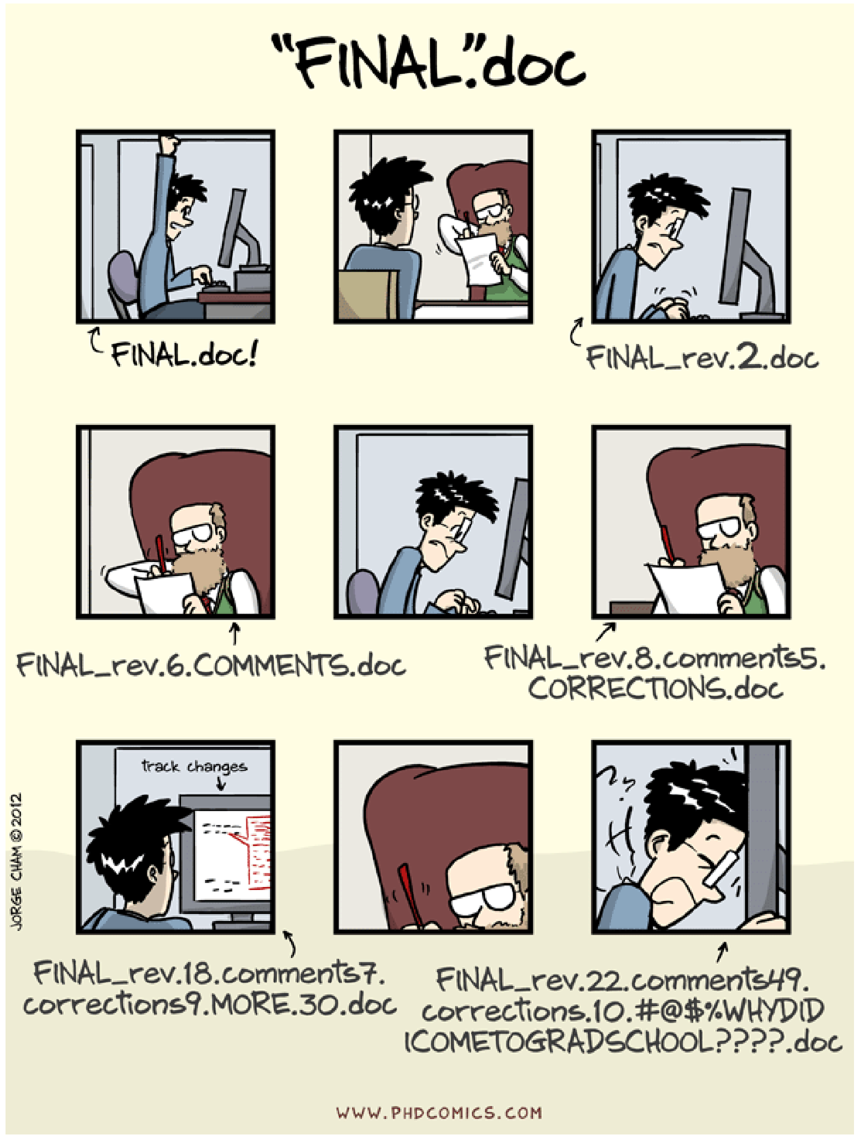
Version control with Git(Hub)

Blogdown + Hugo + Netlify

Git + GitHub



Git

- Git is a distributed version control system
- Think of it as Word's *Track Changes* + Dropbox
- Whenever you want to save changes, you ***commit*** your changes and git takes a snapshot of that version
- Unlike Word, Git keeps a full history of all versions you ***commited***
- There is a (steep) learning curve, but it's worth it.

GitHub

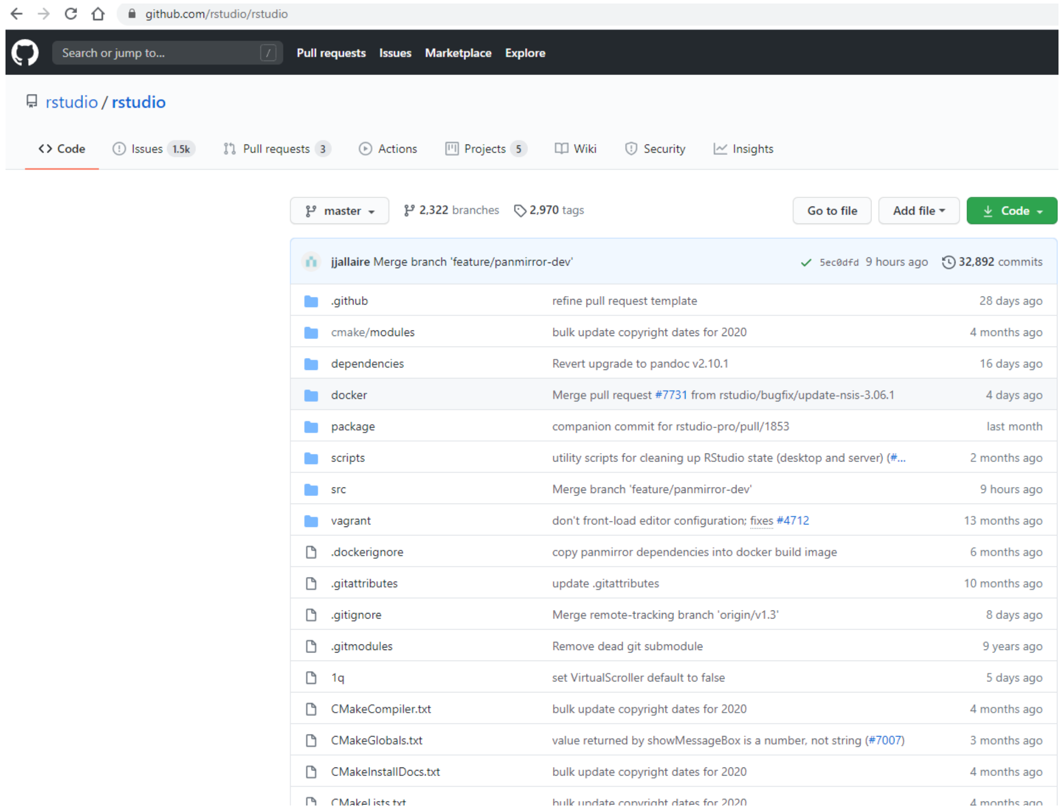
- GitHub is an online hosting platform that provides services built on top of the Git system. Similar platforms include ***Bitbucket*** and ***GitLab***, but most people use Github
- Just like we don't *need* Rstudio to run R code, we don't *need* GitHub to use Git... But will make our lives so much easier

Collaboration tool

A brilliant platform for collaboration

There's a high probability that your favourite app, program or package is built using Git-based tools. (e.g., RStudio)

Git(Hub) helps to operationalise the ideals of open science and reproducibility.



github.com/rstudio/rstudio

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

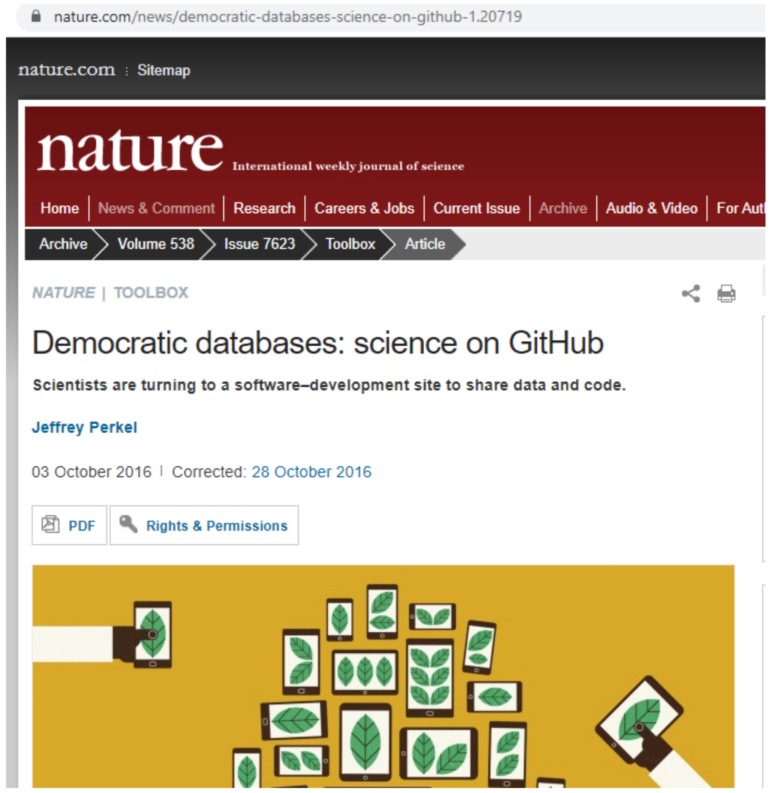
rstudio / rstudio

Code | Issues 1.5k | Pull requests 3 | Actions | Projects 5 | Wiki | Security | Insights

master | 2,322 branches | 2,970 tags | Go to file | Add file | Code

jjallaire Merge branch 'feature/panmirror-dev' ✓ 5ec0dfd 9 hours ago 32,892 commits

.github	refine pull request template	28 days ago
cmake/modules	bulk update copyright dates for 2020	4 months ago
dependencies	Revert upgrade to pandoc v2.10.1	16 days ago
docker	Merge pull request #7731 from rstudio/bugfix/update-nsis-3.06.1	4 days ago
package	companion commit for rstudio-pro/pull/1853	last month
scripts	utility scripts for cleaning up RStudio state (desktop and server) (#...	2 months ago
src	Merge branch 'feature/panmirror-dev'	9 hours ago
vagrant	don't front-load editor configuration; fixes #4712	13 months ago
.dockerignore	copy panmirror dependencies into docker build image	6 months ago
.gitattributes	update .gitattributes	10 months ago
.gitignore	Merge remote-tracking branch 'origin/v1.3'	8 days ago
.gitmodules	Remove dead git submodule	9 years ago
1q	set VirtualScroller default to false	5 days ago
CMakeCompiler.txt	bulk update copyright dates for 2020	4 months ago
CMakeGlobals.txt	value returned by showMessageBox is a number, not string (#7007)	3 months ago
CMakeInstallDocs.txt	bulk update copyright dates for 2020	4 months ago
CMakeLists.txt	bulk update copyright dates for 2020	4 months ago



nature.com/news/democratic-databases-science-on-github-1.20719

nature.com : Sitemap

nature International weekly journal of science

Home | News & Comment | Research | Careers & Jobs | Current Issue | Archive | Audio & Video | For Aut

Archive > Volume 538 > Issue 7623 > Toolbox > Article

NATURE | TOOLBOX

Democratic databases: science on GitHub

Scientists are turning to a software–development site to share data and code.

Jeffrey Perkel

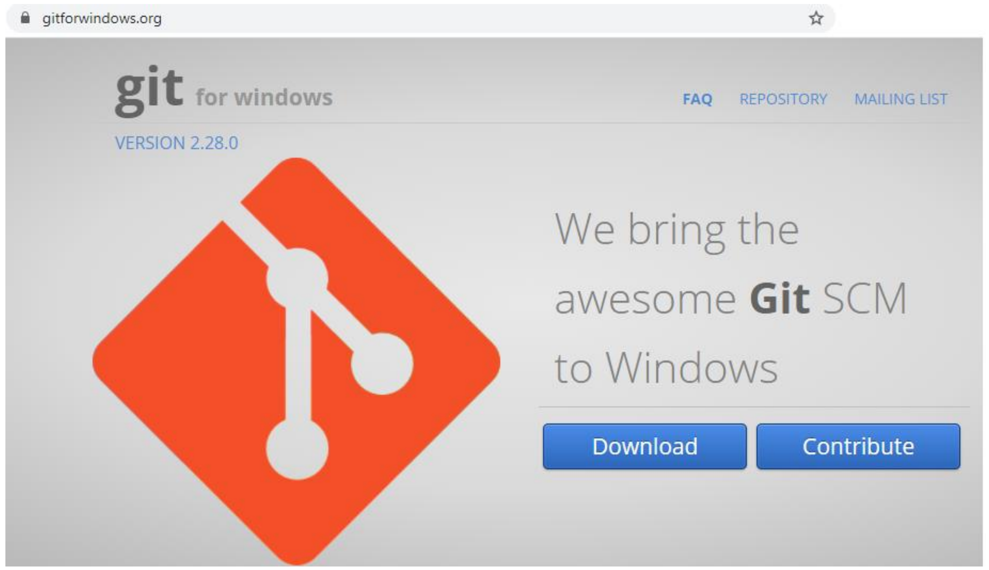
03 October 2016 | Corrected: 28 October 2016

PDF | Rights & Permissions

https://github.com/rstudio/rstudio
https://www.nature.com/news/democratic-databases-science-on-github-1.20719

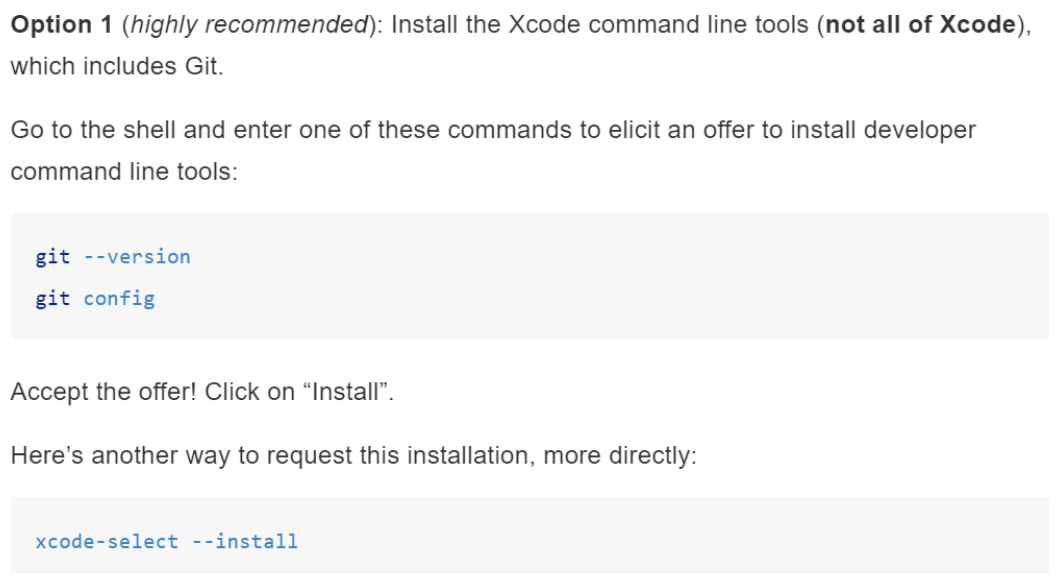
Local Git installation

Windows



https://gitforwindows.org/

macOS



Option 1 (*highly recommended*): Install the Xcode command line tools (**not all of Xcode**), which includes Git.

Go to the shell and enter one of these commands to elicit an offer to install developer command line tools:

```
git --version
git config
```

Accept the offer! Click on “Install”.

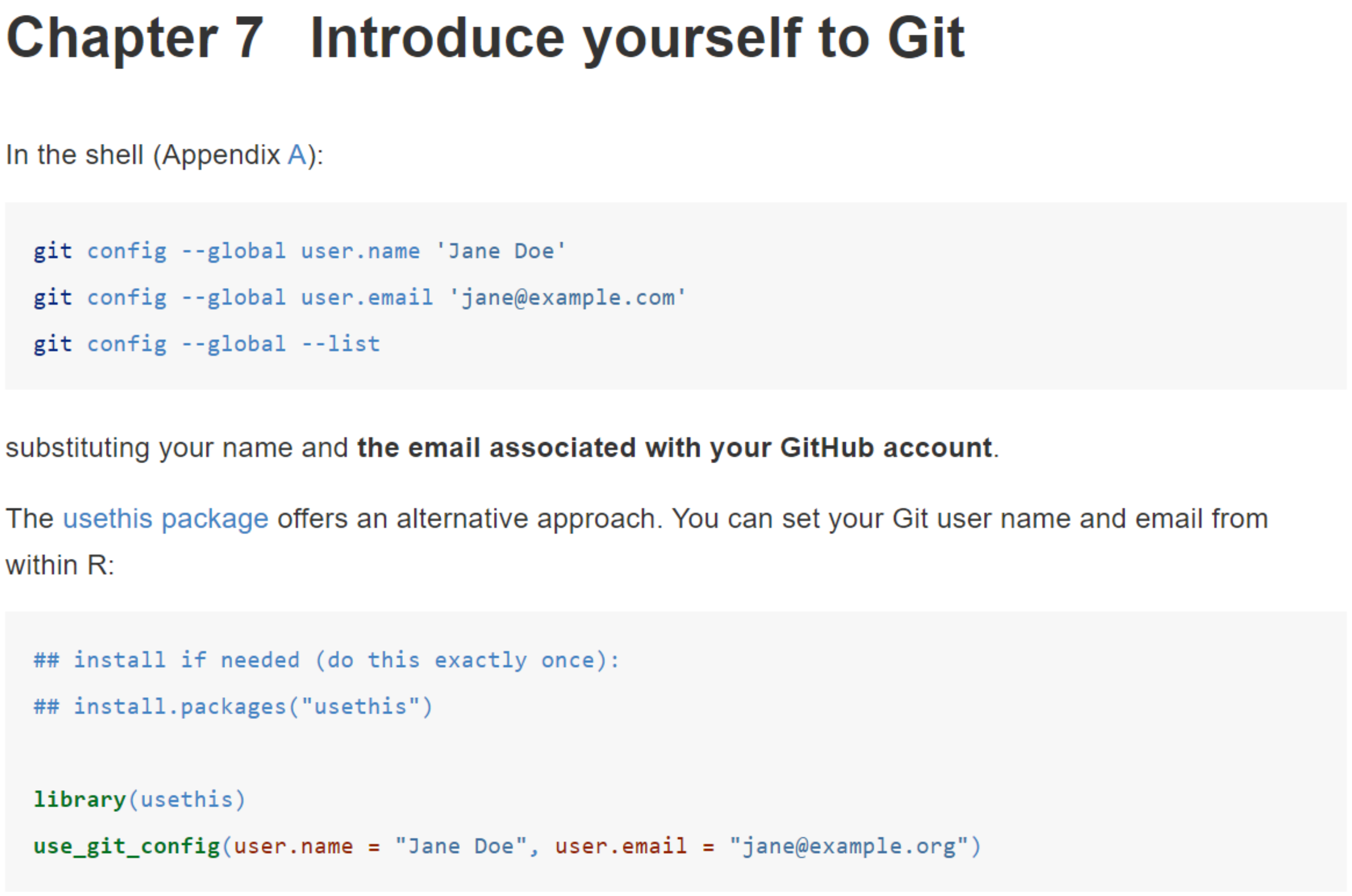
Here’s another way to request this installation, more directly:

```
xcode-select --install
```

Install the Xcode command line tools

Source: https://happygitwithr.com/install-git.html#install-git

Git + Rstudio configuration



Chapter 7 Introduce yourself to Git

In the shell (Appendix A):

```
git config --global user.name 'Jane Doe'
git config --global user.email 'jane@example.com'
git config --global --list
```

substituting your name and **the email associated with your GitHub account**.

The usethis package offers an alternative approach. You can set your Git user name and email from within R:

```
## install if needed (do this exactly once):
## install.packages("usethis")

library(usethis)
use_git_config(user.name = "Jane Doe", user.email = "jane@example.org")
```

Source: https://happygitwithr.com/hello-git.html

Git + RStudio

1. Create a new public repo, called ***test***, on GitHub and initialise with a README.
2. Copy the HTTPS/SSH link (the green **Code** button).
3. Open up RStudio.
4. Navigate to *File -> New Project -> Version Control -> Git*
5. Paste your copied link into the "Repository URL:" box.
6. Choose the project path ("Create project as subdirectory of:") and click Create Project.
7. Look at the top-right panel in your RStudio IDE. Do you see the ***Git*** tab?
8. There should already be some files in there, which we'll ignore for the moment
9. Go to ***Files*** tab in the bottom-right panel and click on the README file.
10. Add some text like "Hello World!" and save (Ctrl/Cmd + S) the README.
11. Do you see any changes in the "Git" panel?

Main Git operations

The four main Git operations, in this order, are:

1. Stage (or "add")

- Tell Git that you want to add changes to the repo history (file edits, additions, deletions, etc.)

2. Commit

- Tell Git that, yes, you are sure these changes should be part of the repo history. This is when you take a snapshot of your work as it is now. Git won't allow you to commit if you don't add a helpful, explanatory message

3.Pull

- Get any new changes made on the GitHub repo (i.e. the upstream remote), either by you on another machine or your collaborators.

4.Push

- Push any (commited) local changes to the GitHub repo

Always ***pull first*** from the upstream repo ***before you push*** any changes, even when working alone. It's a good habit that will save you troubles later on

Git workflow: Distributed Version Control

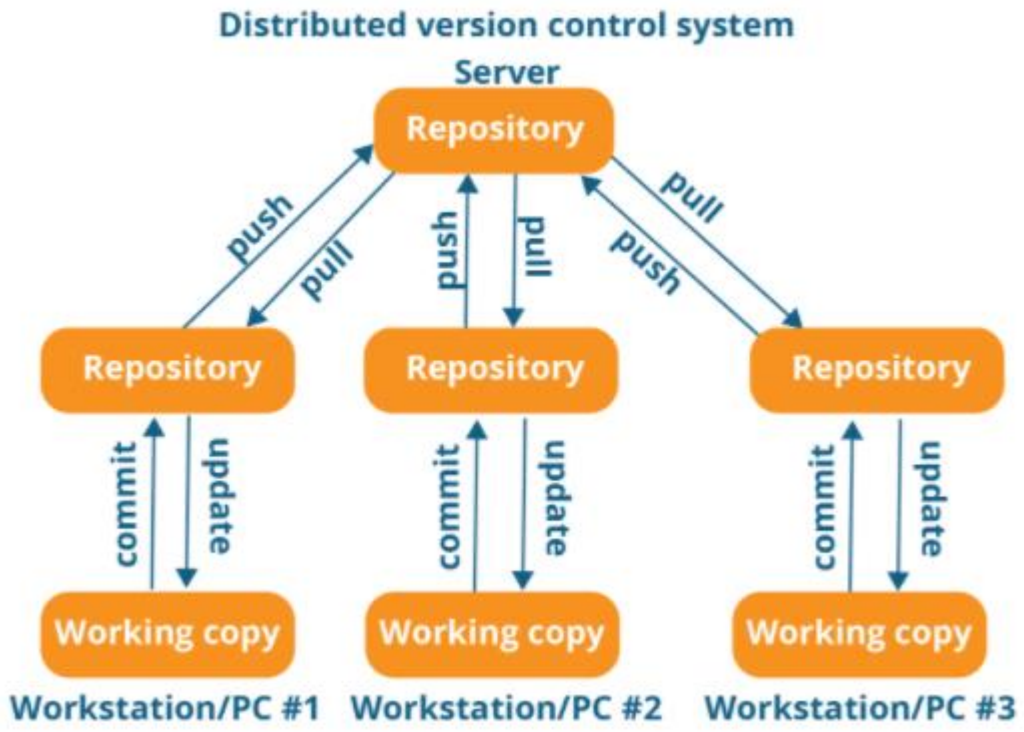
Creating the repo on GitHub first means that it will always be upstream of your (and other) working copies.

In a distributed version control, you clone a copy of the repository locally so that you are not working on the original file. You work on the cloned copy of the project, and once all the changes are finalised, you can commit and push changes to that original file.

GitHub acts as the central node in a distributed network, something especially valuable when you are collaborating on a project with others

If you would like to move an existing project to GitHub, create an empty repo there first, clone it locally, and then copy all your files across.

Github + RStudio Projects is a natural and great way to roganise your work. They also solve absolute vs. relative path problems, since the ***.Rproj*** file acts as an anchor point for all other files in the repo and you never have to manually set your working directory.



Git from the shell/terminal

Besides using the Rstudio Git panel and its GUI, you can also give Git commands from Rstudio shell/terminal.

1. Add ("stage") a file or group of files:	**git add NAME-OF-FILE-OR-FOLDER**

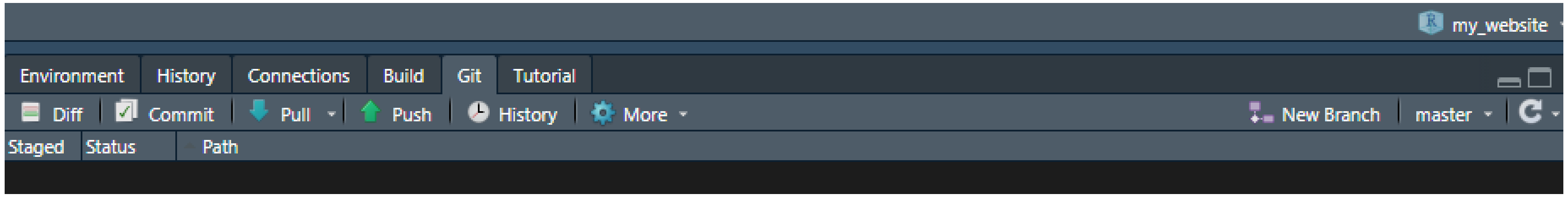
You can use wildcard characters to stage a group of files (e.g. sharing a common prefix).
There are a bunch of useful flag options too:

Stage all files	**git add -A**
Stage updated files only (modified or deleted, but not new)	**git add -u**
Stage new files only (not updated)	**git add .**
2. Commit your changes	**git commit -m "Helpful message"**
3. Pull from the upstream repository (i.e. GitHub)	**git pull**
4. Push local changes that you commited to the upstream repo	**git push**

Clone the repo.	**git clone REPOSITORY-URL**
See the commit history (hit spacebar to scroll down or q to exit)	**git log**
See changes	**git status**

Git Branches

- Branches allow you to take a snapshot of your existing repo and try out a new idea without affecting your main (i.e. "master") branch.
- Only when you (and your collaborators) are 100% satisfied, would you **merge** your work back into the master branch.
  - How bugs are caught and fixed.
  - How most new features in modern software and apps are developed.
  - You can use branches to try out new ideas and analyses
- If you aren't happy, then you can just delete the branch and continue as if it never existed.



- Create a new branch on your local machine and switch to it **git checkout -b NAME-OF-YOUR-BRANCH**
- Push new branch to GitHub **git push origin NAME-OF-YOUR-BRANCH**
- List all branches on your local machine **git branch**
- Switch back to the master branch **git checkout master**
- Delete a branch **git branch -d NAME-OF-YOUR-BRANCH**

Merging Branches and Pull Requests

1. Local changes

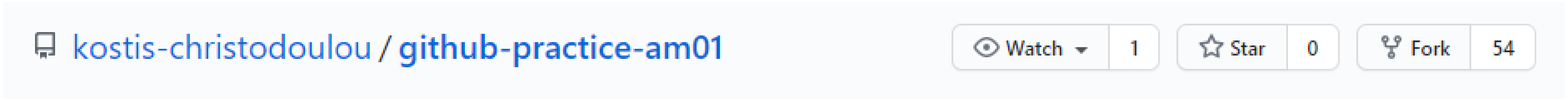
- Commit your final changes to the new branch (say we call it "new-idea").
- Switch back to the master branch: **git checkout master**
- Merge in the new-idea branch changes: **git merge new-idea**
- Delete the new-idea branch (optional): **git branch -d new-idea**

2. Remote changes (e.g., pull requests on GitHub)

- Pull requests are a way to notify collaborators — or yourself! — that you have completed a feature.
- You write a summary of all the changes contained in the branch.
- You then assign suggested reviewers of your code — including yourself potentially — who are then able to approve these changes ("Merge pull request") on GitHub.

Forks

A fork is an independent copy of the repo under your GitHub account. Forking a repository allows you to freely experiment with changes without affecting the original project. To fork a repo on GitHub just click the "Fork" button in the top-right corner of said repo.



Once you fork a repo, you are free to do anything you want to it.
Forking in combination with pull requests is actually how much of the world's software is developed.

In our case, when you added your names to the portfolio website list, you did the following:

1. Outside user X forks kostis-christodoulou/github-practice-am01 repo. She adds a new feature; her name and url for the webpage, or fixes a bug she's identified. She **1. commits her changes** and then **2. issues an upstream pull request**.
2. I am notified and can then decide whether to merge X's contribution with the main project.

https://docs.github.com/en/github/getting-started-with-github/fork-a-repo

Merge conflicts

- Person 1 (P1): Invite Person 2 (P2) to join you as a collaborator on the "test" repo. (See *Settings* tab of your repo.)
- P2: Clone P1's repo to your local machine. Change into a new directory first or give it a different name (P1_test) to avoid conflicts with your own *test* repo. Git tracking will still work if you change the repo name locally. Make some edits to the README (e.g. delete lines of text and add your own). Stage, commit and push these changes.
- P1: Make your own changes to the README on your local machine. Stage, commit, pull, and then try to push your committed changes. P1 should encounter a **merge conflict** error. git status in the shell/terminal should come back with

```
Unmerged paths:
  (use "git add <file>..." to mark resolution)
   * both modified:   README.md
```

- Git is protecting P1 by refusing the merge. It wants to make sure that you don't accidentally overwrite all of your changes by pulling P2's version of the README. Let us edit README

```
# README
Some text here.
<<<<<<< HEAD
Text added by Partner 2.
=======
Text added by Partner 1.
>>>>>>> 8dd2b2aaa67fac14469bda3caa08c75cd616b1d1
More text here.
```

<<<<<<< HEAD Indicates the start of the merge conflict.

======= Indicates the break point used for comparison.

>>>>>>> <long string> Indicates the end of the lines that had a merge conflict.

Fixing conflicts is a simple matter of (manually) editing the README file.
Delete the lines of the text that you don't want and then delete the special Git merge conflict symbols.
Once done, you can stage, commit, pull and finally push your changes to GitHub repo

Line endings and different operating systems

During collaboration with a colleague on a project, you may find cases where Git is highlighting differences on seemingly unchanged sentences.

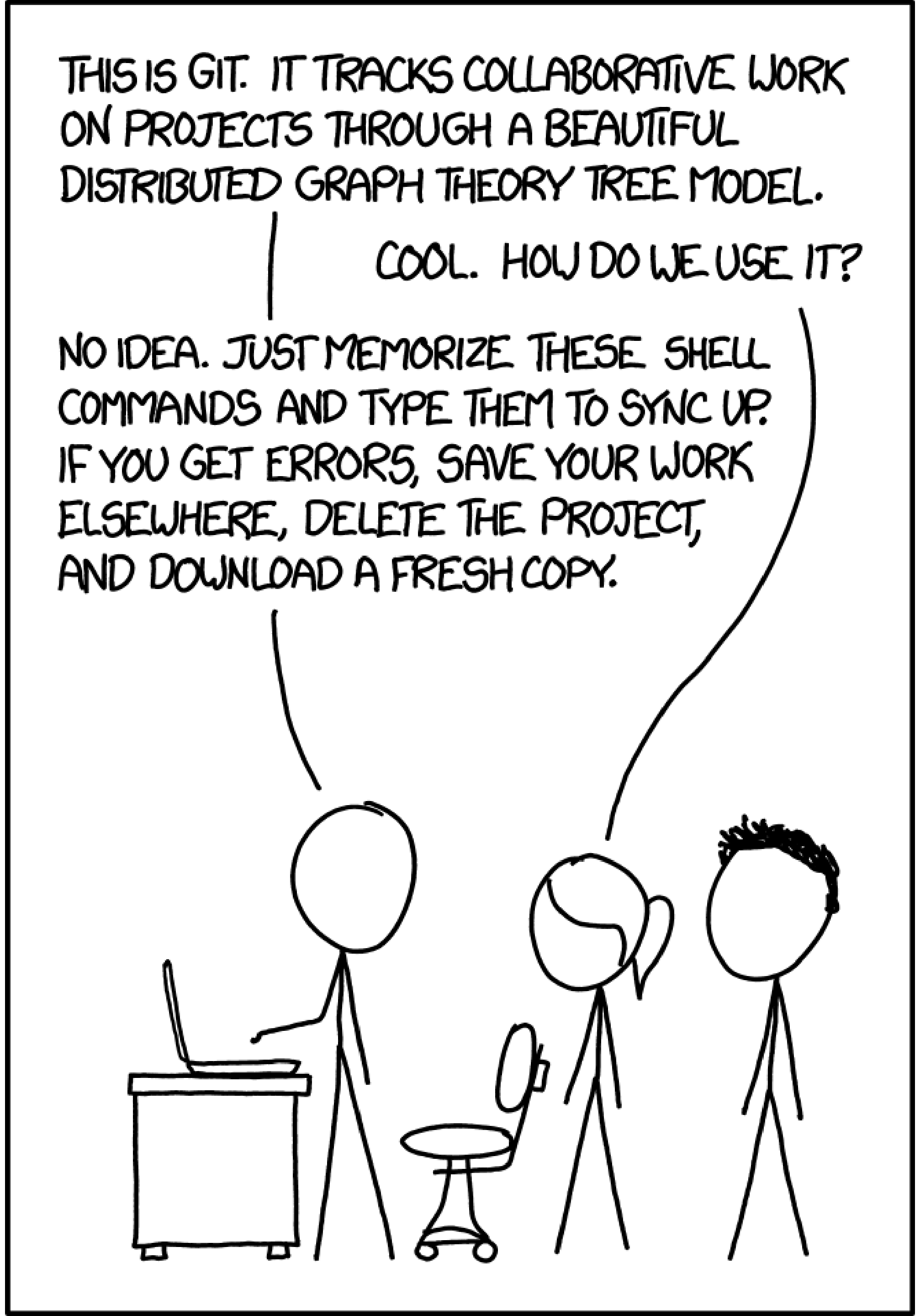
If that is the case, check whether your partner is using a different operating system to you.
The "culprit" is the fact that Git evaluates an invisible character at the end of every line. This is how Git tracks changes.

- For Linux and MacOS, that ending is "LF"
- For Windows, that ending is "CRLF"

To solve this OS incompatibility, open up the shell/terminal and enter

macOS	**git config --global core.autocrlf input**
Windows	**git config --global core.autocrlf true**

https://docs.github.com/en/github/using-git/configuring-git-to-handle-line-endings


THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

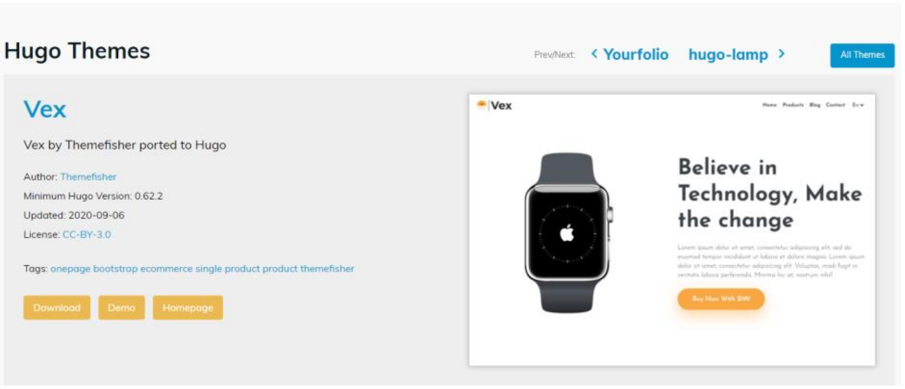
https://xkcd.com/1597/

Blogdown, hugo, your website

Steps to follow to get our website

1. Create a new public repo, called test, on GitHub and initialise with a README.
2. Copy the HTTPS/SSH link (the green Code button).
3. Open up RStudio.
4. Navigate to File -> New Project -> Version Control -> Git
5. Paste your copied link into the "Repository URL:" box.
6. Choose the project path ("Create project as subdirectory of:") and click Create Project.
7. Look at the top-right panel in your RStudio IDE. Do you see the Git tab?
8. There should already be some files in there, which we'll ignore for the moment
9. Go to Rstudio console (bottom-left) and type
   - `library(blogdown)`
   - `install_hugo()`
   - `hugo_version()`
10. Go to https://themes.gohugo.io/ to choose your favourite theme. We use theme Forty, but you can choose anything you like
11. All hugo themes are on github, so to use them you need to note the USER/NAME address.

Github USER/REPO address



Hugo Themes

Prev/Next: < Yourfolio hugo-lamp > All Themes

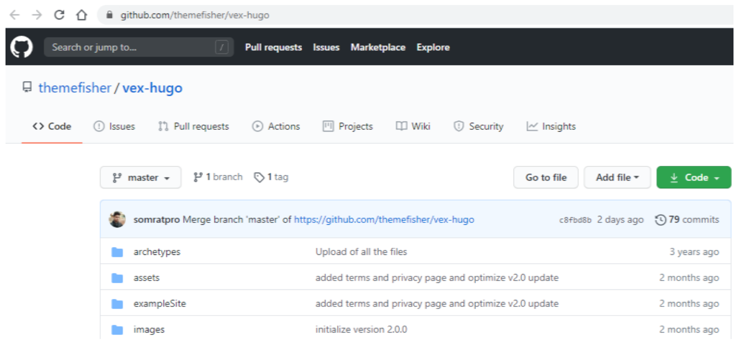
Vex

Vex by Themefisher ported to Hugo

Author: Themefisher
Minimum Hugo Version: 0.62.2
Updated: 2020-09-06
License: CC-BY-3.0

Tags: onepage bootstrap ecommerce single product product themefisher

Download Demo Homepage



github.com/themefisher/vex-hugo

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

themefisher / vex-hugo

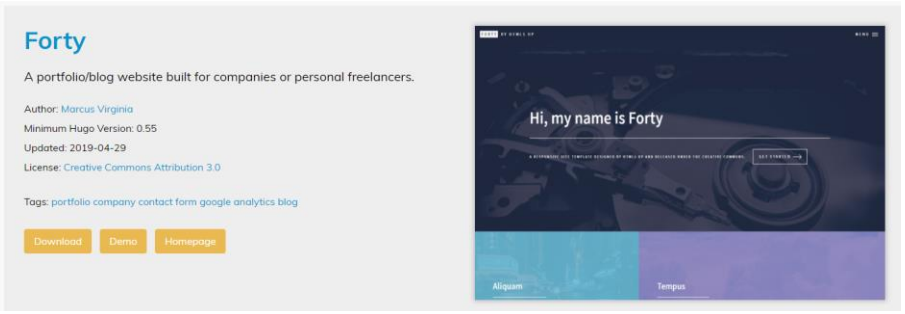
Code Issues Pull requests Actions Projects Wiki Security Insights

master 1 branch 1 tag Go to file Add file Code

somratpro Merge branch 'master' of https://github.com/themefisher/vex-hugo c8fbd8b 2 days ago 79 commits

archetypes	Upload of all the files	3 years ago
assets	added terms and privacy page and optimize v2.0 update	2 months ago
exampleSite	added terms and privacy page and optimize v2.0 update	2 months ago
images	initialize version 2.0.0	2 months ago

https://github.com/**themefisher/vex-hugo**



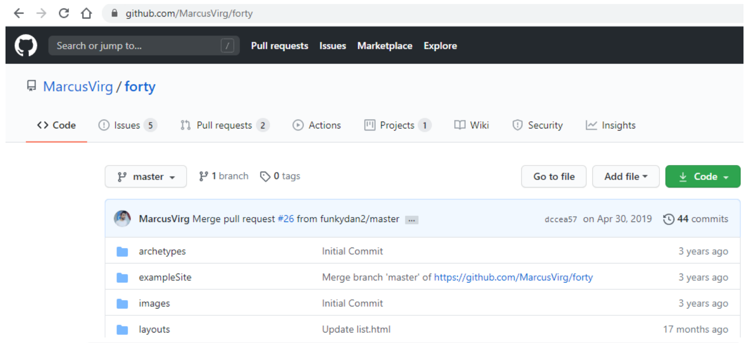
Forty

A portfolio/blog website built for companies or personal freelancers.

Author: Marcus Virginia
Minimum Hugo Version: 0.55
Updated: 2019-04-29
License: Creative Commons Attribution 3.0

Tags: portfolio company contact form google analytics blog

Download Demo Homepage



github.com/MarcusVirg/forty

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

MarcusVirg / forty

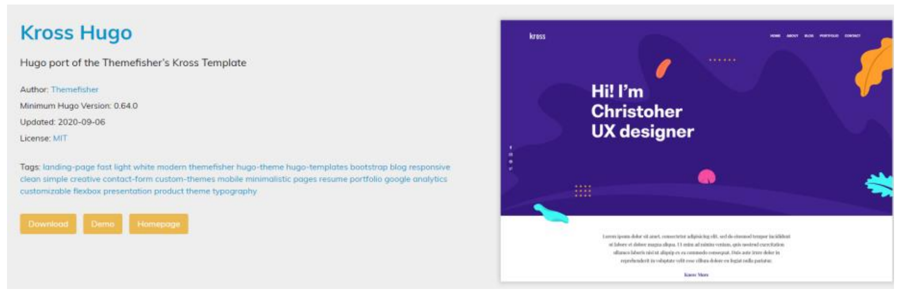
Code Issues 5 Pull requests 2 Actions Projects 1 Wiki Security Insights

master 1 branch 0 tags Go to file Add file Code

MarcusVirg Merge pull request #26 from funkydan2/master ... dccea57 on Apr 30, 2019 44 commits

archetypes	Initial Commit	3 years ago
exampleSite	Merge branch 'master' of https://github.com/MarcusVirg/forty	3 years ago
images	Initial Commit	3 years ago
layouts	Update list.html	17 months ago

https://github.com/**MarcusVirg/forty**



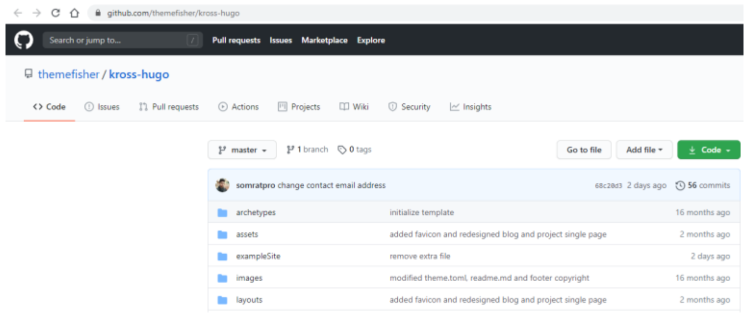
Kross Hugo

Hugo port of the Themefisher's Kross Template

Author: Themefisher
Minimum Hugo Version: 0.64.0
Updated: 2020-09-06
License: MIT

Tags: landing-page fast light white modern themefisher hugo-theme hugo-templates bootstrap blog responsive clean simple creative contact-form custom-themes mobile minimalistic pages resume portfolio google analytics customizable flexbox presentation product theme typography

Download Demo Homepage



github.com/themefisher/kross-hugo

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

themefisher / kross-hugo

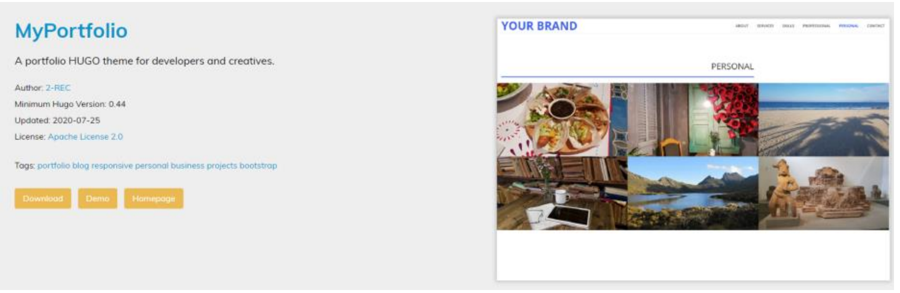
Code Issues Pull requests Actions Projects Wiki Security Insights

master 1 branch 0 tags Go to file Add file Code

somratpro change contact email address 68c20d3 2 days ago 56 commits

archetypes	initialize template	16 months ago
assets	added favicon and redesigned blog and project single page	2 months ago
exampleSite	remove extra file	2 days ago
images	modified theme.toml, readme.md and footer copyright	16 months ago
layouts	added favicon and redesigned blog and project single page	2 months ago

https://github.com/**themefisher/kross-hugo**



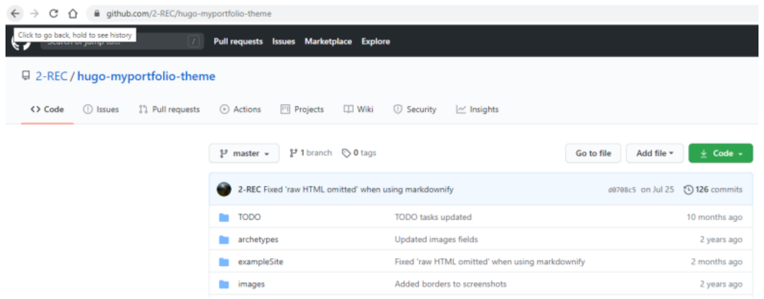
MyPortfolio

A portfolio HUGO theme for developers and creatives.

Author: 2-REC
Minimum Hugo Version: 0.44
Updated: 2020-07-25
License: Apache License 2.0

Tags: portfolio blog responsive personal business projects bootstrap

Download Demo Homepage



github.com/2-REC/hugo-myportfolio-theme

Click to go back, hold to see history

Pull requests Issues Marketplace Explore

2-REC / hugo-myportfolio-theme

Code Issues Pull requests Actions Projects Wiki Security Insights

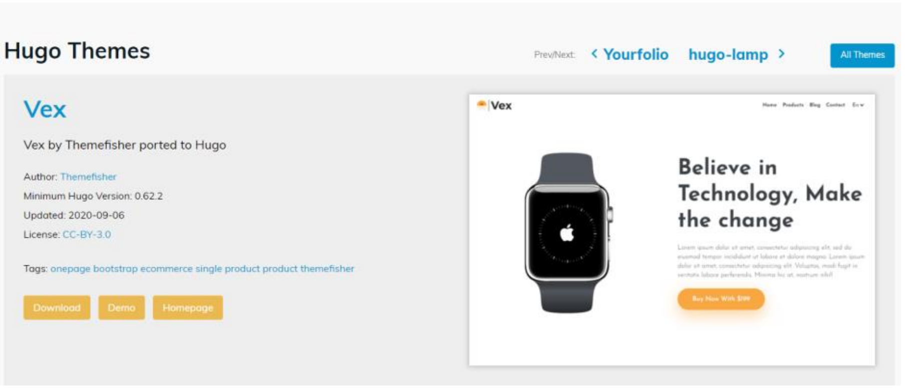
master 1 branch 0 tags Go to file Add file Code

2-REC Fixed 'raw HTML omitted' when using markdownify d0708c5 on Jul 25 126 commits

TODO	TODO tasks updated	10 months ago
archetypes	Updated images fields	2 years ago
exampleSite	Fixed 'raw HTML omitted' when using markdownify	2 months ago
images	Added borders to screenshots	2 years ago

https://github.com/**2-REC/hugo-myportfolio-theme**

Build a template site with your chosen theme



Hugo Themes

Vex

Vex by Themefisher ported to Hugo

Author: Themefisher

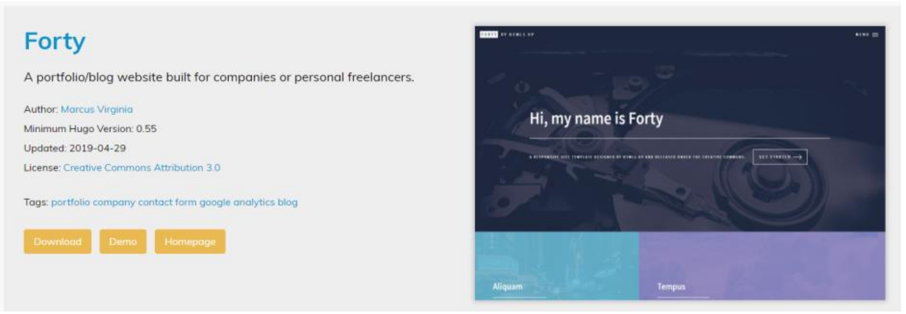
Minimum Hugo Version: 0.62.2

Updated: 2020-09-06

License: CC-BY-3.0

Tags: onepage bootstrap ecommerce single product product themefisher

```r
blogdown::new_site(theme = "themefisher/vex-hugo",
         sample = TRUE,
         theme_example = TRUE,
         empty_dirs = TRUE,
         to_yaml = TRUE)
```



Forty

A portfolio/blog website built for companies or personal freelancers.

Author: Marcus Virginia

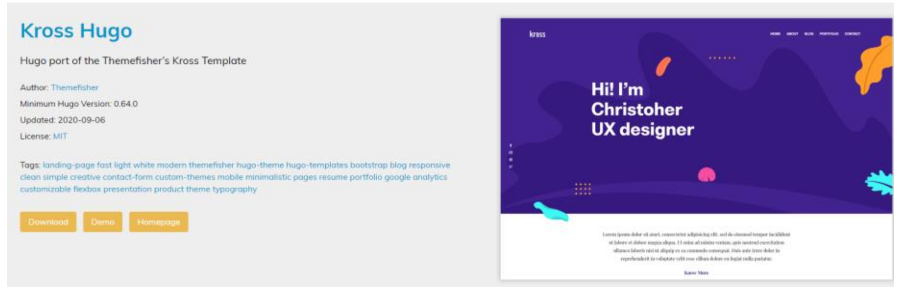
Minimum Hugo Version: 0.55

Updated: 2019-04-29

License: Creative Commons Attribution 3.0

Tags: portfolio company contact form google analytics blog

```r
blogdown::new_site(theme = "MarcusVirg/forty",
         sample = TRUE,
         theme_example = TRUE,
         empty_dirs = TRUE,
         to_yaml = TRUE)
```



Kross Hugo

Hugo port of the Themefisher's Kross Template

Author: Themefisher

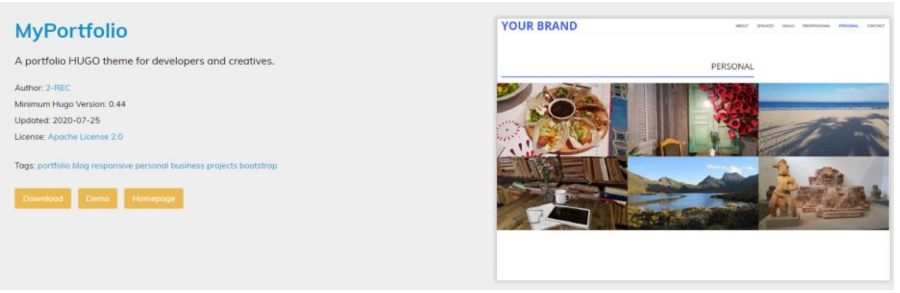
Minimum Hugo Version: 0.64.0

Updated: 2020-09-06

License: MIT

Tags: landing-page fast light white modern themefisher hugo-theme hugo-templates bootstrap blog responsive clean simple creative contact-form custom-themes mobile minimalistic pages resume portfolio google analytics customizable flexbox presentation product theme typography

```r
blogdown::new_site(theme = "themefisher/kross-hugo",
         sample = TRUE,
         theme_example = TRUE,
         empty_dirs = TRUE,
         to_yaml = TRUE)
```



MyPortfolio

A portfolio HUGO theme for developers and creatives.

Author: 2-REC

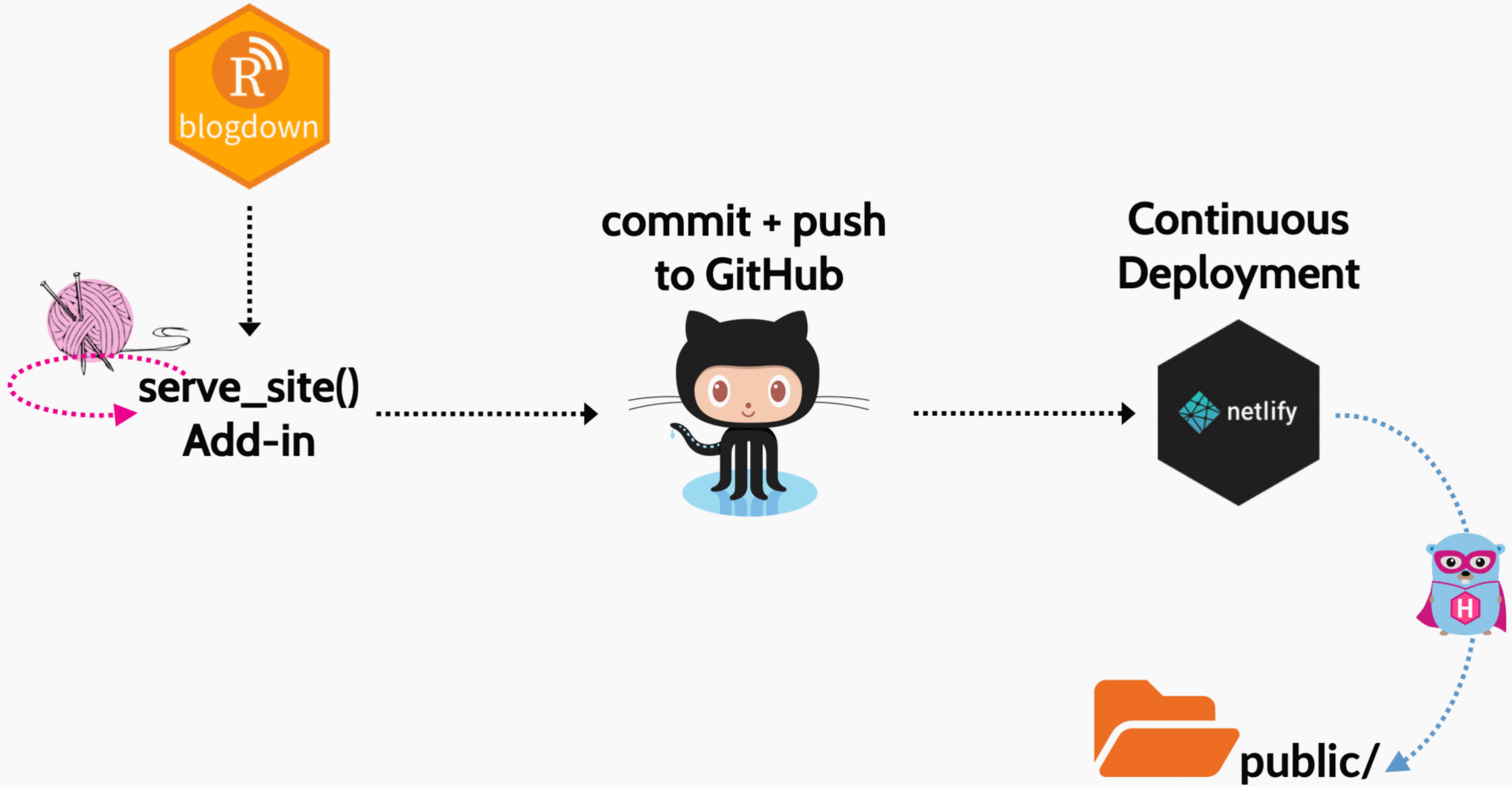
Minimum Hugo Version: 0.44

Updated: 2020-07-25

License: Apache License 2.0

Tags: portfolio blog responsive personal business projects bootstrap

```r
blogdown::new_site(theme = "2-REC/hugo-myportfolio-theme",
         sample = TRUE,
         theme_example = TRUE,
         empty_dirs = TRUE,
         to_yaml = TRUE)
```


blogdown
serve_site()
Add-in
commit + push
to GitHub
Continuous
Deployment
netlify
public/

For continuous deployment, netlify.toml

```
[build]
  publish = "public"
  command = "hugo"

[build.environment]
  HUGO_VERSION = "0.74.3"
  HUGO_ENABLEGITINFO = "true"

[context.production.environment]
  HUGO_ENV = "production"

[context.branch-deploy.environment]
  HUGO_VERSION = "0.74.3"

[context.deploy-preview.environment]
  HUGO_VERSION = "0.74.3"
```

Same as hugo_version()