Deel 1

VOORMIDDAG

Git basics

2016

Version control

INTRODUCTION

Why Source Control?

How do two people collaborate on the same code?

- Email?
- Dropbox?

Why Source Control?

How do two people collaborate on the same code?

- Email?
- Dropbox?

How do two hundred people collaborate?

Why Source Control?

How do we backup our code?

How do we get a history of changes?

Why Source Control?

How do we make big or risky code changes without affecting the *stable* version?

How do we work on new versions and still support old versions?

What is Version/Source Control?

Manages file sharing for
Concurrent Development

Keeps track of changes with
Version Control

Popular version control systems

SubVersion (SVN)

Git

Mercurial (Hg)

CVS

Bazaar

etc.

Concurrent Development

Server holds all original files of a project
Gives out copies to participants (clients)

Participants modify their copies
Submit their changes to server

Automatically merges changes into original files. Huge!

Conflicts only occur when modifications are done

- by more then one participant
- at the same location in their respective copies.
- Then participants have to manually resolve such conflicts. Rare!

Powerful edit and merge tools help make this task easy

Version Control

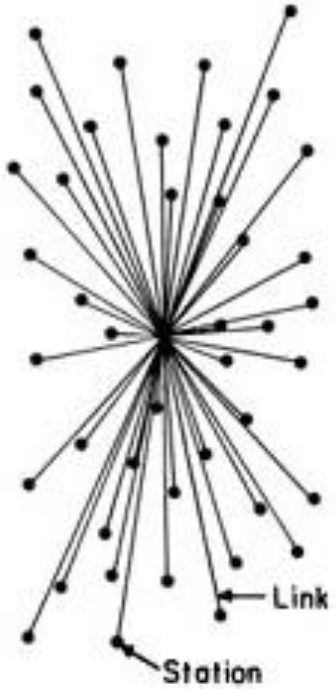
SVN/Git keeps log of any changes made to any file. Ever!
Also keeps copies of those changes. For ever!

Participants can go back and receive older versions of a file
or even an older version of an entire project state

The current version number of your project in SVN is #5
In the future you can always load the project exactly as it is today by requesting project version #5; et voila you can run an age old demo!

Distributed or centralized?

Most Version Control Systems



Linus' Vision of Git

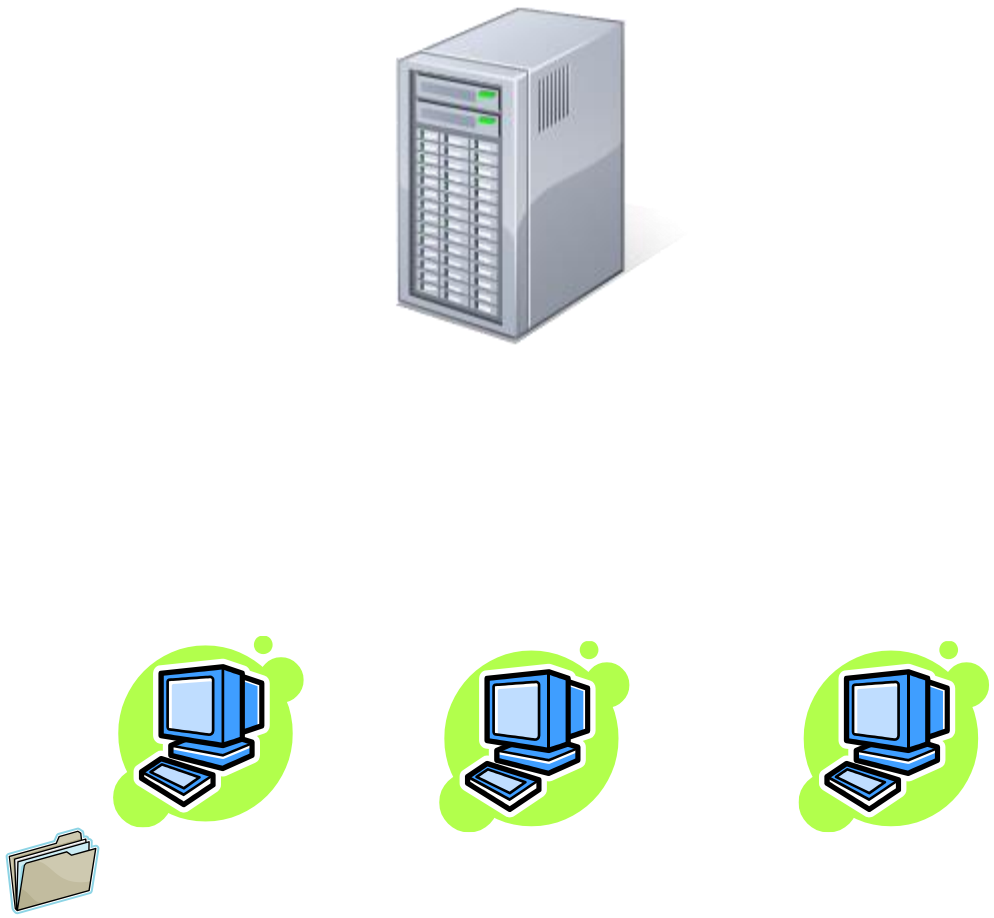
Reality of Distributed Systems

Centralized

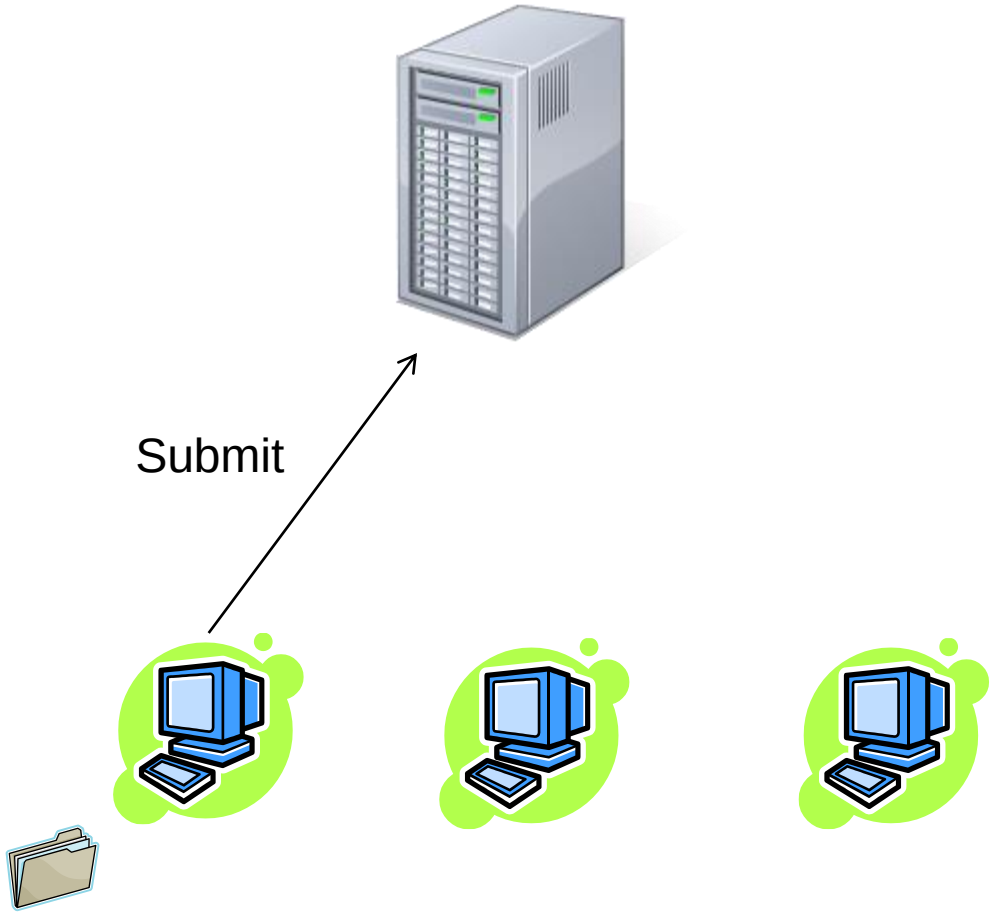
Server holds a centralized *repository*

Clients have a revision

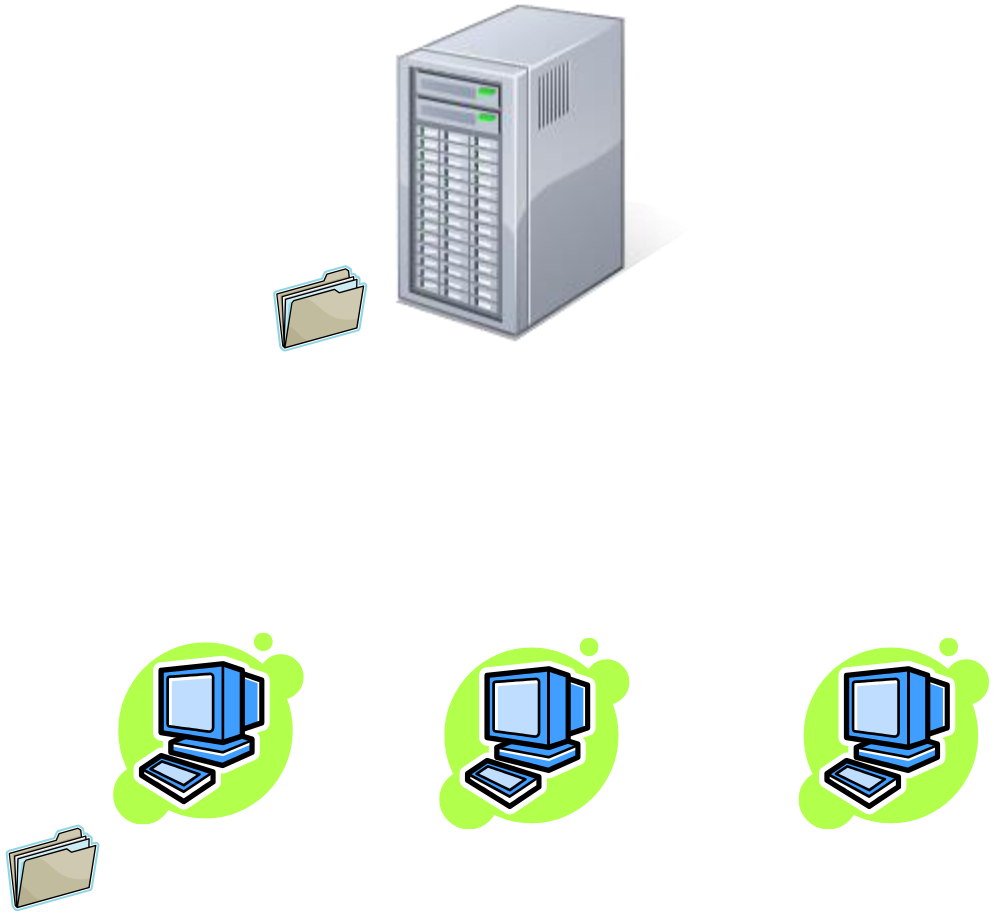
Centralized



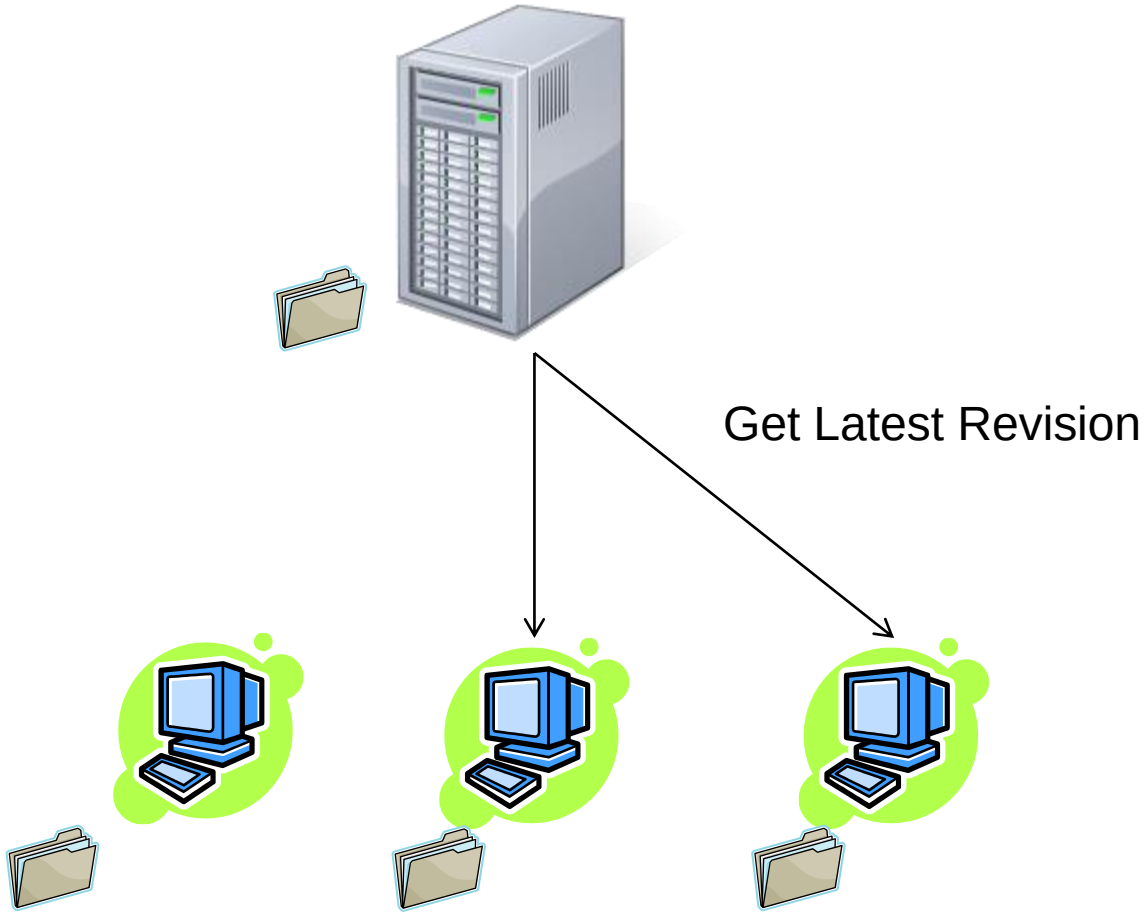
Centralized



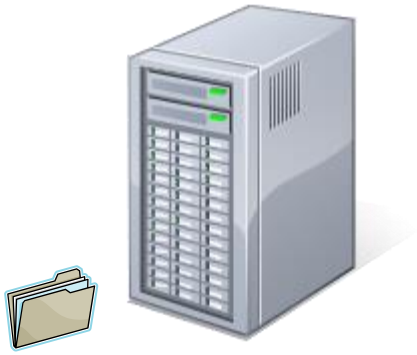
Centralized



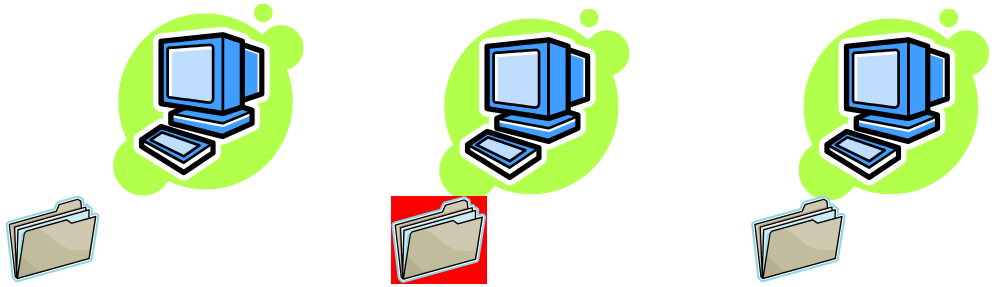
Centralized



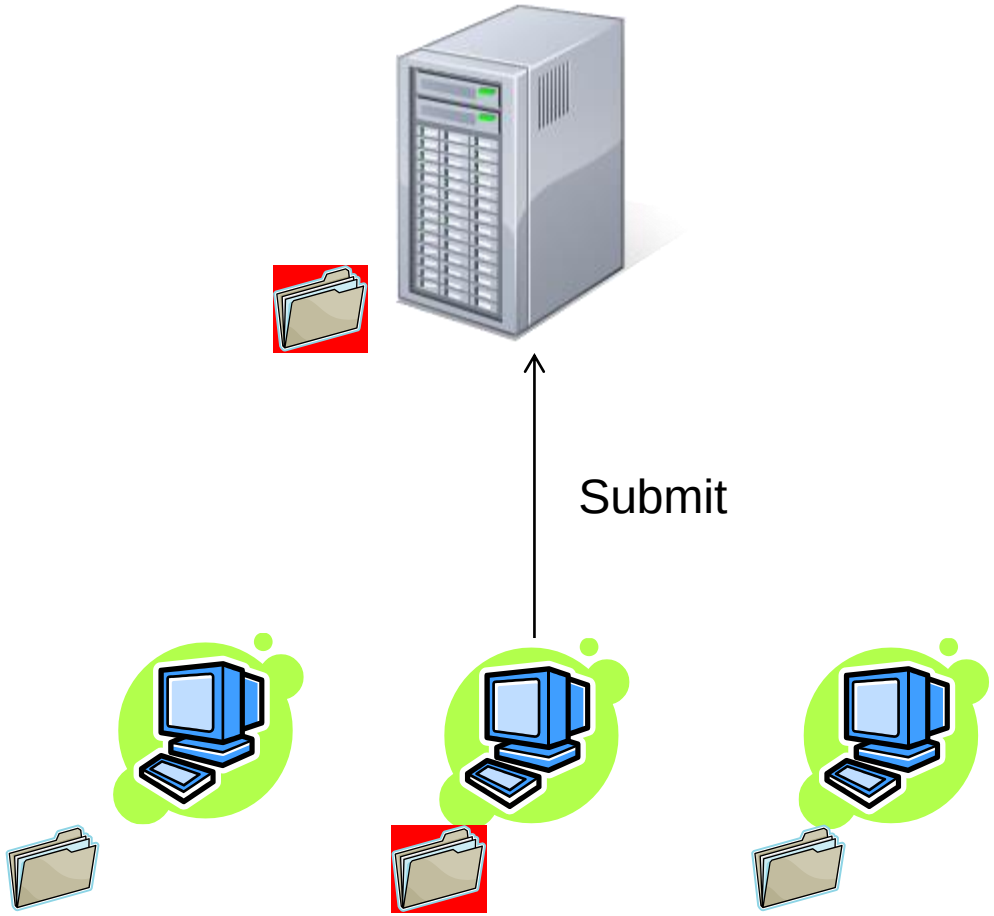
Centralized



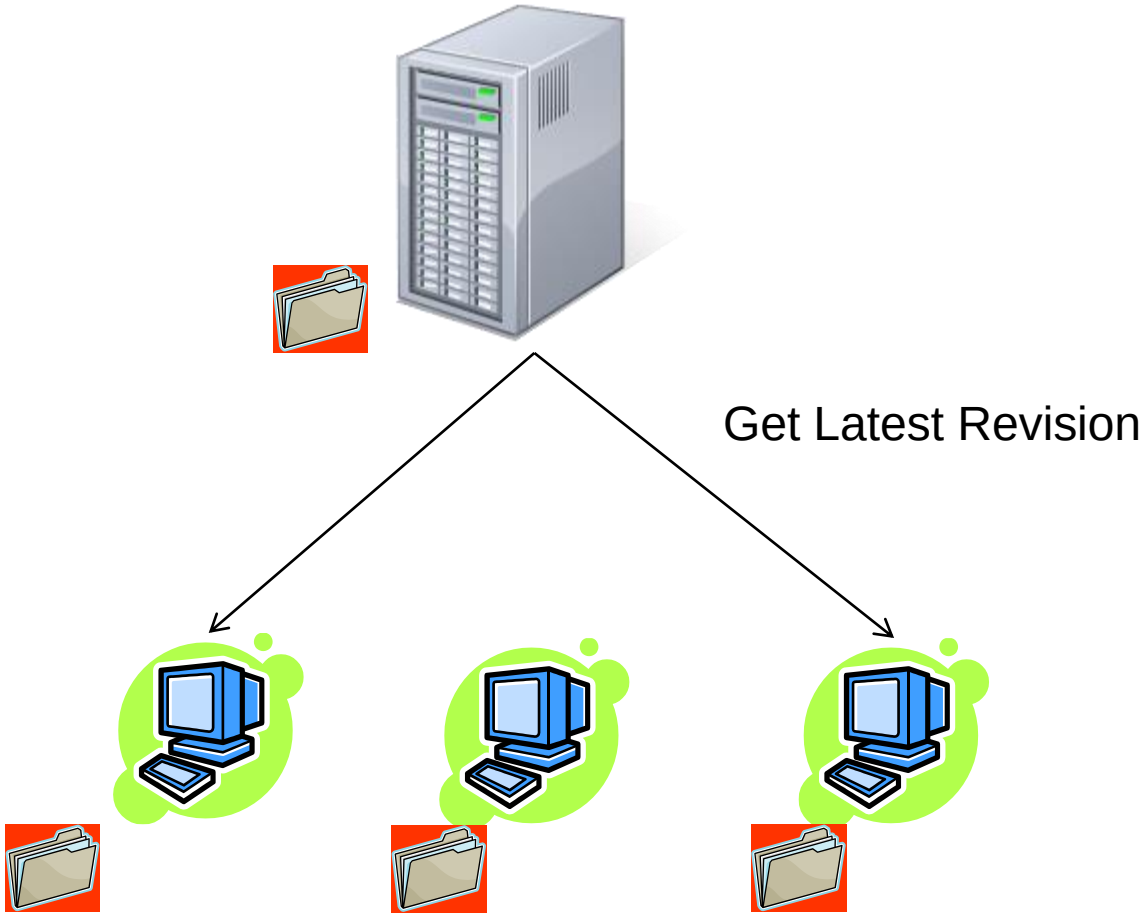
Modify/add/remove files



Centralized



Centralized

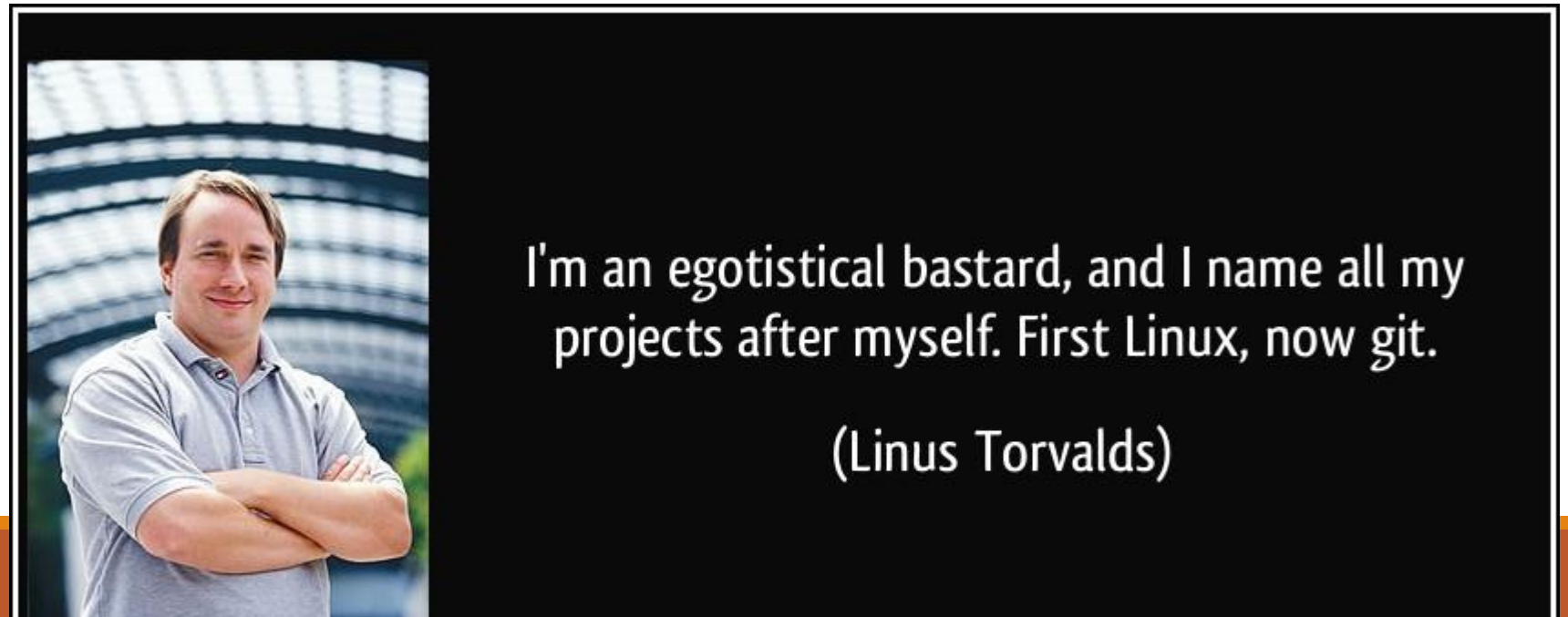


GIT

git

Git is a versioning control system:

- Allows you to have a ‘history’ of changes in your code, text or any other type of file
- Decentralized versioning (compared to older centralized way of versioning)
- Everyone working on a git project has a copy containing the full history of the project.



The End

When I say I hate CVS with a passion, I have to also say that if there are any SVN [Subversion] users in the audience, you might want to leave. Because my hatred of CVS has meant that I see Subversion as being the most pointless project ever started. The slogan of Subversion for a while was "CVS done right", or something like that, and if you start with that kind of slogan, there's nowhere you can go. There is no way to do CVS right.

--Linus Torvalds, as quoted in Wikipedia

Distributed

All machines have a full copy of the repository

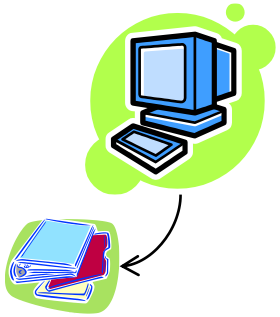
Repositories can be *cloned*

Repositories can be *pushed* to and *pulled* from other machines

Distributed

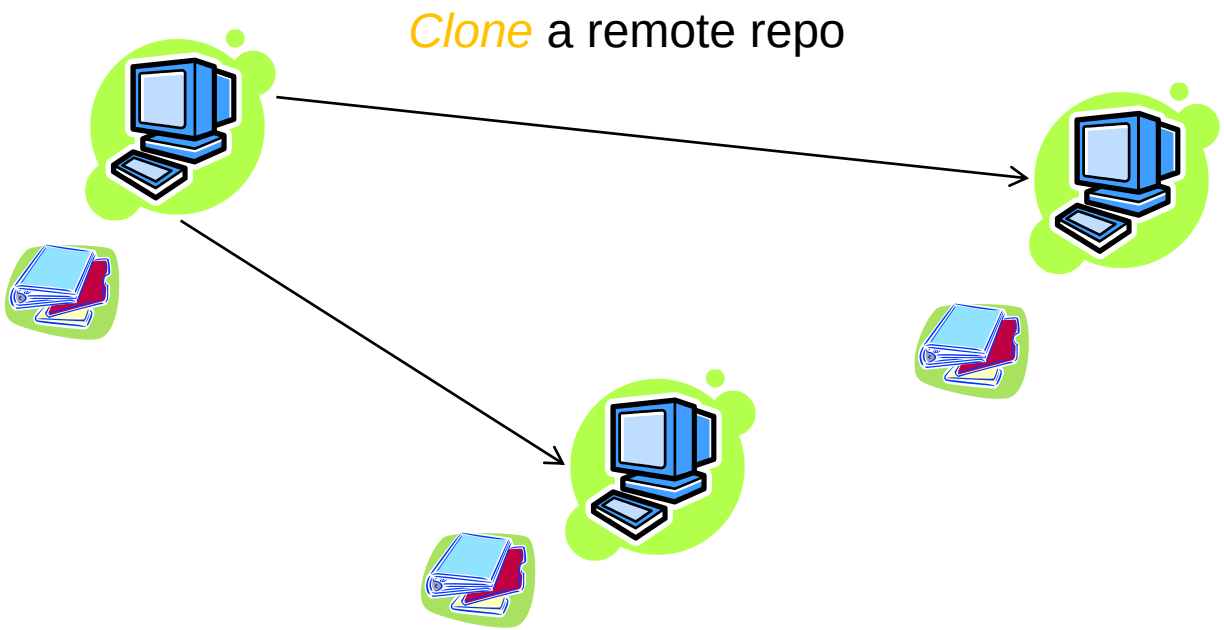
```
git init
// Add files to directory
git add *
git commit -a -m '<description>'
```

Create a local repo
Locally *commit* changes



Distributed

```
git clone git@github.com:CIS565-Spring-2012/cis565testHomework.git
```



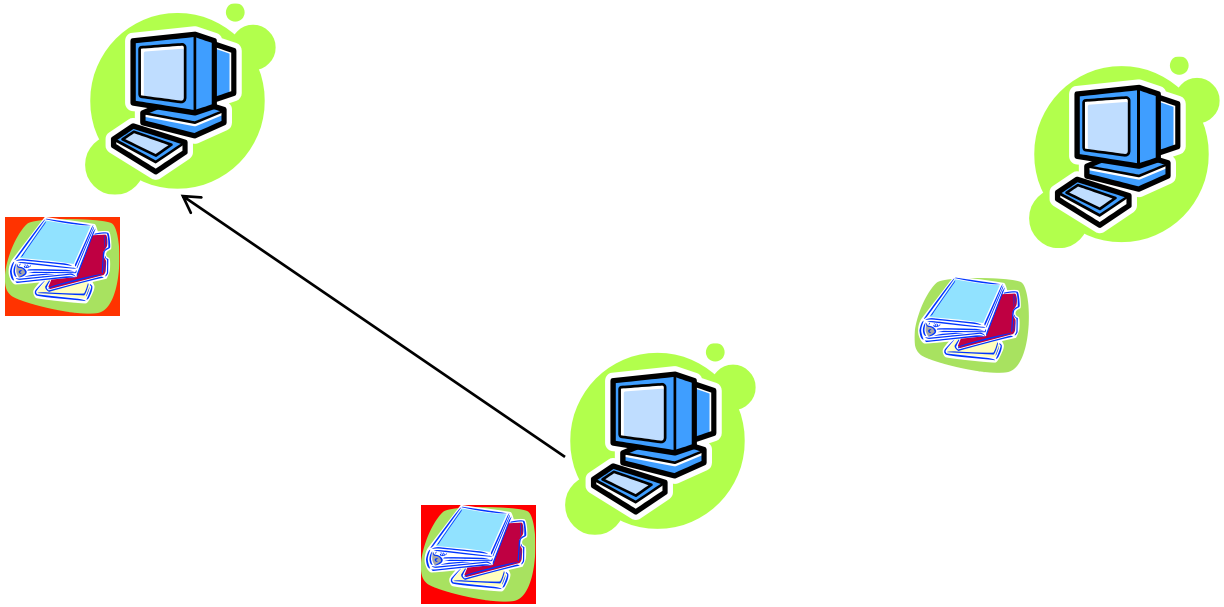
Distributed

```
git commit -a -m '<description>'
```

Locally *commit* changes

Distributed

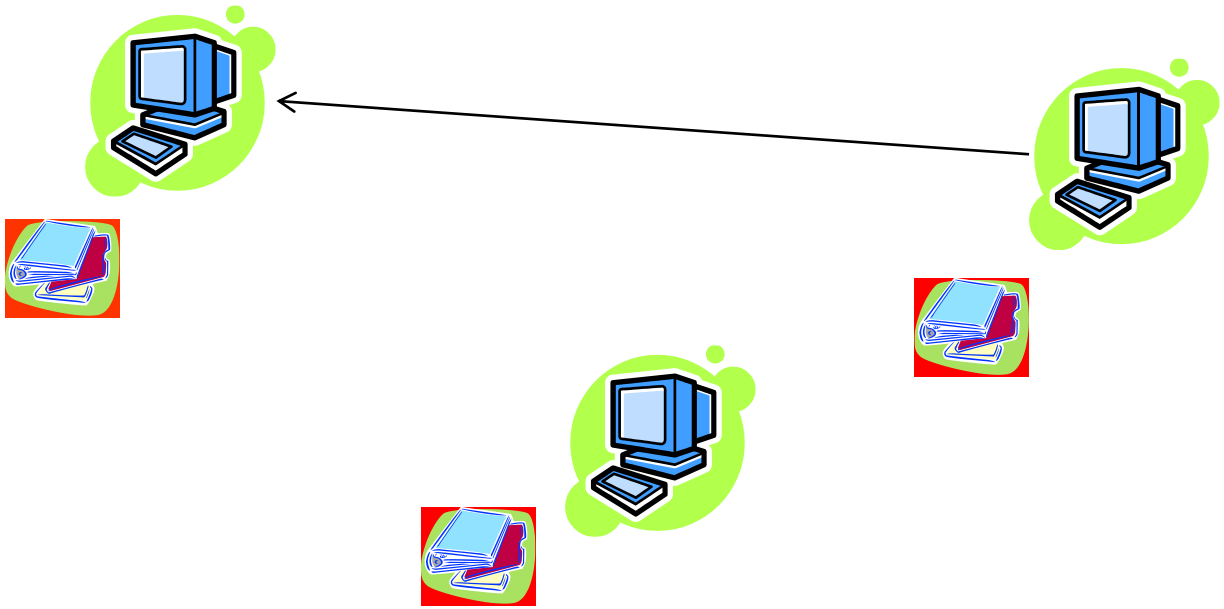
```
git push
```



Push changes to other repos

Distributed

```
git pull
```



Pull changes from other repos

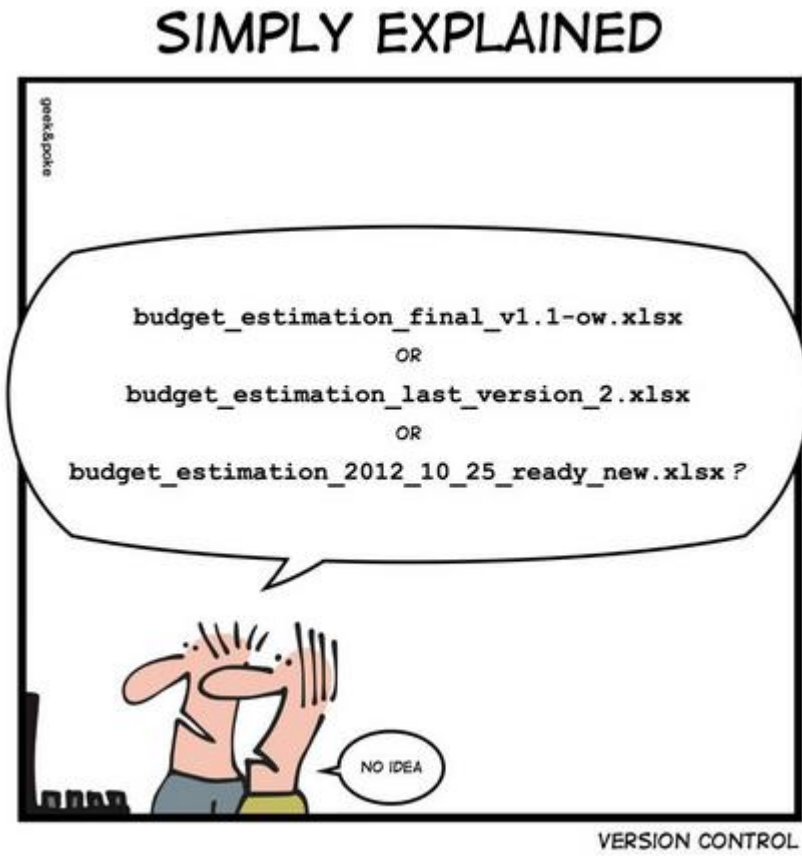
So why should I use git?!

So why should I use git?!

Even when **working alone** on a project!

- Undo changes ("lifelong ctrl-z") or simply revert back to older version
- Complete history of all changes: see how the project looked like on a certain day
  - Handy for bug fixes or when you forgot to write down your logs
- Documents why you changed something (by using commit information)

I use it for all my projects (coding, writing, blogging, parenting, etc)



So why should I use git?!

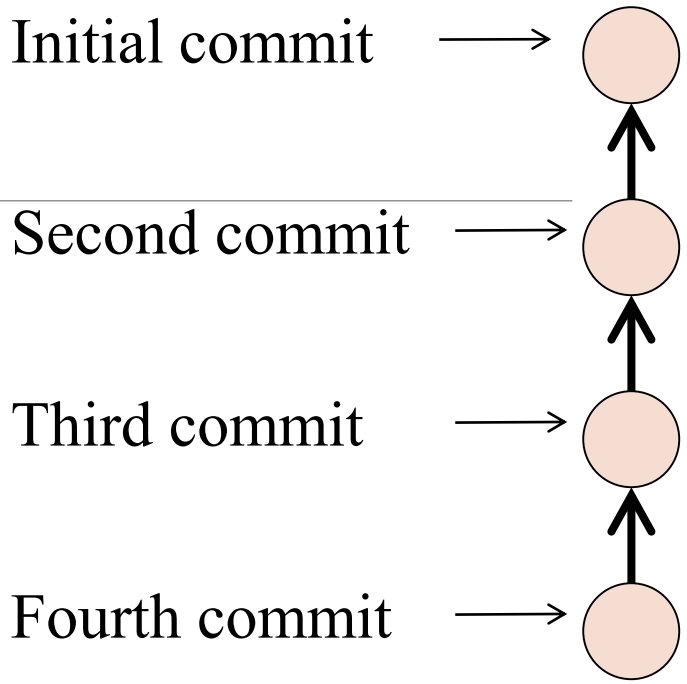
In team:

- **Solve conflicts** in code
- Work together on same files. Git will most of the time **merge** them automagically
- Work on different **branched, independent** of each other, for example: a branch per feature

Commits

A snapshot of your project at a given time

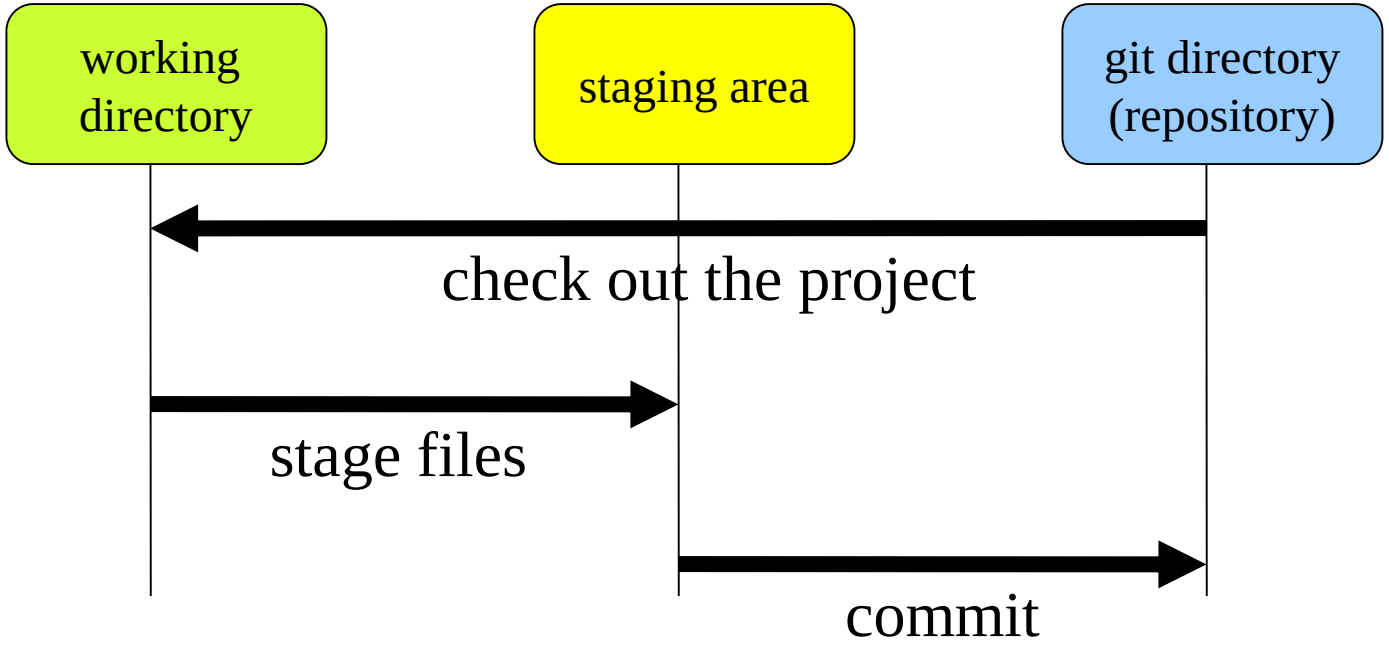
A commit save the changes relative to previous commit



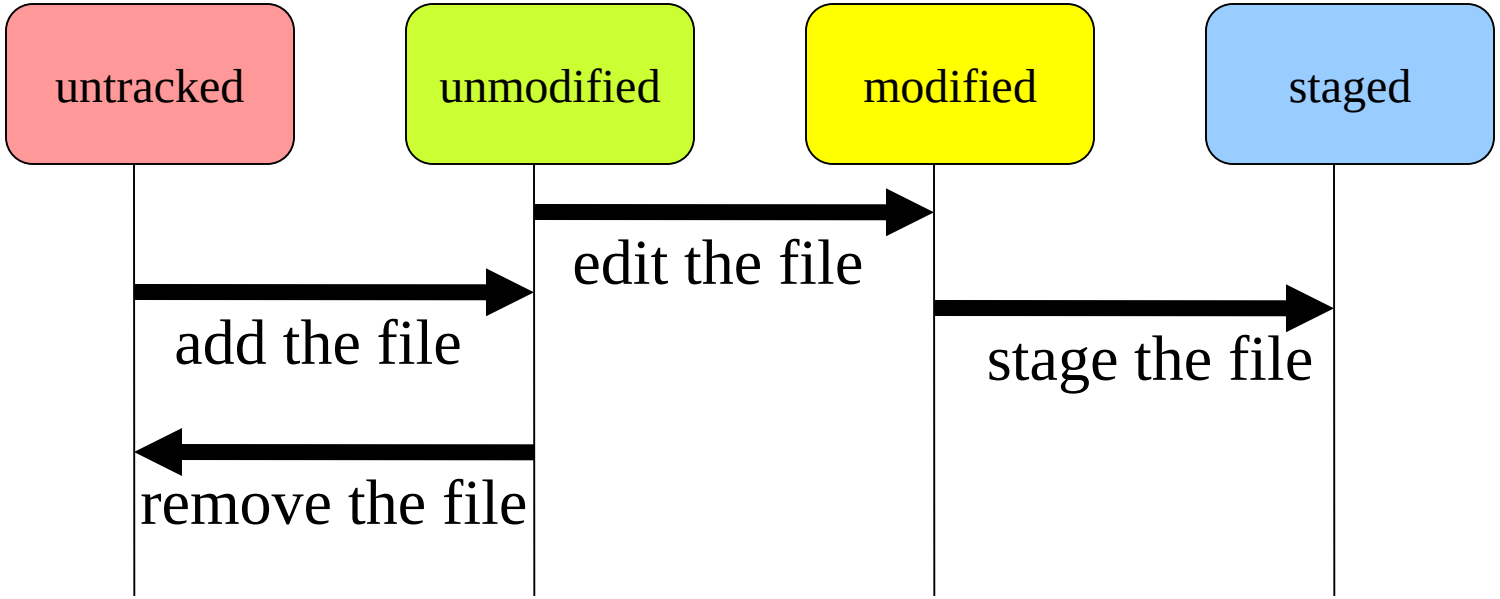
In a nutshell, you will use git add to start tracking new files and also to stage changes to already tracked files, then git status and git diff to see what has been modified and staged and finally git committo record your snapshot into your history. This will be the basic workflow that you use most of the time.

3 States of a File in Git

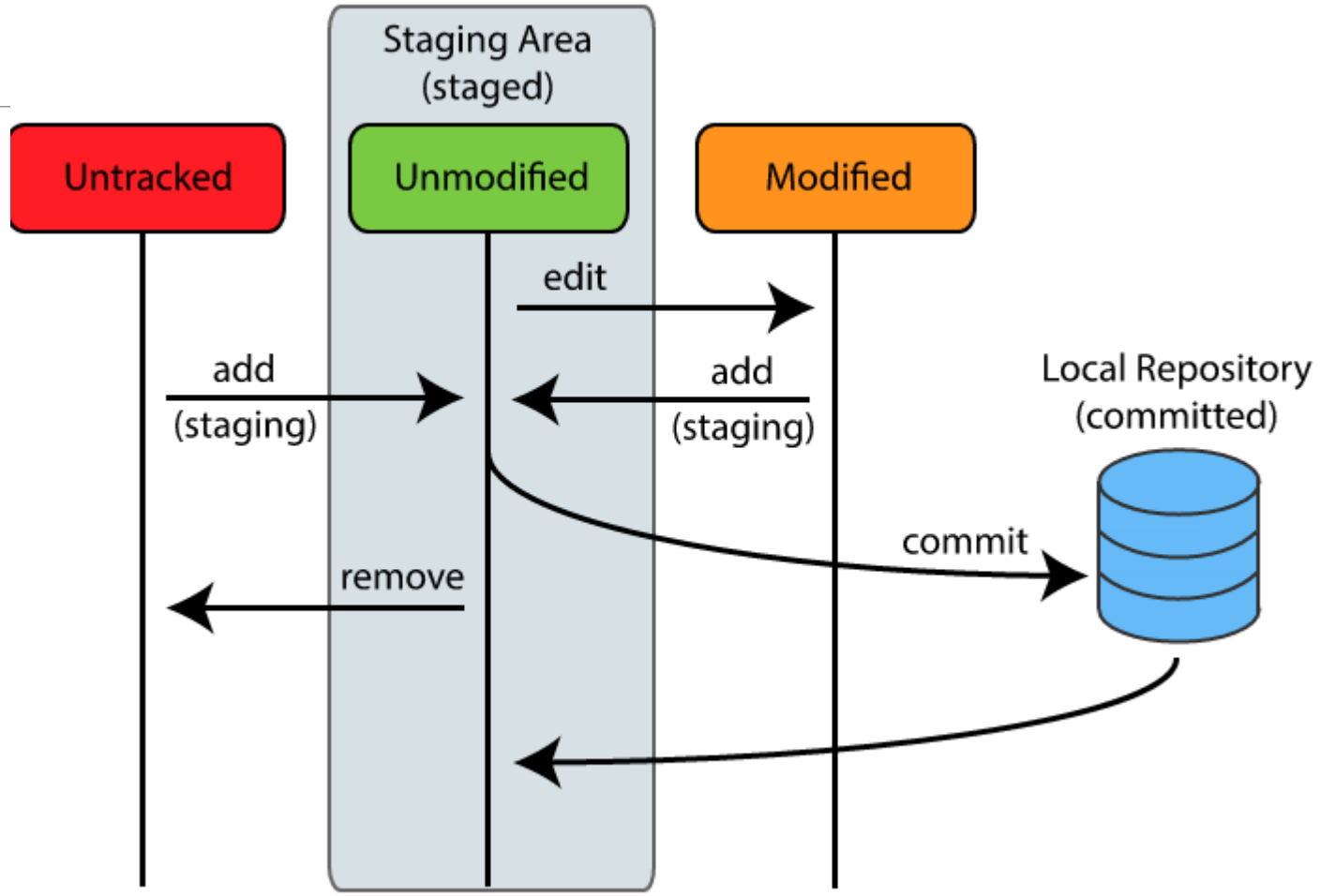
- **Modified** • **Staged** • **Committed**



File Status Lifecycle



All together:



Some Commands

Getting a Repository

- git init

Commits

- git add
- git commit

Getting information

- git help
- git status
- git diff
- git log
- git show

Undoing What is Done

git checkout

- Used to checkout a specific version/branch of the tree

git reset

- Moves the tree back to a certain specified version
- Use the --force to ignore working changes

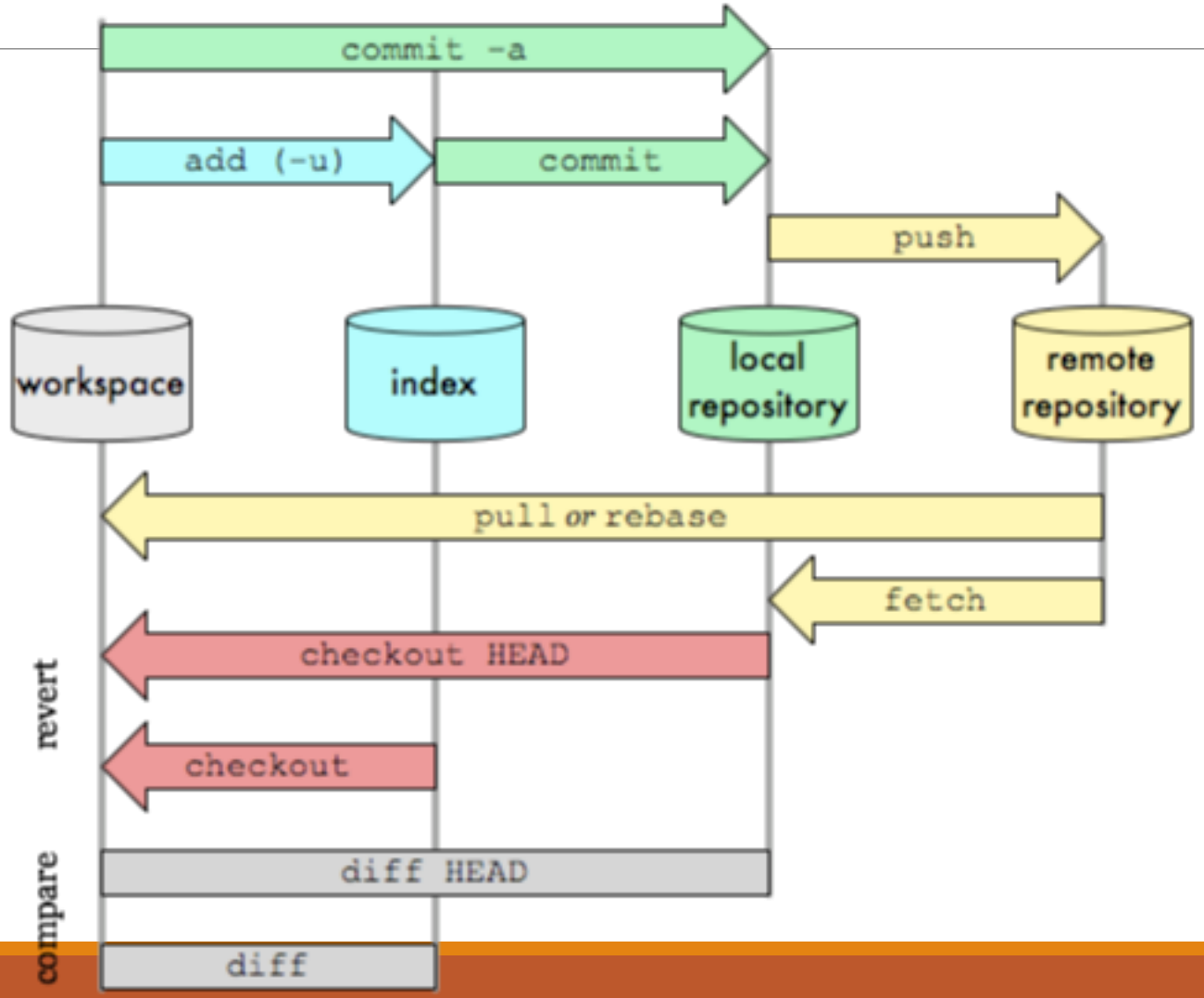
git revert

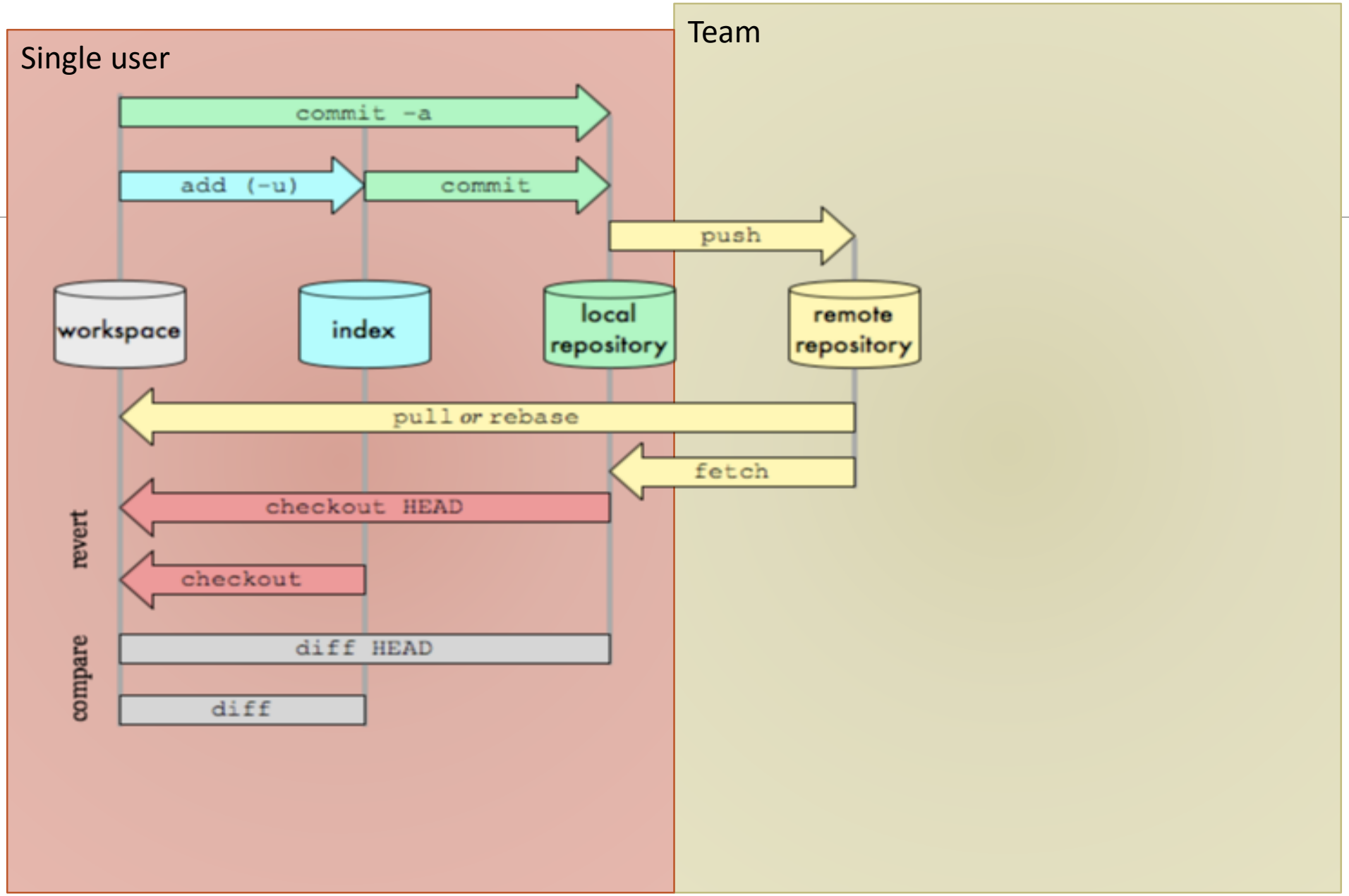
- Reverts a commit
- Does not delete the commit object, just applies a patch
- Reverts can themselves be reverted!

Git never deletes a commit object

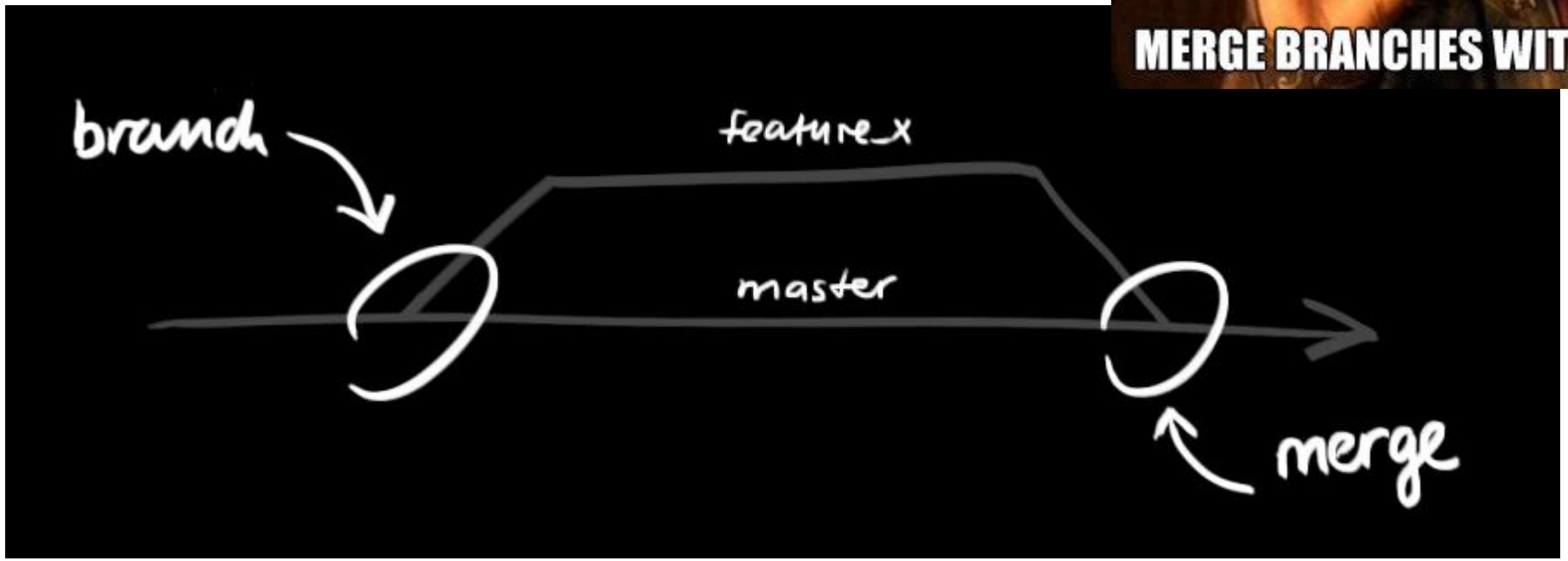
- It is very hard to shoot yourself in the foot!

A simple Git workflow

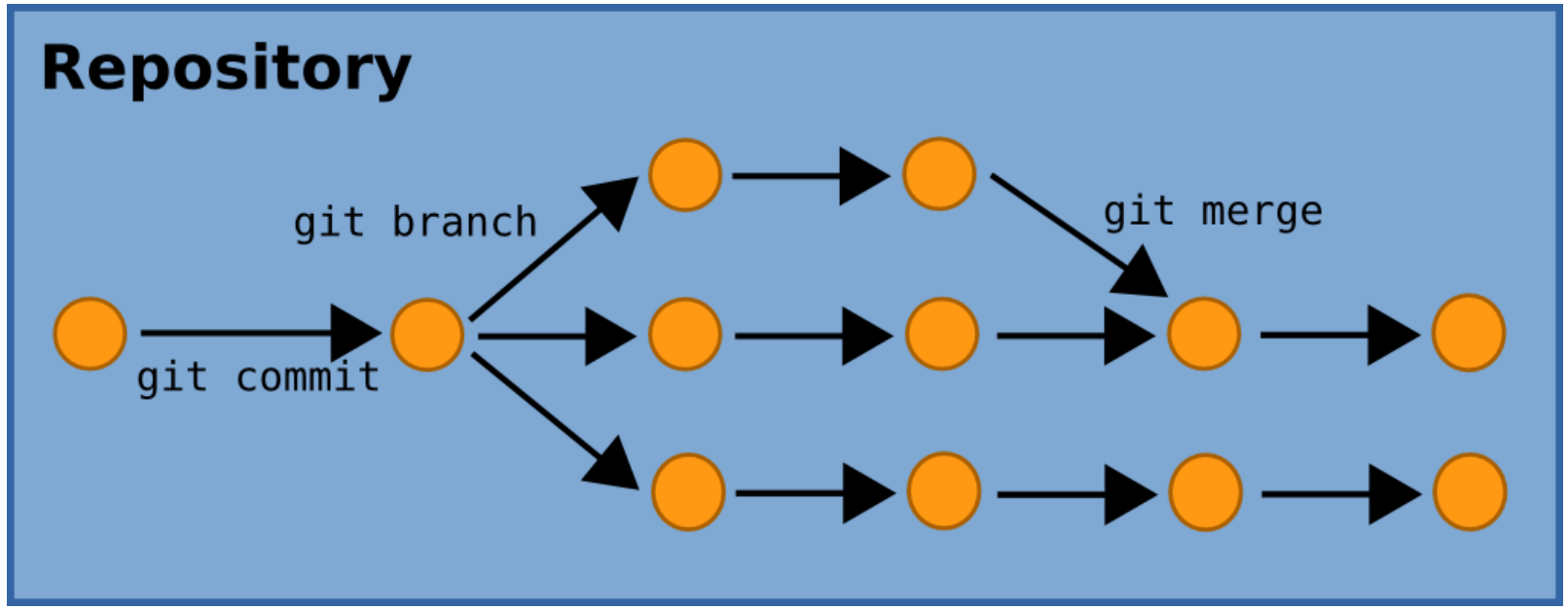



Single user
Team
commit -a
add (-u)
commit
push
workspace
index
local repository
remote repository
pull or rebase
fetch
checkout HEAD
checkout
revert
diff HEAD
diff
compare

Branches and merges



Branching and merging



Useful tools?

Needs a git-client:

- Original Git-client
  - Cli & simple UI
  - https://git-scm.com/downloads
- SmartGit
  - my preferred client, bit more complex
  - http://www.syntevo.com/smartgit/
- Github Desktop
  - simple, but too much automagic for my taste
  - https://desktop.github.com/
- SourceTree
  - Very popular and robust client .

My advice: Get familiar using Github Desktop and then try Original Git client or smartGit later on

Now what?

https://help.github.com/articles/good-resources-for-learning-git-and-github/

Learn yourself: http://try.github.io/levels/1/challenges/1

Great tutorial: http://marklodato.github.io/visual-git-guide/index-en.html

More early start info:

- https://help.github.com/
- http://rogerdudler.github.com/git-guide/
- http://gitimmersion.com
- https://guides.github.com/activities/hello-world/

Basic git commands

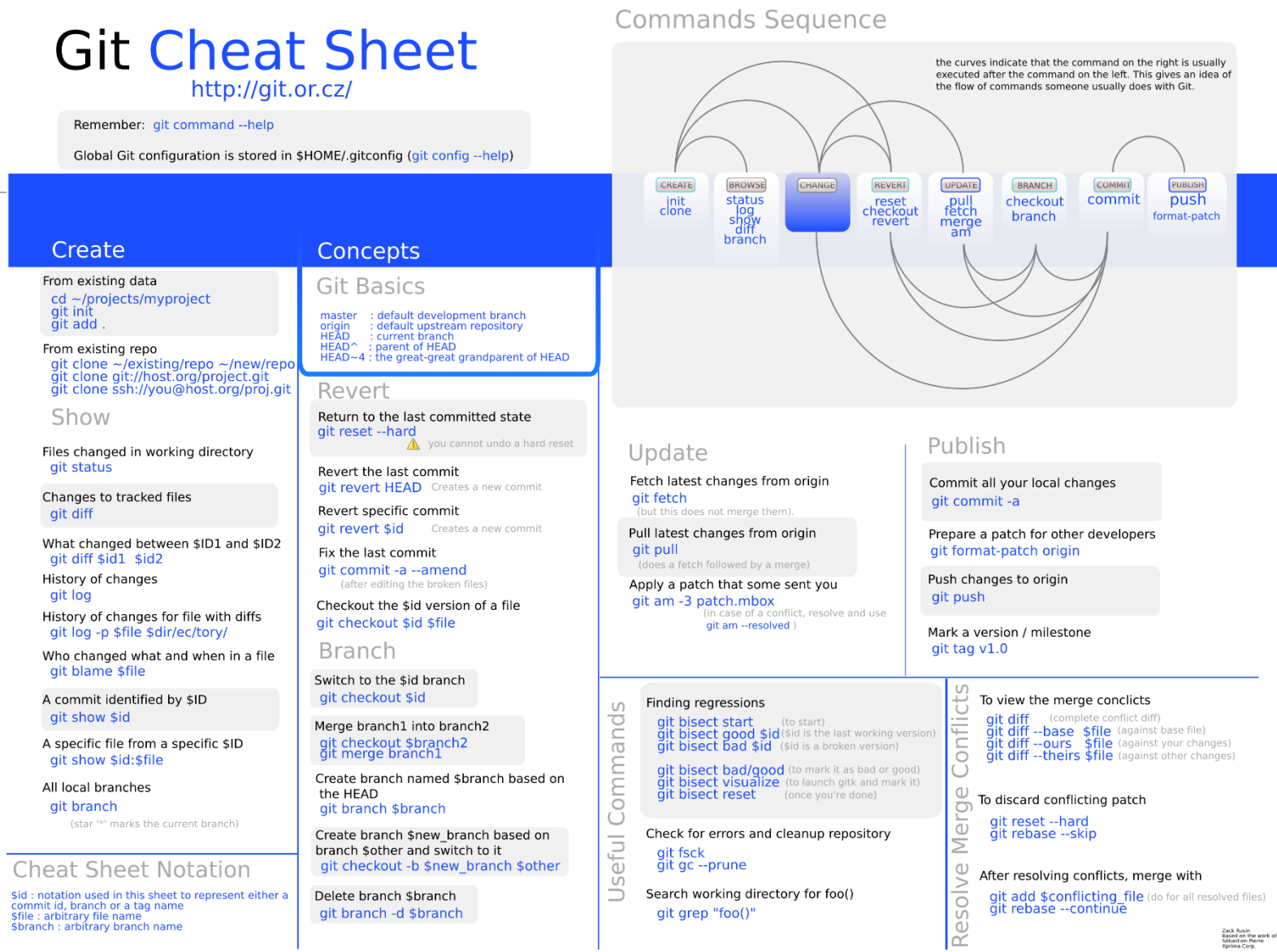
OVERVIEW

Getting started

Install git: https://git-scm.com/downloads

All commands

Good cheatsheet: https://hallcweb.jlab.org/wiki/images/1/1f/Git-cheat-sheet-large.png



Basic Commands - git

git – view all commands

```
$ git
```

Basic Commands - Init

Init - create an empty new repository

```
$ git init
```

Basic Commands - Status

Status - show differences between what has been committed and HEAD

$ git status

Basic Commands - Add

Add – add files to the stage

```
$ git add foo.info
```

Basic Commands - Commit

Commit – stores contents of the index in a commit along with a message

$ git commit –m “Added foo.info”

Basic Commands - Log

Log– view previous commits

$ git log

Basic Commands - Checkout

Checkout = checkout branches or previous commits

```
$ git checkout coolfeaturebranch
```

```
$ git checkout 1c899fed6ed
```

Exercise

Practice

```
$ mkdir test
$ cd test
$ git init .
$ git status
```

1. Create a new directory
2. Move inside the new directory
3. Initialize the new directory as a git repository
4. Show the current status of the repository

Practice

$ notepad hello.txt

Type “Hello World” then save and exit

$ git status

git shows hello.txt as “untracked”

Untracked files are files which are in the current directory but *are not under version control!*

Practice

```
$ git add hello.txt
$ git commit -m “Add hello.txt”
$ git status
$ git log
```

1. Add hello.txt to version control
2. Commit changes in current repository (-m tells git to save a “commit message” with this commit)
3. Show status of repository
4. Show the commit log (a sort of history)

Practice: braching & merging

```
$ git branch goodbye
$ git checkout goodbye
$ git status
```

1. Create a new branch called “goodbye”
2. Switch to the new branch
3. Print out the status of the repository on the current branch. Note the [goodbye] in the top left... we are on the “goodbye” branch!

Practice: braching & merging

```
$ notepad goodbye.txt
$ git add goodbye.txt
$ git commit -m “Add goodbye.txt”
$ git checkout master
$ git merge goodbye
```

1. Put text in goodbye.txt and then save and exit
2. Add goodbye.txt to version control
3. Commit changes to the current repository
4. Switch back to the master branch
5. “Merge” the changes from the “goodbye” branch into the current branch.

Deel 2

NAMIDDAG

.gitignore

Good practice to first add a .gitignore file.

- Lists files, extensions to ignore
- E.g. build files

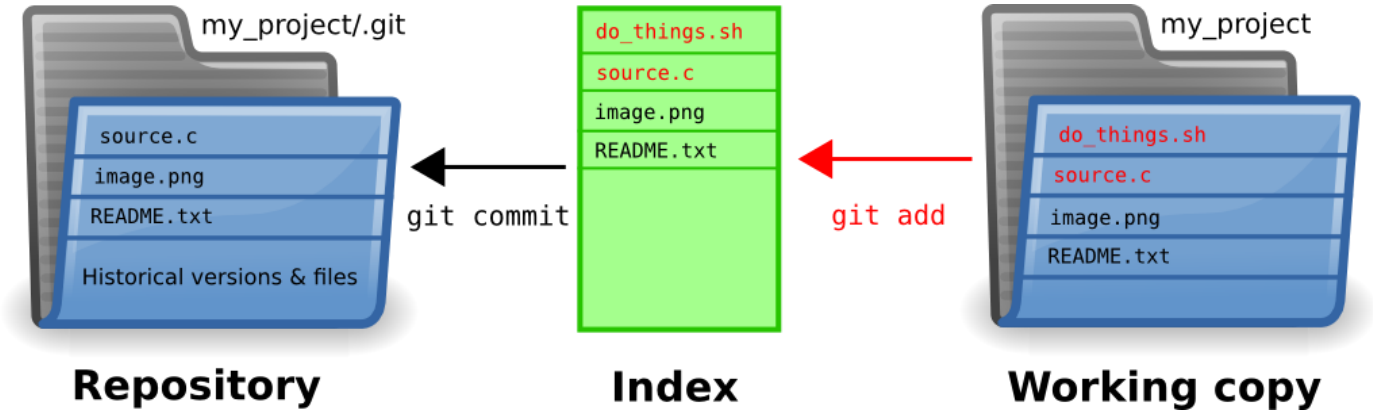
https://github.com/github/gitignore

So way to go:

1. Create new repo (*git init*)
2. Add correct .gitignore files (*git add .gitignore*)
3. Commit changes (*git commit –m "Let's start, .gitignore added"*)
4. Start working

DUS

1. Create new repo (*git init*)
2. Add correct .gitignore files (*git add .gitignore*)
3. Commit changes (*git commit –m "Let's start, .gitignore added"*)
4. Start working



Practice

“Amend” the last commit, telling git who you are:

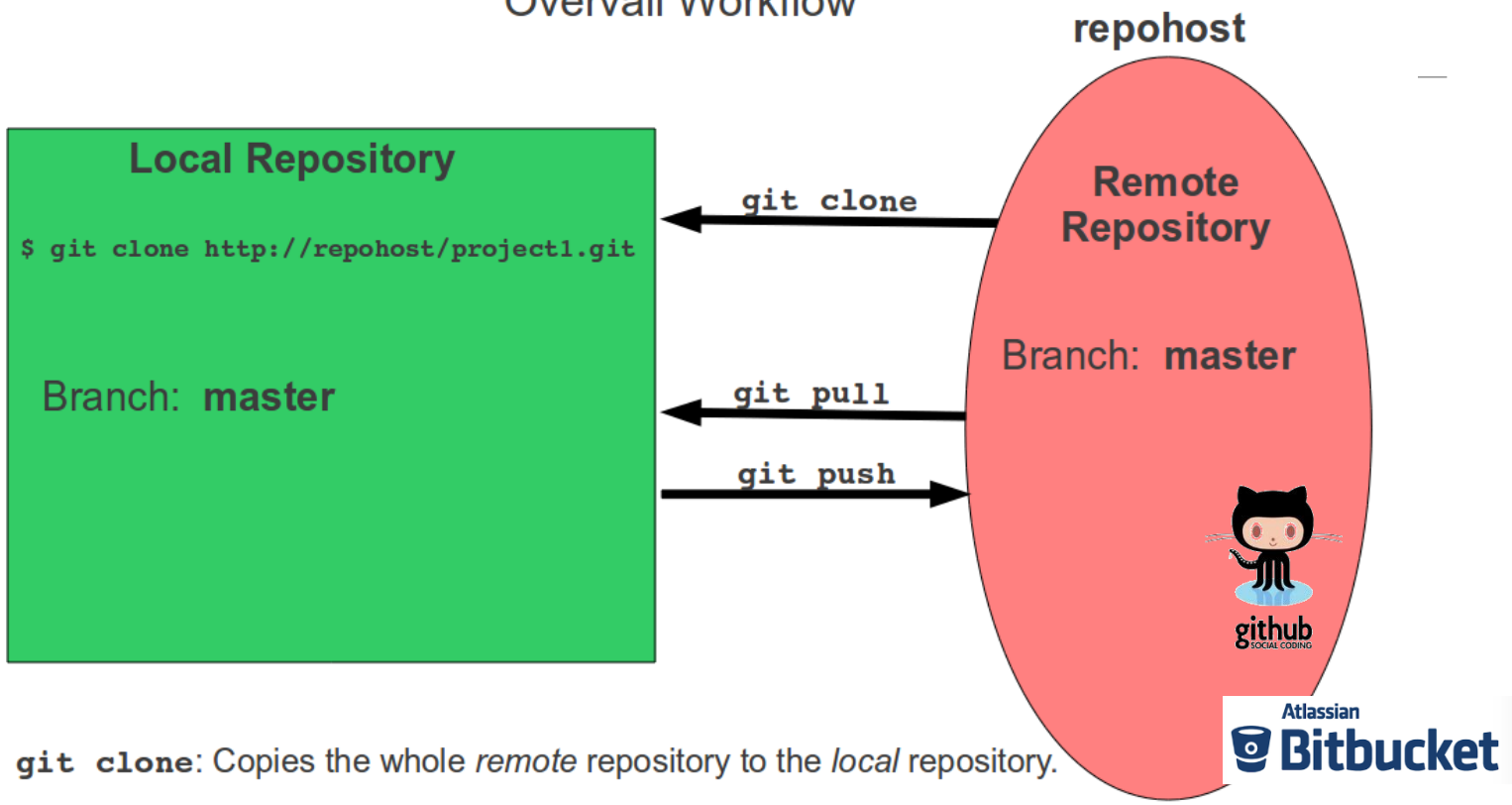
```
$ git commit --ammend --author=”Tim dams<tim.dams@ap.be>”
```

Working in team

LEARNING REMOTE

Overview

Git Remote Repositories: The High-level ("10,000 foot") View:
Overvall Workflow

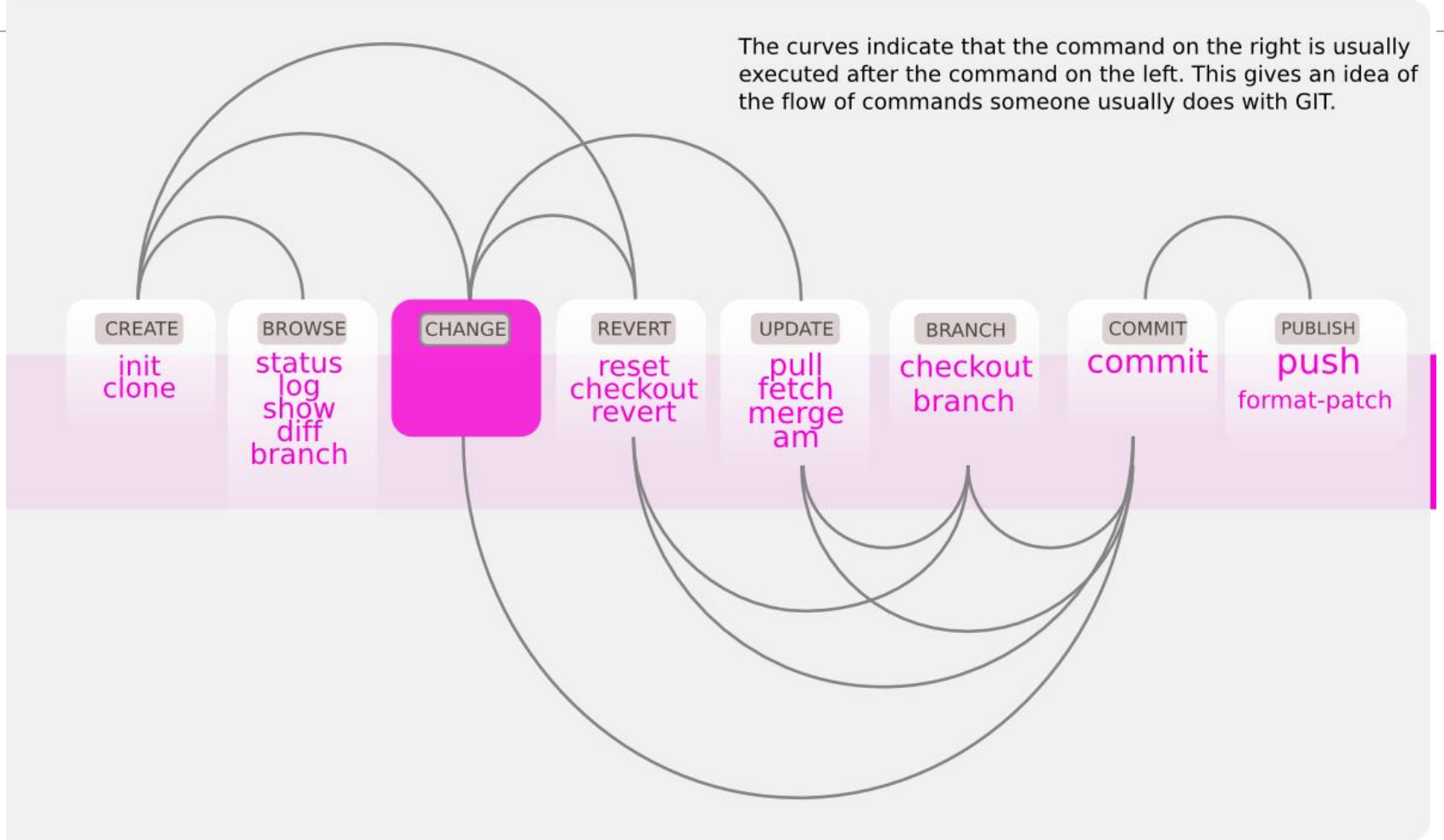


`git clone`: Copies the whole *remote* repository to the *local* repository.

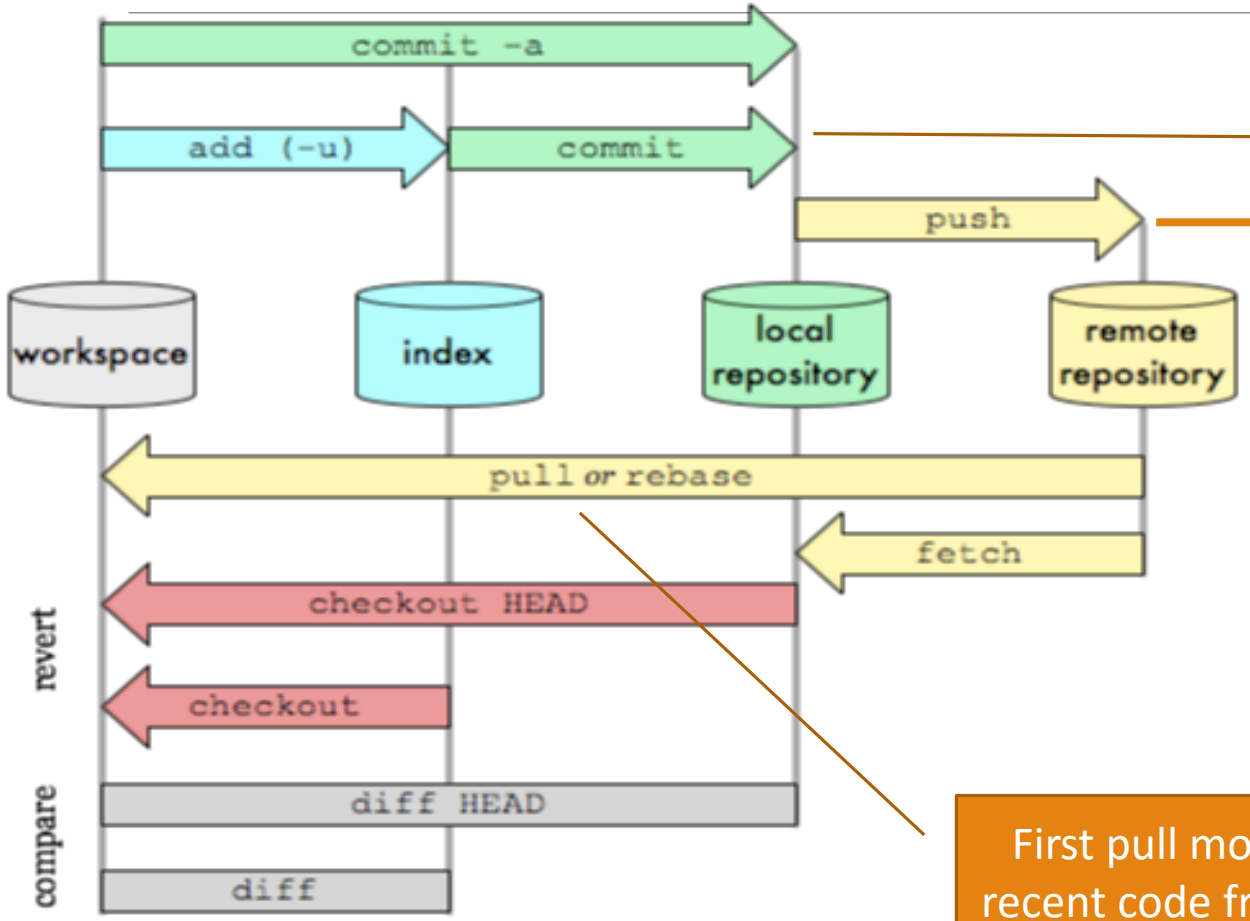
`git pull`: Retrieves any updates from the remote repository that aren't yet in the local repository and merges them into the local repository.

`git push`: Publishes updates from the local repository to the remote repository

Commands Sequence



Always pull before commiting!



Then commit and push the changes

First pull most recent code from github

Using Remote

Use git clone to replicate repository

Get changes with

- git fetch (fetches and merges)
- git pull

Propagate changes with

- git push

Protocols

- Local filesystem
- SSH
- Rsync
- HTTP
- Git protocol

Cloning our Repository

git clone *first-git-repo*

- Now have a full git repository to work with

Changes are pushed back with ***git push***

- Pushing changes WILL NOT change working copy on the repository being worked on

Branches can be based off of remote branches

- *git branch --track new-branch remote/branch*

Remote configuration information stored in *.git/config*

- Can have multiple remote backends!

Online git repos

Free git-repos:

- http://www.Github.com (projects will be publicly visible, unless paid)
- http://www.Bitbucket.org (git)
- http://www.projectlocker.com/ (git & svn)
- http://www.codeplex.com (supports .NET ClickOnce)
- http://code.google.com (git & svn)
- http://www.sourceforge.com (get & svn)
- http://tfs.visualstudio.com/ (git &

Resolving conflicts

http://www.syntevo.com/smartgithg/howtos.html

http://www.slightlymagic.net/wiki/Forge:_How_to_Install_and_Use_SmartGit

Push

Update remote branch with changes from local branch

$ git push –u origin master

-u = add a tracking reference

Clone

Clone a repository into a new directory

```
$ git clone git@github.com:brandonneil/test.git
```

Pull

Fetch from and merge with another repository or local branch

$ git pull

First pull, then commit, then push
Repeat

Credits

Largely based ony slides by esteemed colleague : Kristof Michiels

git & github

WORKING IN TEAM

Introduction

Github is currently changing the way software is being made

Originally a platform for developers on which they could cooperate on open source project

Nowadays: de facto standard platform for joint software development

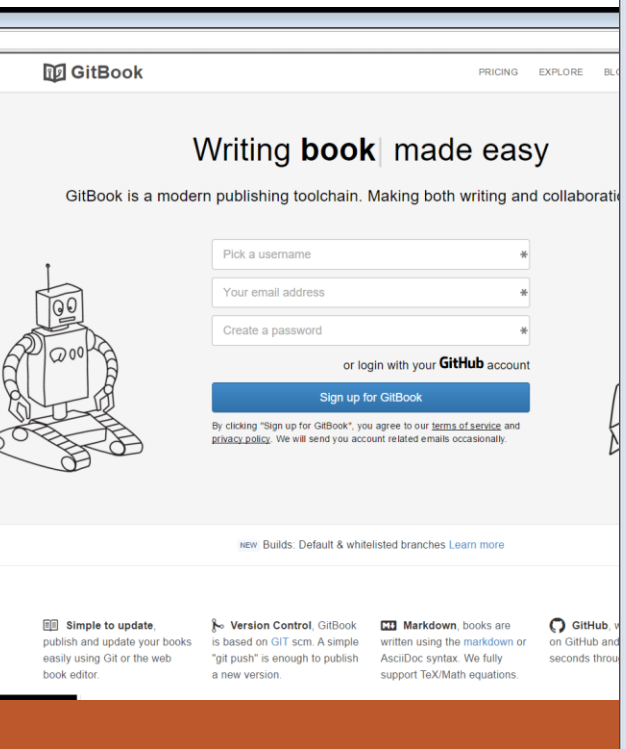
Keywords:

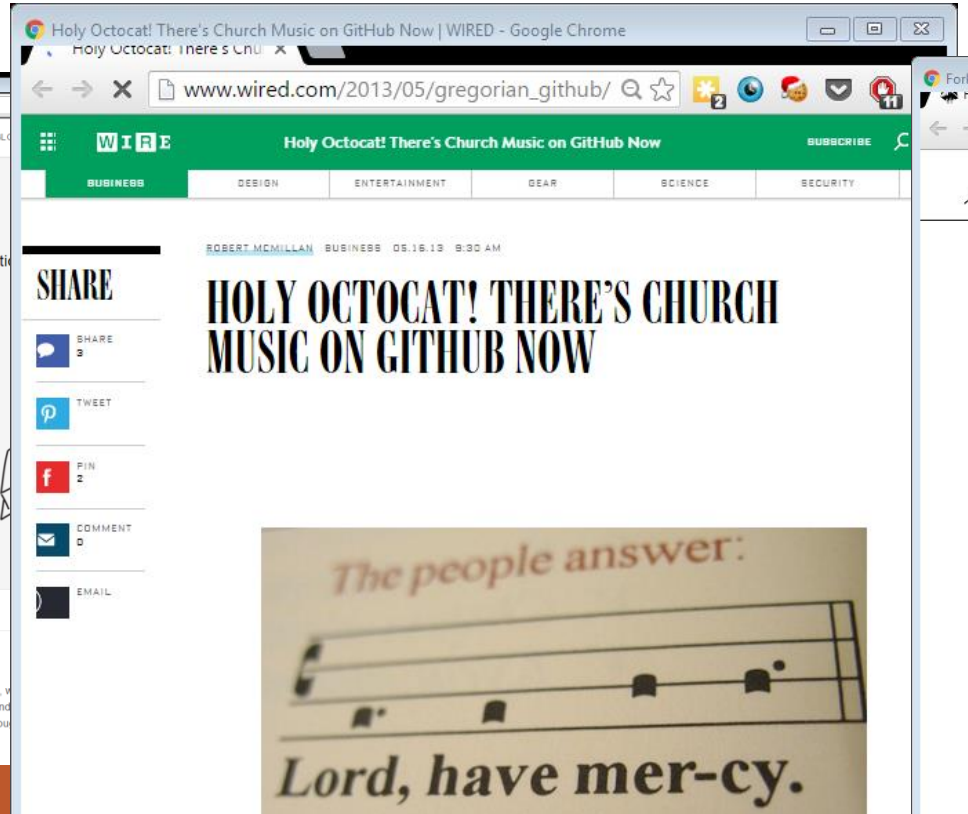
- Versioning/History of project
- Cooperation

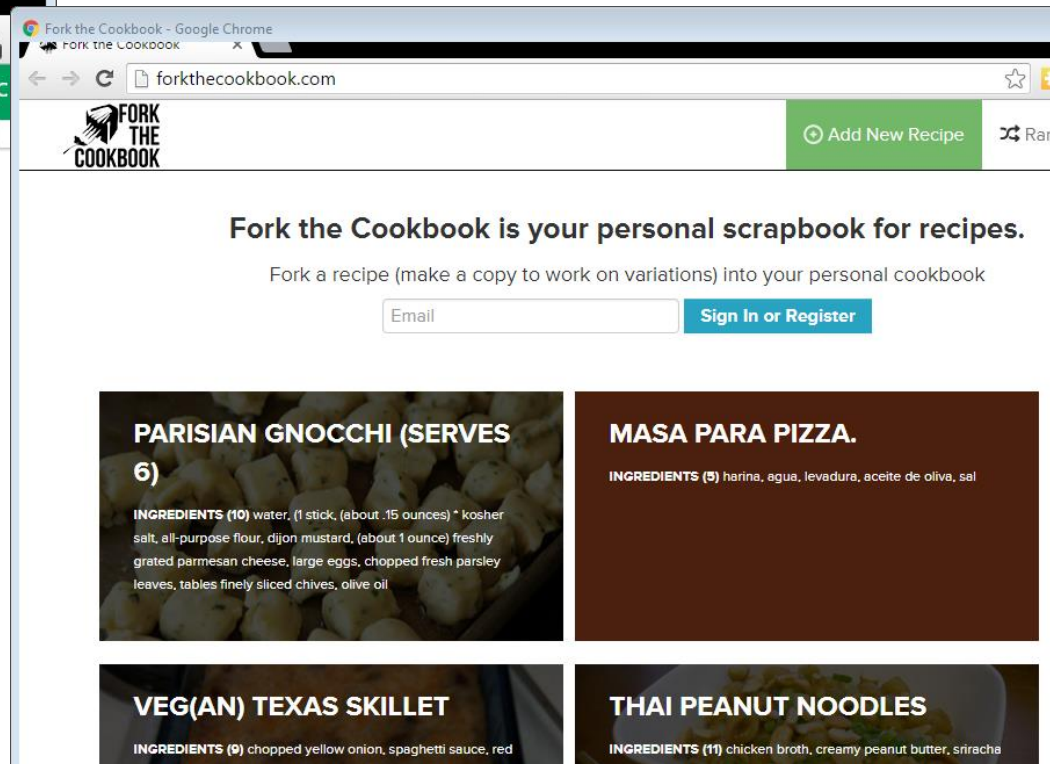
Not only used for software

Main usage is software development, but:

- Is being used for completely other stuff were teamwork and/or versioning is needed
- Some cool examples: http://readwrite.com/2013/11/08/seven-ways-to-use-github-that-arent-coding







Git and github

So what is Github then?

GitHub is a website where you can store a copy of your Git “**reposistory**” (i.e. your project)

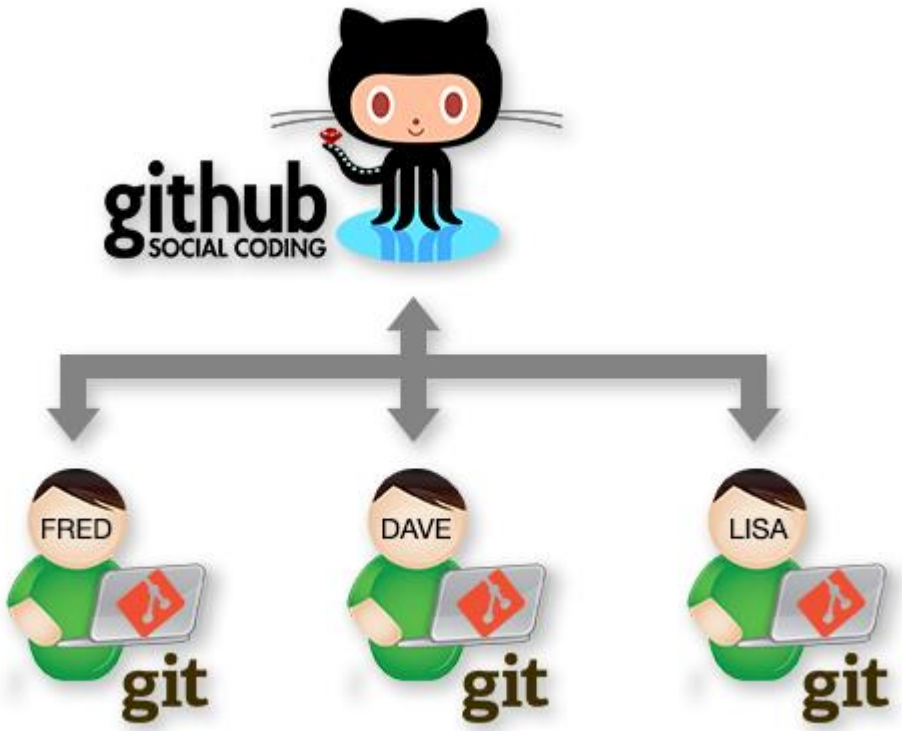
GitHub allows easy-to-use collaboration

GitHub has additional team tools such as wikis, issue management, forking, pull requests etc.

Git and github

Github is one (albeit most popular) cloudprovider on which to collaborate “talking git”

- Others exist, eg: http://www.bitbucket.org

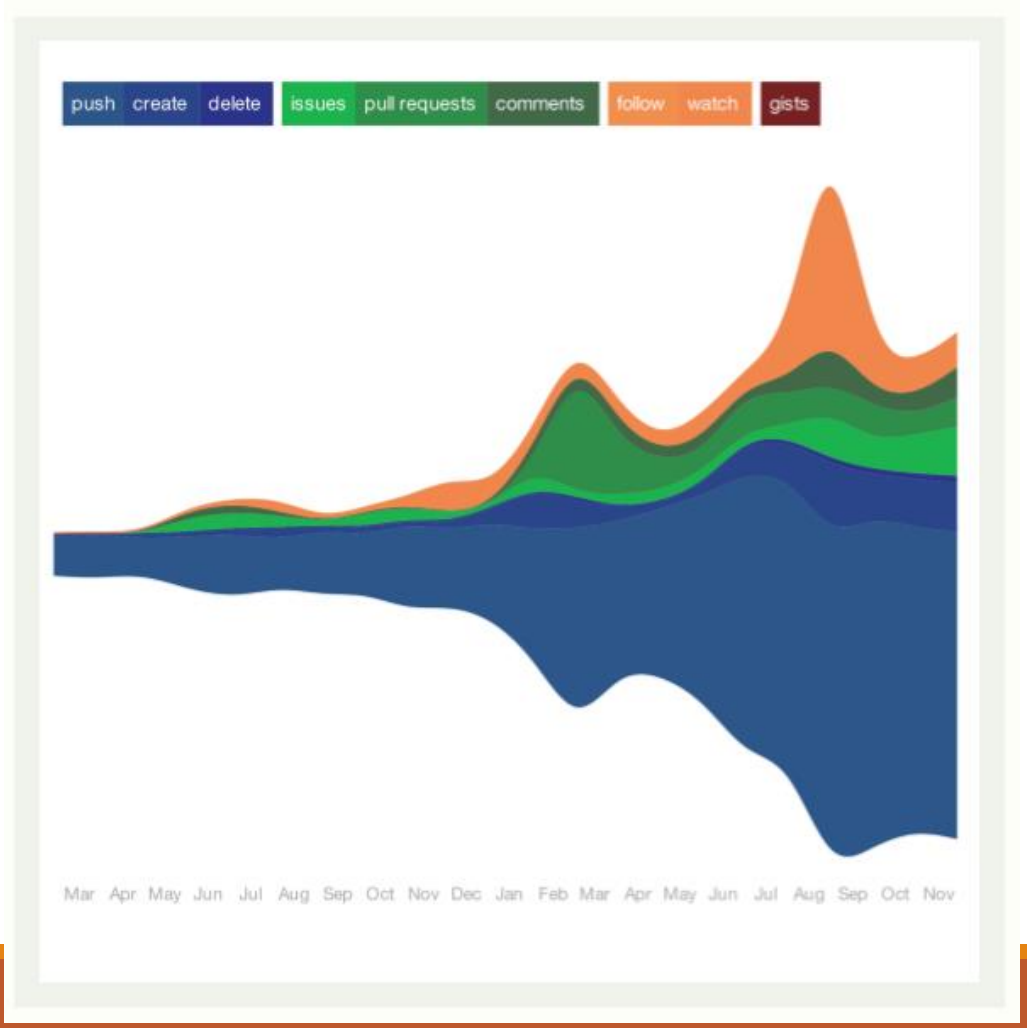


And why then use GitHub

And why then use GitHub

GitHub is not only an online git repo(sitory):

- Documents requirements
- Cooperate on different branches/features
- Review work
- Visualise team progress



Lets have a look at a github project

This repository Search

Pull requests Issues Gist

jonasswiatek / strokes

Unwatch 3 Unstar 34 Fork 5

Code Issues 5 Pull requests 0 Wiki Pulse Graphs

Achievements for Visual Studio http://www.codeachievements.net

403 commits 1 branch 0 releases 4 contributors

Branch: master New pull request New file Find file HTTPS https://github.com/jonasswi Download ZIP

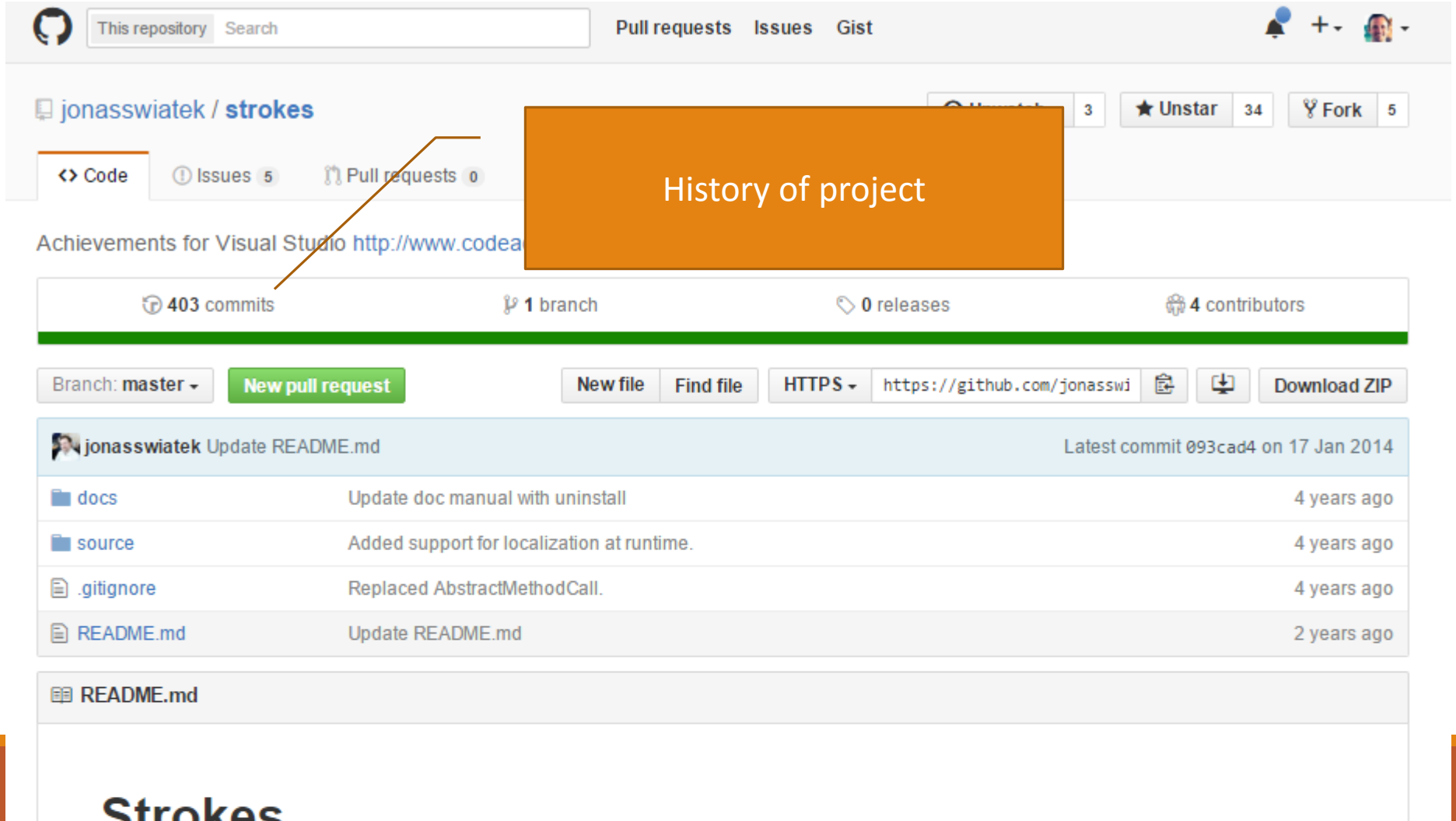
jonasswiatek Update README.md Latest commit 093cad4 on 17 Jan 2014

docs	Update doc manual with uninstall	4 years ago
source	Added support for localization at runtime.	4 years ago
.gitignore	Replaced AbstractMethodCall.	4 years ago
README.md	Update README.md	2 years ago

README.md

Strokes

Commits


This repository
Search
Pull requests
Issues
Gist
jonasswiatek / strokes
3
Unstar
34
Fork
5
Code
Issues 5
Pull requests 0
History of project
Achievements for Visual Studio http://www.codea
403 commits
1 branch
0 releases
4 contributors
Branch: master
New pull request
New file
Find file
HTTPS
https://github.com/jonasswi
Download ZIP
jonasswiatek Update README.md
Latest commit 093cad4 on 17 Jan 2014
docs
Update doc manual with uninstall
4 years ago
source
Added support for localization at runtime.
4 years ago
.gitignore
Replaced AbstractMethodCall.
4 years ago
README.md
Update README.md
2 years ago
README.md
Strokes

Commits

Download most recent code

This repository Search

Pull requests Issues Gist

jonasswiatek / strokes

Unwatch 3 Unstar 34 Fork 5

Code Issues 5 Pull requests 0 Wiki Pulse Graphs

Achievements for Visual Studio http://www.codeachievements.net

403 commits 1 branch 0 releases 4 contributors

Branch: master New pull request New file Find file HTTPS https://github.com/jonasswi Download ZIP

jonasswiatek Update README.md Latest commit 093cad4 on 17 Jan 2014

docs	Update doc manual with uninstall	4 years ago
source	Added support for localization at runtime.	4 years ago
.gitignore	Replaced AbstractMethodCall.	4 years ago
README.md	Update README.md	2 years ago

README.md

Strokes

Commits on Oct 30, 2011

Removed redundant generic specifier
jonasswiatek committed on 30 Oct 2011 — 5a3b27c

Merge branch 'master' of github.com:jonasswiatek/strokes
jonasswiatek committed on 30 Oct 2011 — 7527142

Refactored Challenge-base types. ···
jonasswiatek committed on 30 Oct 2011 — 328bad7

Refactored Challenge-base types. ···
jonasswiatek committed on 30 Oct 2011 — f494526

Ignored 'unused variables/methods/event warnings for the BasicAchieve... ···
clausjoergensen committed on 30 Oct 2011 — a1580c4

Commits on Oct 29, 2011

Replaced AbstractMethodCall. ···
jonasswiatek committed on 29 Oct 2011 — 29beeaf

Commits on Oct 28, 2011

Merge branch 'master' of git@github.com:jonasswiatek/strokes.git ···
timdams committed on 28 Oct 2011 — e72e77d

HintUri added to all achievements. ···
timdams committed on 28 Oct 2011 — 48ffb51

Merge branch 'master' of github.com:jonasswiatek/strokes

Issues

Issues: bugs, feature ideas, etc (incl conversations)

This repository Search

jonasswiatek / strokes

Unstar 34 Fork 5

Code | Issues 5 | Pull requests 0 | Wiki | Pulse | Graphs

Achievements for Visual Studio http://www.codeachievements.net

403 commits | 1 branch | 0 releases | 4 contributors

Branch: master | New pull request | New file | Find file | HTTPS https://github.com/jonasswi | Download ZIP

jonasswiatek Update README.md — Latest commit 093cad4 on 17 Jan 2014

docs	Update doc manual with uninstall	4 years ago
source	Added support for localization at runtime.	4 years ago
.gitignore	Replaced AbstractMethodCall.	4 years ago
README.md	Update README.md	2 years ago

README.md

Strokes

jonasswiatek / strokes

Unwatch 3 | Unstar 34 | Fork 5

Code | Issues 5 | Pull requests 0 | Wiki | Pulse | Graphs

Filters | is:issue is:open | Labels | Milestones | New issue

5 Open | 33 Closed | Author | Labels | Milestones | Assignee | Sort

Refactor NRefactoryContext
#40 opened on 30 Oct 2011 by jonasswiatek Beta release (non onl...

GUID register of all achievements Feature Idea
#39 opened on 30 Oct 2011 by clausjoergensen 3

Feedback on progress in challenge-type achievements Feature Request
#32 opened on 14 Oct 2011 by timdams Beta release (non onl... 2

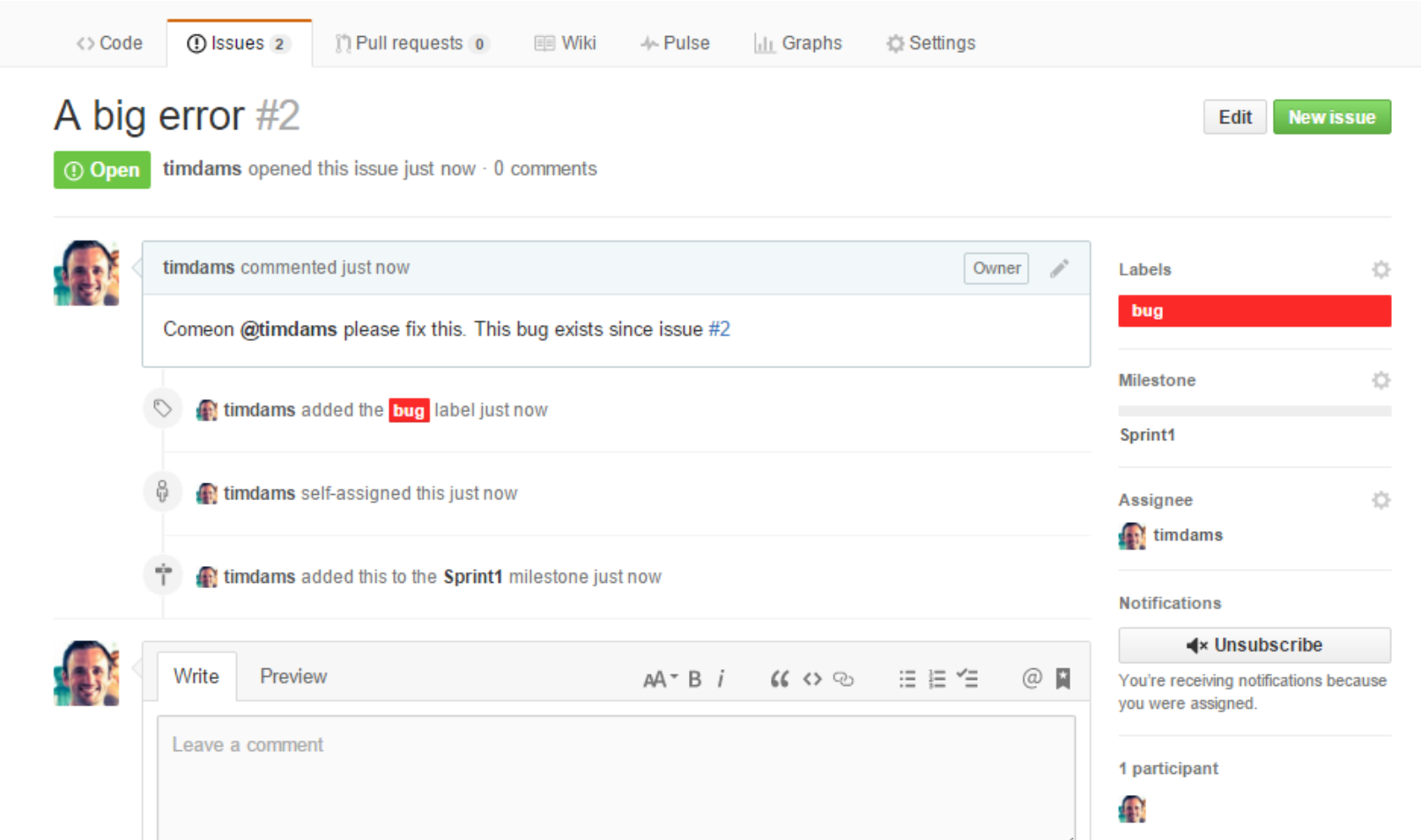
Visual Achievement Tree Feature Request
#28 opened on 13 Oct 2011 by jonasswiatek Beta release (non onl...

List of Achievement Ideas Feature Idea
#26 opened on 13 Oct 2011 by jonasswiatek 11

ProTip! What's not been updated in a month: updated:<2016-01-04.

Usefull Issue usage
Milestones, labels, mentions and references

https://guides.github.com/features/issues/



Pulse

Recent activity

This repository Search

Pull requests Issues Gist

jonasswiatek / **strokes**

Code | Issues 5 | Pull requests 0 | Wiki | Pulse | Graphs

Achievements for Visual Studio http://www.codeachievements.net

403 commits | 1 branch | 0 releases | 4 contributors

Branch: master | New pull request | New file | Find file | HTTPS | https://github.com/jonasswi | Download ZIP

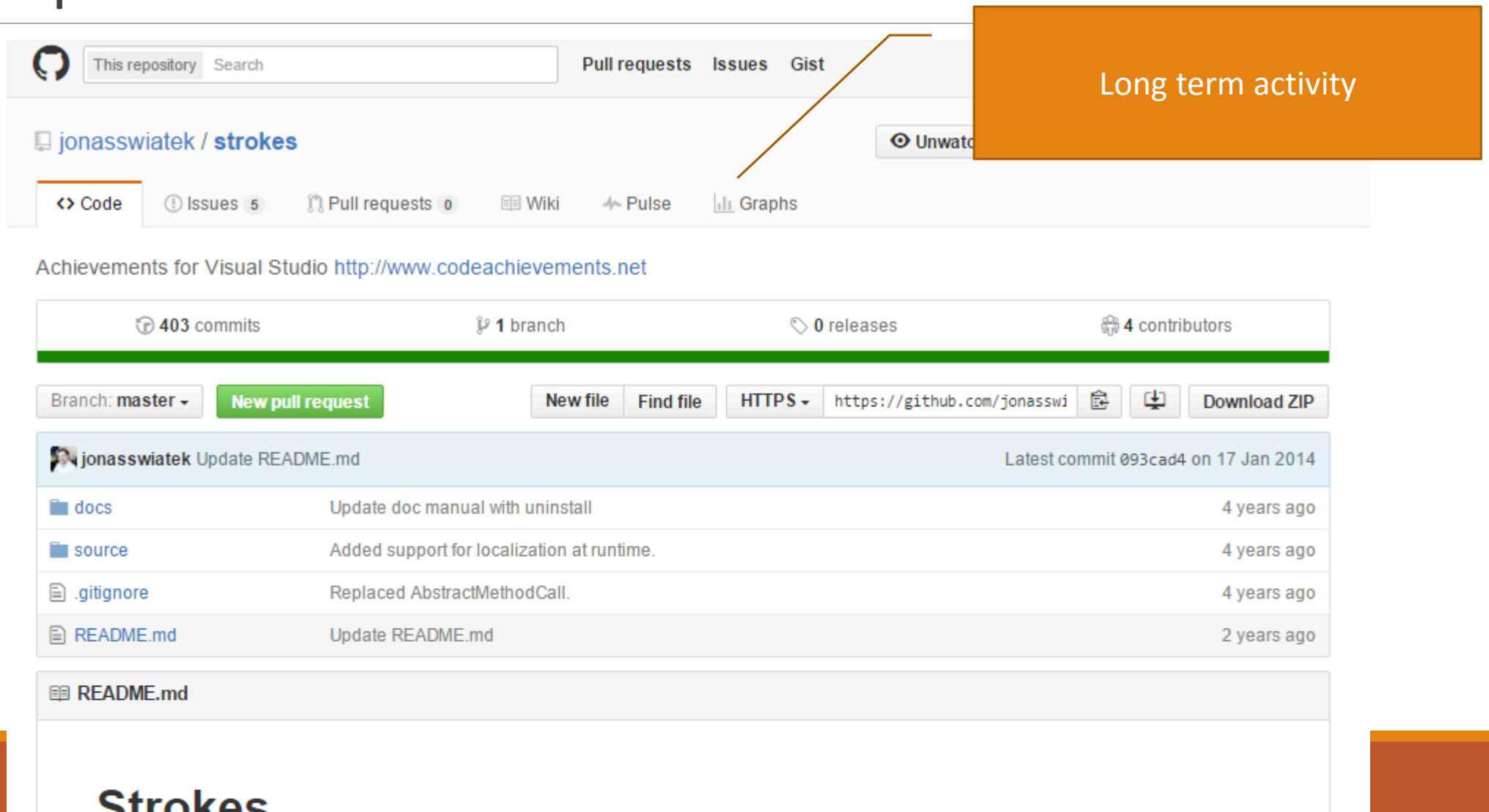
jonasswiatek Update README.md — Latest commit 093cad4 on 17 Jan 2014

docs	Update doc manual with uninstall	4 years ago
source	Added support for localization at runtime.	4 years ago
.gitignore	Replaced AbstractMethodCall.	4 years ago
README.md	Update README.md	2 years ago

README.md

Strokes

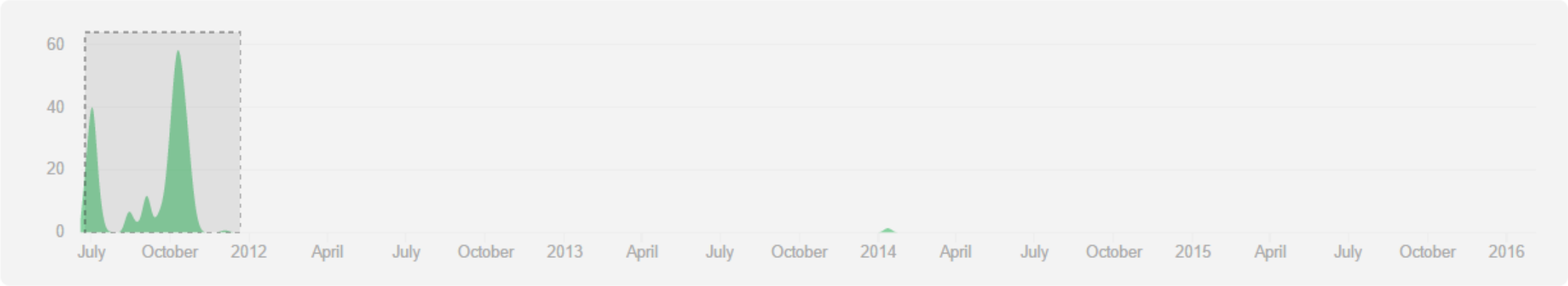
Graphs


This repository
Search
Pull requests
Issues
Gist
Long term activity
jonasswiatek / strokes
Unwatc
Code
Issues 5
Pull requests 0
Wiki
Pulse
Graphs
Achievements for Visual Studio http://www.codeachievements.net
403 commits
1 branch
0 releases
4 contributors
Branch: master
New pull request
New file
Find file
HTTPS
https://github.com/jonasswi
Download ZIP
jonasswiatek Update README.md
Latest commit 093cad4 on 17 Jan 2014
docs
Update doc manual with uninstall
4 years ago
source
Added support for localization at runtime.
4 years ago
.gitignore
Replaced AbstractMethodCall.
4 years ago
README.md
Update README.md
2 years ago
README.md
Strokes

Jun 24, 2011 – Dec 21, 2011

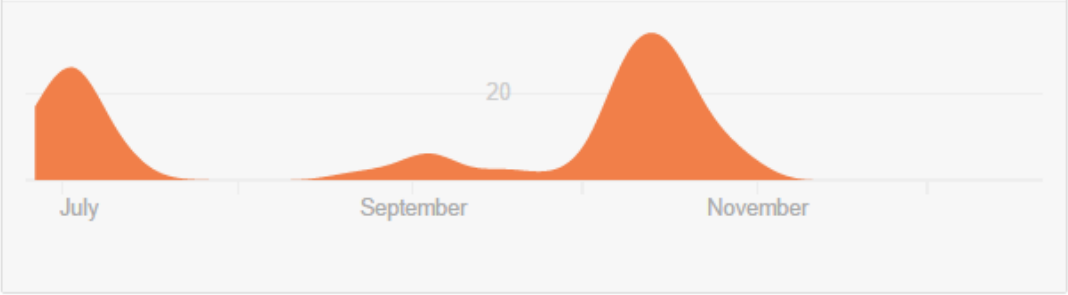
Contributions: **Commits**

Contributions to master, excluding merge commits



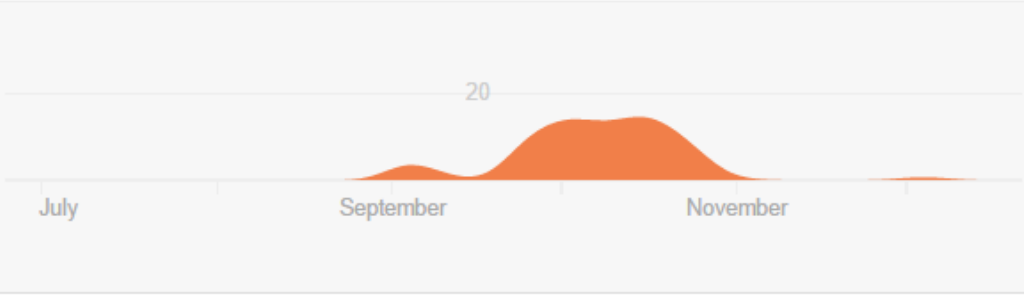
jonasswiatek #1

177 commits / 77,950 ++ / 83,642 --



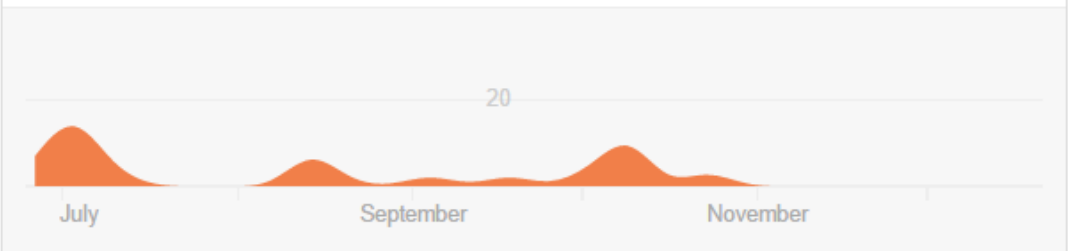
clausjoergensen #2

73 commits / 16,973 ++ / 9,659 --



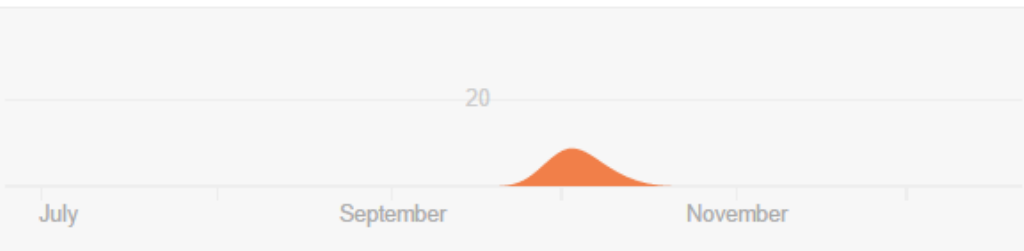
timdams #3

66 commits / 13,741 ++ / 6,415 --



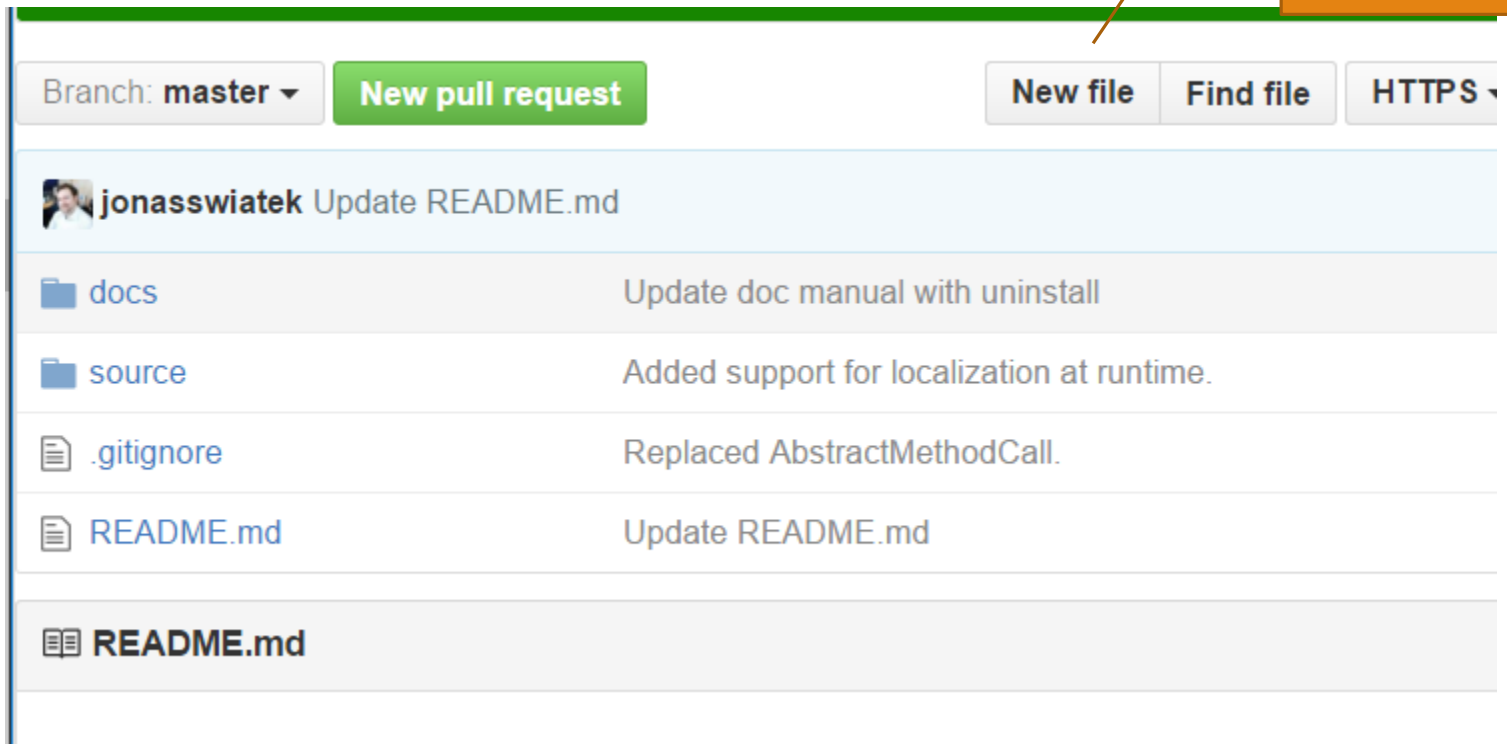
vlad2135 #4

16 commits / 15,347 ++ / 13,445 --



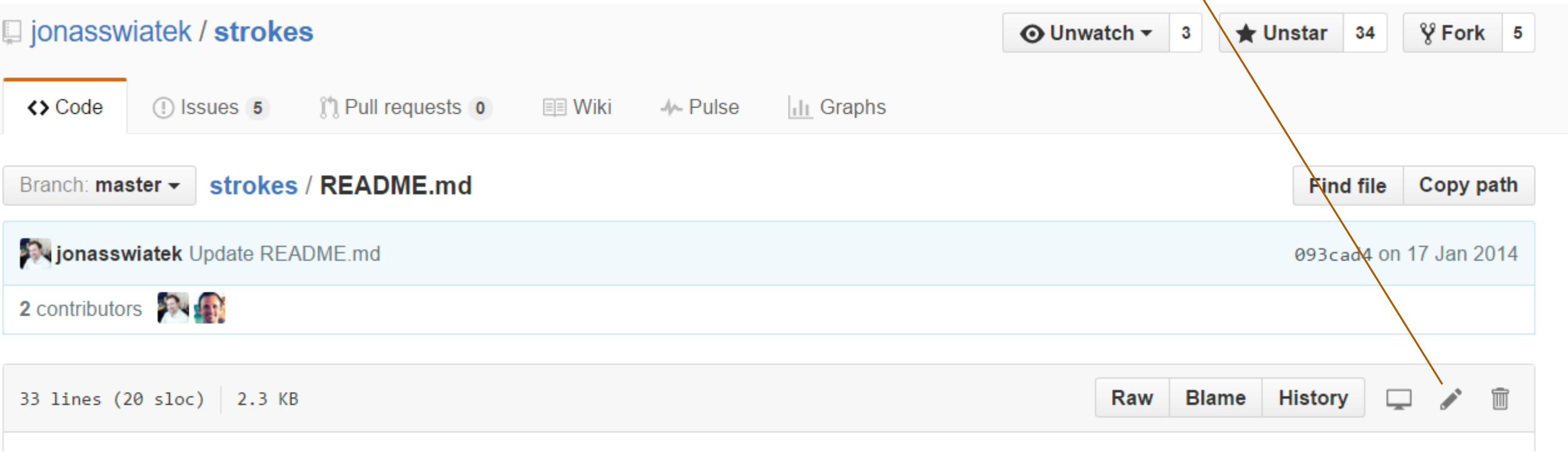
Adding

Add file



Editing

Open file to edit and then this “edit pen”



Markdown

.md files

Simple textbased format

Github likes to work with Markdown

Markdown

Naamloos - Kladblok

Bestand Bewerken Opmaak Beeld Help

```
# Strokes
**Achievements for Visual Studio .NET 2010**

**Important information:** In order to run the project from code, please read the [Wiki Page]
(https://github.com/jonasswiatek/strokes/wiki).

**MORE IMPORTANT INFORMATION:** If you want to try the current alpha-quality releases, remember to use the
"**Strokes Achievement Project**"-template (located in the Visual C# Category in the New Project Dialog of Visual
Studio), since we ONLY check for achievements within projects of this type!

**Information on using the application: A wiki page on how to install , use and uninstall Strokes can be found
here [Wiki Page] (https://github.com/jonasswiatek/strokes/wiki/Getting-started-guide-for-endusers).

This Visual Studio Extension strives to provide a tool for teaching C# to beginners and give experienced
programmers an opportunity to catch up with all the features of the C# programming language.

This project is currently in a beta stage, and you can download a copy from the downloads section here on github.
This extension is a proper Visual Studio 2010 extension, so it is easily removed with visual studios extension
management features. The only data this extension writes to your hard drive (except for it's assemblies deployed
by the VSIX installer), is a small XML file in your AppData Directory that contains which achievements you've
unlocked.

**Create a new Strokes Achievement Project**
![New Project](https://raw.github.com/jonasswiatek/strokes/master/docs/project_template.png)

You can now start writing achievements. If you want to try our challenges, then follow these steps:

**Add a new item**
![New Project](https://raw.github.com/jonasswiatek/strokes/master/docs/project_add_challenge.png)

**Choose your challenge!**
```

Strokes

Achievements for Visual Studio .NET 2010

Important information: In order to run the projec

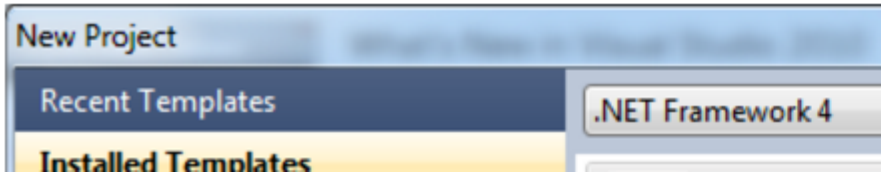
MORE IMPORTANT INFORMATION: If you want
Achievement Project"-template (located in the V
ONLY check for achievements within projects of t

**Information on using the application: A wiki page

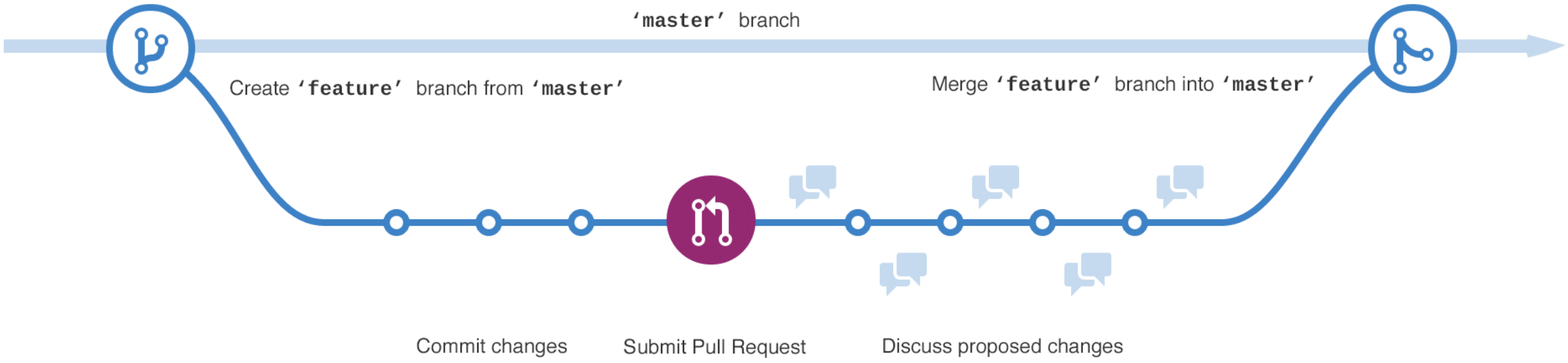
This Visual Studio Extension strives to provide a t
opportunity to catch up with all the features of the

This project is currently in a beta stage, and you c
extension is a proper Visual Studio 2010 extensio
features. The only data this extension writes to yo
small XML file in your AppData Directory that cont

Create a new Strokes Achievement Project



Pull request: notify and chat about possible merge



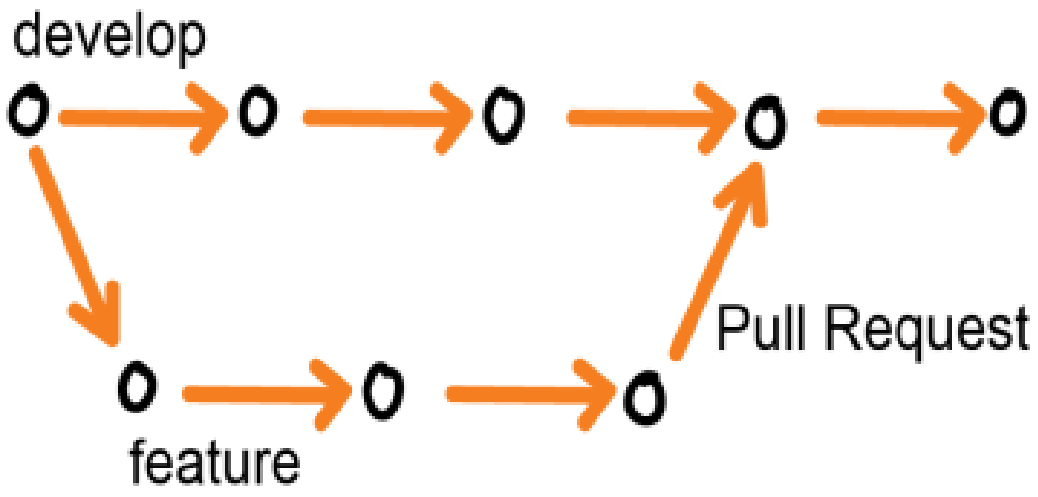
Team tips

Good team practice

Masterbranch: only for ‘live’ code

All testing and features in seperate branch

Merge to mastre once feature, refactor, etc is ‘production ready’



Good team practice

“Voor studenten kan je eventueel iedereen rechten geven aan alle branches, maar wordt er wel verwacht dat ze voor nieuwe features pull (merge) requests creeren en andere studenten als 'approver' toevoegen.

Pas wanneer je branch geapproved is mag je effectief mergen naar de master branch.

Zo verhinder je dat een student de repo naar de vaantjes helpt als hij braaf enkel code naar master merget via pull requests.

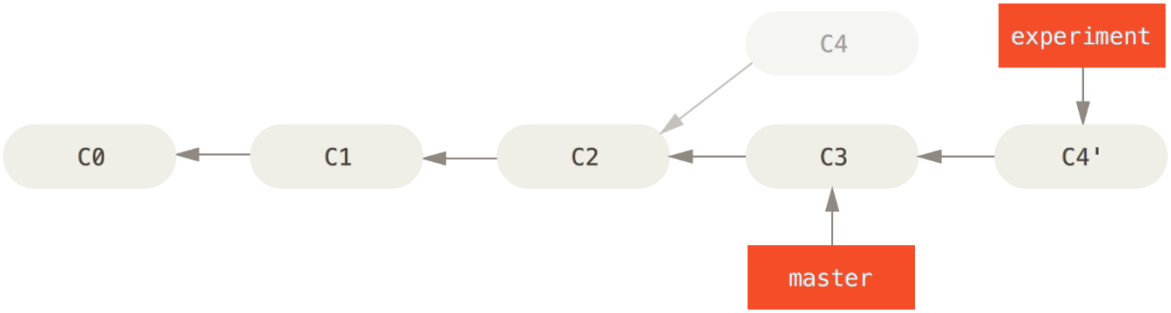
Dan hebben er minstens twee paar ogen naar gekeken. Ze kunnen technisch gezien dan wel rechtstreeks naar master mergen, maar het wordt als 'not done' beschouwd.

“

Rebase

Tip: Gebruik rebase om je merge te testen (https://git-scm.com/book/en/v2/Git-Branching-Rebasing)

- zie http://www.jillesvangurp.com/2011/07/16/using-git-and-feature-branches-effectively/)



Often, you'll do this to make sure your commits apply cleanly on a remote branch – perhaps in a project to which you're trying to contribute but that you don't maintain. In this case, you'd do your work in a branch and then rebase your work onto origin/master when you were ready to submit your patches to the main project. That way, the maintainer doesn't have to do any integration work – just a fast-forward or a clean apply

Git commands overview

All commands

Good cheatsheet: http://www.markus-gattol.name/misc/mm/si/content/git_workflow_and_cheat_sheet.png

Basic Commands - git

git – view all commands

```
$ git
```

Basic Commands - Init

Init - create an empty new repository

```
$ git init
```

Basic Commands - Status

Status - show differences between what has been committed and HEAD

$ git status

Basic Commands - Add

Add – add files to the stage

```
$ git add foo.info
```

Basic Commands - Commit

Commit – stores contents of the index in a commit along with a message

$ git commit –m “Added foo.info”

Basic Commands - Log

Log– view previous commits

$ git log

Basic Commands - Checkout

Checkout = checkout branches or previous commits

```
$ git checkout coolfeaturebranch
```

```
$ git checkout 1c899fed6ed
```

Remote - add

Add a repository that you track

```
$ git remote add origin git@github.com:brandonneil/test.git
```

Push

Update remote branch with changes from local branch

$ git push –u origin master

-u = add a tracking reference

Clone

Clone a repository into a new directory

```
$ git clone git@github.com:brandonneil/test.git
```

Pull

Fetch from and merge with another repository or local branch

```
$ git pull
```