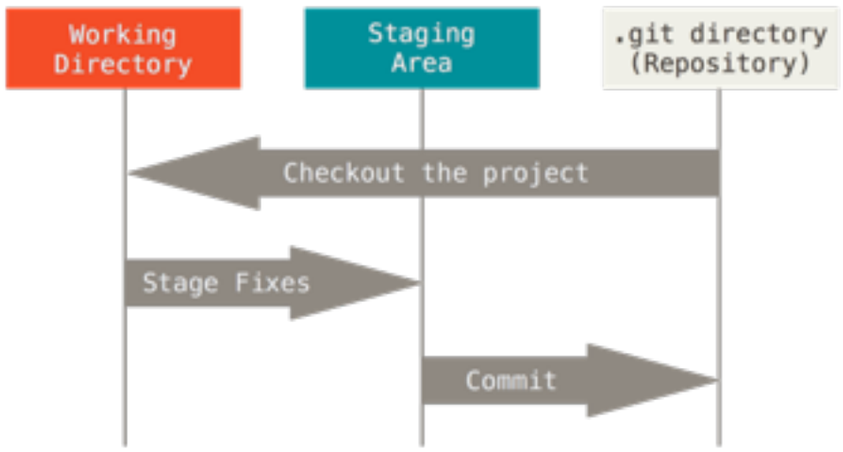
git分享与
gitlab code review

—蔡宜利

git基础

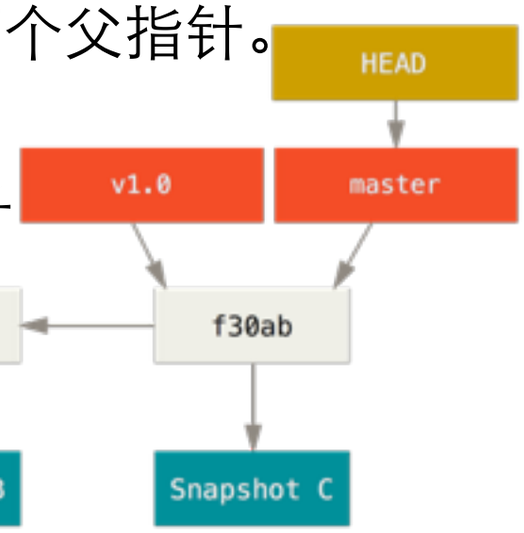
- 三种状态
  - 工作区
  - 暂存区(索引区)
  - .git库



- 工作区：当前修改的还未进入版本库的文件
- 暂存区：git add $path 把要提交的东西添加到暂存区，只有暂存区的东西才可以添加到.git仓库
- .git库：git commit -m “message” 通过git commit 提交到本地仓库的。
- 参考文档：http://www.git-scm.com/book/zh/v2/%E8%B5%B7%E6%AD%A5-Git-%E5%9F%BA%E7%A1%80

git提交

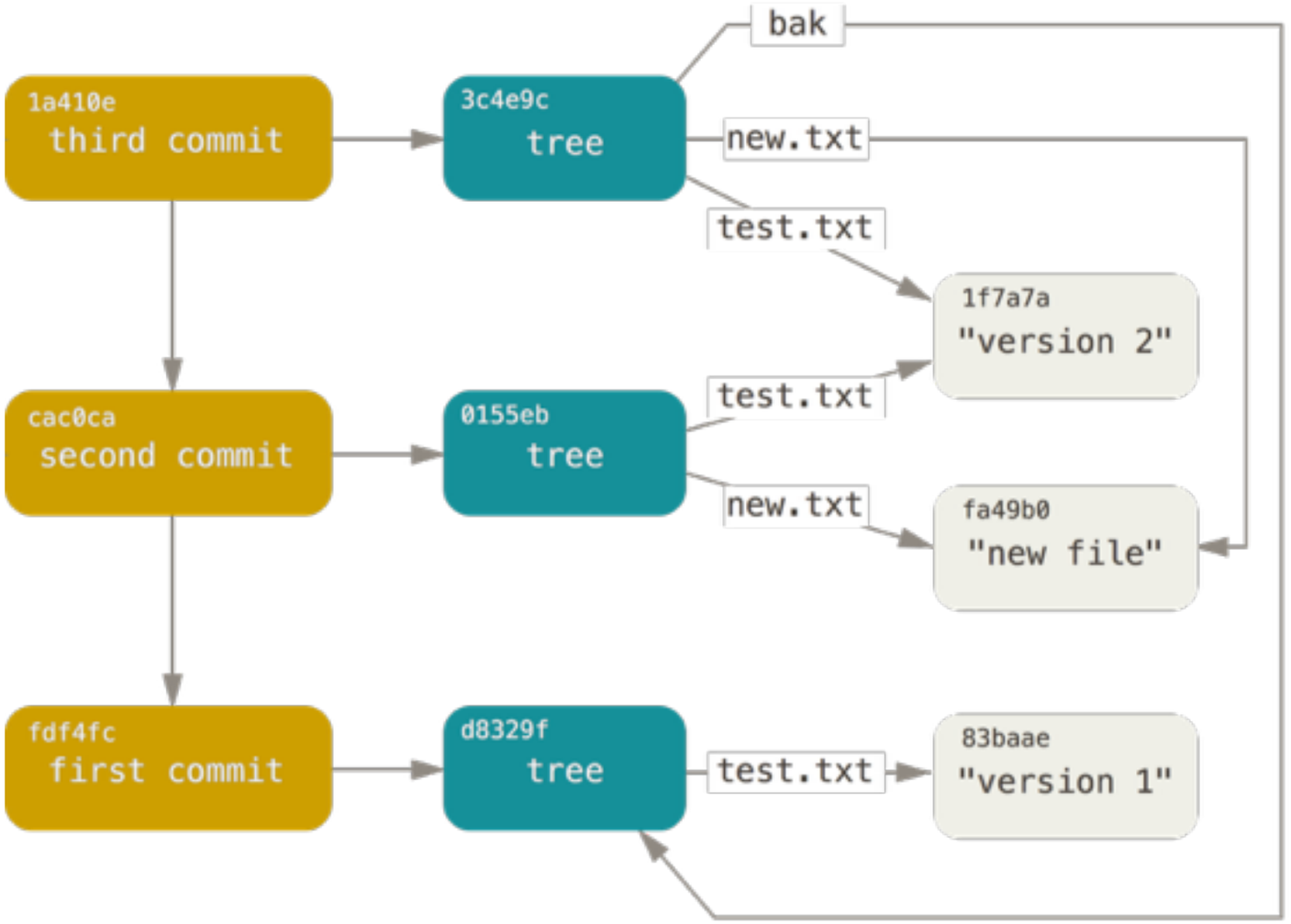
- 每一次提交都有一个唯一的sha-1值。都是一次完整的快照。
- 每一次提交有一个指向父提交的指针，merge的话会有两个父指针。



  - 所有的分支、tag，都是一个指向某一次提交sha值的指针
    - 分支是会随着每次提交向前移动
    - tag是一个静态的指针，创建后就不会变动
- HEAD指针：指向当前的提交。(git checkout xxx；就是更新HEAD指针)

git存储

- 提交对象
- 树对象
- 存储对象



caiyili (ssh) | baijia (bash) | test (bash) | caiyili (ssh) | caiyili

```
[caiyili@mac:~/code/test] (master)⚡!find
find .git/objects/ -type f
.git/objects//02/f78fc10425859bb70e95d4d87005178376085d
.git/objects//15/35dc3a826a0af1e03bd35593a3351b2b551797
.git/objects//25/74c4d6d218dc4a5712005bd5505b6bae62ecef
.git/objects//2f/b60ea9776062bcb0dd19a3f06f28a80825735d
.git/objects//3b/486d3016dabf4845e54dd3c0cbc82edbe8e18a
.git/objects//44/54aec0fd80f5f75640245e4ba4b3db9d607049
.git/objects//4a/ea716ee151d14bef1c2f78c01ae3f42c626641
.git/objects//5b/65f7b6a270e011bdbba46884b0d00a392e8e9f
.git/objects//63/d55c11c24fb661c4581d2ae1a8d229bc5fee30
.git/objects//a2/6cd0d5458da838739e49e3ba5ab93873dbc935
.git/objects//af/e459e9dddb998094db54ff7056c38fdee23288
.git/objects//c5/0c9e195030e4fdcde0b9eef1e79a3f6545f96b
.git/objects//e2/f54c6efb8bbdce8143975a676ed58402e001e7
[caiyili@mac:~/code/test] (master)⚡git show
commit 63d55c11c24fb661c4581d2ae1a8d229bc5fee30
Author: CaiYili <caiyili@baijiahulian.com>
Date:   Fri Oct 30 20:12:43 2015 +0800

    add src/readme

diff --git a/src/readme.txt b/src/readme.txt
new file mode 100644
index 0000000..5b65f7b
--- /dev/null
+++ b/src/readme.txt
@@ -0,0 +1 @@
+this is a readme file
[caiyili@mac:~/code/test] (master)⚡git cat-file -p 63d55c11c24fb661c4581d2ae1a8d229bc5fee30
tree 3b486d3016dabf4845e54dd3c0cbc82edbe8e18a
parent 02f78fc10425859bb70e95d4d87005178376085d
author CaiYili <caiyili@baijiahulian.com> 1446207163 +0800
committer CaiYili <caiyili@baijiahulian.com> 1446207163 +0800

add src/readme
[caiyili@mac:~/code/test] (master)⚡git cat-file -p 3b486d3016dabf4845e54dd3c0cbc82edbe8e18a
100644 blob 1535dc3a826a0af1e03bd35593a3351b2b551797    main.php
040000 tree afe459e9dddb998094db54ff7056c38fdee23288    src
[caiyili@mac:~/code/test] (master)⚡git cat-file -p afe459e9dddb998094db54ff7056c38fdee23288
100644 blob 5b65f7b6a270e011bdbba46884b0d00a392e8e9f    readme.txt
[caiyili@mac:~/code/test] (master)⚡git cat-file -p 5b65f7b6a270e011bdbba46884b0d00a392e8e9f
this is a readme file
[caiyili@mac:~/code/test] (master)⚡git cat-file -p 1535dc3a826a0af1e03bd35593a3351b2b551797
<?php

function main($argv) {
    echo "hello world!\n" ;
}

main($argv) ;

[caiyili@mac:~/code/test] (master)⚡
```

commit对象

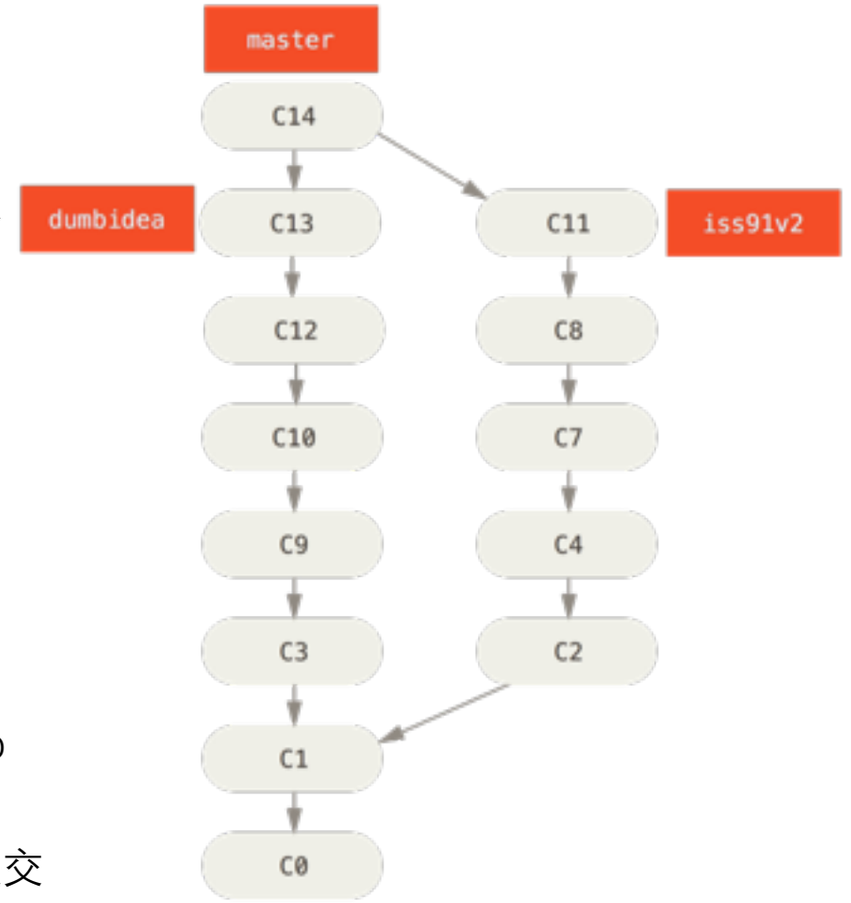
tree对象

存储对象

git分支

- 创建一个新分支
  - git branch $branch_name [<start_point>]
  - git checkout -b $branch_name [<start_point>]
- 切换到一个分支
  - git checkout branch_name

git分支



- git创建分支
  - git checkout -b iss91v2 [<start_point>]
  - echo 02f78fc10425859bb > .git/refs/heads/feature-a
  - git update-ref refs/heads/feature-a 02f78fc10425859bb
  - 本质上就是创建了一个文件，一个git的引用。在以后的每次提交中，都会去更新这个引用。

```
[caiyili@mac:~/code/test] (master)⚡git rev-list --all
02f78fc10425859bb70e95d4d87005178376085d
e2f54c6efb8bbdce8143975a676ed58402e001e7
[caiyili@mac:~/code/test] (master)⚡echo e2f54c6efb8bbdce8143975a676ed58402e001e7 > .git/refs/heads/feature-a
[caiyili@mac:~/code/test] (master)⚡git update-ref refs/heads/feature-b 02f78fc10425859bb70e95d4d87005178376085d
[caiyili@mac:~/code/test] (master)⚡git update-ref refs/tags/v1.0.0 e2f54c6efb8bbdce8143975a676ed58402e001e7
[caiyili@mac:~/code/test] (master)⚡git log --graph --decorate --oneline
* 02f78fc (HEAD, master, feature-b) second commit
* e2f54c6 (tag: v1.0.0, feature-a) first commit
[caiyili@mac:~/code/test] (master)⚡git update-ref refs/tags/v1.0.1 02f78fc10425859bb70e95d4d87005178376085d
[caiyili@mac:~/code/test] (master)⚡git log --graph --decorate --oneline
* 02f78fc (HEAD, tag: v1.0.1, master, feature-b) second commit
* e2f54c6 (tag: v1.0.0, feature-a) first commit
[caiyili@mac:~/code/test] (master)⚡
```

```
" Press ? for help
.. (up a dir)
/Users/caiyili/code/test/
▾ .git/
  ▸ hooks/
  ▸ info/
  ▸ logs/
  ▸ objects/
  ▾ refs/
    ▾ heads/
        feature-a
        feature-b
        master
    ▾ tags/
        v1.0.0
        v1.0.1
    COMMIT_EDITMSG
    config*
    description
    HEAD
    index
  main.php
~
```

```
1 e2f54c6efb8bbdce8143975a67
~
```

远程仓库

- 远程分支是一个的远程引用（只读）
- 远程分支可以有多个（git remote 命令来管理）
    - git remote -v //显示远程分支的信息
    - git remote add **origin** git@github.com:caiyili/test01.git //添加一个远程仓库
    - git push -u **origin** feature-b:feature-b //推送本地分支到远程
- git clone 从远程分支clone代码后默认的远程分支名叫**origin**
- 远程仓库的配置在.git/config里面（其实很多配置都在这里面）

远程仓库的更新

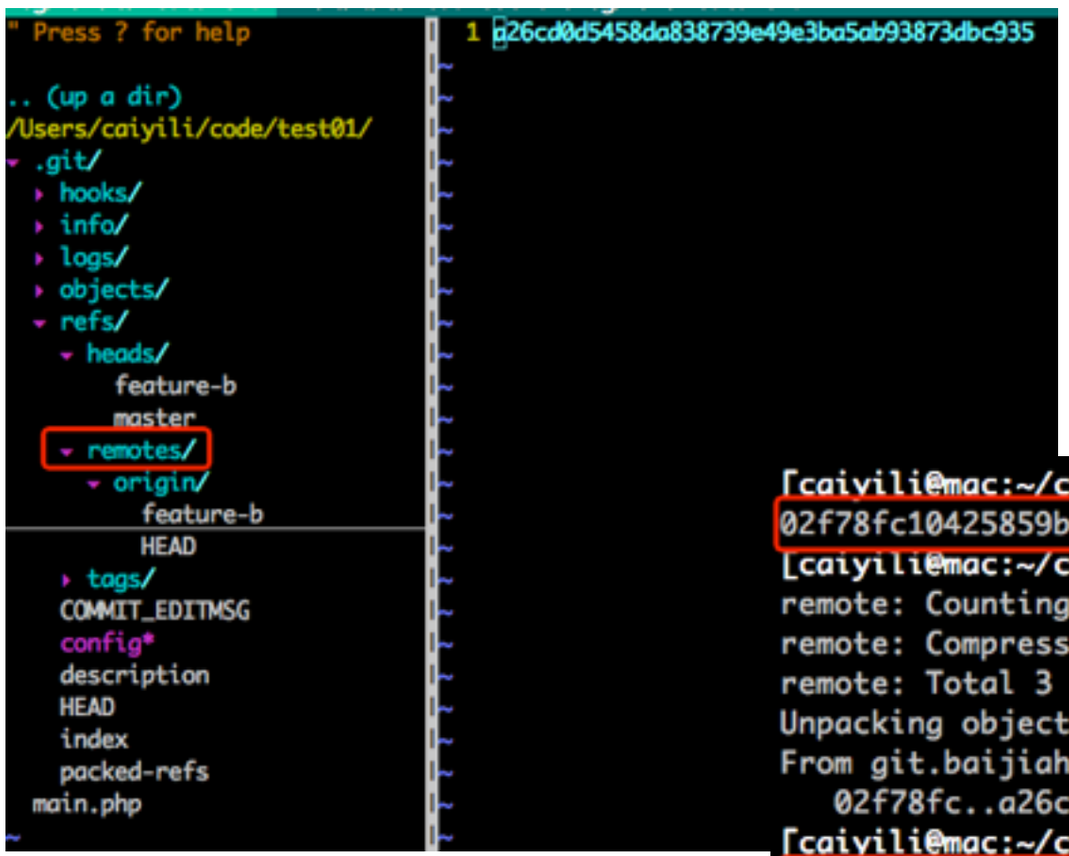
- git大部分命令都是本地操作，不需要联网
- 把远程的更新同步到本地的操作
  - git fetch //拉取远程（默认是origin）的更新，包括其它人push上去的commit/branch/tag，-t指定拉取所有tag，默认是拉分支能到达的tag
  - git pull // 等于git fetch ; git merge origin/xxx
- 把自己的更新推送到远程
  - git push origin local-branch:remote-branch
  - git push origin :branch-name //删除远程的分支/tag

远程分支更新

- fetch更新，更新远程分支的引用，以及分支路径上的objects
- 远程分支的引用在.git/refs/remotes下面。它只代表新近一次同步代码时远程分支所在的位置。
- 远程分支的引用方式：origin/feature-b （git fetch; git diff feature-b origin/feature-b）

Figure 3-23. 本地与远程的工作可以分叉

如果要同步你的工作，运行 `git fetch origin` 命令。这个命令查找 "origin" 是哪一个服务器（在本例中，它是 `git.ourcompany.com`），从中抓取本地没有的数据，并且更新本地数据库，移动 `origin/master` 指针指向新的、更新后的位置。




```
[caiyili@mac:~/code/test] (master)⚡cat .git/refs/remotes/origin/feature-b
02f78fc10425859bb70e95d4d87005178376085d
[caiyili@mac:~/code/test] (master)⚡git fetch
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
From git.baijiahulian.com:caiyili/test01
   02f78fc..a26cd0d  feature-b  -> origin/feature-b
[caiyili@mac:~/code/test] (master)⚡cat .git/refs/remotes/origin/feature-b
a26cd0d5458da838739e49e3ba5ab93873dbc935
```

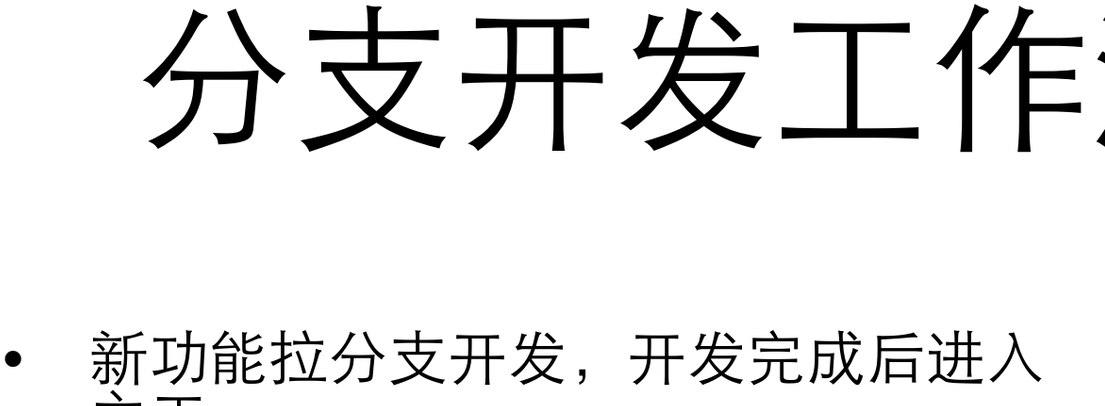
创建本地跟踪分支

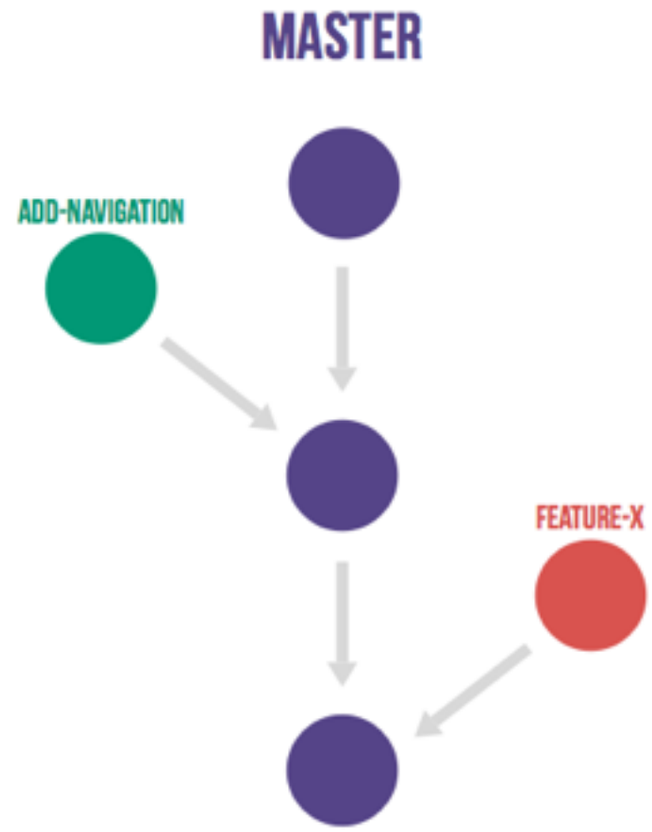
- 基于远程分支创建自己的分支
    - git fetch
    - git checkout -b feature-b origin/feature-b
    - git checkout -t origin/feature-b // —track
- 基于tag创建分支
    - git checkout -b hotfix-xxx release_www_v1.2.3.4
    - git checkout release_www_v1.2.3.4; git checkout -b hotfix-xxx; //checkout 就是移动HEAD指针
- 错误的方式：
    - git checkout -b feature-b //等于git checkout -b feature-b HEAD
    - git push origin feature-b
    - 这样操作其实是基于当前的分支新建了一个分支，并把这个分支推到远程的feature-b分支

分支开发工作流

（一）分支开发主干上线

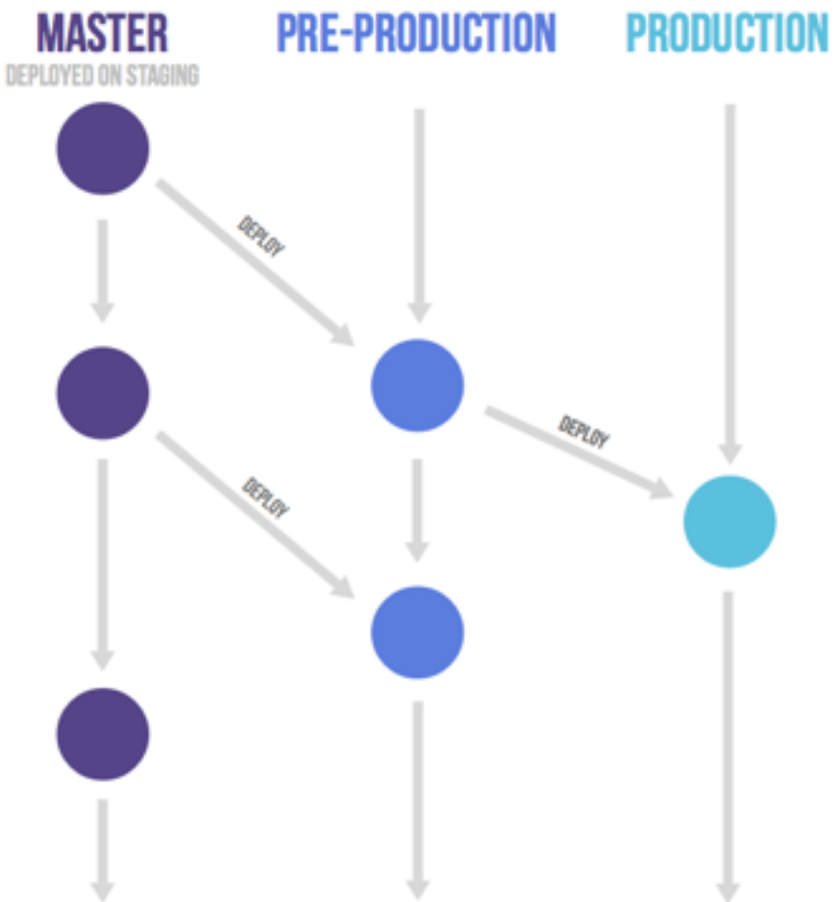
- 新功能拉分支开发，开发完成后进入主干。
- 修复bug基于线上tag修复上线，上完线要把修复的代码合到master分支，以防止下次上线被覆盖。



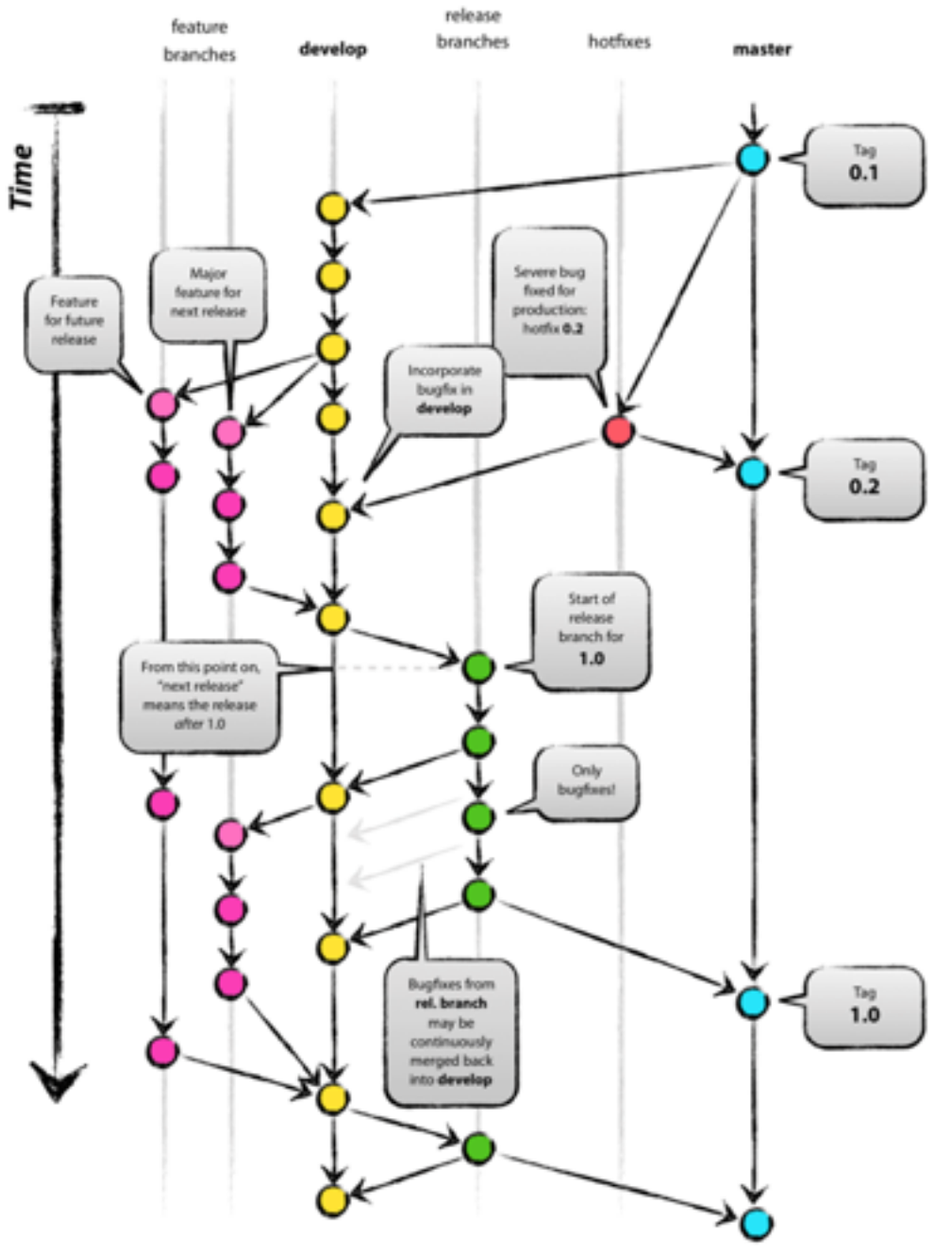


参考：https://about.gitlab.com/2014/09/29/gitlab-flow/

（二）基于环境的分支



（三）git flow



gitlab merge request

- 如何保证主干代码是“干净”的?
- 把git主干分支设置为保护，禁止直接从本地向这个分支push代码。
  - ~~git push origin master //会失败~~
- 只能先把代码push到其它非保护的分支cyl-dev，然后从gitlab平台发起cyl-dev到master分支的merge request。
  - git push -u origin cyl-dev //push自己的分支
  - git push origin master:cyl-dev //如果本地是在master上开发的，就把本地的master推送到远端的cyl-dev分支
- 有权限的同学可以接收request。其它同学也可以浏览、评价。

整体流程

- **git clone git@github.com:caiyili/test01.git test01 //**克隆一份远程仓库到本地，这个远程仓库名默认就是**origin**（使用**git remote -v //**显示已添加的远程分支）
- **git checkout -b cyl-dev master //**基于**master**创建一个自己的分支
- **vim xxx**；**git commit　//**修改文件并提交
- **git pull origin master //**经常性的把**master**的代码拉取并合并到自己的主开发分支（相当于**git fetch origin; git merge origin/master**，把**origin/master**的代码合并到本地的**cyl-dev**）
- **git push -u origin cyl-dev:cyl-dev //**把本地分支推送到远端**origin/cyl-dev**
- 去**gitlab**上创建一个**merge request**，请求把代码合并到**master**分支，指定**code review**审核人员
- **reviewer**反馈问题后
- **vim xxx; git commit ; git push //**本地修改、提交，再**push**上去。
- 由于**merge request**是基于分支到分支的，这边**push**上去后，**reviewer**看到的就是更新后的，不需要再发**merge request**
- **reviewer**审核通过后，代码就进入**master**
- 参考文档：**http://www.git-scm.com/book/zh/v2/Git-%E5%88%86%E6%94%AF-%E8%BF%9C%E7%A8%8B%E5%88%86%E6%94%AF**

修bug

- 小的修改
- 先在自己的主开发分支上修改然后把**commit点cherry-pick到线上的tag**
  - **git checkout cyl-dev**
  - **vim xxx;**
  - **git commit //会有一个唯一的提交值$commit_id**
  - **git checkout release_m_v1.2.3.4**
  - **git cherry-pick $commit_id**
  - **git tag -a release_m_v1.2.3.5 -m “fix bug base on 1234”**
  - 发送一个**cyl-dev分支到master的merge request**

- 大的修改
- 基于线上的**tag拉一个新的分支hotfix/xxx来修复bug，修复完后发一个merge request到master**。然后删除**hotfix/xxx分支**
  - **git checkout -b hotfix/video release_m_v1.2.3.4**
  - **vim xxx;**
  - **git commit**
  - **git tag -a release_m_v1.2.3.5 -m “fix bug base on 1234”**
  - 发送一个**hotfix/video分支到master的merge request**

关于分支的建议

- 建一个自己的常用开发分支cyl-dev
- 小的修改(bug fix)直接在自己的分支上修改。
- 大的修改(新功能的开发)新建其它feature/xxx分支开发。
- 开发完，可以从feature/xxx发merge request到master。自己的常用分支经常性的去拉取远端master的代码。
- 如果feature/xxx开发完但需要等下次上线再一起进master，也可以自己先merge到cyl-dev分支。
- 尽量每一个功能都单独拉分支开发，这样更容易控制哪些功能可以进master哪些不进。
- 新建一个分支只是新建了一个对commit点的引用，没有任何其它代价，why not?
- 开发完成后一定要记得删除分支，不然就分支泛滥。
- 如果撤销一个merge: http://git-scm.com/blog/2010/03/02/undoing-merges.html

Q&A

谢谢！