

Forum:[Hack Forums](#)

Thread: [Introduction in Version Control with Git](#)

Remix - September 7, 2015, 12:18 pm

Introduction in Version Control with Git

Developers around the world use different Version Control Systems (VCS) for several purposes like sharing projects, saving files or just take a look on the project history.

In this thread I want to explain why they use it and I'll give a short introduction into the Git basics.

Why Version Control matters

Imagine you're working on some type of software with some other developers and you want to change something on a file which somebody else is currently editing.

That would be a real waste of time to look who did what etc.

With Git or other Version Control Systems everybody can work on individual branches and merge the changes later to the master branch.

How Version Control works

It's called Version Control because it controls the different versions you're working on.

It "stores" older versions in its history too.

A project is called repository.

It's a collection of all the versions along with some special informations like the changes, a custom description and who committed the changes.

The VCS doesn't know when a new version is finished so you have to tell it with committing the changes. It stores every changes in his history that you can easily check when which change was made.

Everything is controlled in hidden folders and files that the user doesn't have to care for something.

Furthermore it offers you to share your code with others that you can collaborate on your project.

Getting started with Git

First of all you have to install a Git version.

If you're using Windows you can download the latest Git version here: [Download](#)

If you're running a Linux system you can install it on that way: `sudo apt-get install git`

Creating the first repository

To create a repository you have to use the command `git init <rep_name>`.

So for example use the command `git init first_rep`.

Now it should display something like that:

Code Initialized empty Git repository in c:/Users/xirem_000/Desktop/moin/first_rep/.git/

You've created an empty repository successfully.

If you run the ls command now you should see a new directory.

If you run ls -a in the new directory you'd see that there is a hidden directory named .git

Notice that the name of the folder doesn't matter at all.

As long as the .git folder exists the index files will be tracked.

To delete the repository from your folder you just have to delete the .git folder.

Committing changes

For testing just create a file with the touch command and write something in it. (works on Windows in the git bash too)

Every repository should have a readme file or at least possible users could benefit from it.

So create a readme file with touch README.

To add the file to the index you have to use the command git add.

So just type git add README.

Now the readme file is added to the staging area.

The staging area just includes all files which will be committed.

To commit the changes you'll use the command git commit.

If you use it without parameters the default console editor will open with a commit message template.

But you could use the -m parameter following by a string which sets the commit message.

You could use something like "added readme file" for now.

The command would look like this: git commit -m "added readme file".

If you run it you'd get a message like that:

```
Code [master (root-commit) 7026bb2] added readme file
```

```
1 file changed, 1 insertion(+)
```

```
create mode 100644 README
```

That just means that the commit was successfully and the changes were committed.

If you've added a file to the tracking index you can easily commit it again without adding it to the commits again and again.

The parameter -a commits every changed files on the tracking list.

So it would look like this: git commit -a -m "message".

To see which files are untracked just use the git status command.

Amongst other things it will show you which files are untracked.

The commit history

To show the commit history you just have to use the git log command.

Then you'll get a list of the commits with the autor, the date, the id and the message.

Closing words

This was my first tutorial ever so don't hesitate to leave any tips or suggestions.

I hope you enjoyed this guide.

If it's desired more guide about git will follow.

And excuse me if my english isn't perfect. English lessons in Germany aren't the best.

Greetings

iRemix

Remix - September 9, 2015, 6:12 am

Thanks for your feedback

I'm already planning some more guides ?

Lexxi - September 9, 2015, 6:14 am

great tut, will keep in mind if i ever need to follow it, goodjob.

rageSpreed - September 9, 2015, 6:28 am

Thanks man.

Nice work, it will be awesome if you can make one for github.

csrk - September 9, 2015, 8:26 am

(Sep 9, 2015 06:28 AM)rageSpreed Wrote: [\[CITE\]](#) Thanks man.

Nice work, it will be awesome if you can make one for github.

This does work perfectly fine with github. :-) Just google how to use git with github or something.

Remix - September 9, 2015, 9:41 am

(Sep 9, 2015 08:26 AM)Rewbert Wrote: [\[CITE\]](#) This does work perfectly fine with github. :-) Just google how to use git with github or something.

Well for github you need to work with remote repositories.

In this tutorial they aren't covered.

Maybe in the next tutorial.

Ivica.Lovro - September 9, 2015, 10:49 am

Nice tutorial mate. I always asked me, what everybody meant by it.

Now I get it and I know the basics and for what purposes I should use it.

Thank you!

Remix - September 9, 2015, 12:19 pm

Glad to see that you like it.