

<http://www.sitepoint.com/programming/>

Git for Beginners



Shaumik Daityari(<http://www.sitepoint.com/author/sdaityari/>)

May 13, 2014

It is a general tendency of human beings to resist change. Unless [Git \(http://git-scm.com/\)](http://git-scm.com/) was around when you started with version control systems, chances are that you are comfortable with [Subversion \(http://subversion.apache.org/\)](http://subversion.apache.org/). Often, people say that Git is too complex for beginners. Yet, I beg to differ!

In this tutorial, I will explain how to use Git for your personal projects. We will assume you are creating a project from scratch and want to manage it with Git. After going through the basic commands, we will have a look at how you could put your code in the cloud using GitHub.

We will talk about the basics of Git here — how to initialize your projects, how to manage new and existing files, and how to store your code in the cloud. We will avoid relatively complex parts of Git like branching, as this tutorial is intended for beginners.

Installing Git

The official website of Git has [detailed information about installing \(http://git-scm.com/book/en/Getting-Started-Installing-Git\)](http://git-scm.com/book/en/Getting-Started-Installing-Git) on Linux, Mac, or Windows. In this case, we would be using Ubuntu 13.04 for demonstration purposes, where you install Git using `apt-get`.

```
sudo apt-get install git
```

Initial Configuration

Let's create a directory inside which we will be working. Alternately, you could use Git to manage one of your existing projects, in which case you would not create the demo directory as below.

```
mkdir my_git_project  
cd my_git_project
```

The first step is to initialize Git in a directory. This is done using the command `init`, which creates a `.git` directory that contains all the Git-related information for your project.

```
donny@ubuntu:~$ mkdir my_git_project  
donny@ubuntu:~$ cd my_git_project/  
donny@ubuntu:~/my_git_project$ git init  
Initialized empty Git repository in /home/donny/my_git_project/.git/  
donny@ubuntu:~/my_git_project$
```

```
git init
```

Next, we need to configure our name and email. You can do it as follows, replacing the values with your own name and email.

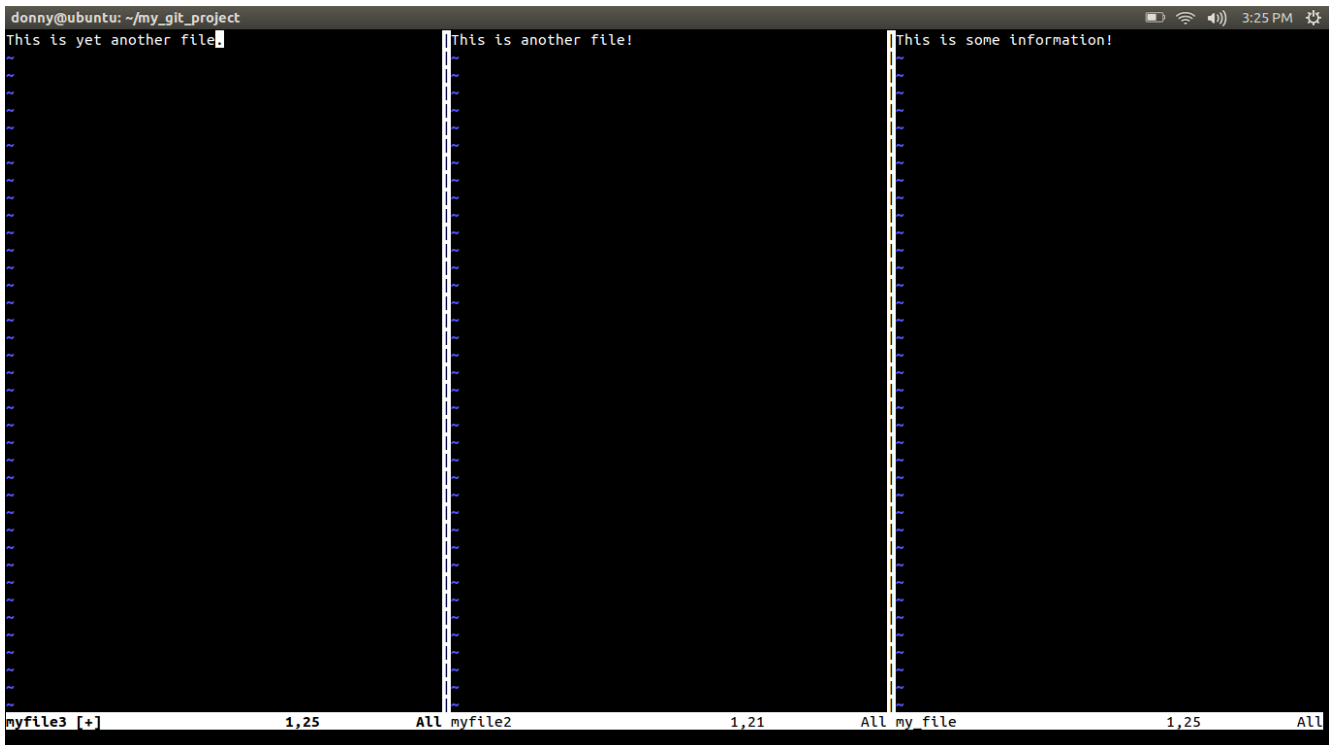
```
git config --global user.name 'Shaumik'  
git config --global user.email 'sd@gmail.com'  
git config --global color.ui 'auto'
```

It is important to note that if you do not set your name and email, certain default values will be used. In our case, the username 'donny' and email 'donny@ubuntu' would be the default values.

Also, we set the UI color to `auto` so that the output of Git commands in the terminal are color coded. The reason we prefix `--global` to the command is to avoid typing these config commands the next time we start a Git project on our system.

Staging Files for Commit

The next step is to create some files in the directory. You could use a text editor like Vim (<http://www.vim.org/>). Note that if you are going to add Git to an already existing directory, you do not need to perform this step.



Check the Status of Your Repository

Now that we have some files in our repository, let us see how Git treats them. To check the current status of your repository, we use the `git status` command.

```
git status
```

```
donny@ubuntu:~/my_git_project$ git status
# On branch master
#
# Initial commit
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       my_file
#       my_file2
#       my_file3
nothing added to commit but untracked files present (use "git add" to track)
donny@ubuntu:~/my_git_project$
```

Adding Files for Git to Track

At this point, we do not have any files for Git to track. We need to add files specifically to Git order to tell Git to track them. We add files using `add`.

```
git add my_file
```

Checking the status of the repository again shows us that one file has been added.

```
donny@ubuntu:~/my_git_project$ git add my_file
donny@ubuntu:~/my_git_project$ git status
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   (use "git rm --cached <file>..." to unstage)
#
#       new file:   my_file
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       myfile2
#       myfile3
donny@ubuntu:~/my_git_project$
```

To add multiple files, we use the following (note that we have added another file for demonstration purposes.)

```
git add myfile2 myfile3
```

You could use `git add recursively`, but be careful with that command. There are certain files (like compiled files) that are usually kept out of the Git repository. If you use `add recursively`, it would add all such files, if they are present in your repository.

Removing Files

Let's say you have added files to Git that you do not want it to track. In such a situation, you tell Git to stop tracking them. Yet, running a simple `git rm` will not only remove it from Git, but will also remove it from your local file system

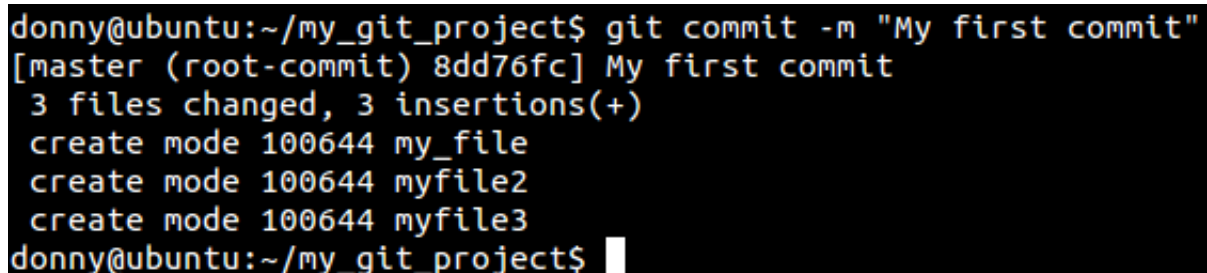
(<http://stackoverflow.com/questions/1143796/git-remove-a-file-from-the-repository-without-deleting-it-from-the-local-filesy>) as well! To tell Git to stop tracking a file, but still keep it on your local system, run the following command:

```
git rm --cached [file_name]
```

Committing Changes

Once you have staged your files, you can `commit` them into Git. Imagine a commit as a snapshot in time where you can return back to access your repository at that stage. You associate a commit message with every commit, which you can provide with the `-m` prefix.

```
git commit -m "My first commit"
```

A terminal window screenshot showing the execution of a git commit command. The prompt is 'donny@ubuntu:~/my_git_project\$'. The command entered is 'git commit -m "My first commit"'. The output shows the commit was successful on the master branch, with a commit hash of 8dd76fc. It lists three files changed: my_file, myfile2, and myfile3, all created with mode 100644.

```
donny@ubuntu:~/my_git_project$ git commit -m "My first commit"
[master (root-commit) 8dd76fc] My first commit
 3 files changed, 3 insertions(+)
 create mode 100644 my_file
 create mode 100644 myfile2
 create mode 100644 myfile3
donny@ubuntu:~/my_git_project$
```

Provide a useful commit message because it helps you in identifying what you changed in that commit. Avoid overly general messages like “Fixed bugs”. If you have an issue tracker, you could provide messages like “Fixed bug #234”. It’s good practice to prefix your branch name or feature name to your commit message. For instance, “Asset management – Added feature to generate PDFs of assets” is a meaningful message.

Git identifies commits by attaching a long hexadecimal number to every commit. Usually, you do not need to copy the whole string, and the first 5-6 characters are enough to identify your commit. In the screenshot, notice that `8dd76fc` identifies our first commit.

Further Commits

Let's now change a few files after our first commit. After changing them, we notice through `git status` that Git notices the change in the files that it is tracking.

```
donny@ubuntu:~/my_git_project$ git status
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   my_file
#       modified:   myfile2
#
no changes added to commit (use "git add" and/or "git commit -a")
donny@ubuntu:~/my_git_project$
```

You can check the changes to the tracked files from the last commit by running `git diff`. If you want to have a look at the changes to a particular file, you can run `git diff <file>`.

```
donny@ubuntu:~/my_git_project$ git diff
diff --git a/my_file b/my_file
index 362eab3..0a0bd57 100644
--- a/my_file
+++ b/my_file
@@ -1,3 @@
-This is some information!
+
+I am changing the content of this file.
diff --git a/myfile2 b/myfile2
index d4a2d15..ec4dcc2 100644
--- a/myfile2
+++ b/myfile2
@@ -1,1 @@
-This is another file!
+This is another file! Changing this file too.
donny@ubuntu:~/my_git_project$
```

You need to add these files again to stage the changes in tracked files for the next commit. You can add all the tracked files by running:

```
git add -u
```

You could avoid this command by prefixing `-a` to `git commit`, which adds all changes to tracked files for a commit. This process, however, is very dangerous as it can be damaging. For instance, let's say you opened a file and changed it by mistake. If

you selectively stage them, you would notice changes in each file. But if you add `-a` to your commit, all files would be committed and you would fail to notice possible errors.

Once you have staged your files, you can proceed to a commit. I mentioned that a message can be associated with every commit, which we entered by using `-m`.

However, it is possible for you to provide multi-line messages by using the command `git commit`, which opens up an interactive format for you to write!

```
git commit
```

```
GNU nano 2.2.6          File: .git/COMMIT_EDITMSG          Modified
- Changed two files
- This looks like a cooler interfact to write commit
messages
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   my_file
#       modified:   myfile2
#
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text   ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text ^T To Spell
```

Managing of Your Project

To check the history of your project, you can run the following command.

```
git log
```

```
donny@ubuntu:~/my_git_project$ git log
commit f934591cd1c04e4009dfa76a9684dda73cb30260
Author: Shaumik <sdaityari@gmail.com>
Date: Tue May 6 15:31:00 2014 +0530

    - Changed two files
    - This looks like a cooler interfact to write commit
      messages

commit 8dd76fc3c4ba77b2ee795aed942cf6e9eaf5204d
Author: Shaumik <sdaityari@gmail.com>
Date: Tue May 6 15:28:03 2014 +0530

    My first commit
donny@ubuntu:~/my_git_project$
```

This shows you the entire history of the project — which is a list of all the commits and their information. The information about a commit contains the commit hash, author, time and commit message. There are many variations of `git log`, which you could explore once you understand the concept of a `branch` in Git. To view the details of a particular commit and the files that were changed, run the following command:

```
git show <hash>
```

Where `<hash>` is the hex number associated with the commit. As this tutorial is for beginners, we will not cover how to get back to the state of a particular commit in time or how to manage branches.


```
donny@ubuntu:~/my_git_project$ git show 8dd76fc3c4ba7
commit 8dd76fc3c4ba77b2ee795aed942cf6e9eaf5204d
Author: Shaumik <sdaityari@gmail.com>
Date: Tue May 6 15:28:03 2014 +0530

    My first commit

diff --git a/my_file b/my_file
new file mode 100644
index 0000000..362eab3
--- /dev/null
+++ b/my_file
@@ -0,0 +1 @@
+This is some information!
diff --git a/myfile2 b/myfile2
new file mode 100644
index 0000000..d4a2d15
--- /dev/null
+++ b/myfile2
@@ -0,0 +1 @@
+This is another file!
diff --git a/myfile3 b/myfile3
new file mode 100644
index 0000000..0eb26d0
--- /dev/null
+++ b/myfile3
@@ -0,0 +1 @@
+This is yet another file.
donny@ubuntu:~/my_git_project$
```

Putting Your Code in the Cloud

Once you have learned how to manage your code on your system, the next step is to put it in the cloud. Since Git doesn't have a central server like Subversion, you need to add each source to collaborate with others. That is where the concept of remotes comes in. A `remote` refers to a remote version of your repository.

If you wish to put your code in the cloud, you could create a project on [GitHub](https://github.com/) (<https://github.com/>), [GitLab](https://www.gitlab.com/) (<https://www.gitlab.com/>), or [BitBucket](https://bitbucket.org/) (<https://bitbucket.org/>) and push your existing code to the repository. In this case, the remote repository in the cloud would act as a remote to your repository. Conveniently, a remote to which you have write access is called the `origin`.

After you create a remote repository, you have the ability to add a remote `origin` and then push the code to the origin.

```
git remote add origin https://github.com/sdaityari/my_git_project.git
git push -u origin master
```

```
donny@ubuntu:~/my_git_project$ git remote add origin https://github.com/sdaityari/my_git_project.git
donny@ubuntu:~/my_git_project$ git push -u origin master
Username for 'https://github.com': sdaityari
Password for 'https://sdaityari@github.com':
Counting objects: 9, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (9/9), 776 bytes, done.
Total 9 (delta 0), reused 0 (delta 0)
To https://github.com/sdaityari/my_git_project.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
donny@ubuntu:~/my_git_project$
```

Conclusion

Git is full of features and we have covered just the basics here. I hope that this post helped you get started with Git. If you have any issues or questions about getting started, let us know in the comments below.

Tags: [git](http://www.sitepoint.com/tag/git/) (<http://www.sitepoint.com/tag/git/>), [git tutorial](http://www.sitepoint.com/tag/git-tutorial/) (<http://www.sitepoint.com/tag/git-tutorial/>), [subversion](http://www.sitepoint.com/tag/subversion/) (<http://www.sitepoint.com/tag/subversion/>), [SVN](http://www.sitepoint.com/tag/svn/) (<http://www.sitepoint.com/tag/svn/>), [version control](http://www.sitepoint.com/tag/version-control-2/) (<http://www.sitepoint.com/tag/version-control-2/>)



[Shaumik Daityari](http://www.sitepoint.com/author/sdaityari/) (<http://www.sitepoint.com/author/sdaityari/>)

[🐦](https://twitter.com/ds_mik) (https://twitter.com/ds_mik) [g+](https://plus.google.com/+ShaumikDaityari) (<https://plus.google.com/+ShaumikDaityari>) [f](https://www.facebook.com/shaumikdaityari) (<https://www.facebook.com/shaumikdaityari>) [🔗](https://github.com/sdaityari/) (<https://github.com/sdaityari/>)

Shaumik is an optimist, but one who carries an umbrella. An undergrad at Indian Institute of Technology Roorkee and the co-founder of The Blog Bowl, he loves writing, when he's not busy keeping the blue flag flying high.

Recommend 4

Share

Sort by Best



Join the discussion...

Ralph Mason SitePoint Staff · 2 years ago

Git for beginners ... "sudo apt-get install git" ... Hehe, you might need to define what a "beginner" is.

10 ^ | v · Reply · Share ›

Pastoolio → Ralph Mason · 3 months ago

"The official website of Git has detailed information about installing on Linux, Mac, or Windows."

Beginners should learn to follow the links in articles.

^ | v · Reply · Share ›

13 Pixlar · 2 years ago

Pretty Good tutorial. One thing that would be nice is if you expanded the section about Github to include setting up SSH Keys.

3 ^ | v · Reply · Share ›

Shaumik Daityari → 13 Pixlar · 2 years ago

Hi,

We have tackled that topic in the next tutorial.

<http://www.sitepoint.com/getting-started-with-git/>

3 ^ | v · Reply · Share ›

Sathya Rajan · 2 years ago

Resist changes??? God dam man. I am trying to embrace change, create a more structured workflow and I am spending fuck a lot of time on learning git than spending time in my project. No body gives a clear direction where to begin, how to go about it. A steep learning curve is not fucking helpful with git. Anyway. Thank a lot you for your explanation. It help me with this shit I've been struggling. Fuck all the other documentations

2 ^ | v · Reply · Share ›

M S i N Lund · 2 years ago

Any good tutorial for Windows-users?

Also i would prefer not to have to dig around in dos anymore.

2 ^ | v · Reply · Share ›

Shaumik Daityari → M S i N Lund · 2 years ago

Hi MS,

In my opinion, the rich features of git can be explored only through the terminal, no matter how good your GUI client is. However, if you want to

work with a Windows client, I suggest installing GitHub for Windows(<https://windows.github.com/>), which (obviously) syncs your repositories in GitHub. The documentation (<https://help.github.com/article...> to get started should be enough, given you know the basics of git.

1 ^ | v • Reply • Share ›

Fezot ➔ M S i N Lund • 2 years ago

Try TortoiseGit. It has excellent gui...

^ | v • Reply • Share ›

This comment was deleted.

M S i N Lund ➔ Guest • 2 years ago

Oh really?

Let me paste in a list of strong arguments proving that its, still 2014, a piece of shit UI:

^v

3 ^ | v • Reply • Share ›



Nilpo ➔ M S i N Lund • 2 years ago

It's not a UI. Your expectations may be too high. I'm an avid Windows power user and I've been writing about and pushing Windows for almost two decades. If you're not using the command line in Windows, you're still not experiencing it's full power. Your allowing the developers and the UI to cherry pick what you can and can't do. The command line should be a tool in your arsenal, not something to be avoided. There are some things that just don't translate well to a GUI.

1 ^ | v • Reply • Share ›



Bwian ➔ M S i N Lund • 2 years ago

SmartGit/Hg (syntevo.com/smartgithg) is the best GUI solution I've come across. It's free for non-commercial use, is intuitive to use, and is packed with features!

^ | v • Reply • Share ›

M S i N Lund ➔ M S i N Lund • 2 years ago

Also:

Do you have to use github.com ?

I really hate low contrast sites.

^ | v • Reply • Share ›

Shaumik Daityari ➔ M S i N Lund • 2 years ago

If you use the GitHub client, then yes- it would sync your repositories to your GitHub account.

^ | v • Reply • Share ›

M S i N Lund → Shaumik Daityari · 2 years ago

So its not something you can just install and run locally,
without involving the almighty cloud at all?

^ | v · Reply · Share ›

Shaumik Daityari → M S i N Lund · 2 years ago

Yes, you can. Even for the GitHub client, you can use it to manage your files locally.

That being said, I keep my code on the cloud because I don't want to lost it- ever. If you want private repos, you can try BitBucket or the GitLab cloud.

1 ^ | v · Reply · Share ›

Shaumik Daityari → M S i N Lund · 2 years ago

Another option though is keeping your code within your network (LAN perhaps?) All the different repositories can be used as remotes. However, if you plan to do that, I would strongly using a self hosted version of GitLab.

^ | v · Reply · Share ›

Steven Hall → M S i N Lund · 2 years ago

If you're looking for a GUI, I would highly recommend Sourcetree for Windows or Mac - <http://www.sourcetreeapp.com>

Jetbrain's IDEs also have Git support built in. I almost never use the command line for Git tasks, personally...

I'm sure that's what puts most beginners off...

^ | v · Reply · Share ›

manager dave · 2 months ago

nice overview! I created a blog post that contains a brief guide to git for managers here: <http://softwaremanagementblog....>

^ | v · Reply · Share ›

sluge · 7 months ago

Please write Part2 of your tutorial

^ | v · Reply · Share ›

ZJ · 9 months ago

thank you..

simple.easy to undestand.

can you please add "branching" tutorial

^ | v · Reply · Share ›

Luiz Gustavo Martins · a veaar ago

Very good.

^ | v · Reply · Share ›

Tanveer · a year ago

thanks for the nice article

^ | v · Reply · Share ›

Josip Pokrajcic · 2 years ago

For everyone interested in good understanding of how Git works there is a great tutorial on [Lynda.com](https://lynda.com) by Kevin Skoglund. I have used Subversion in the past and decided to learn using Git too and that tutorial helped me alot.

^ | v · Reply · Share ›

Ramesh Babu Y · 2 years ago

nice simple tutorial

^ | v · Reply · Share ›



Felix · 2 years ago

I don't get it. I am not a frequent user of the terminal...

^ | v · Reply · Share ›



DEY · 2 years ago

I think the first thing to know in order to understand git or vcs, is the files ****are managed***** by the the vcs. The following example is really appreciated by some beginners and I remember I was confused about it :

```
git init
ls -l
> total 0
git checkout -b experimental
echo "hello world" > file1
git add .
git commit -m "a new hello world"
ls
> file1
git checkout master
ls -l
```

> total 0

^ | v · Reply · Share ›

VLC · 2 years ago

The reason git is hard for beginners is because most people would like to start with a project already existing in github and this happens to be one part that's rarely covered in beginner tutorials. Even in your tutorial, I can't figure out how the local and the cloud projects relate.

^ | v · Reply · Share ›

Shaumik Daityari → VLC · 2 years ago

Hi VLC,

Just an update on this. We have published a new tutorial that deals with already existing Git repositories and how to manage them.

<http://www.sitepoint.com/getti...>

1 ^ | v · Reply · Share ›

Dante → VLC · 2 years ago

After you get the grip on the basics read this tutorial:

<http://nvie.com/posts/a-succes...> - really good Git model....

1 ^ | v · Reply · Share ›

Shaumik Daityari → Dante · 2 years ago

I feel it's still might be overwhelming to someone who is new to Git.

^ | v · Reply · Share ›

Dante → Shaumik Daityari · 2 years ago

that's why I added "After you get the grip on the basics" :) I adopted this model in my work and once you get use to it it's so natural and easy...

1 ^ | v · Reply · Share ›

Shaumik Daityari → VLC · 2 years ago

Hi VLC,

Git is a distributed version control system - which means that many copies of the repository exist, and none of them is centrally located or treated as superior. Imagine the cloud project as just another version of the same repository that you have in your local system.

If you want to work on an already existing project, just clone the repository (For instance, run "git clone <https://github.com/sdaityari/m...>"). Once it has been cloned, you can work on it normally, treating it as yet another local repository, and making file changes or commits just as shown here.

As far as as this tutorial goes, I tried to cover the basic concepts which would allow you to start working with git. I will soon come up with another one which would concentrate on the ideas of branching and working in a team.

1 ^ | v · Reply · Share ›

Radek → Shaumik Daityari · 2 years ago

I hope you will continue this topic. Looking forward to read articles about branching (real life examples how can it be useful)

3 ^ | v · Reply · Share ›

Shaumik Daityari → Radek · 2 years ago

Hi Radek,

We have come up with a tutorial which deals with branches, merges and conflicts among more Git related issues.

<http://www.sitepoint.com/getti...>

1 ^ | v • Reply • Share ›

COURSES >

([https://www.sitepoint.com/premium/content_types\[\]=Course&utm_source=sitepoint&utm_medium=related](https://www.sitepoint.com/premium/content_types[]=Course&utm_source=sitepoint&utm_medium=related))

3:07:36

JavaScript: Next Steps

M. David Green

★★★★★

(https://www.sitepoint.com/premium/course/javascript-next-steps-2921/?utm_source=sitepoint&utm_medium=related)

1:11:20

React The ES6 Way

Darin Haener

★★★★☆

(https://www.sitepoint.com/premium/course/react-the-es6-way-2914/?utm_source=sitepoint&utm_medium=related)

1:49:07

Your First Meteor 1.2 Application

David Turnbull



https://www.sitepoint.com/premium/cou-first-meteor-1-2-application-2919/?utm_source=sitepoint&utm_medium=relat

BOOKS >

[https://www.sitepoint.com/premium/q=&content_types\[\]=Book&utm_s](https://www.sitepoint.com/premium/q=&content_types[]=Book&utm_s)

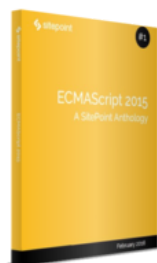


Jump Start Sass

Hugo Giraudel



https://www.sitepoint.com/premium/boc-start-sass/?utm_source=sitepoint&utm_medium=relat



ECMAScript 2015: A SitePoint Anthology

James Hibbard



https://www.sitepoint.com/premium/book/2015-a-sitepoint-anthology/?utm_source=sitepoint&utm_medium=rela



Scrum: Novice to Ninja

M. David Green



https://www.sitepoint.com/premium/book/novice-to-ninja/?utm_source=sitepoint&utm_medium=rela

SCREENCASTS >

[https://www.sitepoint.com/premium/tutorials/?q=&content_types\[\]=ScreenCast&](https://www.sitepoint.com/premium/tutorials/?q=&content_types[]=ScreenCast&)

Finding Bugs in Your Commits with Git Bisect

Shaumik Daityari

https://www.sitepoint.com/premium/tutorial/bugs-in-your-commits-with-git-bisect/?utm_source=sitepoint&utm_medium=rela

Exploring and Working with Photos in Canva

Lisa Larson-Kelley

https://www.sitepoint.com/premium/tutorial/and-working-with-photos-in-canva/?utm_source=sitepoint&utm_medium=rela

Creating Custom Sass Functions

Guilherme Muller

https://www.sitepoint.com/premium/tutorial/custom-sass-functions/?utm_source=sitepoint&utm_medium=relat

About

[Our Story \(/about-us/\)](/about-us/)

[Advertise \(/advertising/\)](/advertising/)

[Press Room \(/press/\)](/press/)

[Reference \(http://reference.sitepoint.com/css/\)](http://reference.sitepoint.com/css/)

[Terms of Use \(/legals/\)](/legals/)

[Privacy Policy \(/legals/#privacy\)](/legals/#privacy)

[FAQ \(https://sitepoint.zendesk.com/hc/en-us\)](https://sitepoint.zendesk.com/hc/en-us)

[Contact Us \(mailto:feedback@sitepoint.com\)](mailto:feedback@sitepoint.com)

[Contribute \(/write-for-us/\)](/write-for-us/)

Visit

[SitePoint Home \(/\)](/)

[Forums \(https://www.sitepoint.com/community/\)](https://www.sitepoint.com/community/)

[Newsletters \(/newsletter/\)](/newsletter/)

[Premium \(/premium/\)](/premium/)

[References \(/sass-reference/\)](/sass-reference/)

[Shop \(https://shop.sitepoint.com\)](https://shop.sitepoint.com)

[Versioning \(https://www.sitepoint.com/versioning/\)](https://www.sitepoint.com/versioning/)

Connect

 [\(http://www.sitepoint.com/feed/\)](http://www.sitepoint.com/feed/) 

[\(/newsletter/\)](/newsletter/) 

[\(https://www.facebook.com/sitepoint\)](https://www.facebook.com/sitepoint) 

[\(http://twitter.com/sitepointdotcom\)](http://twitter.com/sitepointdotcom) 

[\(https://plus.google.com/+sitepoint\)](https://plus.google.com/+sitepoint)

