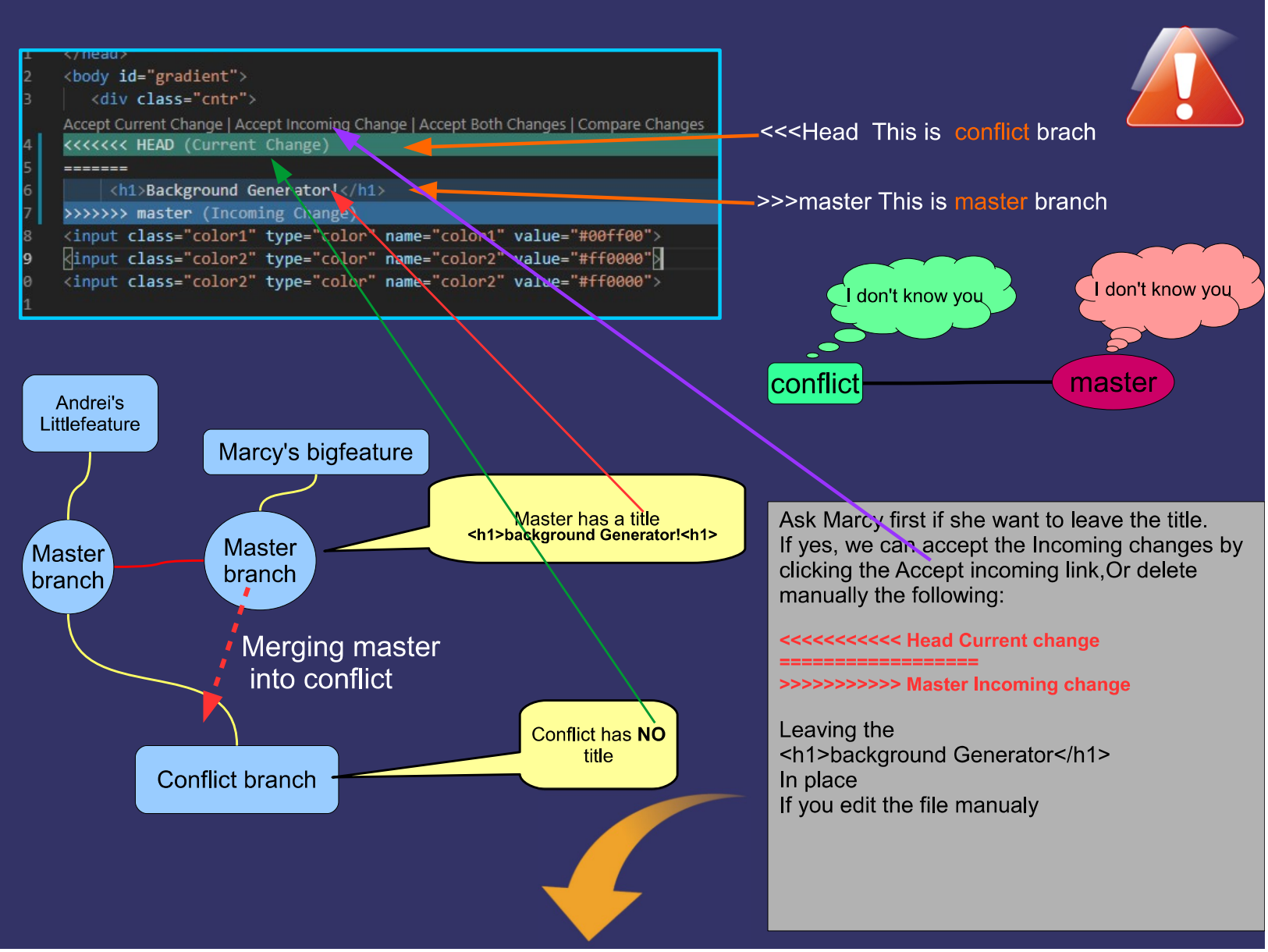
6. By now you should recieve the following error
7. If you Open a (Visual Studio Code/Sublime)
You sould see the the following output:

```
</head>
<body id="gradient">
    <div class="cntr">
Accept Current Change | Accept Incoming Change | Accept Both Changes | Co
<<<<<<< HEAD (Current Change)
=======
        <h1>Background Generator!</h1>
>>>>>>> master (Incoming Change)
<input class="color1" type="color" name="color1" value="#0
<input class="color2" type="color" name="color2" value="#f
<input class="color2" type="color" name="color2" value="#f
```

Let's understand the problem.

1. We Made a conflict brach.
2. Then we removed the title from HTML.
3. Then Andrei did **git add .** Then **git commit -m „deleting title“**
But before pushing these changes back to a GitHub, he decided to Merge whatever in the master, into his conflict branch. Yes, he could push the title-less index.html back to a GitHub. But he came up with another idea: He said let's see what is in master these days....
Let's see what is inside the master now. Let's pull it intothe conflict just to be updated. Yes sure, he didn't know that master is now different from a conflict, because he did not pulled nothing for a while (just an example)

4. So we got back into the master branch.
5. We then pulled master's **current version** (to a local computer)
6. Then we merged the master into the conflict. Just to be sure the conflict is synchronized with the master. But we have invoked an error. Because master contains a title, but a conflict branch doesn't have a title.
So we get an intuitive user frendly notification from VS code or sublime.
Saying that there is a changes to be made, and your files different from master.
So!?...Where is the catch? Why we all need this??? you say....
Okay, You worked on some branch, so once in a while you want to update Your branch with a latest changes. So it is easier to merge a master into Your **OWN** branch, rather then download files manually, copy all of its contents, then Paste all of these into your own branch. **You see the difference??? :-)**
What if a master branch contains thousands of files? You will get nuts until you copy them one by one into your branch. So We simply merging the master, and it does all the dirty work for us. without worrying we missed something.

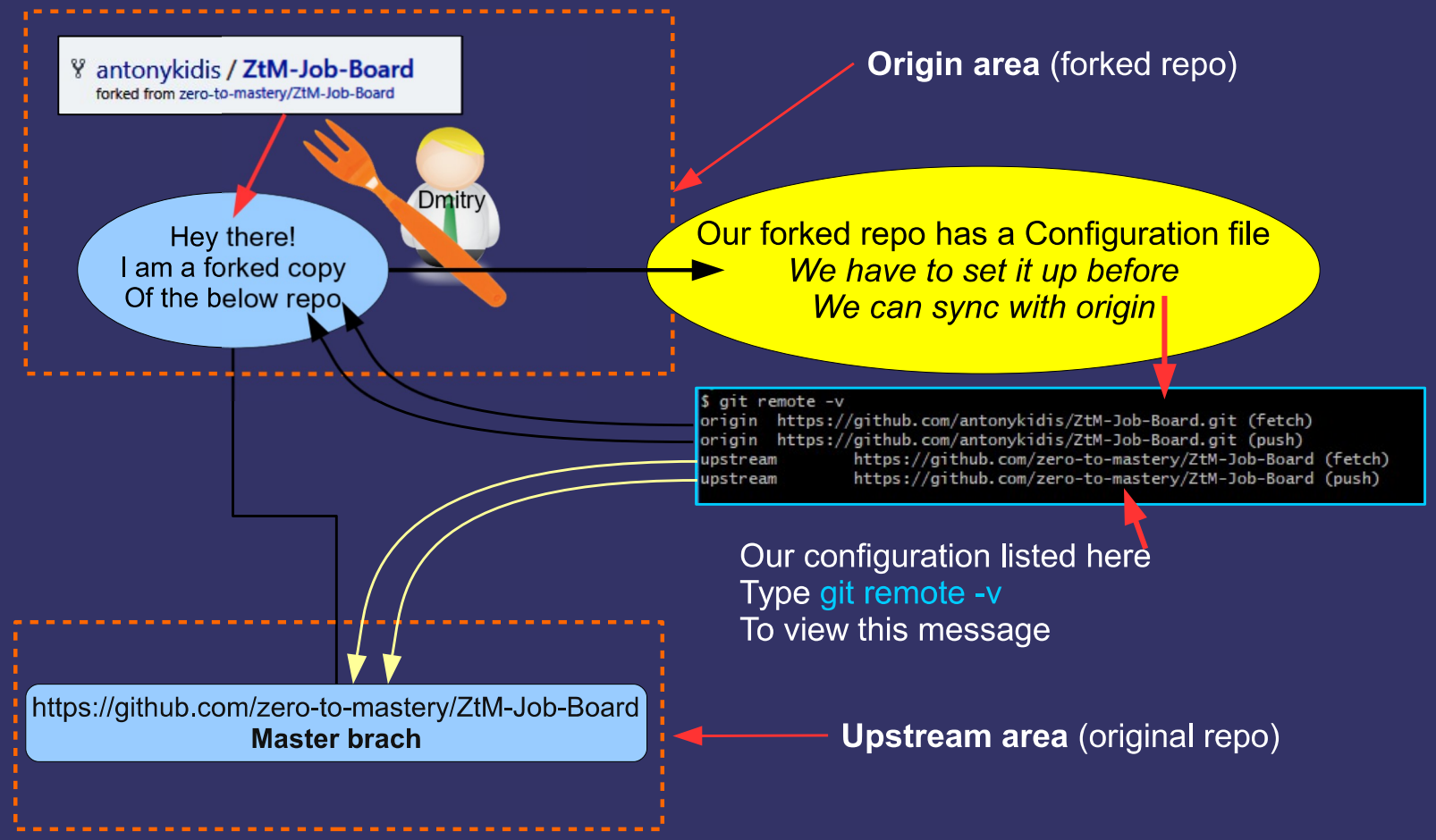

Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
<<<<<<< HEAD (Current Change)
=======
<h1>Background Generator!</h1>
>>>>>>> master (Incoming Change)
<input class="color1" type="color" name="color1" value="#00ff00">
<input class="color2" type="color" name="color2" value="#ff0000">
<input class="color2" type="color" name="color2" value="#ff0000">
<<<Head This is conflict brach
>>>master This is master branch
I don't know you
I don't know you
conflict
master
Andrei's Littlefeature
Marcy's bigfeature
Master branch
Master branch
Master has a title
<h1>background Generator!<h1>
Merging master into conflict
Conflict branch
Conflict has NO title
Ask Marcy first if she want to leave the title.
If yes, we can accept the Incoming changes by clicking the Accept incoming link,Or delete manually the following:
<<<<<<<<<< Head Current change
==========
>>>>>>>>>> Master Incoming change
Leaving the
<h1>background Generator</h1>
In place
If you edit the file manualy

Finally use the following commands

1	git add index.html
2	git commit -m “Okay we leaving the title as is“
3	git push

In the next section We'll see
how to keep your fork up to date

keep your fork up to date

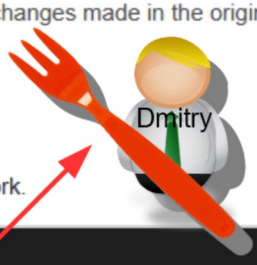


#Step1 Configuring a remote for a fork

Fetch == pull
Push == push

MAC | WINDOWS | LINUX

You must configure a remote that points to the upstream repository in Git to sync changes you make in a fork with the original repository. This also allows you to sync changes made in the original repository with the fork.



1. Open Git Bash.
2. List the current configured remote repository for your fork.

```
$ git remote -v
origin  https://github.com/YOUR_USERNAME/YOUR_FORK.git (fetch)
origin  https://github.com/YOUR_USERNAME/YOUR_FORK.git (push)
```

3. Specify a new remote *upstream* repository that will be synced with the fork.

```
$ git remote add upstream https://github.com/ORIGINAL_OWNER/ORIGINAL_REPOSITORY
```

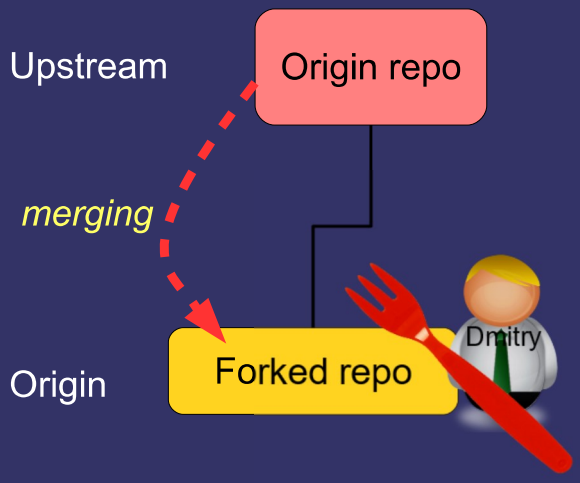
4. Verify the new upstream repository you've specified for your fork.

```
$ git remote -v
origin    https://github.com/YOUR_USERNAME/YOUR_FORK.git (fetch)
origin    https://github.com/YOUR_USERNAME/YOUR_FORK.git (push)
upstream  https://github.com/ORIGINAL_OWNER/ORIGINAL_REPOSITORY.git (fetch)
upstream  https://github.com/ORIGINAL_OWNER/ORIGINAL_REPOSITORY.git (push)
```

We all set up now

But we still have to pull
From upstream, and
Merge with origin.

step 2
Let's Pull, and merge



Done!
We now synced with the original repository

1. Open Git Bash.
2. Change the current working directory to your local project.
3. Fetch the branches and their respective commits from the upstream repository. Commits to `master` will be stored in a local branch, `upstream/master`.

```
$ git fetch upstream
remote: Counting objects: 75, done.
remote: Compressing objects: 100% (53/53), done.
remote: Total 62 (delta 27), reused 44 (delta 9)
Unpacking objects: 100% (62/62), done.
From https://github.com/ORIGINAL_OWNER/ORIGINAL_REPOSITORY
 * [new branch]      master     -> upstream/master
```

4. Check out your fork's local `master` branch.

```
$ git checkout master
Switched to branch 'master'
```

5. Merge the changes from `upstream/master` into your local `master` branch. This brings your fork's `master` branch into sync with the upstream repository, without losing your local changes.

```
$ git merge upstream/master
Updating a422352..5fdff0f
Fast-forward
 README                    |    9 -------
 README.md                 |    7 ++++++
 2 files changed, 7 insertions(+), 9 deletions(-)
 delete mode 100644 README
 create mode 100644 README.md
```

If your local branch didn't have any unique commits, Git will instead perform a "fast-forward":

```
$ git merge upstream/master
Updating 34e91da..16c56ad
Fast-forward
 README.md                 |    5 +++--
 1 file changed, 3 insertions(+), 2 deletions(-)
```

What the hell Was that?

Okay let's try „explanation for dummies“

As I had a real hard time to understand this the easy way

Without
My config file
I am a real dumbass

My forked repo

.Git/Config.file

Tell me how should I
Connect a forked repo
with GitHub's origin
repo, and keep it up to
date?

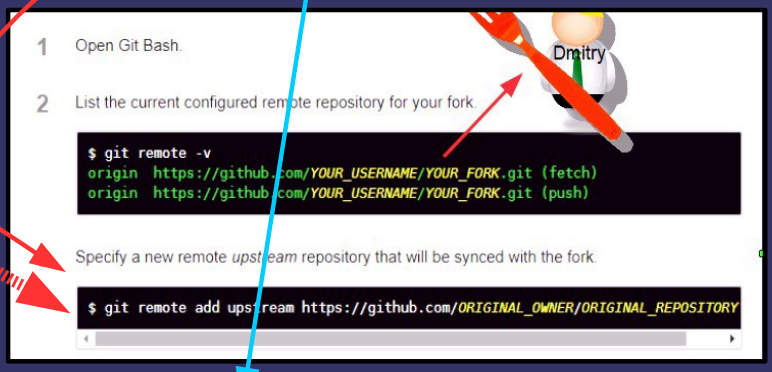
Awaiting your
instructions here.....

Git remote add upstream....

I am a „Configuration“ file.
I can do some tricks.
But tell me what should I do.

Okay seems like you want
To setup your fork, so it could
Comunicate with upstream repository.

Aha...I can now sync all the changes
You made in original repo with the fork.



I forgot to tell you that
I have a special configuration file
In my **.git folder**
Non of your friends thold you of
Its existence?
....I Kill you!

```
$ git remote -v
origin  https://github.com/antonykidis/ZtM-Job-Board.git (fetch)
origin  https://github.com/antonykidis/ZtM-Job-Board.git (push)
upstream        https://github.com/zero-to-mastery/ZtM-Job-Board (fetch)
upstream        https://github.com/zero-to-mastery/ZtM-Job-Board (push)
```

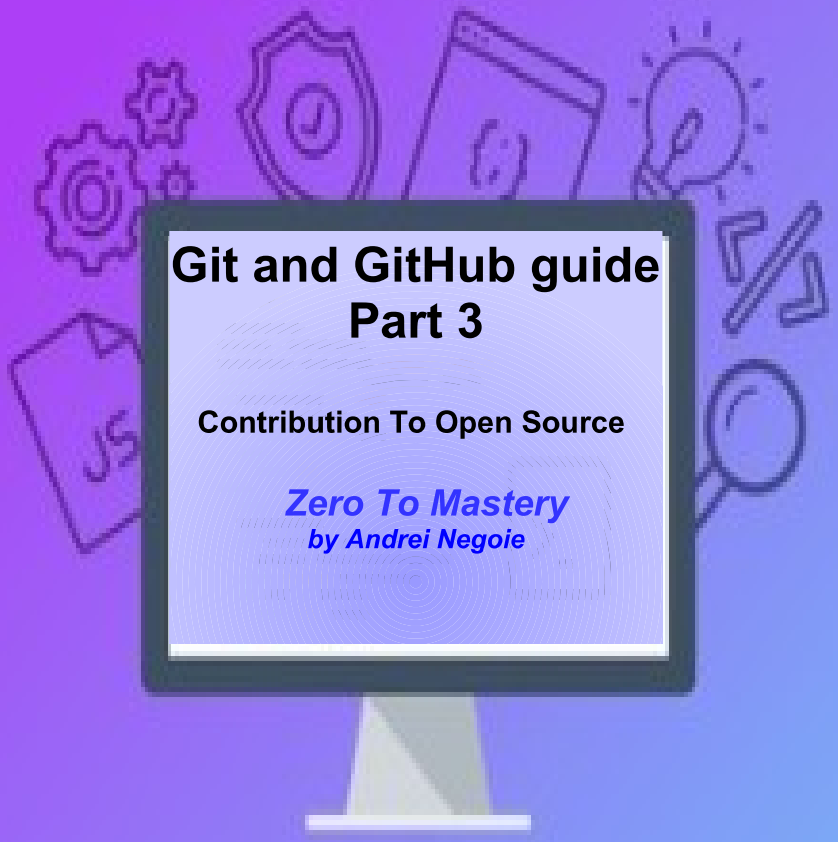
A simple representation of config file
As seen in gitbash terminal.

GitHub

Some repo is here

To sync a fork with the original repo you have to
set up comunication rules between these two in the
Forked repository configuration file (in .git) folder

TO BE CONTINUED
IN PART 3 (Contribute to the open source)



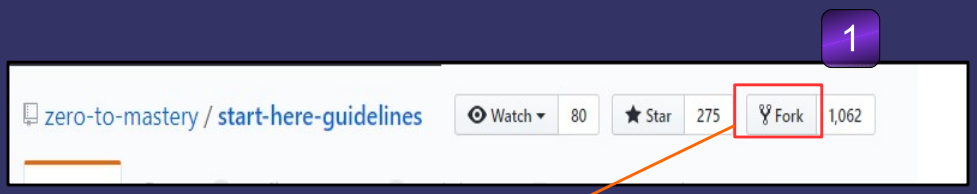
Git and GitHub guide Part 3

Contribution To Open Source

Zero To Mastery

by Andrei Negoie

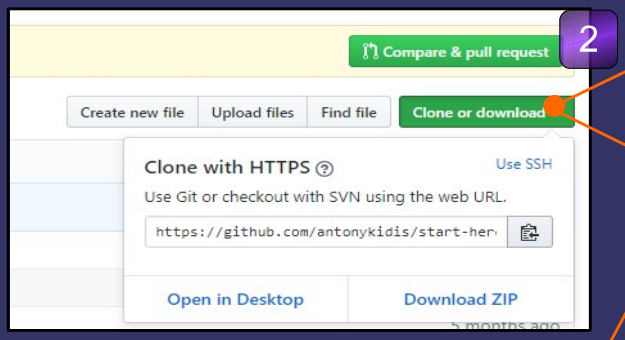
Lection 148. We start from 05:41 minutes (follow these steps)

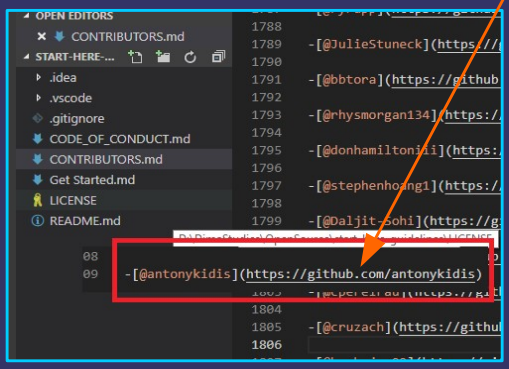


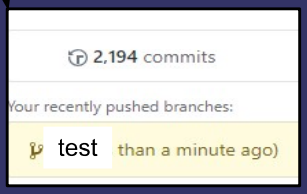
3. git clone https://github.com/antonykidis/start-here-guidelines.git
4.Cd into start-here-guidelines.
5.ls

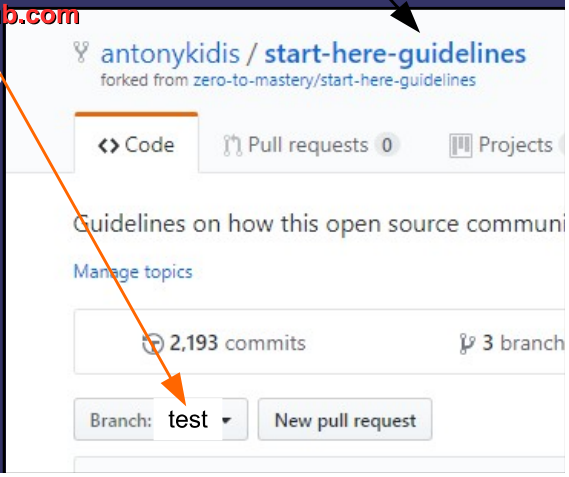
CODE_OF_CONDUCT.md CONTRIBUTORS.md 'Get Started.md' LICENSE README.md

6.Type start code . (for VS Code should open the entire folder)
7.Branch out git checkout -b test
8. Add your name to the list of contributors
9. Save all CTRL+S
10. git add .
11. git commit -m"test"
12. git push origin test
13. git push
14. go back to https://github.com
15. See a notification
16. Finish a pull request as appears in the video.







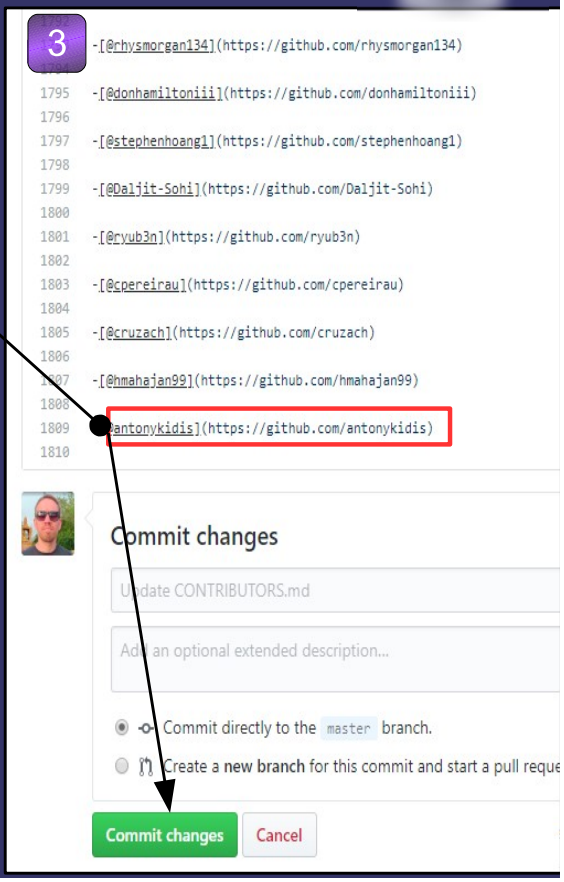


A different way adding yourself to CONTRIBUTORS.MD

We could skip the previous page and do the following

** Another way adding yourself to a CONTRIBUTORS.md list would be editing it straight on a start-here-guidlines GitHub repository.*

1. Go to the forked repo https://github.com/antonykidis/start-here-guidelines
2. At your left hand side click the CONTRIBUTORS.md file.
3. Click Edit (pencil icon)
4. Add your name to the very end of the contributors list (using the mark down syntax)
5. Commit changes.



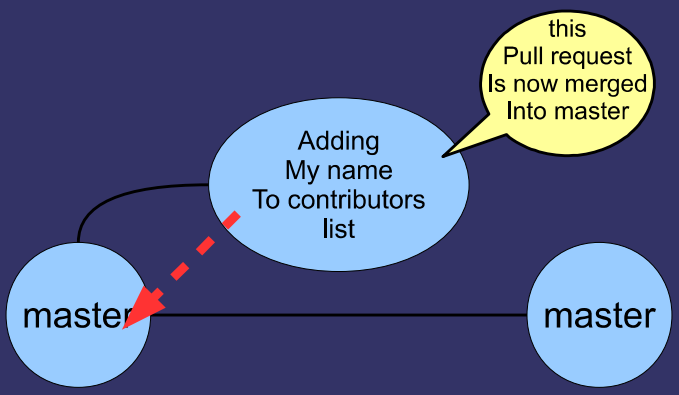
6. You will recieve a new Pul Request notification.
7. Finish this procedure by clicking Compare and Pull request button.
8. Administrators will accept this pull request and merge it into master.

Congratulations on adding yourself as a new contributor!

...Meanwhile somewhere in the cosmos

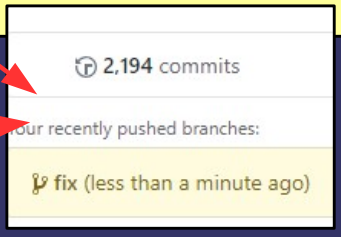
One of the administrators have merged our pull request into the master.

We've been added to a CONTRIBUTORS.md list.



After adding myself to a contriburors list via GitHub website, I finally decided to repeat these steps, but this time by cloning a repository, adding myself to the Contributors list, and pushing it back to a gitHub (as it apears in the original video)

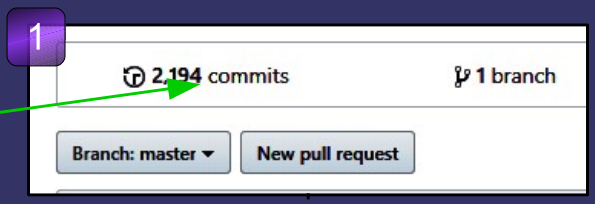
I can say that I've edited the contributor's list two times so far.
1. By editing it on a GitHub website.
2. By cloning a repo, modifying a contributors list, and pushing it back to a GitHub
3. This caused the notification to appear once again.

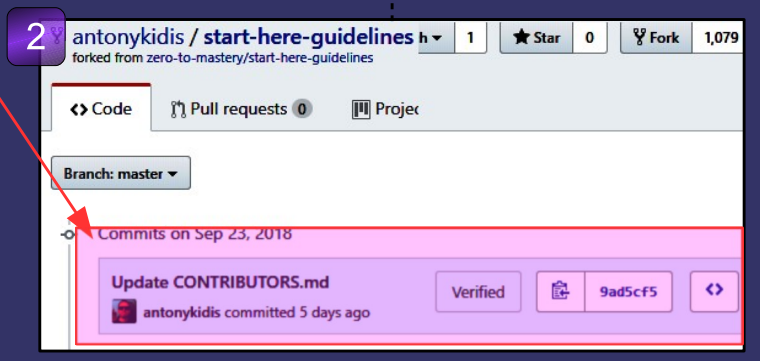


Okay I have a new pull request notification.
But I don't need it because my name is already In the contributors' list
Hmmm...
Is there any way to get rid of it?
It's a good question...

We could try to delete the commit like this

1. Go to a forked repo https://github.com/zero-to-mastery/start-here-guidelines
2. Click commits link.
2. Locate your commit.
3. Copy the commit ID
4. Go to terminal.
5. type git push origin +9ad5cf5^:master





\+ Specify what destination ref to update with what source object.
The format of a <refspec> parameter is an optional plus +, followed by the source object <src>, followed by a colon :, followed by the destination ref <dst>

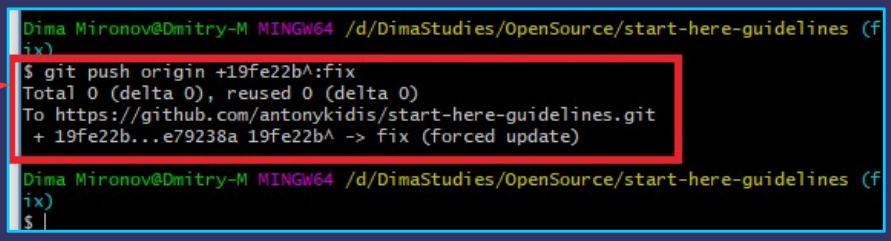
^ HEAD^ means the first parent of the tip of the current branch.

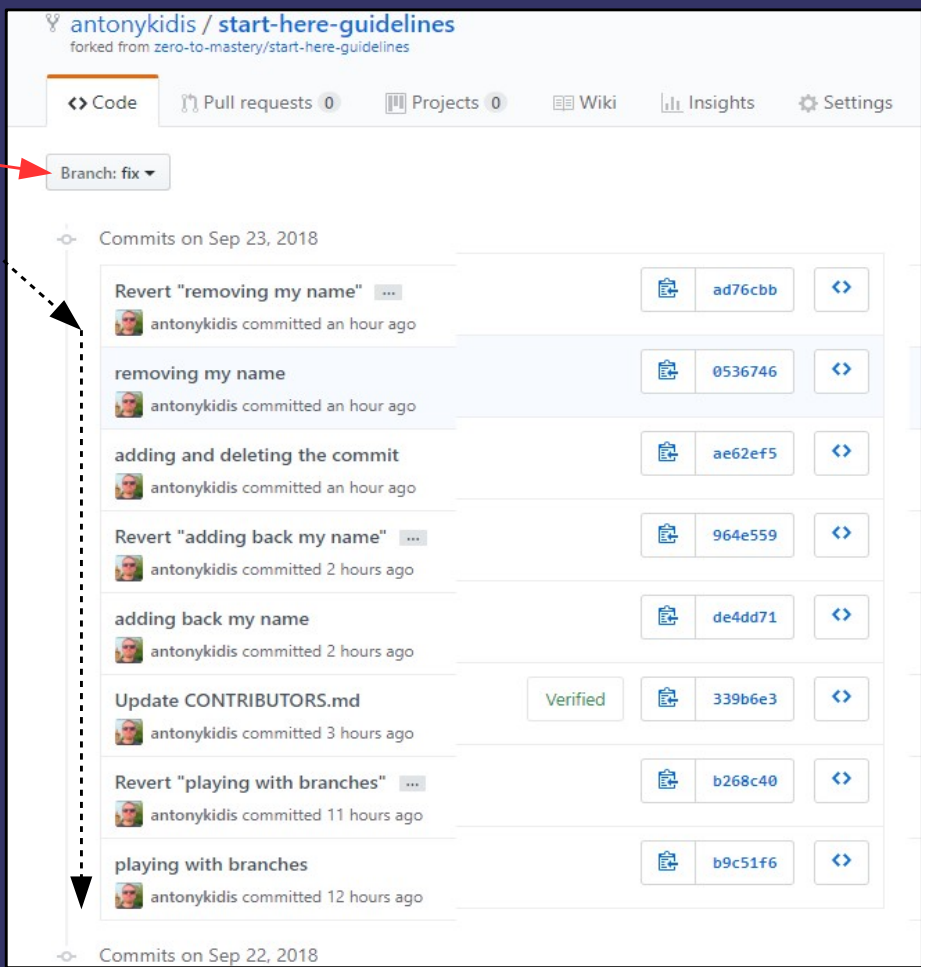
6. You will get the following output
7. Go back to commits (gitHub) and check if it's been deleted.
8. Repeat the previous steps again to delete another desired commints you want.

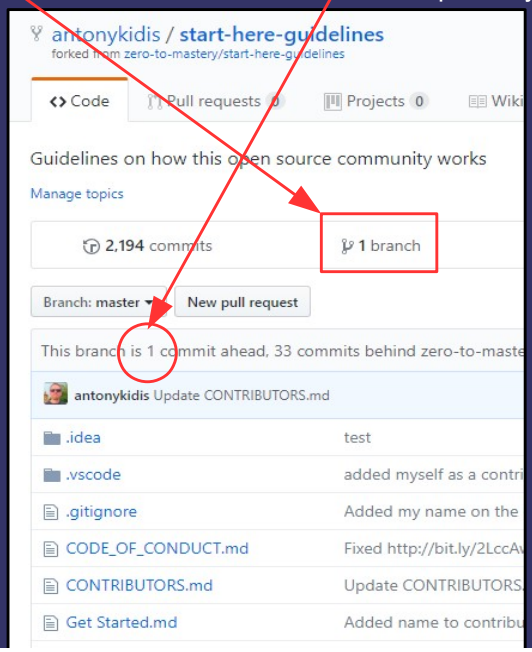
9. Once every commits is been removed, the pull request Notification will no longer appear on the GitHub website.

10. You can then delete a remote branch like this:
11. Type git push origin --delete fix
    In my case **fix** was the name of the branch

As you can see i've successfully deleted Commits and a remote branches.
We now left with only one commit, and one master branch on our forked repository.







Additional Information

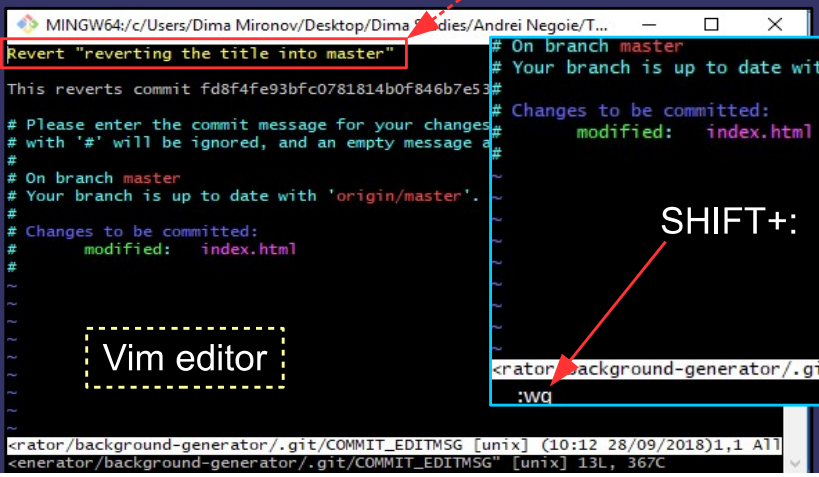
You could encounter with some unexpected things while working on the Git and GitHub section of the course.

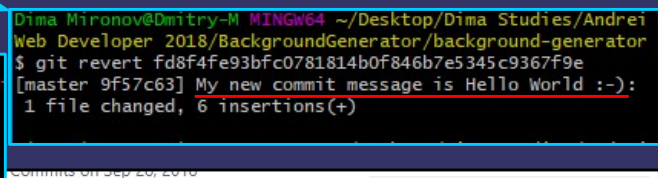
They are as follows:

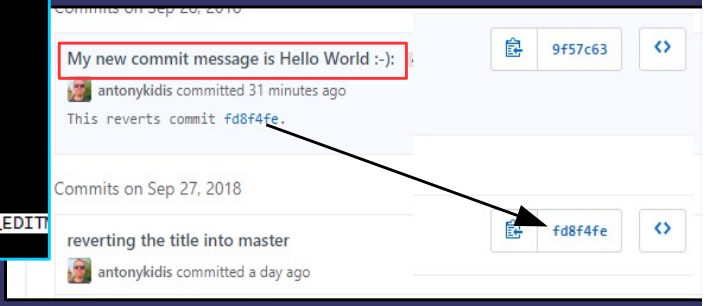
1. What if we mistakenly entered a bad commit's message. How to fix that without pushing a whole new files into the GitHub again?
2. Your bash terminal showed a weird vim editor window (only for win. Users) How to use it?

Solution: How to Edit a bad commit message.

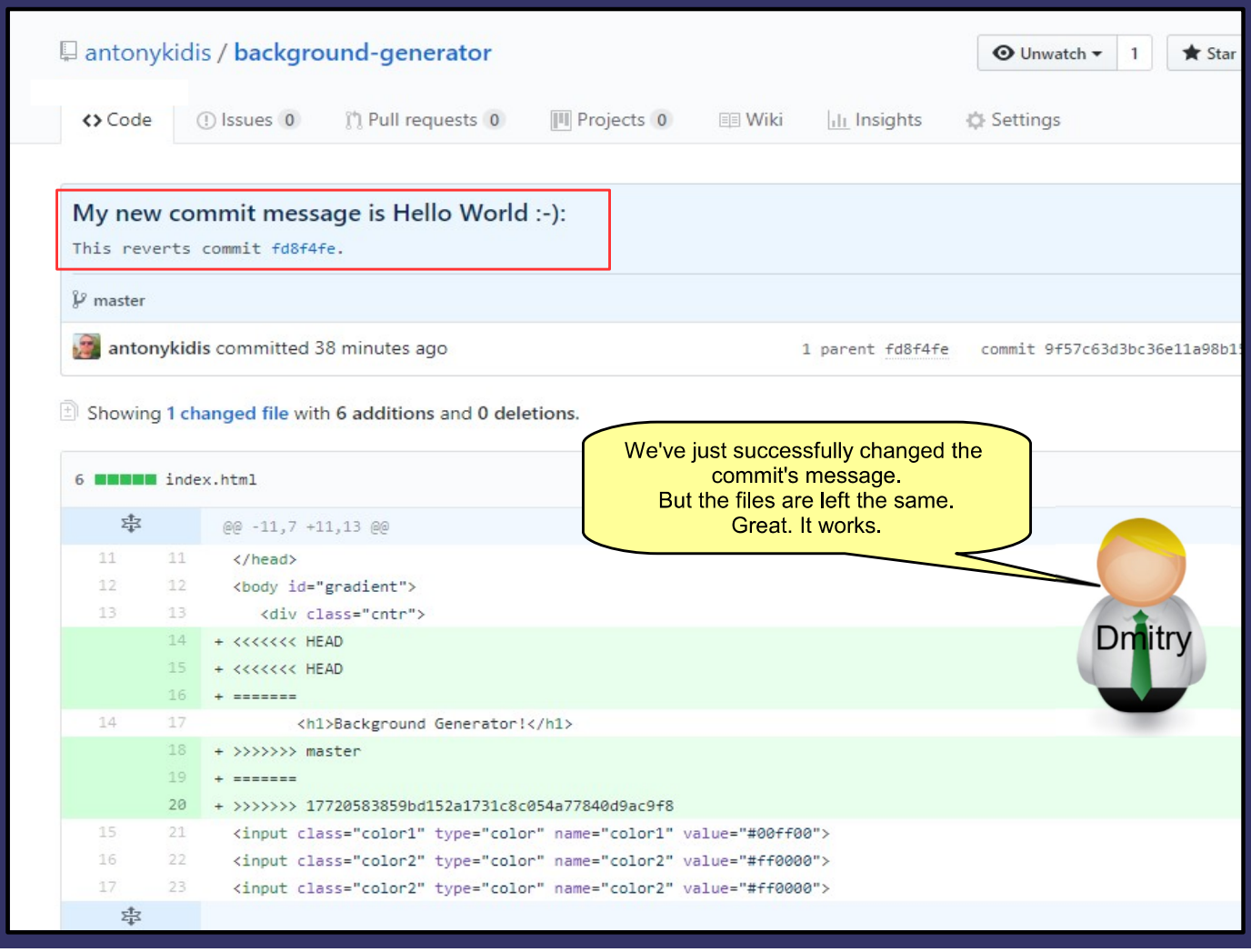
1. Go to a forked repository, open commits link, and find a commit you want to fix.
2. Copy a commit's ID to a clipboard
3. Open terminal window.
4. Use this command. git revert fd8f4fe
5. hit ENTER.
6. The vim editor window will appear.
7. The yellow text will represent a commit message.
8. Hit **insert** button on your keyboard. This will allow you to manipulate a text.(use arrows to navigate)
9. Delete the yellow text, and write hello world (for example)
10. Press ESC button to exit the text editor mode.
10. Hold down SHIFT+ : this will bring you to the vim script mode(check the left bottom corner, you will put commands there)
11. Type wq (w= write) (q=quit saving changes) and hit ENTER
12. You will recieve the following output, with updated commit message
13. type git push to apply changes







Step 14.Go back to GitHub and see the changes




The End of part 3