

BY BOBBY ILIEV

# Introduction to Bash Scripting

FOR DEVELOPERS



# Table of Contents

|                                   |               |
|-----------------------------------|---------------|
| <b>About the book</b>             | <b>4</b>      |
| About the author                  | 5             |
| Sponsors                          | 6             |
| Ebook PDF Generation Tool         | 8             |
| Book Cover                        | 9             |
| License                           | 10            |
| <br><b>Introduction to Git</b>    | <br><b>11</b> |
| <br><b>Version Control</b>        | <br><b>13</b> |
| <br><b>Installing Git</b>         | <br><b>14</b> |
| <br><b>Basic Shell Commands</b>   | <br><b>17</b> |
| <br><b>Git Configuration</b>      | <br><b>20</b> |
| <br><b>Introduction to GitHub</b> | <br><b>21</b> |
| <br><b>Git And VS Code</b>        | <br><b>26</b> |
| <br><b>Git Commit</b>             | <br><b>27</b> |
| <br><b>Git Add</b>                | <br><b>28</b> |
| <br><b>Committing</b>             | <br><b>29</b> |

|                                 |           |
|---------------------------------|-----------|
| <b>Git Push</b>                 | <b>30</b> |
| <b>SSH Keys</b>                 | <b>31</b> |
| <b>Git Push</b>                 | <b>32</b> |
| <b>Git Pull</b>                 | <b>33</b> |
| <b>Git Workflow</b>             | <b>34</b> |
| <b>Git Branches</b>             | <b>35</b> |
| <b>Reverting changes</b>        | <b>36</b> |
| <b>Forking in Git</b>           | <b>37</b> |
| <b>Git And VS Code</b>          | <b>38</b> |
| Installing VS Code              | 39        |
| Cloning a repository in VS Code | 41        |
| Create a branch                 | 42        |
| Setup a commit message template | 43        |
| Conclusion                      | 44        |
| Additional sources:             | 45        |
| <b>Conclusion</b>               | <b>46</b> |

# About the book

- **This version was published on March 12 2021**

This is an open-source introduction to Git and GitHub guide that will help you learn the basics of version control and start using Git for your SysOps, DevOps, and Dev projects. No matter if you are a DevOps/SysOps engineer, developer, or just a Linux enthusiast, you can use Git to track your code changes and collaborate with other members of your team or open source maintainers.

The guide is suitable for anyone working as a developer, system administrator, or a DevOps engineer and wants to learn the basics of Bash scripting.

The first 13 chapters would be purely focused on getting some solid Bash scripting foundations, then the rest of the chapters would give you some real-life examples and scripts.

## About the author

My name is Bobby Iliev, and I have been working as a Linux DevOps Engineer since 2014. I am an avid Linux lover and supporter of the open-source movement philosophy. I am always doing that which I cannot do in order that I may learn how to do it, and I believe in sharing knowledge.

I think it's essential always to keep professional and surround yourself with good people, work hard, and be nice to everyone. You have to perform at a consistently higher level than others. That's the mark of a true professional.

For more information, please visit my blog at <https://bobbyiliev.com>, follow me on Twitter [@bobbyiliev\\_](#) and [YouTube](#).

## Sponsors

This book is made possible thanks to these fantastic companies!

### DigitalOcean

DigitalOcean is a cloud services platform delivering the simplicity developers love and businesses trust to run production applications at scale.

It provides highly available, secure, and scalable compute, storage, and networking solutions that help developers build great software faster.

Founded in 2012 with offices in New York and Cambridge, MA, DigitalOcean offers transparent and affordable pricing, an elegant user interface, and one of the largest libraries of open source resources available.

For more information, please visit <https://www.digitalocean.com> or follow [@digitalocean](#) on Twitter.

If you are new to DigitalOcean, you can get a free \$100 credit and spin up your own servers via this referral link here:

[Free \\$100 Credit For DigitalOcean](#)

### DevDojo

The DevDojo is a resource to learn all things web development and web design. Learn on your lunch break or wake up and enjoy a cup of coffee with us to learn something new.

Join this developer community, and we can all learn together, build together, and grow together.

[Join DevDojo](#)

For more information, please visit <https://www.devdojo.com> or follow [@thedeveloper](#) on Twitter.

# Ebook PDF Generation Tool

This ebook was generated by [Ibis](#) developed by [Mohamed Said](#).

Ibis is a PHP tool that helps you write eBooks in markdown.



## **Book Cover**

The cover for this ebook was created with [Canva.com](https://www.canva.com).

If you ever need to create a graphic, poster, invitation, logo, presentation – or anything that looks good — give Canva a go.

# License

## MIT License

Copyright (c) 2020 Bobby Iliev

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

# Introduction to Git

Welcome to this Git and GitHub basics training guide! In this **Git crash course**, you will learn the **basics of Git** so you can use Git to track your code changes and collaborate with other members of your team or open source maintainers.

Whether you are a newcomer to programming, or an experienced one, you have to know how to use Git. Most of the projects that a small or big group of developers work on are done through GitHub or GitLab.

It makes working with other developers so much more exciting and enjoyable. Just by creating a new branch, adding all your brilliant ideas to the code that can help the project, committing it, and then pushing it to GitHub or GitLab. Then after the PR(pull request) has been opened, reviewed, and then merged, you can get back to your code and continue adding more awesome stuff. After pulling the changes from the main/master branch, of course.

If what you just read doesn't make any sense to you, don't worry. Everything will be explained in this eBook!

This eBook will show you the basics on how to start using Git and try to help you get more comfortable with it.

It does look a bit scary in the beginning, but don't worry. It's not as frightening as it seems, and hopefully, after reading this eBook, you can get a bit more comfortable with Git.

Learning Git is essential for every programmer. Even some of the biggest companies use GitHub for their projects. Remember that the more you use it, the more you're going to get used to it.

Git is without a doubt the most popular open source version control system for tracking changes in source code out there.

The original author of git is Linus Torvalds who is also the creator of **Linux**.

Git is designed to help programmers coordinating work among each other. Its goals include speed, data integrity, and support for distributed workflows.

# Version Control

*Version control*, also called *Source control*, allows you to track and manage all of the changes to your code.

The main benefit of version control is that multiple people could work on the same project at the same time. With version control tools like Git, you can track all of the changes to your code and in case of any problems you could easily revert back to a working state of your source code.



With distributed version control systems like Git, you would have your source code stored on a remote repository like GitHub and also a local repository stored on your computer.

You will learn more about remote and local repositories in the next few chapters, but one of the main points for the moment is that your source code would be stored on a remote repository, so in case that something goes wrong with your laptop you would not lose all of your changes but they will be safely stored on GitHub.

# Installing Git

In order for you to be able to use Git on your local machine, you would need to install it.

Depending on the operating system that you are using, you can follow the steps here.

## Install Git on Linux

With most Linux distributions the Git command line tool comes installed out of the box.

If this is not the case for you, you can install Git with the following command:

- On RHEL Linux:

```
sudo dnf install git-all
```

- On Debian based distributions including Ubuntu:

```
sudo apt install git-all
```

## Install Git on Mac

If you are using Mac, Git should be available out of the box as well. However if this is not the case there are 2 main ways of installing Git on your Mac:

- Using Homebrew: in case that you are using Homebrew you can open your terminal and run the following:

```
brew install git
```

- Git installer: Alternatively, you could use the following installer:

[git-osx-installer](#)

I would personally stick to Homebrew.

## **Install Git on Windows**

If you have a Windows PC, you can follow the steps on how to install Git on Windows here:

[Install Git on Windows](#)


During the installation, make sure to choose the Git Bash option as this would provide you with a Git Bash terminal which you will use while following along.

## **Check Git version**

Once you have installed Git, in order to check the version of Git that you have installed on your machine, you could use the following command:

```
git --version
```

Example output:



```
git version 2.25.1
```

In my case I have Git 2.25.1 installed on my laptop.



# Basic Shell Commands

As throughout this eBook we will be using mainly Git via the command line, it is important to know basic shell commands so that you could find your way around the terminal.

So before we get started, let's go over a few basic shell commands!

## The `ls` command

The `ls` command allows you to list the contents of a folder/directory. All that you need to do in order to run the command is to open a terminal and run the following:

```
ls
```

The output will show you all of the files and folders that are located in your current directory. In my case the output is the following:

```
CONTRIBUTING.md ebook README.md
```

For more information about the `ls` command make sure to check out this page [here](#).

Note: This will work on a Linux/UNIX based systems. If you are on Windows and if you are using the built-in CMD, you would have to use the `dir` command

## The `cd` command

The `cd` command stands for **Change Directory** and allows you to navigate through the filesystem of your computer or server. Let's say that I wanted to go inside the `ebook` directory from the output above, what I would need to do is to run the `cd` command followed by the directory that I want to access:

```
cd ebook
```

If I wanted to go back one level up, I would use the `cd ..` command.

## The `pwd` command

The `pwd` command stands for **Print Working Directory** which essentially means that when you run the command, it will show you the current directory that you are in.

Let's take the example from above, if I run the `pwd` command I would get the full path to the folder that I'm currently in:

```
pwd
```

Output:

```
/home/bobby/introduction-to-git
```

Then I could use the `cd` command and access the `ebook` directory:

```
cd ebook
```

And finally if I was to run the `pwd` command again, I would see the

following output:

```
/home/bobby/introduction-to-git/ebook
```

Essentially what happened was that thanks to the `pwd` command, I was able to see that I'm at the `/home/bobby/introduction-to-git` directory and then after accessing the `ebook` directory, again by using `pwd` I was able to see that my new current directory is `/home/bobby/introduction-to-git/ebook`.

## The `rm` command

The `rm` command stands for `remove` and allows you to delete files and folders. Let's say that I wanted to delete the `README.md` file, what I would have to do is run the following command:

```
rm README.md
```

In case that I had to delete a folder/directory, I would need to specify the `-r` flag:

```
rm -r ebook
```

Note: keep in mind that the `rm` command would completely delete the files and folders and the action is irreversible, meaning that you can't get them back.

One thing that you need to keep in mind is that all shell commands are case sensitive, so if you type `LS` it would not work.

With that, now you know some basic shell commands which will be beneficial for your day-to-day activities.

# Git Configuration

The first time you setup Git on your machine, you would need to do some initial configuration.

There are a few main things that you would need to configure:

- Your details: like your name and email address
- Your Git Editor
- The default branch name: we will learn more about branches later on

We can change all of those things by using the `git config` command.

Let's get started with the initial configuration!

## The `git config` command

# Introduction to GitHub

Before we deep dive into all of the various Git commands, let's quickly get familiar with GitHub.

Git is essentially the tool that you use to track your code changes, and GitHub on the otherside is a website where you can push your local projects to.

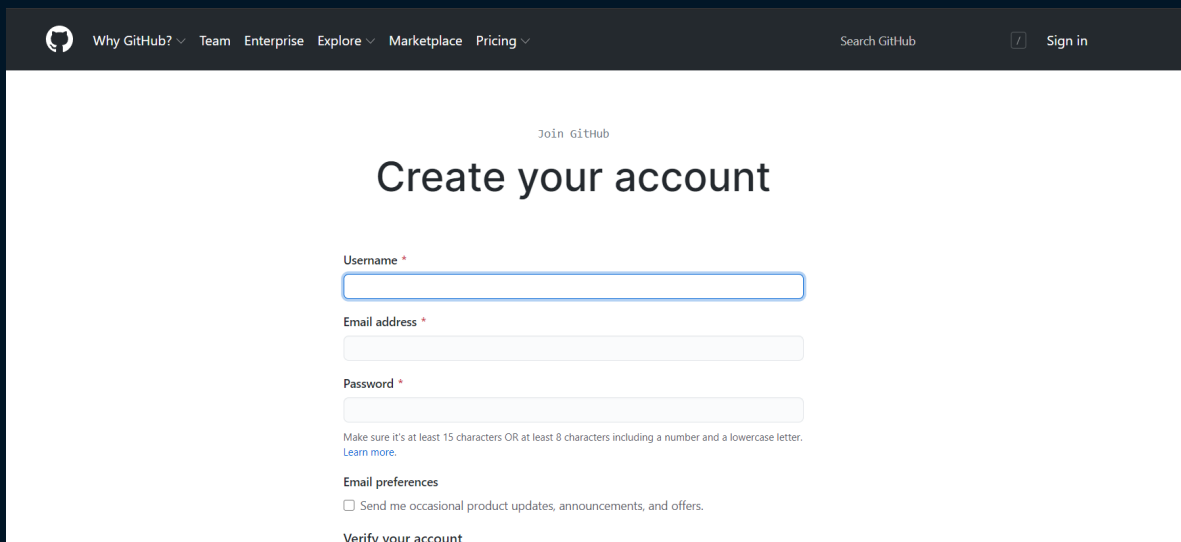
This is essentially needed as it would act as a central hub where you would store your project at and all of your team mates or other people working on the same project as you, would push their changes to.

## GitHub Registration

Before you get started you would need to create an account with GitHub, you can do so via this link here:

- [Join GitHub](#)

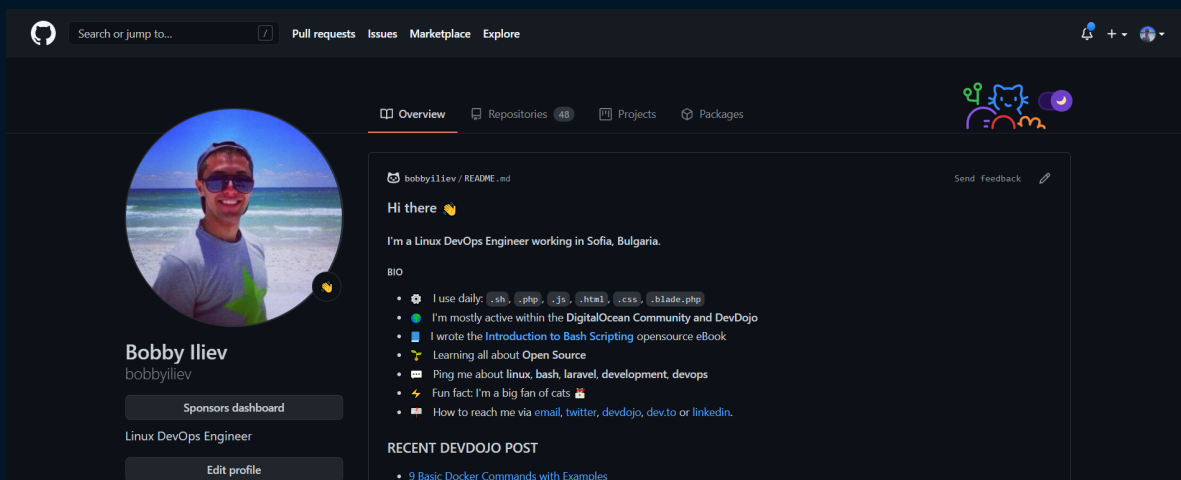
You would get to the following page where you would need to add your new account details:



The screenshot shows the GitHub account creation page. At the top, there's a navigation bar with links like 'Why GitHub?', 'Team', 'Enterprise', 'Explore', 'Marketplace', and 'Pricing'. The main heading is 'Create your account'. Below it, there are three input fields: 'Username', 'Email address', and 'Password'. A note specifies password requirements: 'Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter.' There's also an 'Email preferences' section with a checkbox for 'Send me occasional product updates, announcements, and offers.' and a 'Verify your account' link at the bottom.

## GitHub Profile

Once you've registered, you can go to [https://github.com/YOUR\\_USER\\_NAME](https://github.com/YOUR_USER_NAME), and you would be able to see your public profile where you could add some information about yourself. Here is an example profile which you could check: [GitHub Profile](#)



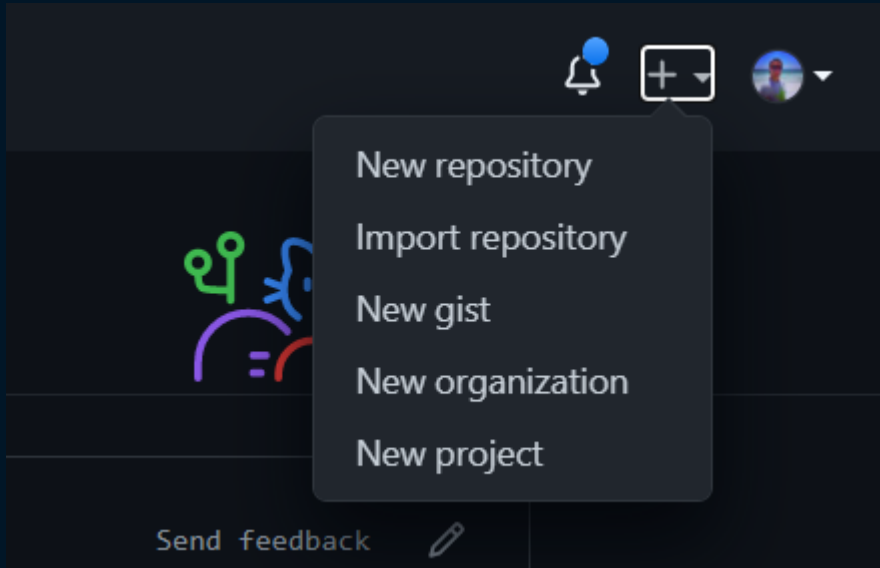
The screenshot shows a GitHub profile for 'Bobby Iliev' (bobbyiliev). The profile includes a circular profile picture of a man with sunglasses. Below the picture, it says 'Bobby Iliev' and 'bobbyiliev'. There are buttons for 'Sponsors dashboard', 'Linux DevOps Engineer', and 'Edit profile'. The 'Overview' tab is selected, showing a bio: 'Hi there 🍌 I'm a Linux DevOps Engineer working in Sofia, Bulgaria.' Below the bio is a 'BIO' section with a list of interests and achievements, such as 'I use daily: .sh, .php, .js, .html, .css, .blade.php' and 'I wrote the Introduction to Bash Scripting opensource eBook'. There's also a 'RECENT DEVDOJO POST' section with a link to '9 Basic Docker Commands with Examples'.

## Creating a new repository

If you are not familiar with the word repository, you could think of it as a project, it would hold all of the files of your application or website that you are build. In many cases people would use repo instead of

repository for short.

To create a new repository on GitHub, you have to click on the + sign on the top right and then click on the **New Repository** button:

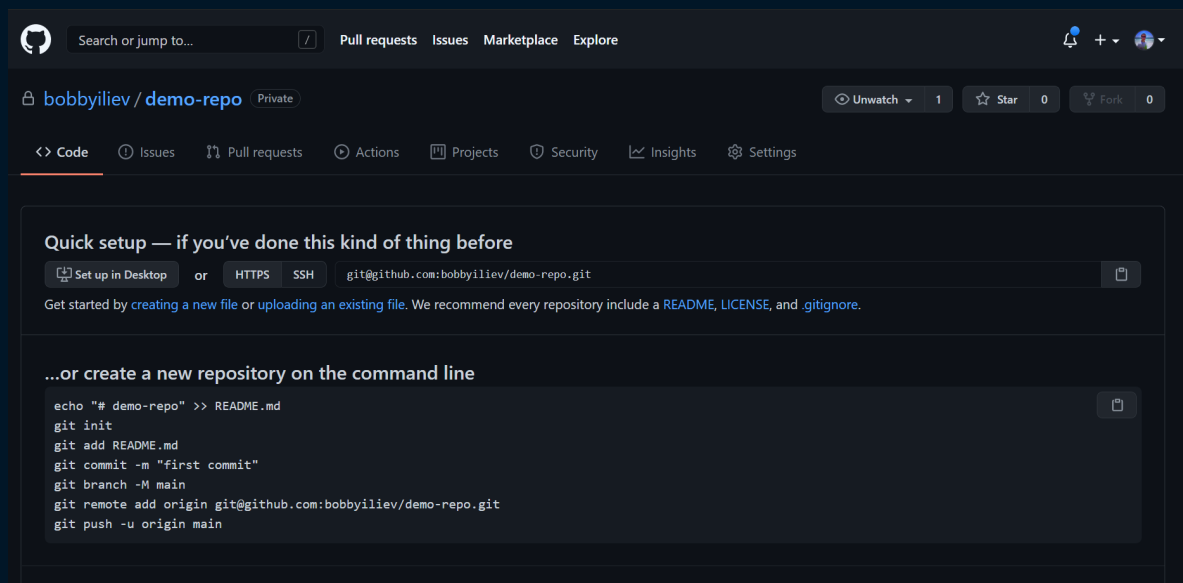


After that you would get to a page where you can specify the information for your new repository like:

- The name of the project: Here make sure to use something descriptive
- Some general description about the project and what it is about
- Choose whether you want the repository to be Public or Private

A screenshot of the 'Create a new repository' page on GitHub. The page has a dark theme. At the top, there is a navigation bar with the GitHub logo, a search bar, and links for 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. The main heading is 'Create a new repository'. Below it, there is a subheading 'Repository template' and a button 'No template'. The 'Owner' field is set to 'bobbyiliev'. The 'Repository name' field is empty. Below the name field, there is a hint: 'Great repository names are short and memorable. Need inspiration? How about miniature-funicular?'. The 'Description (optional)' field is empty. At the bottom, there are two radio buttons: 'Public' (selected) and 'Private'. The 'Public' option has a description: 'Anyone on the internet can see this repository. You choose who can commit.' The 'Private' option has a description: 'You choose who can see and commit to this repository.'

Once you've added the necessary information and hit the create button, you will get to a page with some instructions on how to push your local project to GitHub:



We will go over those steps more in depth in the next few chapters.

## Public vs Private repositories

Depending on the project and whether or not it is open source, you could set your repository to be **public** or **private**.

The main difference is that with a public repository, anyone on the internet can see this repository. But even though that everyone would be able to see the repository and read the code, you would be the maintainer of the project and you will choose who can commit.

With a private repository, it would only be available for you and the people that you've invited.

Public repositories are used for open source projects.



## The README.md file

The `README.md` file is an essential part of each project. The `.md` extension stands for Markdown.

You can think of the `README.md` file as the introduction to your repository. It's very helpful that while looking at someone's repo you can just scroll down to their README file and have a look at what their project is all about.

And it is very important that your project is properly introduced. Because if the project itself isn't introduced properly, no one is going to spend their time in helping to improve it and try to further develop it.

That's why having a good README file shouldn't be overlooked and you should spend a considerable amount of your time on it.

In this post, I am going to share some tips with you about how you can improve your README file, and hopefully, it can help you with your repos.

For more information, make sure to check out this post on [how to write a good README.md file](#).

# Git And VS Code

# Git Commit

# Git Add

# Committing

# Git Push

# SSH Keys

# Git Push



# Git Pull

# Git Workflow

# Git Branches

# Reverting changes

# Forking in Git

# Git And VS Code

As much as I love to use the terminal in order to do my daily tasks in the end I would rather do multiple tasks within one window (GUI) or perform everything from the terminal itself.

In the past, I was using the text editors (vim, nano and etc) in my terminal to edit the code in my repositories and then go along with the git client to commit my changes, but then I switched to Visual Studio Code to manage and develop my code.

I will recommend you to check this article on why you should use Visual Studio. It is an article from Visual Studio's website itself.

## [Why you should use Visual Studio](#)

Visual Studio Code has integrated source control management (SCM) and includes Git support in-the-box. Many other source control providers are available through extensions on the VS Code Marketplace. It also has support for handling multiple Source Control providers simultaneously so you can open all of your projects at the same time and make changes whenever this is needed. I personally find this really handy.

# Installing VS Code

You need to install Visual Studio Code. It runs on the macOS, Linux, and Windows operating systems.

Follow the platform-specific guides below:

- [macOS](#)
- [Linux](#)
- [Windows](#)

You need to install Git first before you get these features. Make sure you install at least version 2.0.0. If you do not have git installed on your machine feel free to check this really useful article on [How to get started with Git](#)

You need to set your username and email in the Git configuration or git will fail back to using information from your local machine when you commit. We need to provide this information because Git embeds this information into each commit we do.

In order to set this you can execute the following commands:

```
git config --global user.name "John Doe"
git config --global user.email "johnde@domain.com"
```

The information will be saved in your `~/.gitconfig` file

```
[user]
  name = John Doe
  email = johndoe@domain.com
```

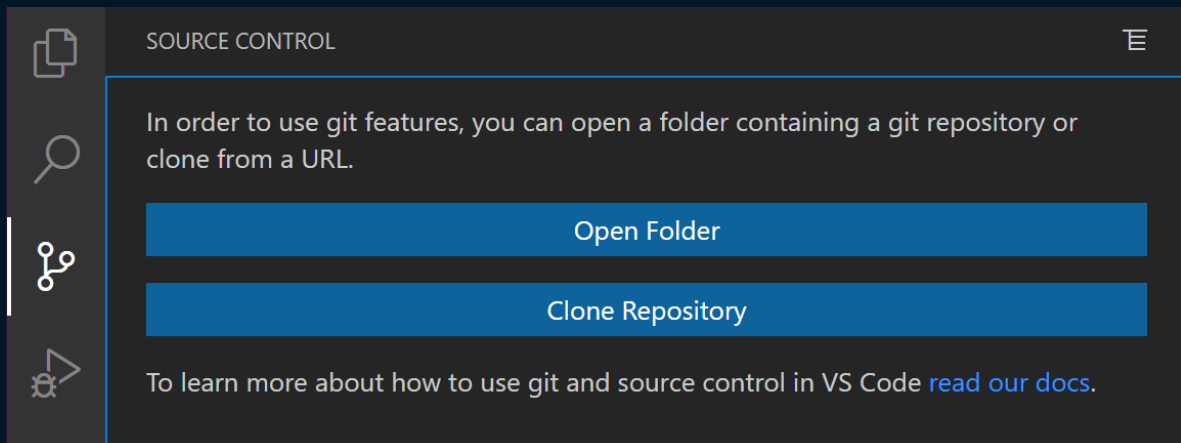
With Git installed and set up on your local machine, you are now ready

to use Git for version control with Visual Studio or using the terminal.



## Cloning a repository in VS Code

The good thing is that Visual Studio will auto-detect if you've opened a folder that is actually a repository. If you've already opened a repository it will be visible in the Source Control View.



If you haven't opened a folder yet, the Source Control view will give you the options to Open Folder from your local machine or Clone Repository.

If you select Clone Repository, you will be asked for the URL of the remote repository (for example on GitHub) and the parent directory under which to put the local repository.

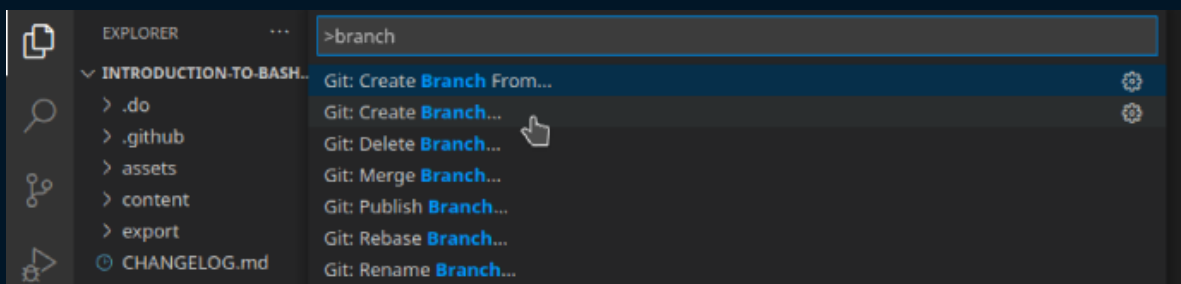
For a GitHub repository, you would find the URL from the GitHub Code dialog.

## Create a branch

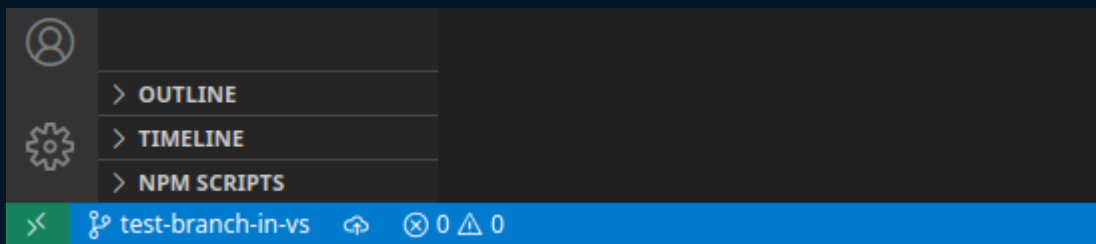
In order to create a branch open the command pallet:

- Windows: Ctrl + Shift + P
- Linux: Ctrl + Shift + P
- MacOS: Shift + CMD + P

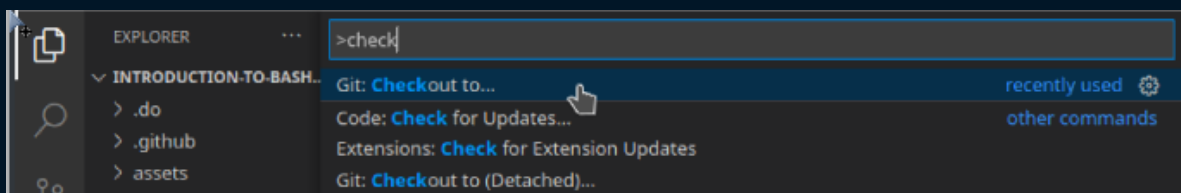
And select **Git Create Branch...**



Then you just need to enter a name for the branch. Keep in mind that in the bottom left corner you can see in which branch you are. The default one will be the main and if you successfully create the branch you should see the name of the newly created branch



If you want to switch branches you can open the command pallet and search for **Git checkout to** and then select the main branch or switch to a different branch.



## Setup a commit message template

If you want to speed up the process and have a predefined template for your commit messages you can create a simple file that will contain this information.

In order to do that open your terminal if you're on Linux or macOS and create the following file: `.gitmessage` in your home directory. In order to create the file, you can open it in your favourite text editor and then simply put the default content you would like and then just save and exit the file. Example content is:

```
cat ~/.gitmessage
```

```
#Title
```

```
#Summary of the commit
```

```
#Include Co-authored-by for all contributors.
```

To tell Git to use it as the default message that appears in your editor when you run `git commit` and set the `commit.template` configuration value:

```
$ git config --global commit.template ~/.gitmessage
$ git commit
```

## Conclusion

If you prefer to code in Visual Studio Code and you also use version control I will definitely recommend you to give it a go and interact with the repositories in VS code. I believe that everyone has their own style and they might do things differently depending from their mood as well. As long as you can add/modify your code and then commit your changes to the repository there is no exactly correct/wrong way to achieve this. For example you can edit your code in vim and push the changes using the git client in your terminal or do the coding in Visual Studio and then commit the changes using the terminal as well. You're free to do it the way you want it and the way you find it more convenient as well. I believe that using git within VS code can make your workflow more efficient and robust.

## Additional sources:

- [Version Control](#) - Read more about integrated Git support.
- [Setup Overview](#) - Set up and start using VS Code.
- [GitHub with Visual Studio](#) - Read more about the GitHub support in VS code
- You can also check this mini video tutorial on how to use the basics of Git version control in Visual Studio Code

Contribured by: [Alex Georgiev](#) Initially posted [here](#).

# Conclusion

Congratulations! You have just completed the Git basics guide!

If you found this useful, be sure to star the project on [GitHub](#)!

If you have any suggestions for improvements, make sure to contribute pull requests or open issues.

In this introduction to Git and GitHub eBook, we just covered the basics, but you still have enough under your belt to start using Git and start contributing to some awesome open source projects!

As a next step try to create a GitHub project, clone it locally and push a project that you've been working on to GitHub!

In case that this eBook inspired you to contribute to some amazing open source project, make sure to tweet about it and tag [@bobbyiliev](#) so that we could check it out!

Congrats again on completing this eBook!