Introduction to Git

(for remoll and other software)

"FINAL".doc


FINAL.doc!
FINAL_rev.2.doc
FINAL_rev.6.COMMENTS.doc
FINAL_rev.8.comments5.
CORRECTIONS.doc
track changes
FINAL_rev.18.comments7.
corrections9.MORE.30.doc
FINAL_rev.22.comments49.
corrections.10.#@$%WHYDID
ICOMETOGRADSCHOOL????.doc
JORGE CHAM © 2012

Resources

Essential:

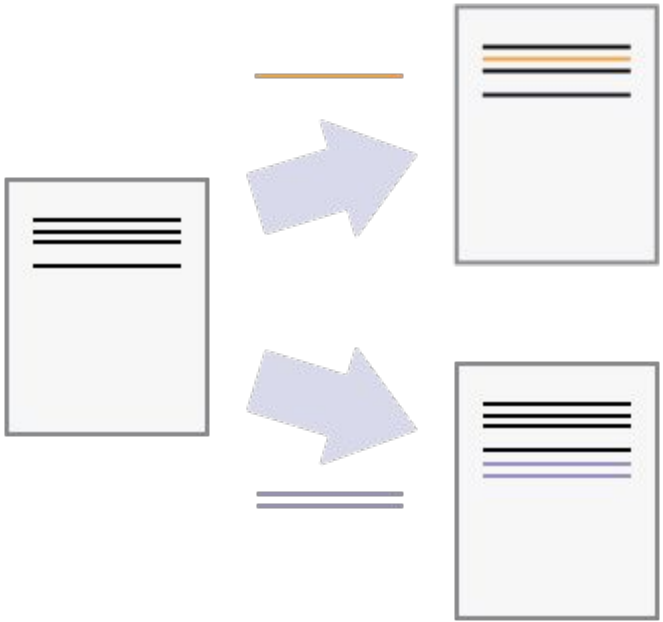
- https://swcarpentry.github.io/git-novice

Recommended:

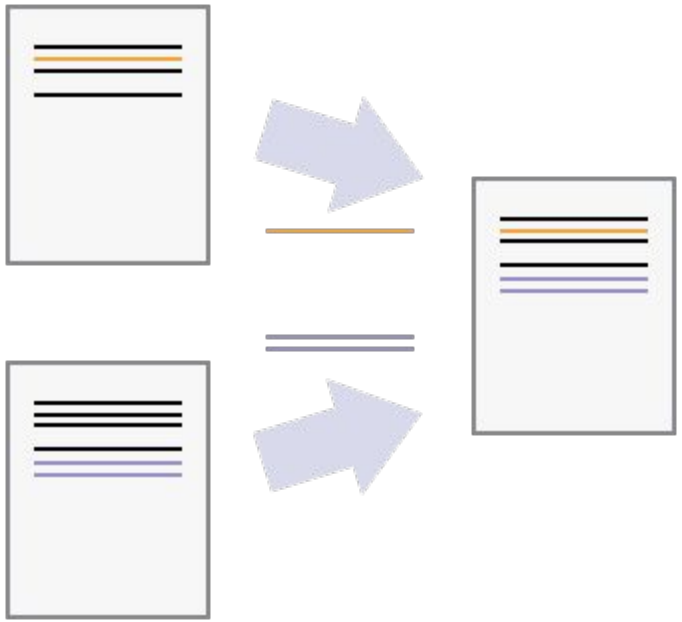
- https://datasift.github.io/gitflow/IntroducingGitFlow.html

Commits

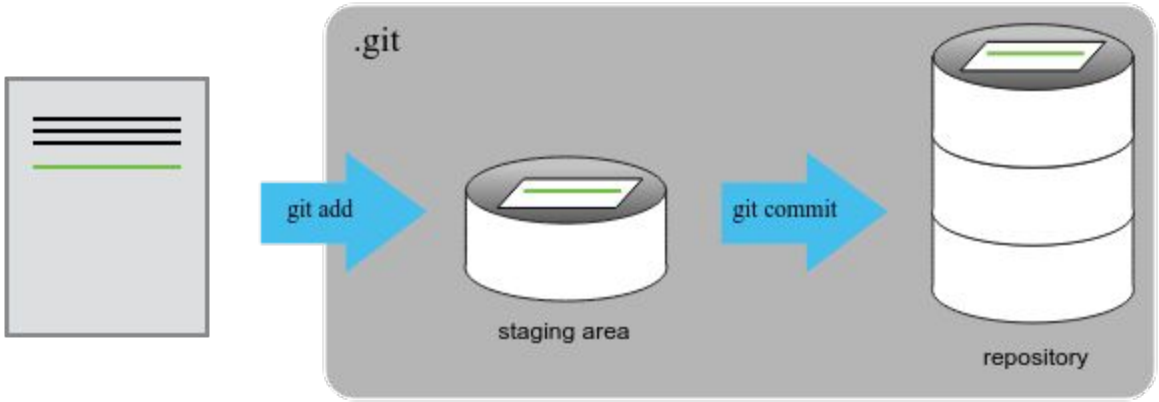
Branching

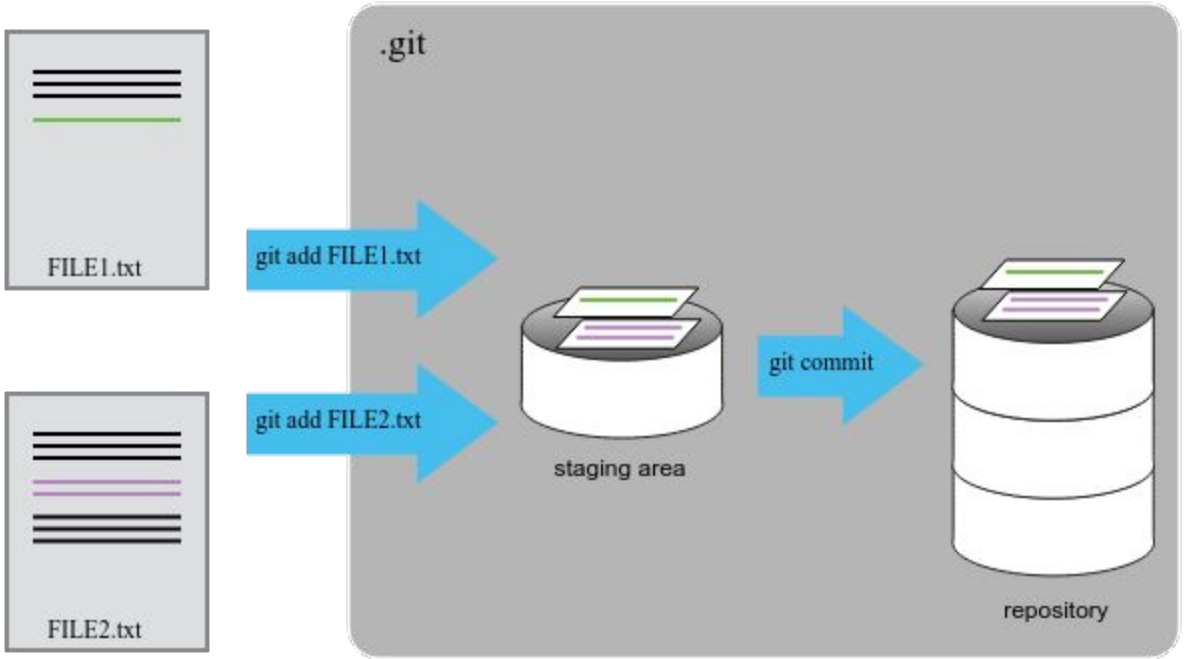


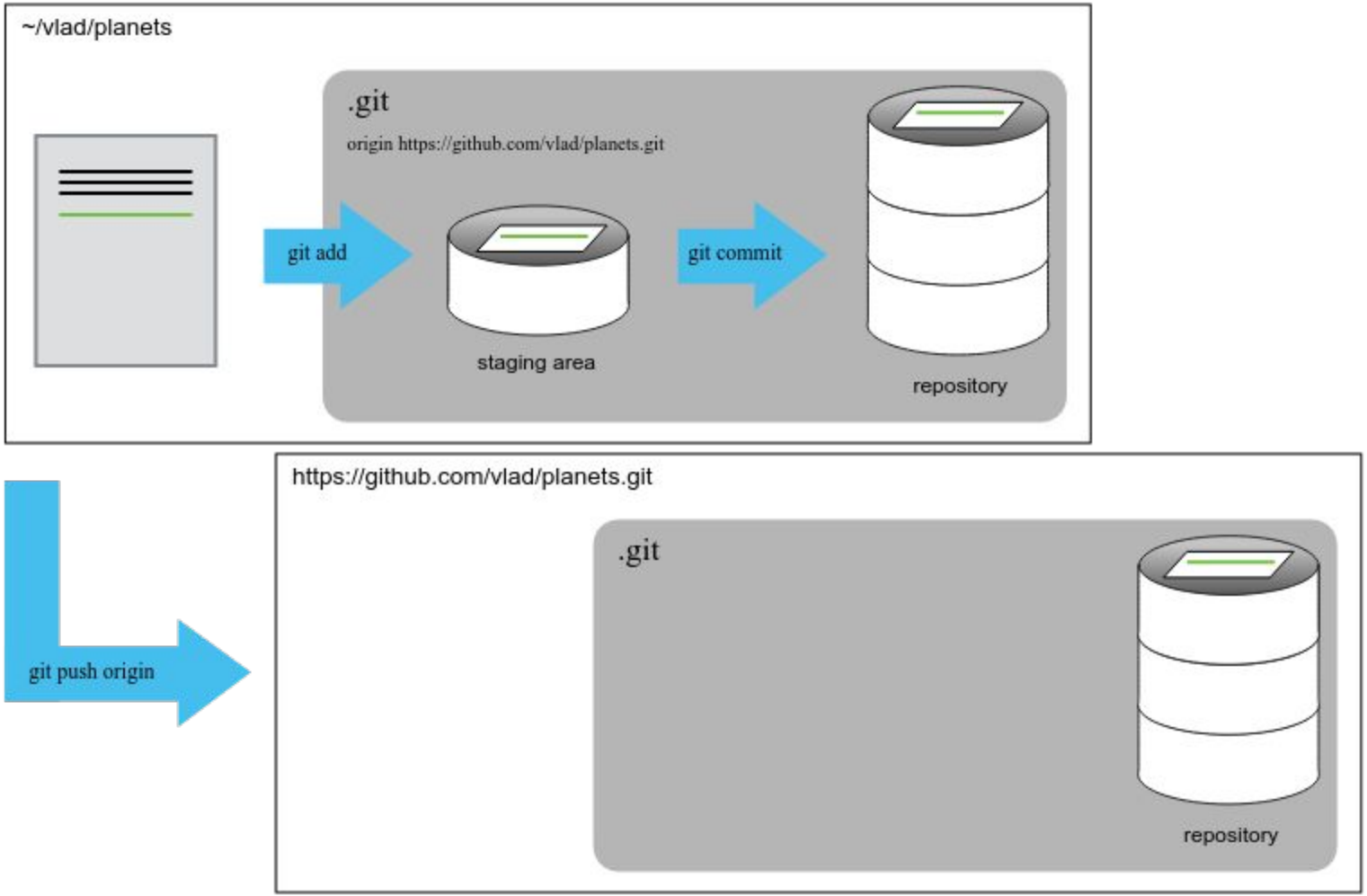
Merging

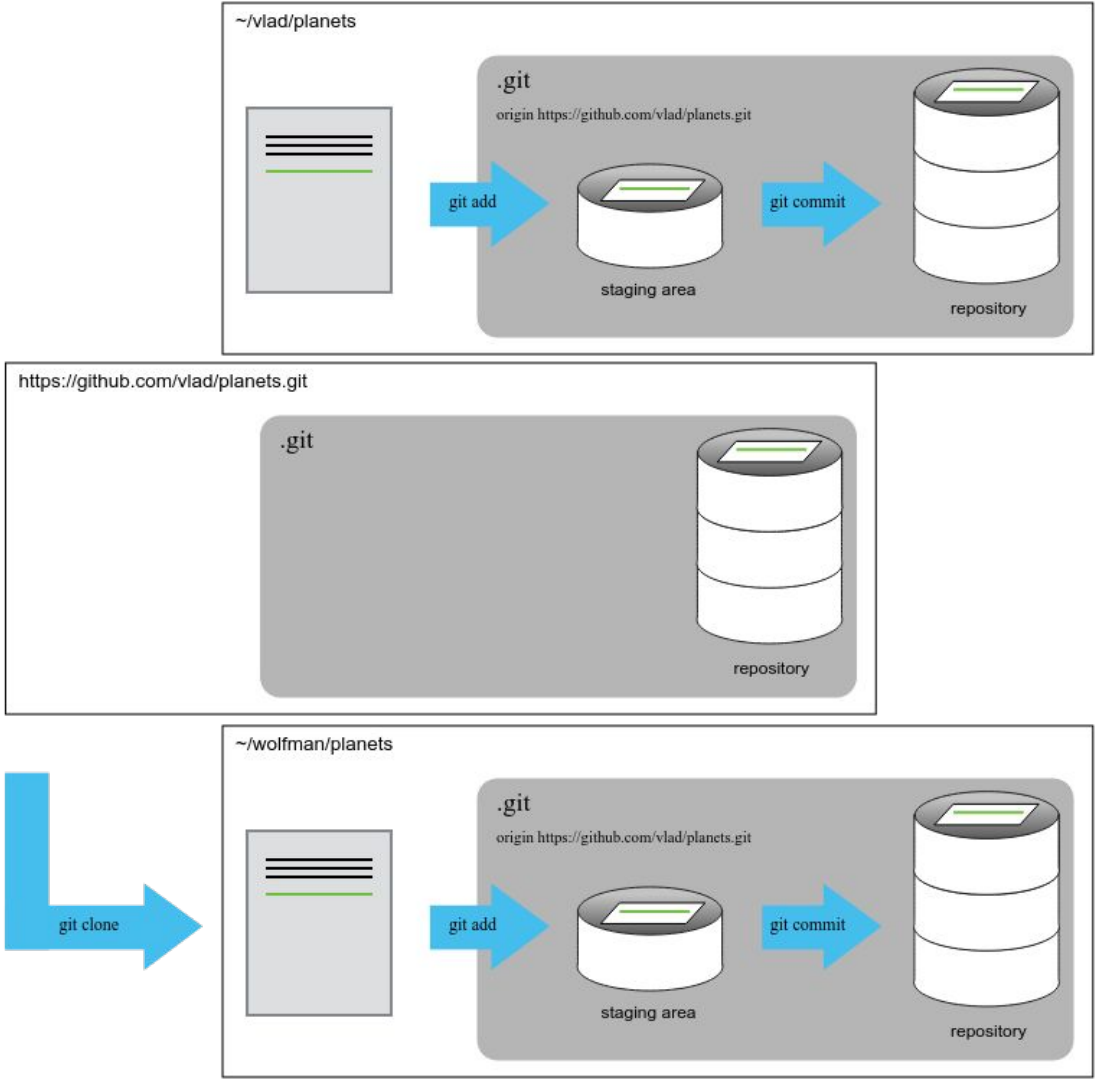


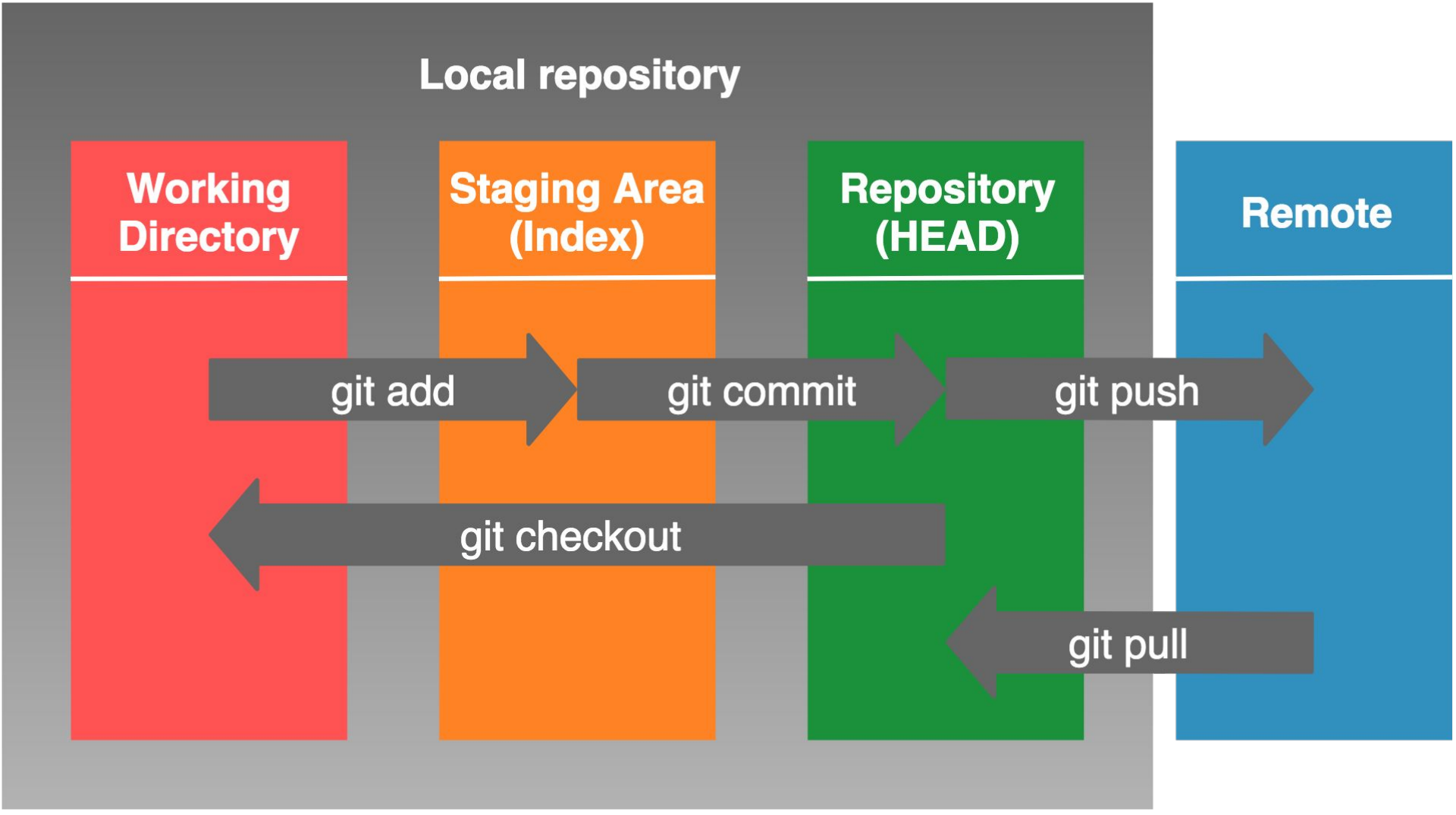
Document / Staging Area / Repository



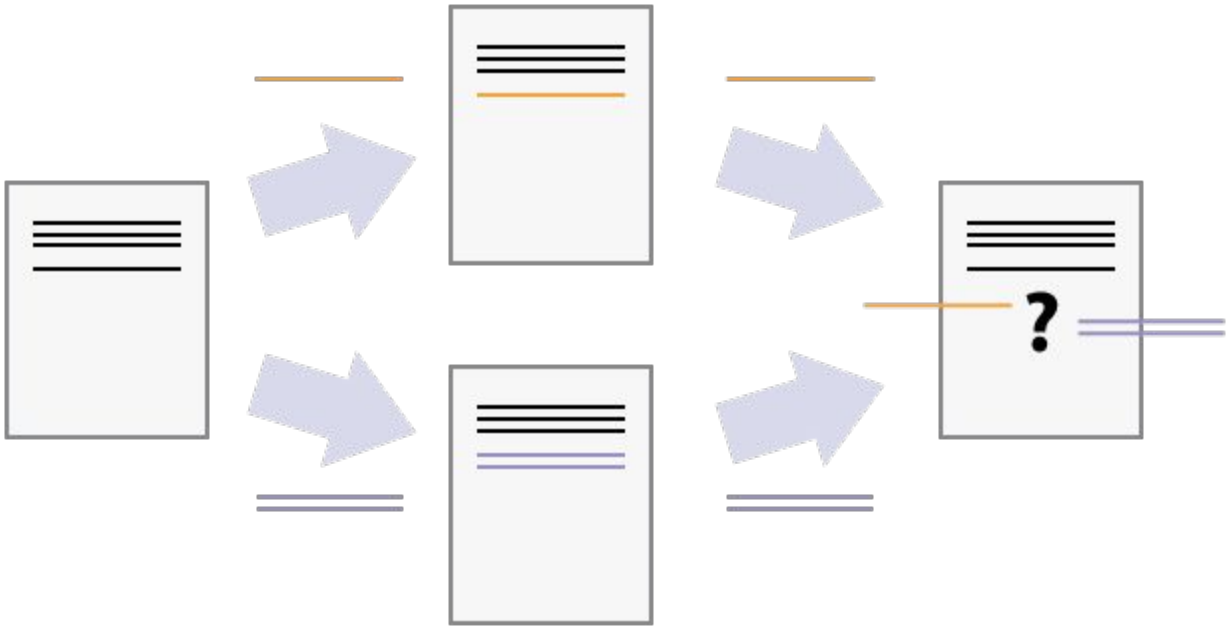

.git
FILE1.txt
FILE2.txt
git add FILE1.txt
git add FILE2.txt
staging area
git commit
repository


~/vlad/planets
.git
origin https://github.com/vlad/planets.git
git add
staging area
git commit
repository
git push origin
https://github.com/vlad/planets.git
.git
repository


~/vlad/planets
.git
origin https://github.com/vlad/planets.git
git add
staging area
git commit
repository
https://github.com/vlad/planets.git
.git
repository
~/wolfman/planets
git clone
.git
origin https://github.com/vlad/planets.git
git add
staging area
git commit
repository


Local repository
Working Directory
Staging Area (Index)
Repository (HEAD)
Remote
git add
git commit
git push
git checkout
git pull

Conflicts



Conflicts

```
Cold and dry, but everything is my favorite color
The two moons may be a problem for Wolfman
But the Mummy will appreciate the lack of humidity
<<<<<<< HEAD
We added a different line in the other copy
=======
This line added to Wolfman's copy
>>>>>>> dabb4c8c450e8475aee9b14b4383acc99f42af1d
```

Branches

- Main branch (often called “master”): default branch when cloning a repo
- Develop branch
- Feature branches
- Bugfix or hotfix branches

They are all the same in the eyes of git.

Most commonly used commands

- `git pull`
- `git status`
- `git diff`
- `git add`
- `git commit`
- `git checkout`

Less commonly used commands

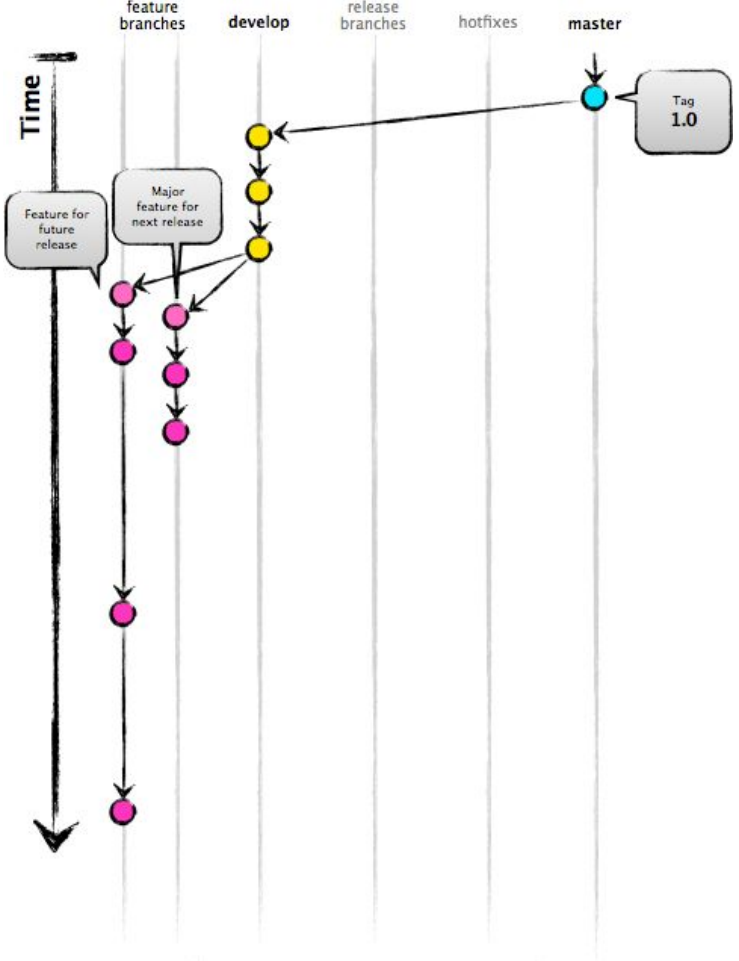
- `git stash push`
- `git stash pop`

- `git checkout -b new-branch-name`

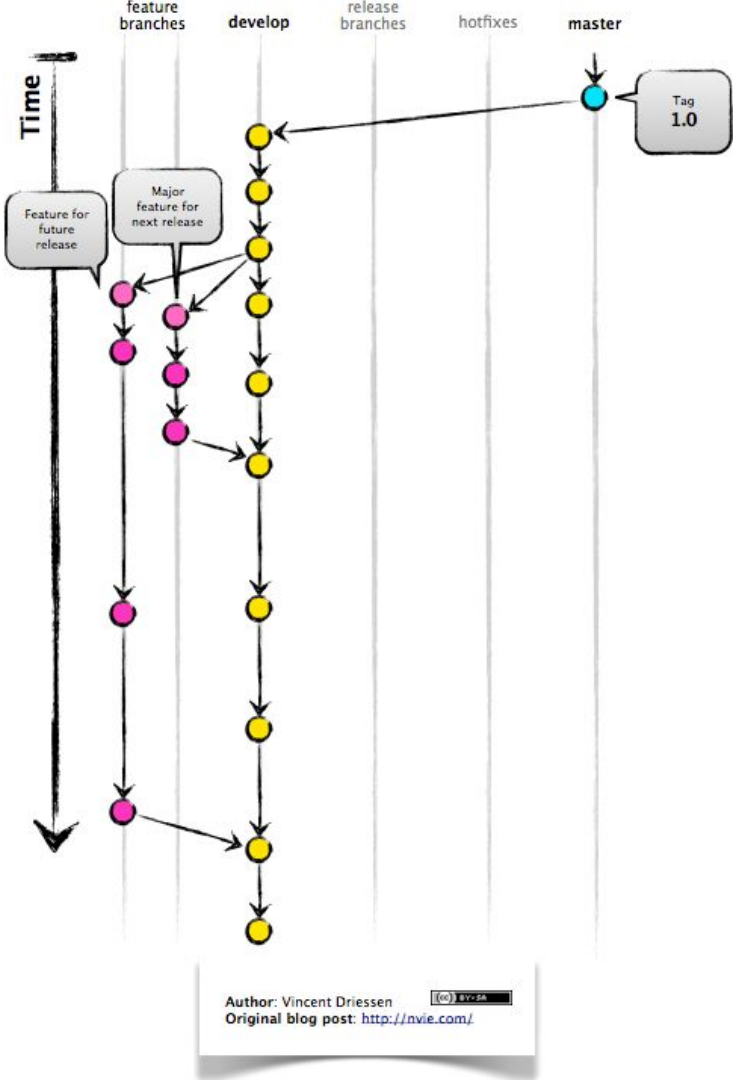

THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.


ONE DOES NOT SIMPLY MERGE TO MASTER
WITHOUT A MERGE REQUEST
imgflip.com

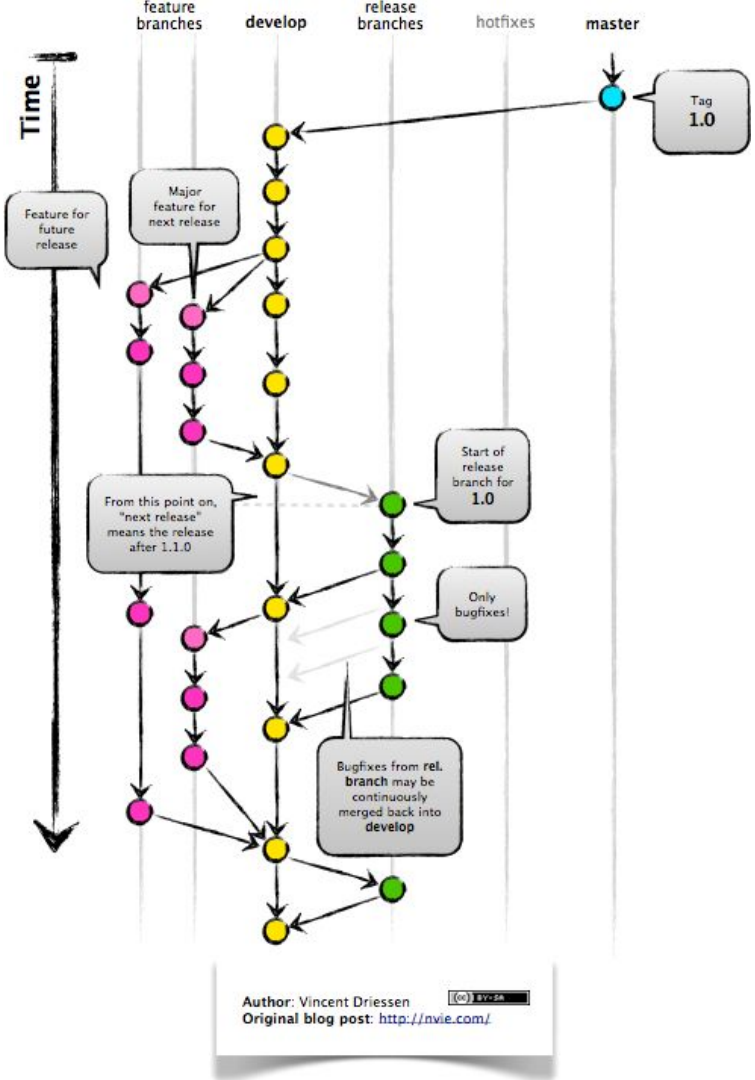
Gitflow


feature branches
develop
release branches
hotfixes
master
Time
Tag 1.0
Feature for future release
Major feature for next release
Author: Vincent Driessen
Original blog post: http://nvie.com/

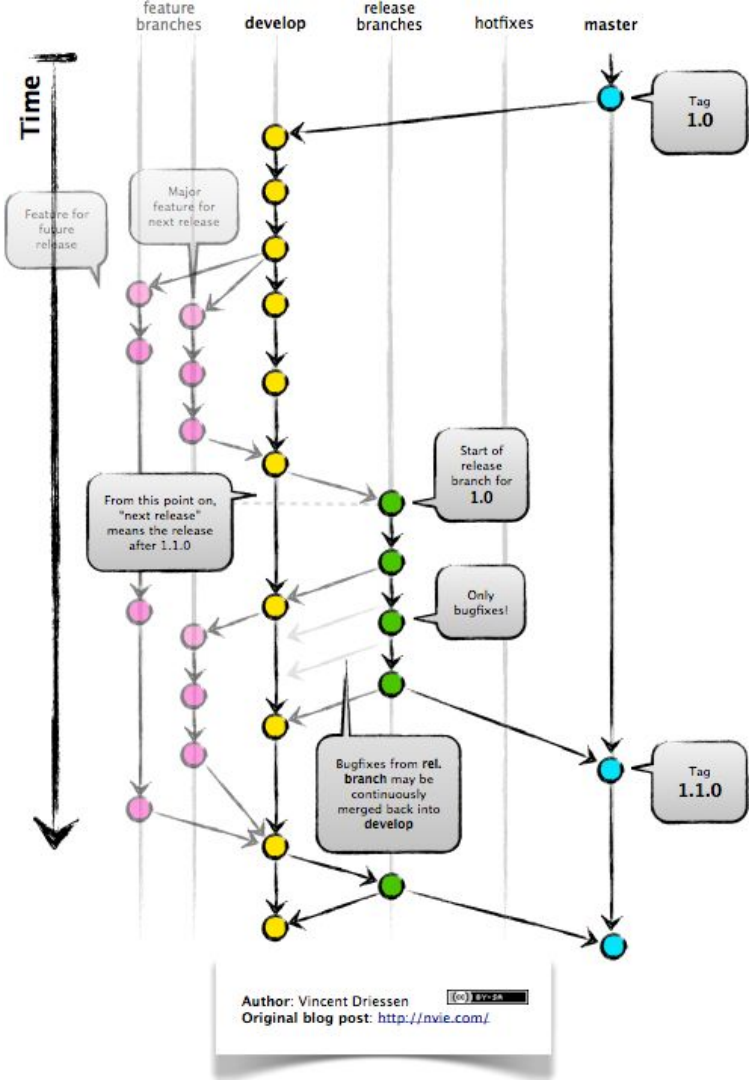
Gitflow


feature branches
develop
release branches
hotfixes
master
Time
Tag 1.0
Feature for future release
Major feature for next release
Author: Vincent Driessen
Original blog post: http://nvie.com/

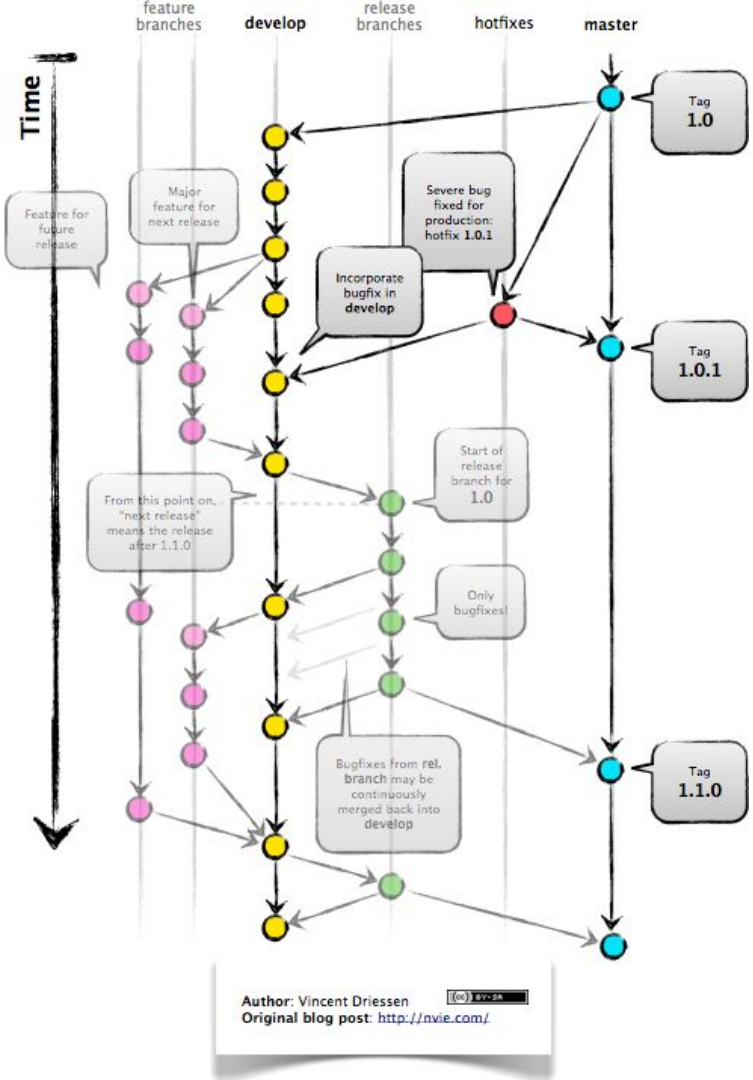
Gitflow


feature branches
develop
release branches
hotfixes
master
Time
Tag 1.0
Feature for future release
Major feature for next release
From this point on, "next release" means the release after 1.1.0
Start of release branch for 1.0
Only bugfixes!
Bugfixes from rel. branch may be continuously merged back into develop
Author: Vincent Driessen
Original blog post: http://nvie.com/

Gitflow


feature branches
develop
release branches
hotfixes
master
Time
Tag 1.0
Feature for future release
Major feature for next release
Start of release branch for 1.0
From this point on, "next release" means the release after 1.1.0
Only bugfixes!
Bugfixes from rel. branch may be continuously merged back into develop
Tag 1.1.0
Author: Vincent Driessen
Original blog post: http://nvie.com/

Gitflow


feature branches
develop
release branches
hotfixes
master
Time
Tag 1.0
Feature for future release
Major feature for next release
Severe bug fixed for production: hotfix 1.0.1
Incorporate bugfix in develop
Tag 1.0.1
Start of release branch for 1.0
From this point on, "next release" means the release after 1.1.0
Only bugfixes!
Bugfixes from rel. branch may be continuously merged back into develop
Tag 1.1.0
Author: Vincent Driessen
Original blog post: http://nvie.com/

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

Do’s and Don’ts

DO

- Keep branches focused on a single task
- Separate changes in small enough commits
- Write meaningful commit messages (title + body)
- Check and test the changes you are committing

DON’T

- Don’t change spacing in an entire file (check your editor settings for tabs)
- Don’t let branches linger for months without keeping them updated
  - `git merge origin/develop`
- Don’t include large and/or binary files

How I work… keeping branches short and sweet

- … code code code …
- `git branch feature-first-new-geometry`
- `git checkout feature-first-new-geometry`
- `git add; git commit`
- `git branch feature-unrelated-new-geometry develop`
- `git checkout feature-unrelated-new-geometry`
- … code code code …
- `git add; git commit`
- `git checkout feature-first-new-geometry`
- `git merge feature-unrelated-new-geometry`

How I work… stashing

- … code code code …
- git checkout feature-new-geometry
- … argh! checkout fails because “would overwrite file”
- git stash push
- git checkout feature-new-geometry
- git stash pop
- … now I can fix the conflicts

Other useful bits

- gitk is generally installed
- git-cola is very useful
- `git add -p, git commit -p`: add chunks of files, not entire file (patch)
-