Git concepts for newcomers — Part 1: What is a DVCS?

Sébastien Dubois. Follow

May 14, 2020 · 4 min read ★

I'm starting a series about Git. The goal of this series is to gently introduce the most important concepts of the Git version control system (VCS).

This is the first article of the series:

- Part #1 — What is a DVCS: https://itnext.io/git-concepts-for-

newcomers-part-1-what-is-a-dvcs-bc873076c424

- Part #2 — Working tree and staging area: https://itnext.io/git-concepts-for-newcomers-part-2-git-repository-working-tree-and-staging-area-a2e720bf3528
- Part #3 — Commits, log and amend: https://medium.com/@dSebastien/git-concepts-for-newcomers-part-3-commits-log-and-amend-6dcbb05370c
- Part #4 — Branches: https://medium.com/@dSebastien/git-concepts-for-newcomers-part-4-branches-52aee1da4385

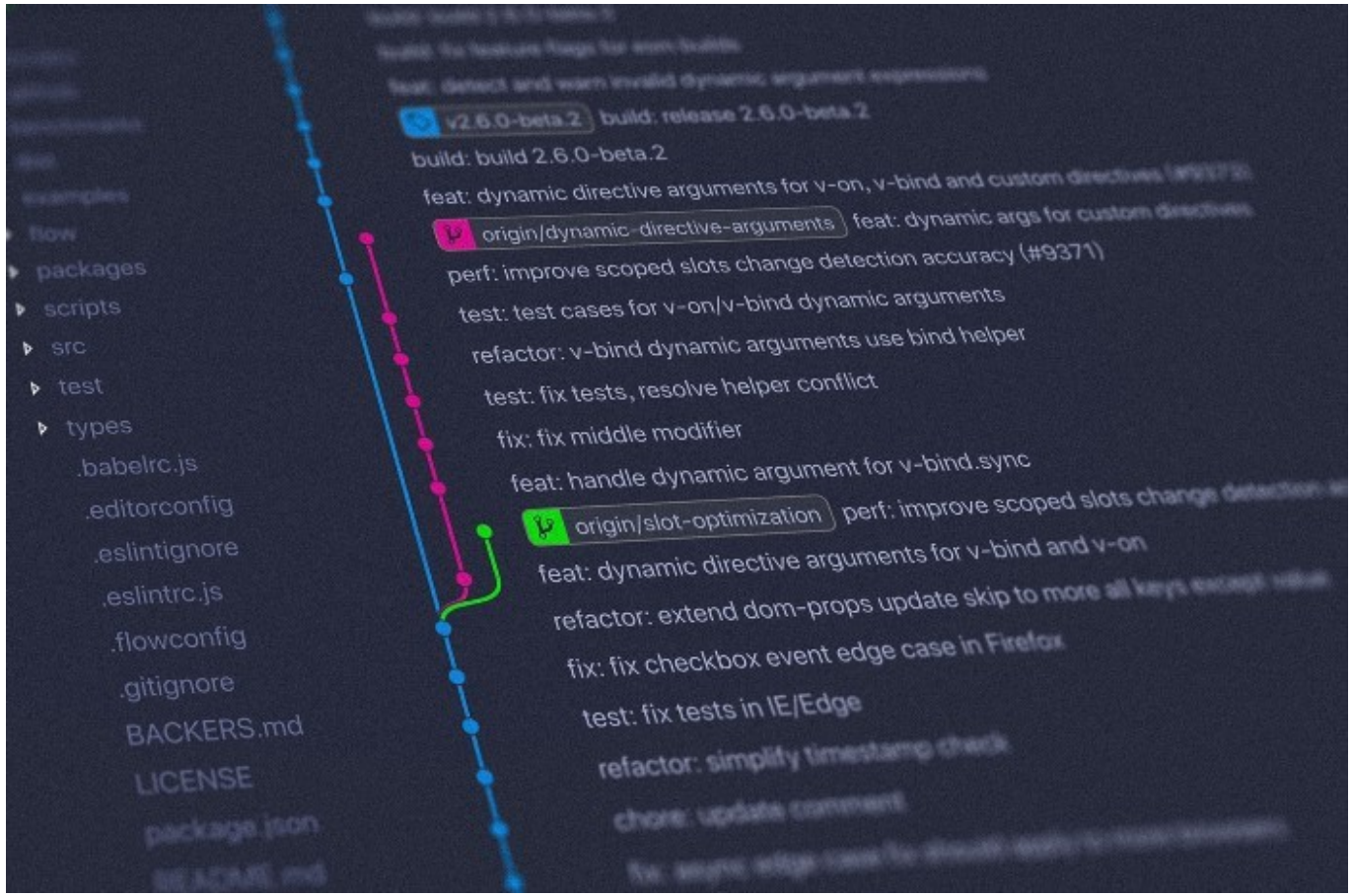

Image courtesy of Yancy Min

This series will be useful for juniors, developers coming from another version control system and even git users that aren't so comfortable with all the concepts.

These articles won't go in depth, that's really not the point. My goal is simply to provide *short* and *easy to understand* explanations about the most important concepts/commands/ideas.

Ultimately, I'd like to finish up with the explanation of a really efficient and powerful git workflow for distributed teams.

But before we get to that, let's start with the foundations. In this first post, I'll explain why the distributed aspect of Git is so awesome and how it differs from centralized systems such as Subversion.

If you never even heard of git, then please start by taking a look at the wikipedia page before continuing.

You can also already install git on your machine if you haven't done it already

Downloads

Git comes with built-in GUI tools (git-gui, gitk), but there are several third-party tools for users looking for a...

git-scm.com

Distributed Version Control System

Git is a *distributed* version control system (DVCS), or peer-to-peer version control system, as opposed to centralized systems like Subversion.

There's no notion of a "master" or "central" repository with Git. This might be perceived as a "flaw" at first, but once you realize what it truly means, you'll see that it's actually the greatest strength of Git.

When you have a git repository on your machine, it is self-contained. It works perfectly on its own. There's no need to have a server running somewhere to enable you to use git.

With older version control systems (VCS) like Subversion, a central

repository had to be available to be able to interact the the versioning system. Without it, no way to check the logs, no way to create new commits, no way to switch to another branch, etc.

Not only that, but since everything had to go through the central repository, everything was also much slower.

With a git repository on your machine, you have everything you need. The repository contains the whole history; all the branches, all the commits, all the tags, *everything*. Thanks to this, creating a backup for a Git repository is incredibly simple: just make a copy of it and you're done.

Creating a git repository is as simple as using:

```
git init
```

With a git repository on your machine, you can create new commits, reorder/rearrange/modify/delete existing ones, create tags, etc. This alone makes git super easy to use. Note that there's a `--bare` flag that you can add, but I won't explain that here. Check out this article if you want to learn

more about this.

In addition, since everything happens locally on your machine and without requiring interactions with a remote server, everything is *blazing fast*.

But this is only the tip of the iceberg. You can push and pull content/changes from other git repositories. And the repositories you push/pull from can be anywhere: on your file system, on a colleague's machine, on a distant server. It doesn't matter one bit.

This is where Git truly shines. You can make changes in your repository, then get (fetch) the changes made by other people and integrate (merge) them in your repository, or push your own changes to other repositories. You can do this with as many remote repositories as you'd like, as often as you'd like. This distributed aspect of git is empowering because it allows you to work in isolation, but also to synchronize your work easily with others.

Remember the centralized approach of systems like Subversion? Well you can do exactly the same with git if you wish to (and people often do). You can create a centralized git repository on a platform like Github, Gitlab,

Bitbucket, your NAS or whatever else and then synchronize your local repository with that one.

There are no topology constraints; you organize as you wish to and pull/push code with whatever you want. That's truly liberating and empowering.

Of course, there are already a few battle-tested *git workflows*. I'll explore my favourite at the end of this series.

Conclusion

In this article, I've explained what it means for a version control system like git to be distributed. And that is *freedom*. Freedom to implement different workflows, to collaborate however you want.

In the next post of this series, we'll explore the different "areas" of git. After reading it, you'll see clearly what the difference is between your git repository, the working directory and the staging area.

That's it for today!

Liked this article? Click that “Like” button below to see more of it others see it too!

PS: If you want to learn tons of other cool things about software/Web development, TypeScript, Vue, Kotlin, Java, Docker/Kubernetes and other cool subjects, then don’t hesitate to grab a copy subscribe to my newsletter!