

Git 与版本管理

1

版本管理

多人协作项目代码怎么管理

- 用 U 盘或者网盘保持同步?
- 用 SVN?
- 你听说过 Git 吗?

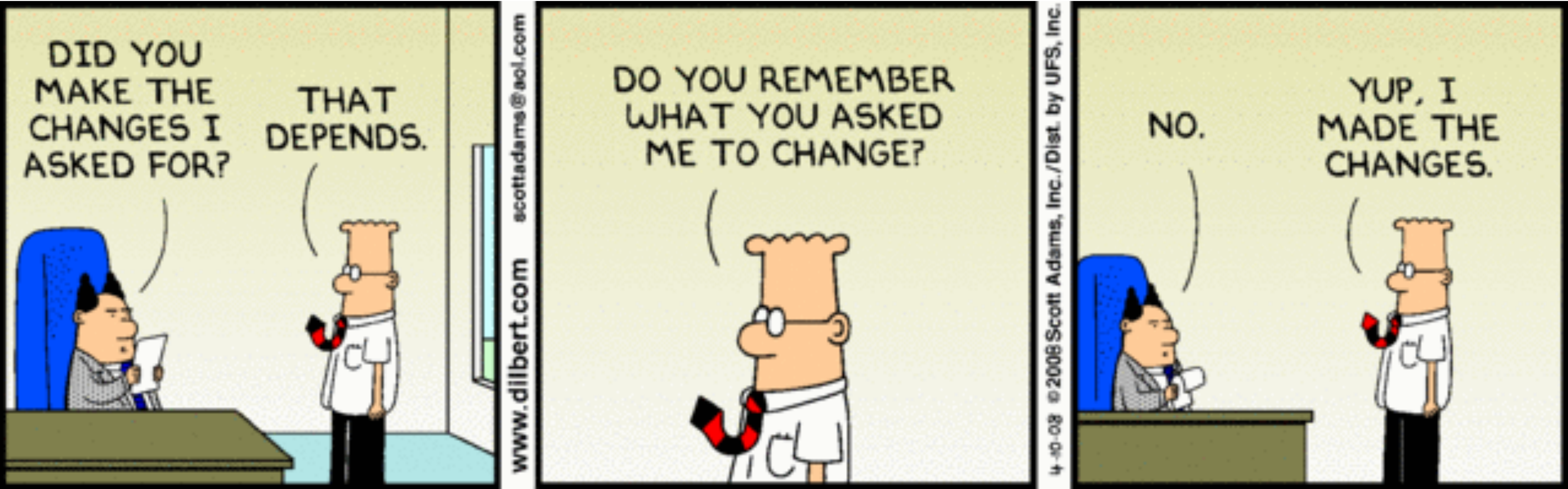


据说你们的学长曾经这么做

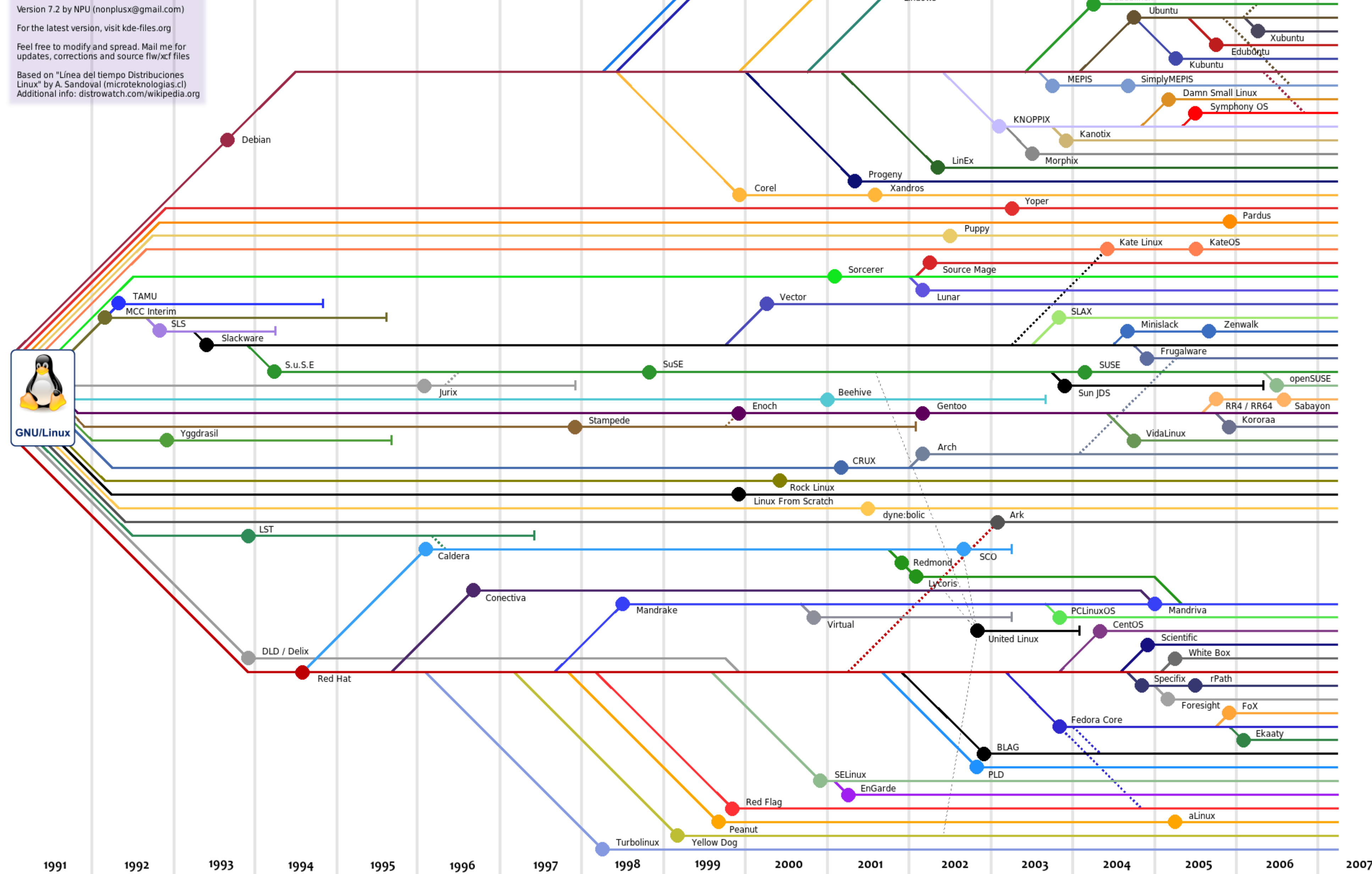
- 毕业论文
- 毕业论文1
- 毕业论文2
- 毕业论文改
- 毕业论文改1
- 毕业论文完成版
- 毕业论文完成版1
- 毕业论文最终版
- 毕业论文最终版1
- 毕业论文最终版2
- 毕业论文最终绝不改版
- 毕业论文最终绝不改版1
- 毕业论文最最最最最最最终版
- 毕业论文最最最最最最最终版1

系统工程第一定则

“No Matter where you are in the system life cycle, the system will change, and the desire to change it will persist throughout the life cycle”



Version 7.2 by NPU (nonplux@gmail.com)
For the latest version, visit kde-files.org
Feel free to modify and spread. Mail me for updates, corrections and source files/xcf files
Based on "Línea del tiempo Distribuciones Linux" by A. Sandoval (microtecnologias.cl)
Additional info: distrowatch.com/wikipedia.org



我们面对的问题

- 01 | 多个人需要共同参与一个软件开发
- 02 | 一个软件可能有多个要支持的版本
- 03 | 一个软件可能会用多种不同的运行环境

术语定义：版本（Version）

- 最初发布或再发布的“代码及其附属品”的组合，它应该是可被完整编译或被认定为完整可用的。
- 不同的版本表现出不同的功能特性。

术语定义：基准（Baseline）

- 根据质量管理所需确定的阶段性规格说明，这个说明应是被正式评审认可的。
- 之后的开发过程都应该要遵循“基准”的要求进行。
- 对于基准的修改是要极为慎重的，要通过严格的变更流程。

例子：

Baseline A：所有接口应被完备的定义，各方法的内容为空。

Baseline B：所有的数据访问方法应该被实现并测试。

Baseline C：GUI 被完成

术语定义：基准（Baseline）

- 基准的命名有一个非常常用的三点命名法。

A.B.C
如 9.3.1

- A：从用户（消费者）角度看到的发布出的标记数（Release）
- B：从开发者角度看到的关键的版本（Version）号
- C：从开发者角度关注的修改版本（Revision）号

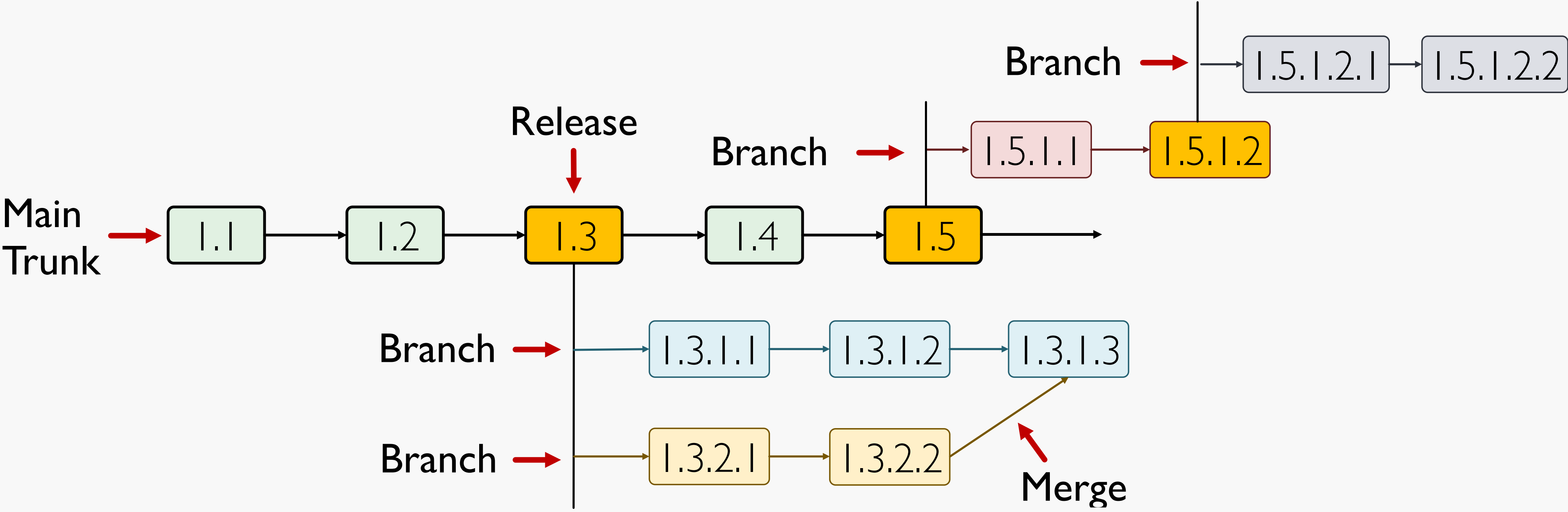
术语定义：发布（Release） / 版本（Version） / 修改（Revision）

- 版本（Version）：最初发布或再发布的“代码及其附属品”的组合，它应该是可被完整编译或被认定为完整可用的。不同的版本表现出不同的功能特性。
- 修改（Revision）：对于一个版本的修改，只做了代码设计的错误修正，对于已经“附属品”中文档所描述的功能特性没有任何改动。
- 发布（Release）：被批准的面向用户进行分发的版本。

版本管理系统

- 用于追踪和管理对于一系列文件和资源修改的软件系统
- 帮助团队在代码项目上进行协作
 - 所有成员可以访问的代码
 - 将当期版本呈现在文件系统中，将过往版本备份在不直接被看到的地方
 - 可以看见过去发生了什么
 - 当不同人修改了同一部分内容时可以解决冲突

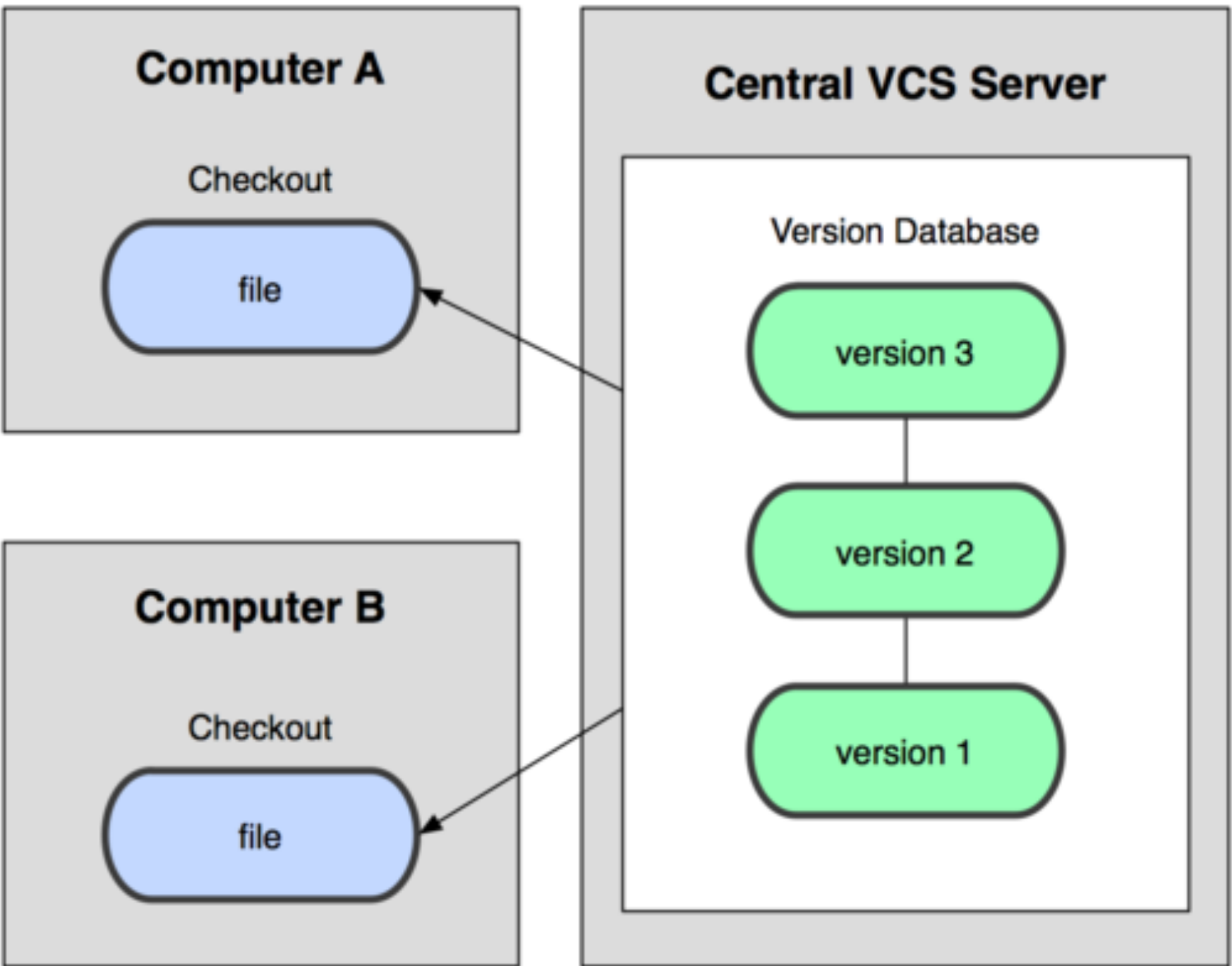
备注：也可以用于非代码的文本版本管理（如论文）



中心式 vs 分布式版本管理

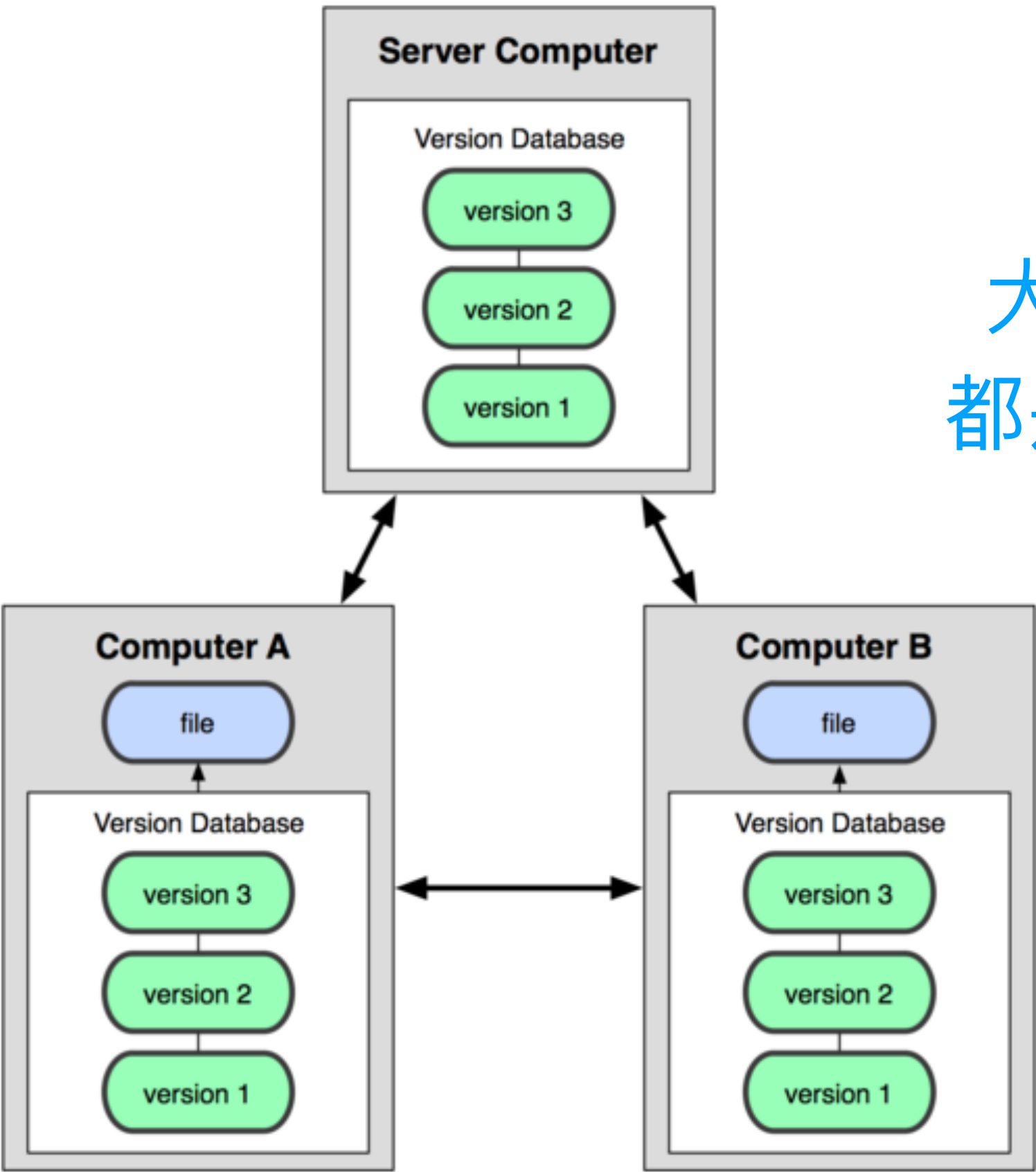
中心式模型

(CVS, Subversion, Perforce)



分布式模型

(Git, Mercurial)



大量的操作
都是本地化的

Git 的安装与基本功能

- 01 | Linux 社区中，为管理 Linux 操作系统内核迭代产生
- 02 | 2005 年 Linus Torvalds 作为初始创作者
- 03 | 为了替代闭源的 BitKeeper 方案而产生

接下来的内容前提

01 | 如果你是 Linux / MacOS 用户：我们假设你已经会使用 Terminal

02 | 如果你是 Windows 用户：我们假设你已经会使用 PowerShell

Git 的安装

01

Linux: 请参考 <http://git-scm.com/download/linux>

02

Mac OS: 请参考 <http://git-scm.com/download/mac>

03

Windows: 请参考 <http://git-scm.com/download/win>

初次使用 Git 前请配置

绿色的部分引号里应该替换成
你的名字 和 你的 email
别直接抄



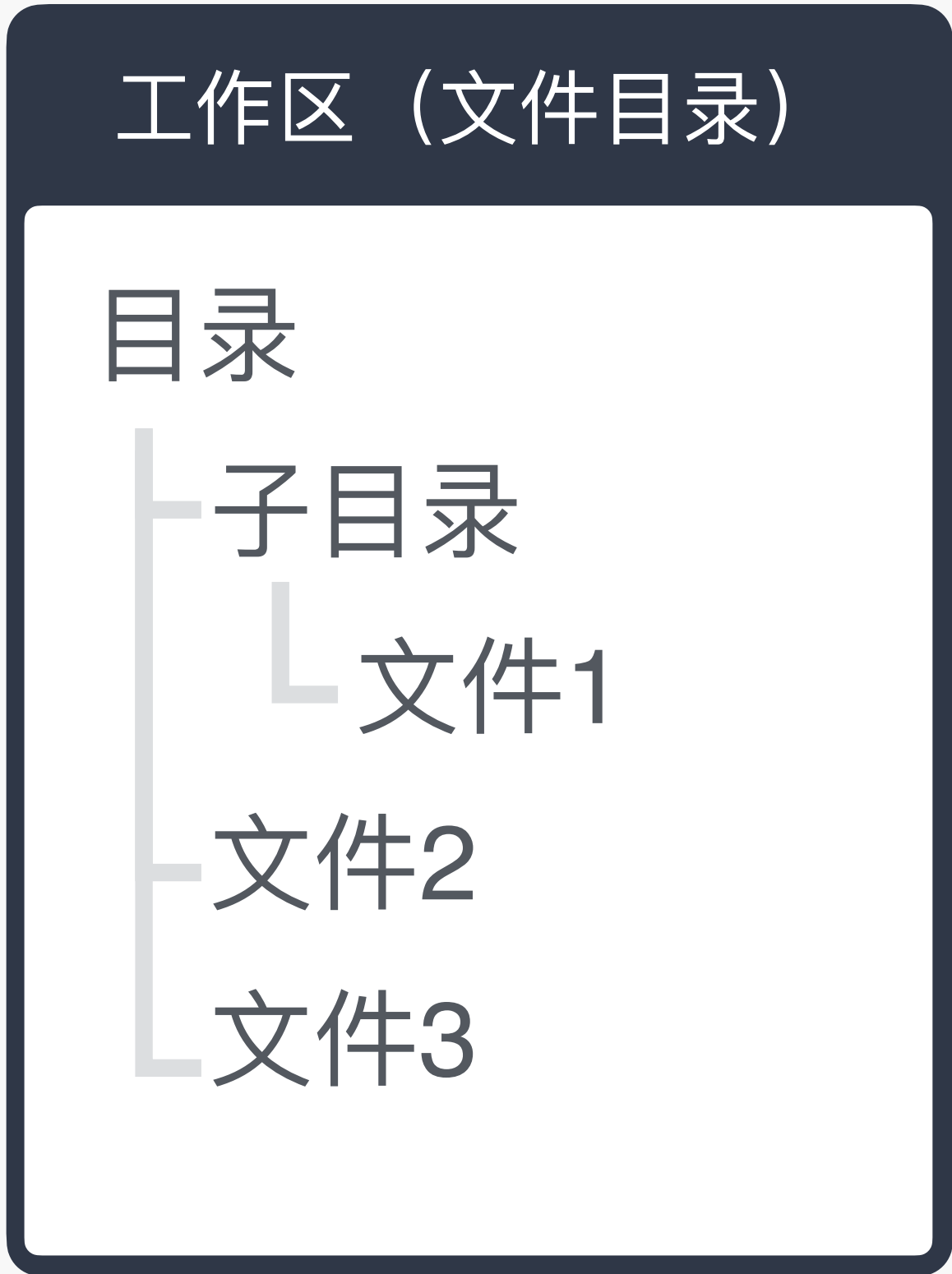
```
git config --global user.name "your name"  
git config --global user.email "your email"
```

告诉 git 以后这个计算机用户的 git 操作记录怎么署名

```
git config --global core.editor emacs
```

如果你希望更换 git 默认用的 vim 编辑器可以这么做

在文件目录创建版本库



版本库（.git隐藏目录）

在文件目录创建版本库



创建仓库

- 01 在当前目录下创建
`git init`
- 02 在指定的目录创建
`git init <指定的目录路径>`
- 03 创建一个 bare git 仓库
`git init --bare myrepo.git`

获取已经存在的版本库



通过克隆获取远端版本库

01

克隆本地仓库

```
git clone /path/to/repo.git
```

02

克隆远端仓库

```
git clone https://se.jisuanke.com/course/python37.git
```

03

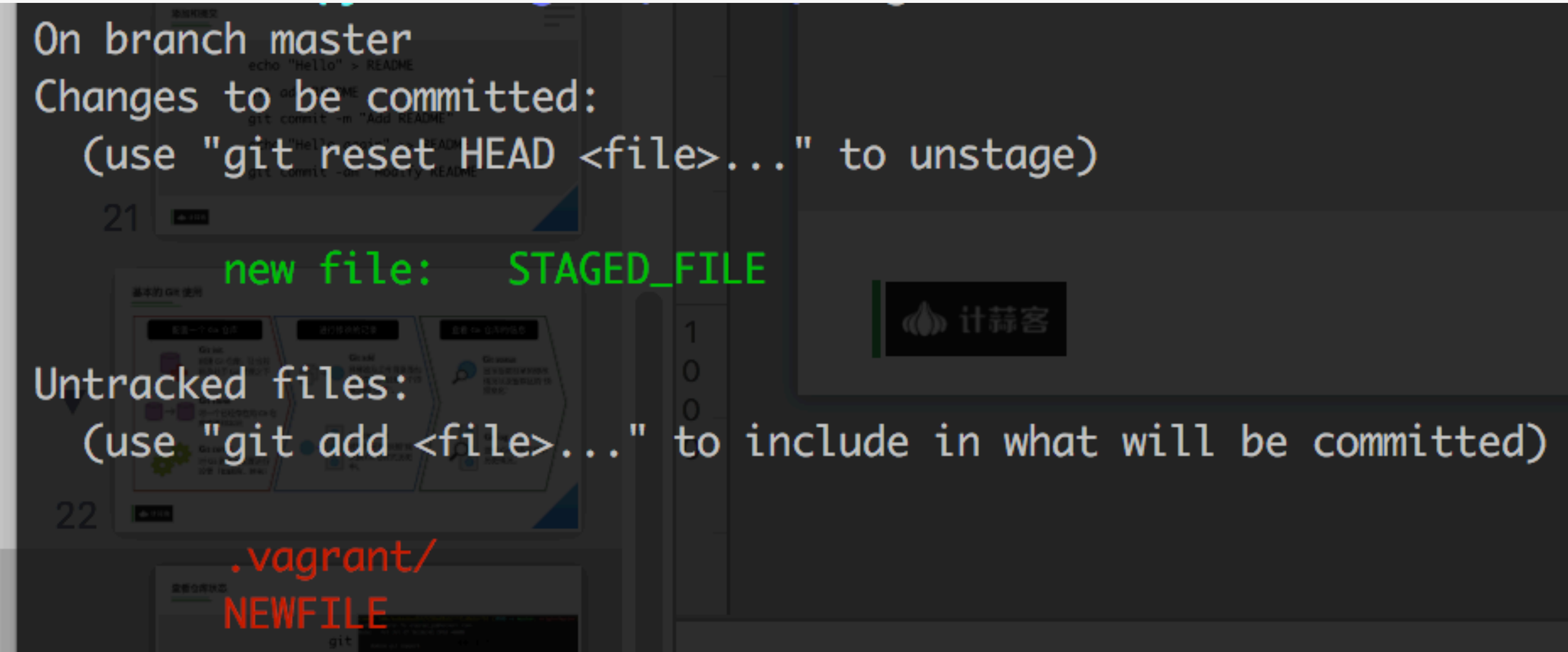
通过 ssh 协议完成克隆

```
git clone git@se.jisuanke.com:course/python37.git
```


本地 Git 版本库的三个分区



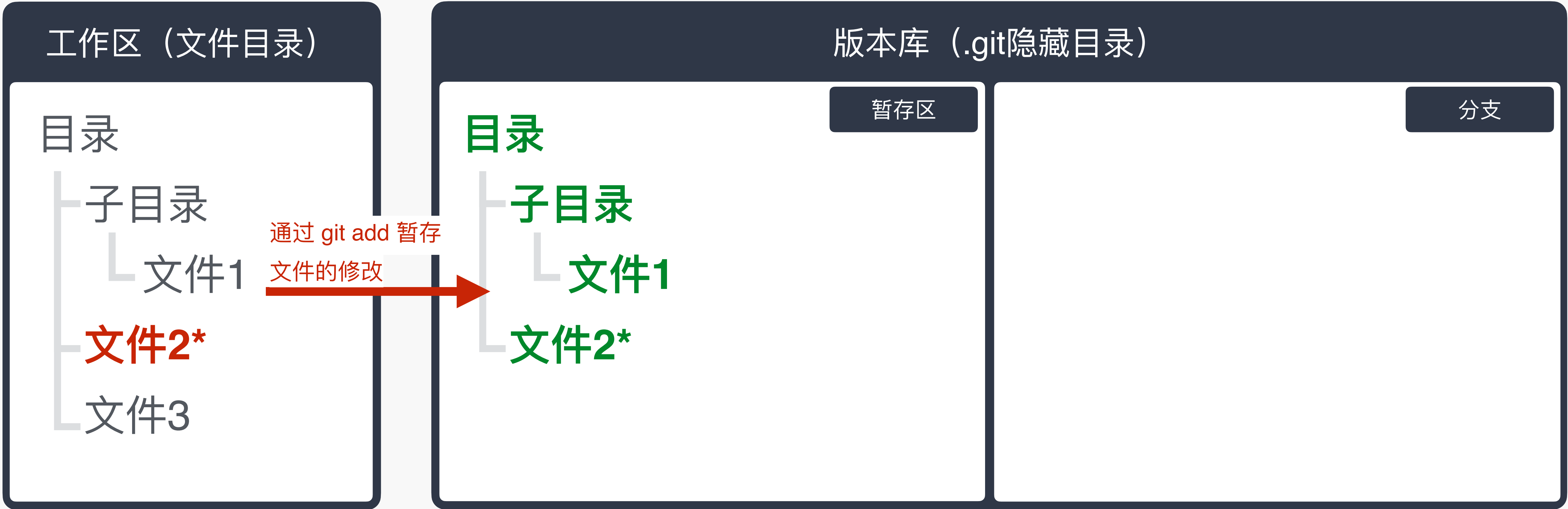
git status



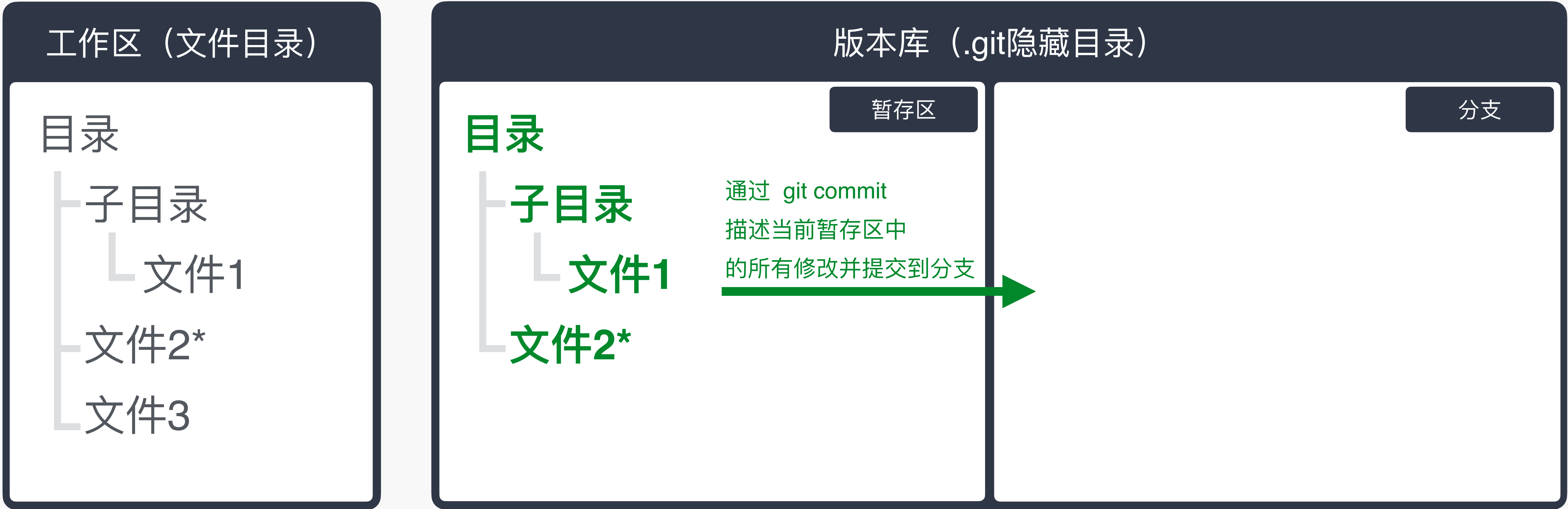
Git 中文件在工作区与版本库间的关系



Git 中文件在工作区与版本库间的关系



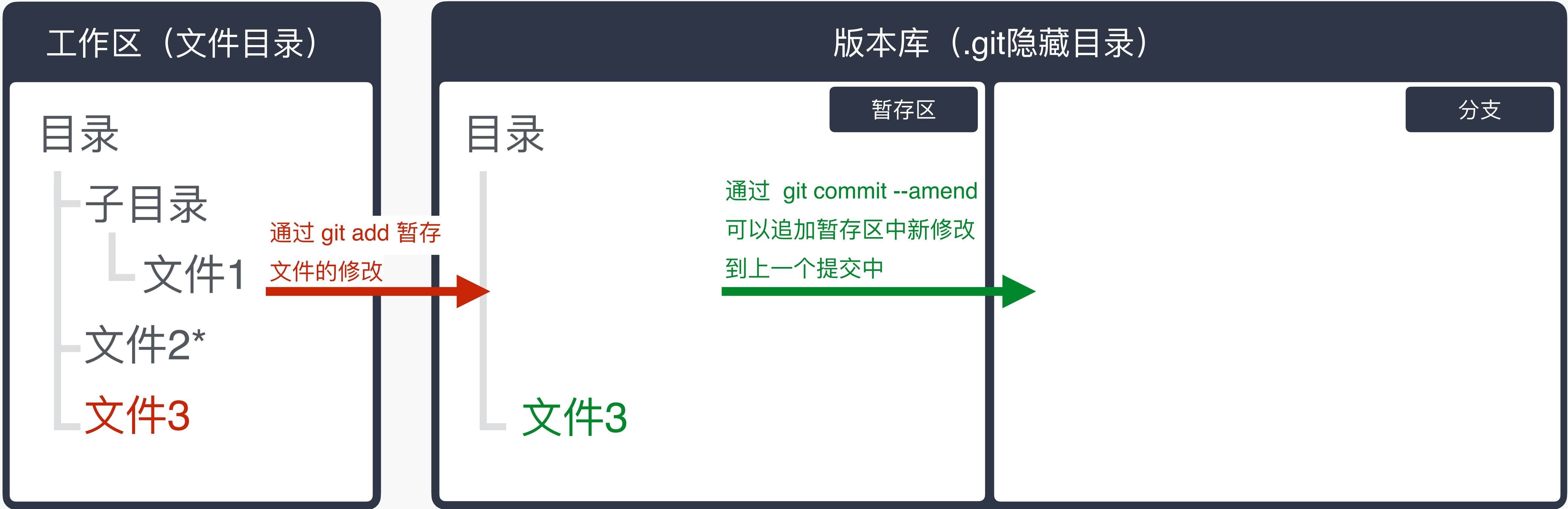
Git 中文件在工作区与版本库间的关系



Git 中文件在工作区与版本库间的关系



Git 中文件在工作区与版本库间的关系



git log

```
commit 180a76d8de5ba82d24780a08ab2f731d0e6ef7bf (HEAD -> master, origin/master)
Author: Haoran Yu <haoran_yu@hotmail.com>
Date:   Fri Jul 27 16:56:42 2018 +0800

    Update git support

commit 40f83f599e801b1d5d298ce499c0712cdb086891
Author: Haoran Yu <haoran_yu@hotmail.com>
Date:   Fri Jul 27 13:48:32 2018 +0800

    Update Readme to guide students better

commit bb47cc3d0cdb82b276228e6dfc928c70c14e2c52
Author: Haoran Yu <haoran_yu@hotmail.com>
Date:   Fri Jul 27 13:34:53 2018 +0800

    Roll back to fedora 27 and support ipython, bpython and pip18

commit b29a7c86397728732b765ca33ddaeba082353d3b
Author: Haoran Yu <haoran_yu@hotmail.com>
Date:   Fri Jul 27 10:43:14 2018 +0800

    Initialize fedora28-python37 environment
```

查看版本库提交历史（最近3次详情）

git log -3 -p

```
commit 180a76d8de5ba82d24780a08ab2f731d0e6ef7bf (HEAD -> master, origin/master)
Author: Haoran Yu <haoran_yu@hotmail.com>
Date:   Fri Jul 27 16:56:42 2018 +0800

    Update git support

diff --git a/README.md b/README.md
index f921a65..be67e04 100755
--- a/README.md
+++ b/README.md
@@ -119,4 +119,15 @@ Type in `pip --version` or `pip3 --version` you should get
    pip 18.0 from /home/vagrant/.local/lib/python3.7/site-packages/pip (python 3.7)
    ...

+#### Extra preparation
+
+Type in `cd ~` and then
+
+```
+git clone https://github.com/django/django.git
+
+
+We would use it later during our lectures.
+
+
+*If any test above cannot pass on your machine, contact Haoran Yu (俞昊然) on IM or through Email
+yuhaoran@jisuanke.com as soon as possible.*
diff --git a/Vagrantfile b/Vagrantfile
index b664b1c..0318673 100755
--- a/Vagrantfile
+++ b/Vagrantfile
@@ -15,10 +15,10 @@ Vagrant.configure(2) do |config|
    end

    ...skipping...
commit 180a76d8de5ba82d24780a08ab2f731d0e6ef7bf (HEAD -> master, origin/master)
Author: Haoran Yu <haoran_yu@hotmail.com>
Date:   Fri Jul 27 16:56:42 2018 +0800

    Update git support

diff --git a/README.md b/README.md
index f921a65..be67e04 100755
```


查看版本库提交历史（逐行看）

git log --oneline --graph

```
| * | 3b5b6cad9 Add send unresolved question feed. #5508
| * | a20bc0dea Merge branch 'nie/5500-admin-backstage-add-ad' into 'master'
| \ \ \
| * | 01ea1a6e5 Show ad list.#5500
| * | 837df329e Merge branch 'max/5509-course-submissions' into 'master'
| \ \ \
| | / / /
| / | |
| * | d2de41f04 Modify table. #5509
| * | c44fa0aa1 Remove unused code. #5509
| * | 32682342a Fix score display. #5509
| * | e8f18a8c7 Add other course submission manage. #5509
| * | b3111915a Merge branch 'nie/5500-admin-backstage-add-ad' into 'master'
| \ \ \
| | _ _ | /
| / | |
| * | 31c0abfdc Fix bug. #5500
| \ / /
| * | 96b66756b Merge branch 'nie/5500-admin-backstage-add-ad' into 'master'
| \ \ \
| | _ _ | /
| / | |
| * | d49762f85 Fix bug. #5500
| \ / /
| * | bf1205ffe Merge branch 'xinmian/5498-course-promotions' into 'master'
| \ \ \
| | _ _ | /
| / | |
| * | 9a94adb6c Resolve conflicts. #5498
| \ \ \
| | / /
| / | |
| * | 44701e296 Merge branch 'nie/5500-admin-backstage-add-ad' into 'master'
| \ \ \
| | _ _ | /
| / | |
| * | 36040d39c Update channel banner. #5500
| * | 49ce5e2de Update channer_banner. #5500
| * | 795a05821 Update channer_banner. #5500
| * | 19c4e80f5 Ad backstage. #5500
| * | e92c566a1 Merge branch 'master' of git.jisuan.ren:Jisuanke/Jisuanke2 into nie/5500-admin-backstage-add-ad
| \ \ \
| | / /
| / | |
```

最近三次添加了啥、删除了啥

Git Pretty Log

```
sergio@soviet-russia < b1.2.4 > : ~/projects/external/rubinius
% git log
commit 107632cd03c00f48c6f90dbbaca8fdaf4ee3a6b6
Author: Evan Phoenix <evan@fallingsnow.net>
Date: Tue Jul 5 18:33:08 2011 -0700

    Update website for 1.2.4

commit 88d9f687c1672662acad9b3ec04cef3cd73b30f5
Author: Evan Phoenix <evan@fallingsnow.net>
Date: Tue Jul 5 17:51:31 2011 -0700

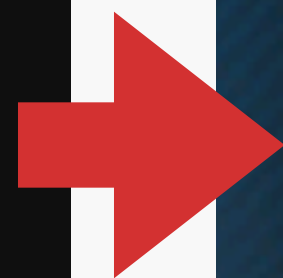
    Bump version number

commit b7df3fd98aabbfce9a33cc5cda3f933d2c9806e0
Author: Shane Becker <veganstraightedge@gmail.com>
Date: Tue Jul 5 18:16:13 2011 -0700

    regenned site for new blog post about status board

commit f76e532b185720b8eba663e325d94f331531918b
Author: Shane Becker <veganstraightedge@gmail.com>
Date: Tue Jul 5 18:13:37 2011 -0700

    new blog post: rubinius status board
```



```
sergio@soviet-russia < b1.2.4 > : ~/projects/external/rubinius
% git log --pretty=format:'%Cred%H%Creset -%C(yellow)%d%Creset %s %Cgreen(%cr)%Creset' --abbrev=40
107632c - (HEAD, release-1.2.4, b1.2.4) Update website for 1.2.4 (1 year, 4 months ago)
88d9f68 - Bump version number (1 year, 4 months ago)
b7df3fd - regenned site for new blog post about status board (1 year, 4 months ago)
f76e532 - new blog post: rubinius status board (1 year, 4 months ago)
42f7c72 - added capitalize to String case benchmarks (1 year, 4 months ago)
bddf636 - yet another way of removing the first elements from an array (1 year, 4 months ago)
6e4ed98 - new bench for Array#slice (1 year, 4 months ago)
049bace - Remove tags for now passing specs (1 year, 4 months ago)
44c3886 - Socket needs it's own shutdown (1 year, 4 months ago)
8374734 - regenned site for new blog post (map pins) (1 year, 4 months ago)
f90da99 - new blog post: rubinius around the world map and pins of shirts/tshirts (1 year, 4 months ago)
cf13e6b - Add a few more errno's based on OS X and Linux (1 year, 4 months ago)
0b8b477 - Add a bunch of errno's from FreeBSD (1 year, 4 months ago)
4b34345 - Load correct digest file, fixes broken Rubygems (1 year, 4 months ago)
e2be2d5 - Remove unused rubinius::guards (1 year, 4 months ago)
23e97d5 - Remove used flag and file it was defined in (1 year, 4 months ago)
cff4ee2 - Remove unused CallFrameList and some maps (1 year, 4 months ago)
dd8f2b1 - Removed unused async message and mailbox code (1 year, 4 months ago)
c4b54ba - Remove unused code (1 year, 4 months ago)
744e9f0 - Fix tiny typo's (1 year, 4 months ago)
912d530 - Cleanup last remnants of dynamic interpreter (1 year, 4 months ago)
6b29b21 - Remove unused IndirectLiterals (1 year, 4 months ago)
83db68a - Fixed Digest requires in const missing. (1 year, 4 months ago)
```


思考一下：操作后的 git log 是什么样的呢？

```
echo "Hello" > README
```

```
git add README
```

```
git commit -m "Add README"
```

```
echo "Hello again" >> README
```

```
git commit -am "Modify README"
```

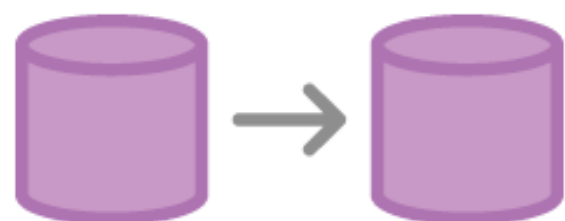
基本的 Git 版本管理使用回顾

配置一个 Git 版本库



Git init

创建 Git 版本库，让当前目录处于 Git 管理之下



Git clone

将一个已经存在的 Git 版本库克隆到本地



Git config

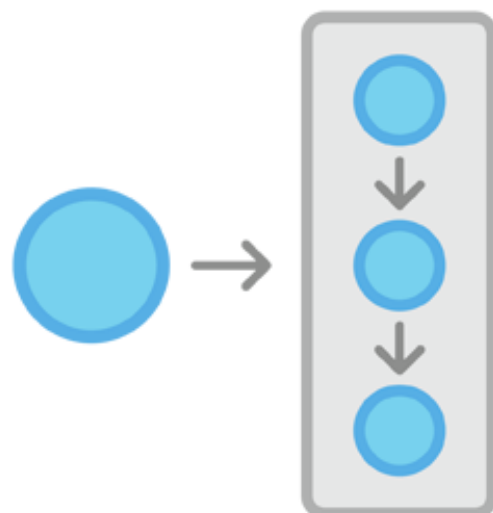
对 Git 的基本配置进行设置（如邮箱、姓名）

进行修改的记录



Git add

将修改从工作目录添加到暂存区。创建一个待提交的“快照”。



Git commit

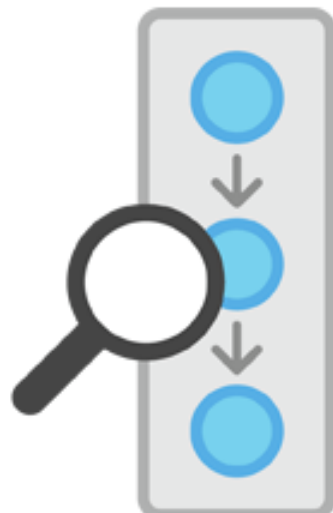
将暂存区的“快照”提交到 Git 版本库的历史中。

查看 Git 版本库的信息



Git status

显示当前目录的修改情况以及暂存区的“快照变化”



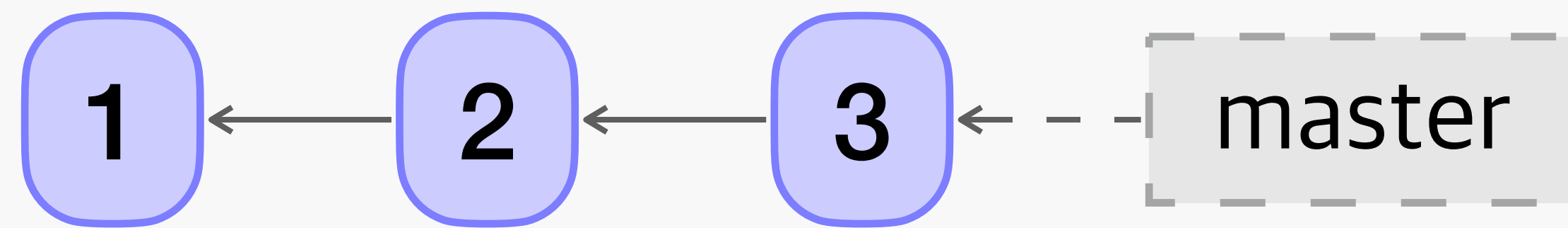
Git log

查看 Git 的各个版本历史情况。

- <https://try.github.io/>
- <https://git-scm.com/book/en/v2>
- <https://speakerdeck.com/mattnketmo/understanding-git>

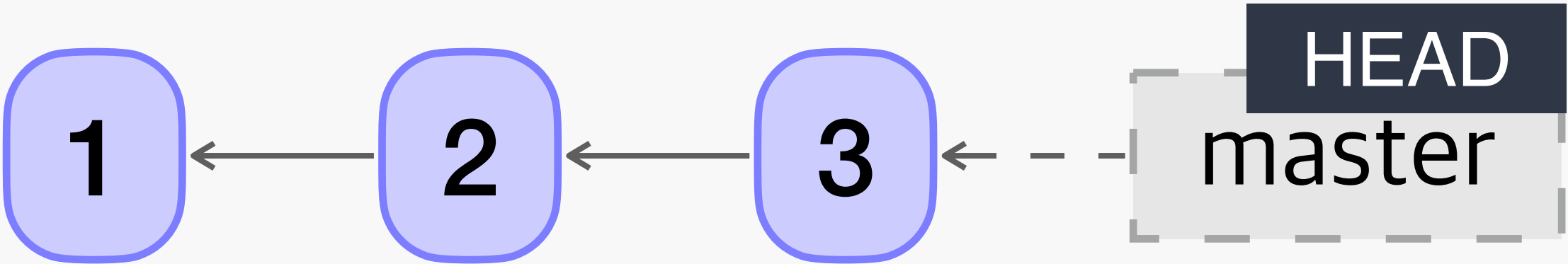
使用 Git 分支与版本切换

提交后的版本形成一条版本链



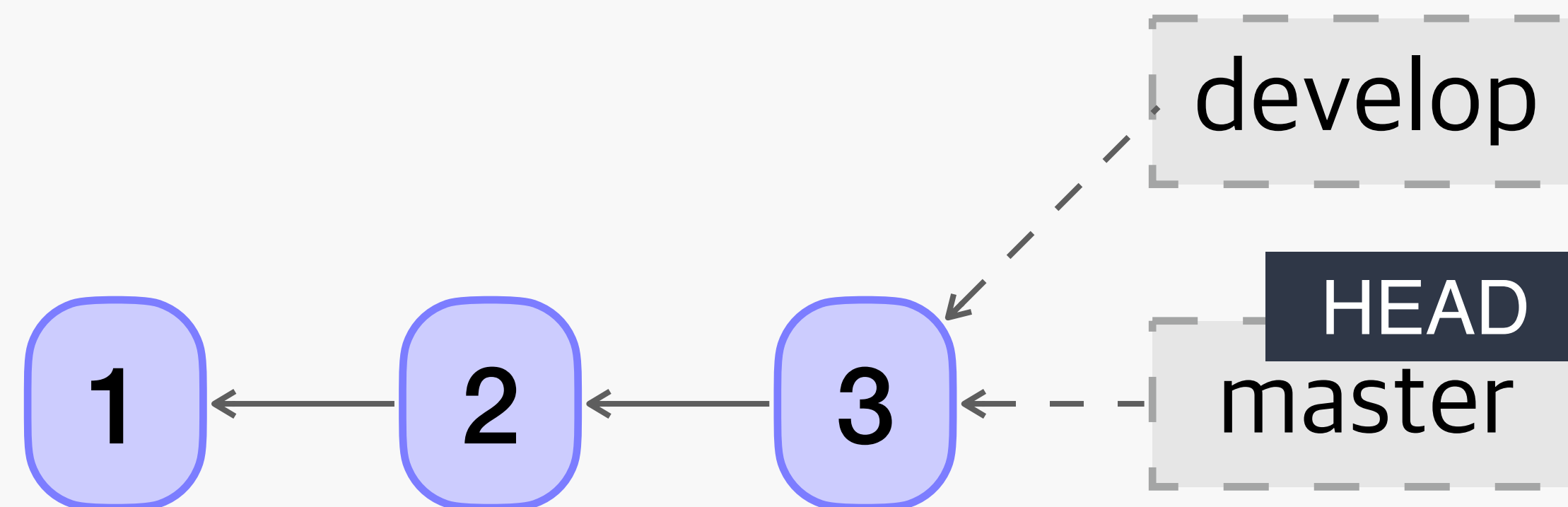
这条版本链位于默认分支 master 上

提交后的版本形成一条版本链



HEAD 表示当前呈现出来的版本

我们可以创建一个新的分支

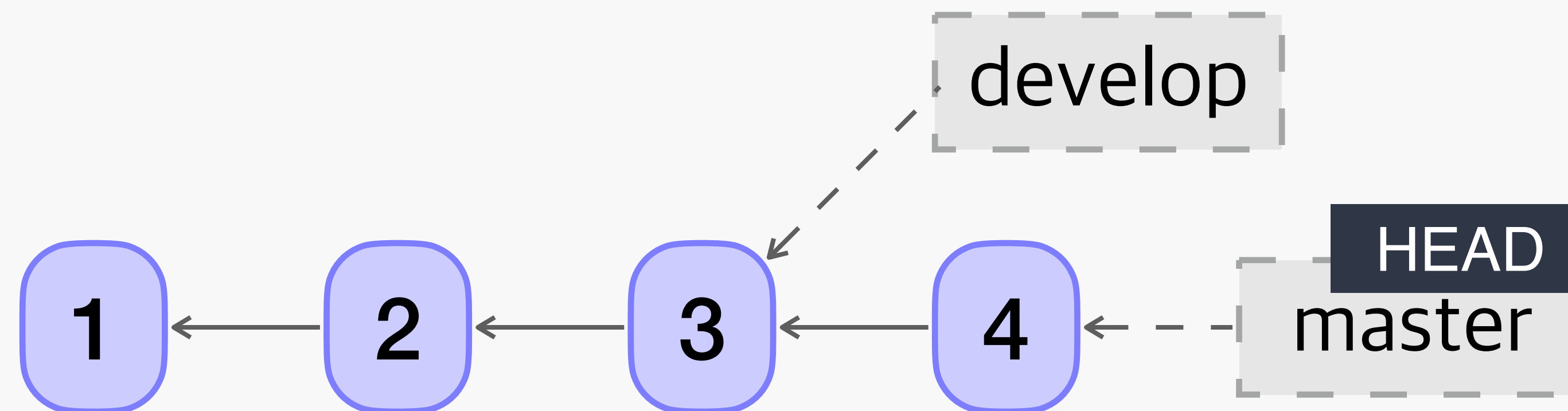


通过 `git branch develop` 创建 develop 分支后

如果没有切换到分支

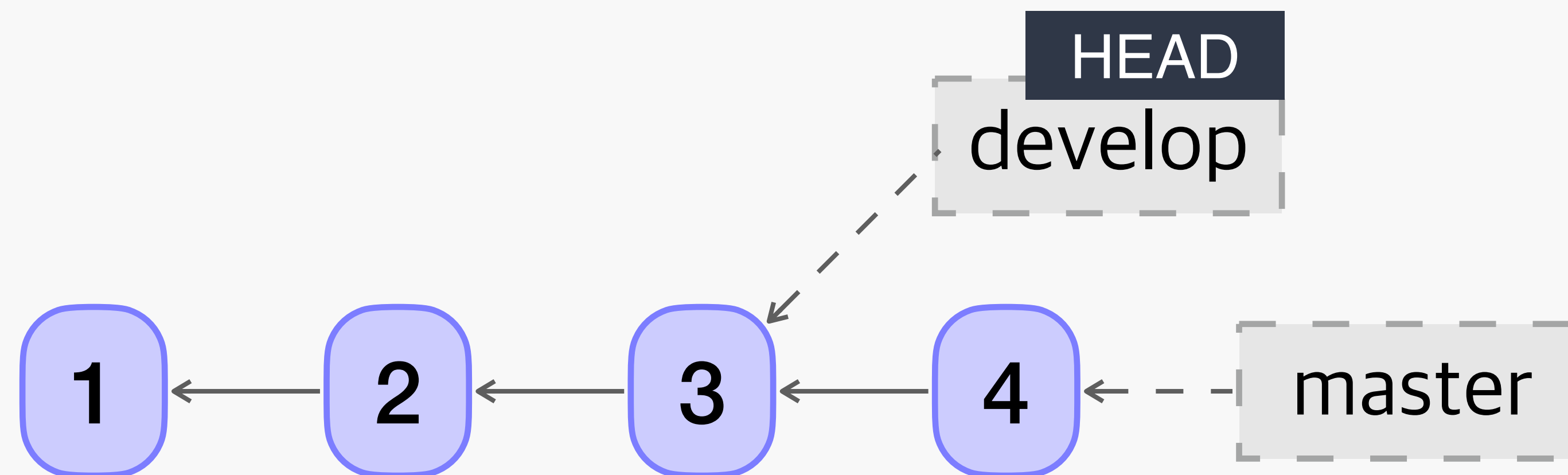
则还会继续在 HEAD 上产生新的版本

在 master 分支继续产生版本



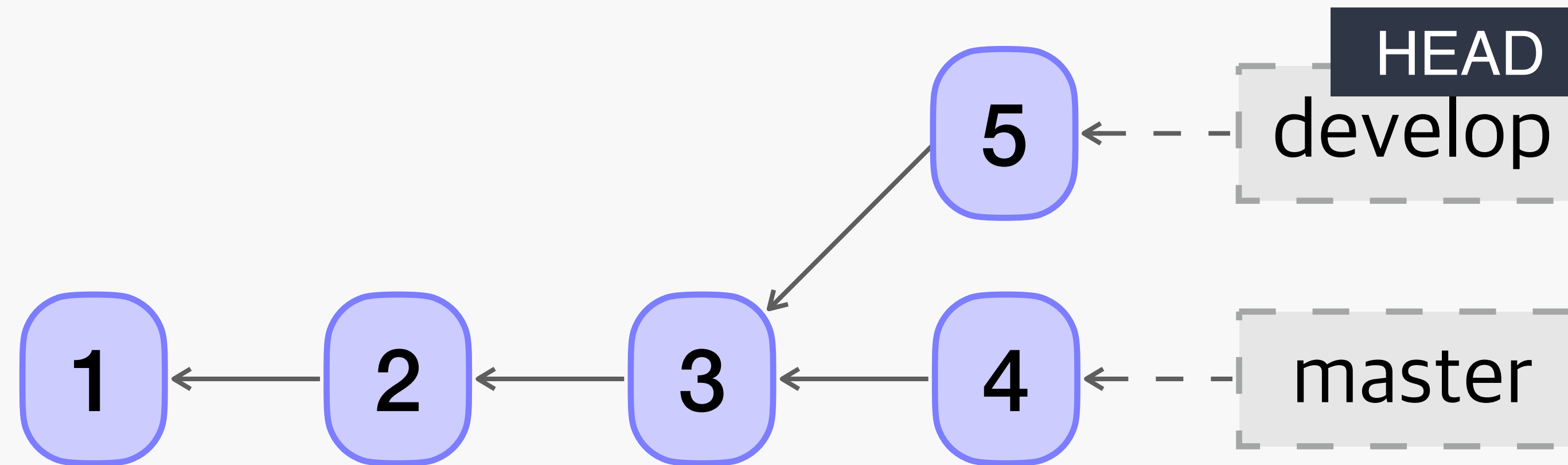
创建分支后，如果没有切换到分支
则还会继续在 HEAD 上产生新的版本

切换到分支



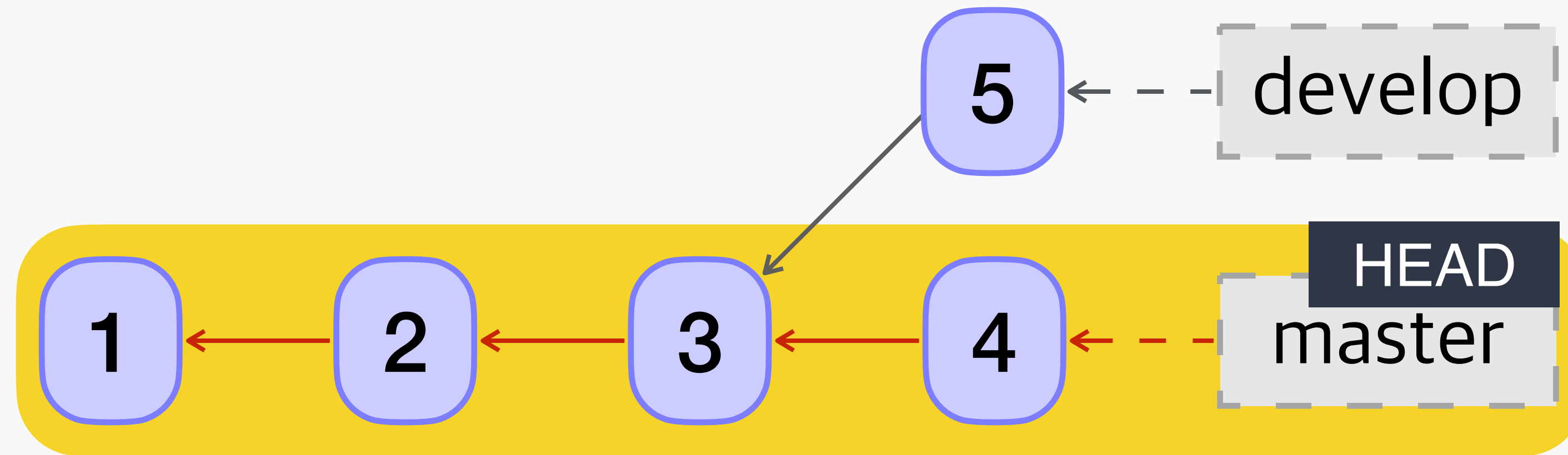
我们可以通过 `git checkout develop` 切换到 develop 分支
(此处 develop 是分支名, 我们希望切换到哪个分支就应该写哪个分支名)

在 develop 分支产生版本



接下来如果产生新的提交将在
HEAD 所切换到的 develop 分支衍生

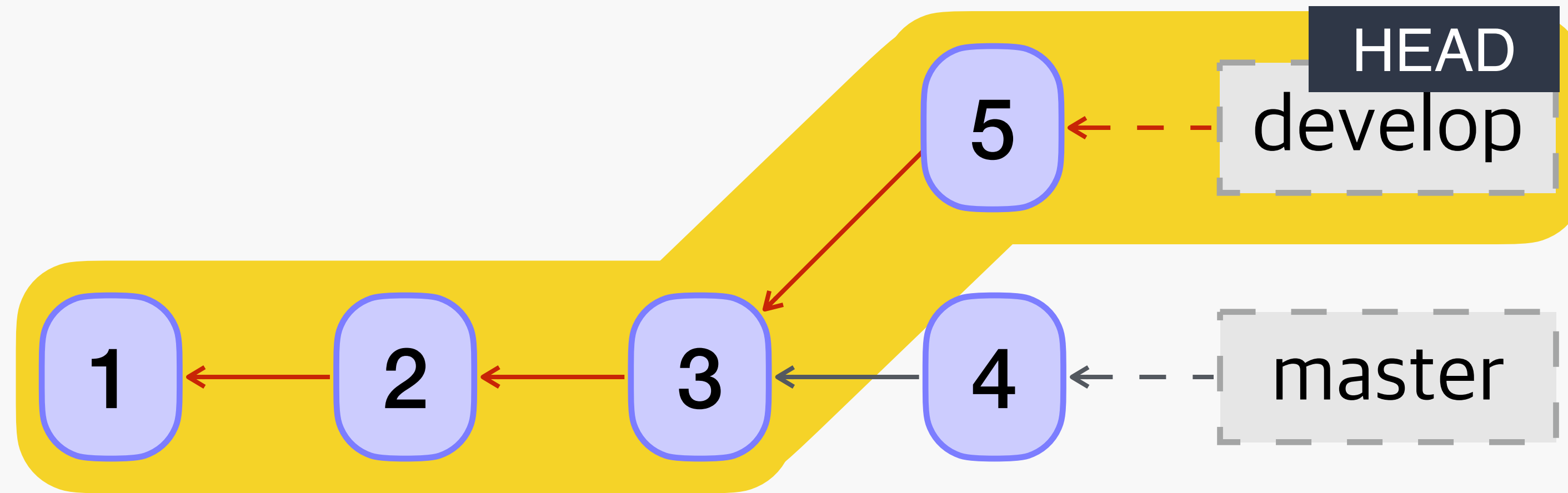
master 分支的版本链



在 master 分支视角的版本链

可以在 `git checkout master` 后通过 `git log` 查看

develop 分支的版本链条



在 `develop` 分支视角的版本链
可以在 `git checkout develop` 后通过 `git log` 查看

分支使用

01

查看现有分支

```
git branch
```

02

创建分支

```
git branch <分支名>
```

03

切换分支

```
git checkout <分支名> / git switch <分支名>
```

04

创建并切换到分支

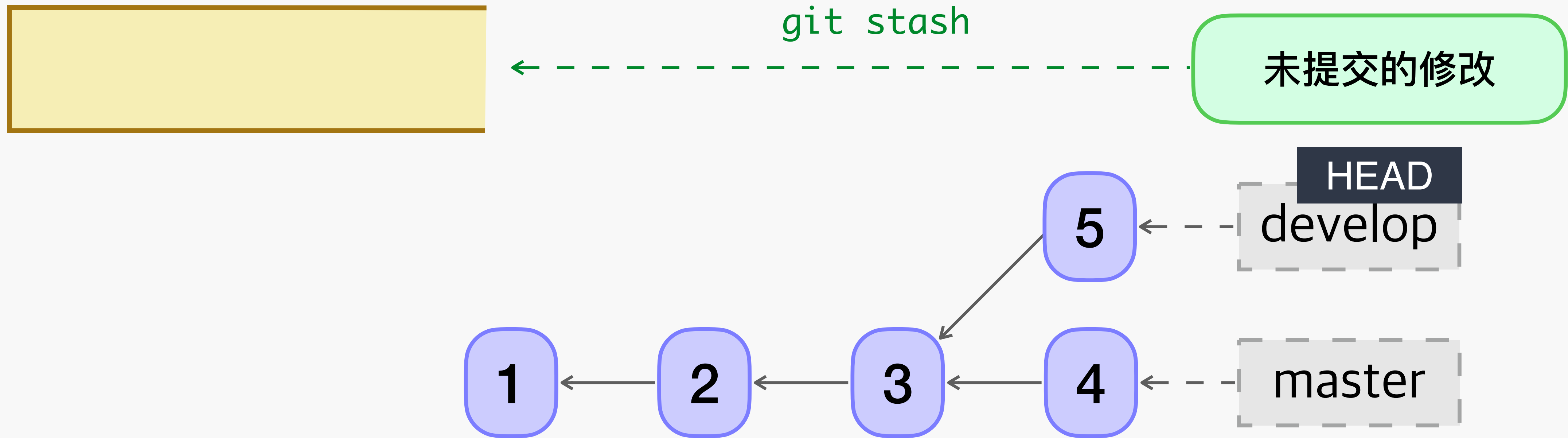
```
git checkout -b <分支名> / git switch -c <分支名>
```

05

删除分支

```
git branch -d <分支名>
```

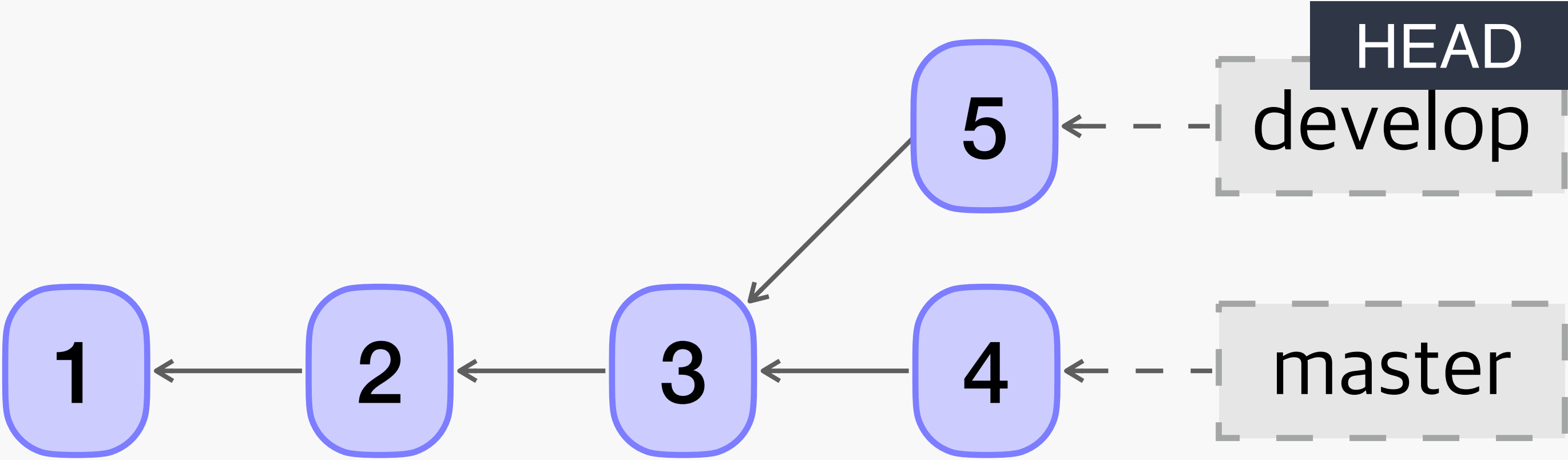
正在写新功能的时候，要插入其他工作怎么办？



这时候我们在 develop 分支开发新功能，有一系列未提交修改
我们可以使用 git stash 保存当前的现场

正在写新功能的时候，要插入其他工作怎么办？

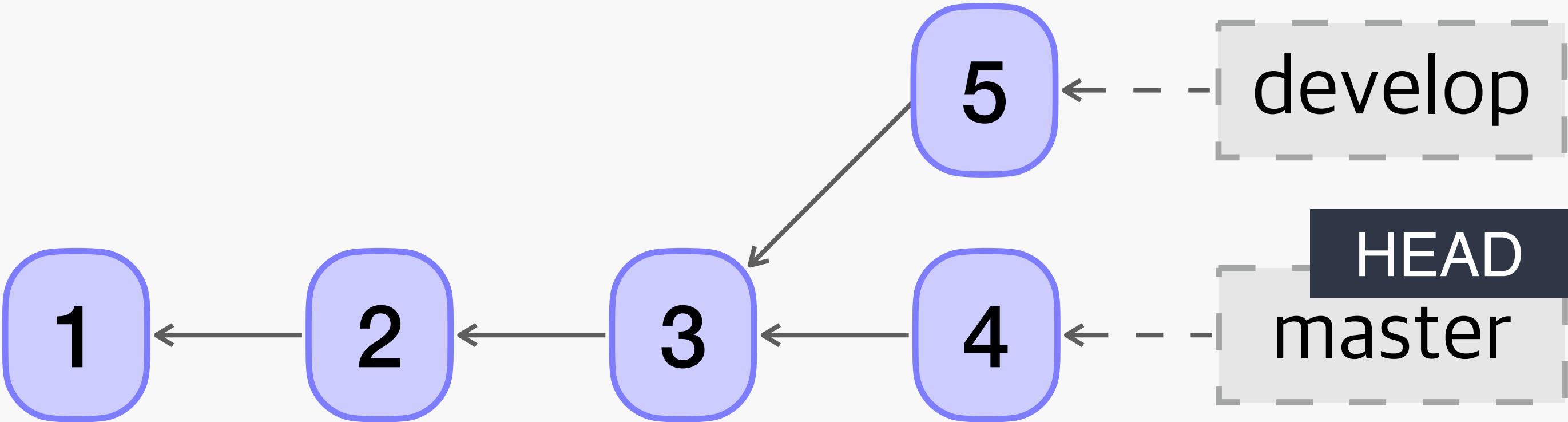
未提交的修改



这时 develop 分支被追踪的文件就像什么都没发生过一样
如果我们要修一个 master 分支上的 bug
我们可以 `git checkout master` 先回到 master 分支

正在写新功能的时候，要插入其他工作怎么办？

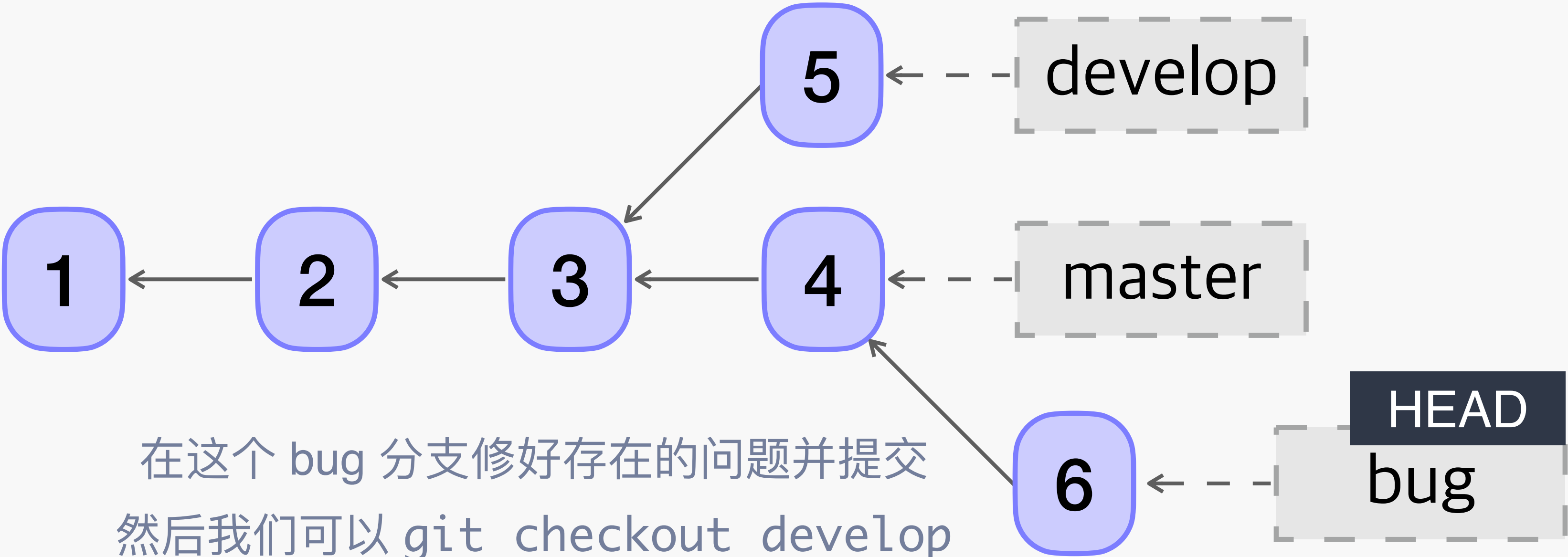
未提交的修改



然后新开并且一个新的修 bug 的 bug 分支 `git checkout -b bug`
在这个 bug 分支修好 master 分支存在的问题并提交

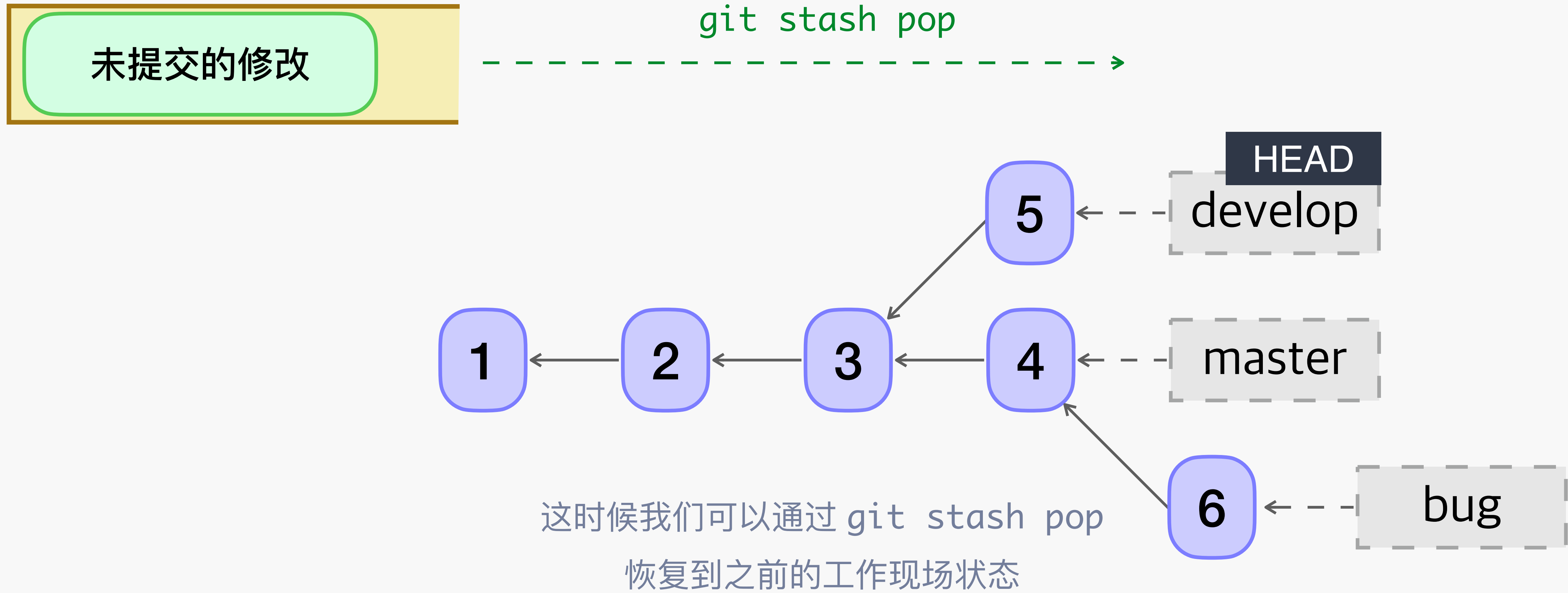
正在写新功能的时候，要插入其他工作怎么办？

未提交的修改

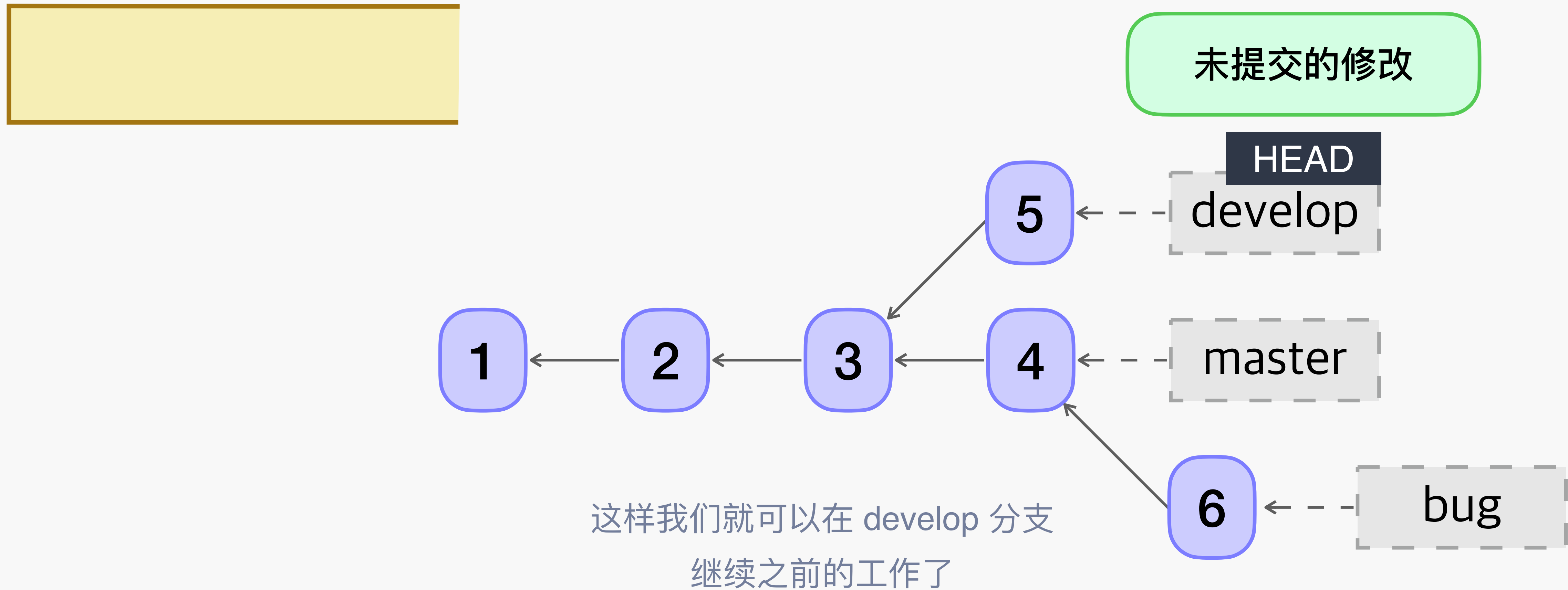


在这个 bug 分支修好存在的问题并提交
然后我们可以 `git checkout develop`
回 develop 分支继续此前的工作

正在写新功能的时候，要插入其他工作怎么办？



正在写新功能的时候，要插入其他工作怎么办？



储藏现场使用

01

储藏当前工作目录现场

```
git stash
```

02

恢复到最近储藏的现场

```
git stash pop
```

03

查看所有储藏的现场

```
git stash list
```

04

恢复到指定的储藏现场

```
git stash apply stash@{0}
```

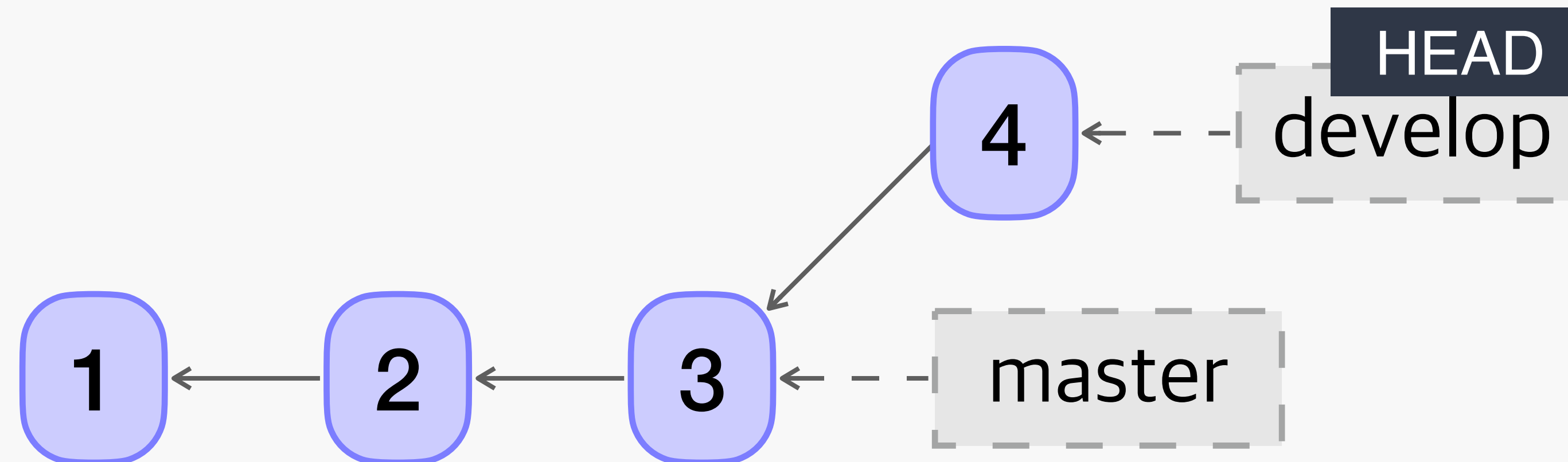
05

删除储藏位置中的指定现场

```
git stash drop stash@{0}
```

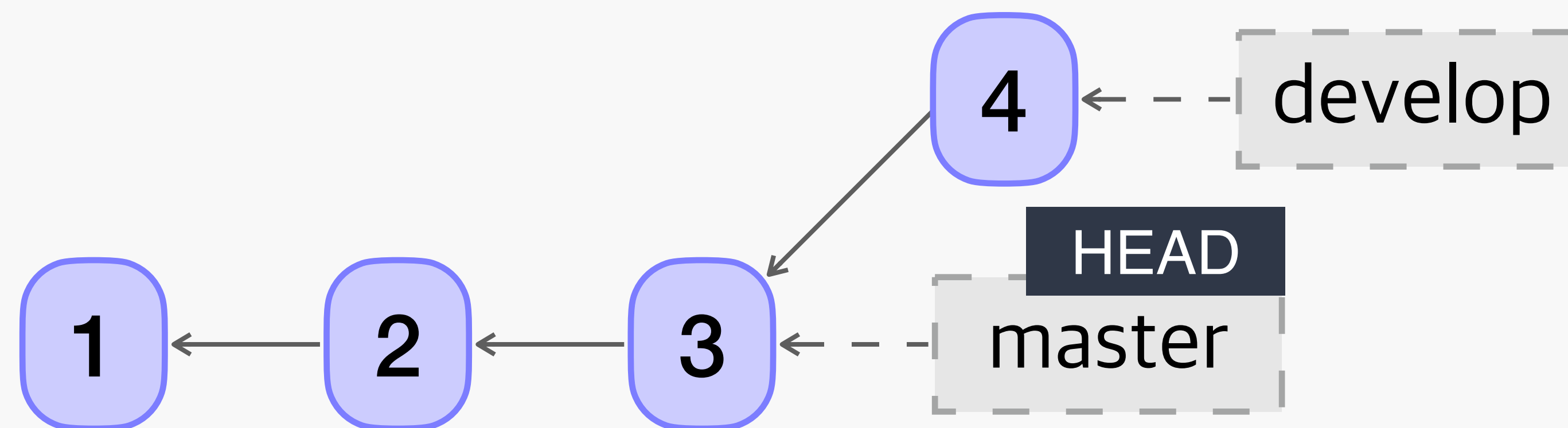
Git 协作之 Merge 策略

将 develop 分支合并到 master 分支



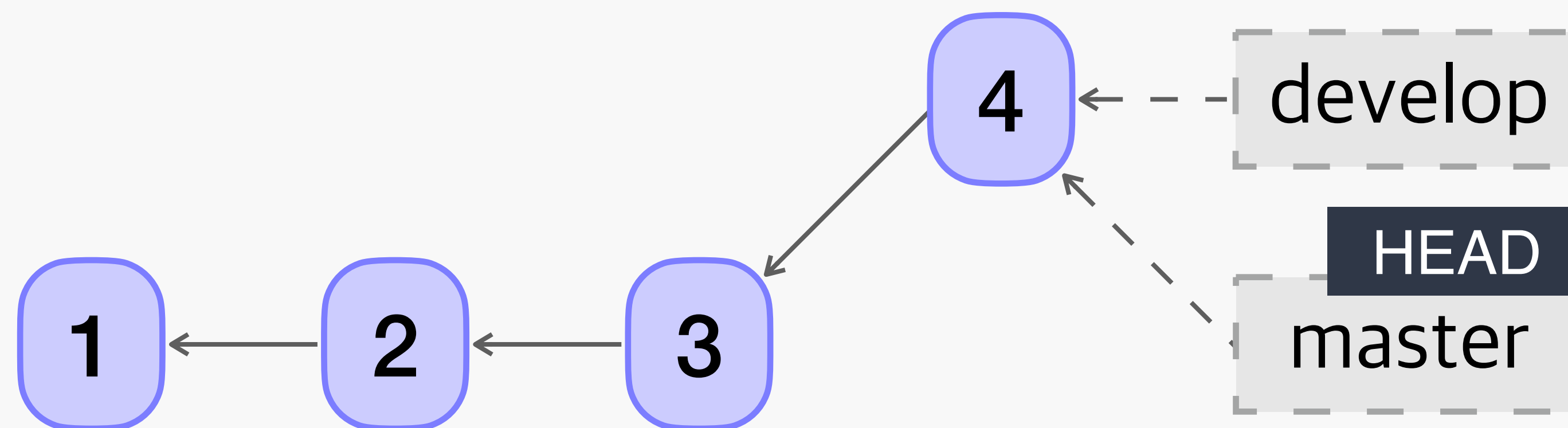
对于 develop 分支领先 master 分支的情况
先要让我们从 develop 分支 checkout 回到 master 分支上

将 develop 分支合并到 master 分支



这时候通过 `git merge develop`
可以将 develop 分支合并到 master 分支了

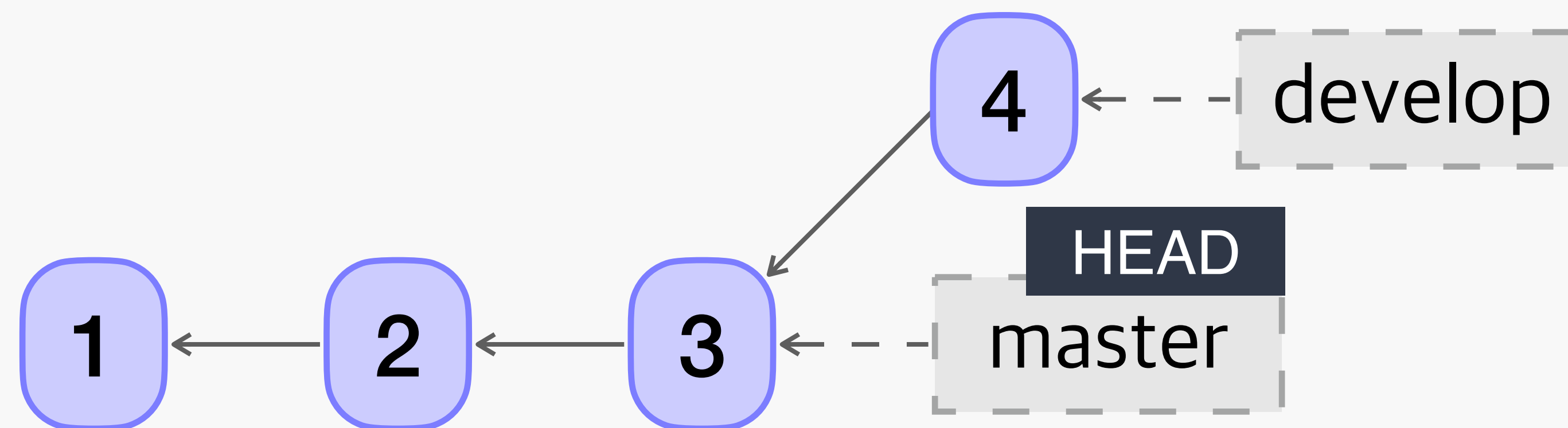
将 develop 分支合并到 master 分支



这种待合并分支领先目标分支的合并被我们称为 fast-forward 合并

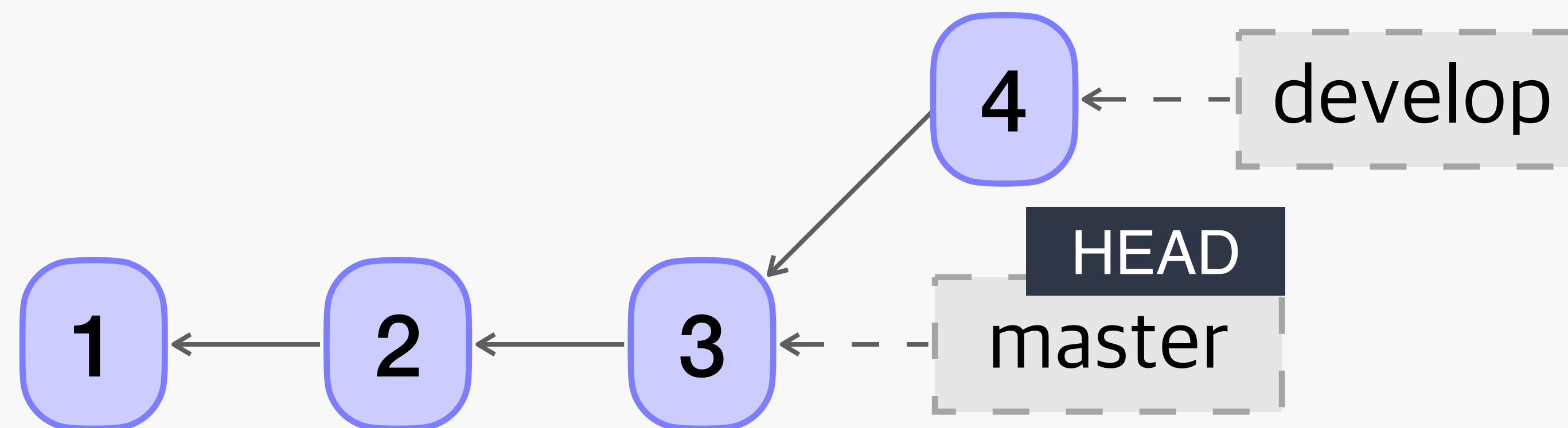
这时候 `git merge` 相当于 `git merge --ff-only`

将 develop 分支合并到 master 分支



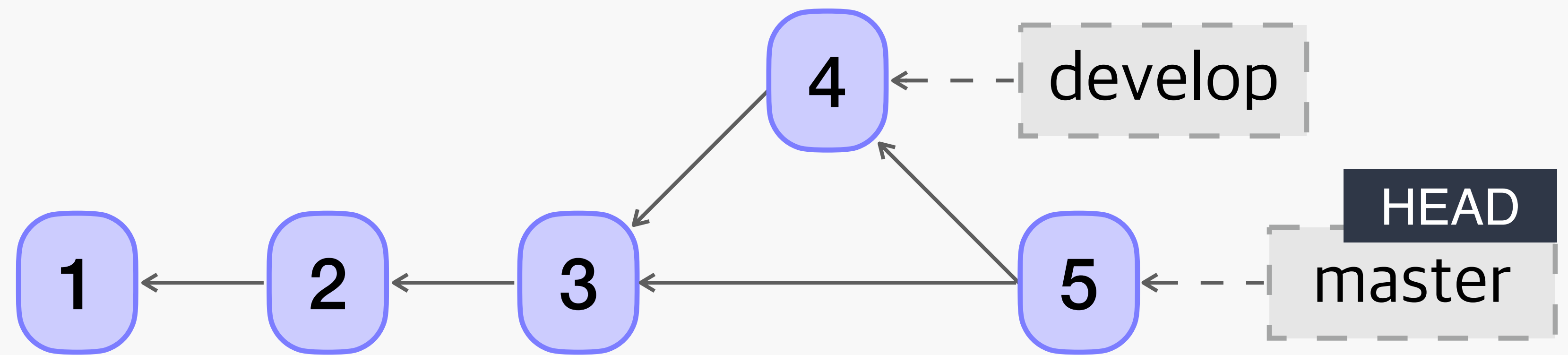
我们对于同样的情况也可以尝试另一种 non-fast-forward 合并

将 develop 分支合并到 master 分支



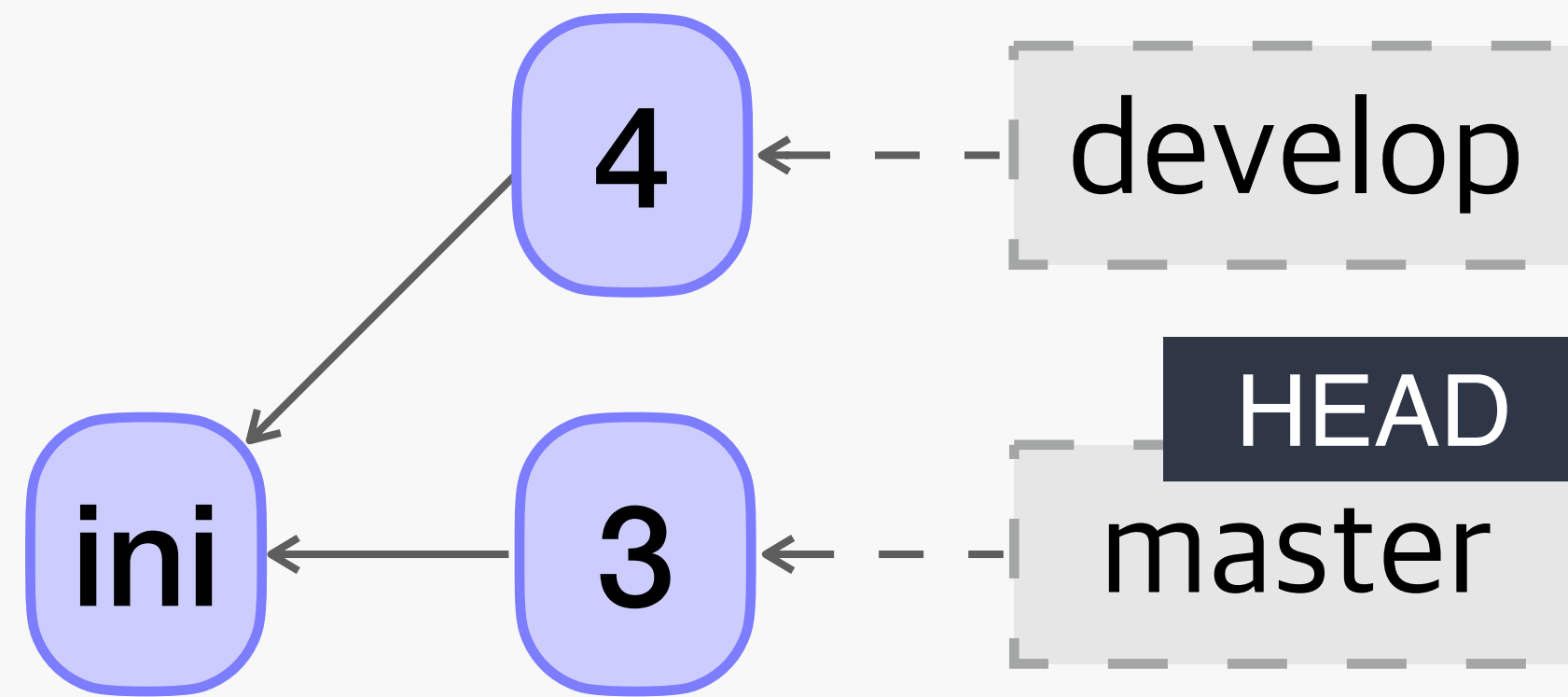
我们对于同样的情况也可以尝试另一种 non-fast-forward 合并
这时候通过 `git merge --no-ff develop`

将 develop 分支合并到 master 分支



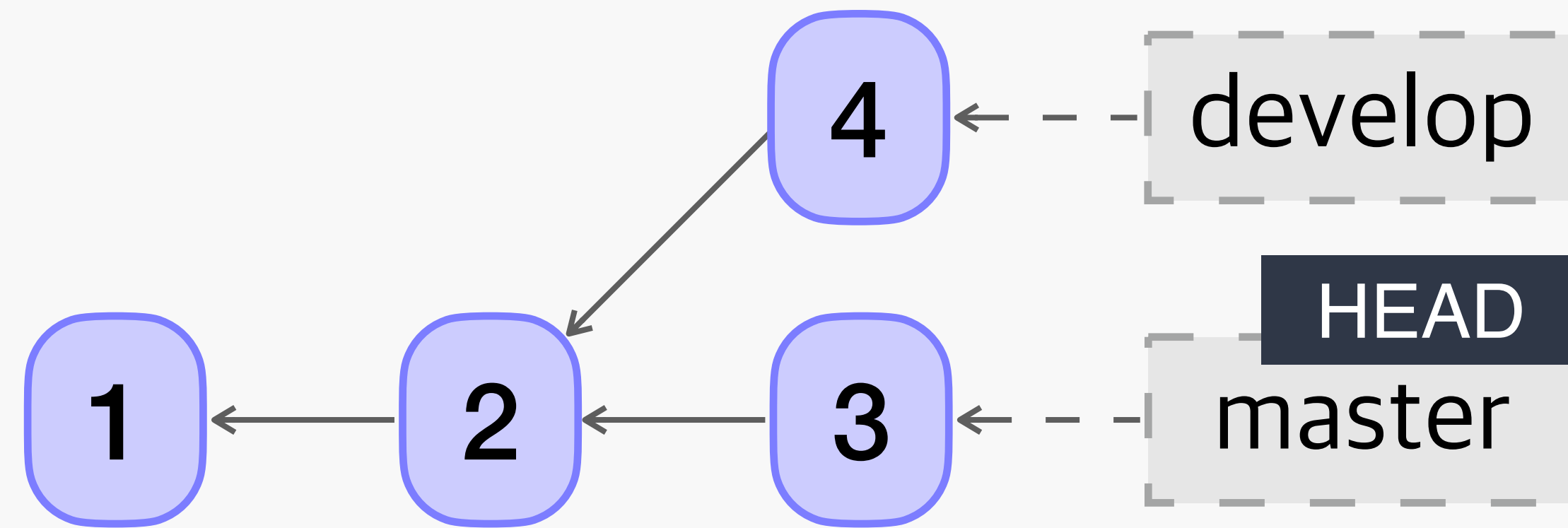
non-fast-forward 合并会产生一个新的提交

是否可以 fast-forward 呢?



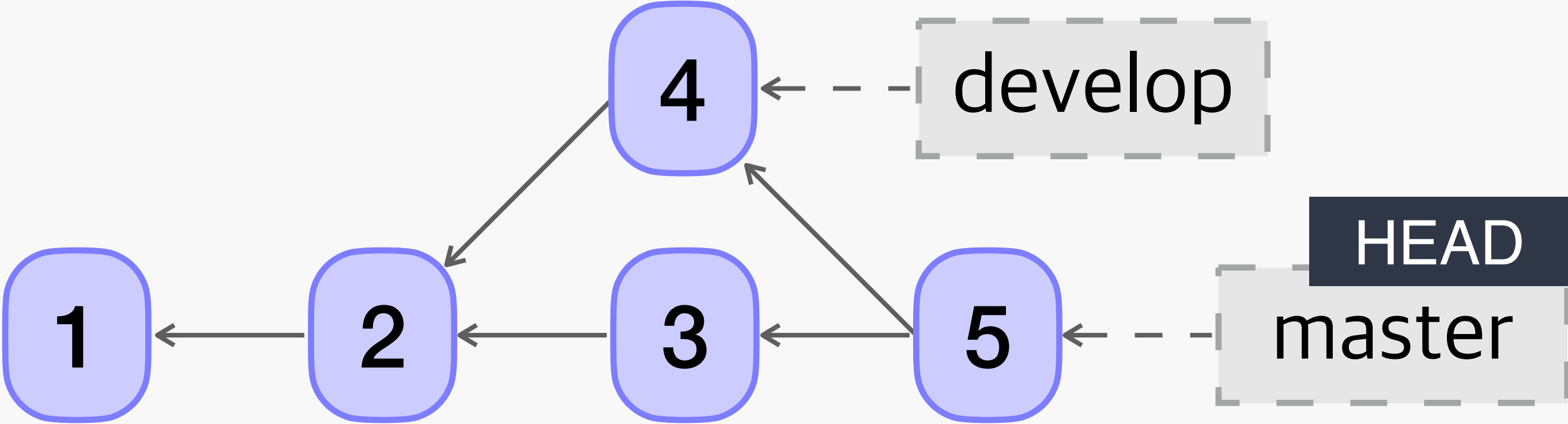
在待合并分支与目标分支都有新的提交，怎么合并？

是否可以 fast-forward 呢?



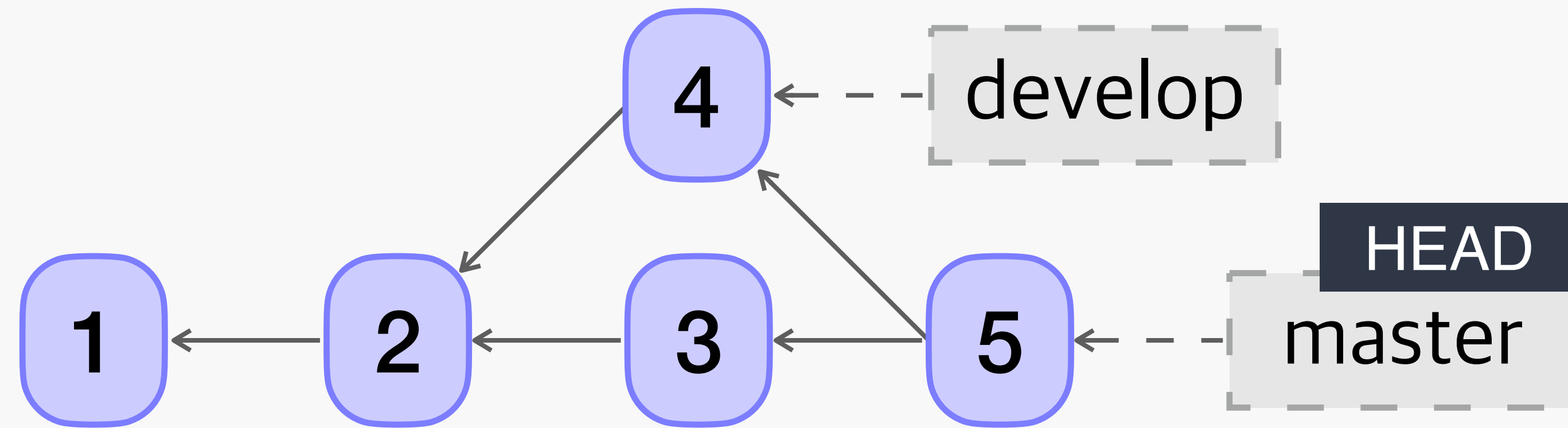
并不能 fast-forward merge，只能进行 non-fast forward merge

自动合并不冲突修改



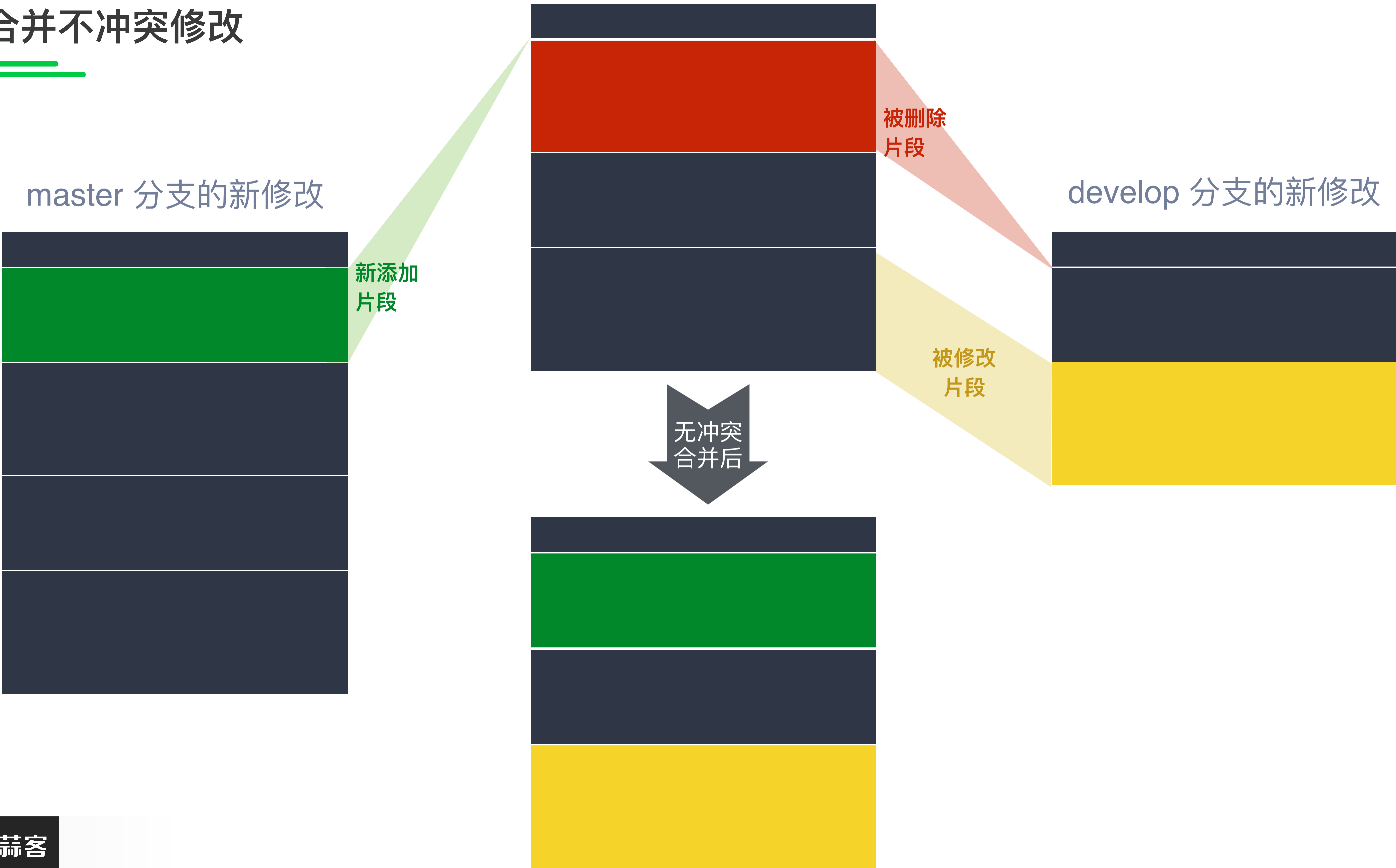
如果 提交4（develop 上的所有新提交） 和 提交3（master 上的所有新提交）
修改的是不同的部分的内容，git 会很聪明的把两个分支的修改放到一起
然后创建一个新的 提交 5 说明进行了合并

是否可以 fast-forward 呢?



并不能 fast-forward merge, 只能进行 non-fast forward merge

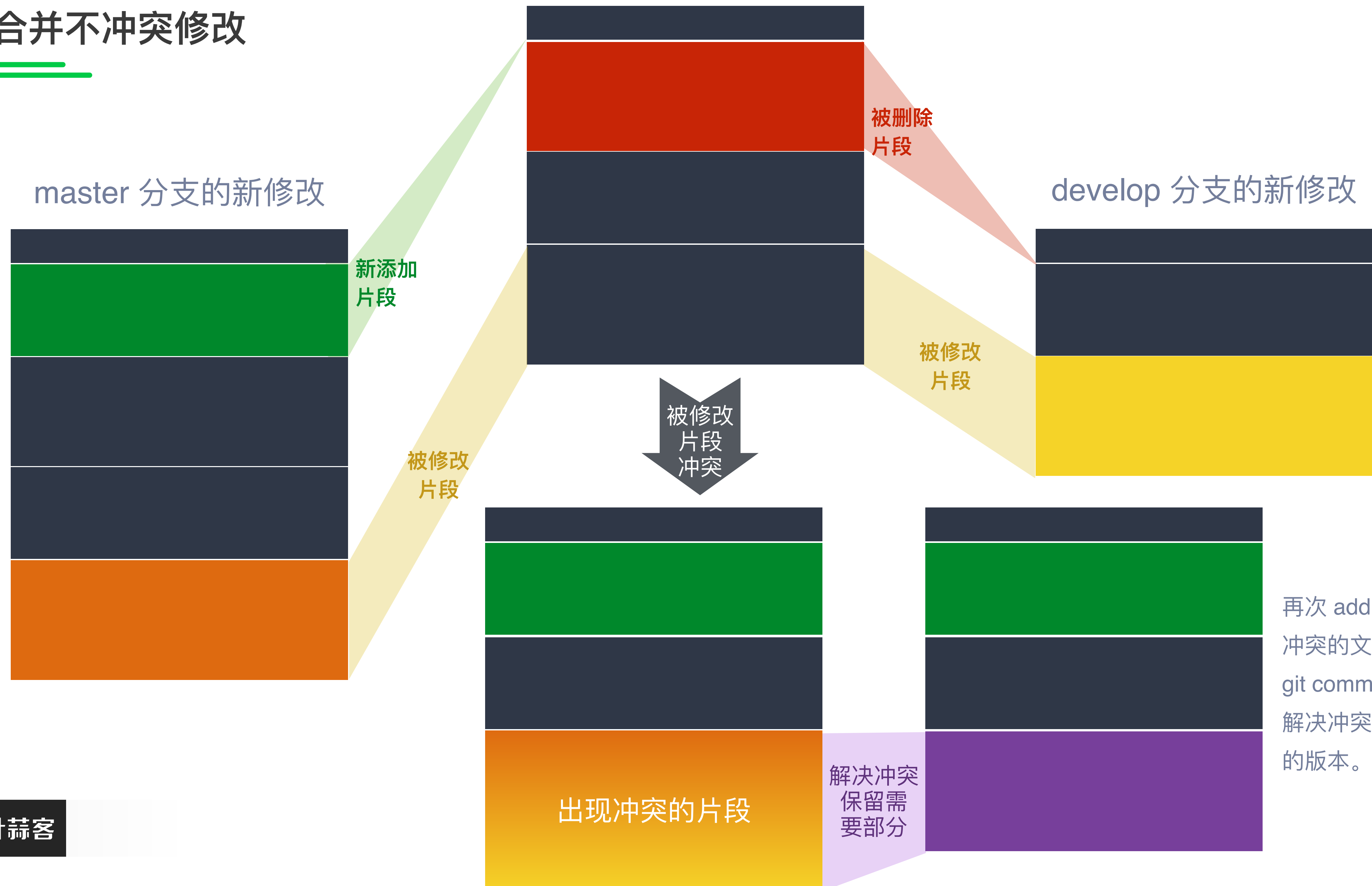
自动合并不冲突修改



合并的默认行为规则

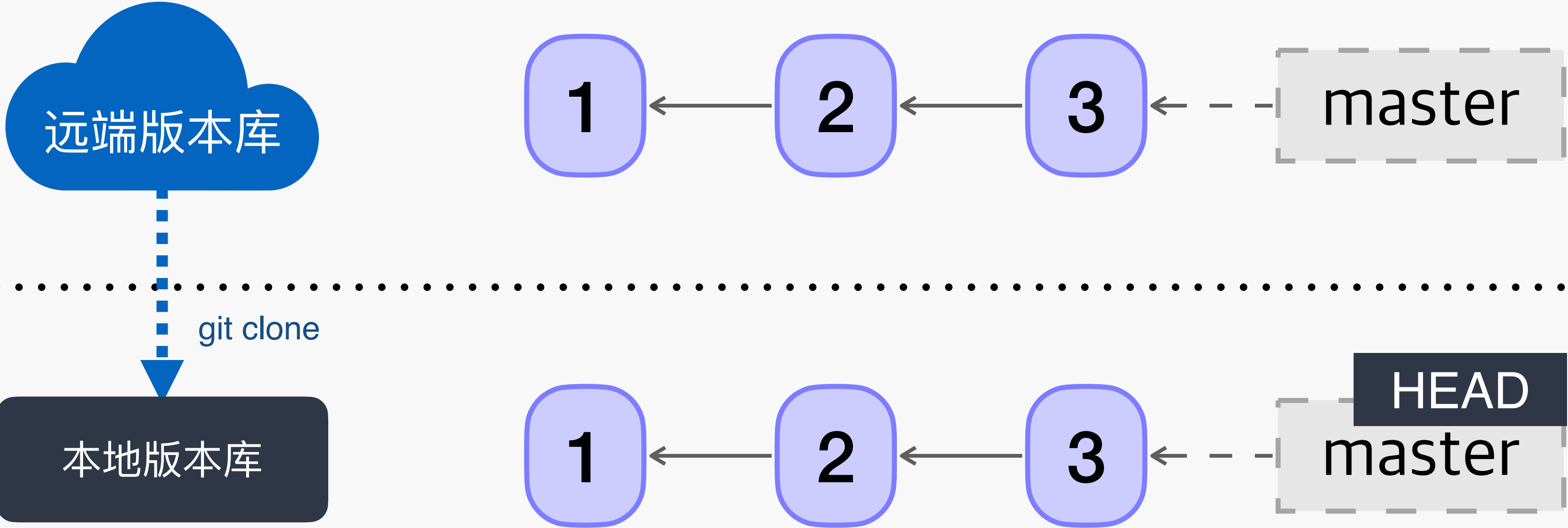
- 只执行 `git merge` 时，如果可以 `fast-forward`，则默认 `fast-forward`，否则进行 `non fast-forward merge`。
- 在 `non-fast-forward` 合并过程中，会尝试合并修改，如果发生不能自动合并的冲突，则需要手动解决冲突。

自动合并不冲突修改



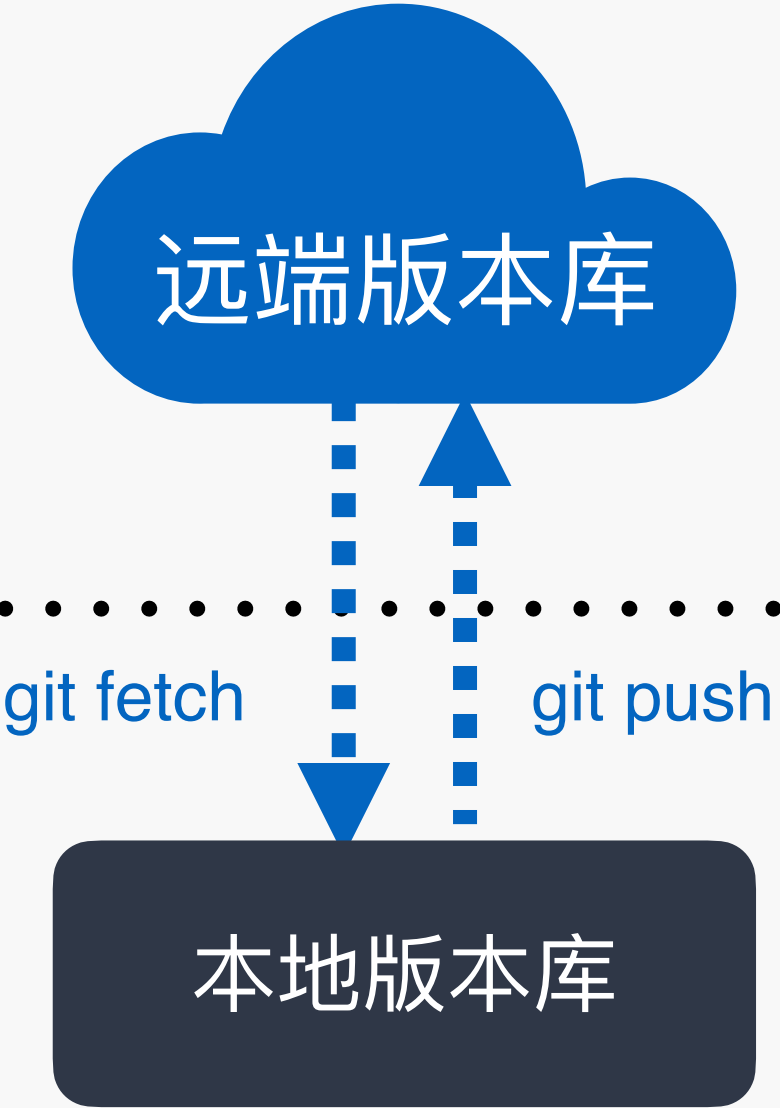
Git 的远端仓库交互

再次理解一下 git clone



git clone <远端版本库地址> 其实把远端的 master 分支放到了我们的本地并且让远端版本库的 master 分支和我们本地的版本库的 master 分支对应起来了

查看远端版本库情况



```
→ dockerenv git:(nginx) git remote -v
origin  git@se.jisuanke.com:course/dockerenv.git (fetch)
origin  git@se.jisuanke.com:course/dockerenv.git (push)
```

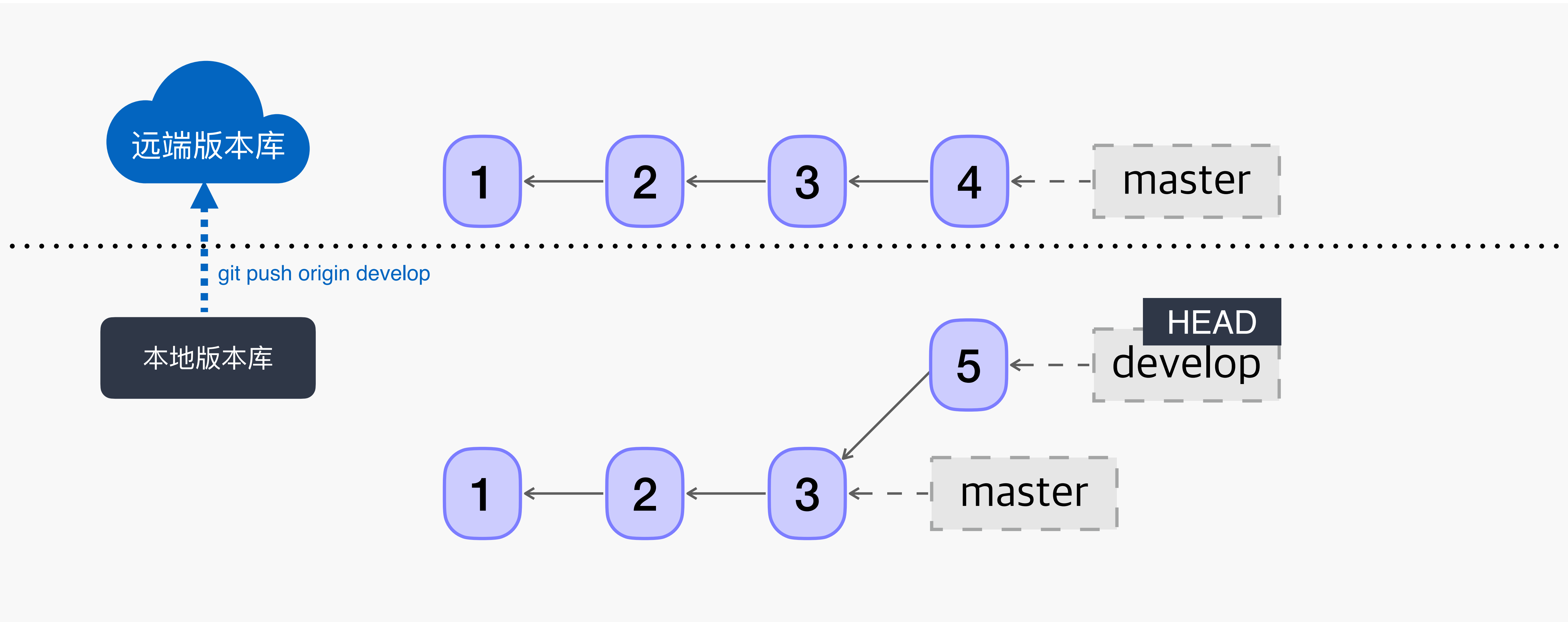
通过 `git remote -v` 可以查看当前本地版本库对应的远端版本库
从远端克隆的版本库的默认远端名称是 `origin`

关联本地的版本库和远端版本库

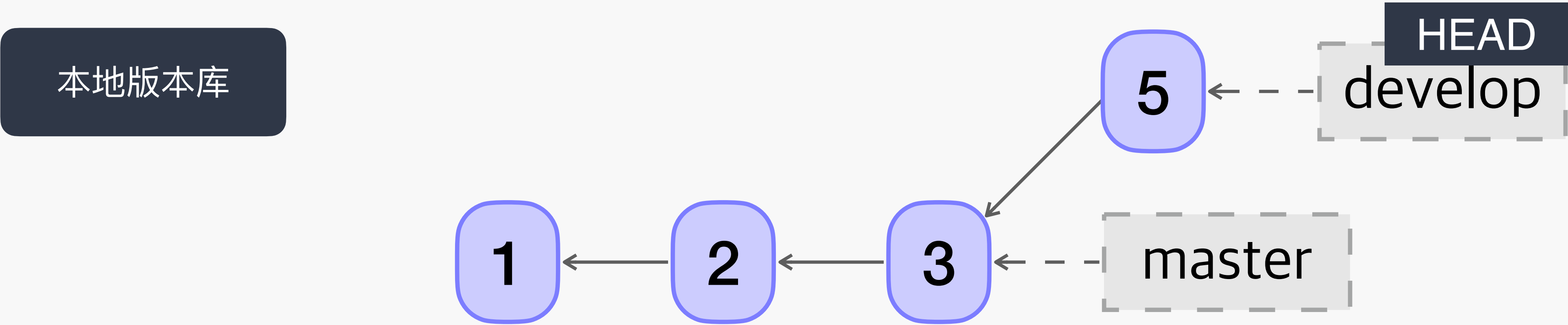
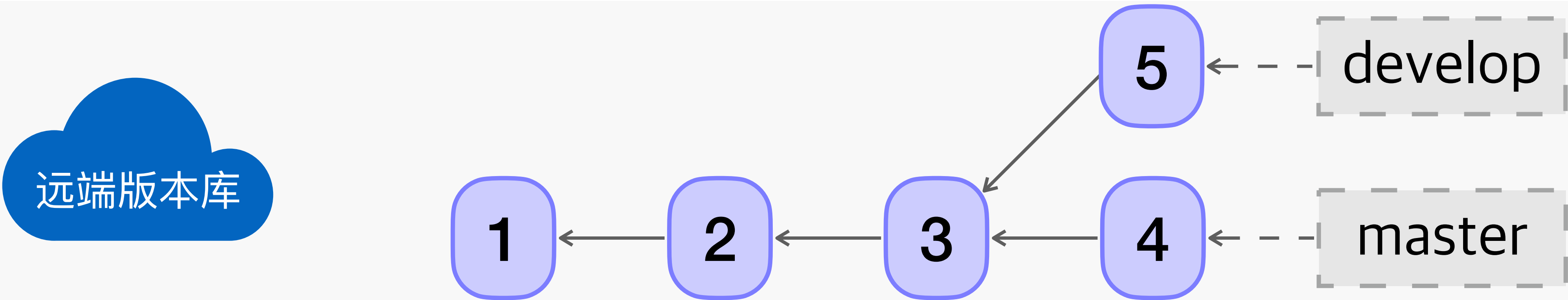


如果我们在本地的版本库没有设置过远端版本库（例如通过 `git init` 在本地生成的）
我们可以通过 `git remote add origin <远端版本库地址>`
的方式添加远端版本库

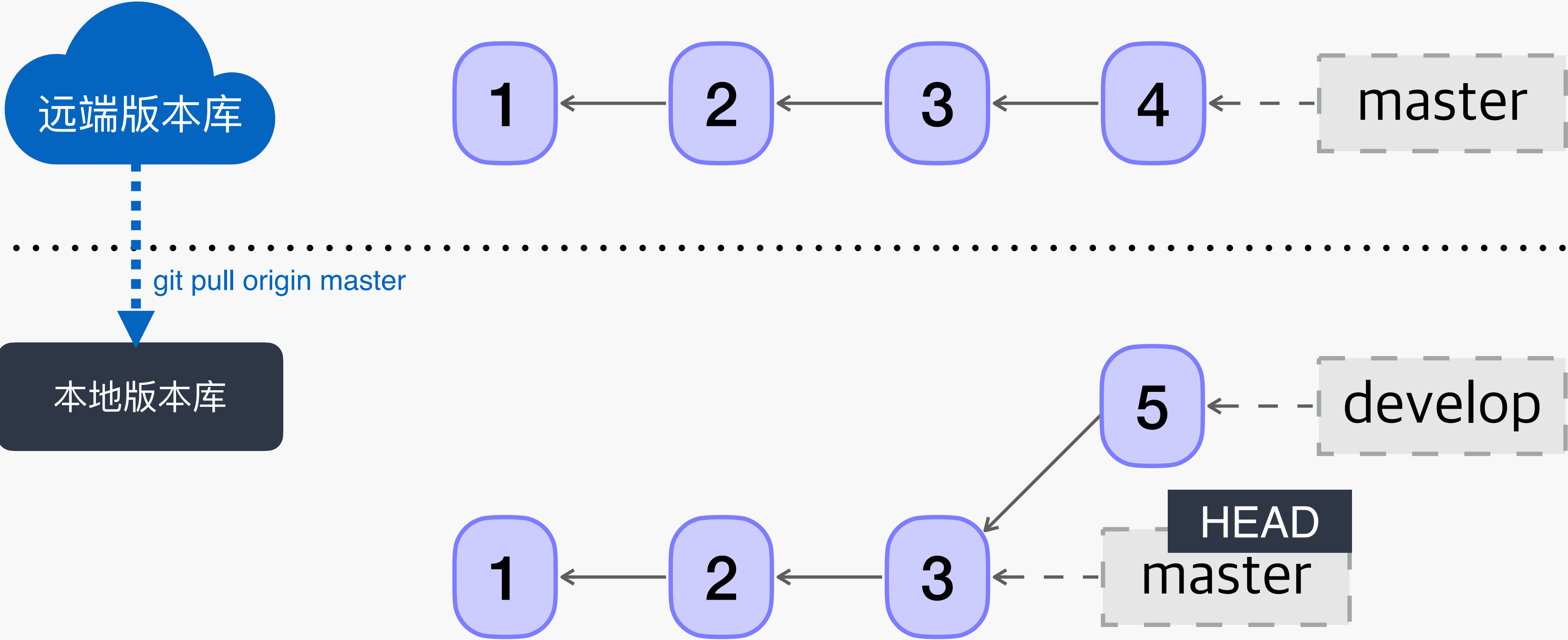
将本地的修改推到远端



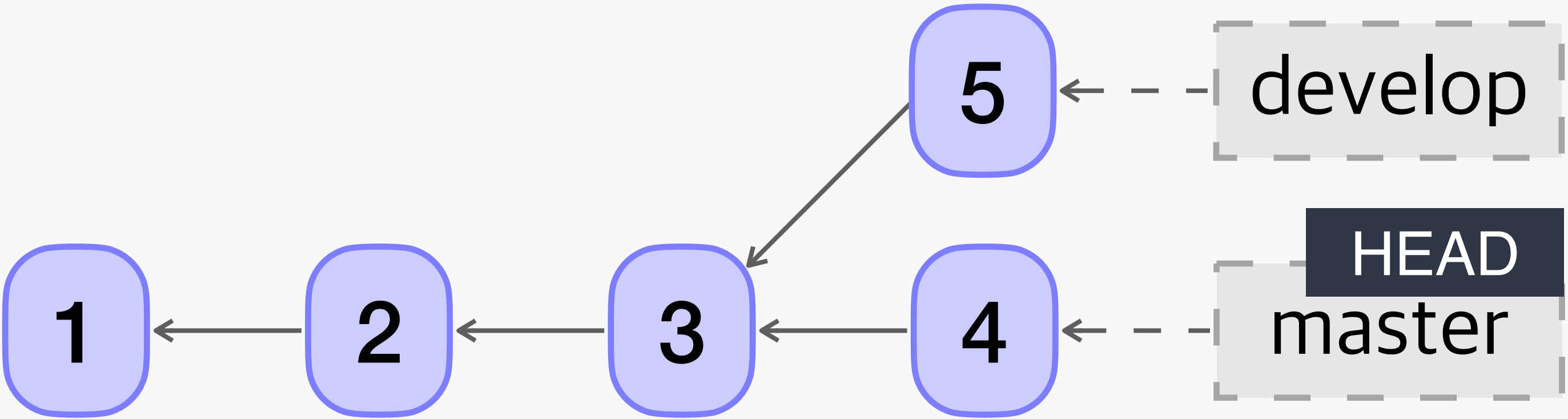
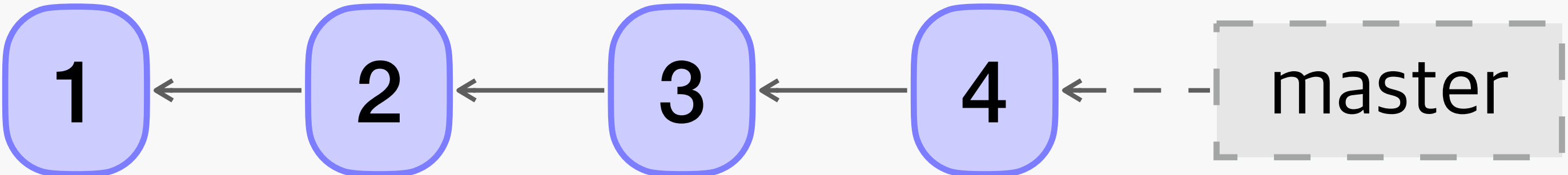
将本地的修改推到远端



将远端的修改拉到本地



将远端的修改拉到本地



拉取和推送

01

拉取（全部分支）更新并且不更新工作目录

```
git fetch
```

02

拉取（全部分支）更新并根据本地分支对应分支更新工作目录

```
git pull
```

03

从远端 origin 的指定分支进行拉取

```
git pull origin <分支名>
```

04

推动更新

```
git push
```

05

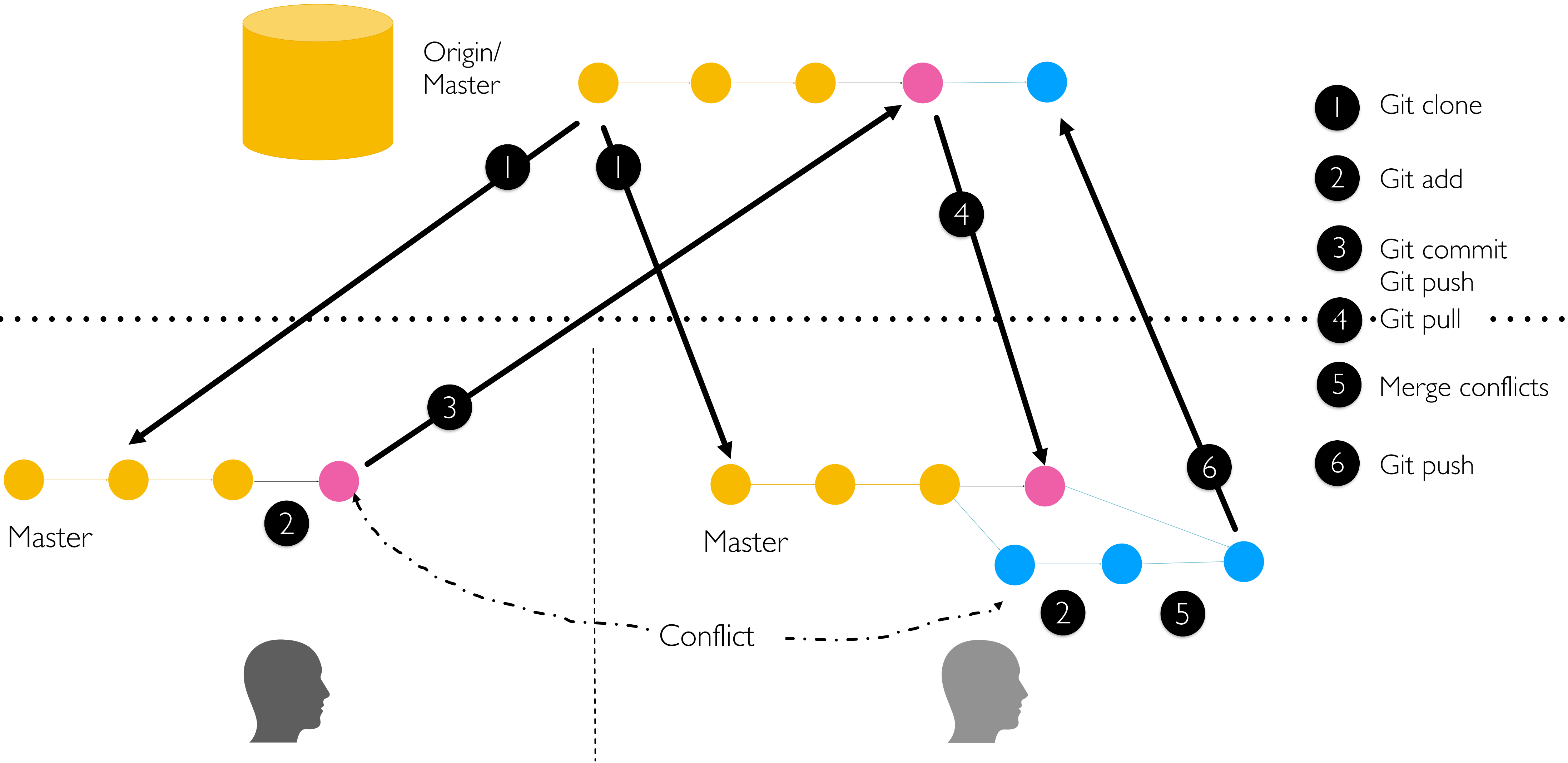
向远端 origin 的指定分支名分支进行推送

```
git push origin <分支名>
```

场景：中心化工作流

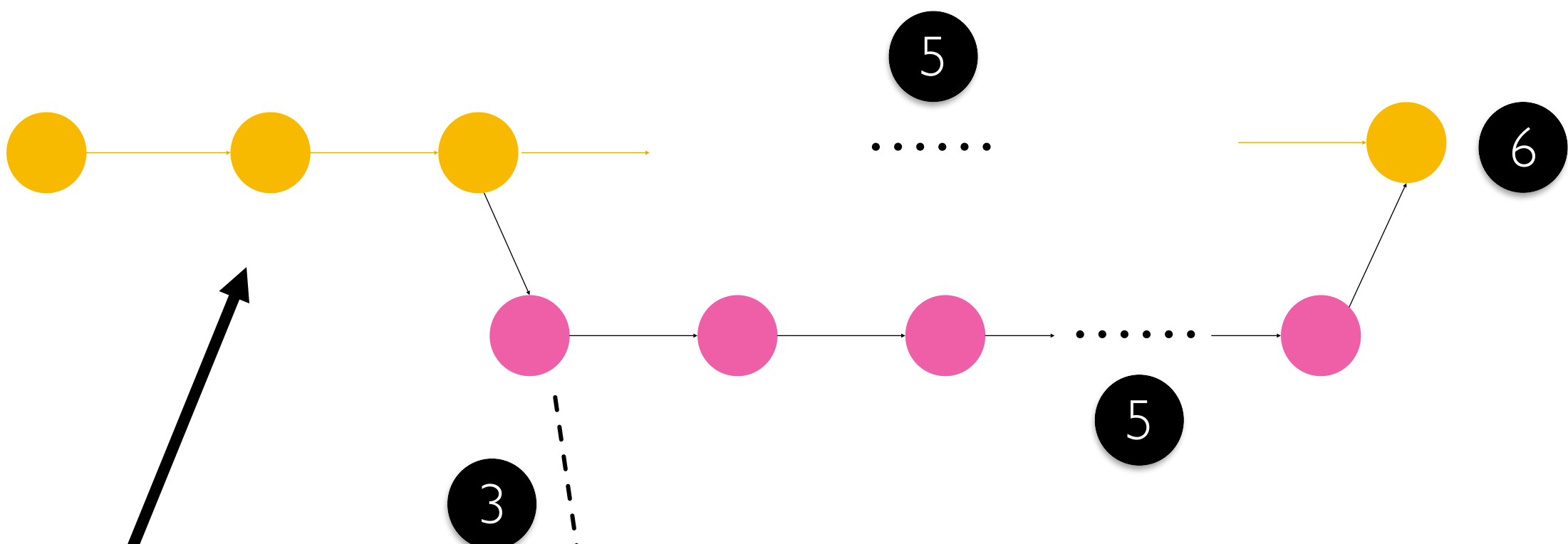
远端仓库

本地仓库



场景：功能分支 workflow

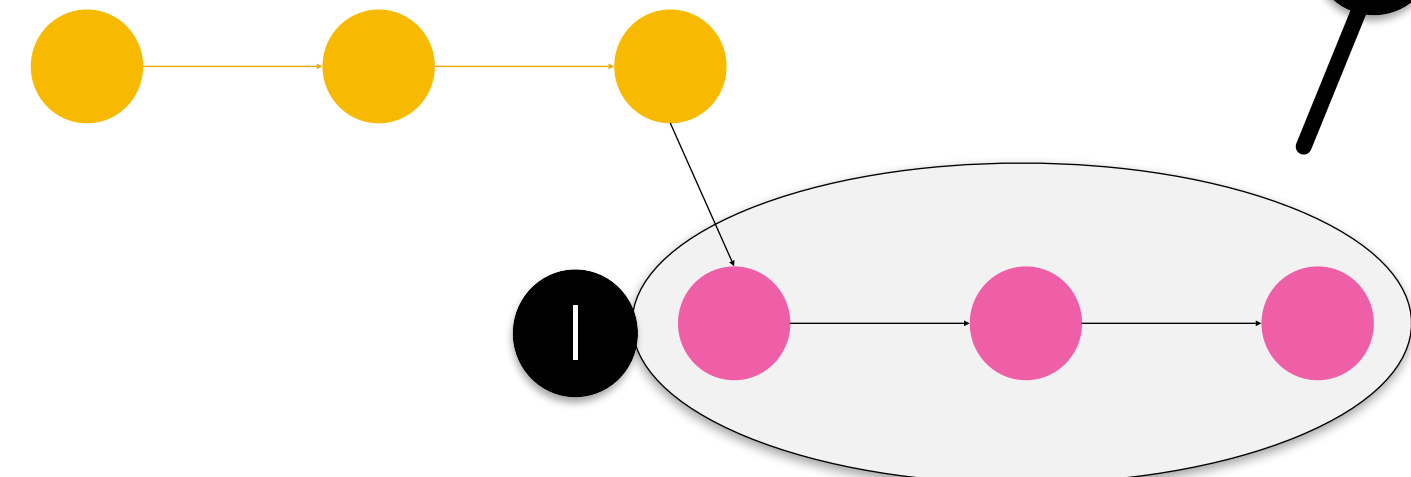
远端仓库



- 1 Git checkout
- 2 Git push
- 3 Notify
- 4 Git pull
- 5 Revision history
- 6 Merge

本地仓库

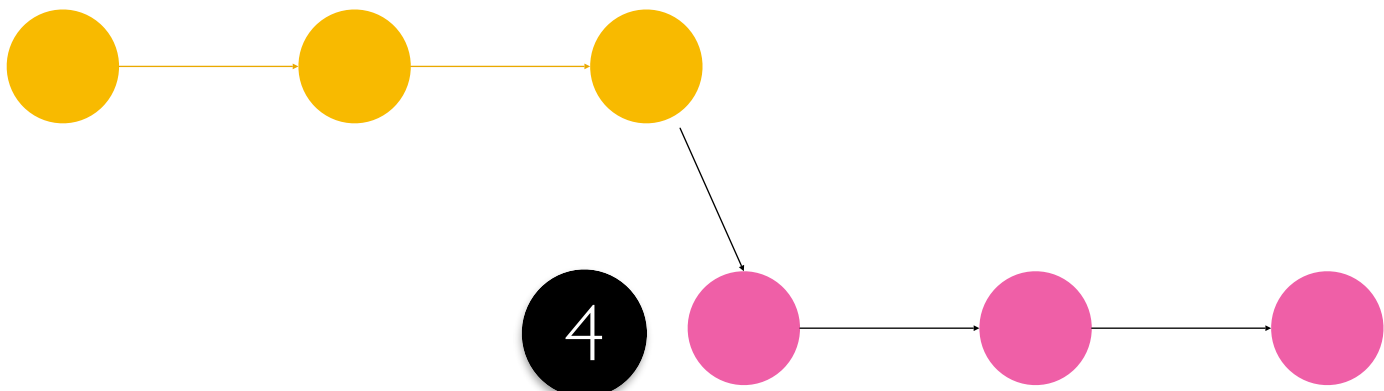
Master



A new branch

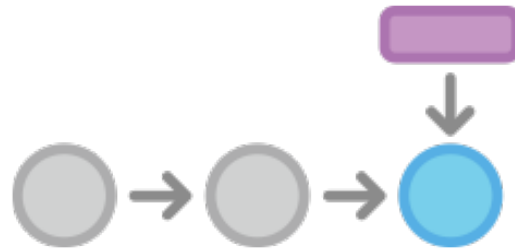


Master



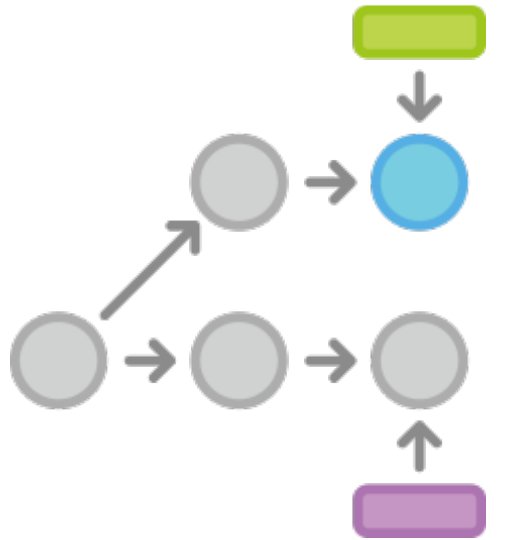
基本的 Git 协作开发使用回顾

Git 分支



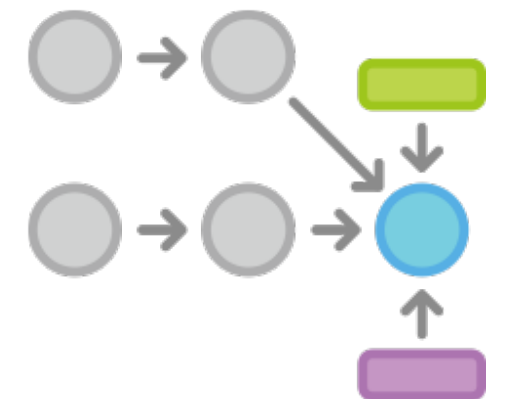
Git branch

使用当前环境创建一个当前点
(含 Log) 完全一致的分支



Git checkout


切换到指定的版本 (或通过分支指定的版本)




Git merge

完成合并: 将一个分支的修改应用到当前分支


远程仓库交互




Git remote




Git fetch



Git pull



Git push

 计蒜客

81

- <https://github.com/tj/git-extras>
- <https://speakerdeck.com/halyph/git-remote-branch-comsamples>
- <https://www.atlassian.com/git/tutorials/using-branches>
- <https://git-scm.com/book/en/v2/Git-Branching-Branched-in-a-Nutshell>
- <https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging>
- <https://speakerdeck.com/matthetmo/understanding-git>
- <https://book.douban.com/subject/26107548/>