Git and GitLab

2015.07 xirong

http://www.ixrong.com

分享目的

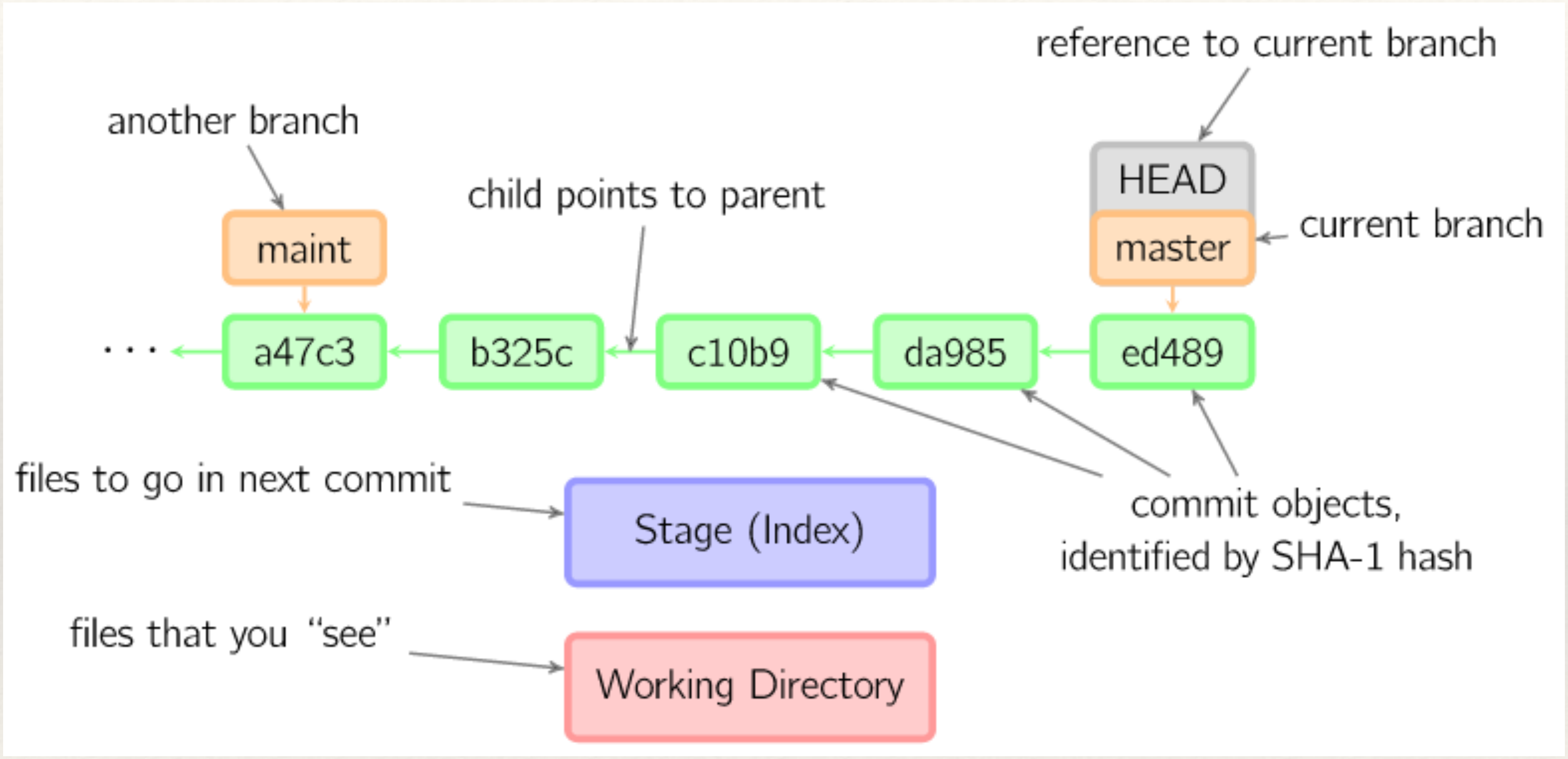
- git只是一个版本控制的工具，使用很简单
- 日常开发常用操作，低成本由svn切换到git
- 分支开发/pull request/code review 介绍
- git 还能做什么

大纲

- 基本概念
- branch分支
- gitlab集成
- 实例演示
- 高级功能
- 常用命令
- 资料推荐

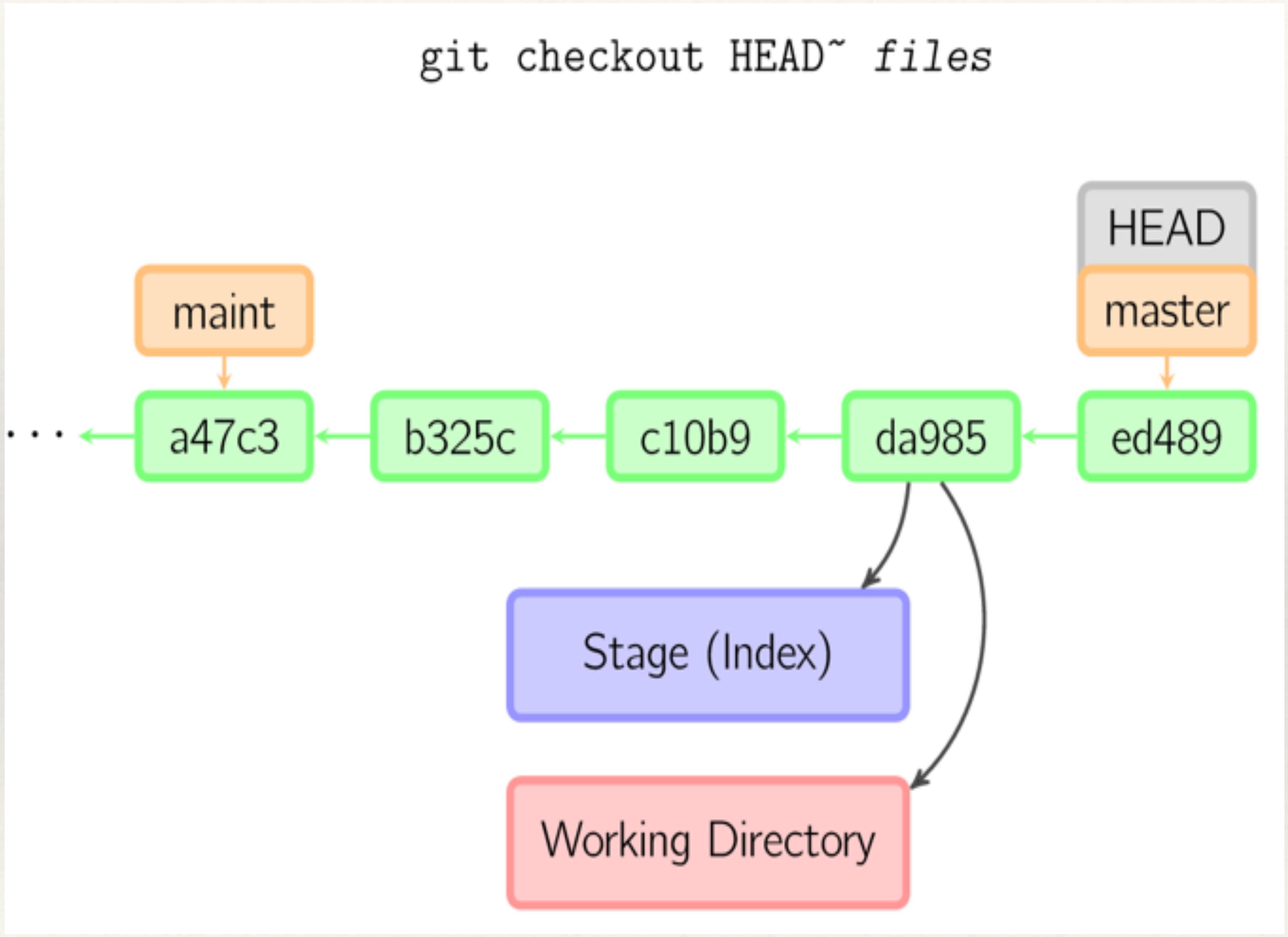
基本概念

工作区-暂存区-仓库



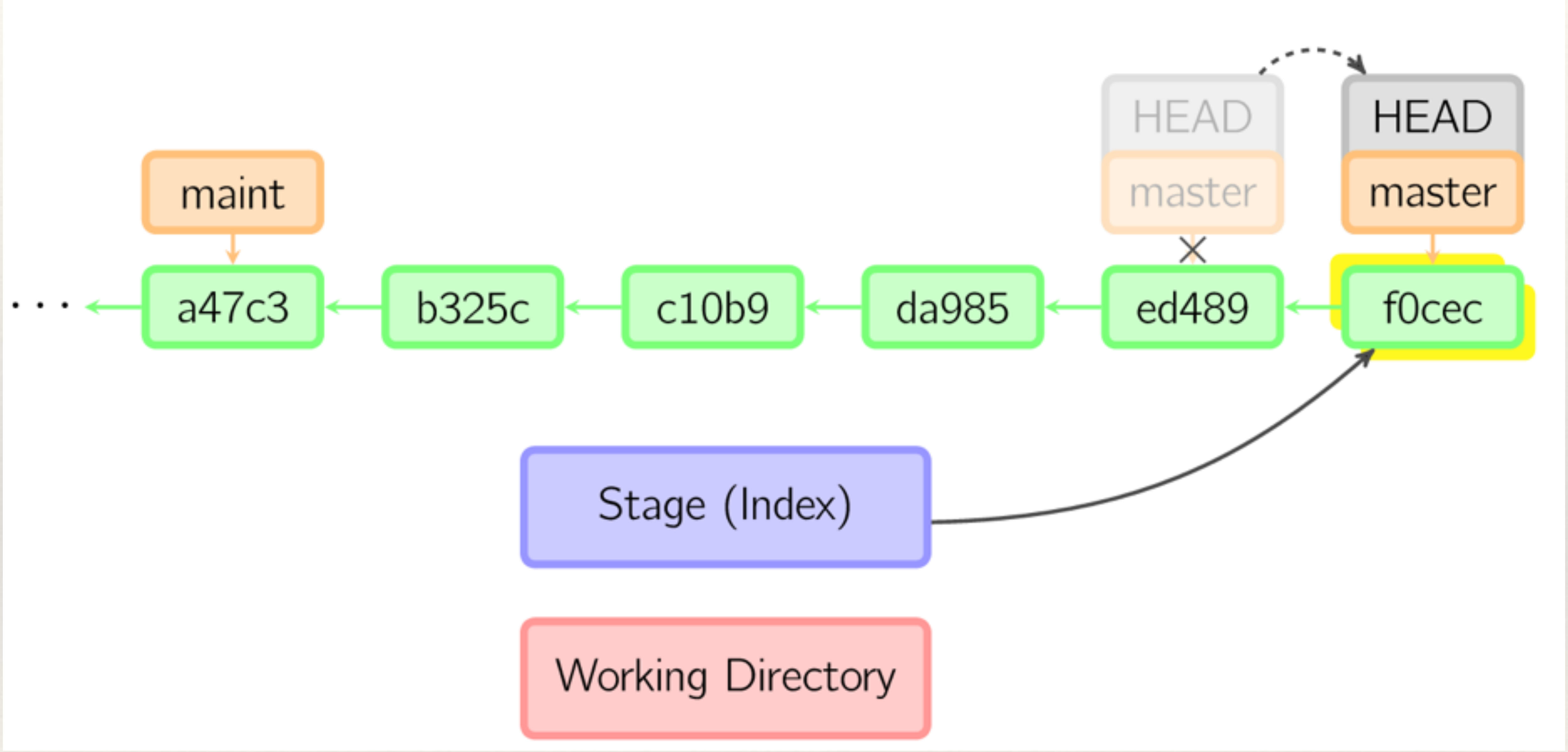
工作区：你所能看到的目录及文件(.git除外)
暂存区：分批/分阶段/文件快照，及时回退
版本库：.git 文件夹
git add && git commit

checkout



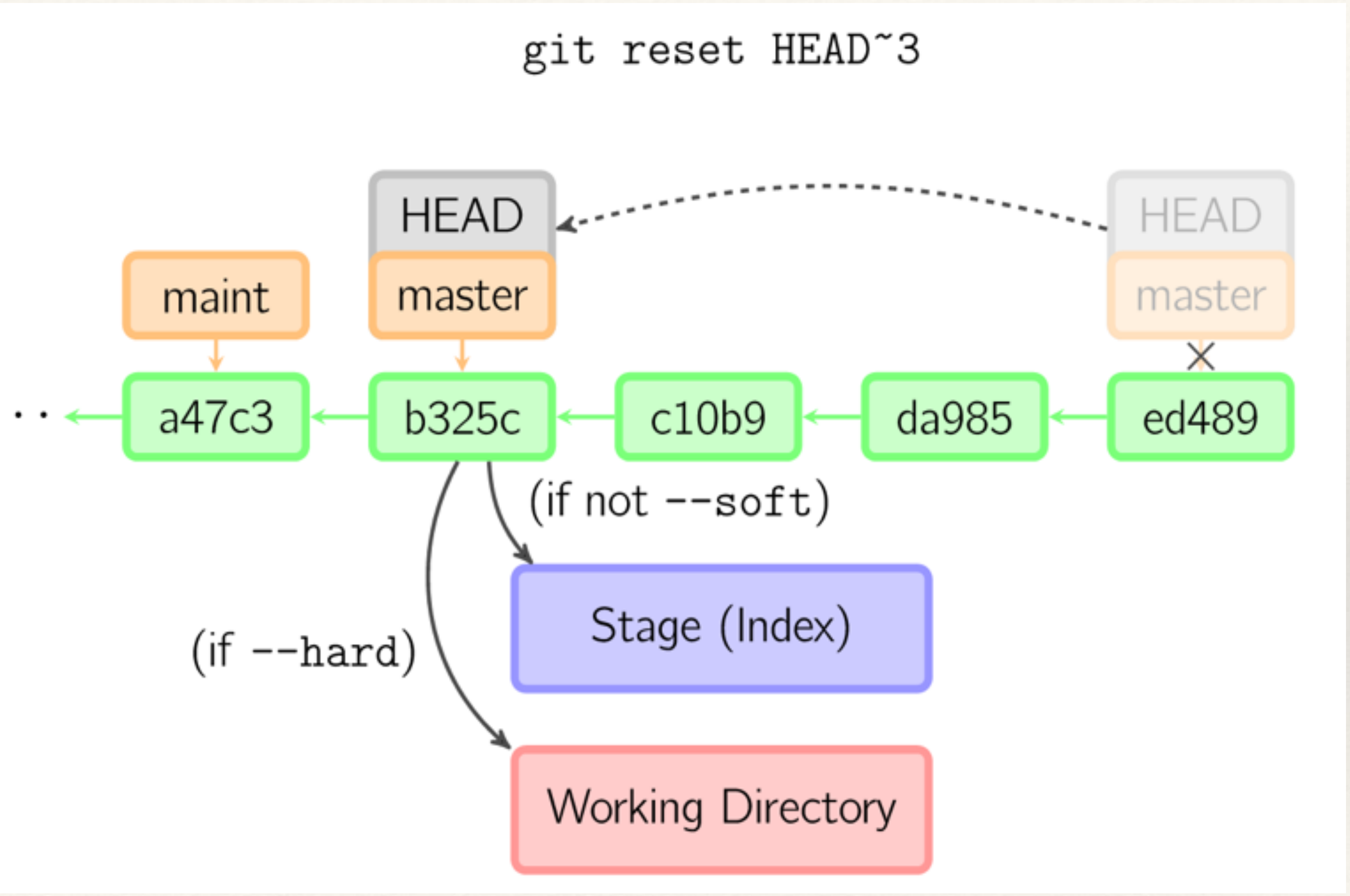
1、切换分支
2、将仓库历史提交或暂存区中文件拷贝到工作区（回滚文件）

commit



git add/stage 将工作区加入暂存区
git commit (-a -m) 将暂存区内容提交到仓库
git commit —amend 修改本次提交(修改提交log信息）

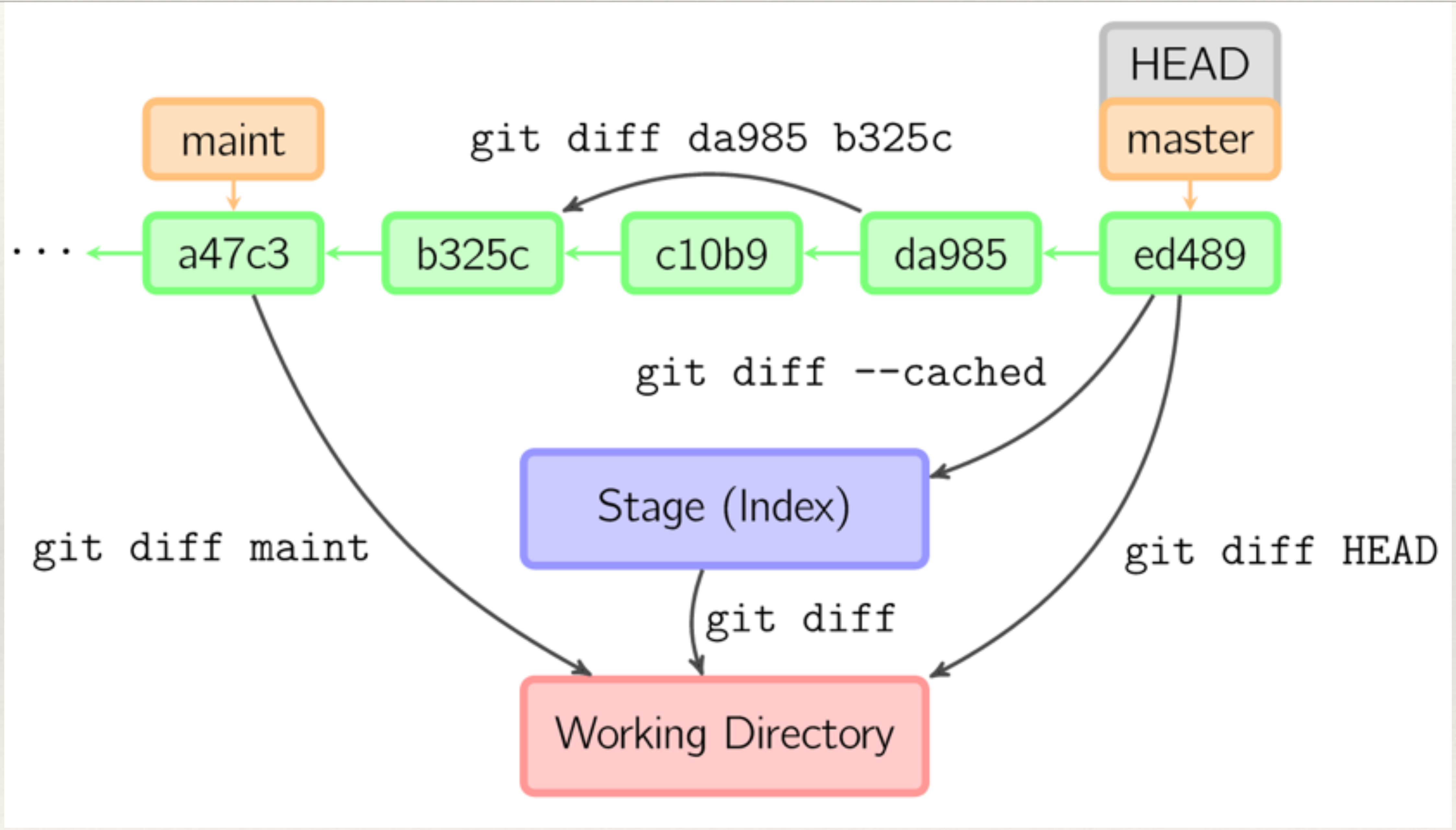
reset



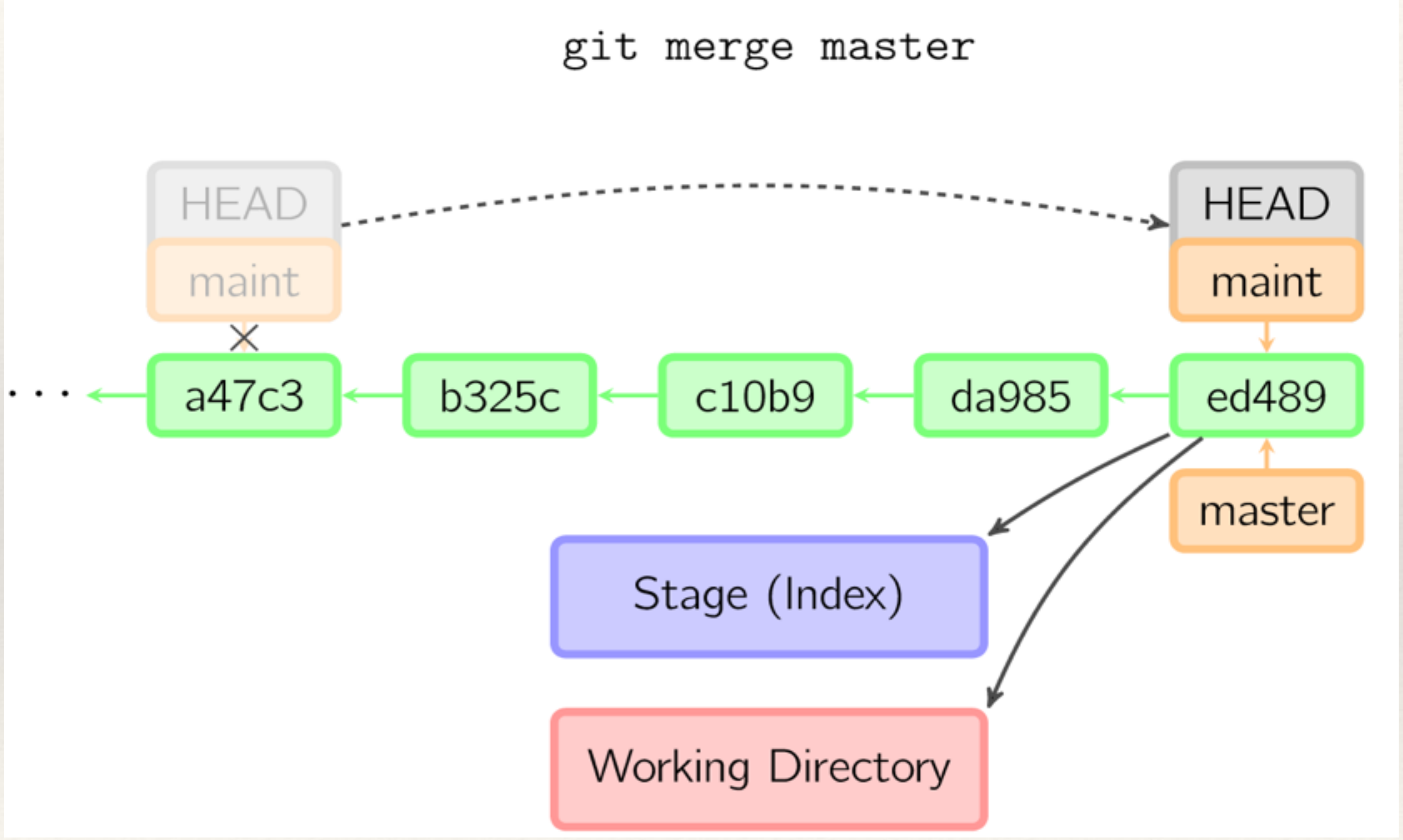
变更分支到指定位置，有
选择变动工作区和索引区

git reset —hard HEAD^
git reset —soft HEAD^
git reset —mixed HEAD

diff

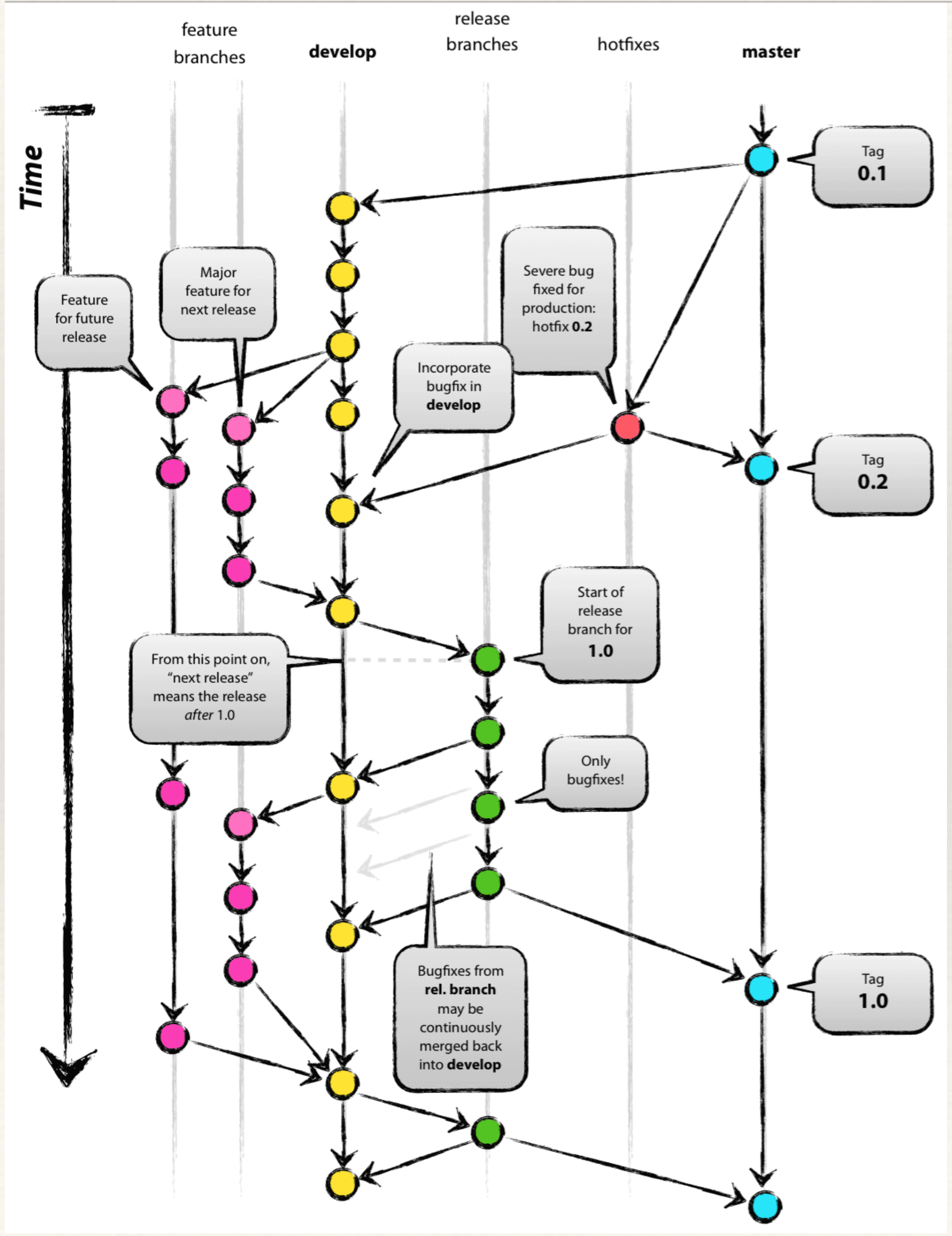

HEAD
maint
master
git diff da985 b325c
...
a47c3
b325c
c10b9
da985
ed489
git diff --cached
Stage (Index)
git diff maint
git diff HEAD
git diff
Working Directory

merge



git merge —no-ff master 产生合并分支记录，防止fast-forward

Branch Tag


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

```
git branch 查看当前所有分支
git branch develop 创建develop分支
git checkout –b develop master 创建并切换到develop分支
git merge master
git merge –no--ff master 合并master到当前分支
git branch –d develop 删除分支
git tag –a 1.0  --创建tag
git push origin –tags
```

GitLab

- new repository
- ssh-keygen -t rsa -C “email@gmail.com”
- git clone
- add,commit,log,diff,merge,push
- pull request
- code review

实例演示

new repo,add,commit,reset,diff,merge,branch dev,code review

what else

- alias
- stash
- submodule
- rebase
- revert vs reset

常用命令

```
git init 把当前的目录变成可以管理的git仓库，生成隐藏.git文件。
git config —global user.name “xx.x”/user.email “ex@corp.elong.com” 配置信息
git add XX 把xx文件添加到暂存区去。
git commit —m “XX” 提交文件 —m 后面的是注释。
git status 查看仓库状态
git diff XX 查看XX文件修改了那些内容
git diff --查看冲突文件（图形化工具）
git log 查看历史记录
git reset --hard HEAD^ 或者 git reset --hard HEAD~ 回退到上一个版本
git checkout -- XX 把XX文件在工作区的修改全部撤销。
git rm XX 删除XX文件
git clone git@gitlab.dev:online_web/web_logplatform.git 从远程库中克隆
git remote add origin git@gitlab.dev:online_web/web_logplatform.git 关联一个远程库
git remote 查看远程仓库
git remote  -v 远程仓库详细信息
git push —u origin master 把当前master分支推送到远程库
git branch 查看当前所有的分支
git branch develop —创建一个名叫develop的分支
git checkout master 切换回master分支
git checkout —b develop master  从master分支上开出develop分支，并切换到下面
git branch  —列出本地仓库项目所有的分支
git merge —no--ff master  --develop和master分支各进行了修改，合并两个分支
git branch —d develop — 分支完成使命，删除分支
git tag —a 1.0  --创建tag
git push origin —tags
git stash 把当前的工作隐藏起来 等以后恢复现场后继续工作
git stash list 查看所有被隐藏的文件列表
git stash apply 恢复被隐藏的文件，但是内容不删除
git stash drop 删除文件
git stash pop 恢复文件的同时 也删除文件
```

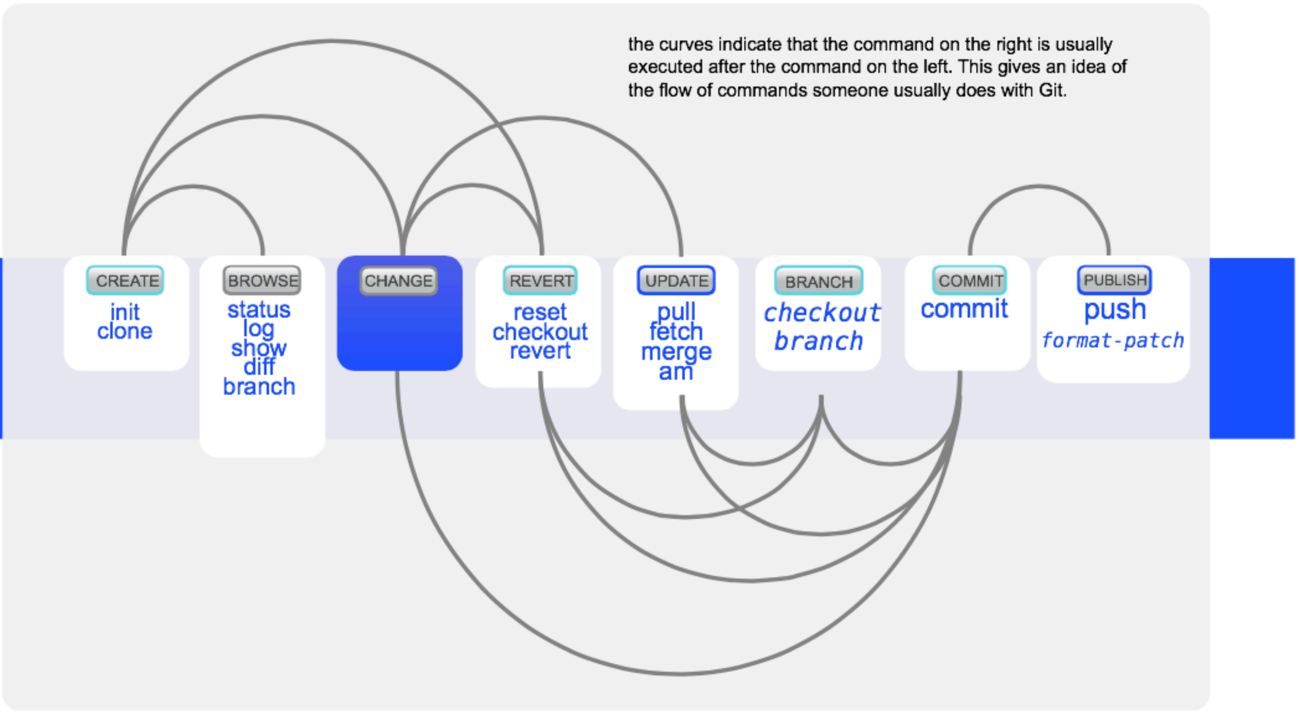
Git Cheat Sheet

http://git.or.cz/

Remember: git command --help

Global Git configuration is stored in $HOME/.gitconfig (git config --help)

Commands Sequence



the curves indicate that the command on the right is usually executed after the command on the left. This gives an idea of the flow of commands someone usually does with Git.
CREATE
init
clone
BROWSE
status
log
show
diff
branch
CHANGE
REVERT
reset
checkout
revert
UPDATE
pull
fetch
merge
am
BRANCH
checkout
branch
COMMIT
commit
PUBLISH
push
format-patch

Create

From existing data
cd ~/projects/myproject
git init
git add .

From existing repo
git clone ~/existing/repo ~/new/repo
git clone git://host.org/project.git
git clone ssh://you@host.org/proj.git

Show

Files changed in working directory
git status

Changes to tracked files
git diff

What changed between $ID1 and $ID2
git diff $id1 $id2

History of changes
git log

History of changes for file with diffs
git log -p $file $dir/ec/tory/

Who changed what and when in a file
git blame $file

A commit identified by $ID
git show $id

A specific file from a specific $ID
git show $id:$file

All local branches
git branch
(star '*' marks the current branch)

Cheat Sheet Notation

$id : notation used in this sheet to represent either a commit id, branch or a tag name
$file : arbitrary file name
$branch : arbitrary branch name

Concepts

Git Basics

master	: default development branch
origin	: default upstream repository
HEAD	: current branch
HEAD^	: parent of HEAD
HEAD~4	: the great-great grandparent of HEAD

Revert

Return to the last committed state
git reset --hard
⚠ you cannot undo a hard reset

Revert the last commit
git revert HEAD — Creates a new commit

Revert specific commit
git revert $id — Creates a new commit

Fix the last commit
git commit -a --amend
(after editing the broken files)

Checkout the $id version of a file
git checkout $id $file

Branch

Switch to the $id branch
git checkout $id

Merge branch1 into branch2
git checkout $branch2
git merge branch1

Create branch named $branch based on the HEAD
git branch $branch

Create branch $new_branch based on branch $other and switch to it
git checkout -b $new_branch $other

Delete branch $branch
git branch -d $branch

Update

Fetch latest changes from origin
git fetch
(but this does not merge them).

Pull latest changes from origin
git pull
(does a fetch followed by a merge)

Apply a patch that some sent you
git am -3 patch.mbox
(in case of a conflict, resolve and use git am --resolved)

Publish

Commit all your local changes
git commit -a

Prepare a patch for other developers
git format-patch origin

Push changes to origin
git push

Mark a version / milestone
git tag v1.0

Useful Commands

Finding regressions
git bisect start (to start)
git bisect good $id ($id is the last working version)
git bisect bad $id ($id is a broken version)

git bisect bad/good (to mark it as bad or good)
git bisect visualize (to launch gitk and mark it)
git bisect reset (once you're done)

Check for errors and cleanup repository
git fsck
git gc --prune

Search working directory for foo()
git grep "foo()"

Resolve Merge Conflicts

To view the merge conclicts
git diff (complete conflict diff)
git diff --base $file (against base file)
git diff --ours $file (against your changes)
git diff --theirs $file (against other changes)

To discard conflicting patch
git reset --hard
git rebase --skip

After resolving conflicts, merge with
git add $conflicting_file (do for all resolved files)
git rebase --continue

Zack Rusin
Based on the work of
Sébastien Pierre

学习资料

https://github.com/xirong/my-git

QA