Introduction to Git Version Control System

Junru (Bill) Zhong

March 1st, 2018

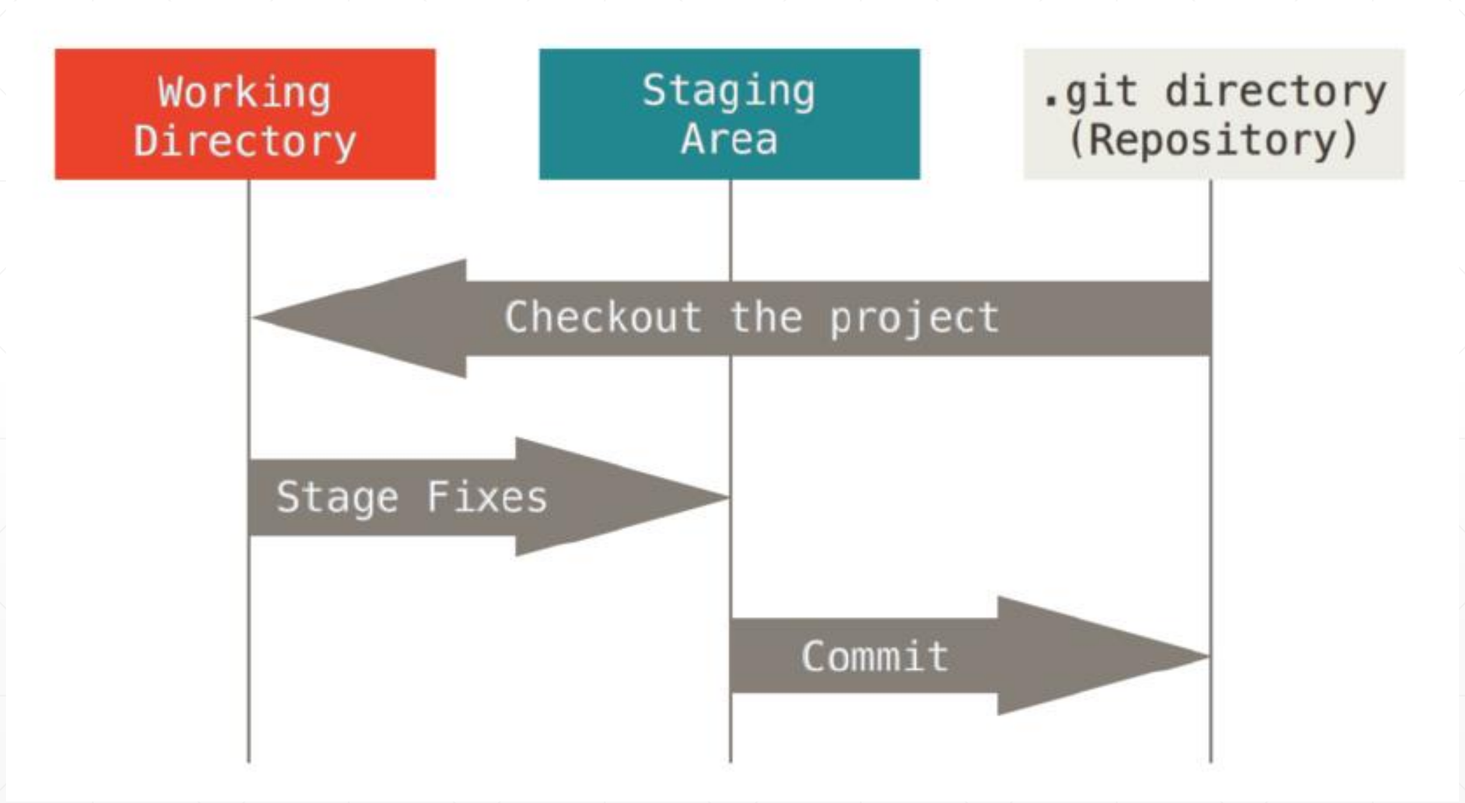
Ver. 2.0 – 28/02/2018

Contents

What is Git?

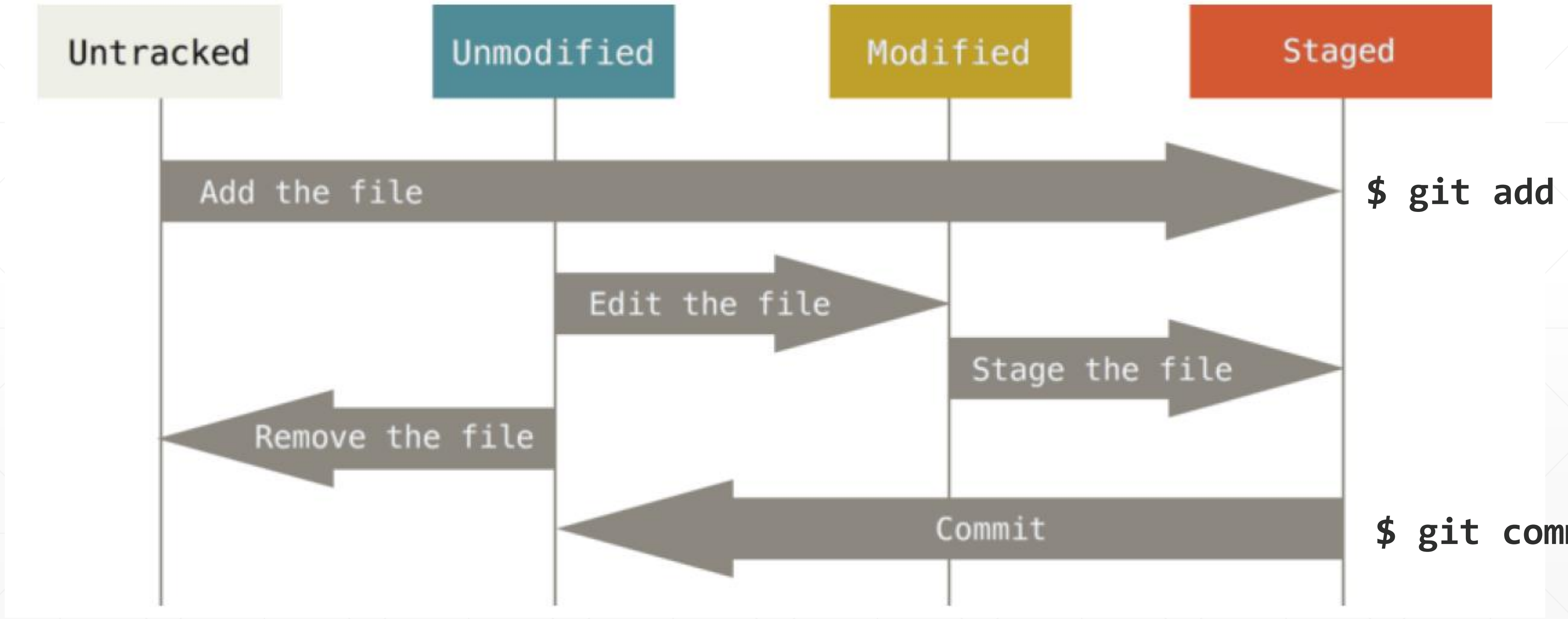
- A distributed version control system
- Developed by Linux kernel community in 2005
- Characteristics
  - Fast
  - Simple design
  - Strong support to non-linear development

Local Git Workflow



Credit: S. Chacon & B. Straub, Pro Git (2nd Edition), 2014

Status of Files



```
$ git add
```

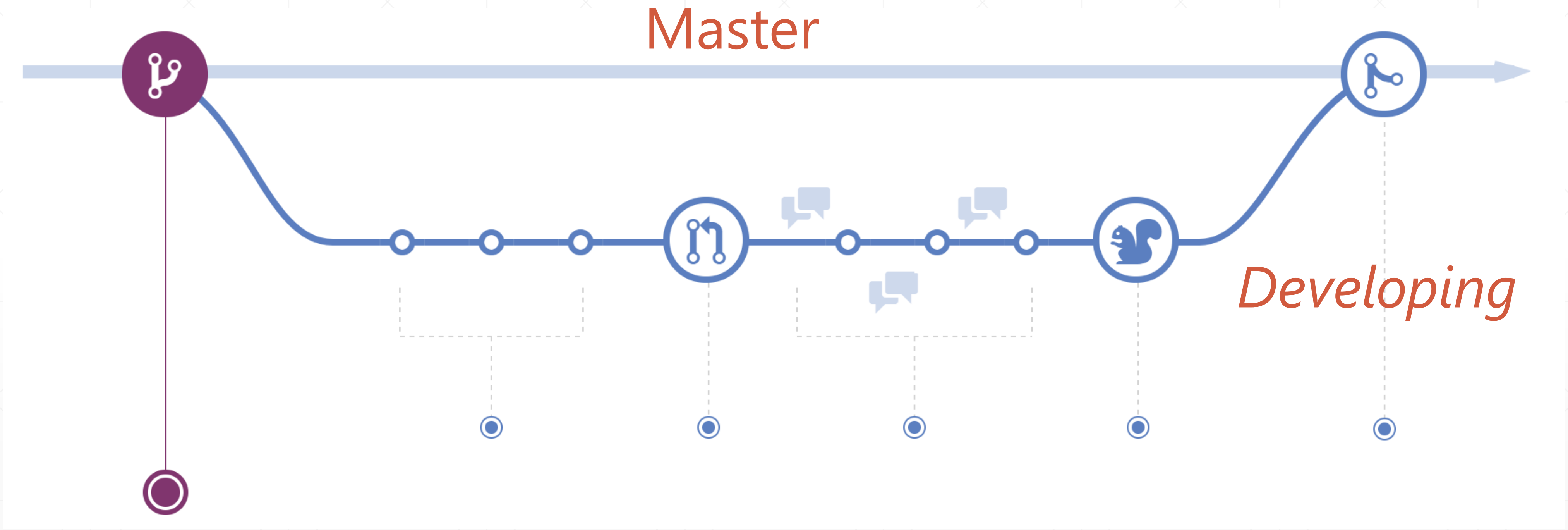
```
$ git commit
```

Credit: S. Chacon & B. Straub, Pro Git (2nd Edition), 2014

Basic Operations (local)

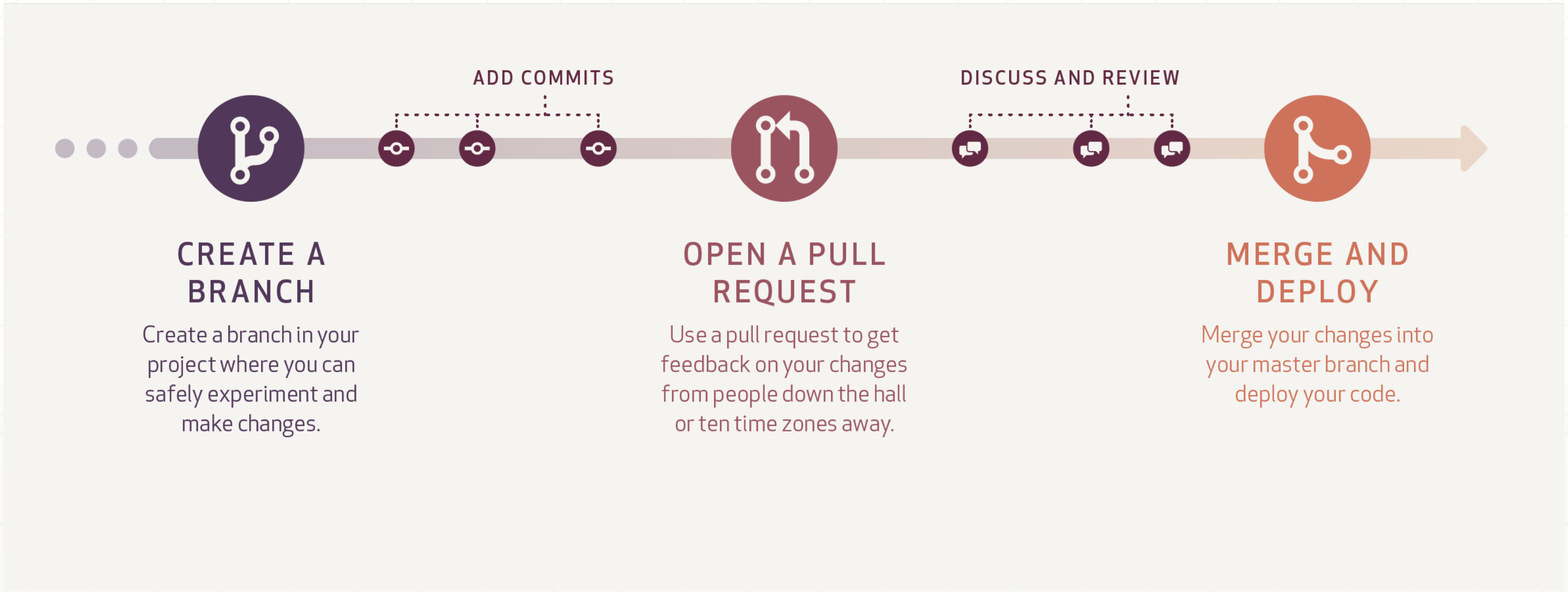
- **Initialize a repository:**
  - `$ git init`
- **Stage files**
  - `$ git add #file name, use * or . for all files`
- **Commit changes**
  - `$ git commit –m “Your message”`
- **Check status**
  - `$ git status`

Using Branches



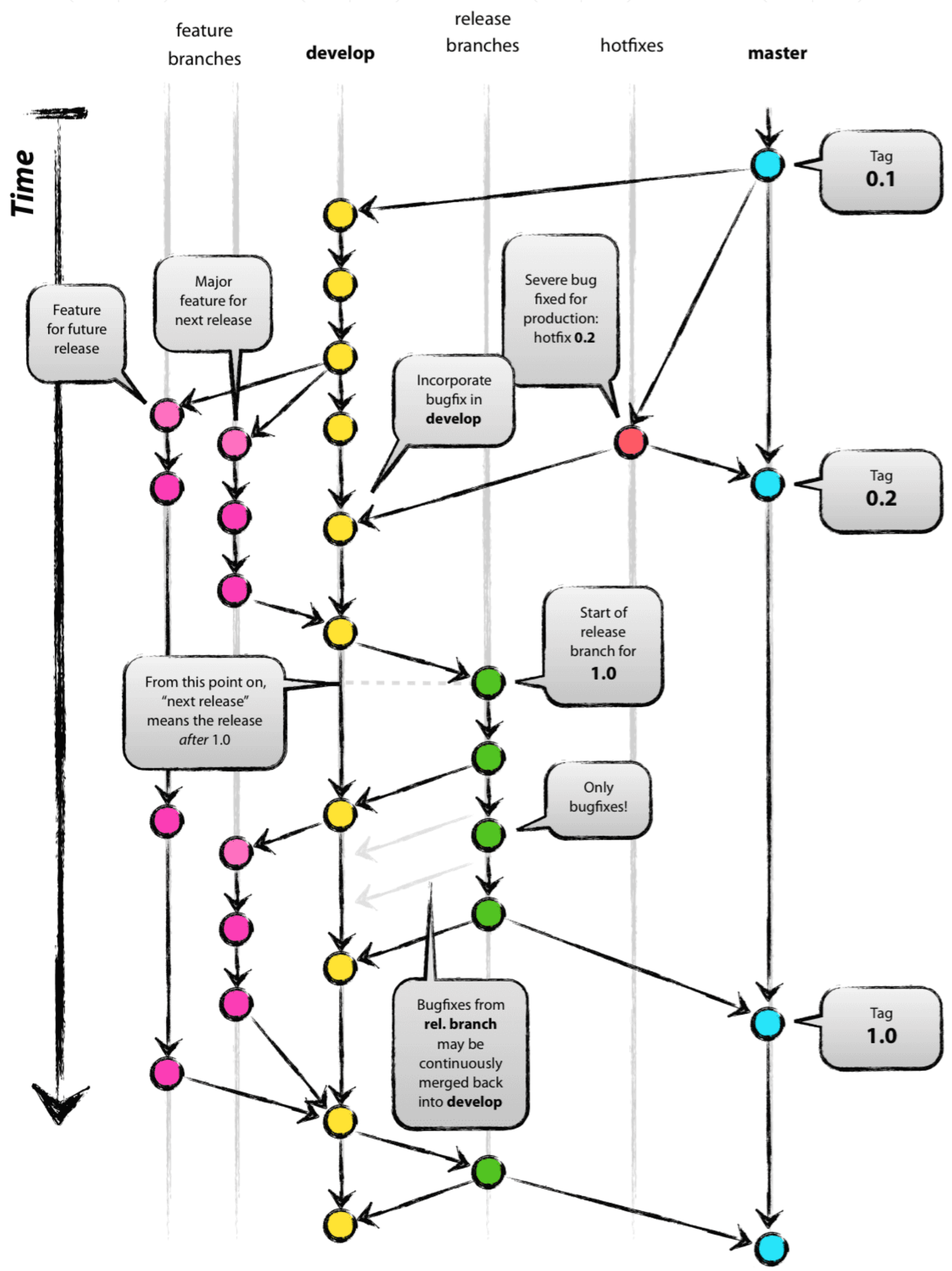
Credit: *https://guides.github.com/introduction/flow/*

Using Branches (Cont'd)


ADD COMMITS
DISCUSS AND REVIEW
CREATE A BRANCH
Create a branch in your project where you can safely experiment and make changes.
OPEN A PULL REQUEST
Use a pull request to get feedback on your changes from people down the hall or ten time zones away.
MERGE AND DEPLOY
Merge your changes into your master branch and deploy your code.

Credit: https://guides.github.com/introduction/flow/

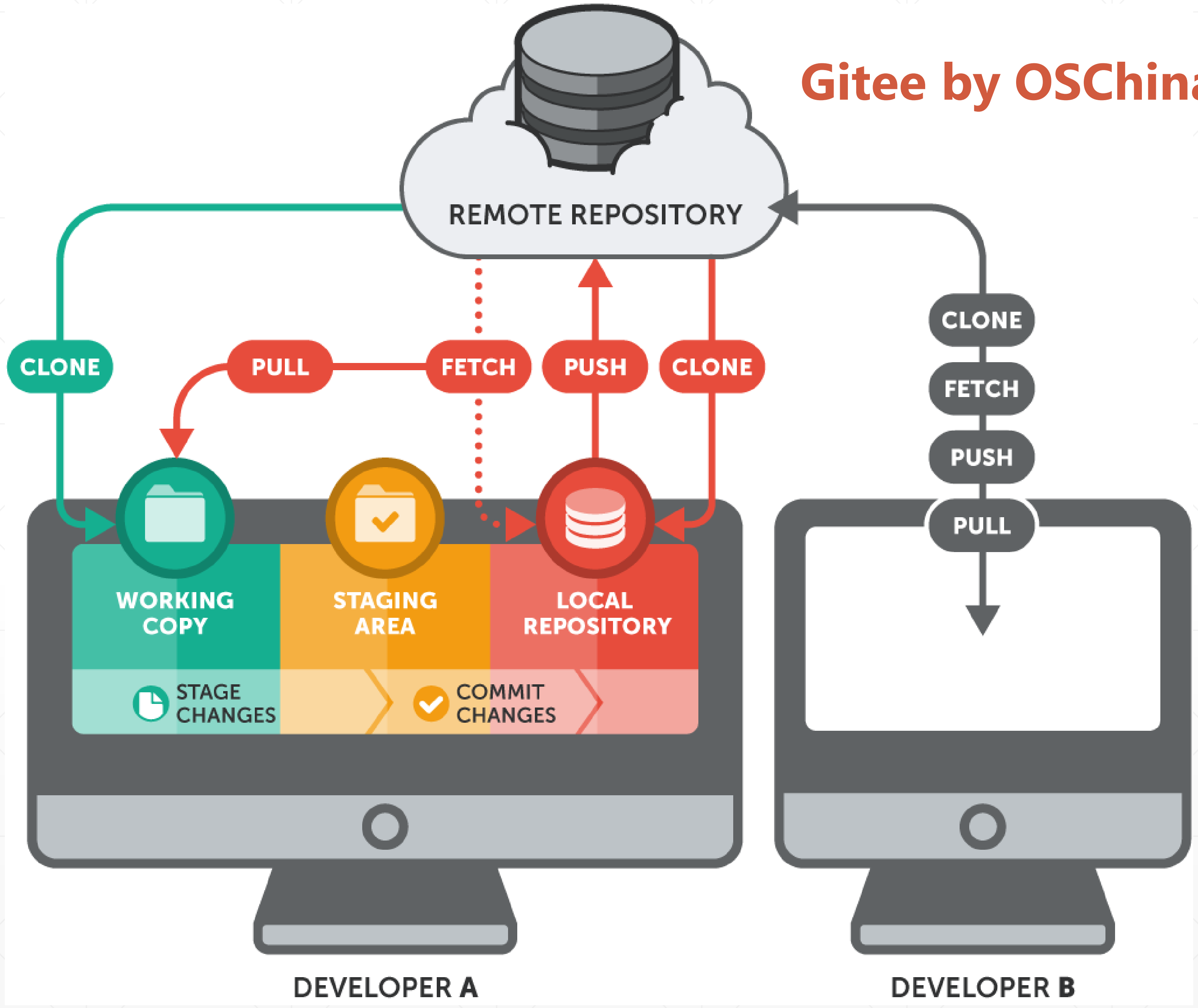
Using Branches (Cont’d)


feature branches
develop
release branches
hotfixes
master
Time
Tag 0.1
Feature for future release
Major feature for next release
Severe bug fixed for production: hotfix 0.2
Incorporate bugfix in develop
Tag 0.2
Start of release branch for 1.0
From this point on, "next release" means the release after 1.0
Only bugfixes!
Bugfixes from rel. branch may be continuously merged back into develop
Tag 1.0

Credit: http://nvie.com/img/git-model@2x.png

Working with Git Remotely

Gitee by OSChina (开源中国社区)

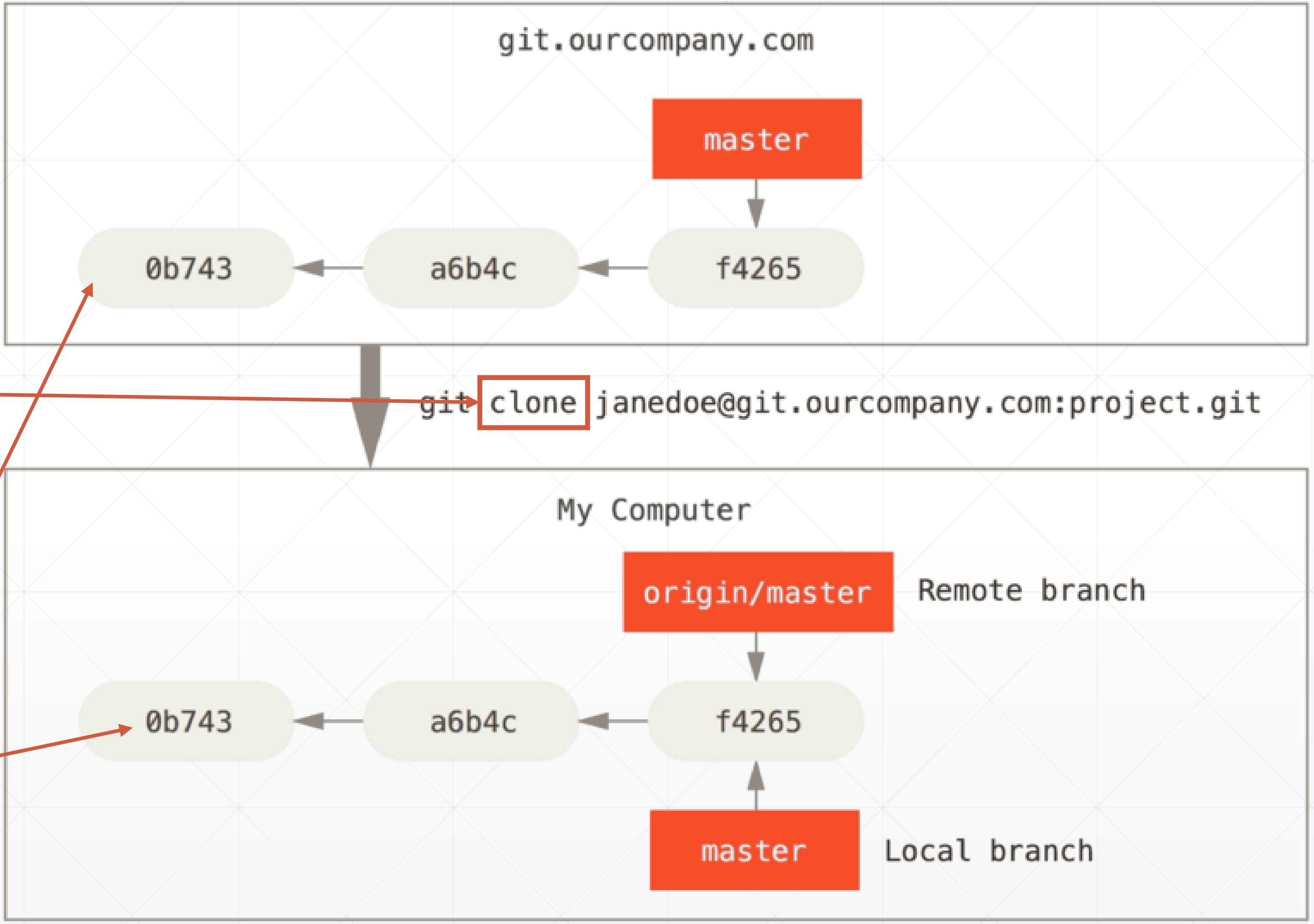


Credit: https://www.git-tower.com/learn/git/ebook/en/command-line/remote-repositories/introduction

Remote Branches

Clone: An exactly same copy

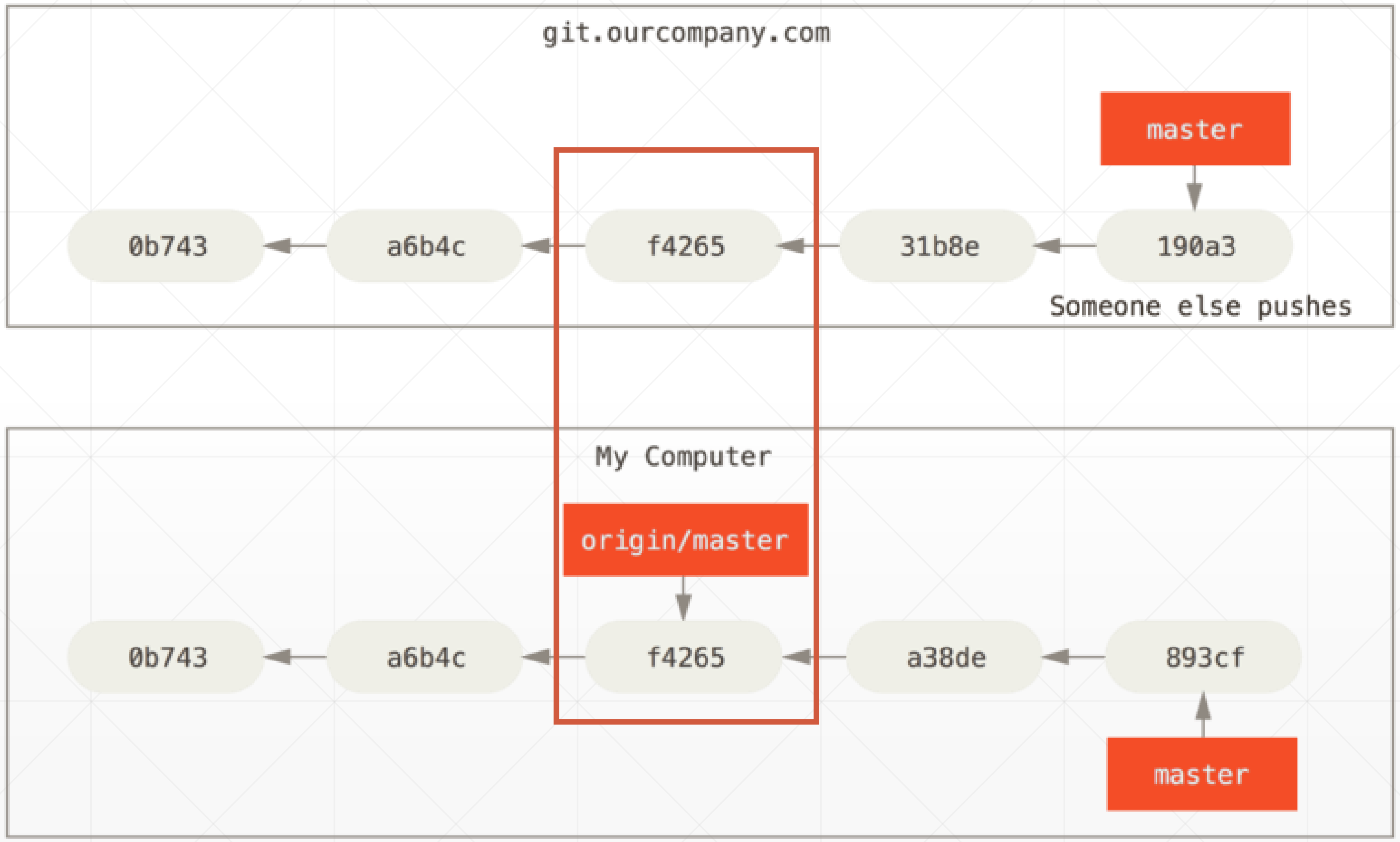
Commits



```
git clone janedoe@git.ourcompany.com:project.git
```

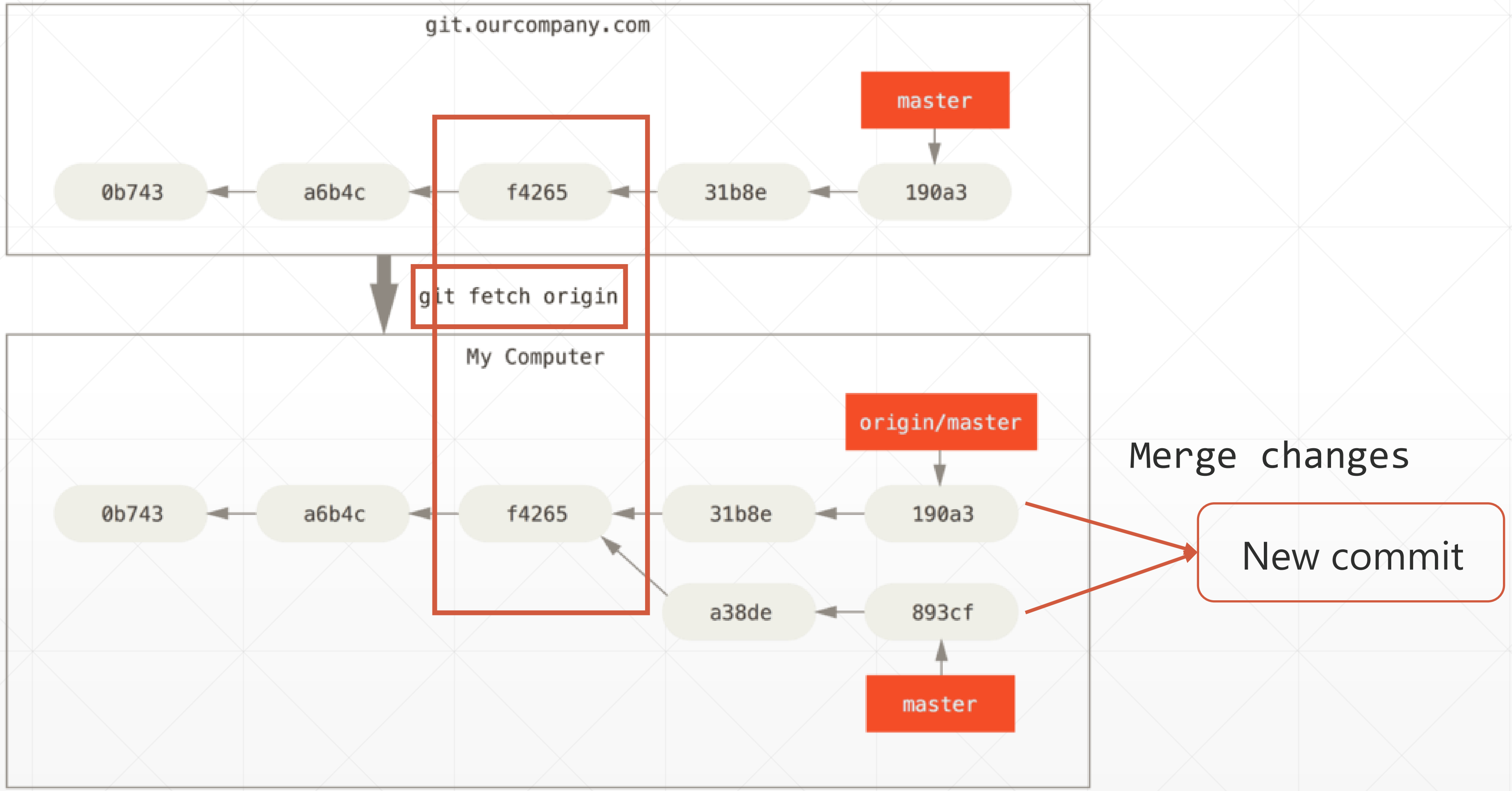
Credit: https://git-scm.com/book/en/v2/Git-Branching-Remote-Branches

Handling *Fast-forward* Problem



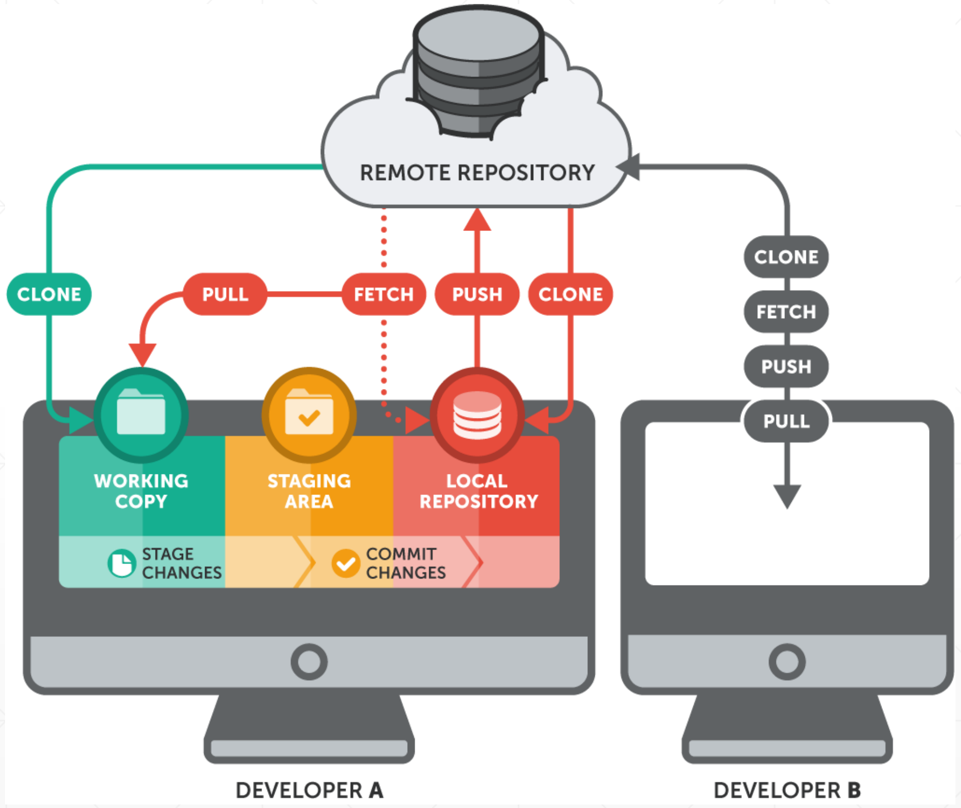
Credit: https://git-scm.com/book/en/v2/Git-Branching-Remote-Branches

Handling *Fast-forward* Problem (Cont'd)



Credit: https://git-scm.com/book/en/v2/Git-Branching-Remote-Branches

Basic Operations (with remote)

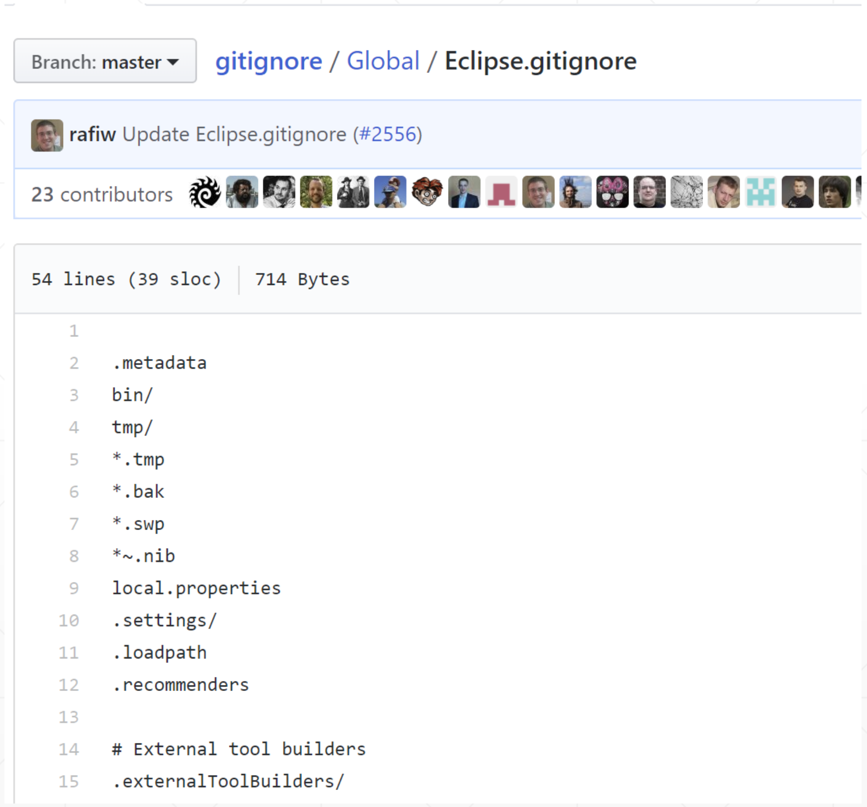


- **From remote to local:**
  - `$ git clone #from a remote repo`
  - `$ git fetch #changes`
  - `$ git pull`

    `#Fetch changes and merge to local repo.`
- ***After staged, committed your changes:***
  - `$ git push #local changes to remote`
- **Add a remote repo. to a local repo.:**
  - `$ git remote add #to a local repo.`

Useful Tips

The .gitignore File


Branch: master
gitignore / Global / Eclipse.gitignore
rafiw Update Eclipse.gitignore (#2556)
23 contributors
54 lines (39 sloc) | 714 Bytes

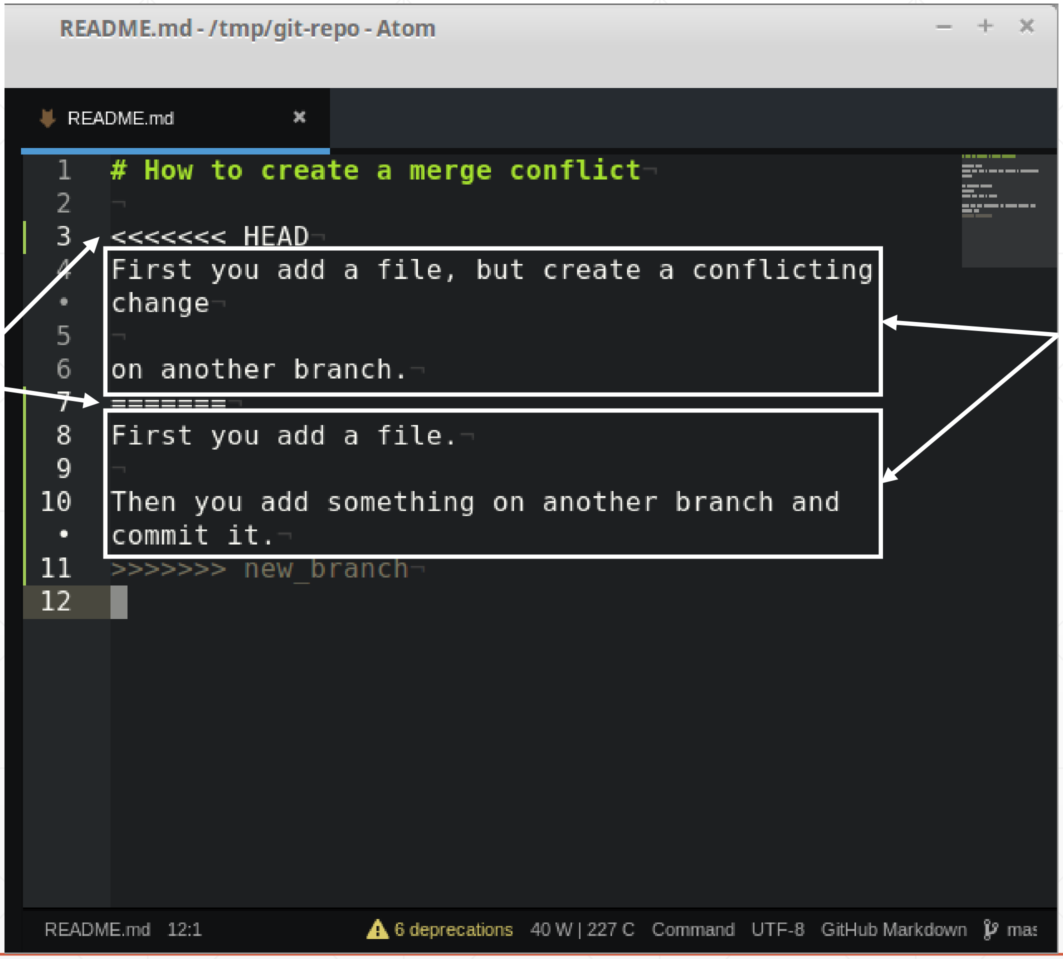
```

.metadata
bin/
tmp/
*.tmp
*.bak
*.swp
*~.nib
local.properties
.settings/
.loadpath
.recommenders

# External tool builders
.externalToolBuilders/
```

- **Tells Git to ignore some files.**
- **Avoid conflicts.**
- Download templates from:
  - https://github.com/github/gitignore
  - according to your source languages, editors, IDEs.
- Put it in your tracked directories with file name "`.gitignore`".

Handling Conflicts

- Conflict changes from different branches
  - e.g. Different modifications in a same line.
- Git don’t know how to merge changes.
- Requires human to make choices.
  - Accept changes from one branch.

Handling Conflicts (Cont'd)



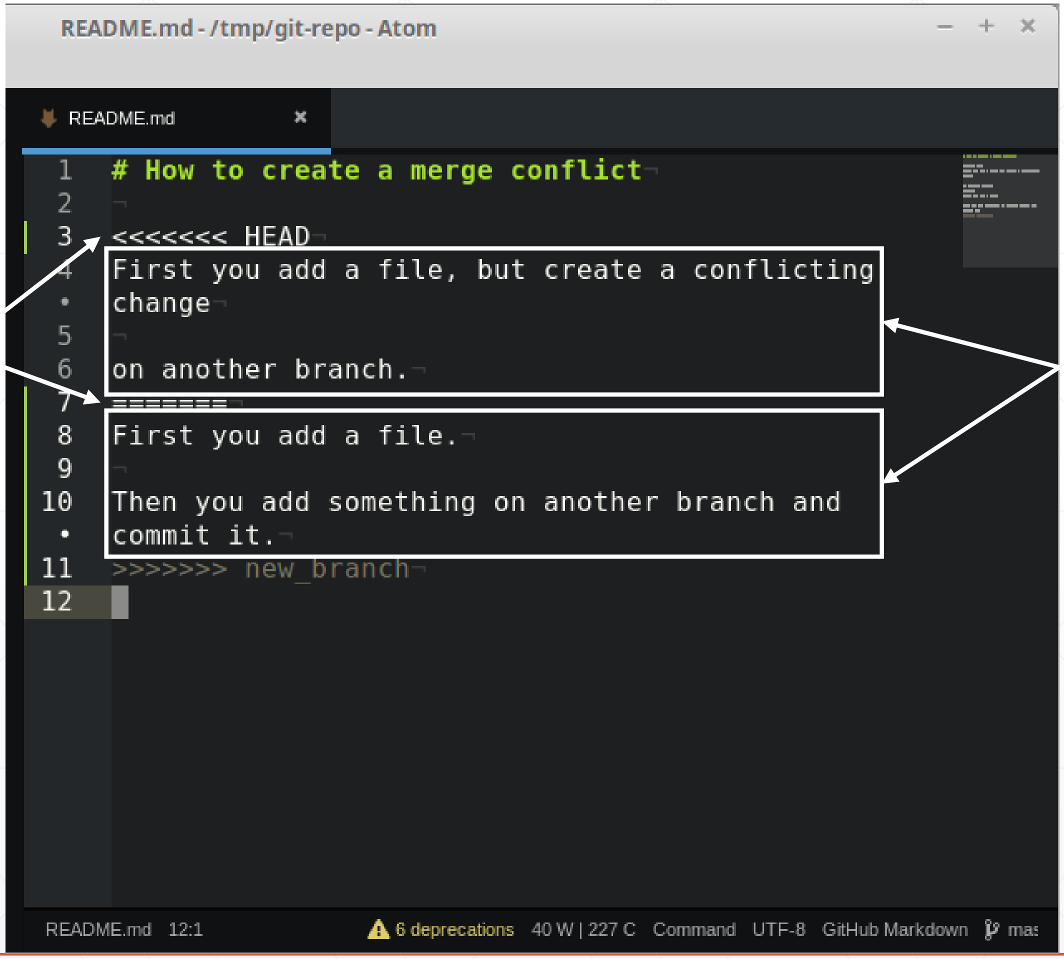
```
# How to create a merge conflict

<<<<<<< HEAD
First you add a file, but create a conflicting
change

on another branch.
=======
First you add a file.

Then you add something on another branch and
commit it.
>>>>>>> new_branch
```

Notations added by Git

Changes from all branches

Credit: *http://jonathanmh.com/wp-content/uploads/2015/06/temporary-git-merge-conflict-files.png*

Handling Conflicts (Cont'd)



```
# How to create a merge conflict

<<<<<<< HEAD
First you add a file, but create a conflicting
change

on another branch.
=======
First you add a file.

Then you add something on another branch and
commit it.
>>>>>>> new_branch
```

Delete all notations

Leave the correct change

Delete change you don't want

Commit and push again.

Credit: http://jonathanmh.com/wp-content/uploads/2015/06/temporary-git-merge-conflict-files.png

Useful Tips

- Always create repository with a **`gitignore`** file.
- Always **`pull`** before modify files.
- Always **`commit`** with **meaningful** information.
- Try to make use of **branches** by **`forking`**.
- Try to work with git by **command line**.

References & Useful Resources

- GitHub Guides: https://guides.github.com/
- Git Cheat sheet: https://services.github.com/on-demand/resources/cheatsheets/
- GitHub Help: https://help.github.com/
- Gitee Documentation (Chinese): http://git.mydoc.io/
- The Book *Pro Git*: https://git-scm.com/book/en/v2
- Handling Conflicts: https://stackoverflow.com/questions/161813/how-to-resolve-merge-conflicts-in-git