Get Good at Git!

(xD)

Natalia Maniakowska

skygate

25-02-2020

Outline

Outline

What is Git? What it isn't?

- a **distributed version control system**
- created by Linus Torvalds for development of the Linux cernel (2005 – in, like, a few days!)
- free, open source – GNU General Public License
- other DVCS-s: Mercurial, BitKeeper, DCVS... (Subversion too, but it's not distributed – it's centralised)
- GitHub $\neq$ Git (!)

What is Git? What it isn't?

- a **distributed version control system**
- created by Linus Torvalds for development of the Linux cernel (2005 – in, like, a few days!)
- free, open source – GNU General Public License
- other DVCS-s: Mercurial, BitKeeper, DCVS... (Subversion too, but it's not distributed – it's centralised)
- GitHub $\neq$ Git (!)

What is Git? What it isn't?

- a **distributed version control system**
- created by Linus Torvalds for development of the Linux cernel (2005 – in, like, a few days!)
- free, open source – GNU General Public License
- other DVCS-s: Mercurial, BitKeeper, DCVS... (Subversion too, but it's not distributed – it's centralised)
- GitHub $\neq$ Git (!)

What is Git? What it isn't?

- a **distributed version control system**
- created by Linus Torvalds for development of the Linux cernel (2005 – in, like, a few days!)
- free, open source – GNU General Public License
- other DVCS-s: Mercurial, BitKeeper, DCVS... (Subversion too, but it's not distributed – it's centralised)
- GitHub $\neq$ Git (!)

What is Git? What it isn't?

- a **distributed version control system**
- created by Linus Torvalds for development of the Linux cernel (2005 – in, like, a few days!)
- free, open source – GNU General Public License
- other DVCS-s: Mercurial, BitKeeper, DCVS... (Subversion too, but it's not distributed – it's centralised)
- GitHub $\neq$ Git (!)

What is a distributed version control system?

- *distributed* – every developer has her/his own copy
- *version control* – keep track of the **changes**, revert them if needed, etc.
- Joel Spolsky* (in 2010): *"possibly the biggest advance in software development technology in the [past] ten years"*

  ▸ Article

  (That was actually about Mercurial, but still. . .)

  *That guy who invented Trello, Glitch, and co-founded StackOverflow, among others

What is a distributed version control system?

- *distributed* – every developer has her/his own copy
- *version control* – keep track of the **changes**, revert them if needed, etc.
- Joel Spolsky* (in 2010): *"possibly the biggest advance in software development technology in the [past] ten years"*

  ▸ Article

  (That was actually about Mercurial, but still...)

  *That guy who invented Trello, Glitch, and co-founded StackOverflow, among others

What is a distributed version control system?

- *distributed* – every developer has her/his own copy
- *version control* – keep track of the **changes**, revert them if needed, etc.
- Joel Spolsky* (in 2010): *"possibly the biggest advance in software development technology in the [past] ten years"*

  ▸ Article

  (That was actually about Mercurial, but still. . .)

  *That guy who invented Trello, Glitch, and co-founded StackOverflow, among others

OK, sounds cool. Let's learn Git.

So if Git really is so cool and useful, what's the problem with it?

▸ git manual

Outline

Git – definitions

- repository (repo), .git directory
- local vs. remote [repository], origin, upstream
- working directory (working tree)
- staging area (index)
- fetch vs. pull

Git – definitions

- repository (repo), .git directory
- local vs. remote [repository], origin, upstream
- working directory (working tree)
- staging area (index)
- fetch vs. pull

Git – definitions

- repository (repo), .git directory
- local vs. remote [repository], origin, upstream
- working directory (working tree)
- staging area (index)
- fetch vs. pull

Git – definitions

- repository (repo), .git directory
- local vs. remote [repository], origin, upstream
- working directory (working tree)
- staging area (index)
- fetch vs. pull

Git – definitions

- repository (repo), .git directory
- local vs. remote [repository], origin, upstream
- working directory (working tree)
- staging area (index)
- fetch vs. pull

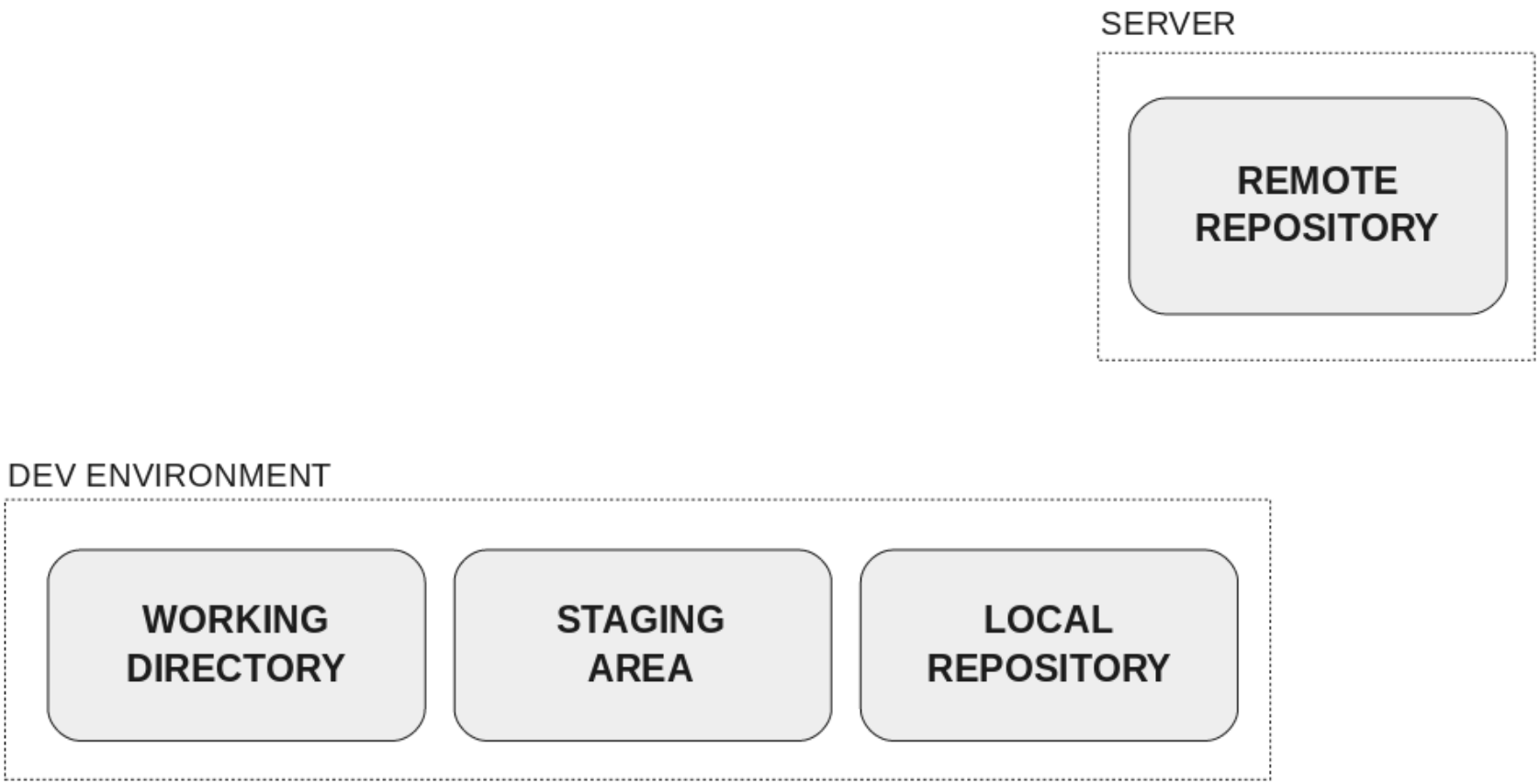

Figure: A distributed version control system. Source of the picture: `https://rachelcarmena.github.io/2018/12/12/how-to-teach-git.html`

And more definitions...

- commit, commit hash
- branch as a label
- `HEAD` (and `ORIG_HEAD`, and `HEAD^`, `HEAD~5`, ...)
- detached HEAD state

And more definitions...

- commit, commit hash
- branch as a label
- `HEAD` (and `ORIG_HEAD`, and `HEAD^`, `HEAD~5`, ...)
- detached HEAD state

And more definitions...

- commit, commit hash
- branch as a label
- `HEAD` (and `ORIG_HEAD`, and `HEAD^`, `HEAD~5`, ...)
- detached HEAD state

And more definitions...

- commit, commit hash
- branch as a label
- `HEAD` (and `ORIG_HEAD`, and `HEAD^`, `HEAD~5`, ...)
- detached HEAD state

Rebase

- Bitbucket tutorial: Merging vs. Rebasing (link)
- `git rebase --interactive <commit>` – delete, squash, amend, change the order of, etc. the commits after `<commit>`

Rebase

- Bitbucket tutorial: Merging vs. Rebasing (link)
- `git rebase --interactive <commit>` – delete, squash, amend, change the order of, etc. the commits after `<commit>`

Some useful commands

- `git config`
  - ⋆ set editor: `git config --global core.editor emacs`
- `git status`, `git log`, `git diff`, `git show`
- `git add --patch` (or `-p`) – review each changed piece of code and decide whether to add it or not
- `git stash`
- `git reset`
- `git revert`
  - ⋆ see also: How to revert a merge commit (link)
- `git cherry-pick`
- `git reflog` – when something goes terribly wrong

Some useful commands

- `git config`
  - ⋆ set editor: `git config --global core.editor emacs`
- `git status`, `git log`, `git diff`, `git show`
- `git add --patch` (or `-p`) – review each changed piece of code and decide whether to add it or not
- `git stash`
- `git reset`
- `git revert`
  - ⋆ see also: How to revert a merge commit (link)
- `git cherry-pick`
- `git reflog` – when something goes terribly wrong

Some useful commands

- `git config`
  - ⋆ set editor: `git config --global core.editor emacs`
- `git status`, `git log`, `git diff`, `git show`
- `git add --patch` (or `-p`) – review each changed piece of code and decide whether to add it or not
- `git stash`
- `git reset`
- `git revert`
  - ⋆ see also: How to revert a merge commit (link)
- `git cherry-pick`
- `git reflog` – when something goes terribly wrong

Some useful commands

- `git config`
  - ⋆ set editor: `git config --global core.editor emacs`
- `git status`, `git log`, `git diff`, `git show`
- `git add --patch` (or `-p`) – review each changed piece of code and decide whether to add it or not
- `git stash`
- `git reset`
- `git revert`
  - ⋆ see also: How to revert a merge commit (link)
- `git cherry-pick`
- `git reflog` – when something goes terribly wrong

Some useful commands

- `git config`
  - ⋆ set editor: `git config --global core.editor emacs`
- `git status`, `git log`, `git diff`, `git show`
- `git add --patch` (or `-p`) – review each changed piece of code and decide whether to add it or not
- `git stash`
- `git reset`
- `git revert`
  - ⋆ see also: How to revert a merge commit (link)
- `git cherry-pick`
- `git reflog` – when something goes terribly wrong

Some useful commands

- `git config`
  - ⋆ set editor: `git config --global core.editor emacs`
- `git status`, `git log`, `git diff`, `git show`
- `git add --patch` (or `-p`) – review each changed piece of code and decide whether to add it or not
- `git stash`
- `git reset`
- `git revert`
  - ⋆ see also: How to revert a merge commit (link)
- `git cherry-pick`
- `git reflog` – when something goes terribly wrong

Some useful commands

- `git config`
  - ⋆ set editor: `git config --global core.editor emacs`
- `git status`, `git log`, `git diff`, `git show`
- `git add --patch` (or `-p`) – review each changed piece of code and decide whether to add it or not
- `git stash`
- `git reset`
- `git revert`
  - ⋆ see also: How to revert a merge commit (link)
- `git cherry-pick`
- `git reflog` – when something goes terribly wrong

Some useful commands

- `git config`
  - ⋆ set editor: `git config --global core.editor emacs`
- `git status`, `git log`, `git diff`, `git show`
- `git add --patch` (or `-p`) – review each changed piece of code and decide whether to add it or not
- `git stash`
- `git reset`
- `git revert`
  - ⋆ see also: How to revert a merge commit (link)
- `git cherry-pick`
- `git reflog` – when something goes terribly wrong

Some useful commands

- `git config`
  - ⋆ set editor: `git config --global core.editor emacs`
- `git status`, `git log`, `git diff`, `git show`
- `git add --patch` (or `-p`) – review each changed piece of code and decide whether to add it or not
- `git stash`
- `git reset`
- `git revert`
  - ⋆ see also: How to revert a merge commit (link)
- `git cherry-pick`
- `git reflog` – when something goes terribly wrong

Some useful commands

- `git config`
  - ⋆ set editor: `git config --global core.editor emacs`
- `git status`, `git log`, `git diff`, `git show`
- `git add --patch` (or `-p`) – review each changed piece of code and decide whether to add it or not
- `git stash`
- `git reset`
- `git revert`
  - ⋆ see also: How to revert a merge commit (link)
- `git cherry-pick`
- `git reflog` – when something goes terribly wrong

A few less crucial, but still useful commands

- Oh-my-zsh aliases! Do check them out. ▸ Cheatsheet
- `git log --graph --oneline --all`
- `git blame` – "annotate" in PyCharm, or a plugin in VS
- `git push --dry-run`
- Cleanup (if your repo is getting hefty):
  - `git gc` – garbage collection
  - `git remote prune origin` or `git fetch --all --prune` – remove references to deleted remote branches (e.g. after they were merged)
  - `git reflog expire --expire=now --all && git gc --prune=now --aggressive` – erase all Git recovery/backup files (don't use it after `git reset --hard`!)

A few less crucial, but still useful commands

- Oh-my-zsh aliases! Do check them out. ▸ Cheatsheet
- `git log --graph --oneline --all`
- `git blame` – "annotate" in PyCharm, or a plugin in VS
- `git push --dry-run`
- Cleanup (if your repo is getting hefty):
  - `git gc` – garbage collection
  - `git remote prune origin` or `git fetch --all --prune` – remove references to deleted remote branches (e.g. after they were merged)
  - `git reflog expire --expire=now --all && git gc --prune=now --aggressive` – erase all Git recovery/backup files (don't use it after `git reset --hard`!)

A few less crucial, but still useful commands

- Oh-my-zsh aliases! Do check them out. ▸ Cheatsheet
- `git log --graph --oneline --all`
- `git blame` – "annotate" in PyCharm, or a plugin in VS
- `git push --dry-run`
- Cleanup (if your repo is getting hefty):
  - `git gc` – garbage collection
  - `git remote prune origin` or `git fetch --all --prune` – remove references to deleted remote branches (e.g. after they were merged)
  - `git reflog expire --expire=now --all && git gc --prune=now --aggressive` – erase all Git recovery/backup files (don't use it after `git reset --hard`!)

A few less crucial, but still useful commands

- Oh-my-zsh aliases! Do check them out. ▸ Cheatsheet
- `git log --graph --oneline --all`
- `git blame` – "annotate" in PyCharm, or a plugin in VS
- `git push --dry-run`
- Cleanup (if your repo is getting hefty):
  - `git gc` – garbage collection
  - `git remote prune origin` or `git fetch --all --prune` – remove references to deleted remote branches (e.g. after they were merged)
  - `git reflog expire --expire=now --all && git gc --prune=now --aggressive` – erase all Git recovery/backup files (don't use it after `git reset --hard`!)

A few less crucial, but still useful commands

- Oh-my-zsh aliases! Do check them out. ▸ Cheatsheet
- `git log --graph --oneline --all`
- `git blame` – "annotate" in PyCharm, or a plugin in VS
- `git push --dry-run`
- Cleanup (if your repo is getting hefty):
  - `git gc` – garbage collection
  - `git remote prune origin` or `git fetch --all --prune` – remove references to deleted remote branches (e.g. after they were merged)
  - `git reflog expire --expire=now --all && git gc --prune=now --aggressive` – erase all Git recovery/backup files (don't use it after `git reset --hard`!)

A few less crucial, but still useful commands

- Oh-my-zsh aliases! Do check them out. ▸ Cheatsheet
- `git log --graph --oneline --all`
- `git blame` – "annotate" in PyCharm, or a plugin in VS
- `git push --dry-run`
- Cleanup (if your repo is getting hefty):
  - ⋆ `git gc` – garbage collection
  - ⋆ `git remote prune origin` or `git fetch --all --prune` – remove references to deleted remote branches (e.g. after they were merged)
  - ⋆ `git reflog expire --expire=now --all && git gc --prune=now --aggressive` – erase all Git recovery/backup files (don't use it after `git reset --hard`!)

A few less crucial, but still useful commands

- Oh-my-zsh aliases! Do check them out. ▸ Cheatsheet
- `git log --graph --oneline --all`
- `git blame` – "annotate" in PyCharm, or a plugin in VS
- `git push --dry-run`
- Cleanup (if your repo is getting hefty):
  - ⋆ `git gc` – garbage collection
  - ⋆ `git remote prune origin` or `git fetch --all --prune` – remove references to deleted remote branches (e.g. after they were merged)
  - ⋆ `git reflog expire --expire=now --all && git gc --prune=now --aggressive` – erase all Git recovery/backup files (don't use it after `git reset --hard`!)

A few less crucial, but still useful commands

- Oh-my-zsh aliases! Do check them out. ▸ Cheatsheet
- `git log --graph --oneline --all`
- `git blame` – "annotate" in PyCharm, or a plugin in VS
- `git push --dry-run`
- Cleanup (if your repo is getting hefty):
  - ⋆ `git gc` – garbage collection
  - ⋆ `git remote prune origin` or `git fetch --all --prune` – remove references to deleted remote branches (e.g. after they were merged)
  - ⋆ `git reflog expire --expire=now --all && git gc --prune=now --aggressive` – erase all Git recovery/backup files (don't use it after `git reset --hard`!)

Outline

Fun facts

- Git is used in its own development. Git source code is stored on GitHub.
- Linux kernel developers don't do pull requests; they send code patches via email. Hence `git format-patch`, `git send-email`, and other commands.
  - Linus' comment
- Git also supports octopus merges, which have more than two parents. Probably the biggest octopus merge in Linux kernel had 66 (!) parents.
  - Article

Fun facts

- Git is used in its own development. Git source code is stored on GitHub.
- Linux kernel developers don't do pull requests; they send code patches via email. Hence `git format-patch`, `git send-email`, and other commands.

  ▸ Linus' comment
- Git also supports octopus merges, which have more than two parents. Probably the biggest octopus merge in Linux kernel had 66 (!) parents.

  ▸ Article

Fun facts

- Git is used in its own development. Git source code is stored on GitHub.
- Linux kernel developers don't do pull requests; they send code patches via email. Hence `git format-patch`, `git send-email`, and other commands.

  ▸ Linus' comment
- Git also supports octopus merges, which have more than two parents. Probably the biggest octopus merge in Linux kernel had 66 (!) parents.

  ▸ Article

Wrap-up

Main takeaways:

- **Never push with `--force` to public branches!**
- It's actually hard to permanently lose some of your work.
- Git is cool ♡

Wrap-up

Main takeaways:

- **Never push with `--force` to public branches!**
- It's actually hard to permanently lose some of your work.
- Git is cool ♡

Wrap-up

Main takeaways:

- **Never push with `--force` to public branches!**
- It's actually hard to permanently lose some of your work.
- Git is cool ♡

Questions?

That's it. Thank you!

Resources & further reading

- Rachel M. Carmena, *How to teach Git*
  `https://rachelcarmena.github.io/2018/12/12/how-to-teach-git.html`
- Nico Riedmann *Learn git concepts, not commands*
  `https://github.com/UnseenWizzard/git_training`
- `https://learngitbranching.js.org/` — an interactive browser "game" to learn Git concepts ♡