**git branch -f main o/main** 强制移动main 到 o/main

**git checkout -b feature C2** 新建 feature 在C2 节点上

**git checkout -b <not main> o/main**; 这里创建的totally not main分支具备了跟踪远程分支的属性 (可以 git pull; git commit; git push 试验一下)

**git branch -u o/main <not\_main>;** 如果当前就在 totally\_not\_main 分支上，可以直接写 git branch -u o/main

**git push origin feature** 推送 feature 到远程仓库 Git 默认会将本地分支名作为远程分支名。因此，执行 git push origin feature 后，**远程仓库会自动创建一个名为 feature 的新分支**（如果不存在同名分支）

**git push origin main** \*\*这里main已经指定了push的目的地\*\*

**git push origin <source>:<destination>** 这里给出了 推送 的 来源分支 和 远程的去向分支

**git push origin <source>:<newBranch>** 如果推送的去向分支不存在，git会在远程新建一个分支 \*\*newBranch\*\*

**git push origin :<destination>** 传空值到远程仓库分支，会\*\*删除\*\*仓库中相应的分支

**git fetch origin :<destination>** 拉取空值到本地，会在本地新建一个<destination> 分支

**git fetch origin Foo 拉取到o/main**

**git fetch origin <source>:<destination>** 指定来源 和 下载到得位置， 这里下载到得位置就是本地的实际分支 而不是o/foo

**git fetch** 没有参数的情况下，会下载所有的提交记录到**各个远程分支**

**git pull origin <place>**

**git pull origin <source>:<destination>**

**git pull origin foo** 相当于

**git fetch origin foo; git merge o/foo**

**git pull origin bar:bugFix** 相当于

**git fetch origin bar:bugFix;**

**git merge bugFix;**

关于 **git pull = fetch + merge** 即在无参数或普通参数下 fetch 到本地的 o/main **再 merge 到当前HEAD所在的分支上** **即 pull的数据 是合并到了当前分支 与本地的跟踪分支无关**