ThoughtWorks®

GLOBAL SOFTWARE CONSULTANCY

# Git 101

唐洪 htang@thoughtworks.com
姜玉珍 htang@thoughtworks.com

# 自我介绍

8

唐洪

2018

分布式

Senior Consultant

DDD

高性能

# 自我介绍

5

姜玉珍

2018

web前端

Consultant

可视化

# 目录 CONTENTS

ThoughtWorks®

# 1 课程导入

# 课程目标

版本控制

ThoughtWorks®

# 2 版本控制简介

# 毕业设计

- 毕业设计.doc
- 毕业设计1.0.doc
- 毕业设计2.0.doc
- 毕业设计3.0.doc
- 毕业设计最终版.doc
- 毕业设计最终版1.0.doc
- 毕业设计最终不改版.doc
- 毕业设计打死不改版本.doc
- 毕业设计打死不改版本1.0.doc

.
.
.

# 版本控制

版本控制最主要的功能就是追踪文件的变更。它将什么时候、什么人更改了文件的什么内容等信息忠实地了记录下来。每一次文件的改变，文件的版本号都将增加。除了记录版本变更外，版本控制的另一个重要功能是并行开发。软件开发往往是多人协同作业，版本控制可以有效地解决版本的同步以及不同开发者之间的开发通信问题，提高协同开发的效率。并行开发中最常见的不同版本软件的错误(Bug)修正问题也可以通过版本控制中分支与合并的方法有效地解决。

# 版本控制类型

本地式

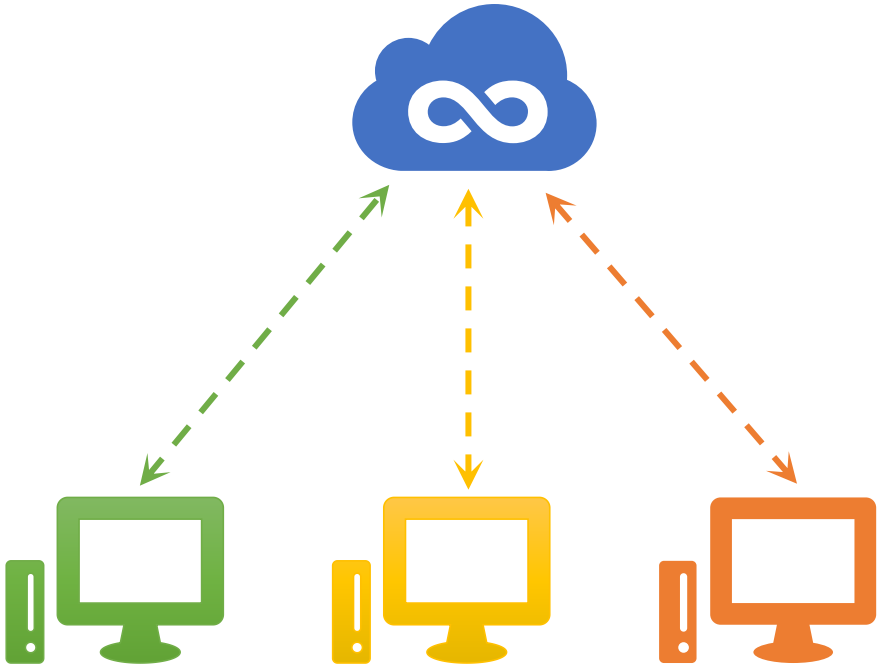
**优势**

- 搭建简单
- 没有网络连接要求

**劣势**

- 无法协同工作
- 有数据丢失风险
- 高依赖使用者

# 版本控制类型



集中式

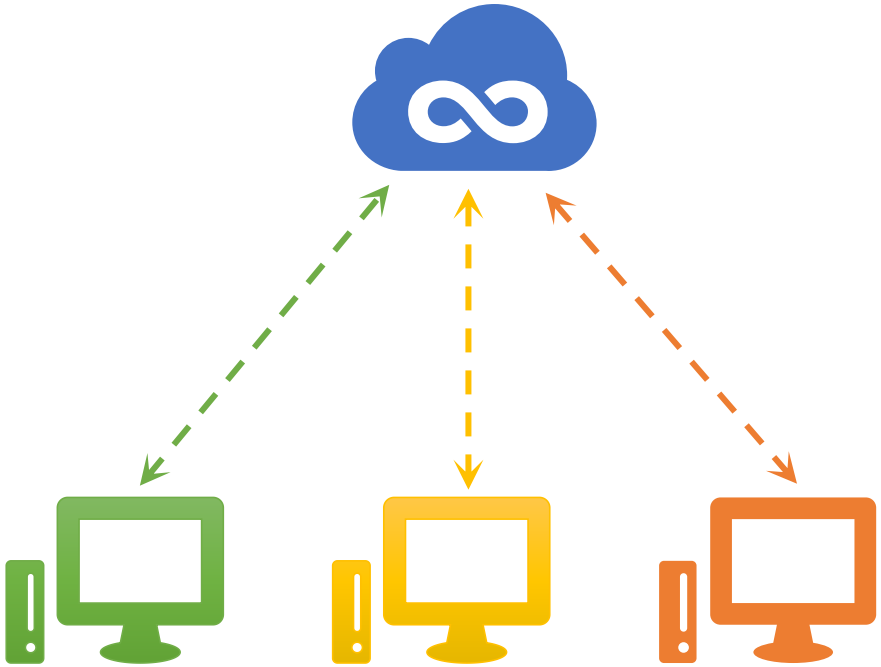
**优势**

- 服务化
- 可以协同开发

**劣势**

- 和中心服务强耦合
- 文件传输慢
- 冲突管理困难

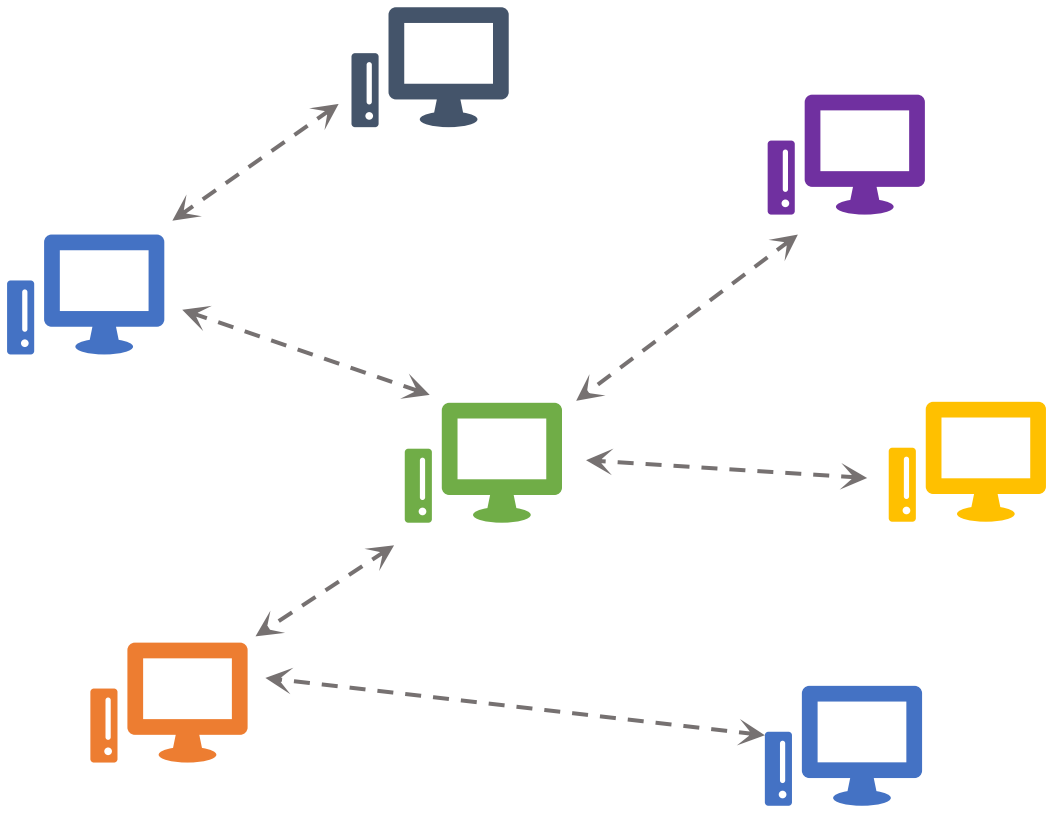
# 版本控制类型



集中式

代表产品


# 版本控制类型



分布式

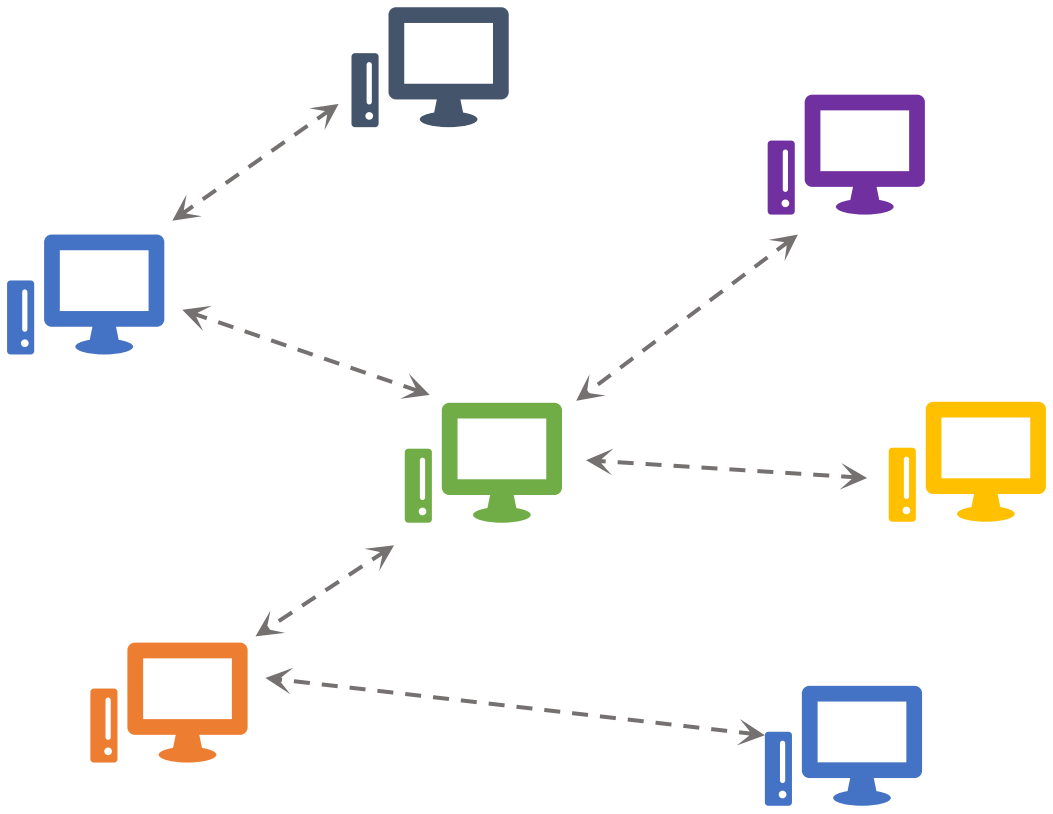
**优势**

- 不依赖中心服务
- 不强依赖网络
- 可以大规模协同开发
- 冲突管理简单

**劣势**

- 版控流程复杂

# 版本控制类型

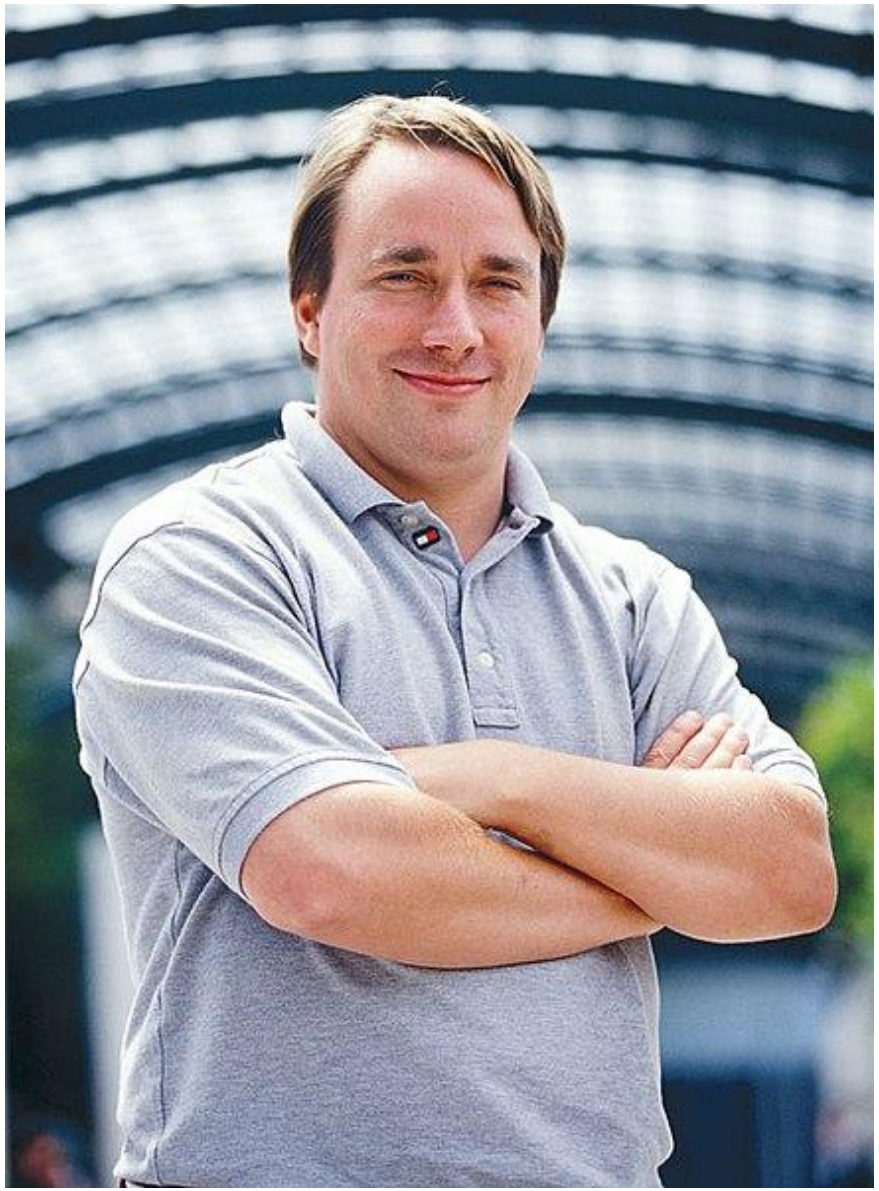


分布式

代表产品

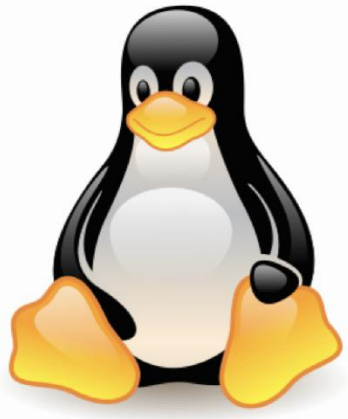

ThoughtWorks®

# 3 Git实操

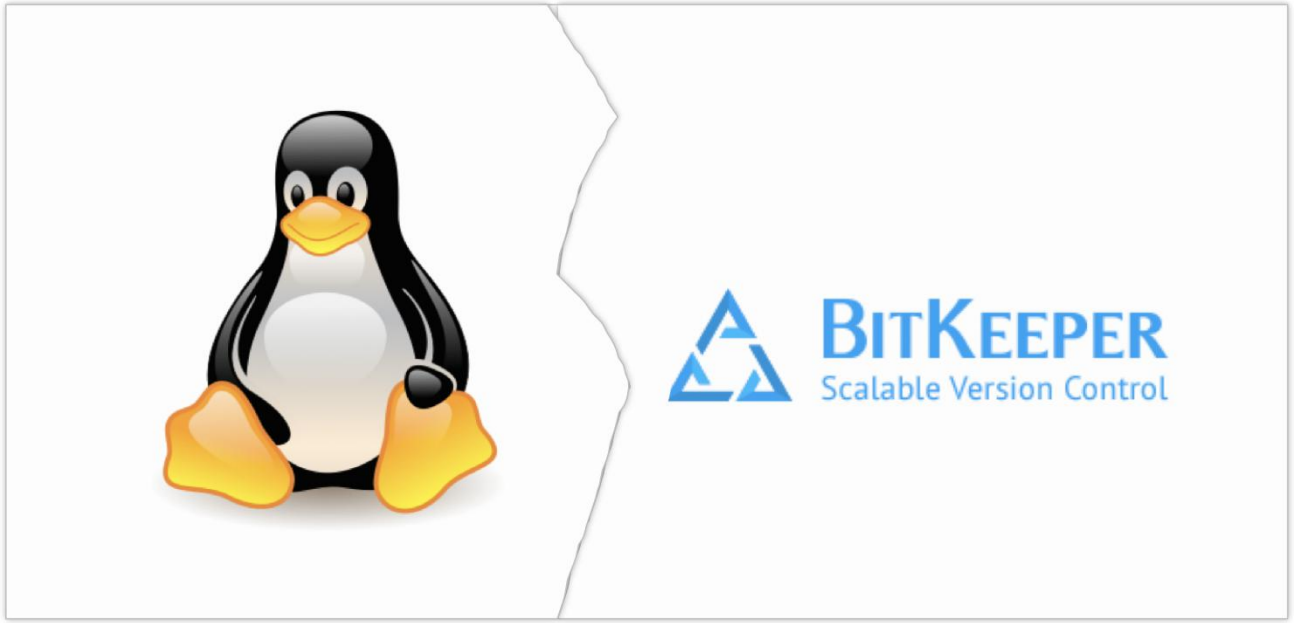


BitKeeper
Scalable Version Control

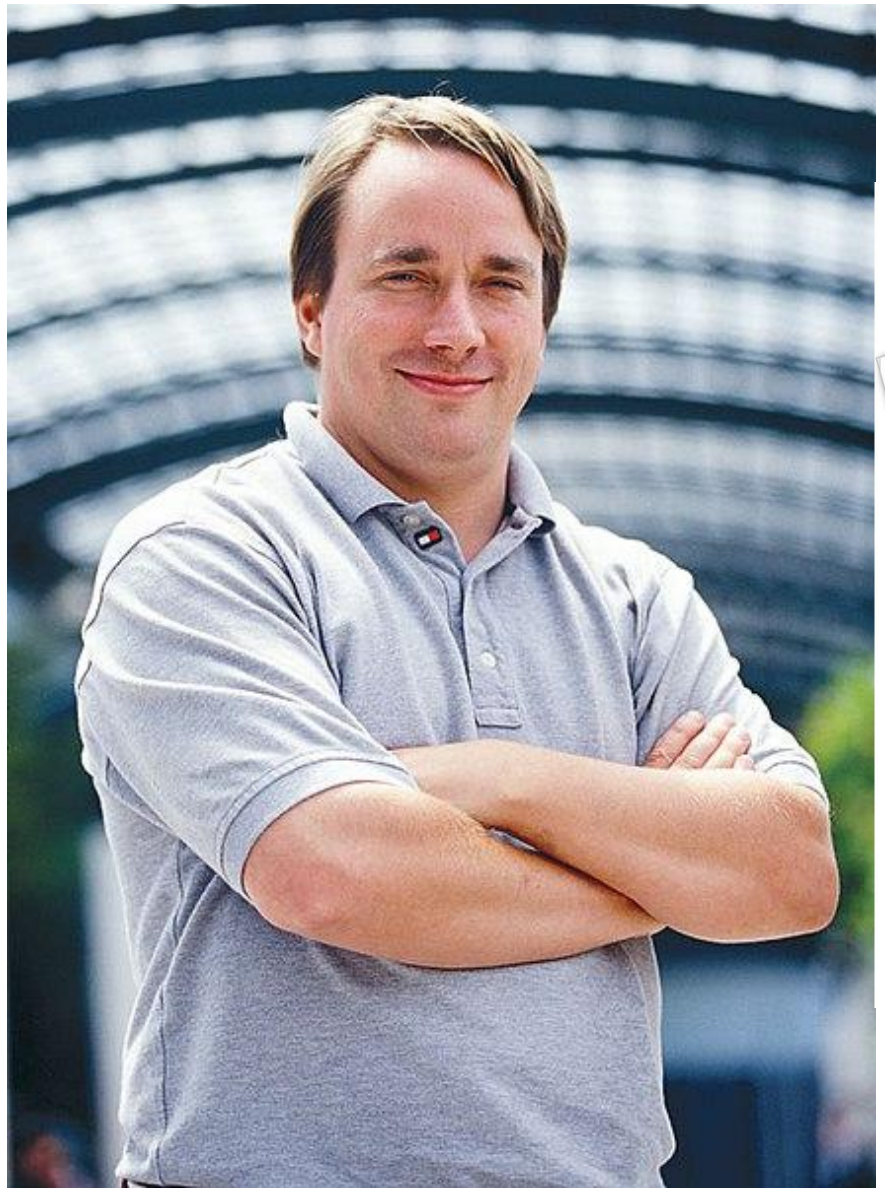
BitKeeper
Scalable Version Control



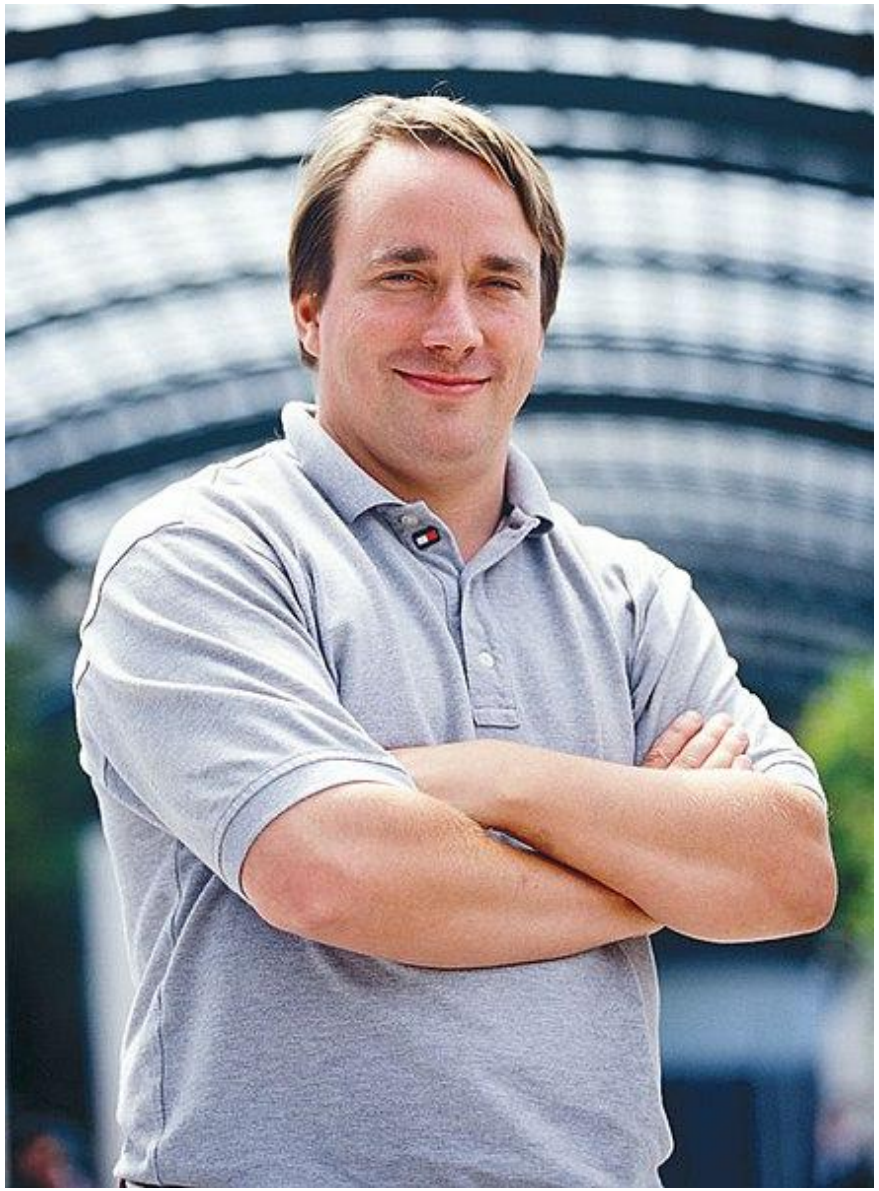
BitKeeper
Scalable Version Control


BitKeeper
Scalable Version Control


BitKeeper
Scalable Version Control

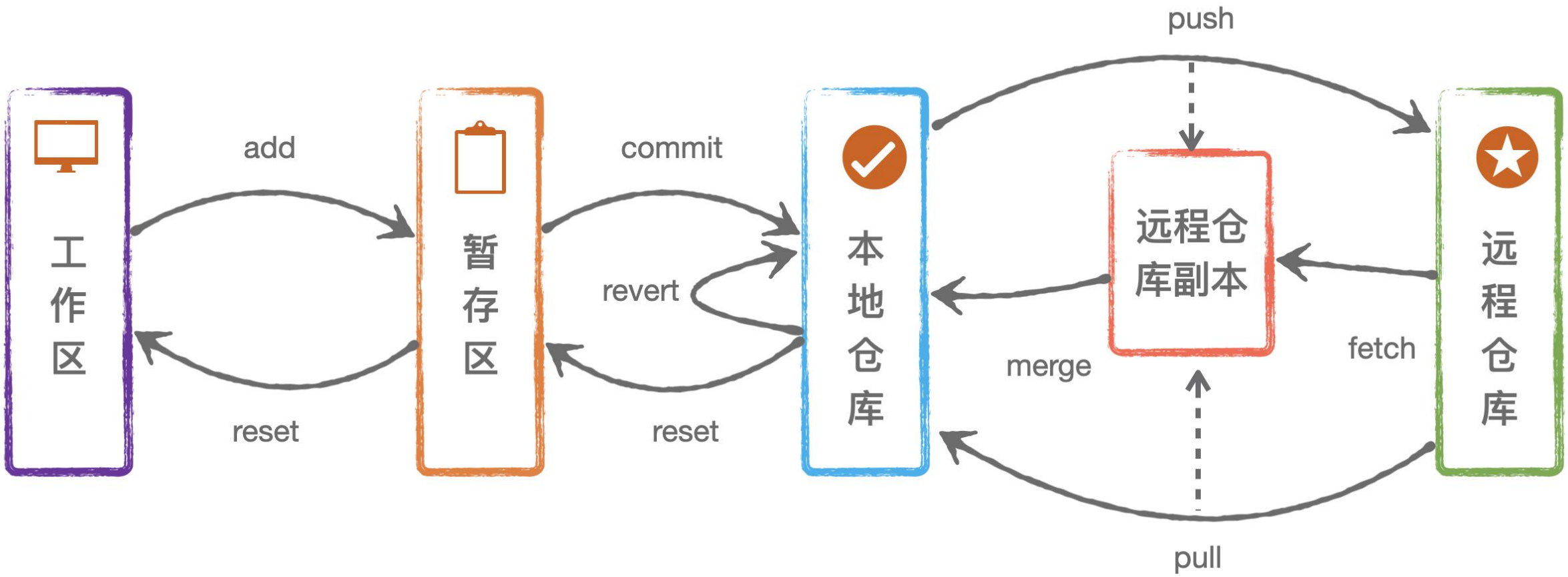



BitKeeper
Scalable Version Control




git

# 变更的一生


push
add
commit
工作区
暂存区
本地仓库
远程仓库副本
远程仓库
revert
merge
fetch
reset
reset
pull

# 准备工作

远程仓库

git clone

远程仓库

git init

# 前置环境的准备


Git 环境准备.pdf

# 我在哪，我干了什么

删除的文件

修改的文件

新增的文件

terminal

```
master$ git status
On branch master
Changes not staged for commit:
        deleted:    deleted.txt
        modified:   modified.txt
Untracked files:
        new.txt
```

工作区

# 谁还没点历史

提交摘要

作者

提交时间

提交描述

terminal

```
master$ git log
commit 33fef62b0893c30be2f381c66d10d20738952a09
(HEAD -> master, origin/master, origin/HEAD)
Author: XXX <XX@xx.com>
Date:   Sun Jun 20 21:56:56 2020 +0800

    commit message
```

仓库


工作区
add
暂存区
本地仓库
远程仓库副本
远程仓库

# 是时候登上舞台了

删除的文件

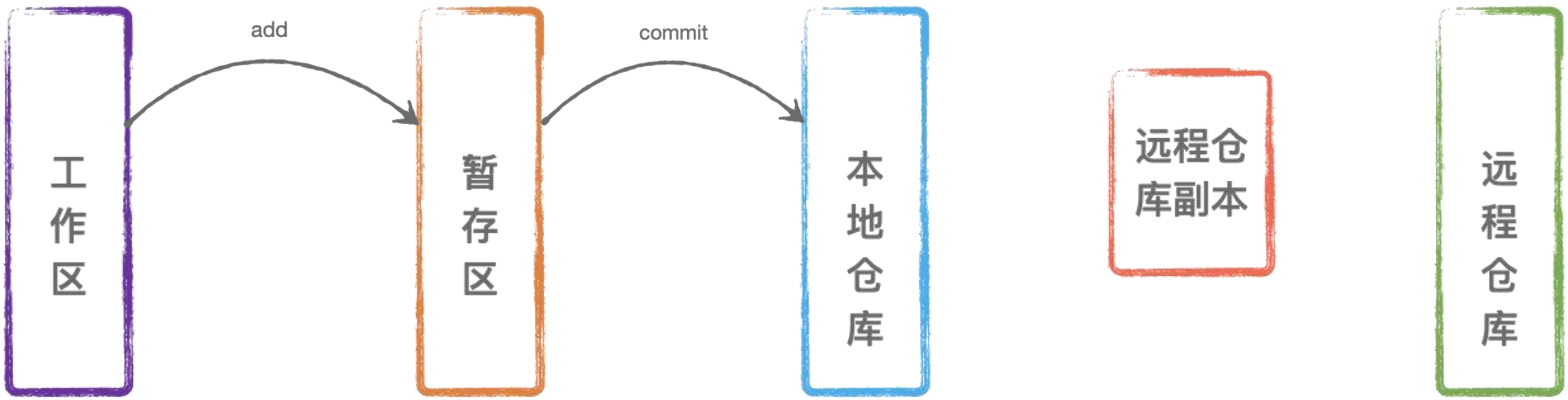
修改的文件

新增的文件

terminal

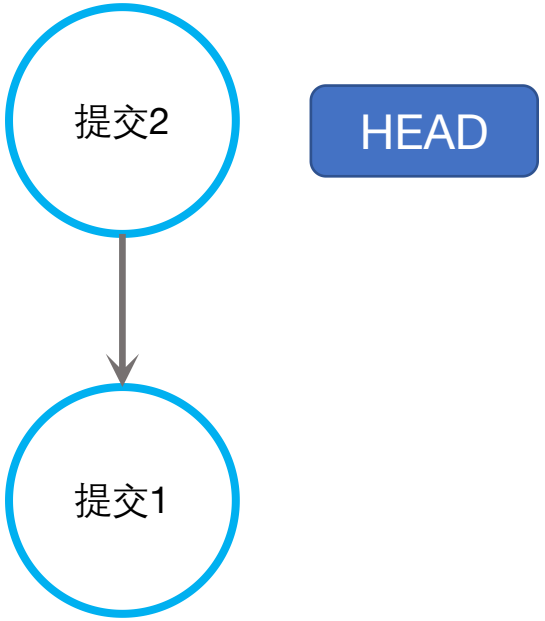
```
master$ git add .
master$ git status
On branch master
Changes to be committed:
        deleted:    deleted.txt
        modified:   modified.txt
        new file:   new.txt
```

工作区 → 舞台区


工作区
add
暂存区
commit
本地仓库
远程仓库副本
远程仓库

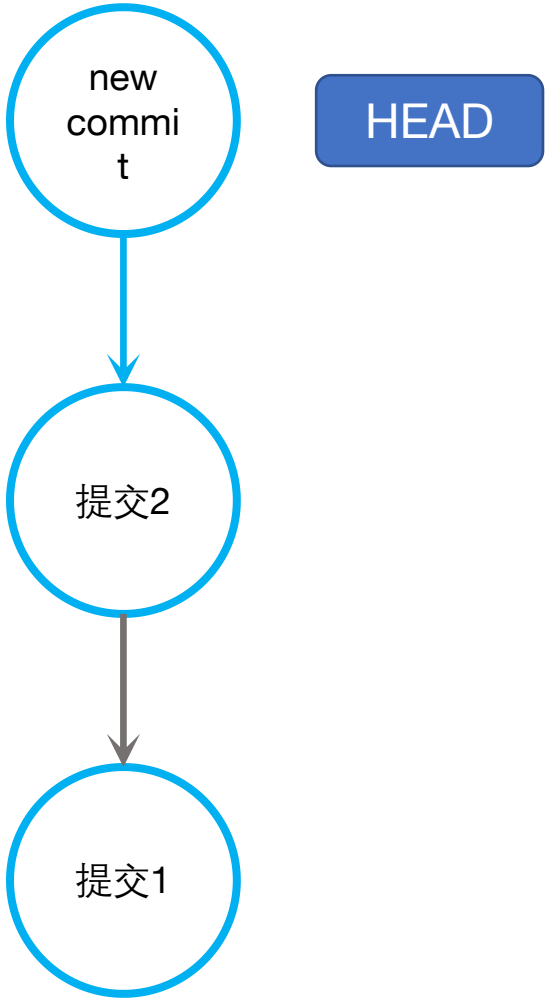
# 我确认好了

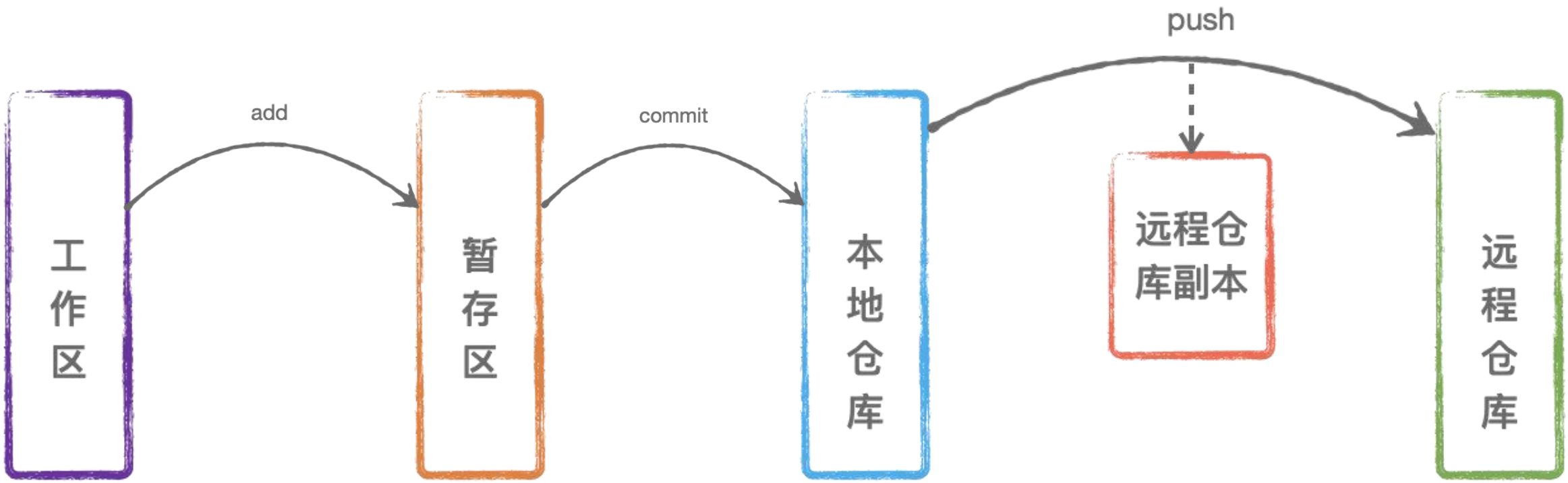
```
terminal
master$ git commit -m "new commit"
 3 files changed, 9 insertions(+), 2 deletions(-)
 delete mode 100644 deleted.txt
 create mode 100644 new.txt
```

# 我确认好了

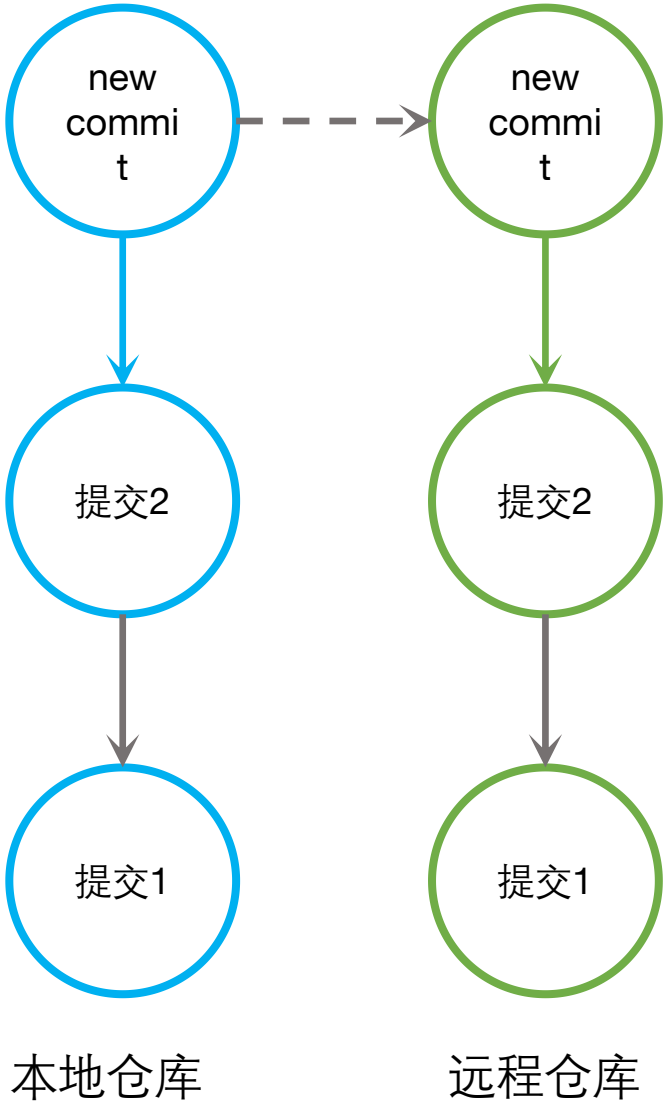
```
master$ git commit -m "new commit"
 3 files changed, 9 insertions(+), 2 deletions(-)
 delete mode 100644 deleted.txt
 create mode 100644 new.txt
```

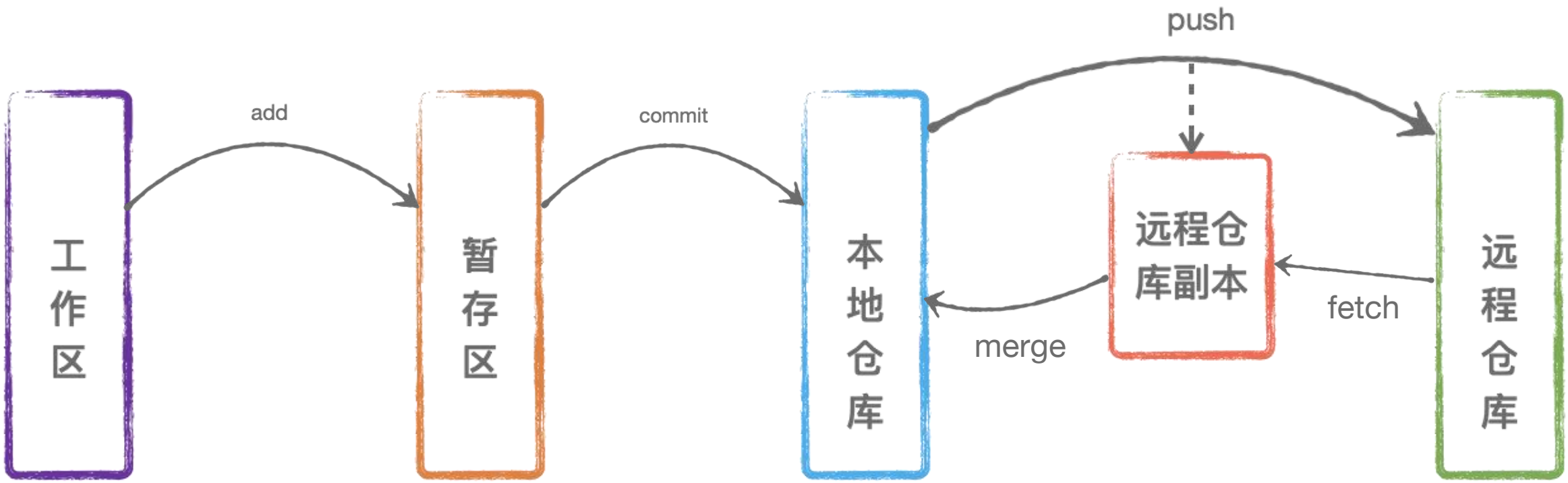




push
add
commit
工作区
暂存区
本地仓库
远程仓库副本
远程仓库

# 这里，我有一个很棒的想法

```
terminal
master$ git push
```

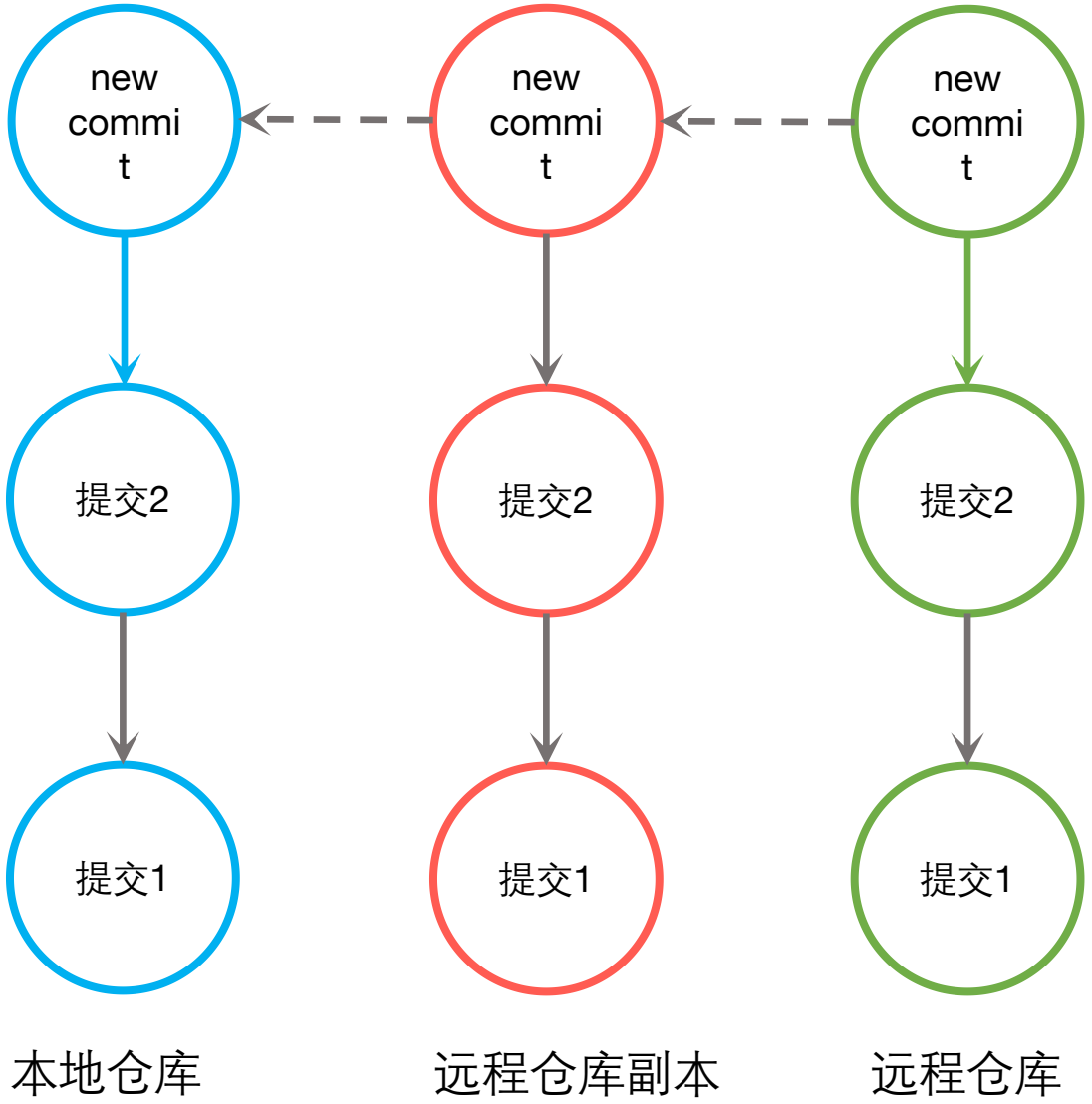



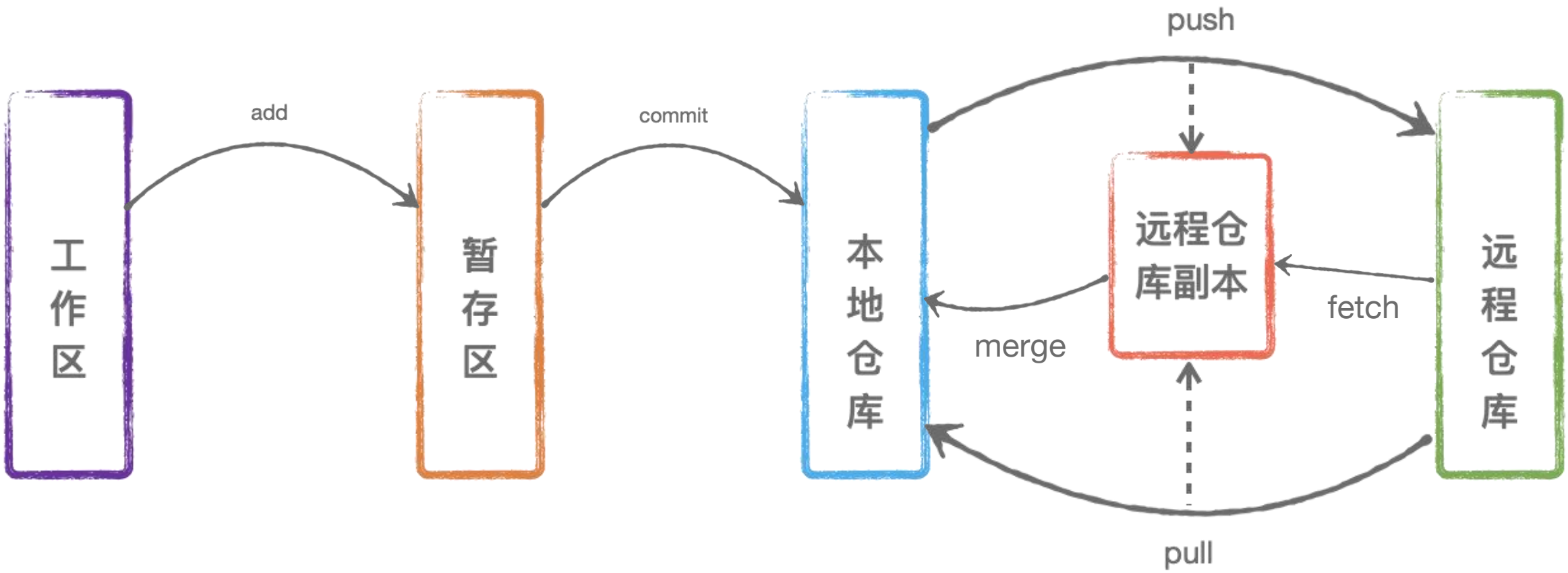

push
add
commit
工作区
暂存区
本地仓库
远程仓库副本
远程仓库
fetch
merge

# 让我来瞅瞅有多牛

```
terminal
master$ git fetch
master$ git merge origin/master
```

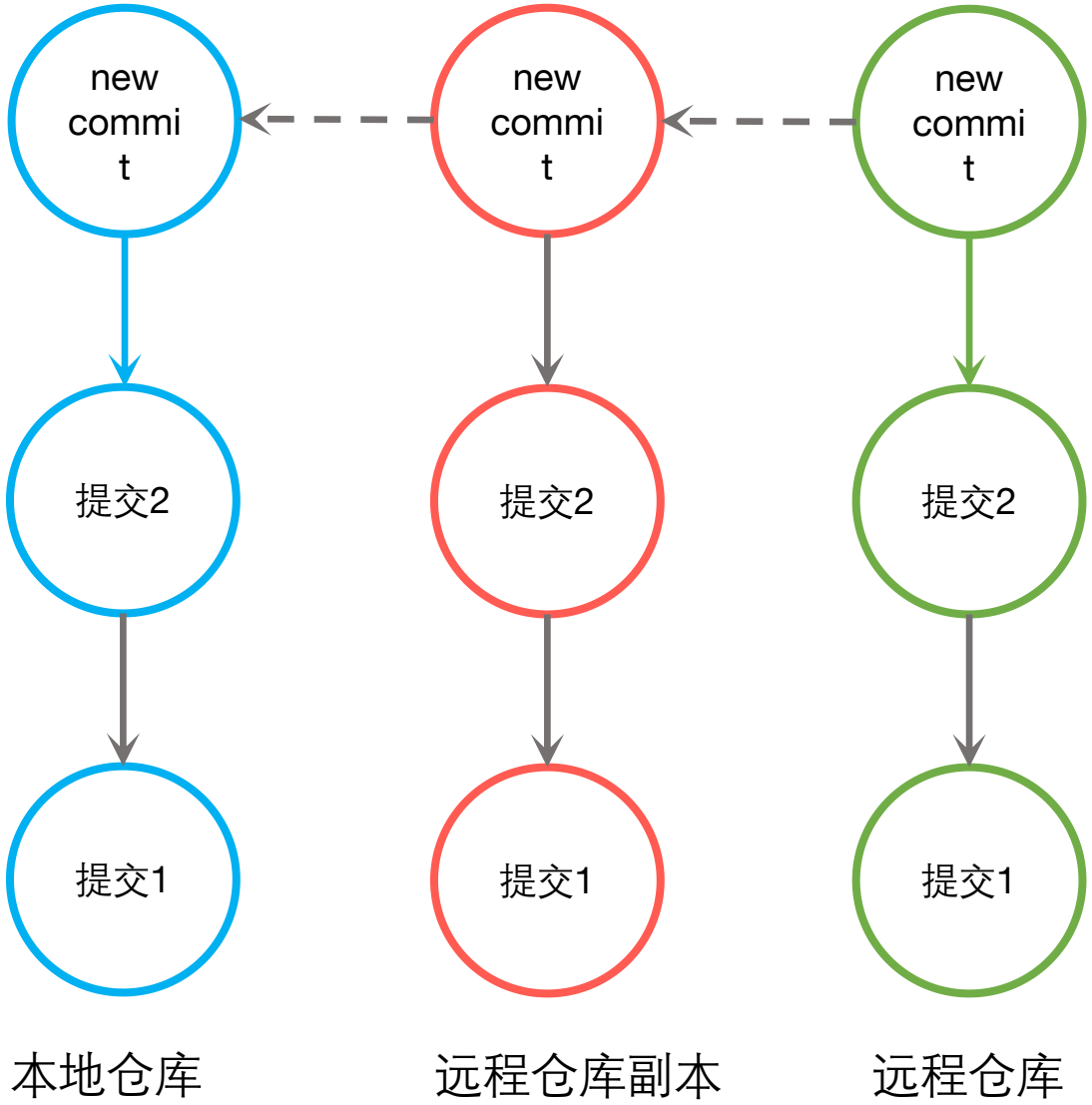
本地仓库 < 远程仓库

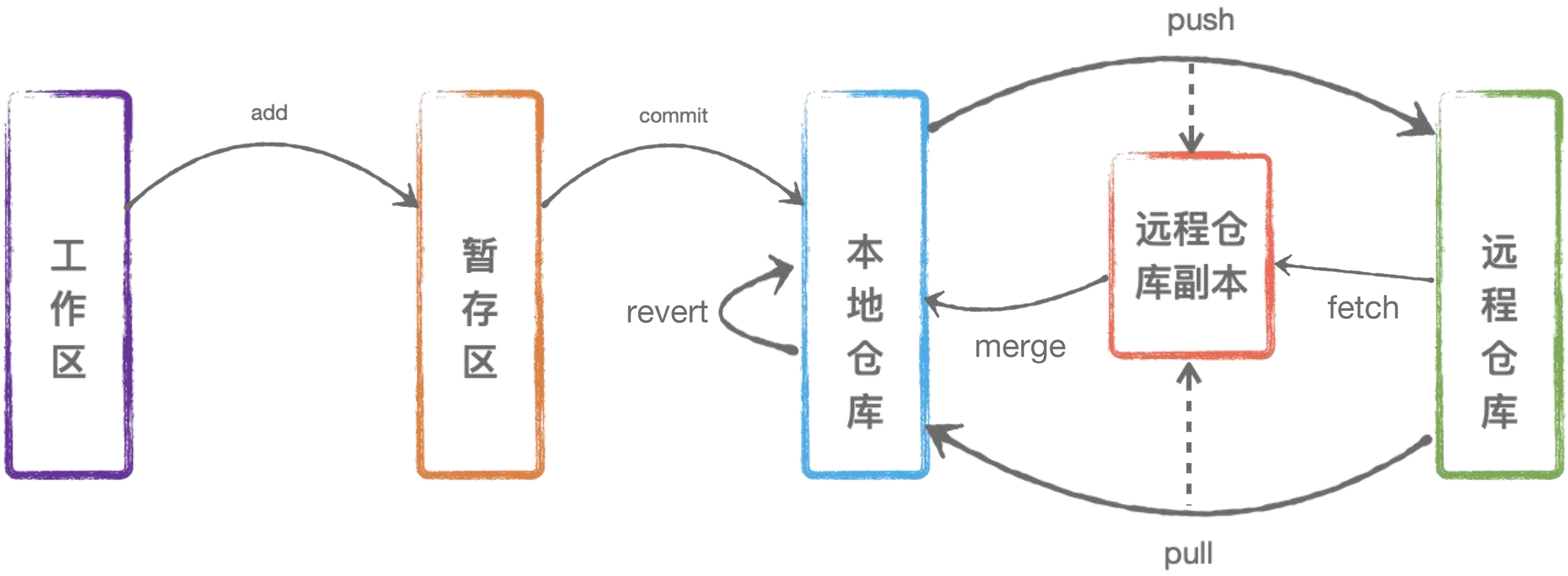



push
add
commit
工作区
暂存区
本地仓库
远程仓库副本
远程仓库
fetch
merge
pull

# 换个姿势来瞅瞅有多牛

```
terminal
master$ git pull
```

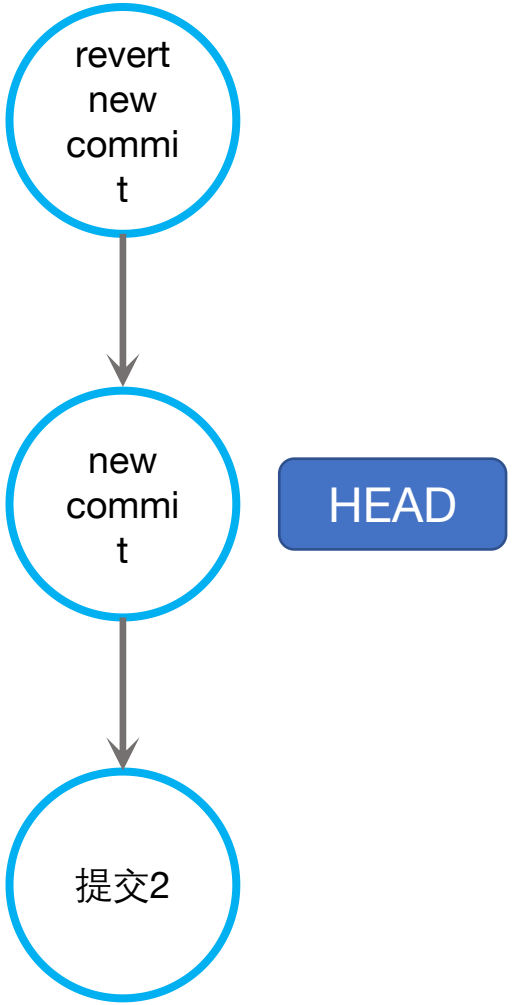




push
add
commit
工作区
暂存区
本地仓库
远程仓库副本
远程仓库
revert
merge
fetch
pull

# 你这也不行呀，从我眼前消失吧

```
terminal
master$ git revert sha1
```

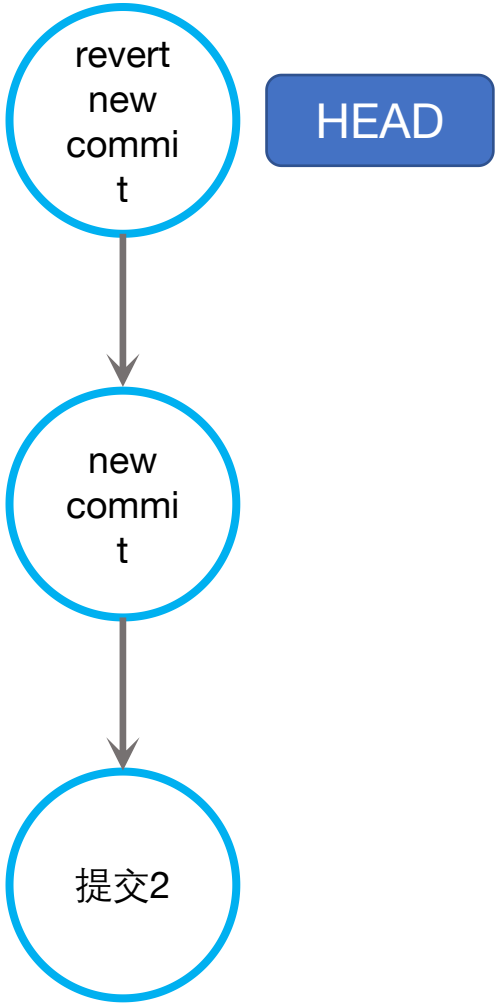
本地仓库

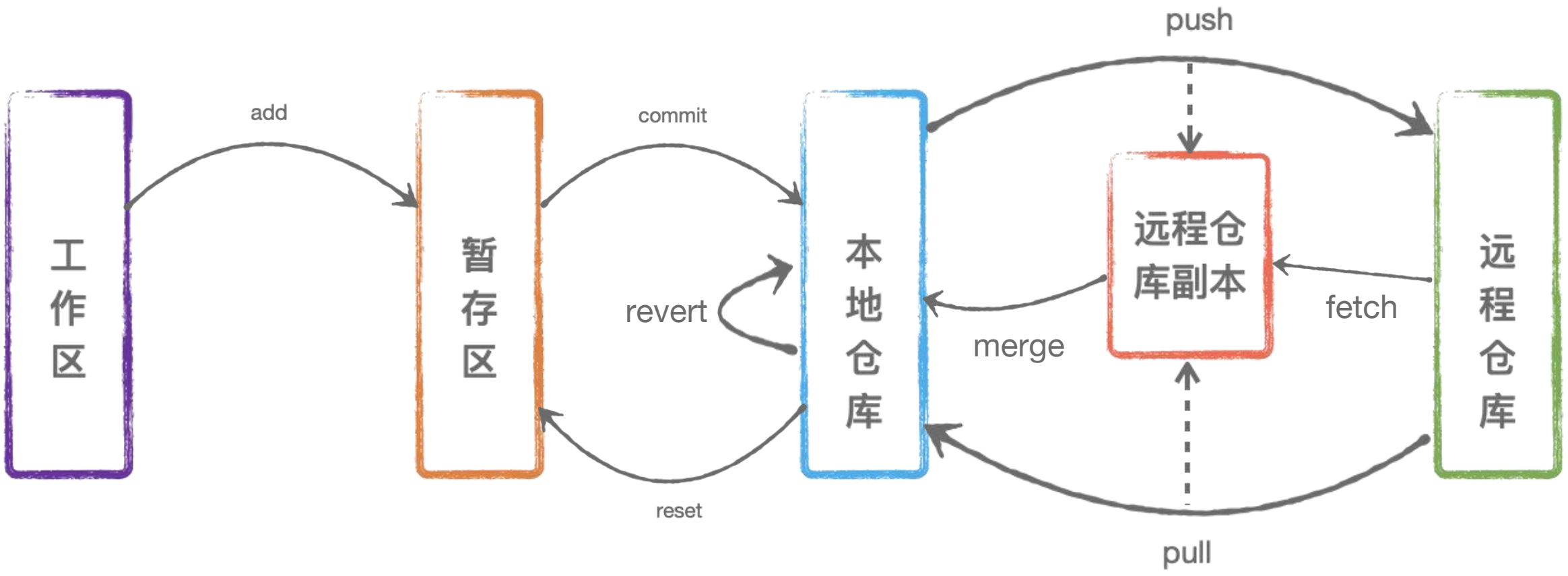


# 你这也不行呀，从我眼前消失吧

```
terminal
master$ git revert sha1
```

本地仓库

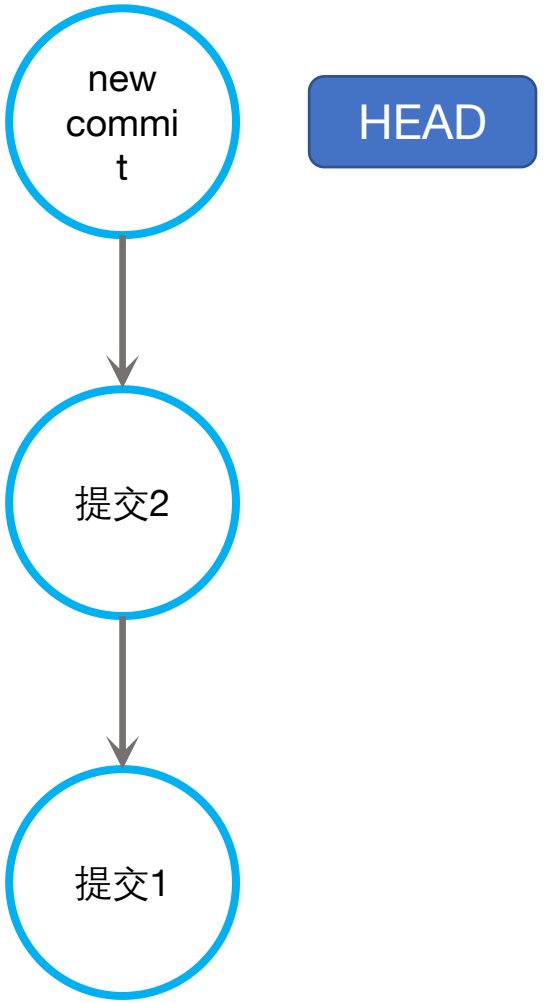



push
add
commit
工作区
暂存区
本地仓库
远程仓库副本
远程仓库
revert
merge
fetch
reset
pull

# 让你消失的第二种方式

terminal

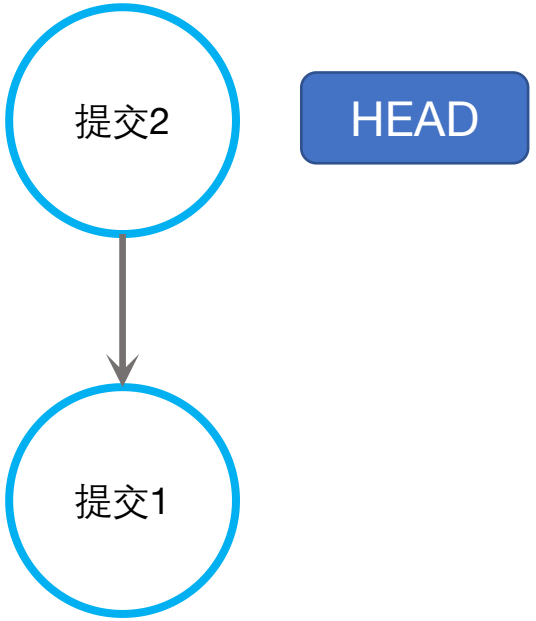
```
master$ git reset --soft HEAD^
```

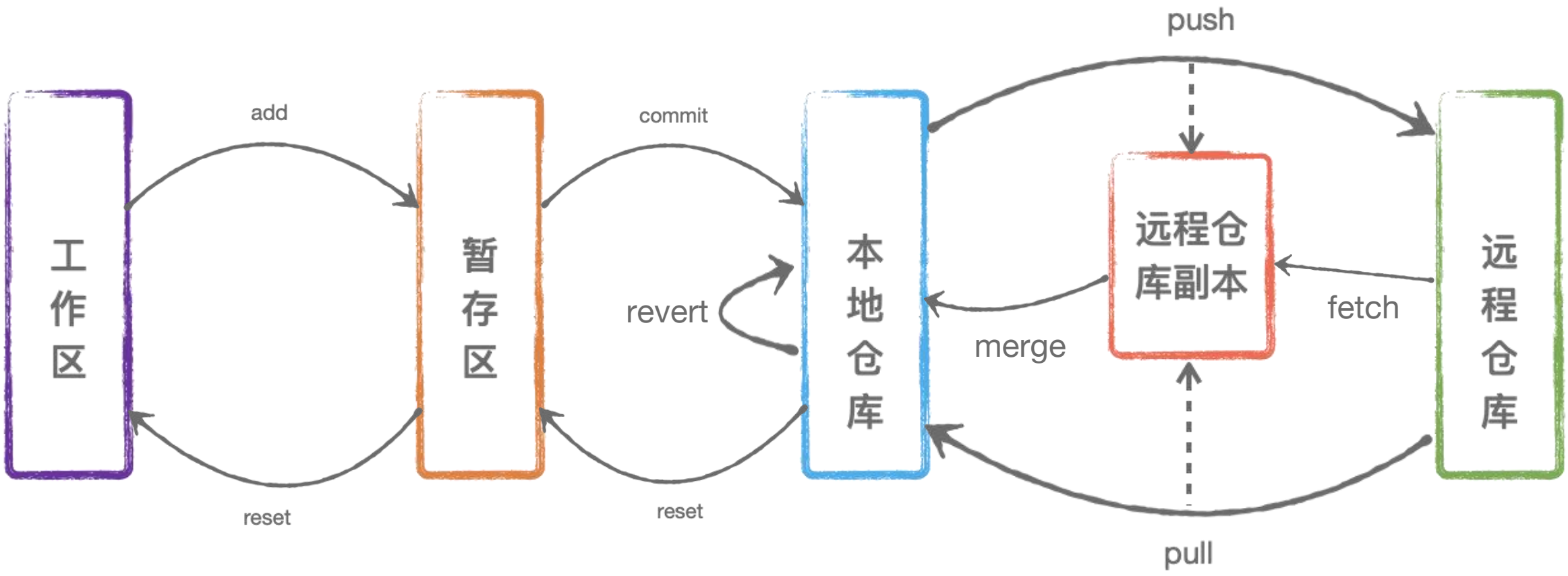

本地仓库

# 让你消失的第二种方式

```
terminal
master$ git reset --soft HEAD^
master$ git status
On branch master
Changes to be committed:
        deleted:    deleted.txt
        modified:   modified.txt
        new file:   new.txt
```

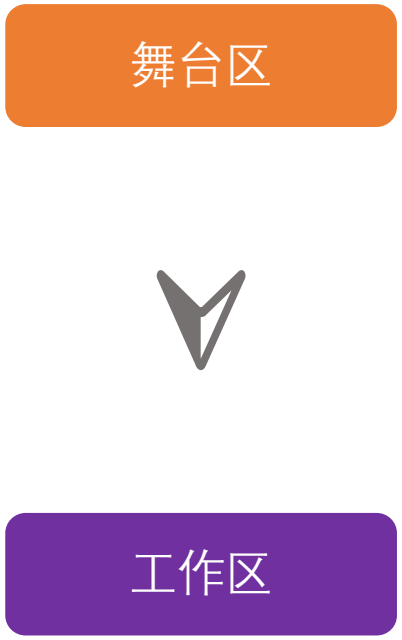

本地仓库


push
add
commit
工作区
暂存区
本地仓库
远程仓库副本
远程仓库
revert
merge
fetch
reset
reset
pull

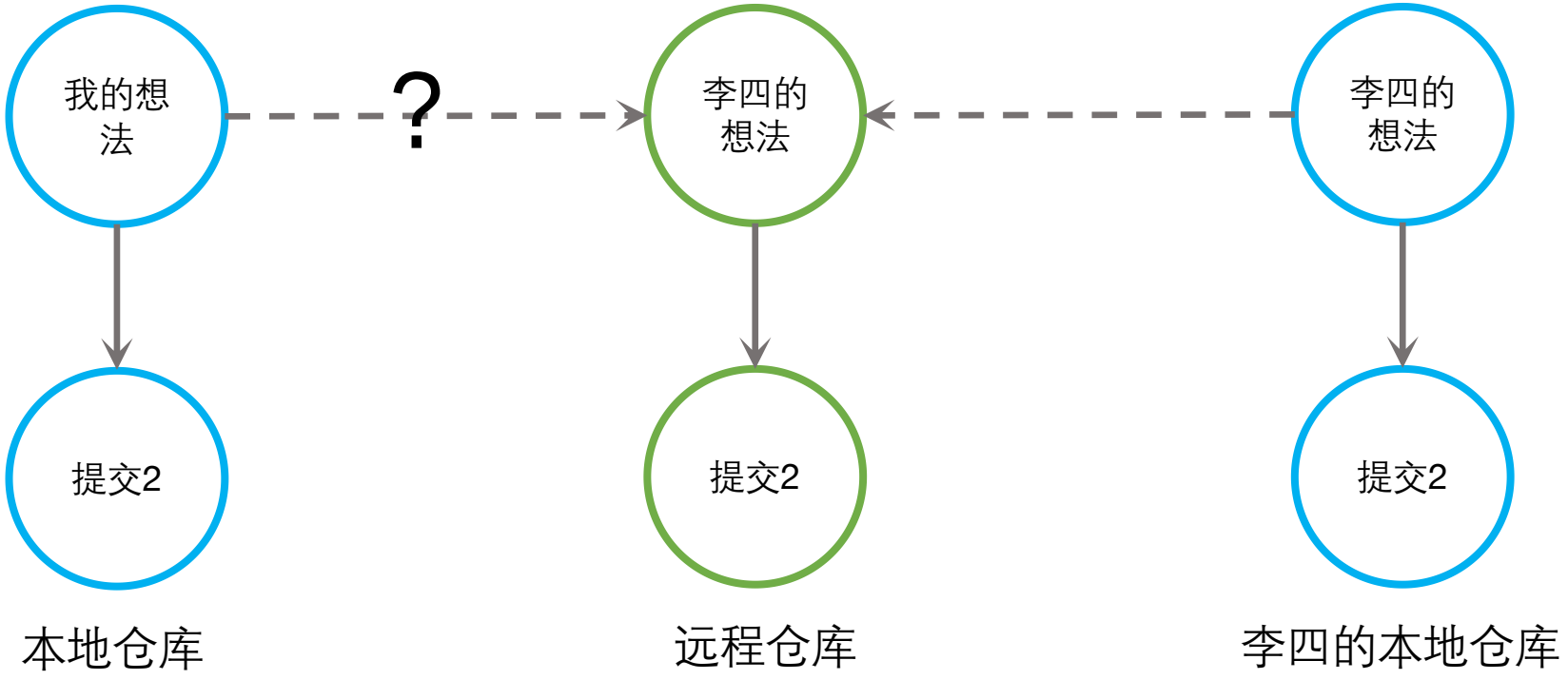
# 放开这个改动，让我来

terminal

```
master$ git status
On branch master
Changes to be committed:
        deleted:    deleted.txt
        modified:   modified.txt
        new file:   new.txt
master$ git reset HEAD -- deleted.txt
master$ git status
On branch master
Changes to be committed:
        modified:   modified.txt
        new file:   new.txt
Changes not staged for commit:
        deleted:    deleted.txt
```



# 这里他也有一个想法



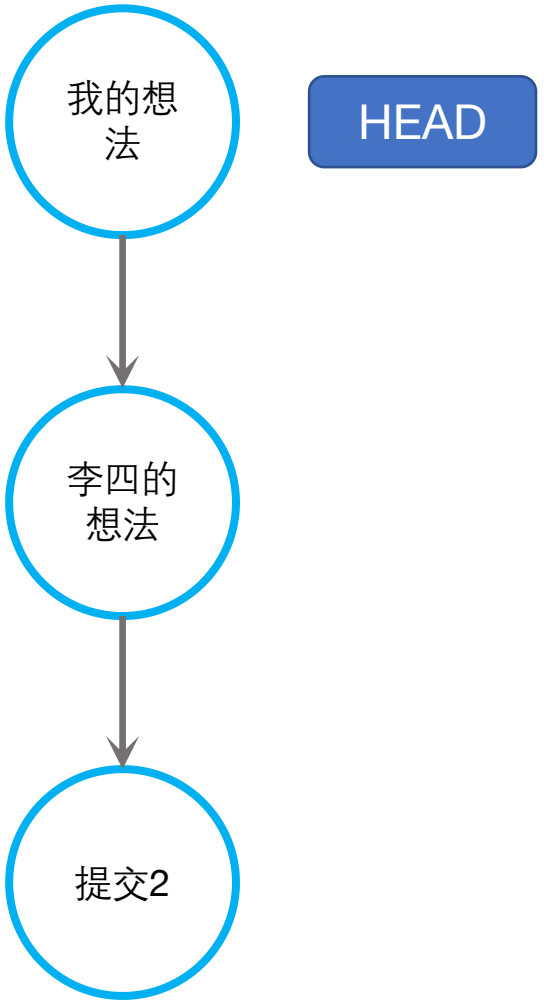
# 这里他也有一个想法

terminal

```
master$ git push
To xxx.git
 ! [rejected]        master -> master (non-fast-forward)
error: failed to push some refs to 'xxx.git'
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. Integrate the remote changes (e.g.
hint: 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

# 不同想法和平共处

```
terminal
master$ git pull --rebase
master$ git rebase --continue
```




本地仓库

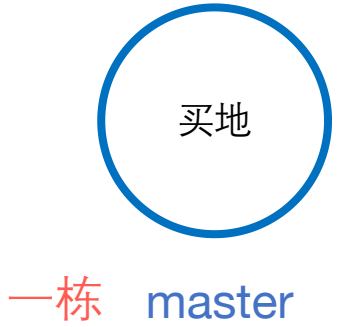
# 实战练习一


Git101 实战练习一.pdf

# 我们的目标是：高楼大厦

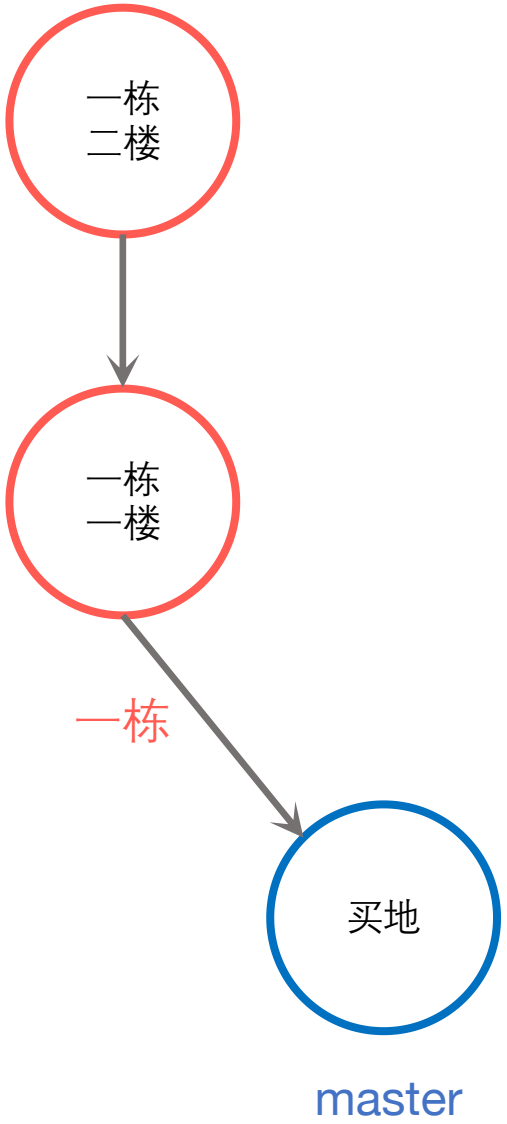
# 一起来搬砖吧

```
terminal
master$ git branch 一栋
master$
```



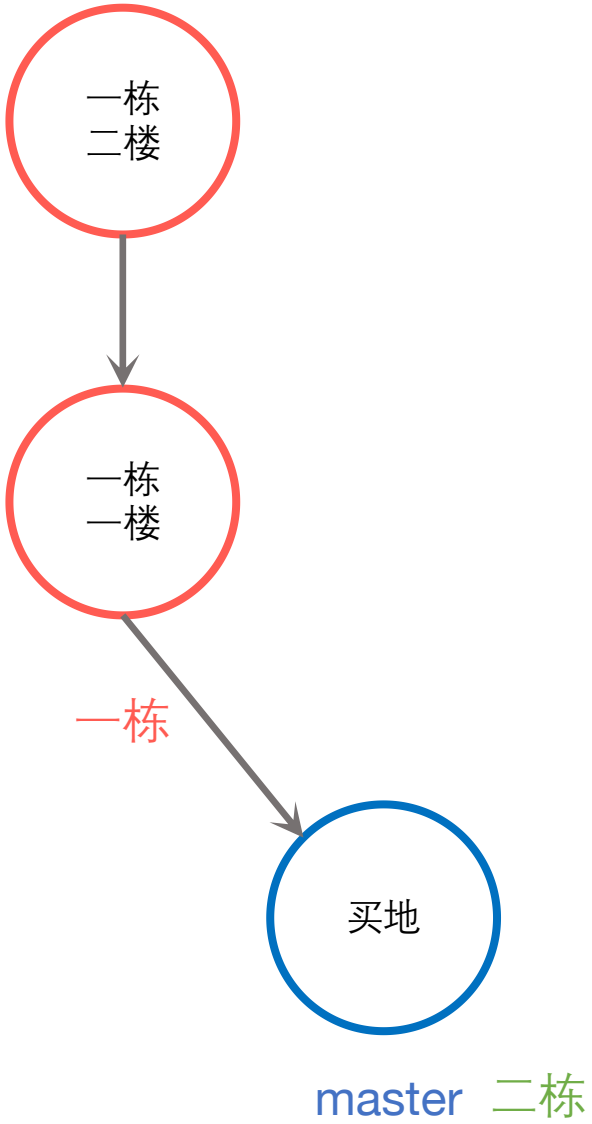
# 一起来搬砖吧

```
terminal
master$ git branch 一栋
master$
```



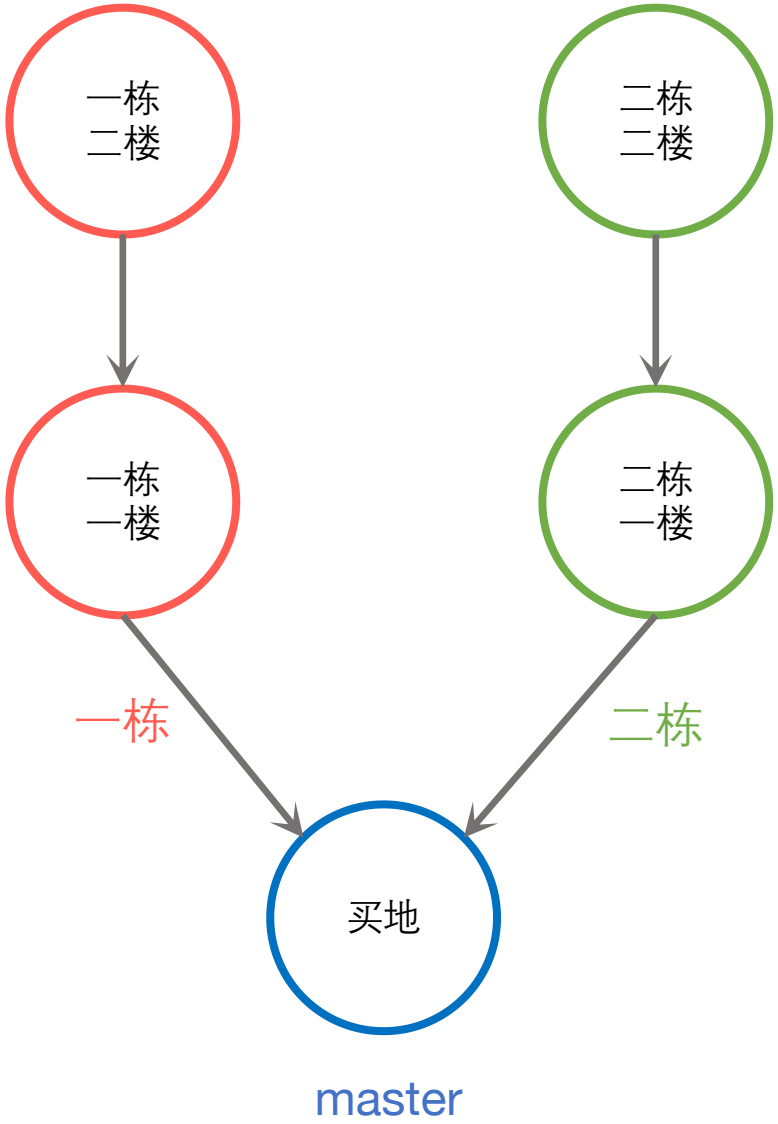
# 一起来搬砖吧

```
terminal
master$ git branch 一栋
master$ git branch 二栋
```



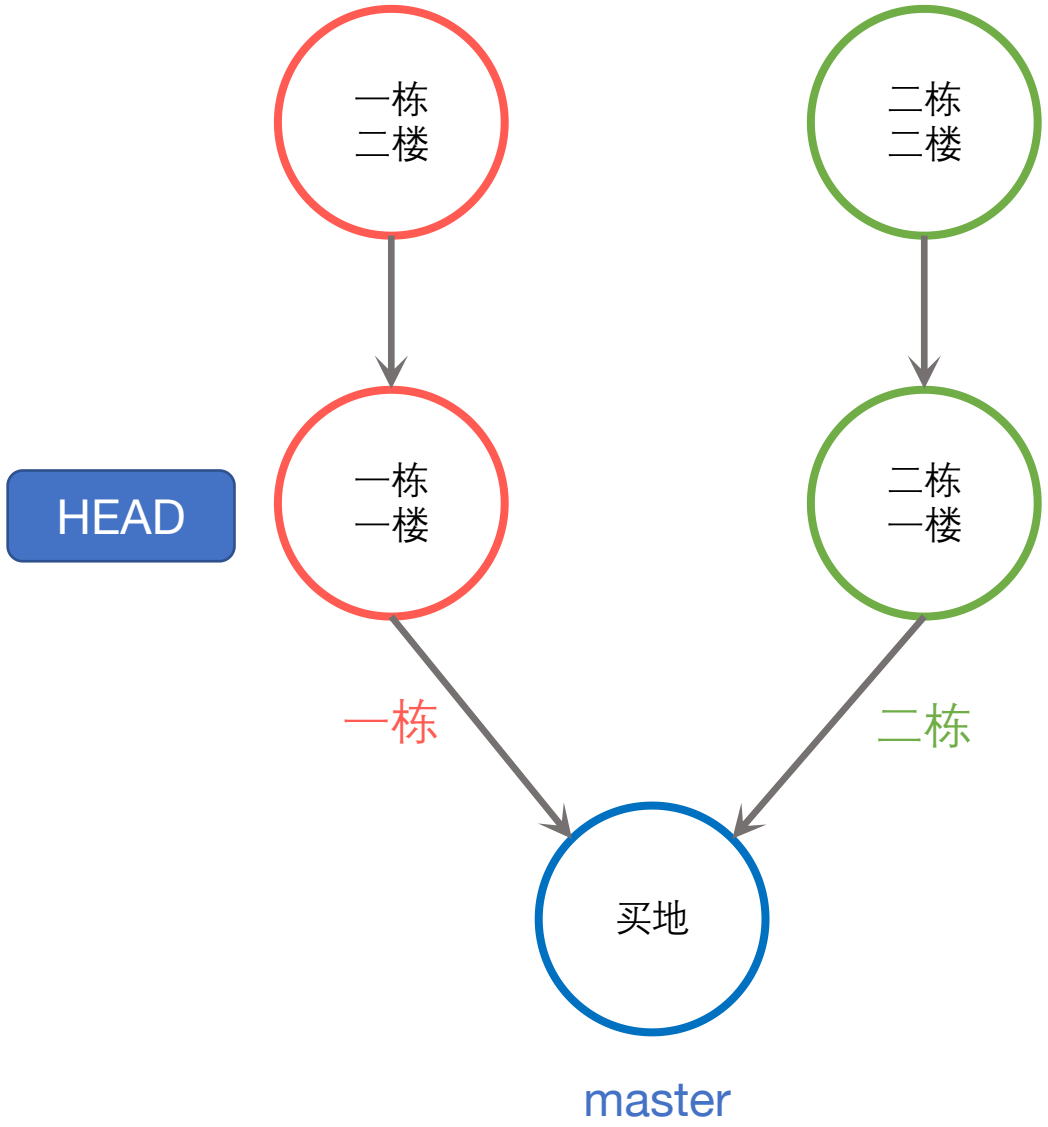
# 一起来搬砖吧

```
terminal
master$ git branch 一栋
master$ git branch 二栋
```



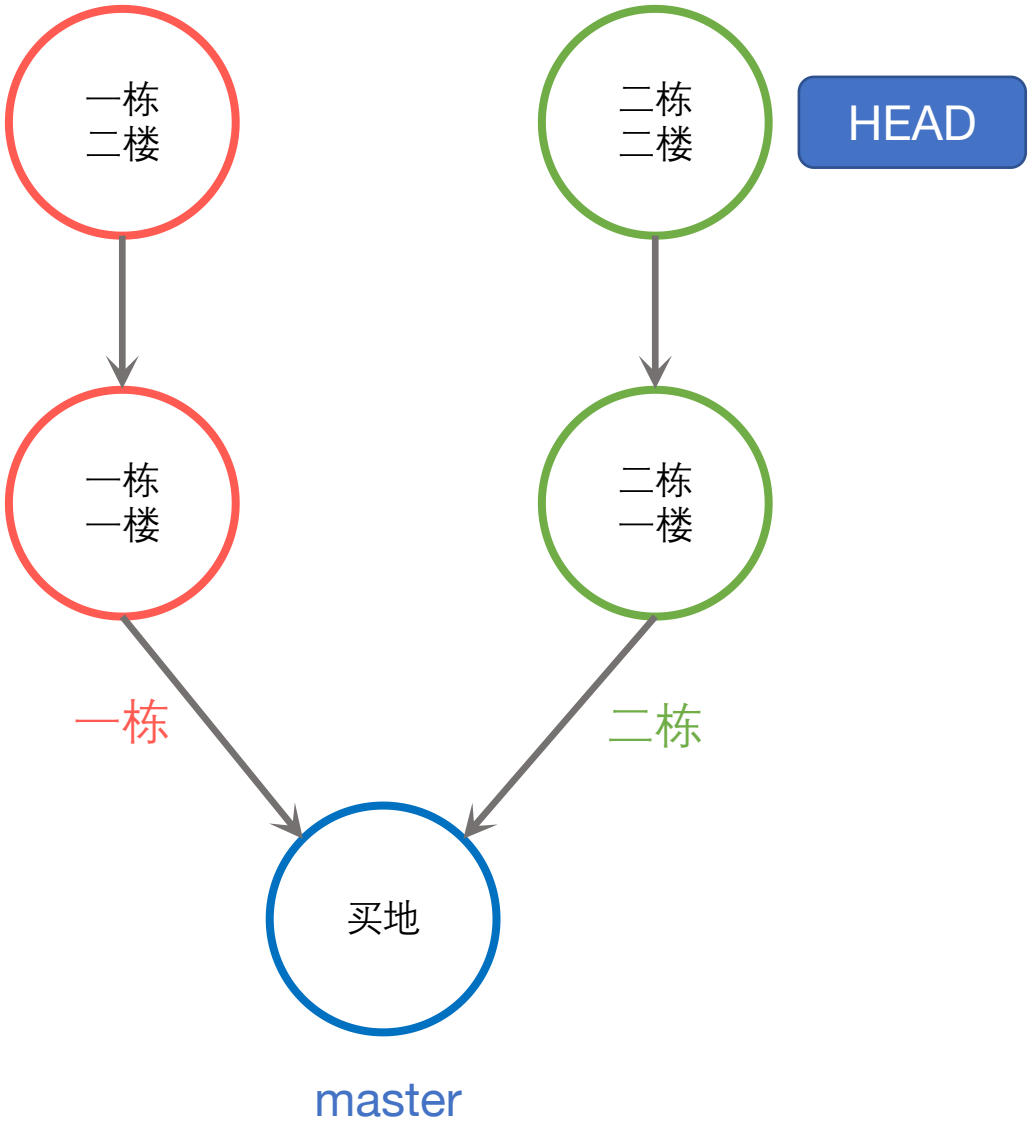
# 帮助监工想去哪就去哪的传送门

```
terminal
master$ git checkout 一栋一楼
master$ git checkout 二栋二楼
```



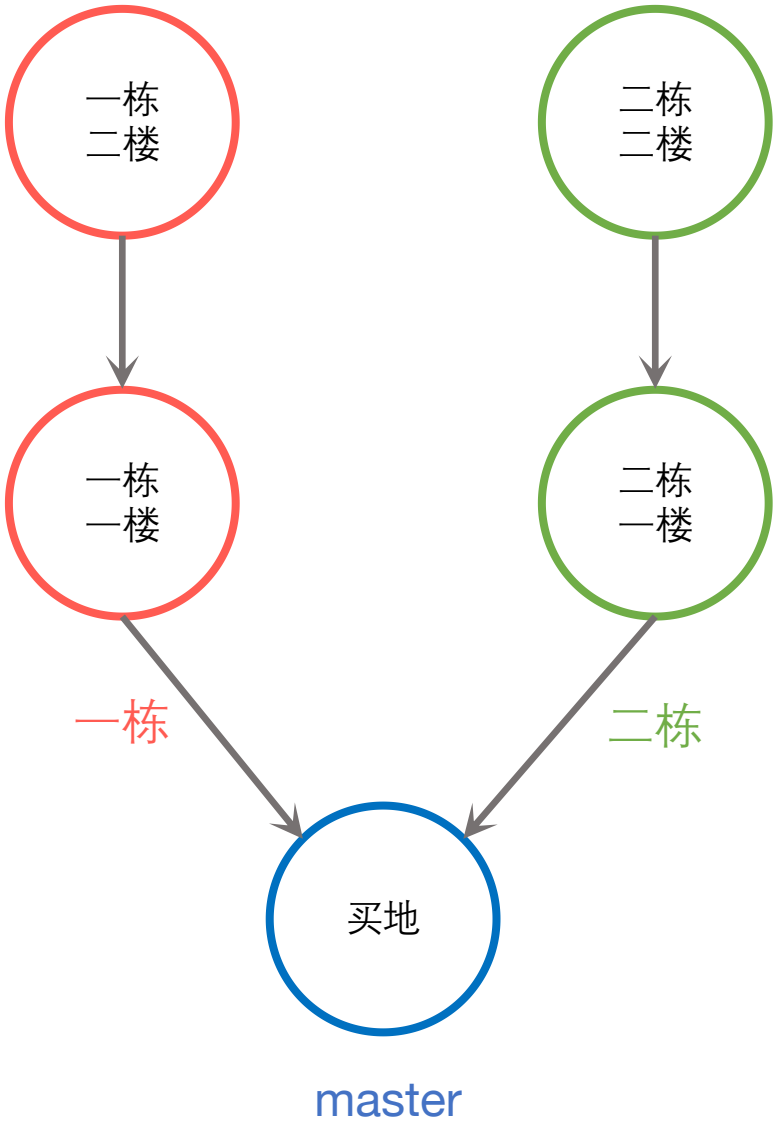
# 帮助监工想去哪就去哪的传送门

```
terminal
master$ git checkout 一栋一楼
master$ git checkout 二栋二楼
```



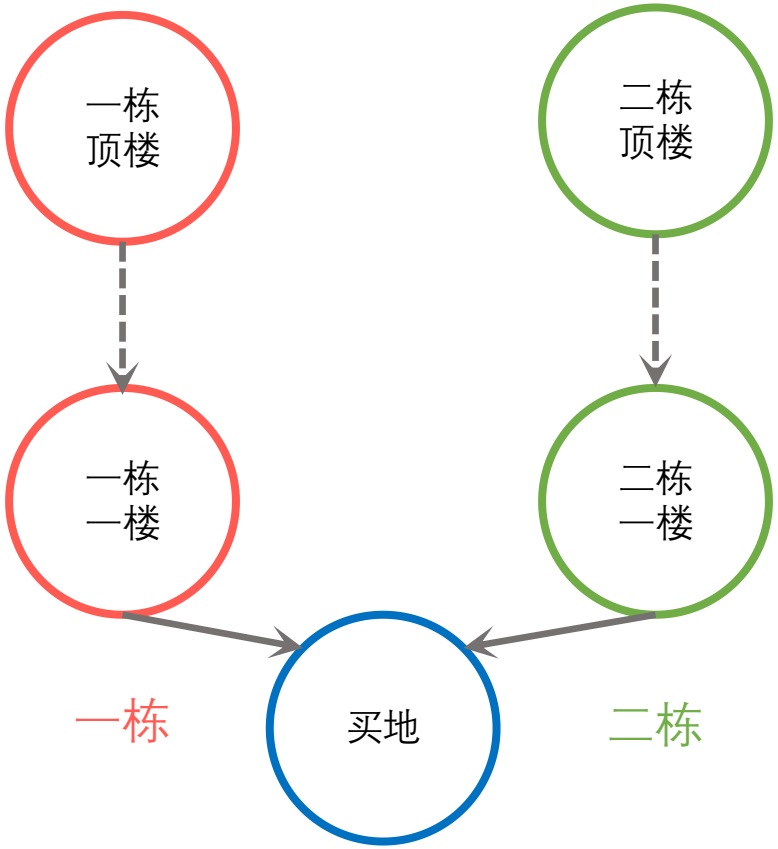
# 准备交房

```
terminal
master$
```



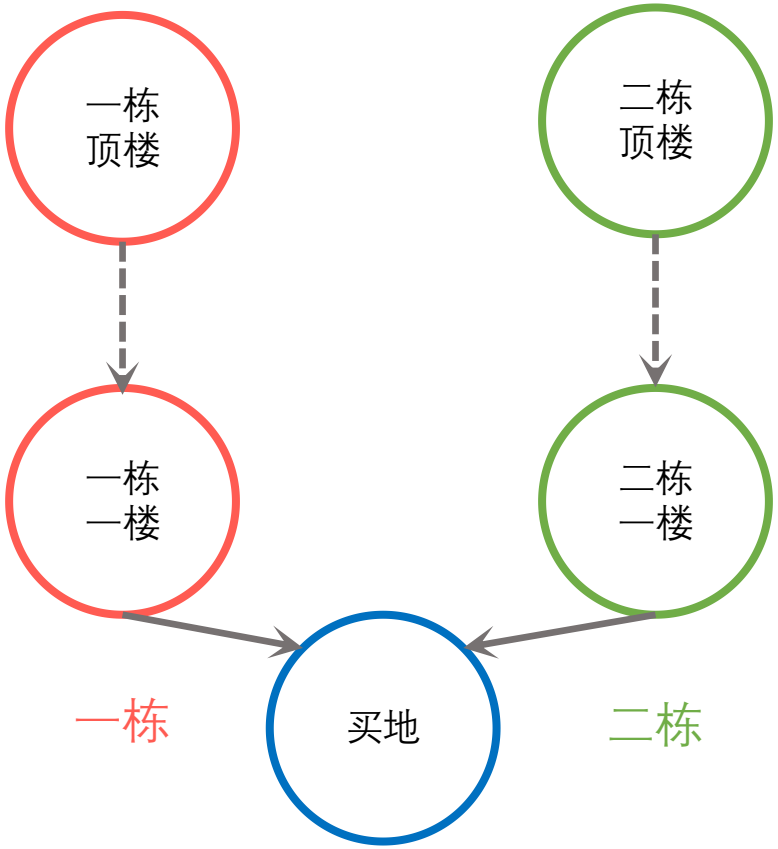
# 准备交房

```
terminal
master$
```



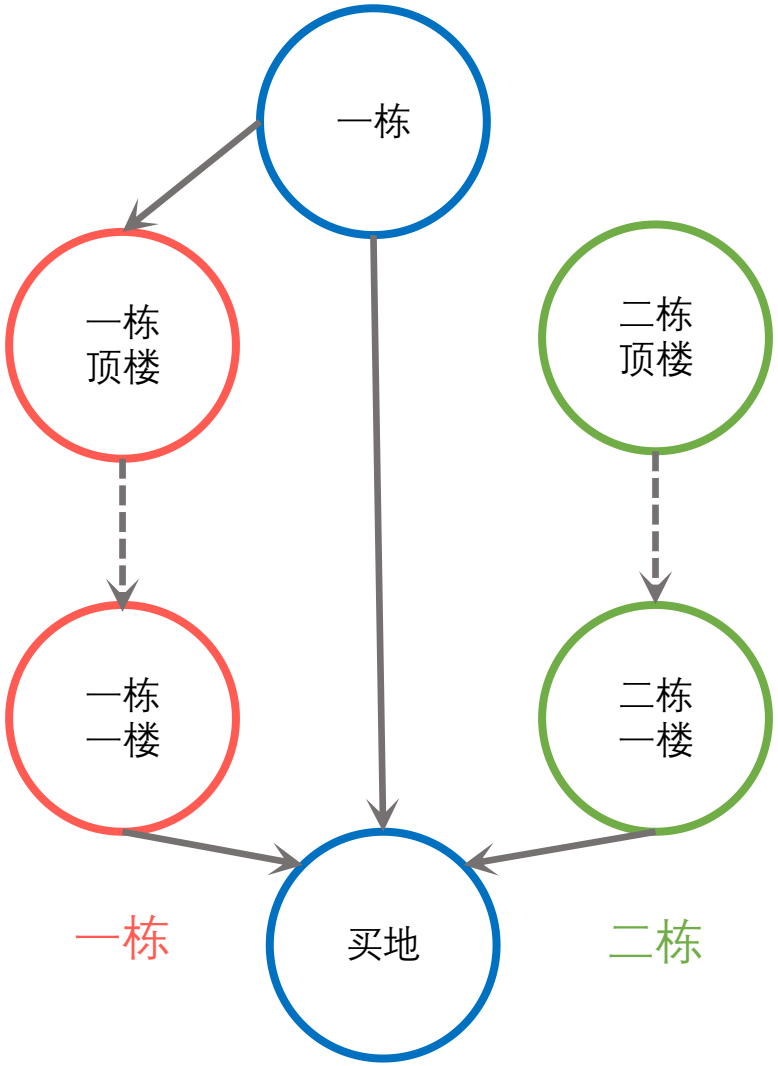
# 准备交房

```
terminal
master$ git merge 一栋
```



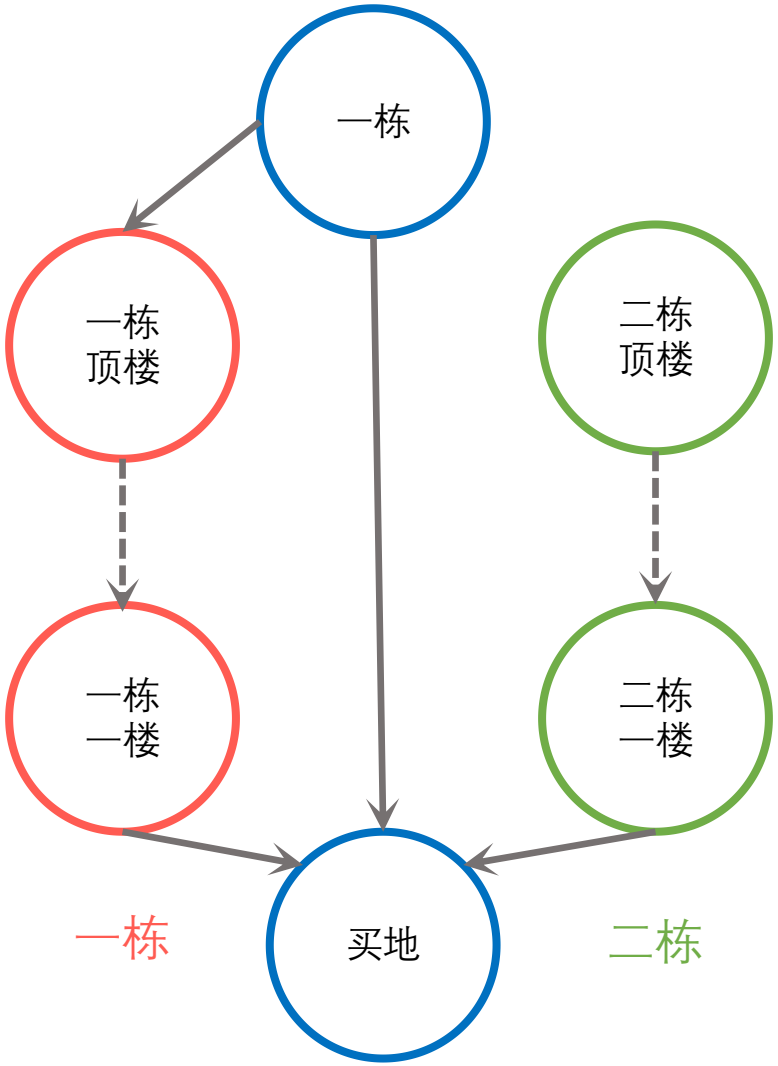
# 准备交房

```
terminal
master$ git merge 一栋
```



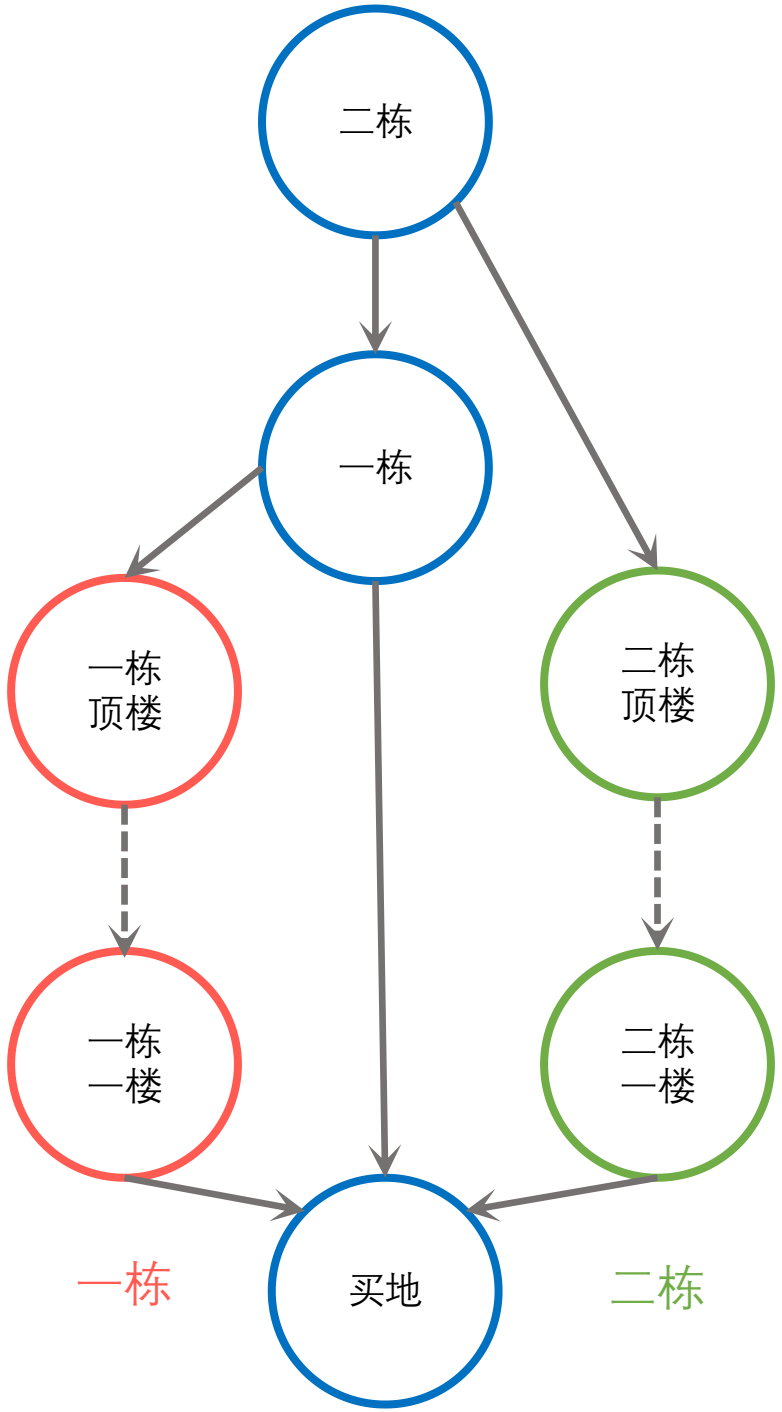
# 准备交房

```
terminal
master$ git merge 一栋
master$ git merge 二栋
```



# 准备交房

```
terminal
master$ git merge 一栋
master$ git merge 二栋
```



# 实战练习二


Git101 实战练习二.pdf

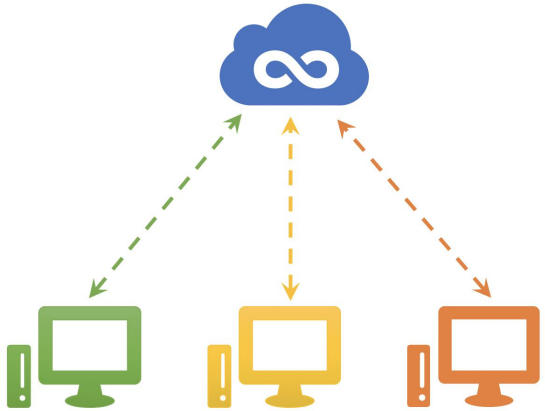
ThoughtWorks®

# 4 课程回顾

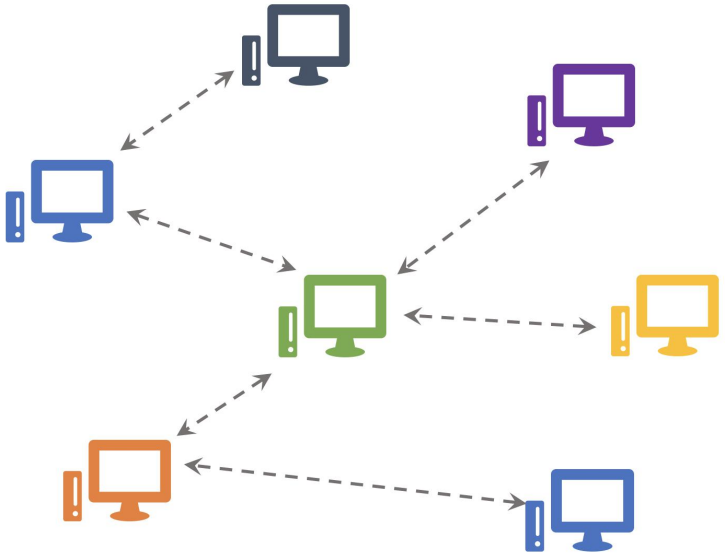
# 课程回顾

## 版本控制

本地式



集中式

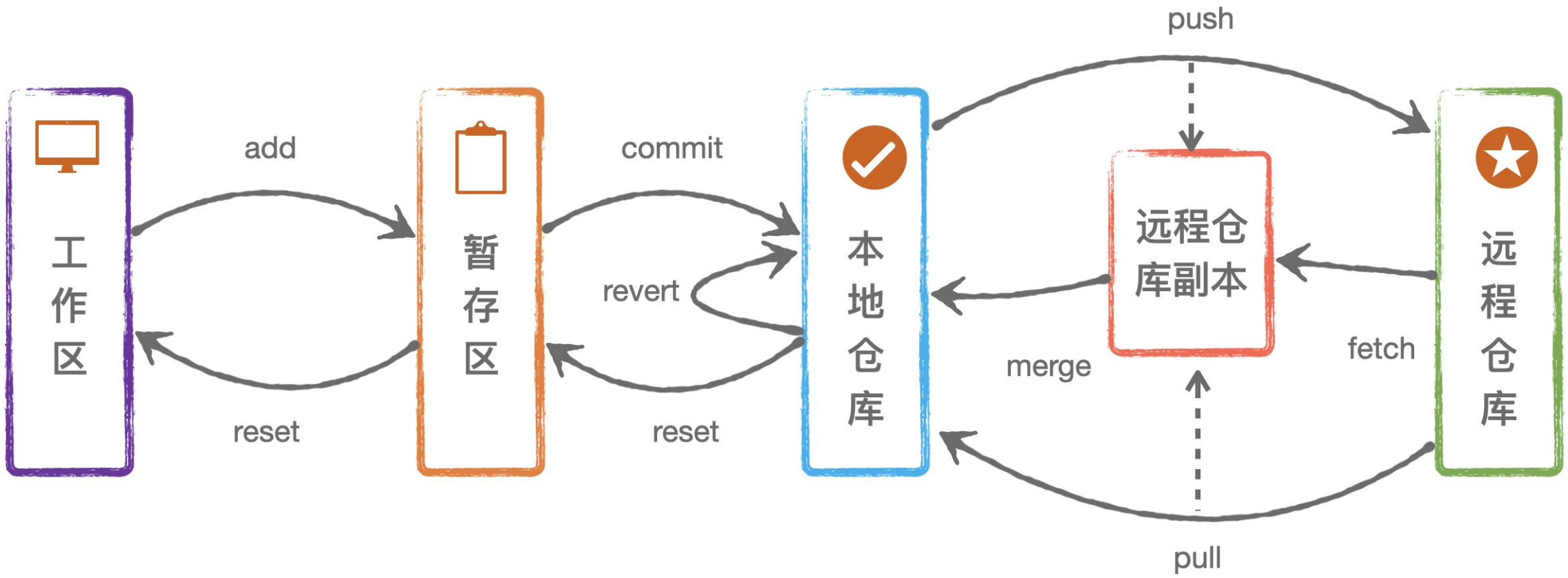


分布式

# 课程回顾

版本控制

# 课程回顾


push
add
commit
工作区
暂存区
本地仓库
远程仓库副本
远程仓库
revert
merge
fetch
reset
reset
pull

# 课程回顾

版本控制 ✓

git ✓

# Talk is cheap, show me code

https://learngitbranching.js.org/?locale=zh_CN

Thanks!!!