

Course: PHP from scratch

by Sergey Podgornyy

git basics



About me



Sergey Podgorny

Full-Stack Web Developer



and



Overview

- What is git?
- How to install git?
- Initialize repository
- Add remote repository
- Statuses and commits
- Push and Pull
- Branches
- Merging branches
- Logging changes
- Blame and Stash
- GitHub

What is git?

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



How to install git?



Download from official website - <https://git-scm.com/downloads>

Or via terminal

```
sudo apt-get install git (GNU/Linux)  
brew install git (MacOS)
```

Global variables

List existing variables

```
git config --list
```

Set variables

```
git config --global user.name "FirstName LastName"
```

```
git config --global user.email "mail@example.com"
```

Initialize new repo

```
git init
```

This creates a new subdirectory named `.git` that contains all of your necessary repository files – a Git repository skeleton. At this point, nothing in your project is tracked yet

Cloning repositories

```
git clone <path_to_remote_repo> [<folder_name>]
```

That creates a directory for project, initializes a `.git` directory inside it, pulls down all the data for that repository, and checks out a working copy of the latest version

Working with Remotes

List remote repositories

```
git remote -v
```

Adding Remote Repositories

```
git remote add <shortname> <url>
```

Inspecting a Remote

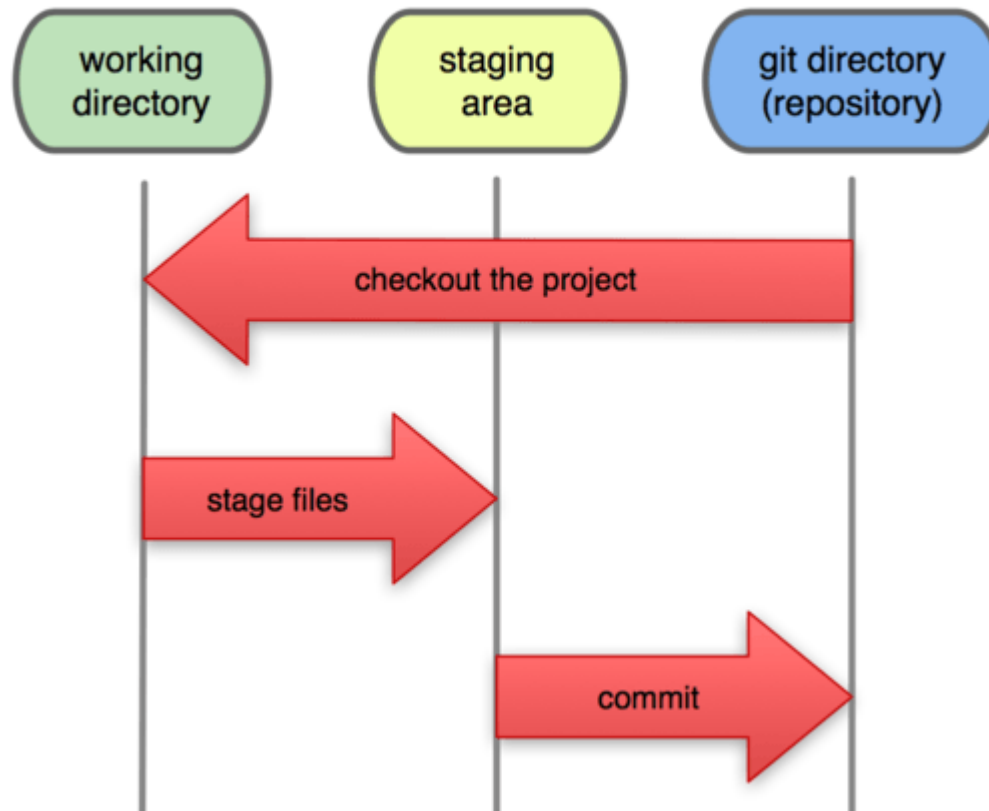
```
git remote show [remote-name]
```

Removing and Renaming Remotes

```
git remote rename origin default  
git remote remove default
```

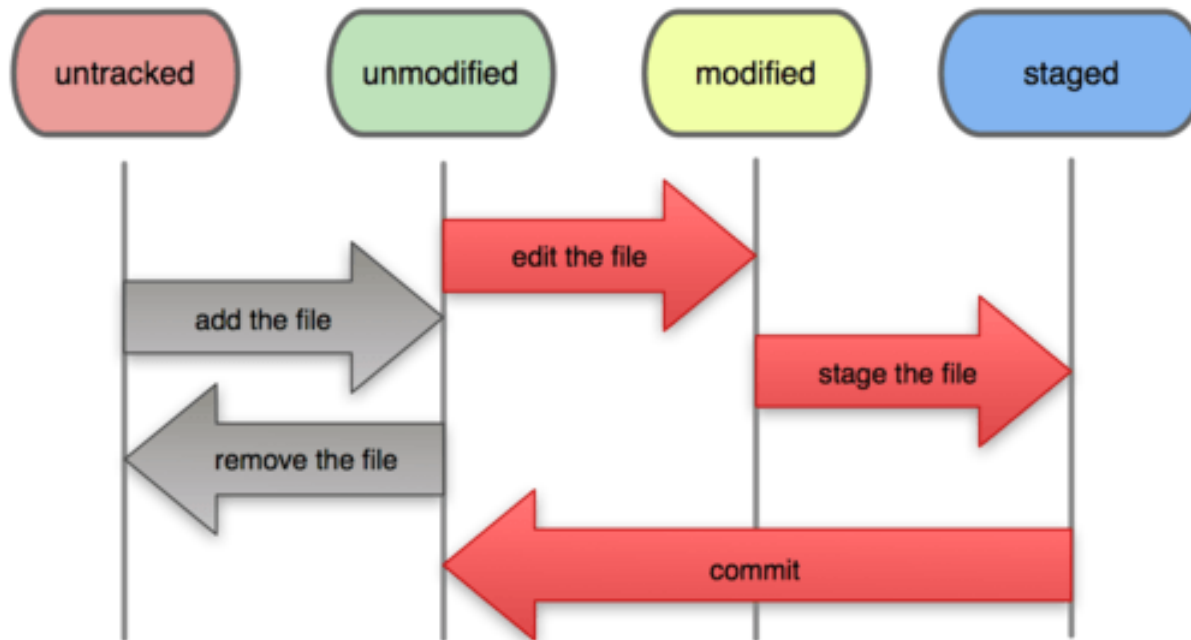
Statuses

Local Operations



Statuses

File Status Lifecycle



`git status`

Recording Changes to the Repository

Add files to staging

```
git add [<filename> | --all | -A | .]
```

Remove files from staging

```
git reset [<filename> | --all | -A | .]
```

Ignoring files

Add them to .gitignore file

To see what you've changed but not yet staged:

```
git diff
```

Committing Changes

The simplest way to commit is to type:

```
git commit
```

Alternatively, you can type your commit message inline with the commit command by specifying it after a -m flag

```
git commit -m "Some commit message"
```

Removing Files

```
git rm <filename> [--cached]
```

to keep the file on your hard drive but not have Git track, use the --cached option

Rolling back

Rolling back untracked files to previous commit

```
git checkout [<filename> | .]
```

Hard reset to specific commit

```
git reset --hard <commit-hash>
```

Potentially dangerous command, since it throws away all your uncommitted changes

Soft reset to specific commit

```
git reset --soft <commit-hash>
```

Roll back to specific commit without throws away files, so it only deletes commits history

Pushing and Pulling

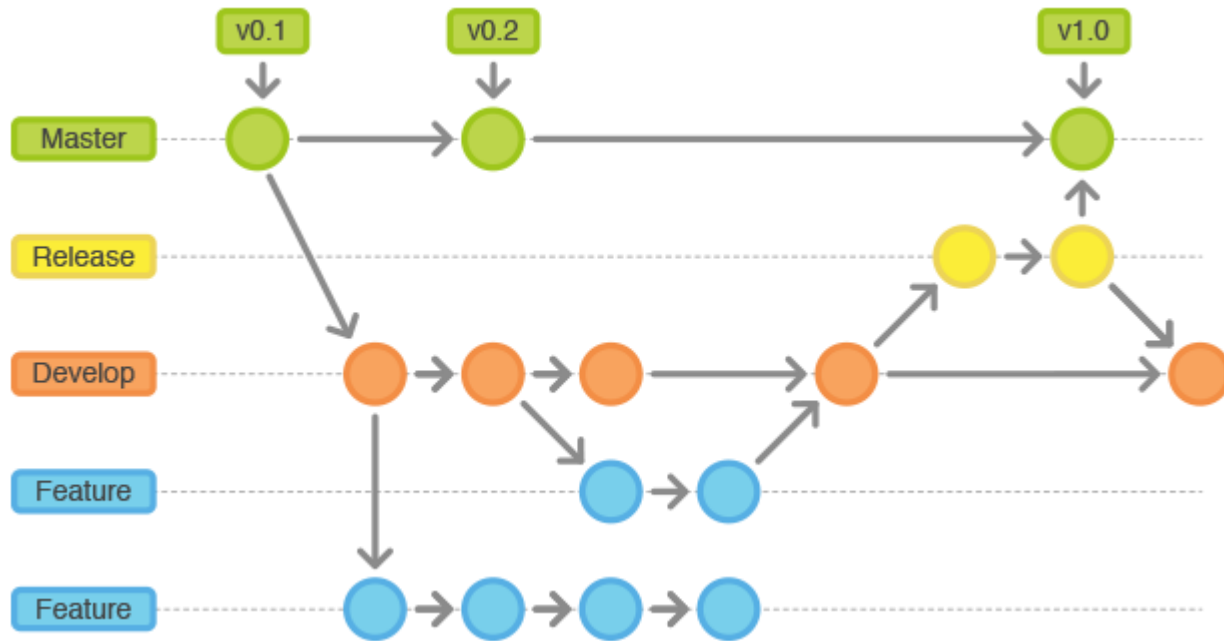
Pushing

```
git push <remote> <branch>
```

Pulling

```
git pull <remote> <branch>
```

Branches



List all branches

```
git branch
```


Working with branches

To switch to branch, type:

```
git checkout <branch_name>
```

To create new branch, type:

```
git branch <branch_name>
```

To switch to a new branch, type:

```
git checkout -b <branch_name>
```

To delete branch, type:

```
git branch -D <branch_name>
```

Merging and resolving conflicts

All you have to do is check out the branch you wish to merge into and then run the “git merge” command

```
git merge <source_branch>
```

Anything that has merge conflicts and hasn't been resolved is listed as unmerged. Git adds standard conflict-resolution markers to the files that have conflicts, so you can open them manually and resolve those conflicts

```
<<<<<< HEAD:index.html
<div id="footer">contact : email.support@github.com</div>
=====
<div id="footer">please contact us at support@github.com</div>
>>>>>> iss53:index.html
```

Viewing the Commit History

`git log` lists the commits made in that repository in reverse chronological order – that is, the most recent commits show up first

```
git log
```

One of the more helpful options is `-p`, which shows the difference introduced in each commit. You can also use `-<num>`, which limits the output

```
git log -p -2
```

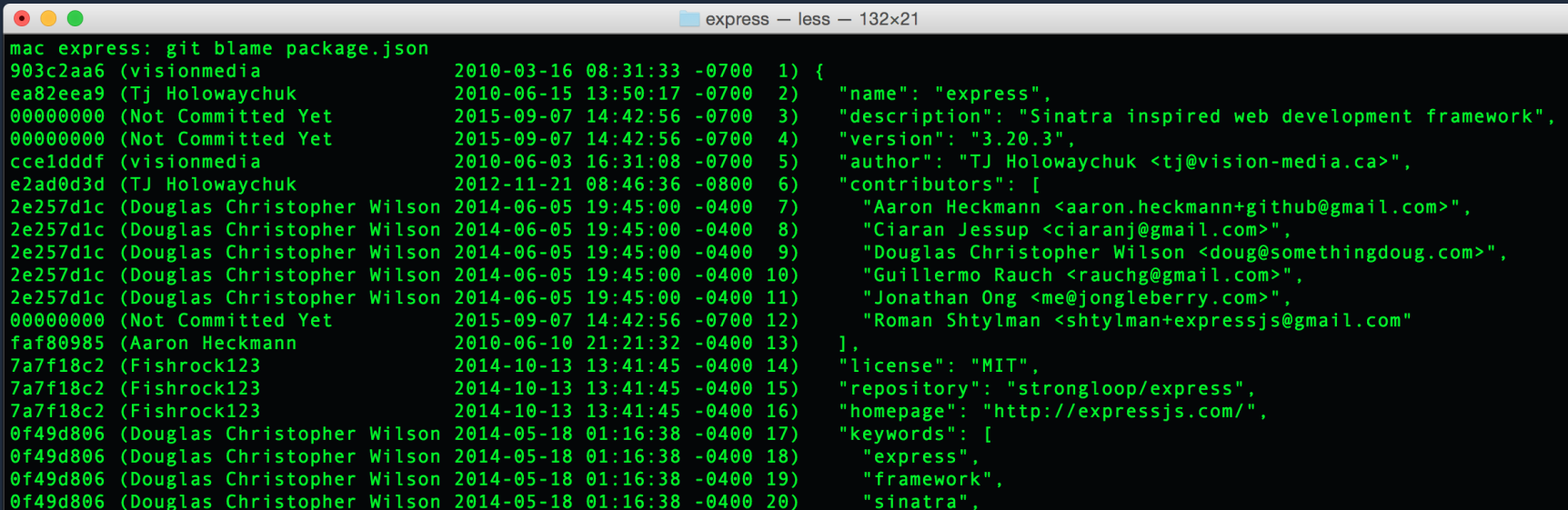
pretty format

```
git log --pretty=format:"%h - %an, %ar : %s" --graph
```

Blame

If you want to blame someone for some code, you can check who was the last who modify the string

```
git blame <file_name>
```



```
mac express: git blame package.json
903c2aa6 (visionmedia) 2010-03-16 08:31:33 -0700 1) {
ea82eea9 (Tj Holowaychuk) 2010-06-15 13:50:17 -0700 2)   "name": "express",
00000000 (Not Committed Yet) 2015-09-07 14:42:56 -0700 3)   "description": "Sinatra inspired web development framework",
00000000 (Not Committed Yet) 2015-09-07 14:42:56 -0700 4)   "version": "3.20.3",
cce1dddf (visionmedia) 2010-06-03 16:31:08 -0700 5)   "author": "TJ Holowaychuk <tj@vision-media.ca>",
e2ad0d3d (TJ Holowaychuk) 2012-11-21 08:46:36 -0800 6)   "contributors": [
2e257d1c (Douglas Christopher Wilson) 2014-06-05 19:45:00 -0400 7)     "Aaron Heckmann <aaron.heckmann+github@gmail.com>",
2e257d1c (Douglas Christopher Wilson) 2014-06-05 19:45:00 -0400 8)     "Ciaran Jessup <ciaranj@gmail.com>",
2e257d1c (Douglas Christopher Wilson) 2014-06-05 19:45:00 -0400 9)     "Douglas Christopher Wilson <doug@somethingdoug.com>",
2e257d1c (Douglas Christopher Wilson) 2014-06-05 19:45:00 -0400 10)    "Guillermo Rauch <rauchg@gmail.com>",
2e257d1c (Douglas Christopher Wilson) 2014-06-05 19:45:00 -0400 11)    "Jonathan Ong <me@jongleberry.com>",
00000000 (Not Committed Yet) 2015-09-07 14:42:56 -0700 12)    "Roman Shtylman <shtylman+expressjs@gmail.com"
faf80985 (Aaron Heckmann) 2010-06-10 21:21:32 -0400 13)  ],
7a7f18c2 (Fishrock123) 2014-10-13 13:41:45 -0400 14)   "license": "MIT",
7a7f18c2 (Fishrock123) 2014-10-13 13:41:45 -0400 15)   "repository": "strongloop/express",
7a7f18c2 (Fishrock123) 2014-10-13 13:41:45 -0400 16)   "homepage": "http://expressjs.com/",
0f49d806 (Douglas Christopher Wilson) 2014-05-18 01:16:38 -0400 17)   "keywords": [
0f49d806 (Douglas Christopher Wilson) 2014-05-18 01:16:38 -0400 18)     "express",
0f49d806 (Douglas Christopher Wilson) 2014-05-18 01:16:38 -0400 19)     "framework",
0f49d806 (Douglas Christopher Wilson) 2014-05-18 01:16:38 -0400 20)     "sinatra",
```

Stashing changes

Push changes to temporary storage

```
git status      # check if uncommitted files exists
git stash       # stash this files
git status      # check that there is no more uncommitted files
git stash list  # See list of stashed data
git stash pop    # Remove a single stashed state from the stash
list and apply it on top of the current working tree state
git stash clear          # Remove all the stashed states
git stash drop <stash>  # Remove a single stashed state from
the stash list
git stash apply          # Apply files back from stash (do not
remove the state from the stash list)
git stash apply --index # Apply files back from stash with
saving index relations
```

Squashing commits

With `git` it's possible to squash previous commits into one. This is a great way to group certain changes together before sharing them with others

The command to accomplish that is:

```
git rebase -i HEAD~n
```

This will open up your editor. Replace “pick” with “squash” behind commit message you want to squash

```
pick f392171 Added new feature X  
squash ba9dd9a Added new elements to page design  
squash df71a27 Updated CSS for new elements
```

When done, save and quit your editor. Git will now squash the commits into one. All done!

Change commit message

If commit has not been pushed online

```
git commit --amend
```

Amending the message of older or multiple commit messages

- Use the `git rebase -i HEAD~n` command to display a list of the last `n` commits in your default text editor
- Replace `pick` with `reword` before each commit message you want to change
- Save and close the commit list file
- In each resulting commit file, type the new commit message, save the file, and close it
- Force-push the amended commits `git push --force`

GitHub

GitHub



GitHub is a web-based **Git** or version control repository, that offers all of the distributed version control and source code management (SCM) functionality

Useful resources

- [ОСНОВЫ git \(RU\)](#)
- [Git basics \(EN\)](#)
- [Download git](#)
- [Git tutorials](#)
- [Git Branching Rebasing](#)
- [Reset Demystified](#)

Thanks for your attention

Q & A

