环境准备：

esekilxv7320 [9:56] [/repo/eqiasui] -> git clone --recurse-submodules ssh://gerrit.ericsson.se:29418/radio/NGR\_LFT

Cloning into 'NGR\_LFT'...

esekilxv7320 [9:59] [/repo/eqiasui] -> cd NGR\_LFT/

esekilxv7320 [9:59] [/repo/eqiasui/NGR\_LFT] -> ls

coveritykey dfe\_deliveries external generic rru\_app\_file\_list.xlsx rulib tddlte

esekilxv7320 [9:59] [/repo/eqiasui/NGR\_LFT] -> git log

Broken pipe

esekilxv7320 [9:59] [/repo/eqiasui/NGR\_LFT] -> git submodule update --init --recursive

esekilxv7320 [10:00] [/repo/eqiasui/NGR\_LFT] -> git config --global user.name 'sui qiao'

esekilxv7320 [10:00] [/repo/eqiasui/NGR\_LFT] -> git config --global user.email 'sui.qiao@ericsson.com'

esekilxv7320 [10:00] [/repo/eqiasui/NGR\_LFT] -> /proj/crbs/radiosw/git/hooks/setup\_hooks.sh

cd /repo/eqiasui/NGR\_LFT

ln -s /proj/crbs/radiosw/git/hooks/commit-msg .git/hooks/commit-msg

ln -s /proj/crbs/radiosw/git/hooks/post-commit .git/hooks/post-commit

cd /repo/eqiasui/NGR\_LFT

esekilxv7320 [10:01] [/repo/eqiasui/NGR\_LFT] -> cd .git

1. 关联远程库 git remote add origin git@github.com:michaelliao/learngit.git

2. git push origin master git push <远程库> <本地分支> <远程分支>

git push <远程主机名> <本地分支名> <远程分支名>

2.1 git push origin master

如果远程分支被省略，如上则表示将本地分支推送到与之存在追踪关系的远程分支（通常两者同名），

如果该远程分支不存在，则会被新建

2.2 git push origin ：refs/for/master

　　 如果省略本地分支名，则表示删除指定的远程分支，因为这等同于推送一个空的本地分支到远程分支，等同于 git push origin --delete master

refs/for 的意义在于我们提交代码到服务器之后是需要经过code review 之后才能进行merge的，而refs/heads 不需要

3.git remote -v查看远程版本库信息

4.git pull origin master

5.git remote rm origin删除远程git仓库

6.$ git pull origin master --allow-unrelated-histories

7.git branch 查看本地分支

8.git branch -a查看远程分支

eqiasui@CN00214190 MINGW64 ~/Documents/practice2/qspractice (temp)

$ git branch -a

master

\* temp

remotes/origin/HEAD -> origin/master

remotes/origin/master

remotes/origin/temp

9./\*1.新建分支\*/

git checkout -b temp //新建分支并切换到temp分支

git push origin temp:temp //将代码push到temp分支

10./\*2.删除主分支\*/

git push origin --delete master //删除远端主分支

git branch -d master //删除本地主分支

11./\*3.新建主分支\*/

git checkout -b master //新建主分支并切换到主分支

git push origin master //提交主分支

12./\*4.删除暂存分支\*/

git branch -d temp

git push origin --delete temp

13.总结

远程分支回滚的三种方法：

自己的分支回滚直接用reset： git reset --hard HEAD/HEAD^/HEAD^^/HEAD~3

公共分支回滚用revert: git revert <commid-id>/git revert HEAD/HEAD^/HEAD^^/HEAD~3

错的太远了直接将代码全部删掉，用正确代码替代:撤销：

git checkout HEAD <file> #撤销指定的未提交文件的修改内容

git reset --hard HEAD #撤销工作目录中所有未提交文件的修改内容

git revert <commit-id> #撤销指定的提交

14.push之前查看push的内容：$ git diff origin/temp HEAD --name-status

A b.txt

A git command.txt

A qs.jpg

Note: dev可以为任何分支名；去掉--name-status则显示每个文件的具体变化，加上则只显示有变化的文件。显示结果中D代表delete的文件；A代表add的文件；M代表modified的文件。

15.从远程代码库拉取某个分支（默认是master）

git pull <alias> <branch name>

如git pull origin dev

也可以通过git pull直接拉取不指定分支，不过不在master分支上时git pull远程代码到本地可能需要先设置一下远程和本地分支的对应：

$git branch --set-upstream-to=origin/dev dev

Branch dev set up to track remote branch dev from origin.

git2.0以上的git pull规则好像是这样的：如果远程的文件比本地新（修改时间），则会与本地合并，生成<===标记的冲突来手动解决（pull后直接输入命令git mergetool）；如果远程文件比本地旧，则本地文件并不会改变，仍和远程不一样。

git pull实际是git fetch和git merge，只是它把过程的细节都隐藏了起来，以至于你不用去了解git中各种类型分支的区别和使用方法

其它分支的提交

>>dev

git checkout -b dev

git push -u origin dev

16.使用 git revert <commit\_id>操作实现以退为进，

git revert 不同于 git reset 它不会擦除"回退"之后的 commit\_id ,而是正常的当做一次"commit"，产生一次新的操作记录，所以可以push，不会让你再pull

17.git diff #是工作区(work dict)和暂存区(stage)的比较

git diff --cached #是暂存区(stage)和分支(master)的比较

18.当多个文件修改完成放入暂存区的时候，发现其中一个文件的代码有问题，这时候你可以用checkout单独将这个文件还原重改；如果你将这些文件一次性全部放进库里，等你发现有问题时就不能单独拿出一个文件了，只能版本回档，那时你就要重新修改所有文件的代码。

19.git checkout -- file #撤销工作目录中所有未提交文件的修改内容

git checkout -- file可以丢弃工作区的修改：

$ git checkout -- readme.txt

命令git checkout -- readme.txt意思就是，把readme.txt文件在工作区的修改全部撤销，这里有两种情况：

一种是readme.txt自修改后还没有被放到暂存区，现在，撤销修改就回到和版本库一模一样的状态；

一种是readme.txt已经添加到暂存区后，又作了修改，现在，撤销修改就回到添加到暂存区后的状态。

总之，就是让这个文件回到最近一次git commit或git add时的状态。

20.总结撤销修改：

20.1没有git add 用这两个命令撤销

eqiasui@CN00214190 MINGW64 ~/Documents/practice2/qspractice (temp)

$ git status

On branch temp

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git checkout -- <file>..." to discard changes in working directory)

modified: a.txt

no changes added to commit (use "git add" and/or "git commit -a")

加了许多文件，全部撤销用

eqiasui@CN00214190 MINGW64 ~/Documents/practice2/qspractice (master)

$ git checkout -- .

20.2已经git add了用这个命令撤销修改：

eqiasui@CN00214190 MINGW64 ~/Documents/practice2/qspractice (temp)

$ git add a.txt

eqiasui@CN00214190 MINGW64 ~/Documents/practice2/qspractice (temp)

$ git status

On branch temp

Changes to be committed:

(use "git reset HEAD <file>..." to unstage)

modified: a.txt

20.3已经git commit了用这个命令撤销修改：

eqiasui@CN00214190 MINGW64 ~/Documents/practice2/qspractice (temp)

$ git reset --hard HEAD^

HEAD is now at 1ed5706 Merge branch 'master' of github.com:qshilary/qspractice

20.4 git reset HEAD file 从暂存区退回到工作区

eqiasui@CN00214190 MINGW64 ~/Documents/practice2/qspractice (temp)

$ git add a.txt

eqiasui@CN00214190 MINGW64 ~/Documents/practice2/qspractice (temp)

$ git status

On branch temp

Changes to be committed:

(use "git reset HEAD <file>..." to unstage)

modified: a.txt

eqiasui@CN00214190 MINGW64 ~/Documents/practice2/qspractice (temp)

$ git reset HEAD a.txt

Unstaged changes after reset:

M a.txt

eqiasui@CN00214190 MINGW64 ~/Documents/practice2/qspractice (temp)

$ git status

On branch temp

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git checkout -- <file>..." to discard changes in working directory)

modified: a.txt

no changes added to commit (use "git add" and/or "git commit -a")

eqiasui@CN00214190 MINGW64 ~/Documents/practice2/qspractice (temp)

$

20.5已经push到远程版本库了(参考13)

git revert HEAD

小结

又到了小结时间。

场景1：当你改乱了工作区某个文件的内容，想直接丢弃工作区的修改时，用命令git checkout -- file。

场景2：当你不但改乱了工作区某个文件的内容，还添加到了暂存区时，想丢弃修改，分两步，第一步用命令git reset HEAD <file>，就回到了场景1，第二步按场景1操作。

场景3：已经提交了不合适的修改到版本库时，想要撤销本次提交，参考版本回退一节，不过前提是没有推送到远程库

21.删除文件

rm test.txt

这个时候，Git知道你删除了文件，因此，工作区和版本库就不一致了，git status命令会立刻告诉你哪些文件被删除了：

$ git status

On branch master

Changes not staged for commit:

(use "git add/rm <file>..." to update what will be committed)

(use "git checkout -- <file>..." to discard changes in working directory)

deleted: test.txt

no changes added to commit (use "git add" and/or "git commit -a")

现在你有两个选择，一是确实要从版本库中删除该文件，那就用命令git rm删掉，并且git commit：

$ git rm test.txt

rm 'test.txt'

$ git commit -m "remove test.txt"

[master d46f35e] remove test.txt

1 file changed, 1 deletion(-)

delete mode 100644 test.txt

现在，文件就从版本库中被删除了。

另一种情况是删错了，因为版本库里还有呢，所以可以很轻松地把误删的文件恢复到最新版本：

$ git checkout -- test.txt

git checkout其实是用版本库里的版本替换工作区的版本，无论工作区是修改还是删除，都可以“一键还原”。

22.eqiasui@CN00214190 MINGW64 ~/Documents/practice2/qspractice (temp2)

$ git push origin temp2:master #本地分支temp2推送到远程分支master

Total 0 (delta 0), reused 0 (delta 0)

To github.com:qshilary/qspractice.git

95068ab..d4abb14 temp2 -> master

23.分支：

查看分支：git branch

创建分支:git branch <name>

切换分支：git checkout <name>

创建+切换分支：git checkout -b <name>

合并某分支到当前分支:git merge <name>

删除分支:git branch -d <name>

git merge 以后也需要git push origin temp2

我在本地用git branch -d dev删除了dev分支

但是不知道如何将这个操作同步到github上？

需要手动在github删除分支吗？

可以用git push origin :dev 删除远程分支dev，origin和:dev之间有空格

24.如何理解我们当前master分支比远程的master分支要超前1个提交

当前master分支: A->B->C->D

远程master分支: A->B->C

25.git diff --cached 或 $ git diff --staged 查看尚未提交的更新

26.git stash push 将文件给push到一个临时空间中

27.git stash pop 将文件从临时空间pop下来

28. git config --list

29.du查看目录大小，df查看磁盘使用情况。

我常使用的命令（必要时，sudo使用root权限），

1).查看某个目录的大小：du -hs /home/master/documents

查看目录下所有目录的大小并按大小降序排列：sudo du -sm /etc/\* | sort -nr | less

2).查看磁盘使用情况（文件系统的使用情况）：sudo df -h

df --block-size=GB

-h是使输出结果更易于人类阅读；du -s只展示目录的使用总量（不分别展示各个子目录情况），-m是以

MB为单位展示目录的大小（当然-k/-g就是KB/GB了）。

1. 关联远程库 git remote add origin [git@github.com:michaelliao/learngit.git](mailto:git@github.com:michaelliao/learngit.git)
2. git push origin master git push <远程库> <本地分支> <远程分支>

一 添加远程库

什么是版本库呢？版本库又名仓库，英文名**repository**，你可以简单理解成一个目录，这个目录里面的所有文件都可以被Git管理起来，每个文件的修改、删除，Git都能跟踪，以便任何时刻都可以追踪历史，或者在将来某个时刻可以“还原”。

所以，创建一个版本库非常简单，首先，选择一个合适的地方，创建一个空目录：

$ mkdir learngit

$ cd learngit

$ pwd

/Users/michael/learngit

二步，通过git init命令把这个目录变成Git可以管理的仓库：

$ git init

Initialized empty Git repository **in** /Users/michael/learngit/.git/

瞬间Git就把仓库建好了，而且告诉你是一个空的仓库（empty Git repository），细心的读者可以发现当前目录下多了一个.git的目录，这个目录是Git来跟踪管理版本库的，没事千万不要手动修改这个目录里面的文件，不然改乱了，就把Git仓库给破坏了。

把一个文件放到Git仓库只需要两步。

第一步，用命令git add告诉Git，把文件添加到仓库：

$ git add readme.txt

执行上面的命令，没有任何显示，这就对了，Unix的哲学是“没有消息就是好消息”，说明添加成功。

第二步，用命令git commit告诉Git，把文件提交到仓库：

$ git **commit** -m "wrote a readme file"

[master (root-**commit**) eaadf4e] wrote a readme file

1 file changed, 2 insertions(+)

**create** mode 100644 readme.txt

简单解释一下git commit命令，-m后面输入的是本次提交的说明，可以输入任意内容，当然最好是有意义的，这样你就能从历史记录里方便地找到改动记录。

现在总结一下今天学的两点内容：

初始化一个Git仓库，使用git init命令。

添加文件到Git仓库，分两步：

1. 使用命令git add <file>，注意，可反复多次使用，添加多个文件；
2. 使用命令git commit -m <message>，完成
3. 在GitHub上的这个learngit仓库还是空的，GitHub告诉我们，可以从这个仓库克隆出新的仓库，也可以把一个已有的本地仓库与之关联，然后，把本地仓库的内容推送到GitHub仓库。
4. 现在，我们根据GitHub的提示，在本地的learngit仓库下运行命令：

$ git remote add origin [git@github.com:michaelliao/learngit.git](mailto:git@github.com:michaelliao/learngit.git)

1. 下一步，就可以把本地库的所有内容推送到远程库上：

$ git push -u origin master

Counting objects: 20, done.

Delta compression using up to 4 threads.

Compressing objects: 100% (15/15), done.

Writing objects: 100% (20/20), 1.64 KiB | 560.00 KiB/s, done.

Total 20 (delta 5), reused 0 (delta 0)

remote: Resolving deltas: 100% (5/5), done.

To github.com:michaelliao/learngit.git

\* [new branch] master -> master

Branch 'master' **set** up **to** track remote branch 'master' **from** 'origin'.

把本地库的内容推送到远程，用git push命令，实际上是把当前分支master推送到远程。

由于远程库是空的，我们第一次推送master分支时，加上了-u参数，Git不但会把本地的master分支内容推送的远程新的master分支，还会把本地的master分支和远程的master分支关联起来，在以后的推送或者拉取时就可以简化命令。