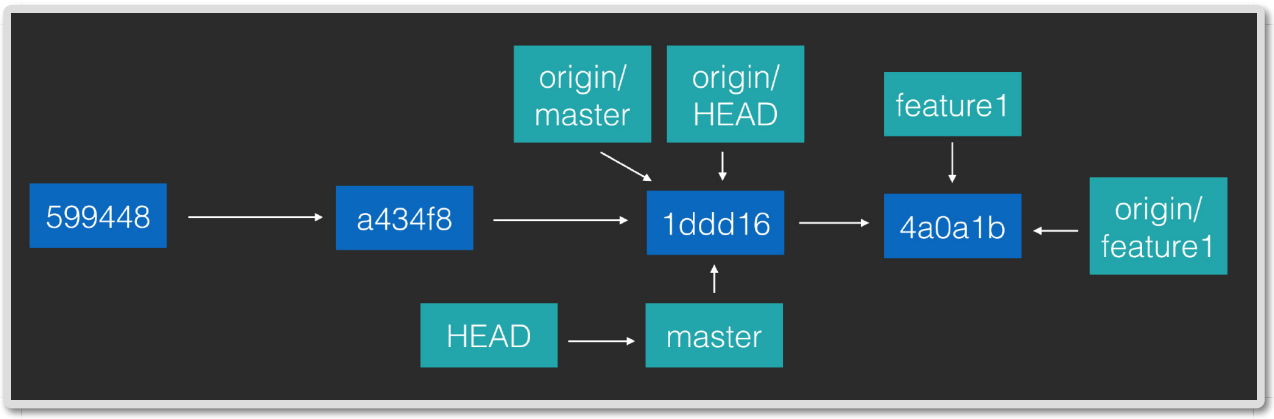
HenCoder Plus 第 27 课 讲义

Git 深入之核心概念：一切皆引用

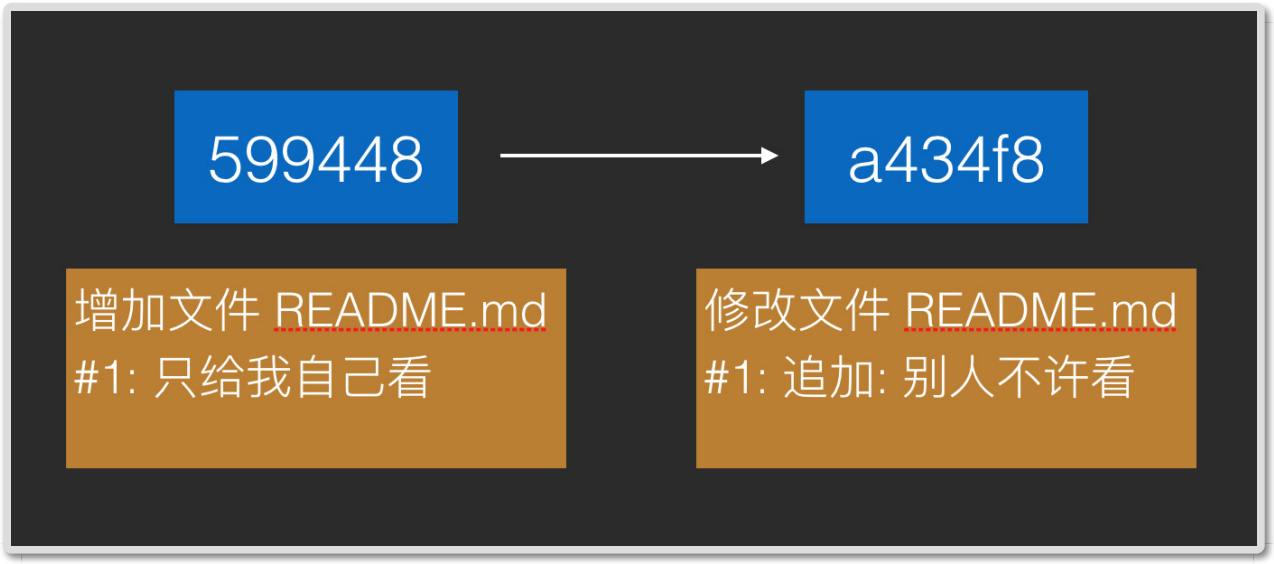
Git 仓库数据结构

- Git 仓库由一个个的 commit 组成
- 某些 commit 上会有一些 branch 指向它们，这些 branch 的本质是引用
- 有一个特殊的引用叫做 HEAD，它始终指向当前的位置，这个位置可以是 commit，也可以是 branch



常用概念：commit

commit 表示对于一次改动的提交，它可以代表当前时刻下 Git 仓库的完整快照，但本质上，commit 只是记录了距离上一次 commit 之间的改动。




```
git commit
```

常用概念：staging area 暂存区和 add

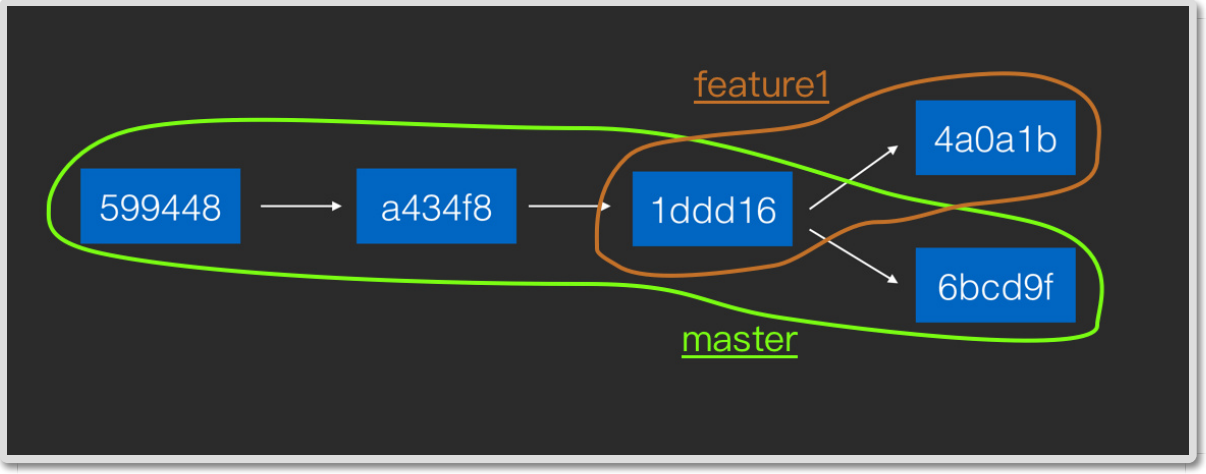
- staging 原意：舞台表演前的筹划准备（例如汇集道具和演员）。Git 中的意思：把改动内容汇集起来以待提交。
- staging area：待提交的修改内容暂时存放的地方。主要用于和已经改动但不打算提交的内容区分开来。
- add 指令：把指定的内容放进暂存区。

```
git add README.md
```

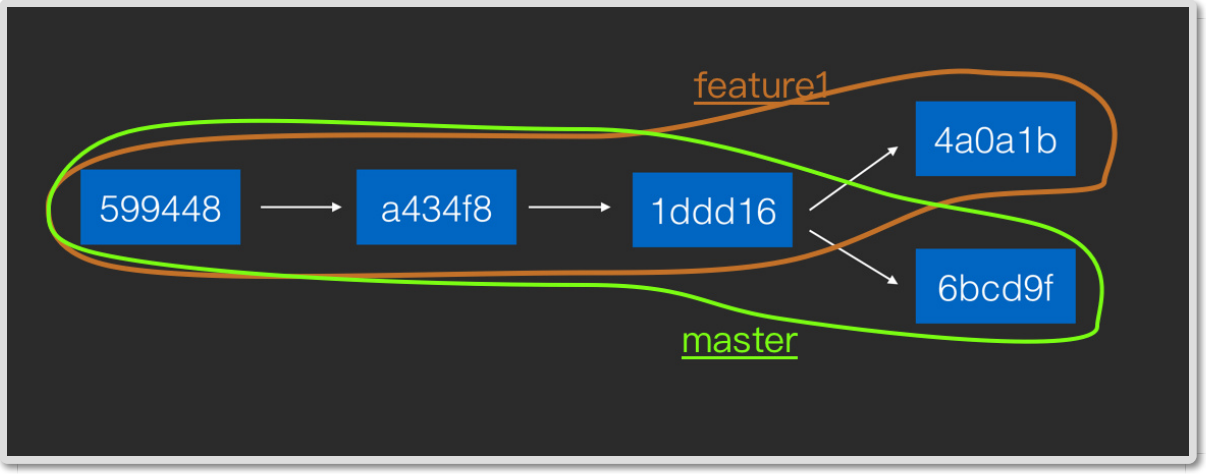
常用概念：branch 和 master

- branch 的含义是分支，指的是仓库结构出现分叉时的不同的「叉」
- 本质上，git 的 branch 是引用（reference），即指向某个 commit 的指针

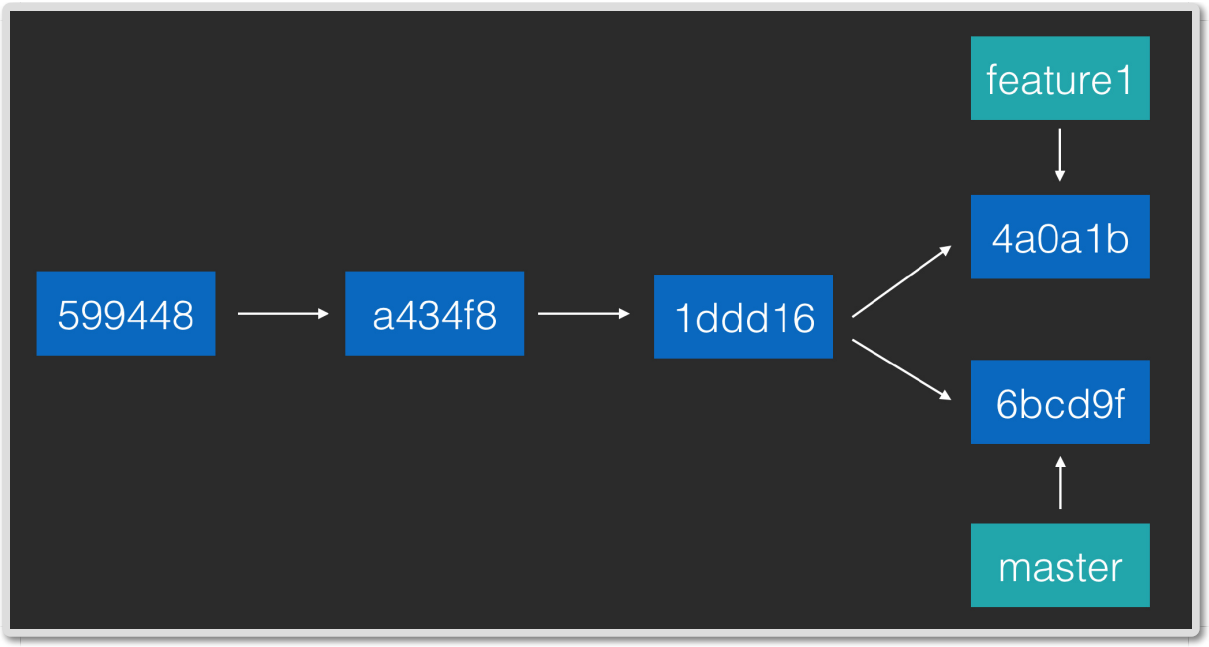
> 直观感觉的 branch：



> 或：



> 而实质上的 branch：



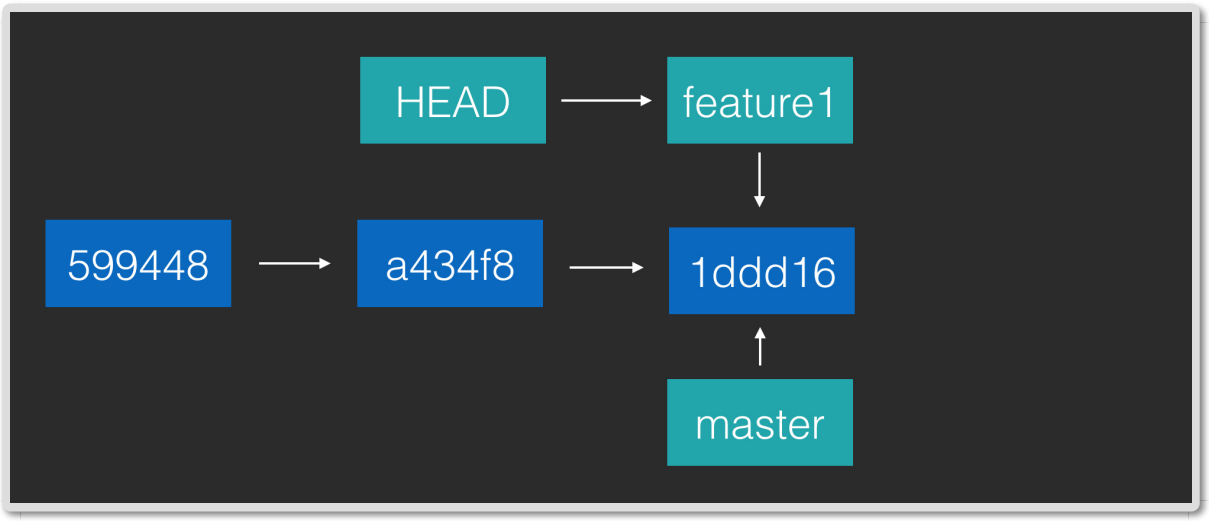
所以，branch 和你什么时候创建的它无关，也和仓库的起点无关，只和它当前指向哪个 commit 有关。

- master 是一个特殊的 branch，因为它是 Git 的默认 branch（默认 branch 可以修改）。默认 branch 的特点：
  - 执行 clone 方法把仓库取到本地的时候，默认 checkout 出来的是默认 branch，即 master；
  - 在执行 push 命令把本地内容推送到远端仓库的时候，远端仓库的 HEAD 永远跟随默认 branch，而不是和本地 HEAD 同步。换句话说，只有 push master 分支到远端的时候，远端的 HEAD 才会移动。

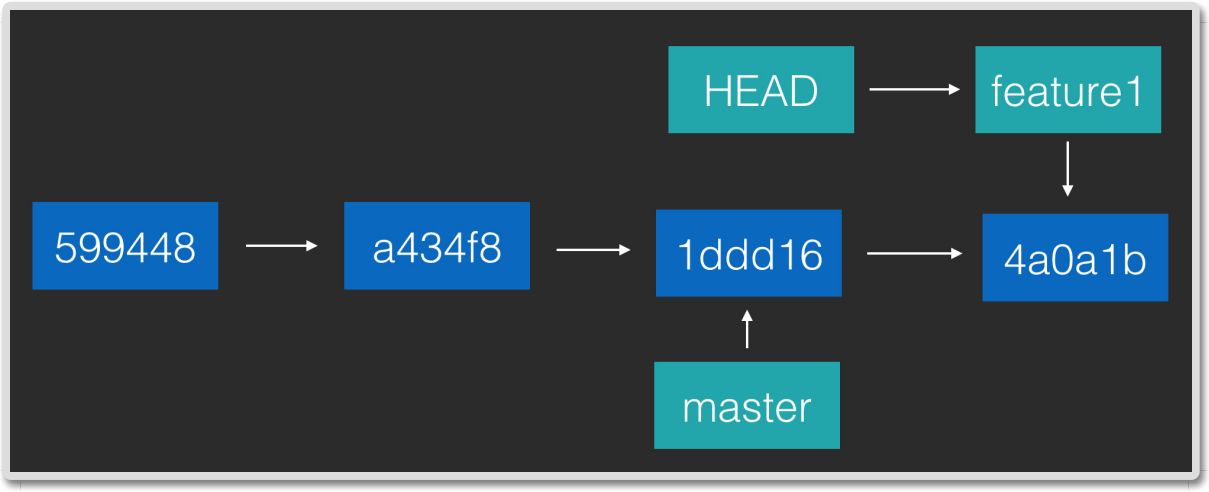
常用概念： HEAD

- HEAD 也是引用，但它不是 branch，它代表了当前所处的位置。HEAD 不仅可以指向某个 commit，也可以指向某个 branch（例如 master、feature1）
- 当每次 commit 的时候，HEAD 不仅随着新的 commit 一起移动，而且如果它指向了某个 branch，那么它也会带着 branch 一起移动

> commit 前



> commit 后



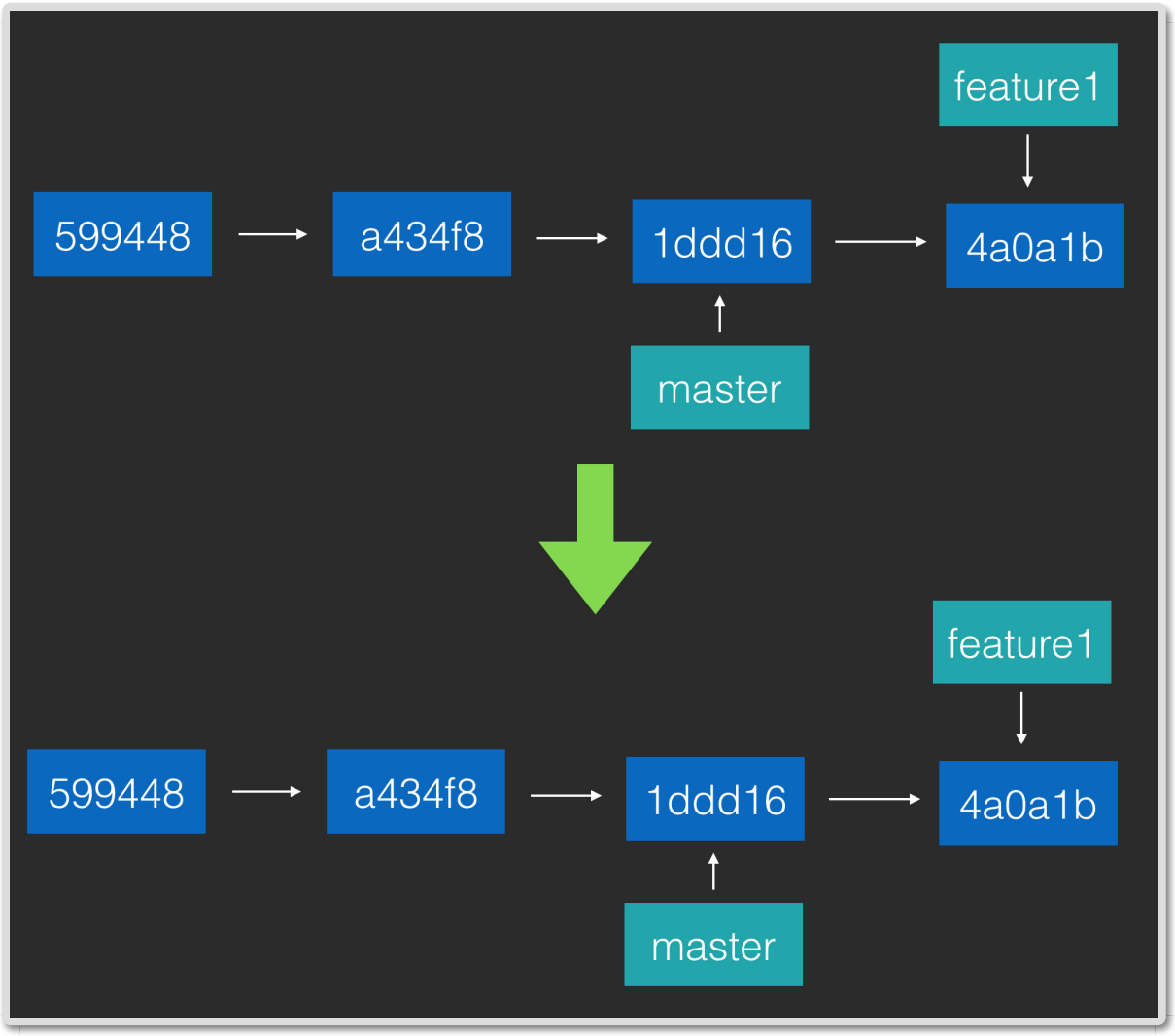
常用概念：clone

clone 是从远端仓库初次把数据取下来：

```
git clone https://github.com/rengwuxian/git-demo.git
```

clone 命令具体会做两件事：

1. 把整个仓库中的所有 branch 取下来，并把从初始 commit 到达这些 branch 的路径上的所有 commit 都取下来：



2. 从初始 commit 开始，向 master 指向的 commit，一个个地把每个 commit 应用，最终得到一个「当前」状态的仓库内容，写进 Git 所在的目录（这个目录叫做 working tree）

常用概念：log

```
git log
```

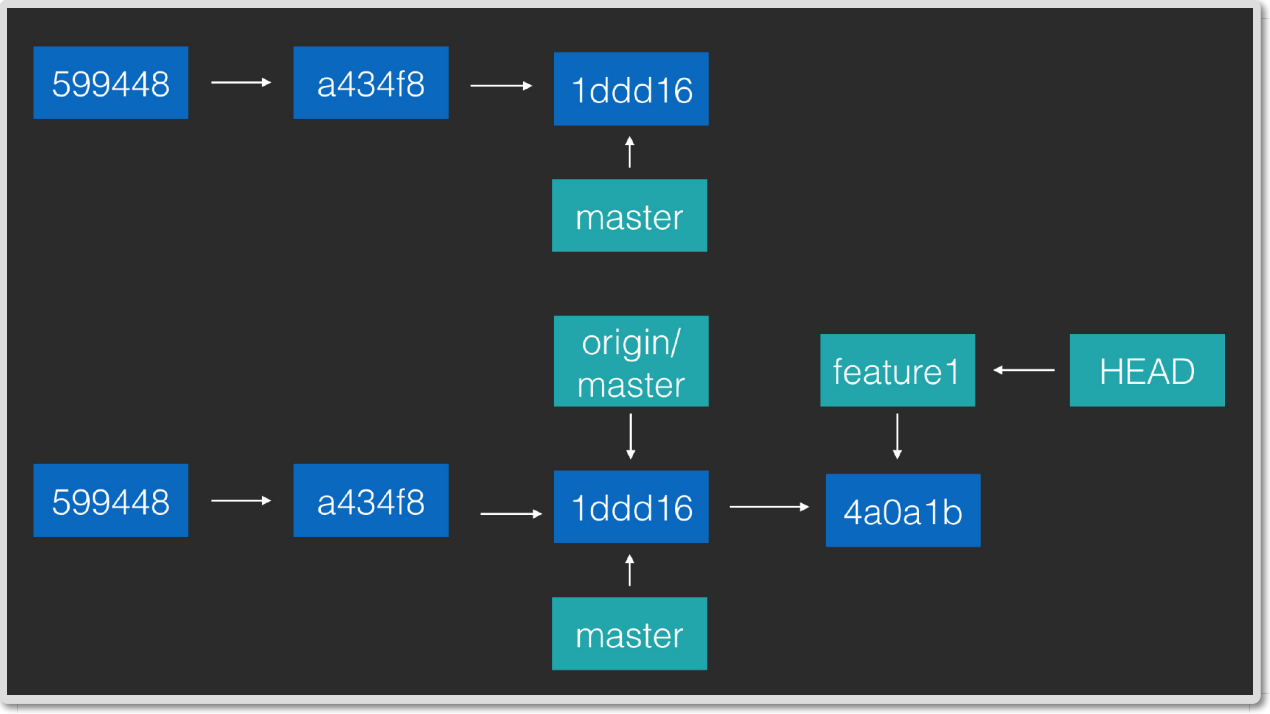
从 HEAD 指向的 commit 开始，倒序显示每一个 commit 的摘要信息

```
commit c8e8d3c1d0975f725691597732c5603b4f0a61fe (HEAD ->
 master)
Author: Kai Zhu <rengwuxian@gmail.com>
Date:   Tue Aug 14 22:04:29 2018 +0800

    status2

commit fce5595ddd0b25608af34458ad8529f4c1e8b3b9
Author: Kai Zhu <rengwuxian@gmail.com>
Date:   Tue Aug 14 22:00:29 2018 +0800

    status

commit 896dc9ee7e83be55f0e13d5aa185c2604d466ae4
Merge: f50b196 4a0a1bb
Author: Kai Zhu <rengwuxian@gmail.com>
Date:   Tue Aug 14 21:51:44 2018 +0800

    Merge branch 'feature1'
```

关于 `origin/` 打头的 branch

本地仓库中，有一些以 `origin/` 打头的 branch，它们是远端仓库（别名为 origin）的本地镜像。它们的作用是方便在本地查看远端仓库的 branch 状态。

> - 远端仓库默认名称是 origin，但也可以给它们起别的名称



`origin/` 分支并不在本地直接操作，它们一般只在两种情况下会进行自动更新：

1. 在执行 push 的时候，push 成功后，push 成功的 branch 会把它对应的 `origin/branch` 更新到当前 commit（因为远端的 branch 已经随着 push 的成功而更新，所以本地镜像也一起更新）
2. 在执行 pull 或者 fetch 的时候，由于从远端拿到了所有最新的 branch 状态，所以也会一同更新所有的 `origin/` branch

> 关于 origin/HEAD：这是一个永远跟随 origin/master 的引用，它最大的作用使用来标记默认 branch

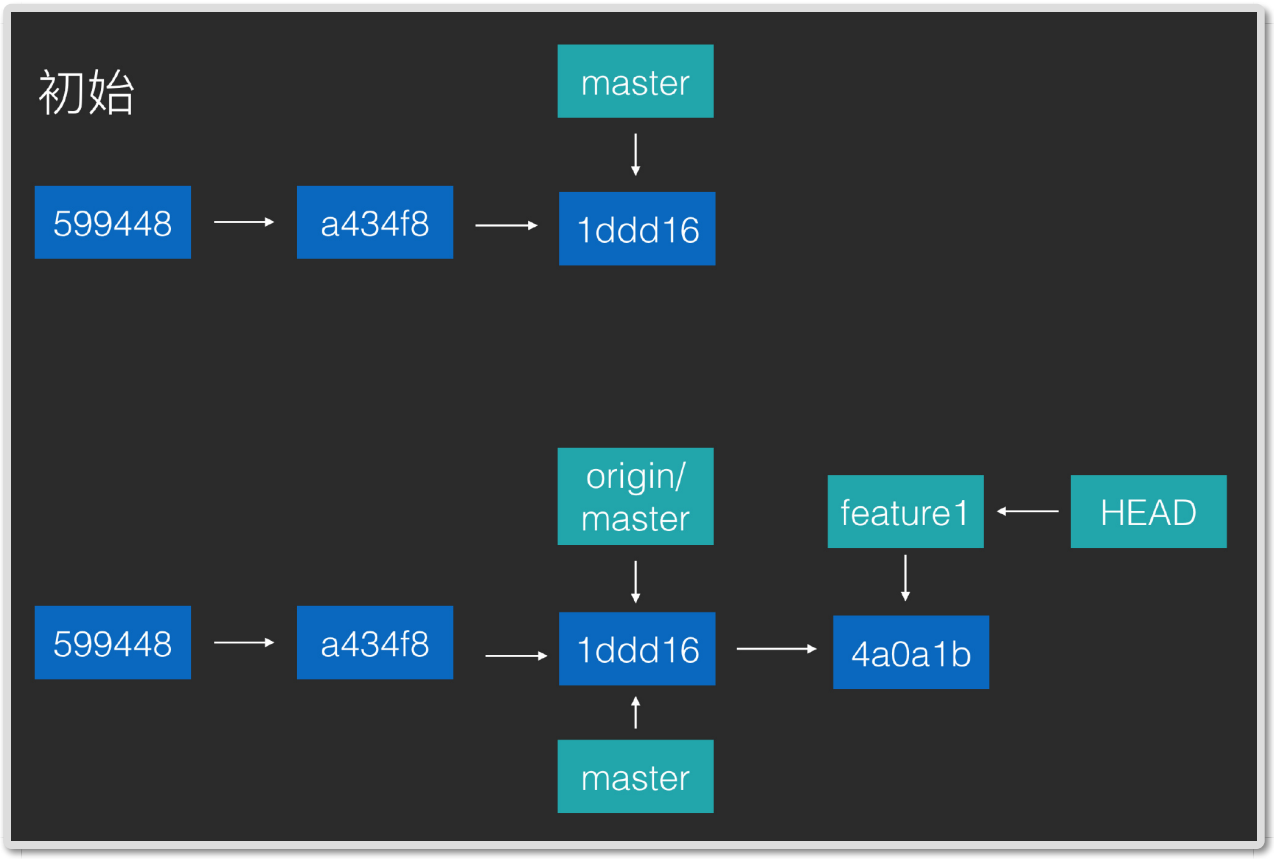
常用概念：push

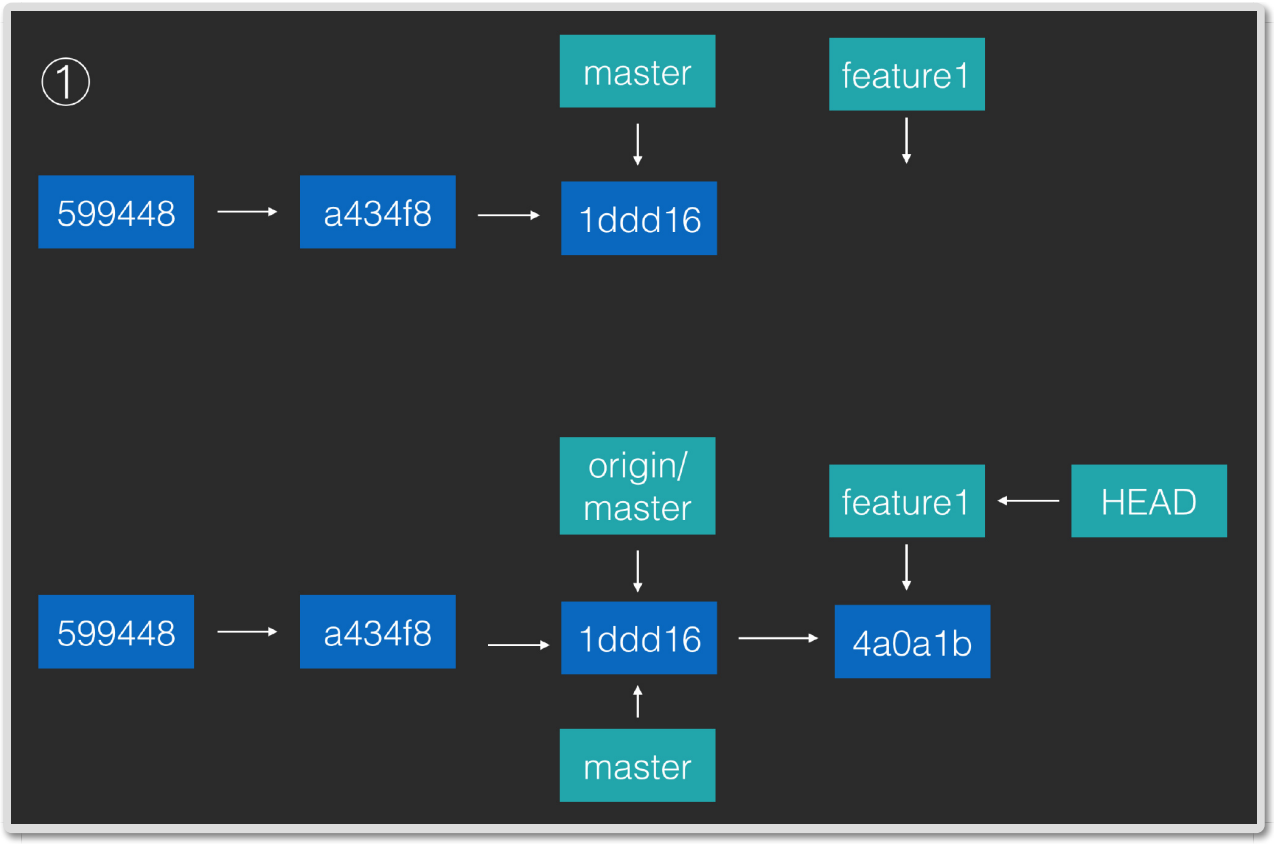
```
git push origin feature1
```

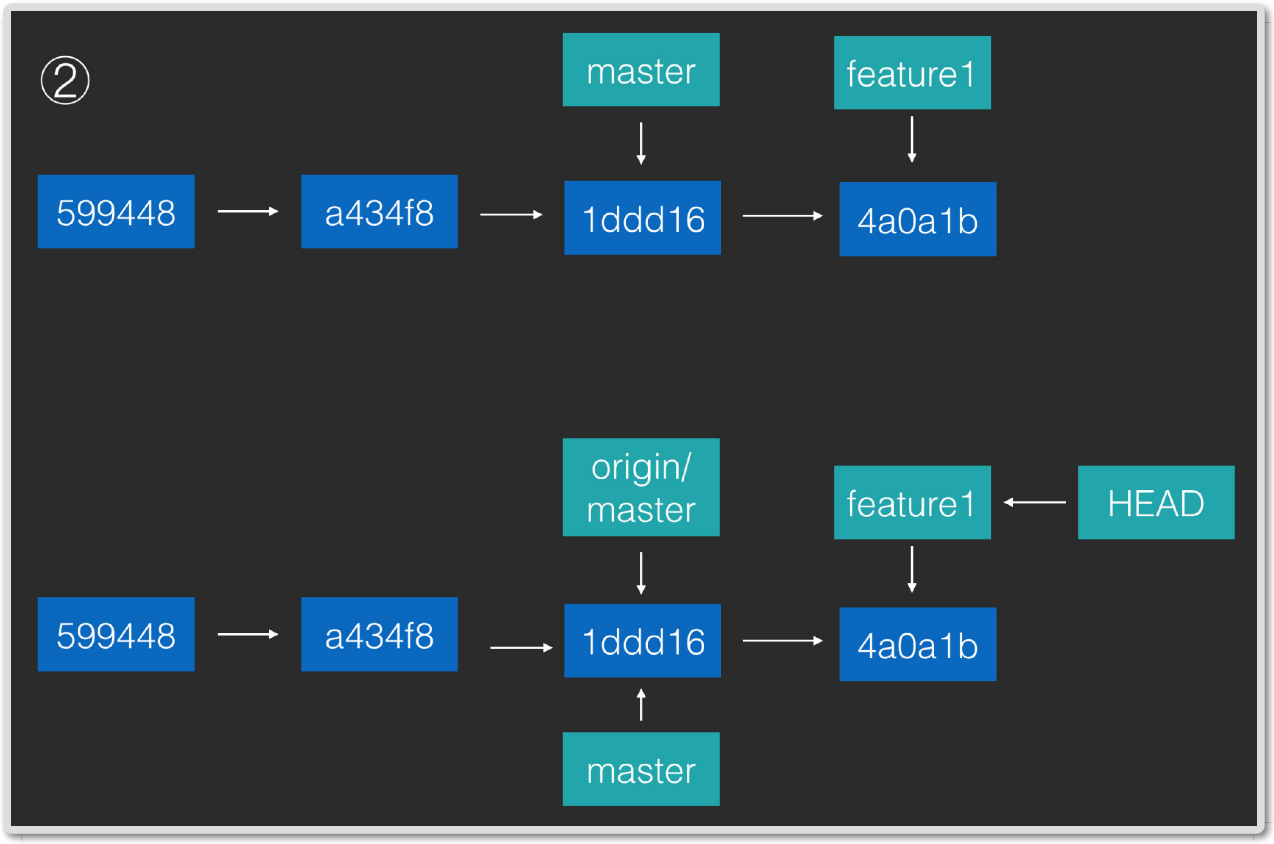
把当前 branch 推送到远端仓库的指定分支。

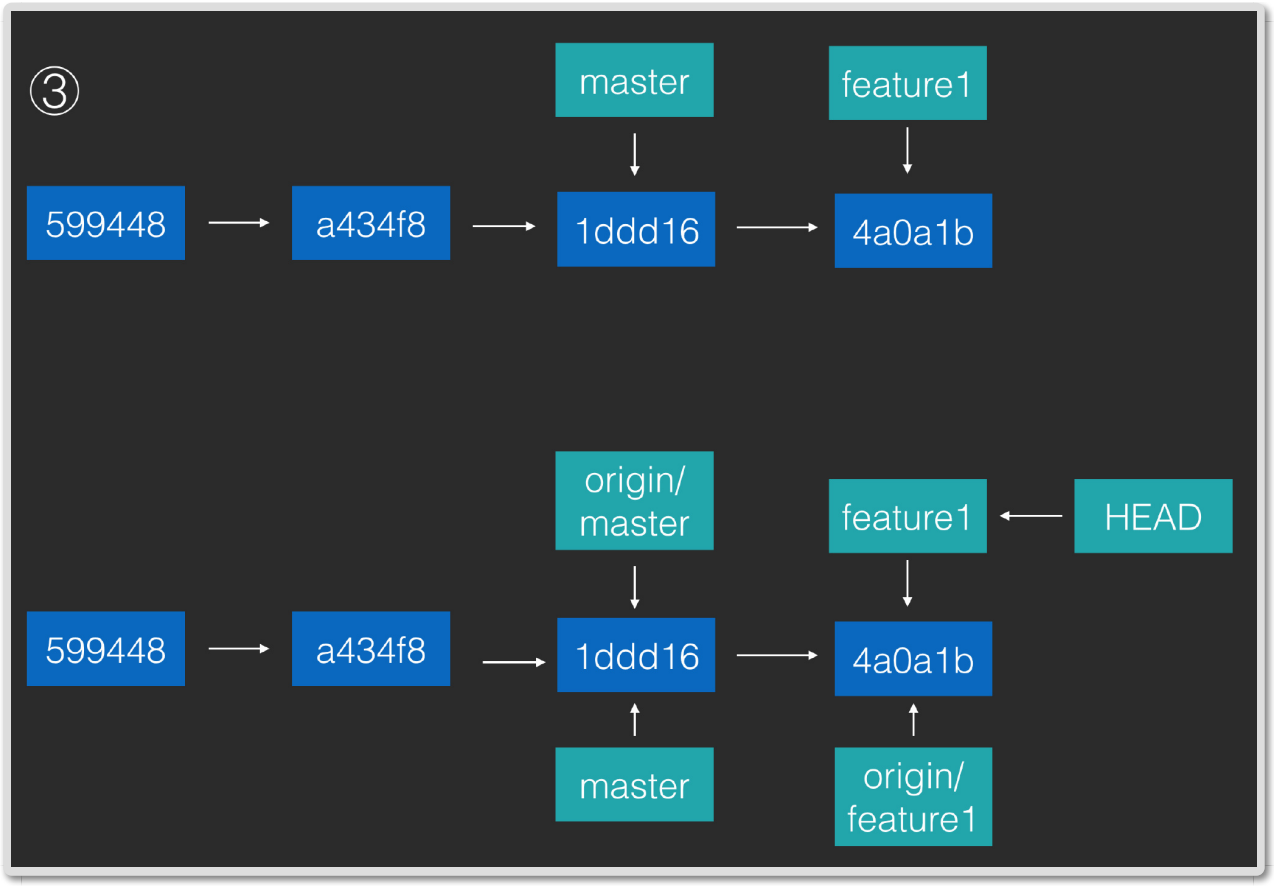
具体做两件事：

1. 把 HEAD 所指向的 branch （只是一个引用哦）推送到远端仓库
2. 从这个 branch 向前回溯，远端仓库缺少的每一个 commit 也推送到远端仓库。
3. 将 push 的 branch 的本地镜像 `origin/xxx` 更新




①
master
feature1
599448
a434f8
1ddd16
origin/
master
feature1
HEAD
599448
a434f8
1ddd16
4a0a1b
master


②
master
feature1
599448
a434f8
1ddd16
4a0a1b
origin/
master
feature1
HEAD
599448
a434f8
1ddd16
4a0a1b
master



> 注意：origin/HEAD 并没有在图上画出来，但如果 push 是 master（即默认 branch），那么本地的 `origin/HEAD` 也会更新到 master 的最新位置；当 push 的是其他 branch 的时候，`origin/HEAD` 并不会更新。
>
> 也就是说，`origin/HEAD` 只和默认分支相关，和 `HEAD` 是无关的。

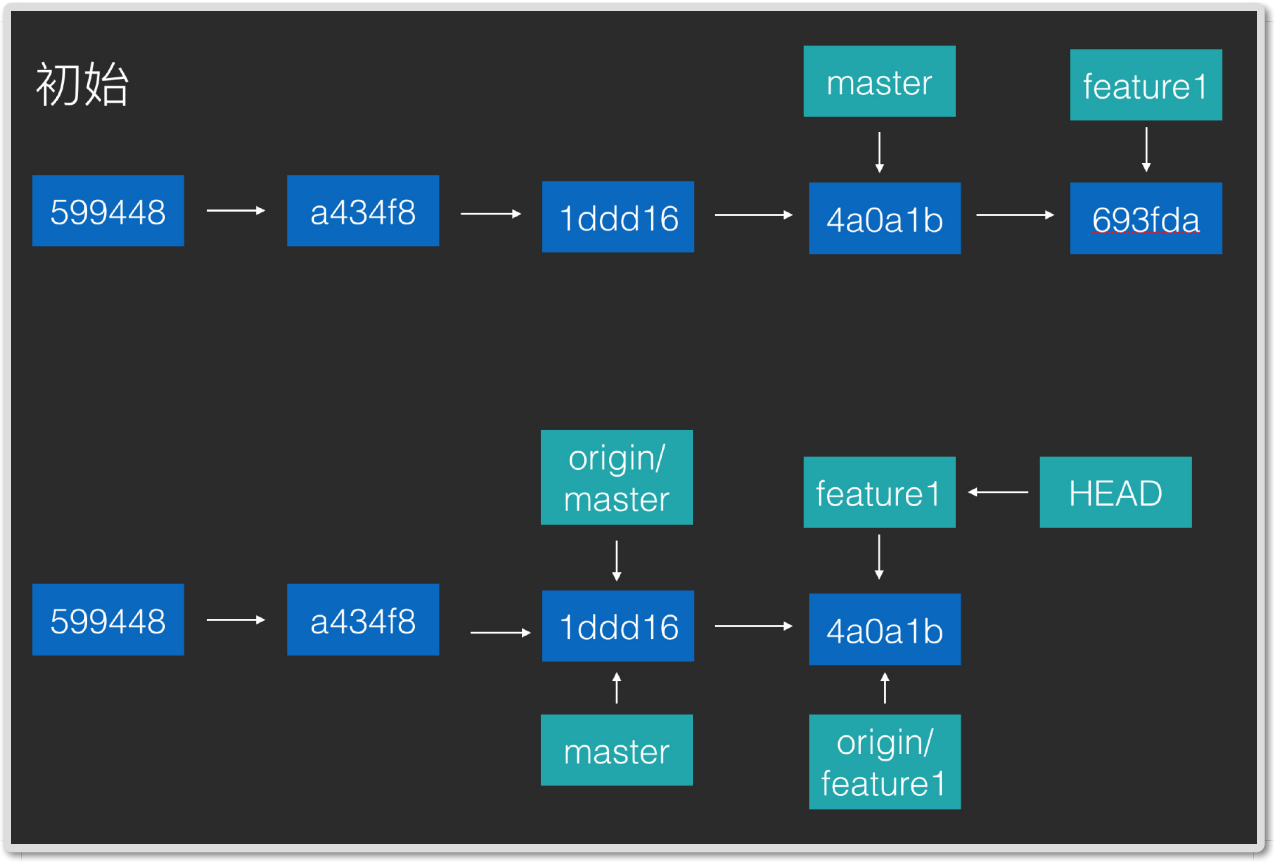
常用概念：pull

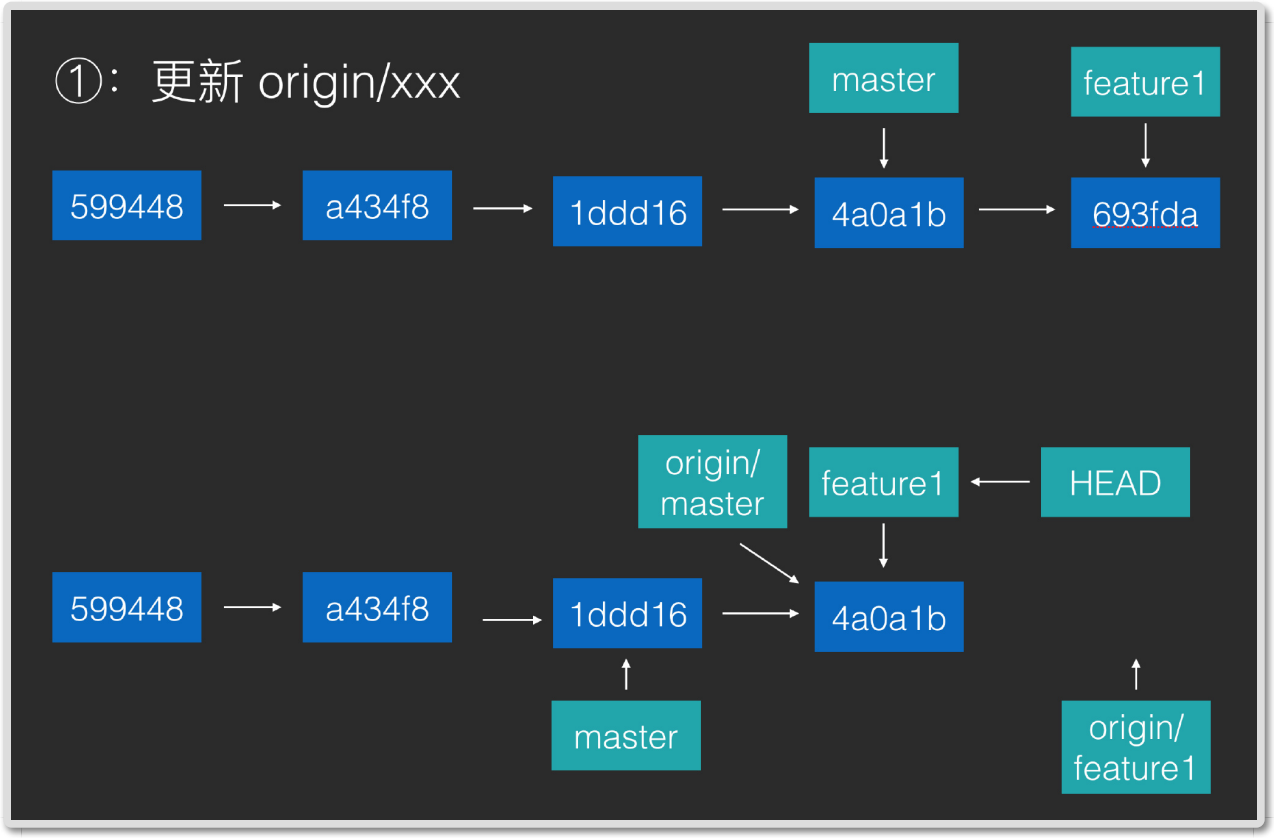
```
git pull origin feature1
```

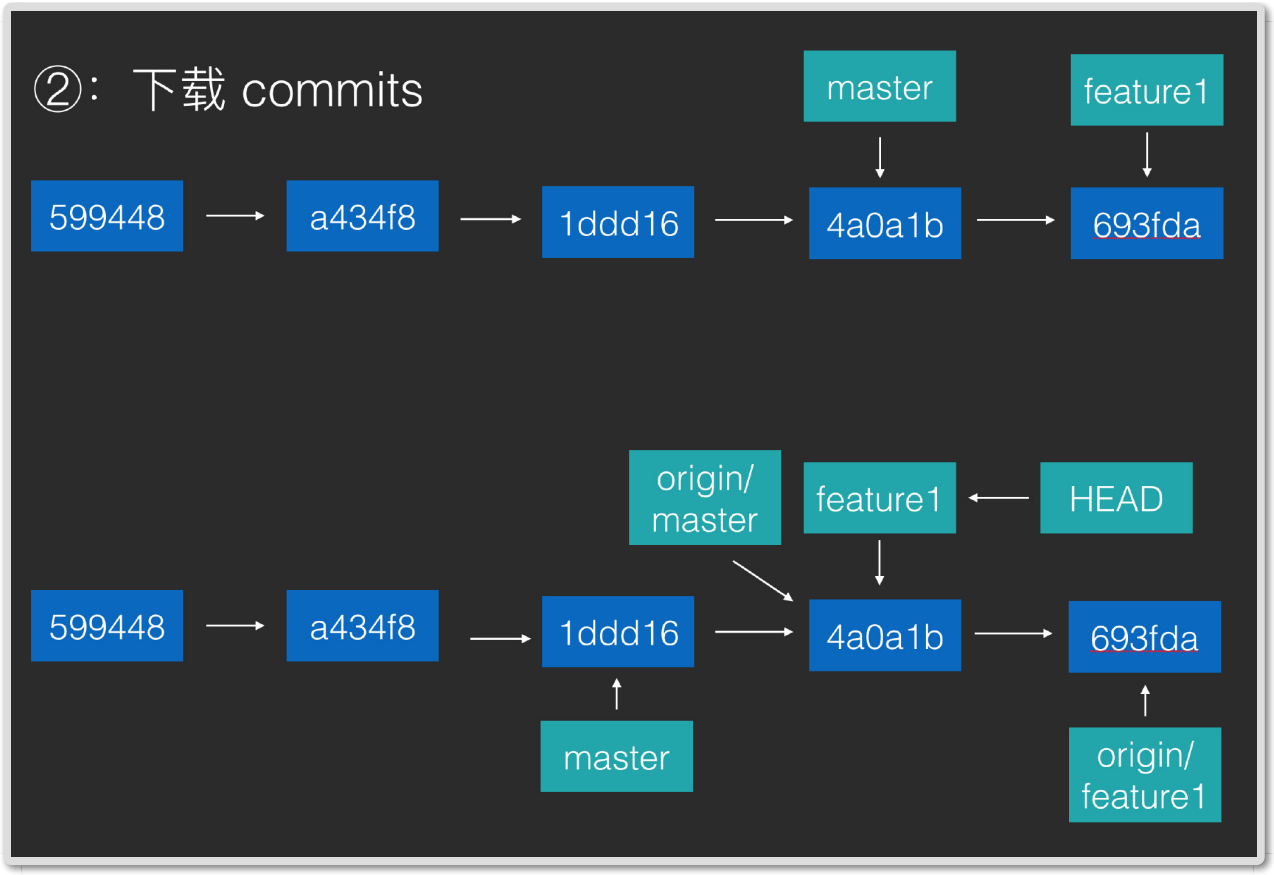
把远端 branch 取到本地。

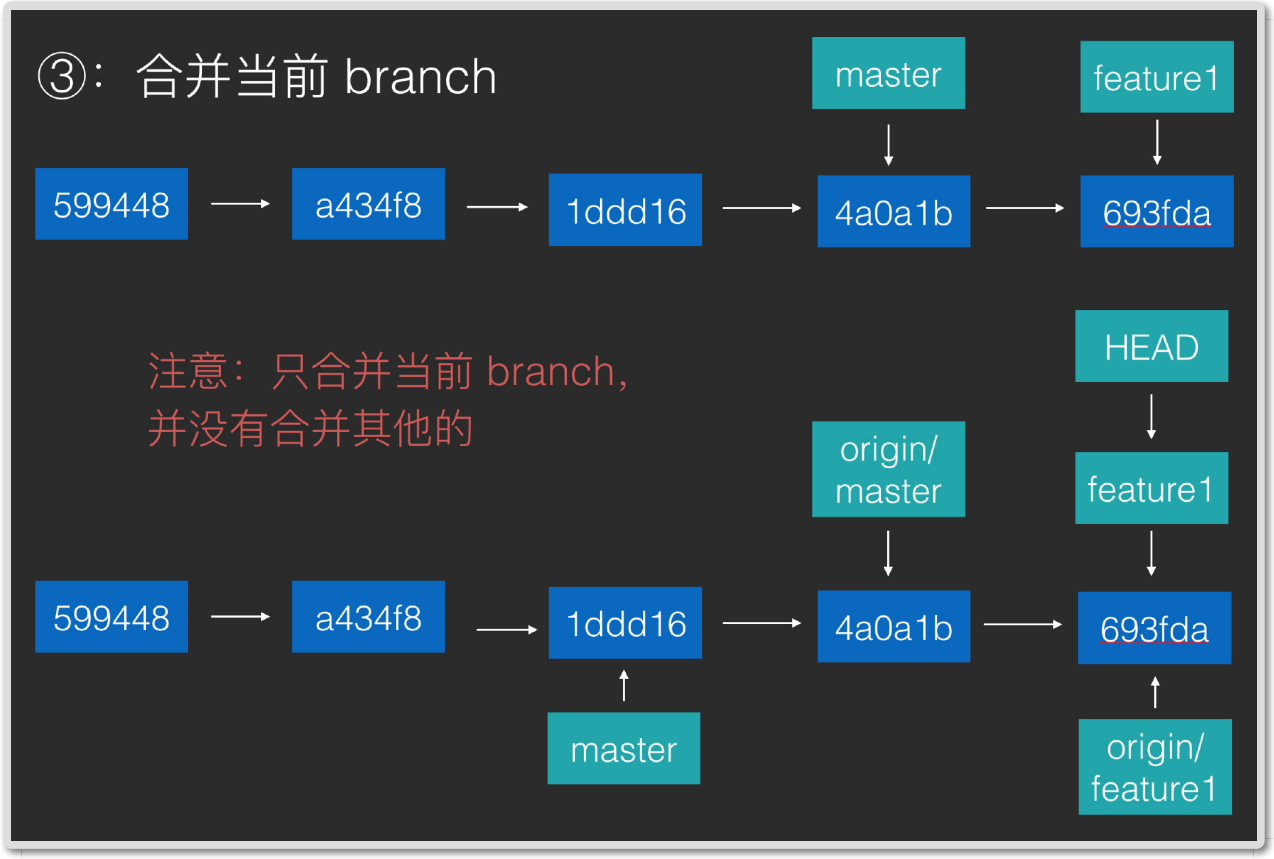
具体做的事有三件：

1. 把远端所有 branch 的最新位置更新到本地的 `origin/xxx` 镜像
2. 要到达这些 branch，本地所缺少的所有 commit，也取到本地
3. 把 `origin/当前branch` 的内容合并到当前 branch


初始
master
feature1
599448
a434f8
1ddd16
4a0a1b
693fda
origin/
master
feature1
HEAD
599448
a434f8
1ddd16
4a0a1b
master
origin/
feature1


①：更新 origin/xxx
master
feature1
599448
a434f8
1ddd16
4a0a1b
693fda
origin/
master
feature1
HEAD
599448
a434f8
1ddd16
4a0a1b
master
origin/
feature1





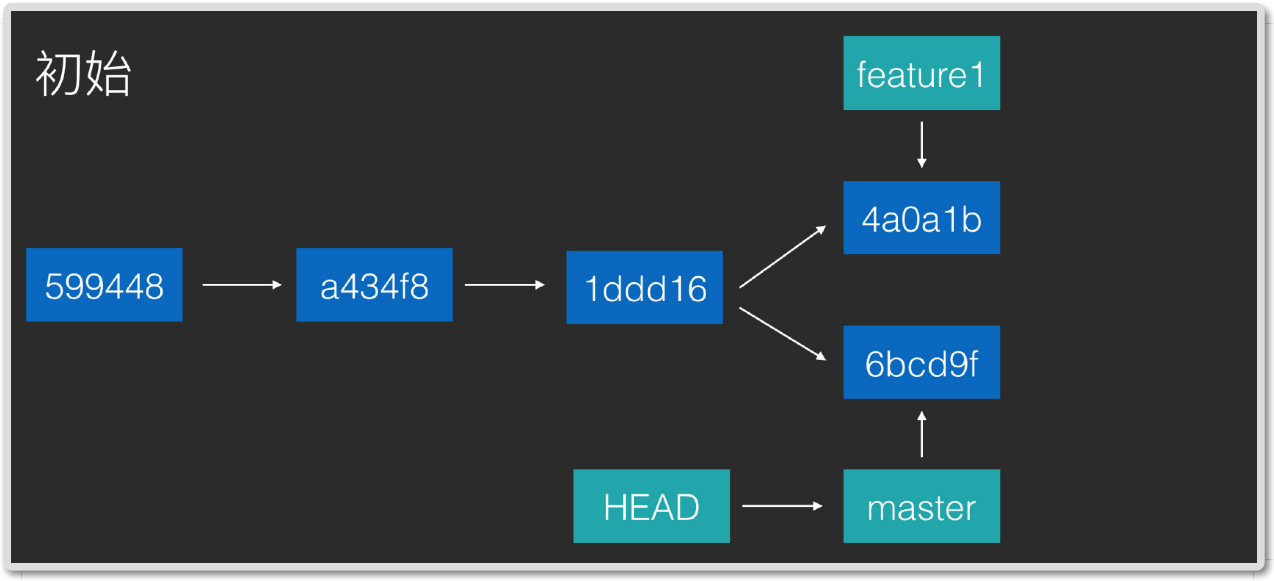
事实上，`git pull origin feature1` 会分成两部执行，它等价于下面两行：

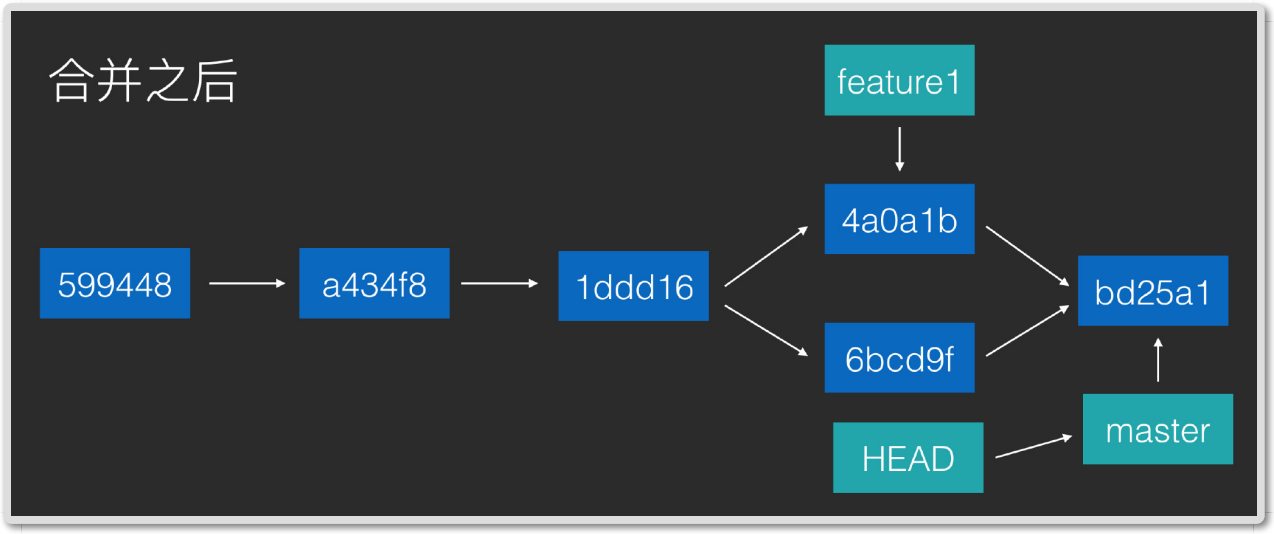
```
git fetch
git merge origin/feature1
```

常用概念：merge

merge 就是合并，它会把当前 commit 和指定 commit（所谓 commit，可以直接用它的 hash 值来指定，例如 `4a0a1b`，也可以用一个直接或间接指向它的引用来指定，例如 `master` 或者 `HEAD`）进行合并，并把这个合并行为创建成一个新的 commit。

```
git merge feature1
```





merge 行为所产生的 commit，是一种特殊的 commit：

1. 它不需要有改动，只要指定两个（或更多个）父 commit 就好
2. 正如上面这句说的，它有两个或更多个父 commit，这是一般的 commit 不具有的性质