

Pull Requests, Part II

Dan Burnett
Verifiable Claims
January 2017

After your first PR

- Unless successive PRs depend on prior ones, better for each to start from the most up-to-date version of the original repo
- To do this,
 - You need to keep your own repo in sync with the original one
 - The easiest way is using the command-line interface

The remaining slides cover

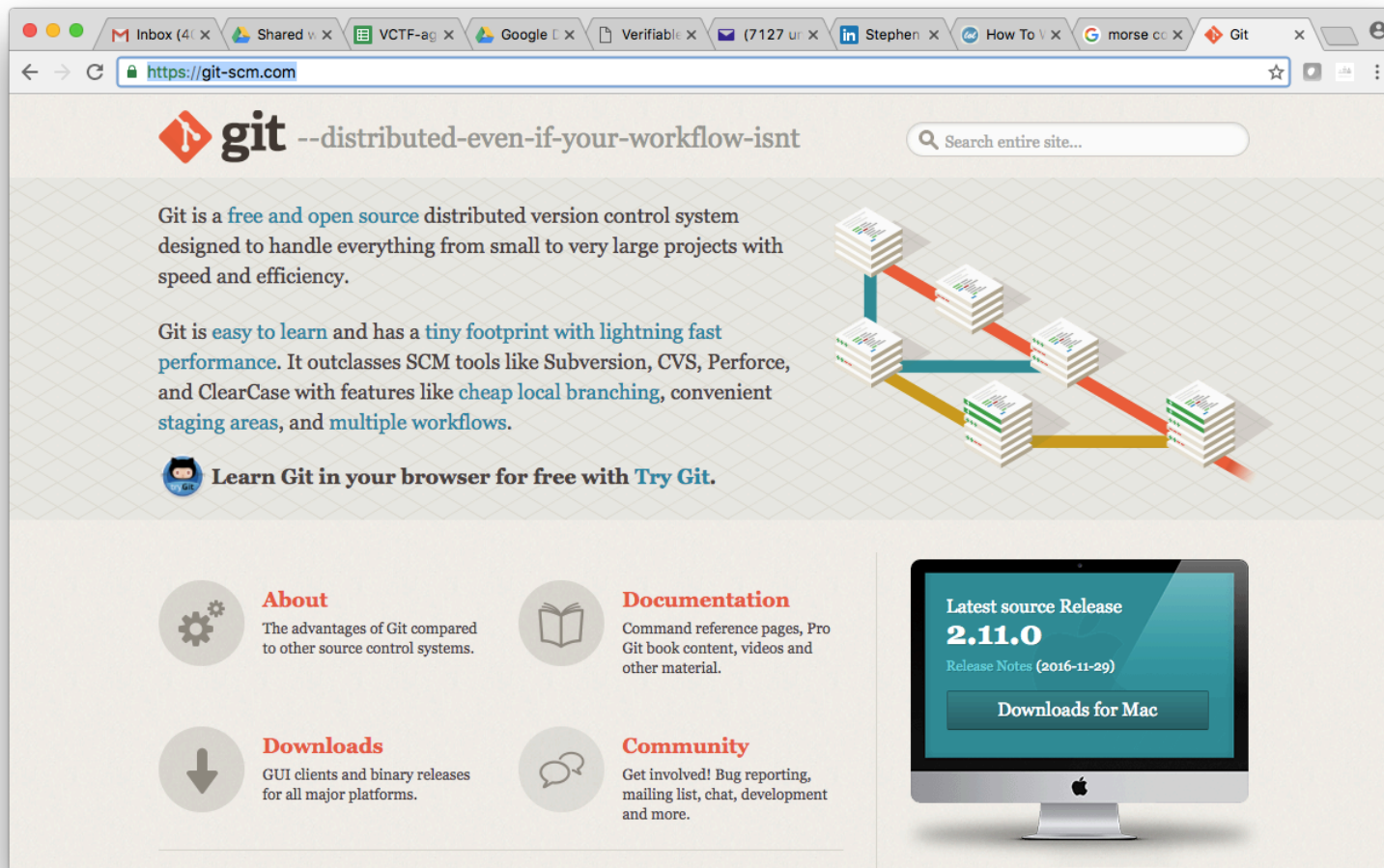
- How to set things up so it's easy to work locally
- A template set of commands and actions to use when creating a new PR
- Other useful tidbits
- NOTE: This tutorial shows command-line usage on Mac/Linux. For Windows, optionally consider using <https://git-for-windows.github.io/>

Working locally

- "Locally" means on your own computer
- Network connection only needed when interacting with server ('fetch', 'push', 'pull')
 - Otherwise, you can even work offline
- If you screw up your local copy, you can always start over!
 - since it doesn't affect anyone else

Install Git

- Install Git on your machine if it's not already there (<https://git-scm.com/>)




The screenshot shows the Git website homepage in a browser. The browser's address bar displays <https://git-scm.com>. The page features the Git logo and the tagline "--distributed-even-if-your-workflow-isnt". A search bar is located in the top right corner. The main content area includes a description of Git as a free and open source distributed version control system, followed by a list of features such as being easy to learn, having a tiny footprint, and lightning fast performance. Below this, there is a section for learning Git in the browser for free with Try Git. The bottom of the page is divided into four columns: About, Documentation, Downloads, and Community, each with a brief description and an icon. On the right side, there is a section for the latest source release, 2.11.0, with a button for downloading for Mac.

git --distributed-even-if-your-workflow-isnt

Git is a **free and open source** distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is **easy to learn** and has a **tiny footprint with lightning fast performance**. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like **cheap local branching**, convenient **staging areas**, and **multiple workflows**.

 **Learn Git in your browser for free with Try Git.**

About
The advantages of Git compared to other source control systems.

Documentation
Command reference pages, Pro Git book content, videos and other material.

Downloads
GUI clients and binary releases for all major platforms.

Community
Get involved! Bug reporting, mailing list, chat, development and more.

Latest source Release
2.11.0
Release Notes (2016-11-29)
[Downloads for Mac](#)

Copy clone path as before

The screenshot shows the GitHub interface for the repository 'burnburn / vc-data-model'. The URL in the browser is 'https://github.com/burnburn/vc-data-model'. The repository is a fork of 'opencreds/vc-data-model'. The page displays 146 commits, 2 branches, 0 releases, and 9 contributors. A dropdown menu is open for the 'Clone or download' button, showing the 'Clone with HTTPS' option with the URL 'https://github.com/burnburn/vc-data-model.' highlighted. Other options include 'Open in Desktop' and 'Download ZIP'. The repository description is 'Verifiable Claims Data Model and Representations specification' with a link to 'https://opencreds.github.io/vc-data-m...'. The file list includes 'CONTRIBUTING.md', 'LICENSE.md', 'index.html', and 'privacy-spectrum.svg'. A 'Add a README' button is visible at the bottom.

This time, clone locally

```
[DANIEL:burnburn $ git clone https://github.com/burnburn/vc-data-model.git ]
Cloning into 'vc-data-model'...
remote: Counting objects: 794, done.
remote: Total 794 (delta 0), reused 0 (delta 0), pack-reused 794
Receiving objects: 100% (794/794), 184.44 KiB | 248.00 KiB/s, done.
Resolving deltas: 100% (185/185), done.
Checking connectivity... done.
DANIEL:burnburn $ █
```

In new repo, see what remotes are defined (should just be 'origin')

```
[DANIEL:burnburn $ git clone https://github.com/burnburn/vc-data-model.git ]
Cloning into 'vc-data-model'...
remote: Counting objects: 794, done.
remote: Total 794 (delta 0), reused 0 (delta 0), pack-reused 794
Receiving objects: 100% (794/794), 184.44 KiB | 248.00 KiB/s, done.
Resolving deltas: 100% (185/185), done.
Checking connectivity... done.
[DANIEL:burnburn $ cd vc-data-model ]
[(gh-pages) DANIEL:vc-data-model $ git remote -v ]
origin https://github.com/burnburn/vc-data-model.git (fetch)
origin https://github.com/burnburn/vc-data-model.git (push)
[(gh-pages) DANIEL:vc-data-model $ █
```


Go to original ('upstream') repo

The screenshot shows a web browser window displaying the GitHub repository page for 'burnburn / vc-data-model'. The URL in the address bar is 'https://github.com/burnburn/vc-data-model'. The repository is a fork of 'opencreds/vc-data-model', which is circled in red. The page shows 146 commits, 2 branches, 0 releases, and 9 contributors. A 'Clone or download' button is visible, and a modal is open showing the 'Clone with HTTPS' option with the URL 'https://github.com/burnburn/vc-data-model'. The modal also includes 'Open in Desktop' and 'Download ZIP' buttons. The repository description is 'Verifiable Claims Data Model and Representations specification' with a link to 'https://opencreds.github.io/vc-data-m...'. The 'Code' tab is selected, and the file list includes 'CONTRIBUTING.md', 'LICENSE.md', 'index.html', and 'privacy-spectrum.svg'. A 'Add a README' button is also present.

Copy upstream repo address

The screenshot shows the GitHub repository page for 'opencreds / vc-data-model'. The repository has 146 commits, 1 branch, 0 releases, and 9 contributors. The 'Clone or download' button is highlighted with a red circle, and a dropdown menu is open, showing the 'Clone with HTTPS' option selected. The URL 'https://github.com/opencreds/vc-data-model' is displayed in the dropdown, and a red circle highlights the copy icon next to it. Below the dropdown, there are buttons for 'Open in Desktop' and 'Download ZIP'. At the bottom of the repository page, there is a section for adding a README.

Verifiable Claims Data Model and Representations specification <https://opencreds.github.io/vc-data-m...>

146 commits 1 branch 0 releases 9 contributors

Branch: gh-pages New pull request Create new file Upload files Find file **Clone or download**

Clone with HTTPS Use SSH

Use Git or checkout with SVN using the web URL.

<https://github.com/opencreds/vc-data-model>

Open in Desktop Download ZIP

Help people interested in this repository understand your project by adding a README. [Add a README](#)

Add original repo as 'upstream'

```
[DANIEL:burnburn $ git clone https://github.com/burnburn/vc-data-model.git ]
Cloning into 'vc-data-model'...
remote: Counting objects: 794, done.
remote: Total 794 (delta 0), reused 0 (delta 0), pack-reused 794
Receiving objects: 100% (794/794), 184.44 KiB | 25.00 KiB/s, done.
Resolving deltas: 100% (185/185), done.
Checking connectivity... done.
[DANIEL:burnburn $ cd vc-data-model ]
[(gh-pages) DANIEL:vc-data-model $ git remote -v ]
origin https://github.com/burnburn/vc-data-model.git (fetch)
origin https://github.com/burnburn/vc-data-model.git (push)
[(gh-pages) DANIEL:vc-data-model $ git remote add upstream https://github.com/ope
ncreds/vc-data-model.git
[(gh-pages) DANIEL:vc-data-model $ git remote -v ]
origin https://github.com/burnburn/vc-data-model.git (fetch)
origin https://github.com/burnburn/vc-data-model.git (push)
upstream https://github.com/opencreds/vc-data-model.git (fetch)
upstream https://github.com/opencreds/vc-data-model.git (push)
[(gh-pages) DANIEL:vc-data-model $ ]
```

Setup is done

- At this point you can now remotely
 - branch and do work on your local repo copy
 - without affecting your GitHub repo
 - update your local repo to match the 'upstream' repo
 - push changes to your source repo on GitHub (the 'origin')
 - which on GitHub can be used to create a PR

So, how do we do it?

- Locally
 - Checkout default branch and make sure it is up-to-date with server copy
 - Fetch and merge in any upstream changes (and push to server copy)
 - Create a new branch
 - Make your edits and commit them to local copy
 - Push the new branch to the server
- On GitHub
 - Create a new PR from that branch as before

Checkout default and sync with GitHub

```
[(somebranch) DANIEL:vc-data-model $ git checkout gh-pages  
Switched to branch 'gh-pages'  
Your branch is up-to-date with 'origin/gh-pages'.  
[(gh-pages) DANIEL:vc-data-model $ git pull  
Already up-to-date.  
(gh-pages) DANIEL:vc-data-model $ █
```

Fetch, merge, and push upstream changes since last time

```
[(somebranch) DANIEL:vc-data-model $ git checkout gh-pages  
Switched to branch 'gh-pages'  
Your branch is up-to-date with 'origin/gh-pages'.  
[(gh-pages) DANIEL:vc-data-model $ git pull  
Already up-to-date.  
[(gh-pages) DANIEL:vc-data-model $ git fetch upstream  
From https://github.com/opencreds/vc-data-model  
* [new branch]      gh-pages    -> upstream/gh-pages  
[(gh-pages) DANIEL:vc-data-model $ git merge upstream/gh-pages  
Already up-to-date.  
[(gh-pages) DANIEL:vc-data-model $ git push  
Everything up-to-date  
(gh-pages) DANIEL:vc-data-model $ █
```

Create new branch, edit, commit, and push to GitHub

```
[(gh-pages) DANIEL:vc-data-model $ git fetch upstream
From https://github.com/opencreds/vc-data-model
 * [new branch]      gh-pages -> upstream/gh-pages
[(gh-pages) DANIEL:vc-data-model $ git merge upstream/gh-pages
Already up-to-date.
[(gh-pages) DANIEL:vc-data-model $ git push
Everything up-to-date
[(gh-pages) DANIEL:vc-data-model $ git checkout -b anothersample
Switched to a new branch 'anothersample'
[(anothersample) DANIEL:vc-data-model $ emacs index.html
[(anothersample) DANIEL:vc-data-model $ git commit -am "add comment about updating
g a paragraph"
[anothersample c8ac04a] add comment about updating a paragraph
 1 file changed, 1 insertion(+)
[(anothersample) DANIEL:vc-data-model $ git push origin anothersample
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 360 bytes | 0 bytes/s, done.
Total 3 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/burnburn/vc-data-model.git
 * [new branch]      anothersample -> anothersample
(anothersample) DANIEL:vc-data-model $
```


Create new PR from that branch

The screenshot shows the GitHub interface for the repository 'burnburn / vc-data-model'. The repository is a fork of 'opencreds/vc-data-model'. The main content area displays the repository's metadata: 146 commits, 3 branches, 0 releases, and 9 contributors. Below this, a section titled 'Your recently pushed branches:' shows a branch named 'another-sample' pushed 2 minutes ago. A red circle highlights the 'Compare & pull request' button next to this branch. Below the branch list, there are buttons for 'Branch: gh-pages', 'New pull request', 'Create new file', 'Upload files', 'Find file', and 'Clone or download'. The commit history section shows the latest commit by 'stevenrowat' with the message '3 typos of "representd", in 4.1.1, 4.1.2, 4.3.2', dated 17 days ago. Below the commit history, there is a list of files: CONTRIBUTING.md, LICENSE.md, index.html, and privacy-spectrum.svg, each with a brief description and a timestamp. At the bottom, there is a prompt to 'Add a README'.

This is just like in Part I

The screenshot shows the GitHub interface for creating a pull request. At the top, the browser address bar displays the URL: `https://github.com/opencreds/vc-data-model/compare/gh-pages...burnburn:another-sample?expand=1`. The main heading is "Open a pull request", followed by a sub-heading: "Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#)."

Below this, there are two dropdown menus for "base fork" (set to "opencreds/vc-data-model") and "base" (set to "gh-pages"), and two more for "head fork" (set to "burnburn/vc-data-model") and "compare" (set to "another-sample"). A green checkmark indicates "Able to merge. These branches can be automatically merged."

A yellow banner below the dropdowns reads: "Please review the [guidelines for contributing](#) to this repository."

The main content area features a text input field with the placeholder text "Add comment about updating a paragraph", which is circled in red. Below the input field are "Write" and "Preview" tabs, and a rich text editor toolbar. The text entered in the field is "This is the sample PR from Part II of the Pull Request tutorial. DO NOT MERGE.", also circled in red. Below the text area is a dashed line and the instruction: "Attach files by dragging & dropping, selecting them, or pasting from the clipboard."

At the bottom left, there is a checked checkbox for "Allow edits from maintainers. [Learn more](#)". At the bottom right, a green button labeled "Create pull request" is circled in red. On the right side of the page, there are sections for "Reviewers" (No reviews—request one), "Assignees" (No one—assign yourself), "Labels" (None yet), and "Milestone" (No milestone), each with a gear icon for settings.

About the preceding steps

- This is only **one** possible workflow
 - It only handles simple branching
 - But if used each time will keep you in sync with the main W3C repository
 - It also results in PRs that are fairly simple for the editors to merge
- If it fails somewhere in the process
 - then you need to learn more about Git or ask for help

GitHub resources

- Google search is your friend – no kidding
 - Most of my questions have already been answered at StackOverflow.com
 - Many others at git-scm.com
- When you are ready to learn what's really going on,
 - I highly recommend the "Pro Git" book:
<https://git-scm.com/book/en/v2>