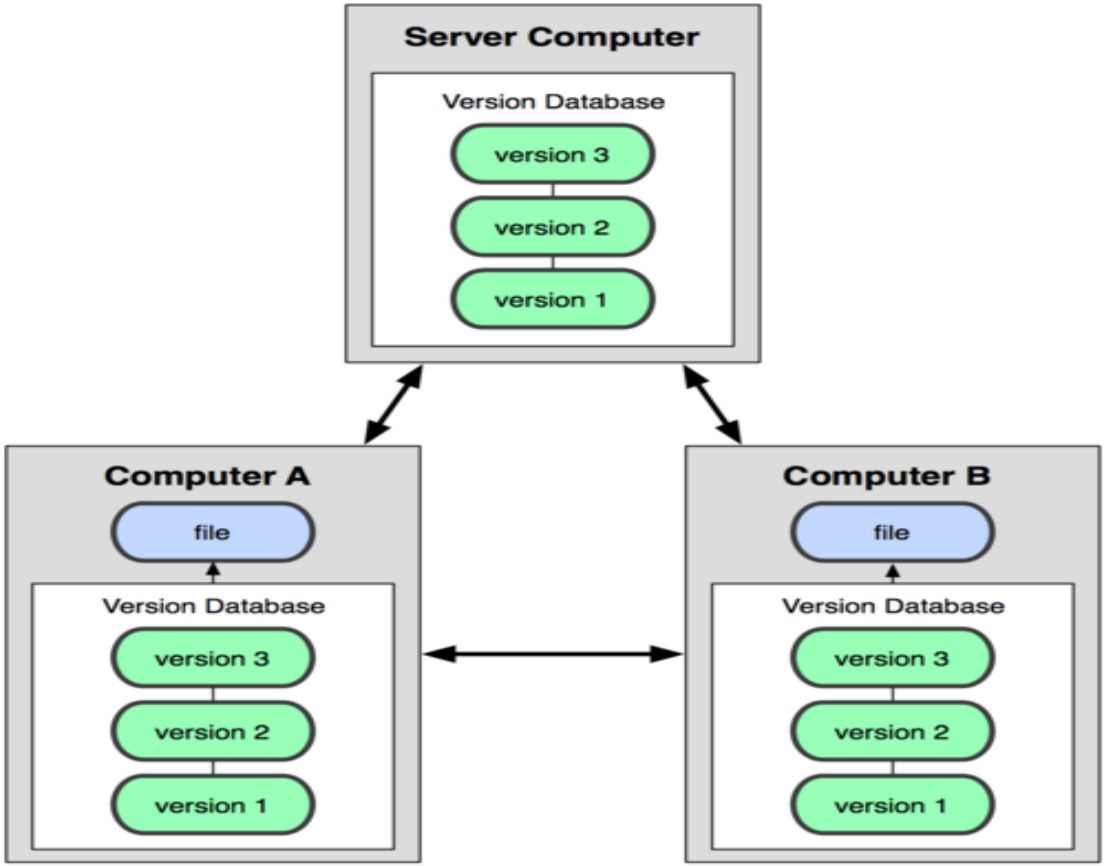
Git Basic

Outline

1. Overview
2. GIT command
3. Insight GIT
4. Branch strategies
5. Note

+ Distributed Version Control Systems



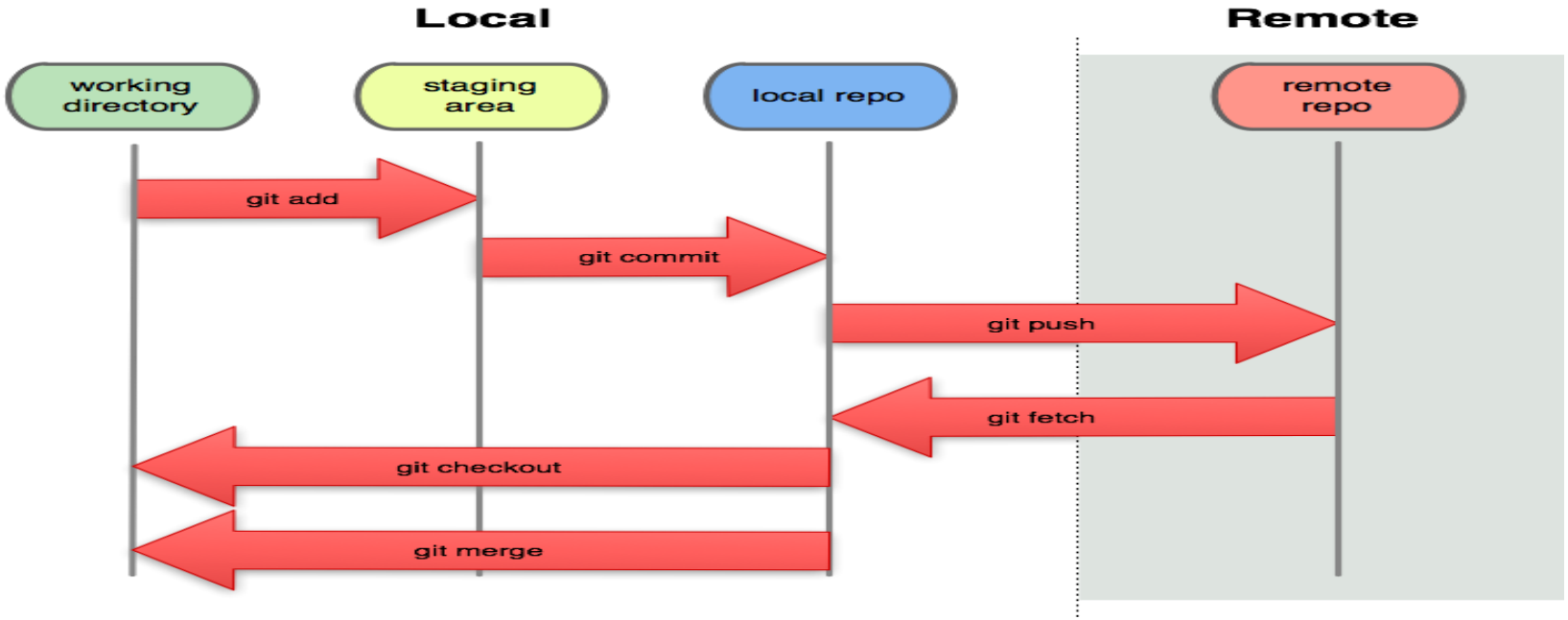
Overview

Term	Definition
Repository	Kho chứa mã nguồn cùng lịch sử thay đổi
Branch	Mỗi nhánh gắn liền với một ngữ cảnh
Tag	Đánh dấu một nhánh để dễ nhớ, so sánh
Commit	Xác nhận, cập nhật thay đổi vào repository
Merge	Trộn

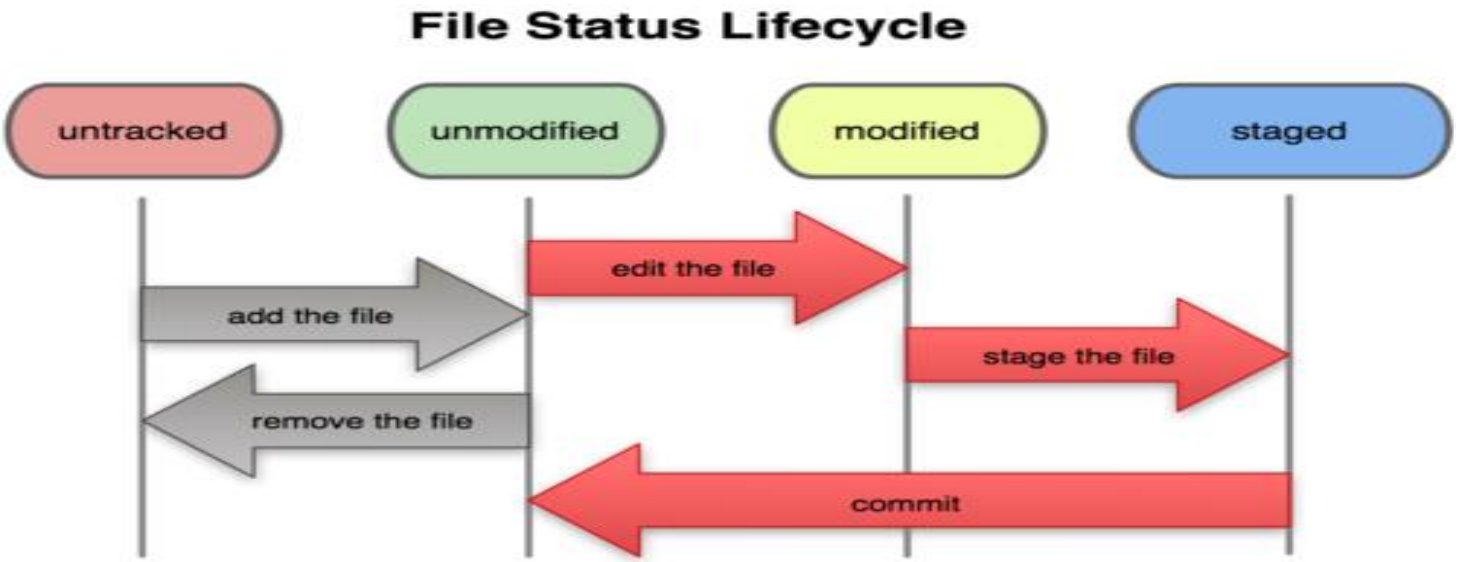
Overview

Everything is local

- working directory = one version of the project
- local repo = metadata + object database
- staging area(or index) = control what parts of working directory tree to go into the repository on the next commit operation



Overview



GIT command

- Tell git who you are
  $ git config --global user.name "Your Name Comes Here"
  $ git config --global user.email you@yourdomain.example.com

- Create new project
  - git init
  - git add .
  - git commit

- Making changes
  - Edit file
  - git add file1 file2 ….
  - git diff –cached
  - git status
  - git commit

GIT command

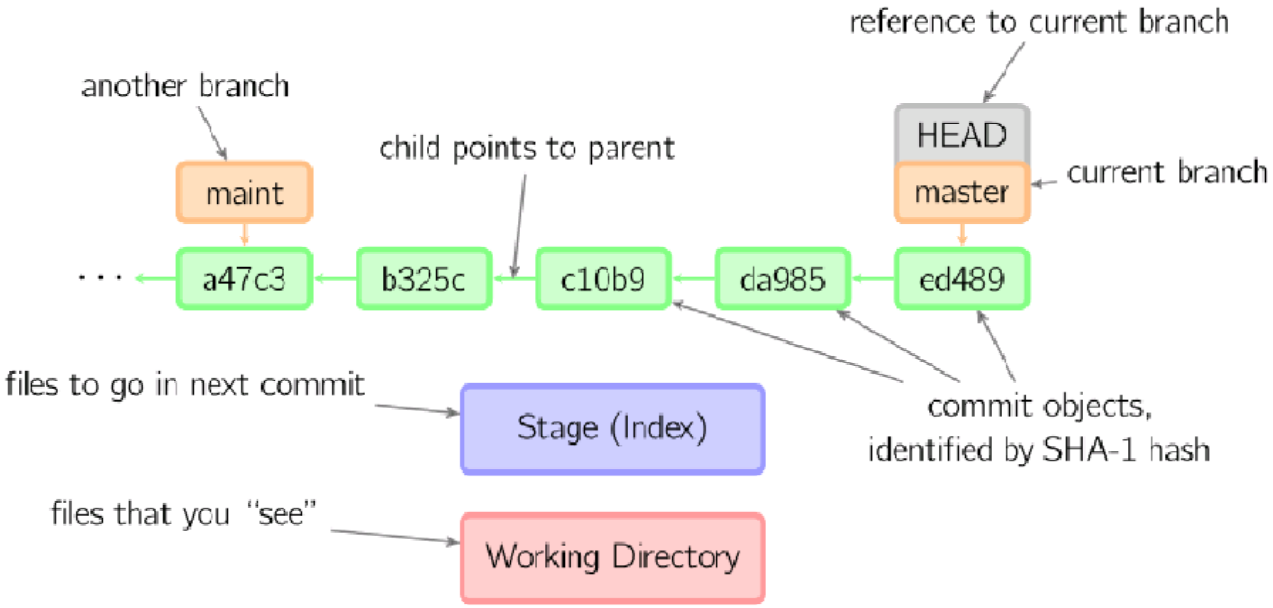
- Viewing project history
  - git log
  - git log –stat –summary

- Managing branches
  - git branch experimental (create branch)
  - git branch (list branch)
  - git checkout experimental (switch to branch)
  - git merge experimental (merge branch change)

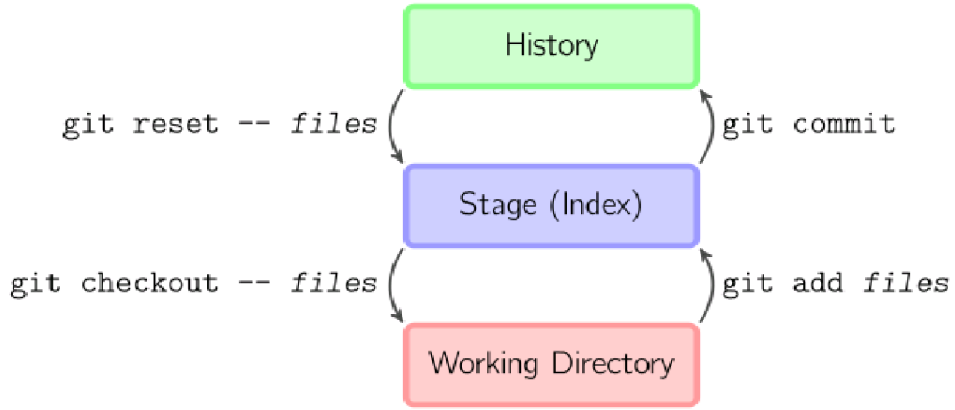
GIT command

- git clone url (get remote repository)
- git pull url (origin) (fetch remote repository change and merge to local branch)
- git fetch url (fetch remote repository change only)
- git push url (push your change to remote repository)

GIT command

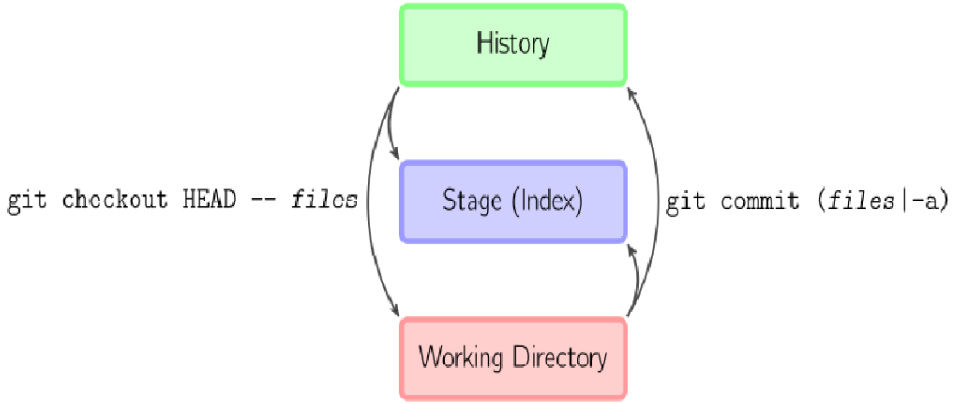


GIT command

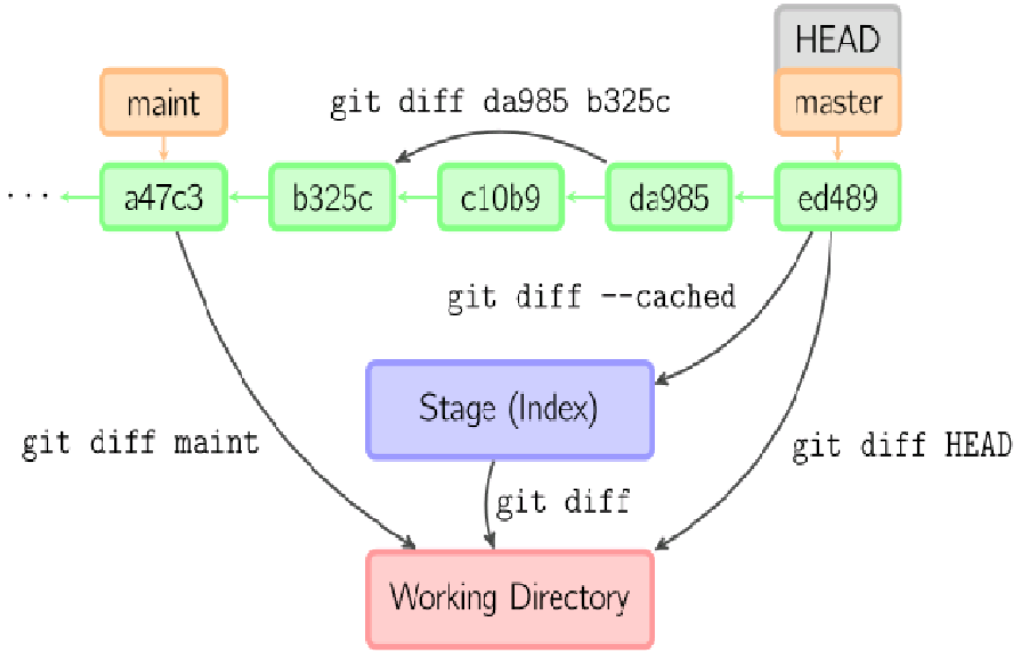


Git add --files copies files to the stage
Git commit save snapshot of the stage as a commit
Git reset –files unstages files, equivalent “undo” git add, git-reset to unstage everything, if have –-hard flag, git will delete newer commit from it
Git reset –-hard 78bcd
Git checkout –files copies files from stage to working directory

GIT command

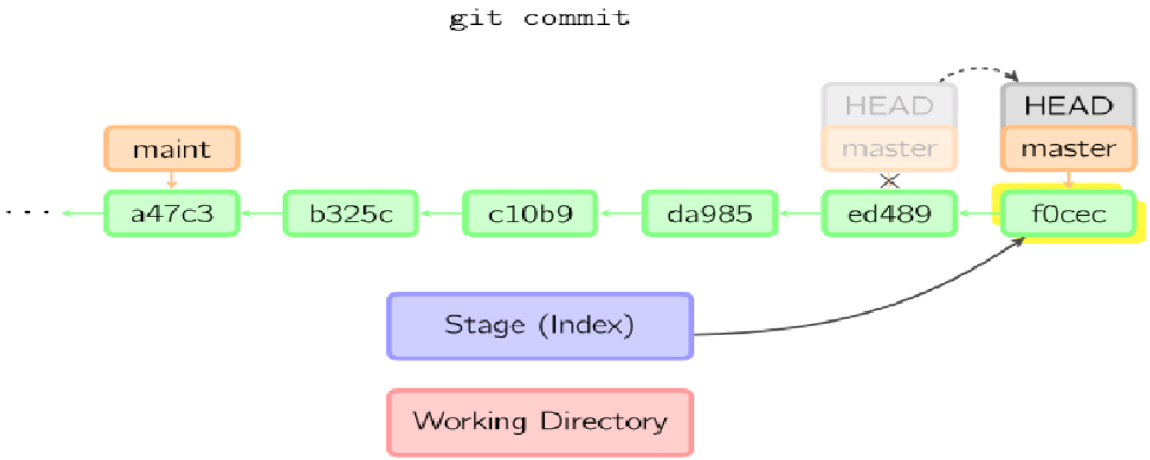


GIT command

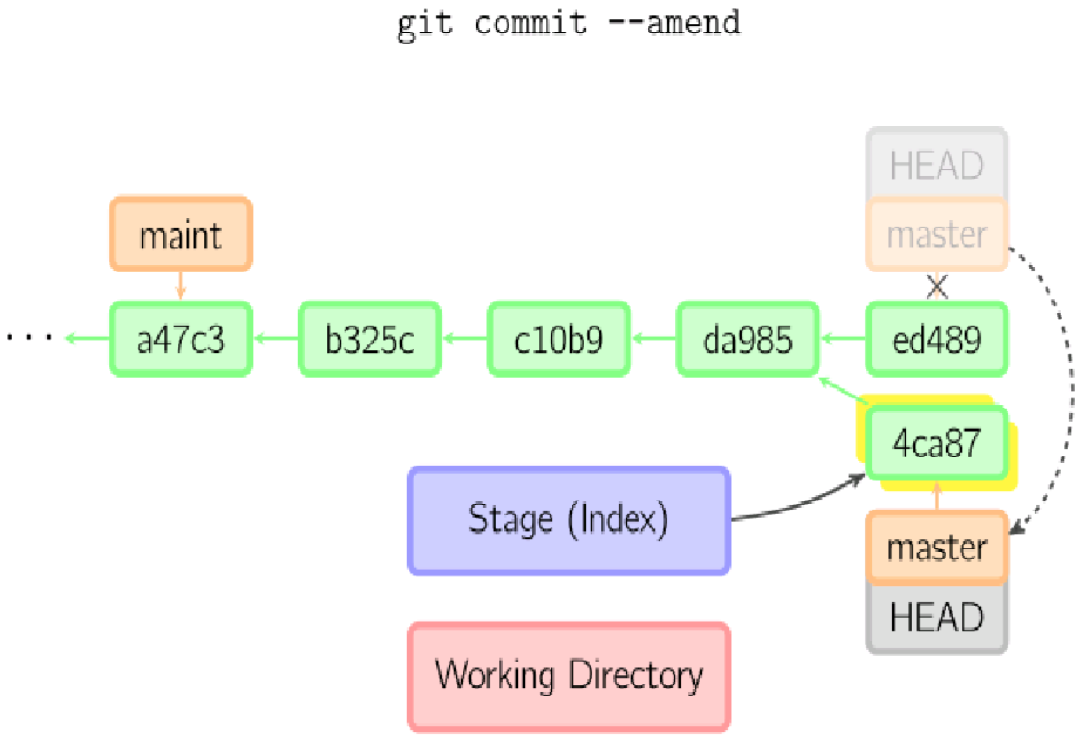


Git diff can create patch file, we can use it for apply patch
$ git diff B A | git apply

GIT command

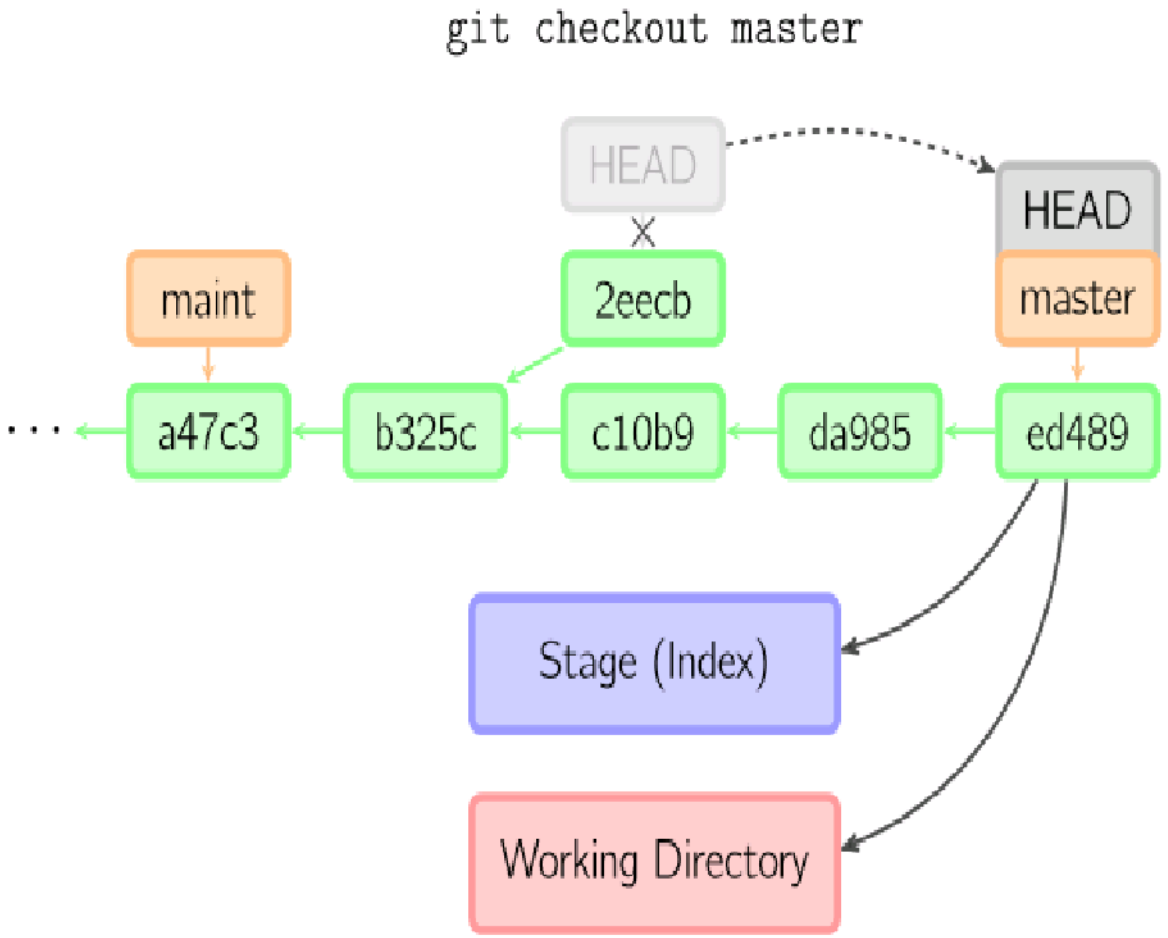


GIT command

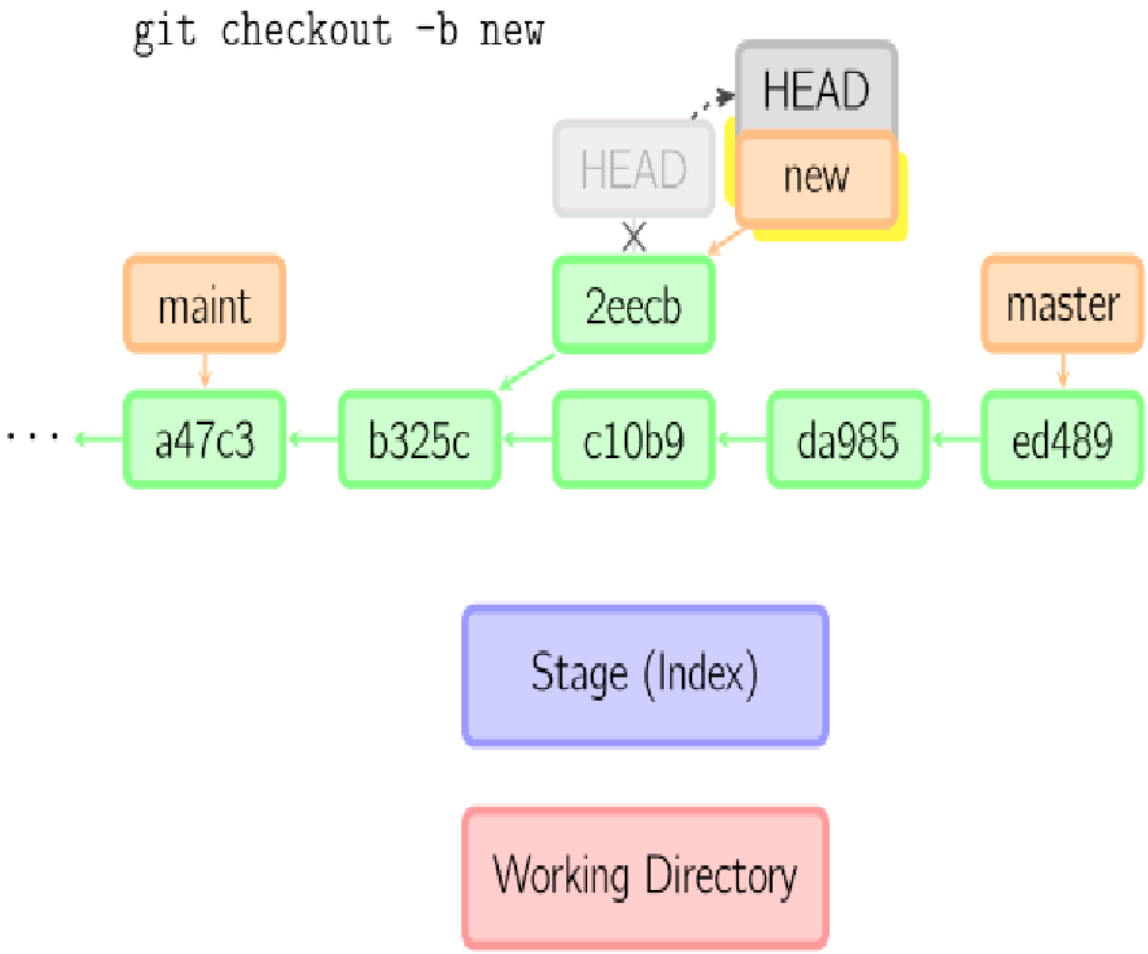


Create new commit with same parent as current commit

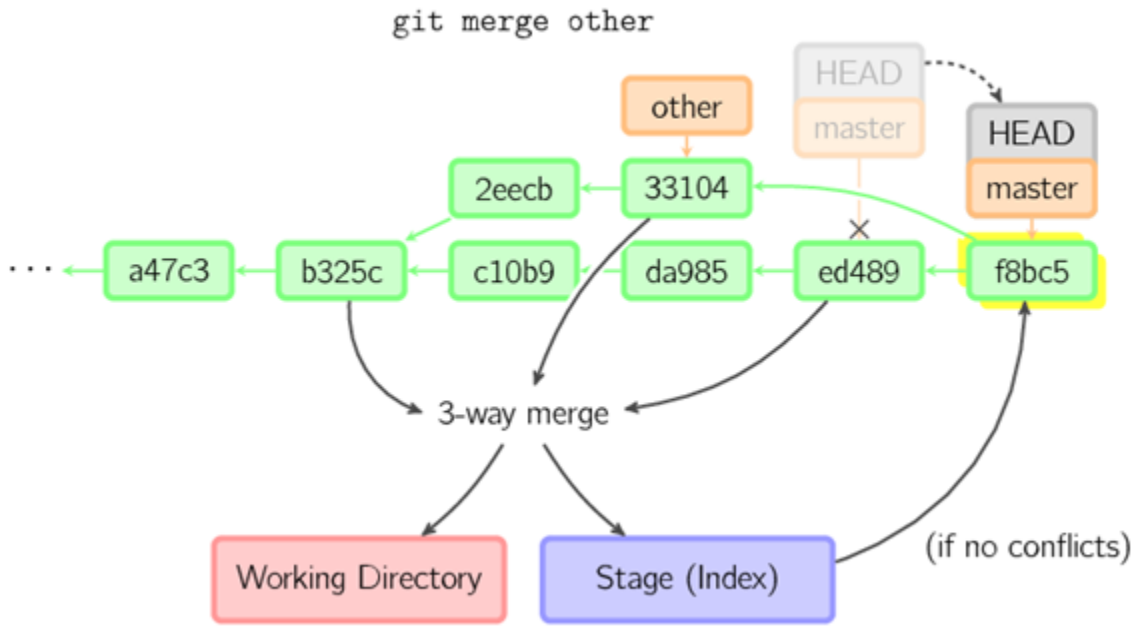
GIT command



GIT command

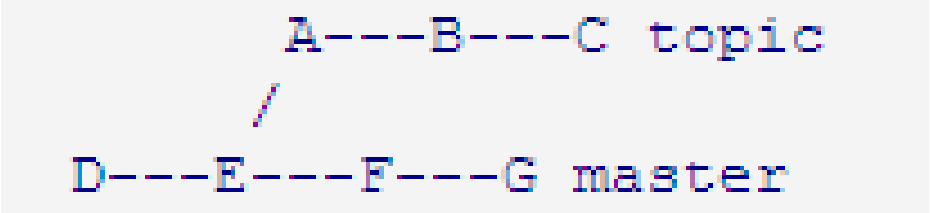


GIT command

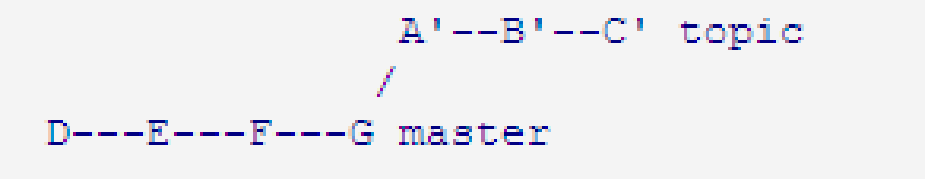


Git rebase: The final result for the source code is the same as with merge but the commit history is cleaner; the history appears to be linear.

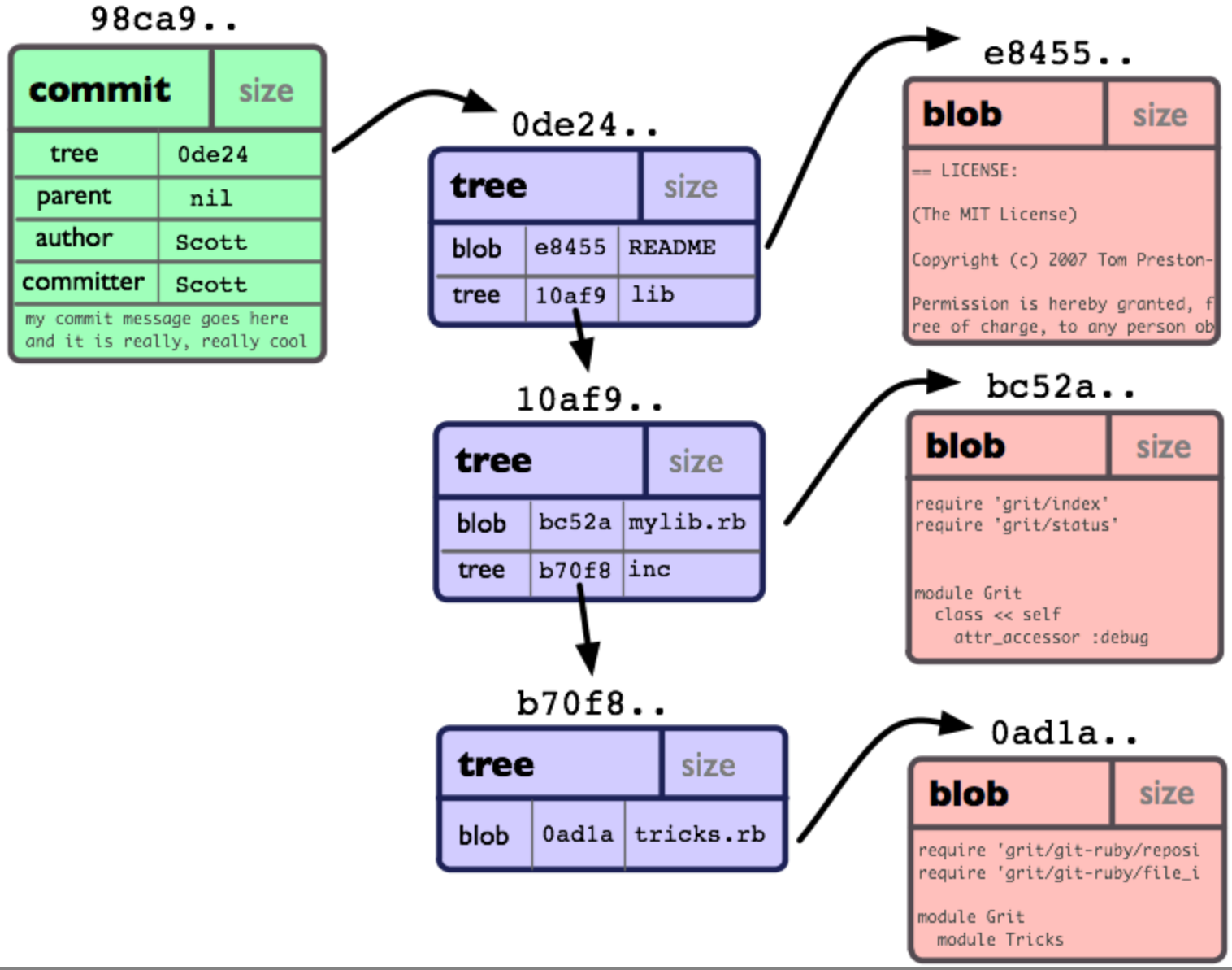
- Before rebase



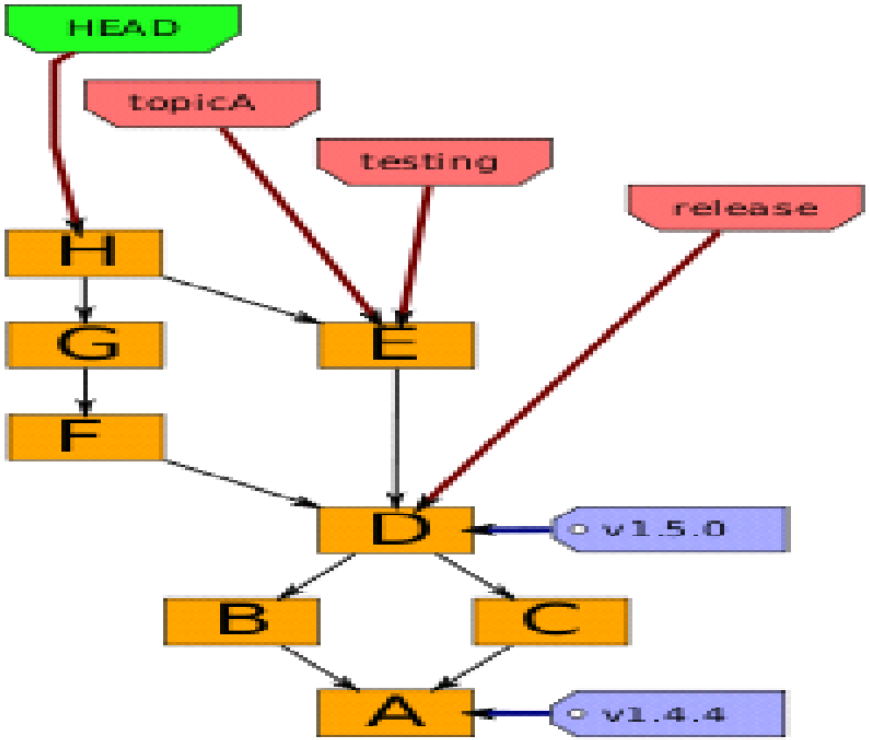
- After rebase



Insight Git

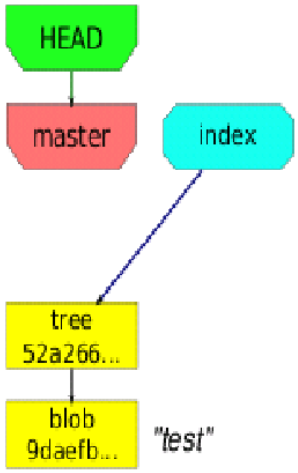


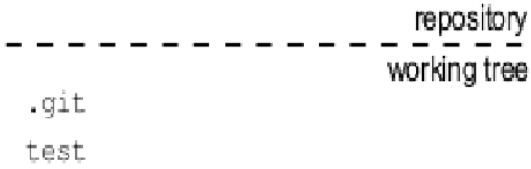
Insight GIT



Insight GIT

```
$ echo test > test
$ git add test
```



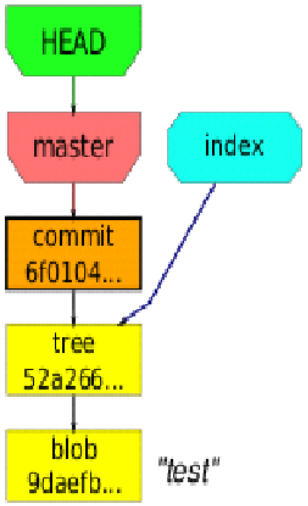


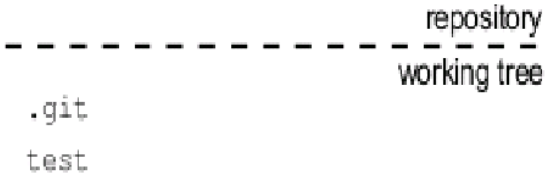
Insight GIT

```
$ echo test > test
$ git add test
```

```
$ git commit -m“test”
Created initial commit 6f01040:  test
1 files changed, 1 insertions(+),
0 deletions(-)

create mode 100644 test
```



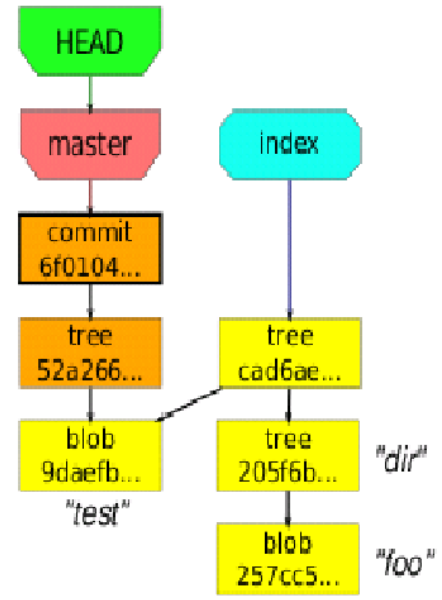


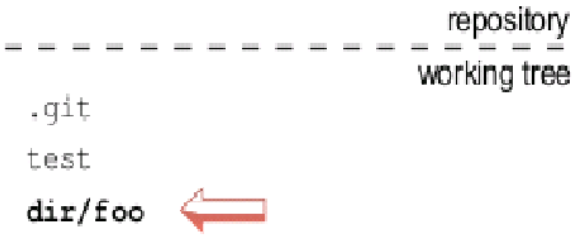
Insight GIT

```
$ echo test > test
$ git add test

$ git commit -m‘‘test’’

$ mkdir dir
$ echo foo > dir/foo
$ git add dir/foo
```





Insight GIT

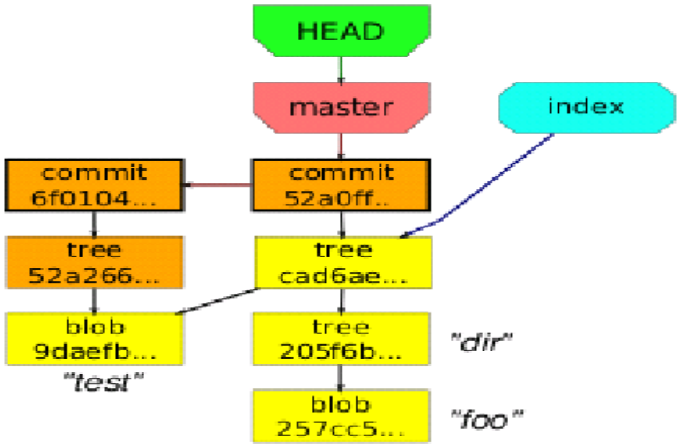
```
$ echo test > test
$ git add test
```

```
$ git commit -m''test''
```

```
$ mkdir dir
$ echo foo > dir/foo
$ git add dir/foo
```

```
$ git commit -m''foo''
Created commit 52a0ff4:  foo
1 files changed, 1 insertions(+),
0 deletions(-)

create mode 100644 dir/foo
```



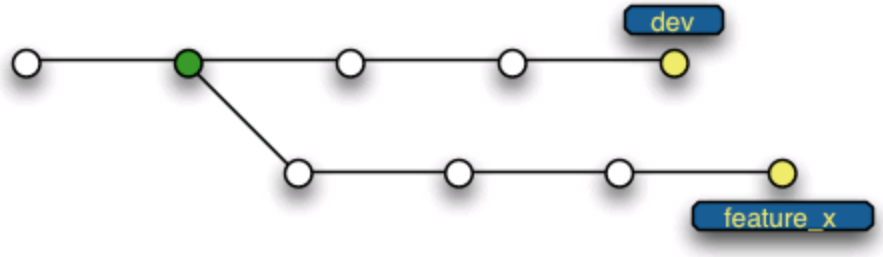
Insight GIT

- Log is commits tree
- Branch is HEAD of tree, current branch will be move ahead when create commit, is simply a movable pointer to one of these commits
- Tag is friend name of Hash.

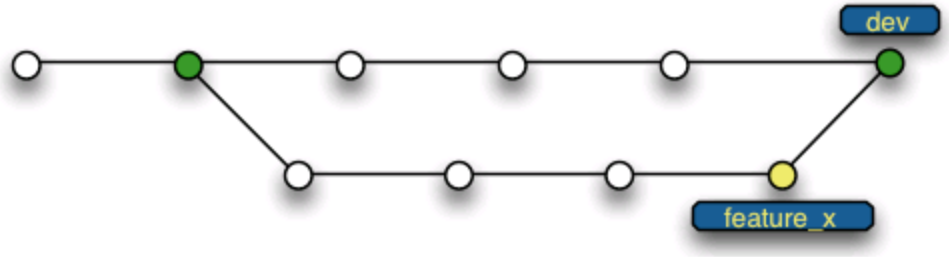
Branch strategies

For single dev:

```
$ git branch feature_x dev
$ git checkout feature_x
# add/commit... add/commit...
```

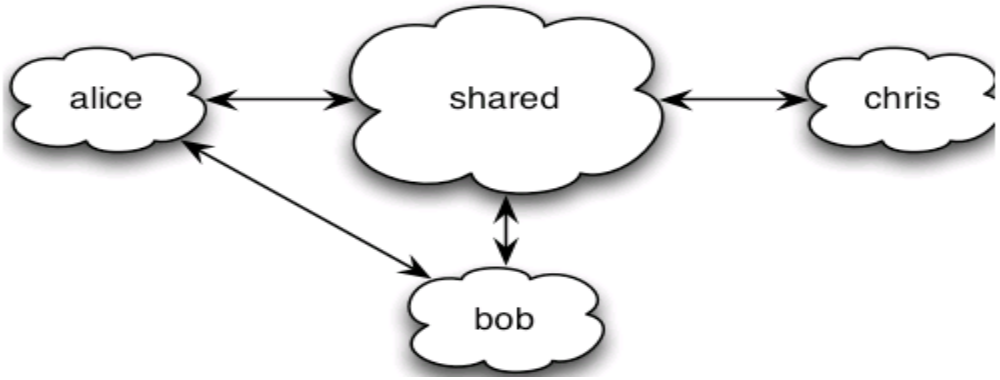


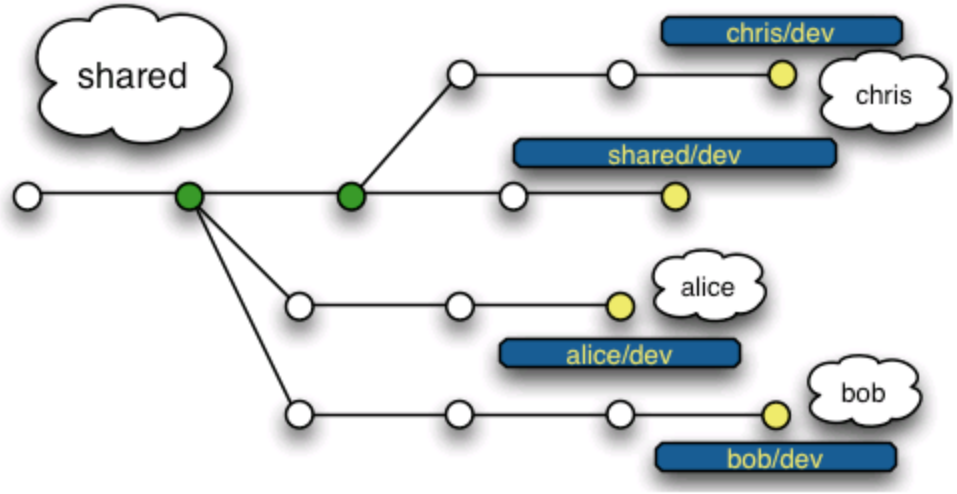

```
$ git checkout dev
$ git merge feature_x
```



Branch strategies

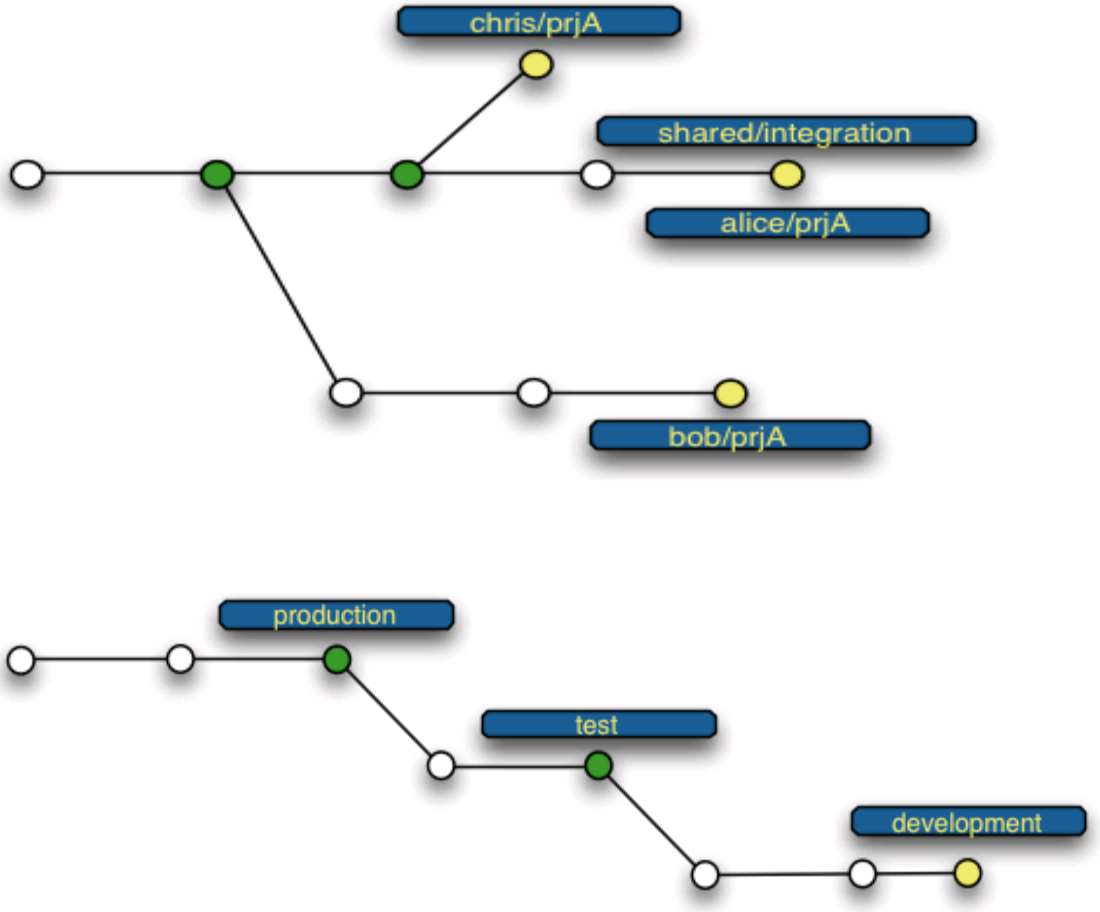
For team:



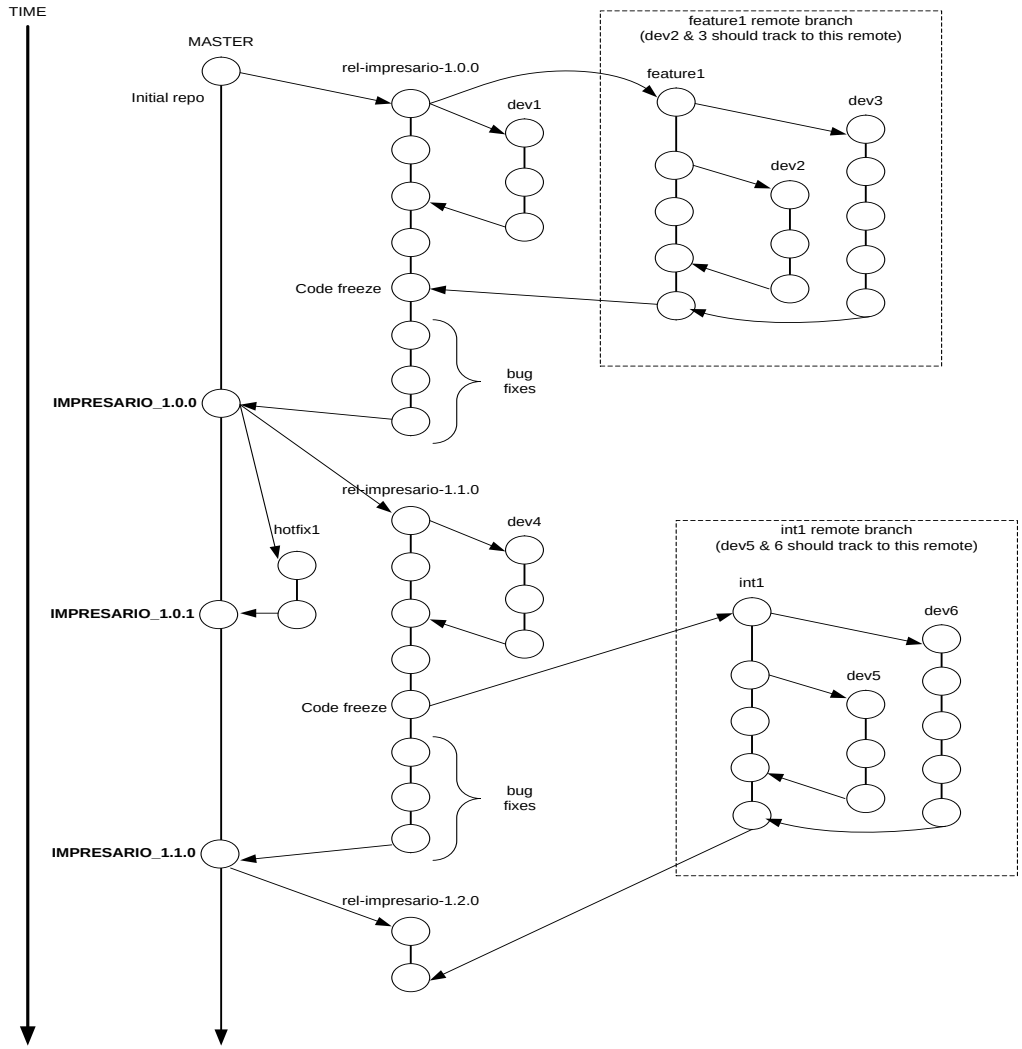


Branch strategies

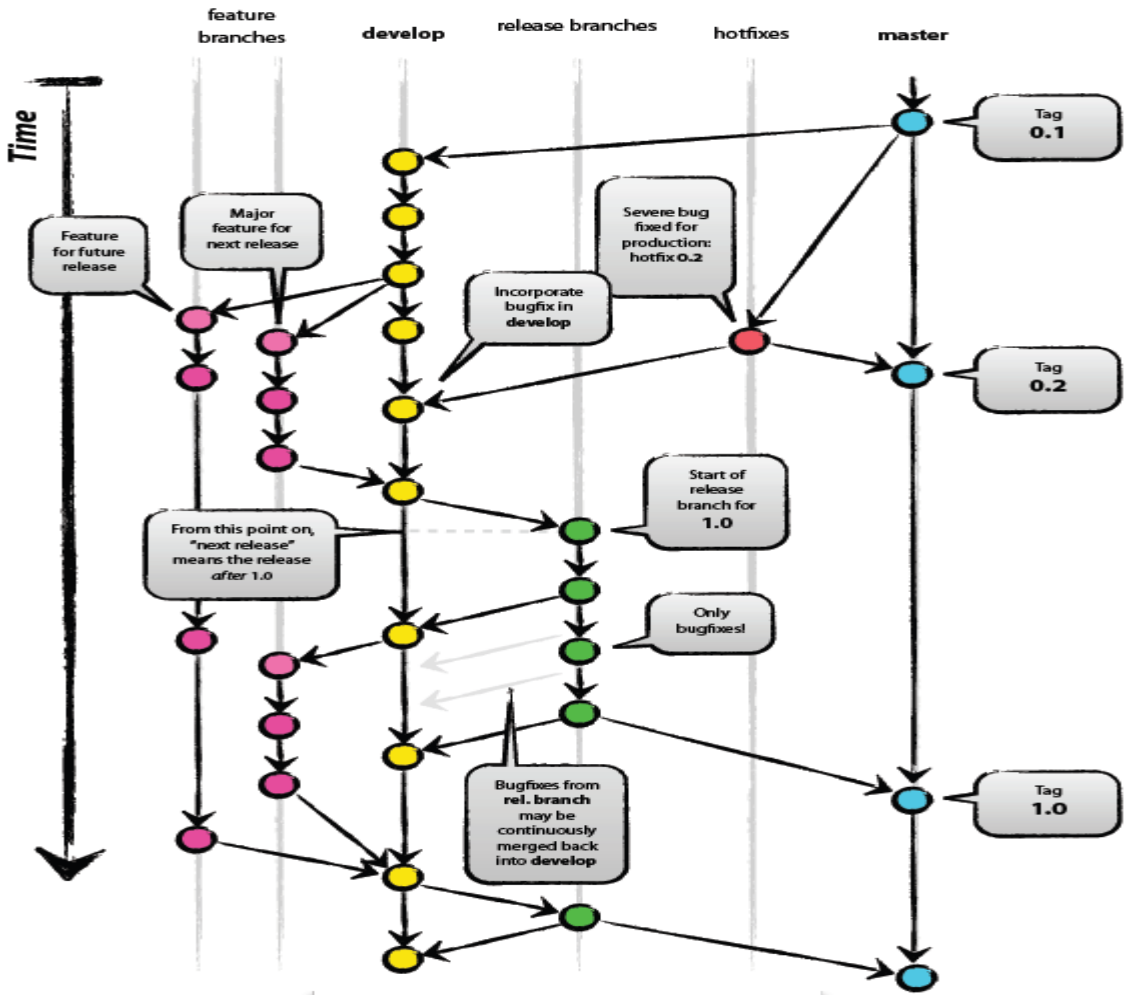
For project:



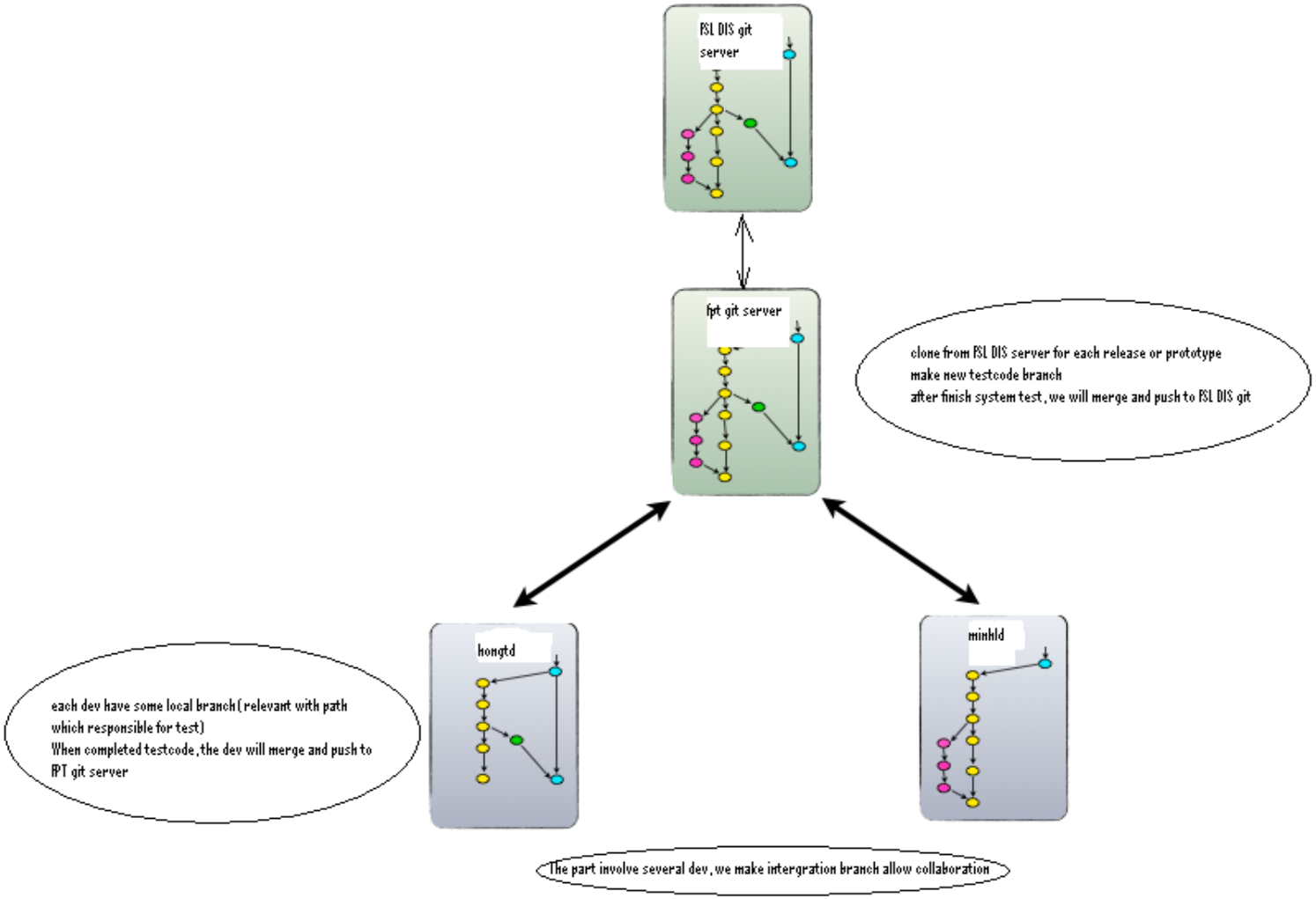
Branch strategies



Branch strategies



Branch strategies



Note

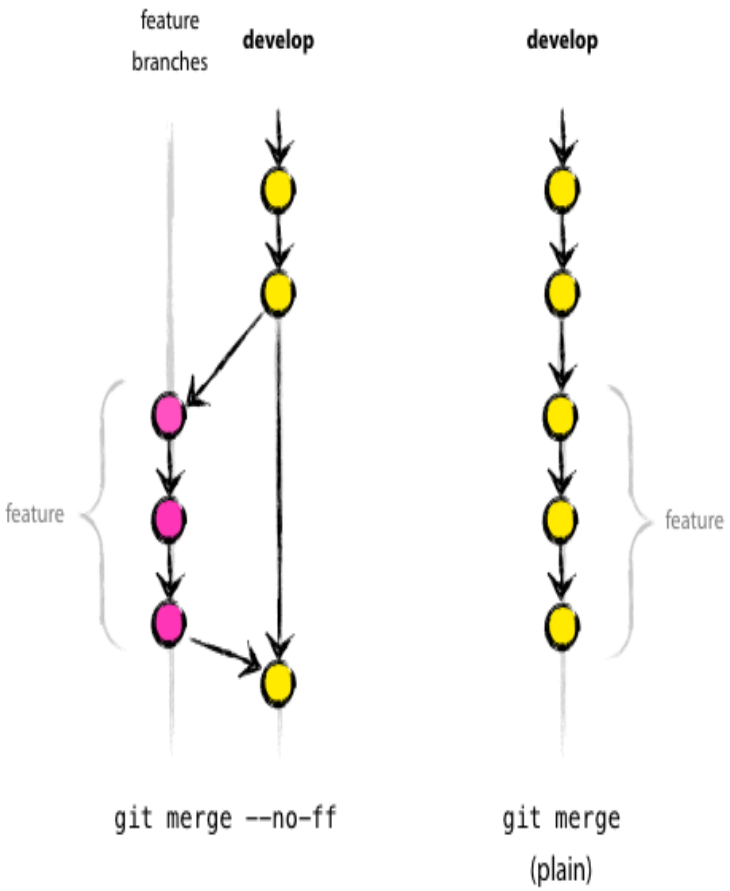
1. don’t use *git pull*, pls use *git fetech* then *git rebase*
2. Release branch naming convention:
   rel-<project name>-<release version>
   rel-impresario-1.0.0
3. Tag naming convention:
   <project name>_<release version>
   IMPRESARIO_1.0.0
4. Coding style check before commit

Note

5. Commit log format:
   1st Line - ENGRxxxxxxxx<space>A short description used for patch file name
   2nd Line – Blank
   Remaining lines - A detailed description including origin of the patch if external.
   Signed-off line which notes the author of the patch (not the reviewer).
6. Local development branches should be used for all development
7. Branches should be deleted after pushed to release branch to keep the Git as clean as possible
8. Merge should be use to track the changes=>
   git merge –no-ff (no fast forward)

Note

--no-ff flag causes the merge to always create a new commit object, so this avoid losing information



Q&A