1. **git的工作流程**

在git的分布式工作流程中，每一位开发者都可以在本地的版本库中修改和更新代码，并将更新后的版本库推送到远程仓库中。

1、**创建本地仓库**

两种方式：

① 使用git clone 将远程仓库克隆到本地。

② 使用git init 先创建一个本地仓库，随后通过git remote add 添加远程仓库。

2、**管理本地仓库**

git本地有三个工作区域：

① 工作目录（Working Directory）

② 暂存区 (Stage)

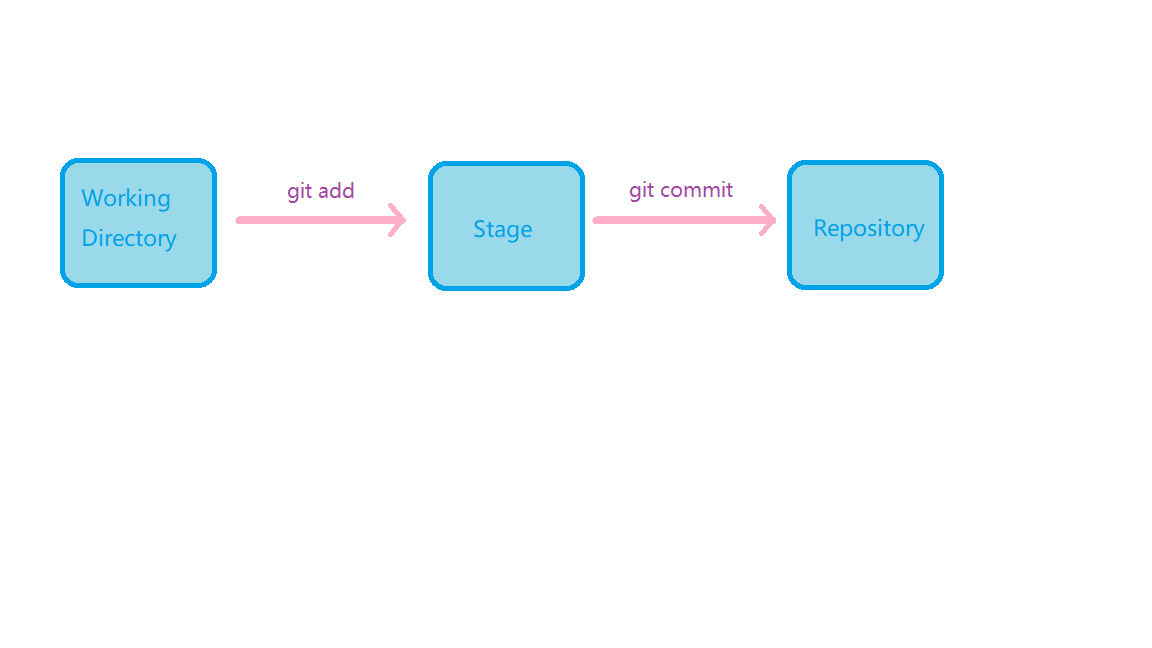
 ③ 版本库 (Repository)

图1

三个工作区域如图1所示，在工作目录中的文件处于未跟踪状态，使用git add 可以将工作目录中的文件加入暂存区中，让其进入被跟踪状态，这个时候文件还没有真正进入仓库，使用git commit 可以将其提交到本地仓库中。

3、**更新与推送**

在本地仓库中修改和更新代码后，要想真正为团队的工作做出贡献，需要将更新后的版

本库推送到远程仓库中，如果远程仓库中有版本更新，也可以及时拉取到本地仓库。

1. 拉取更新：运行git pull，从最初克隆的远程仓库上抓取数据并尝试合并到当前分支。
2. 推送：git push <远程仓库名> <远程分支名>，将当前分支所指向的提交对象推送到远程仓库的分支中。

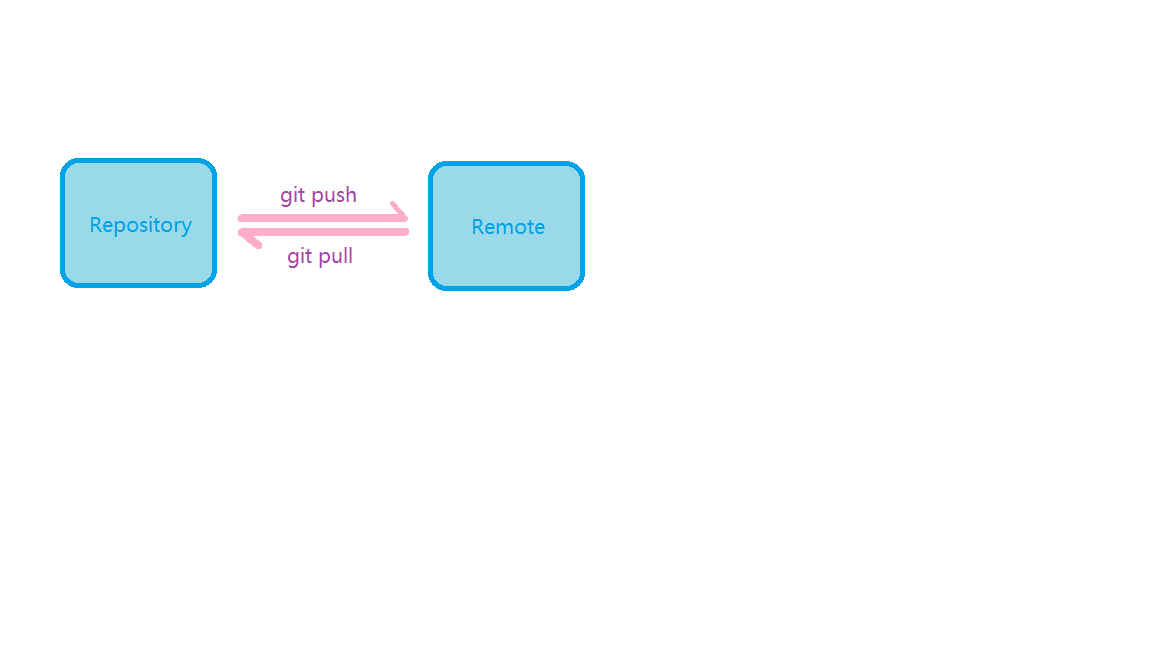


图2

1. **git的分支**

git中的分支，实际上是一个指向提交对象的指针，创建新的分支，其实就是在当前分支对应指针所指向的提交对象上创建一个新的指针。每一次提交新的版本到本地仓库时，都会将当前分支所对应的指针移动到新的提交对象，而其他分支所对应的指针并不会发生移动，因此，每个分支所对应的指针总是指向在该分支下提交的最新版本。

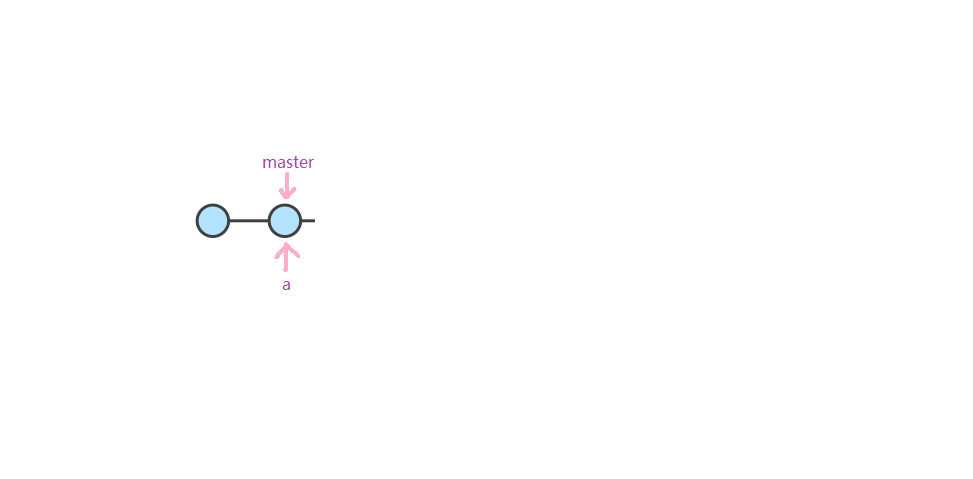
如图3所示，每一个圆圈代表一个提交对象，假设现在有两个分支，分别为master和a，它们指向同一个提交对象，即最新的提交对象上有两个代表分支的指针。

图3

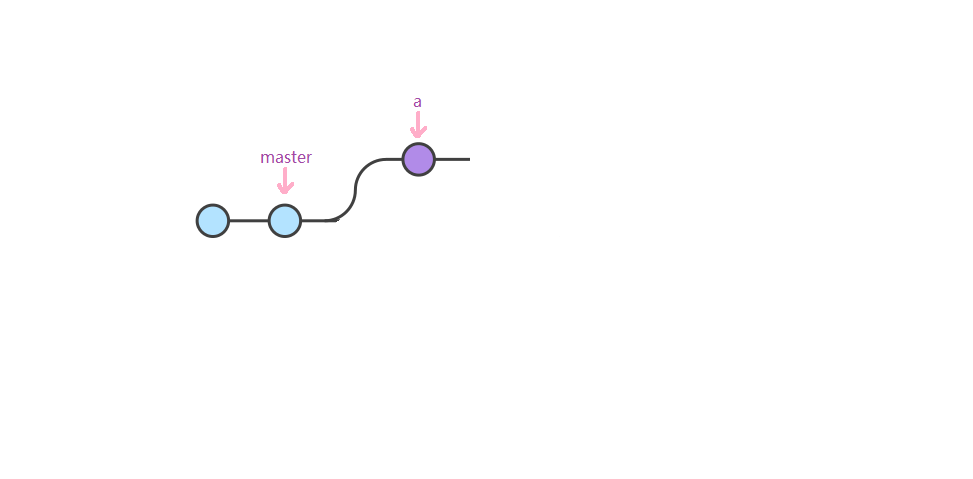
现在使用git checkout a命令可以切换a分支(指针)，然后修改仓库中的文件并提交，a分支对应的指针便移动到新的提交对象上，而master分支仍然指向上一个提交对象，结果如图4所示。

图4

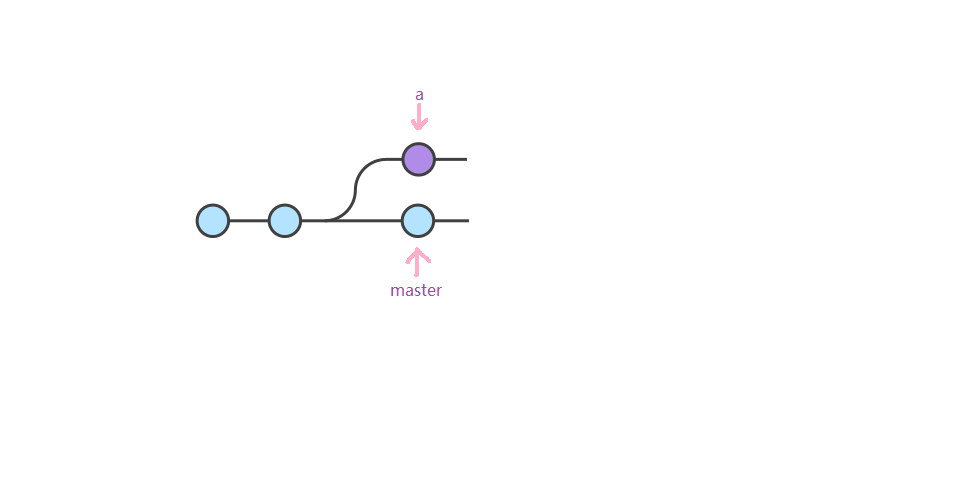
如果我们再使用git checkout master 命令切换回master分支，然后修改master所指向的版本并提交，同样会出现一个新的提交对象，而master也会指向这个新的提交对象，但这个时候提交对象已经有了两条支路，如图5所示，a和master分别指向两条支路上最新的提交对象，大概这也是分支一词的缘由。

图5

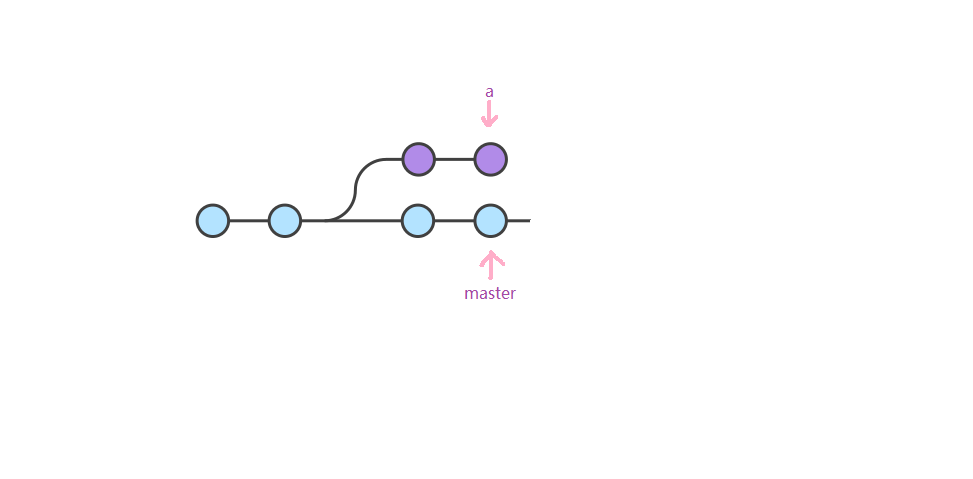
我们可以继续分别在a和master所指向的提交对象的基础上进行更新，从而使得a和master在各自的提交对象支路上渐行渐远。

图6

对于多分支的情况也是同理，每次创建的新分支都会和当前分支指向同一个提交对象，但是切换分支并提交更新之后，就会开始出现支路。

**分支的合并**

关于分支的合并，应该分两种情况：

1. 要合并的两个分支处于同一条支路上，如下图：

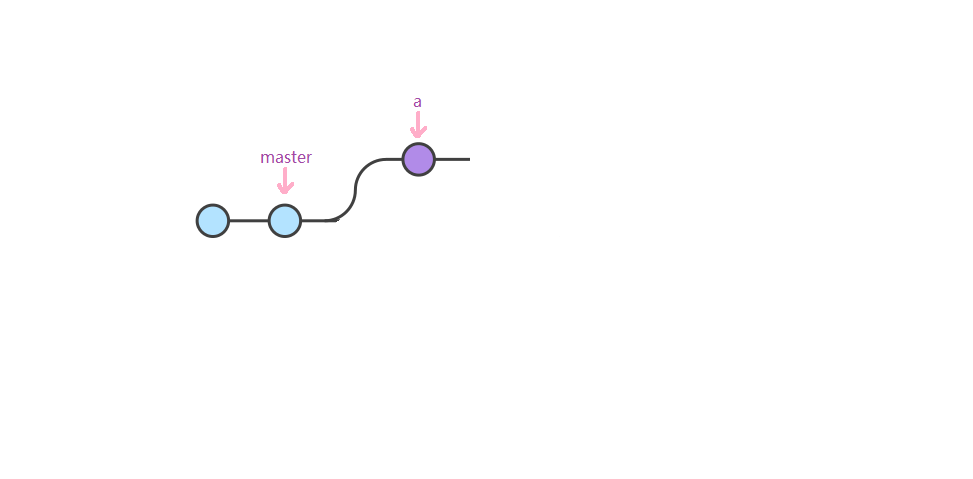


图7

git checkout master 切换到master分支，再git merge a 将master合并到a，实际上进行的操作只是将master指针向前移动至a指针的位置，如图8所示。

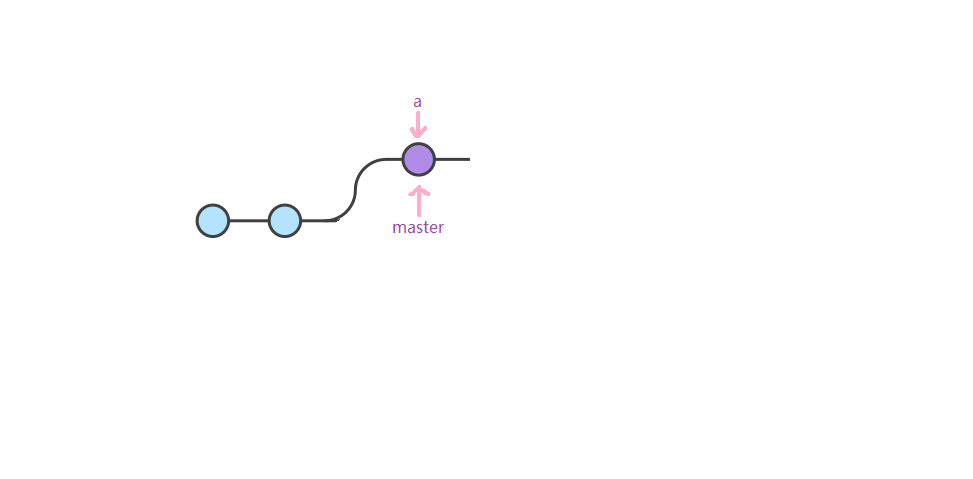


图8

1. 要合并的两个分支在不同支路上，如下图：

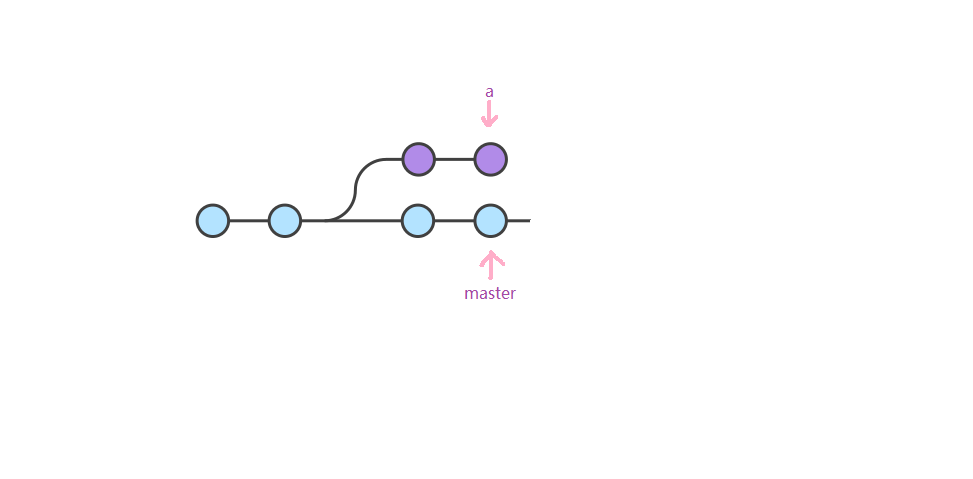


图9

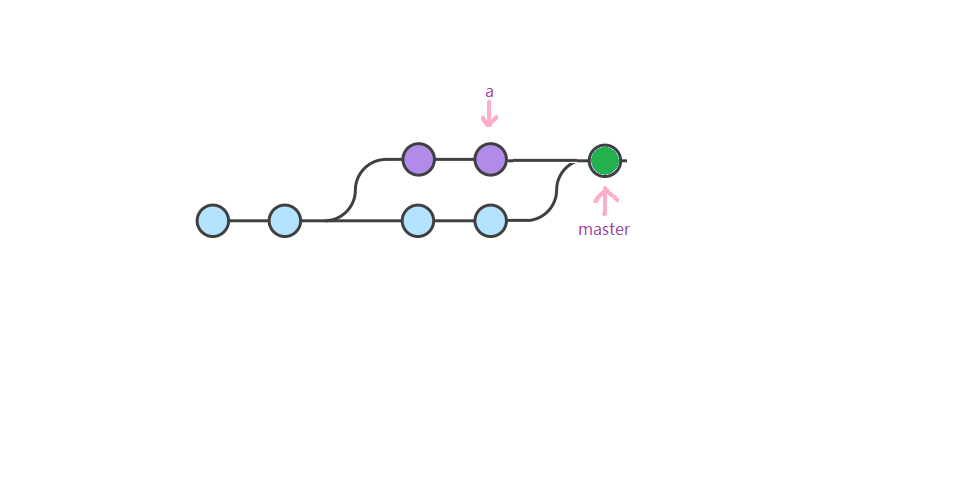
同样git checkout master 切换到master分支，再git merge a 进行合并，这个时候git会做一些额外的工作，将a和master所指的提交对象和它们的共同祖先(开始出现分叉的提交对象)做一个三方合并，并创建一个新的提交对象(下图中绿色圆圈)，指向原先a和master所指对象，再将master移动到这个新的对象上。

图10

在考虑分支合并时，避免合并两个修改同一文件同一部分的分支，以防产生冲突，最好能对当前提交对象的结构和各分支对应指针所在位置有大致的概念，才不至于合并后分支结构混乱。如果合并前的分支已经不需要，在合并后可以进行删除。此外，如果对分支的结构不清楚，可以使用gitkraken，在GUI界面中对分支进行操作。