

Git 基础

潘魏增@美团网

什么是Git

- 分布式版本控制系统

什么是Git

- 分布式版本控制系统
- 2005年由Linus Torvalds开发

什么是Git

- 分布式版本控制系统
- 2005年由Linus Torvalds开发
- the stupid content tracker

什么是Git

- 分布式版本控制系统
- 2005年由Linus Torvalds开发
- the stupid content tracker



Git的核心竞争力

- 快、快、快

Git的核心竞争力

- 快、快、快
- 分支开发如同家常便饭

Git的核心竞争力

- 快、快、快
- 分支开发如同家常便饭
- 灵活可扩展的工作流

哪些项目在使用Git







为什么要迁移到Git

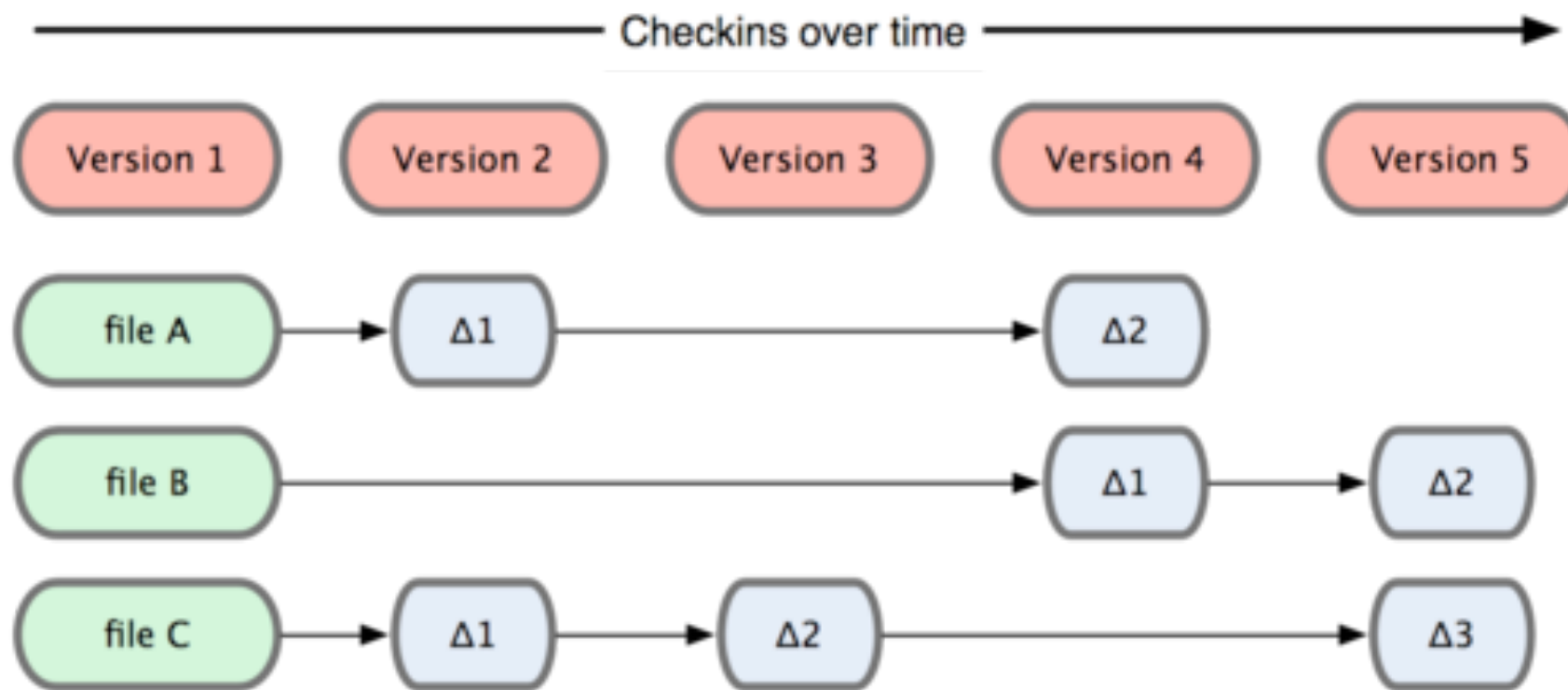
- 团队飞速成长，代码协作带来前所未有的挑战

为什么要迁移到Git

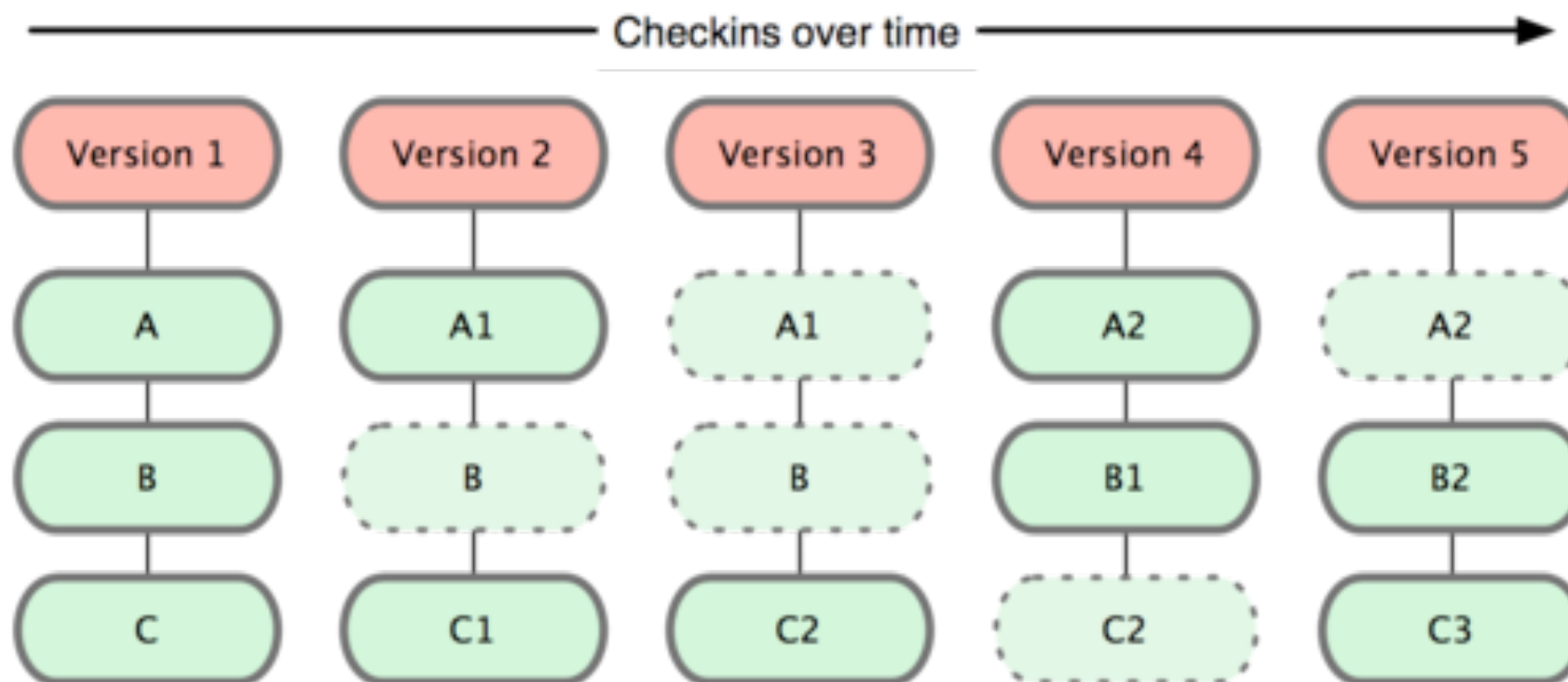
- 团队飞速成长，代码协作带来前所未有的挑战
- 以前的分支开发就像一次分娩

基本原理

其他系统记录文件差异

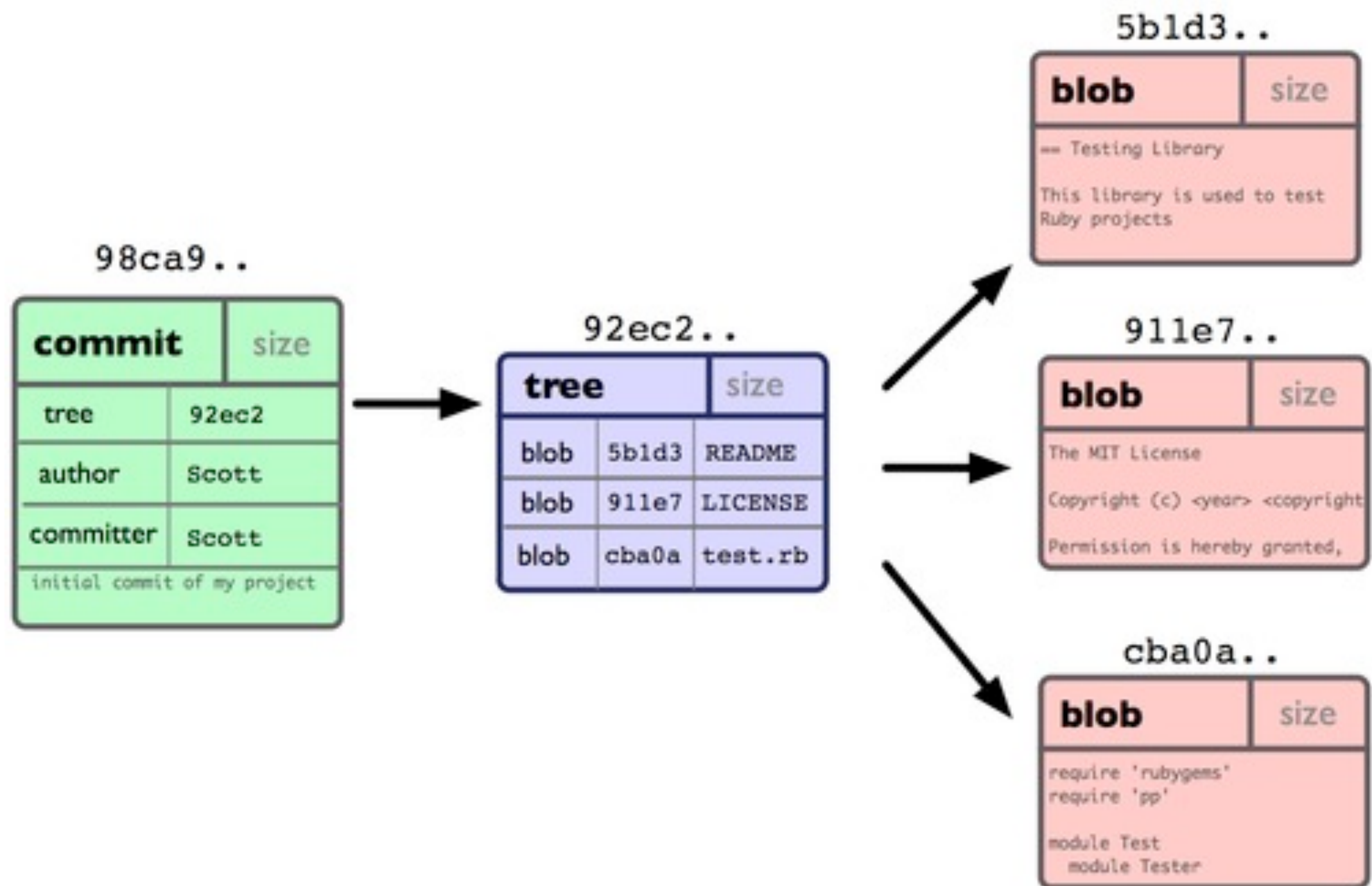


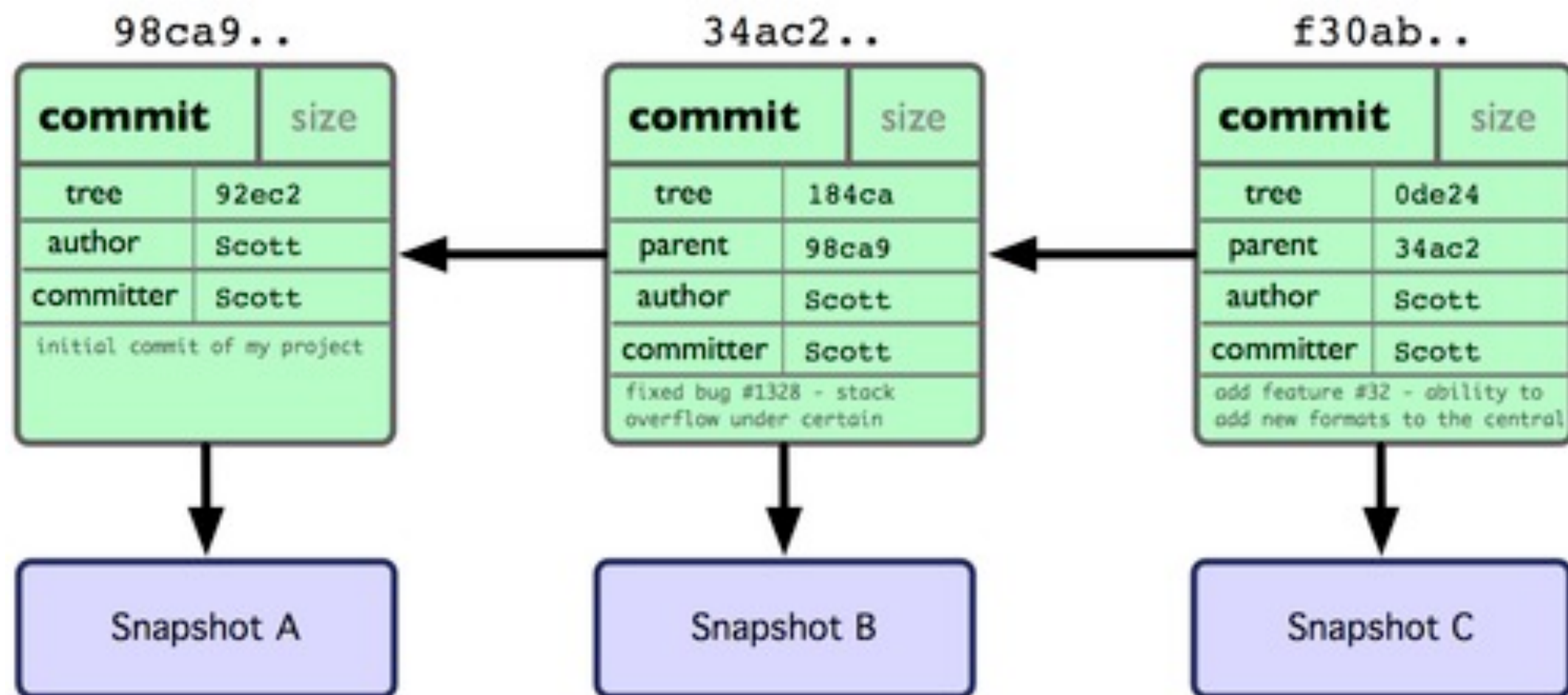
Git 保存文件快照



四种对象

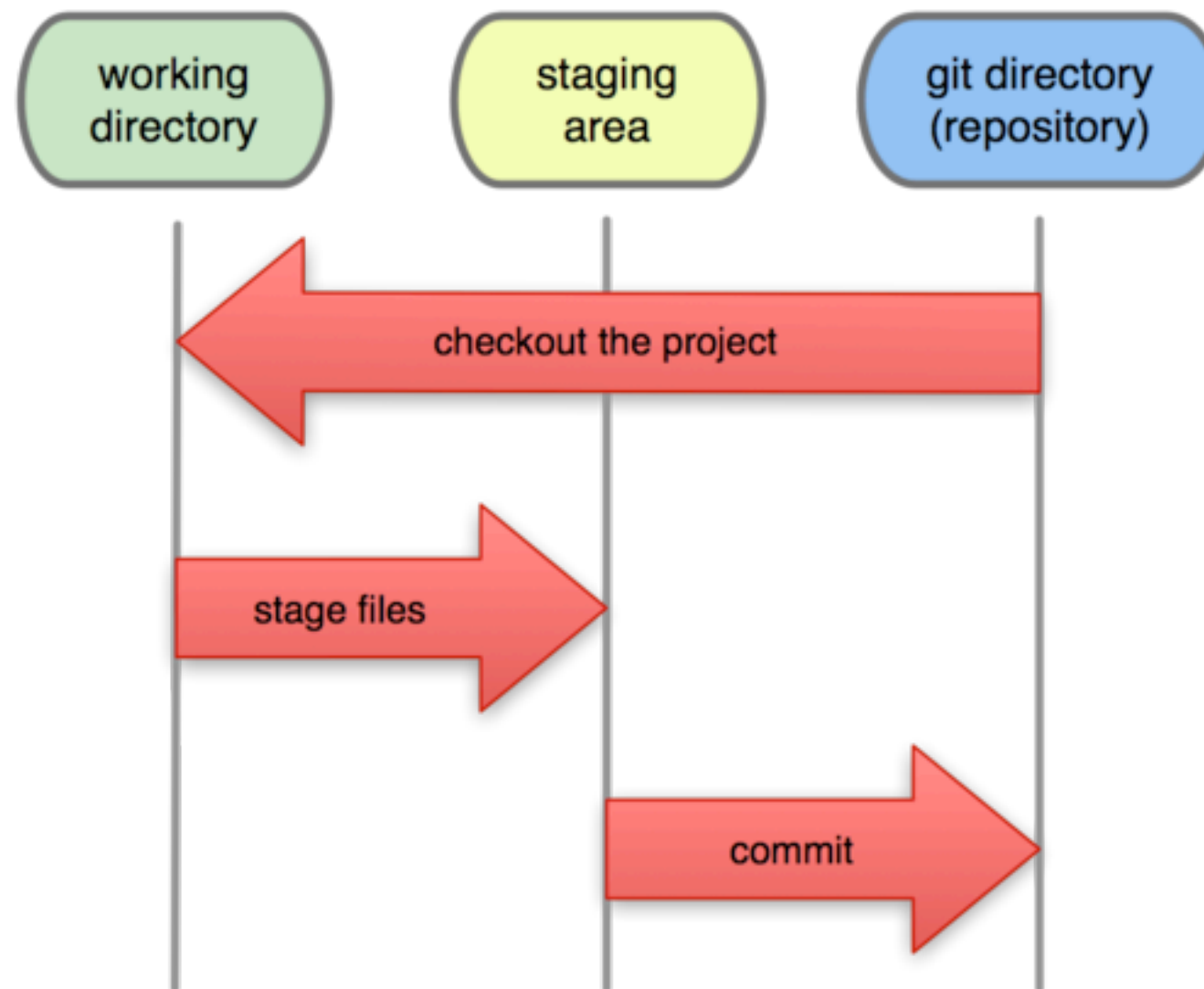
- blob - 文件内容，没有属性信息
- tree - 所有文件名字以及其属性的列表
- commit - 表示修改历史，描述一个个 tree 之间如何联系起来
- tag - 标签，它可以指向 blob, tree 和 commit，大部分时候是 commit





三种状态

Local Operations



基本操作

获取代码库

- `git clone git@cat.sankuai.com:www`
- `git clone git@cat.sankuai.com:www mt`

最初设置

- `$git config --global user.name "panweizeng"`
- `$git config --global user.email panweizeng@sankuai.com`

提交代码到本地仓库

- `git add template/deal/default.php`
- `git commit -m "update deal ui"`

推送代码到远程仓库

- `git push`

推送代码到远程仓库

- `git push`
- `git push origin master`

检查代码状态

- `git status`

检查代码状态

- `git status`
- `git status -s`

文件操作

- `git add`
- `git reset HEAD`
- `git checkout`
- `git rm`
- `git mv`

查看差异

- `git diff`
- `git diff --cached`
- `git diff HEAD`
- `git diff origin`

查看提交历史

- `git log`
- `git log -5`
- `git log -p`
- `git log -p --author=panweizeng`
- `git log --since=2011-05-24`
- `git log --until=2011-05-25`
- `git log --name-only`

查看提交历史

- `git log --pretty=oneline`
- `git log --pretty=format:%h:%s`
- `git log --graph`

查看提交历史

- `git log --pretty=oneline`
- `git log --pretty=format:%h:%s`
- `git log --graph`
- fisheye - <http://task.sankuai.com/code/>

更新本地仓库

- `git pull`

更新本地仓库

- `git pull`
- `git fetch && git merge origin/master`

查看帮助信息

- git help command

git help commit

git help push

git help log

分支开发

- Git之杀手锏

创建分支

- `git branch hotfix`
- `git checkout -b hotfix`
- `git checkout -b hotfix master`

切换分支

- `git checkout hotfix`
- `git checkout master`

分支合并

- `git merge hotfix`
- `git merge origin/master`
- `git merge --squash`

分支管理

- `git branch -v`
- `git branch -d hotfix`
- `git branch -D hotfix`
- `git branch --merged`
- `git branch --no-merged`

远程分支

- `git push origin abc` #将本地分支保存到远程
- `git push origin :abc` #将远程分支删除
- `git branch -a` #查看目录下的所有分支
- `git push origin panweizeng/abc`
- `git push origin :panweizeng/abc`

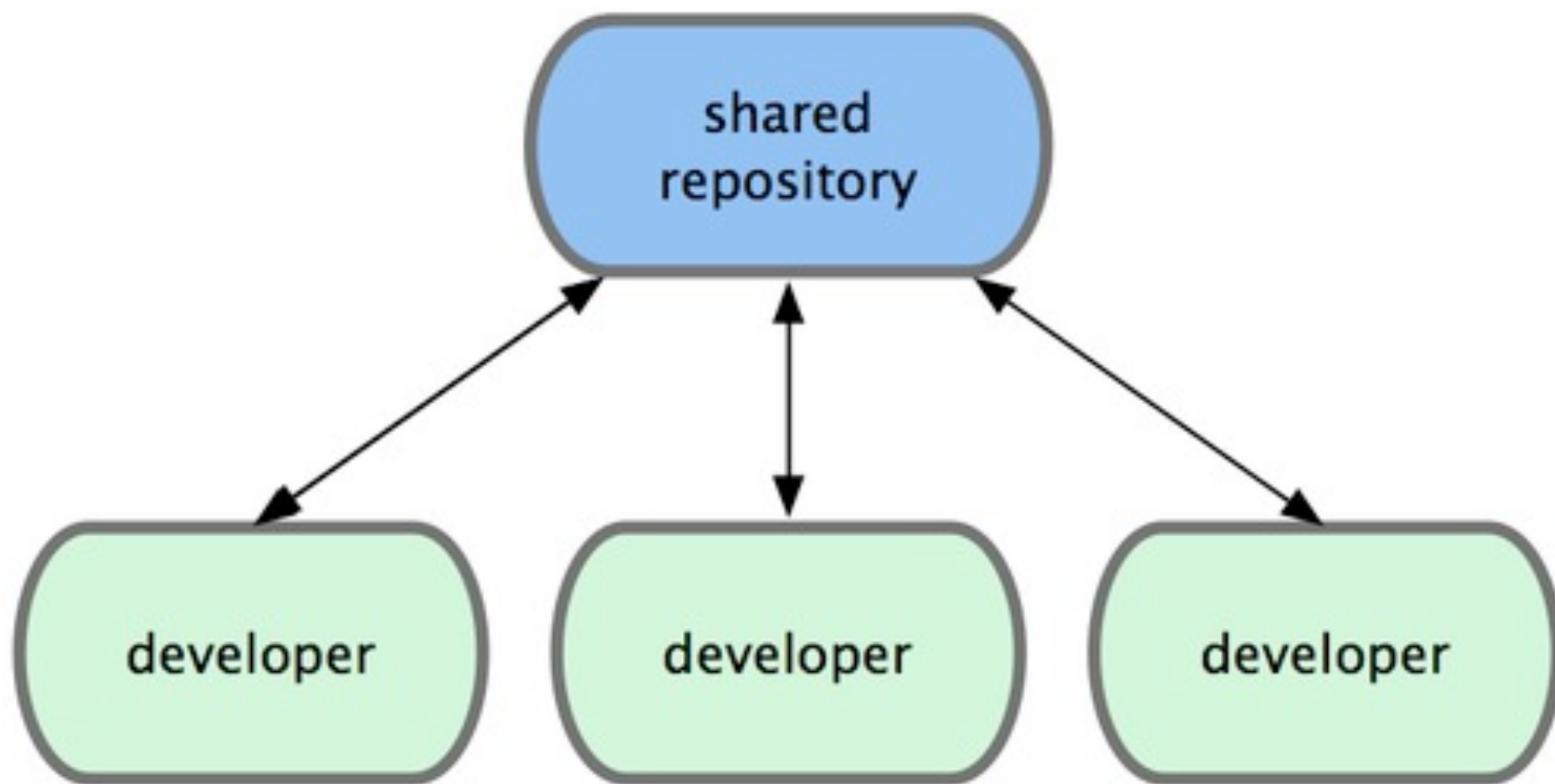
美团工作流

- 快、快、快

美团工作流

- 快、快、快
- 快速开发，快速部署，快速迭代

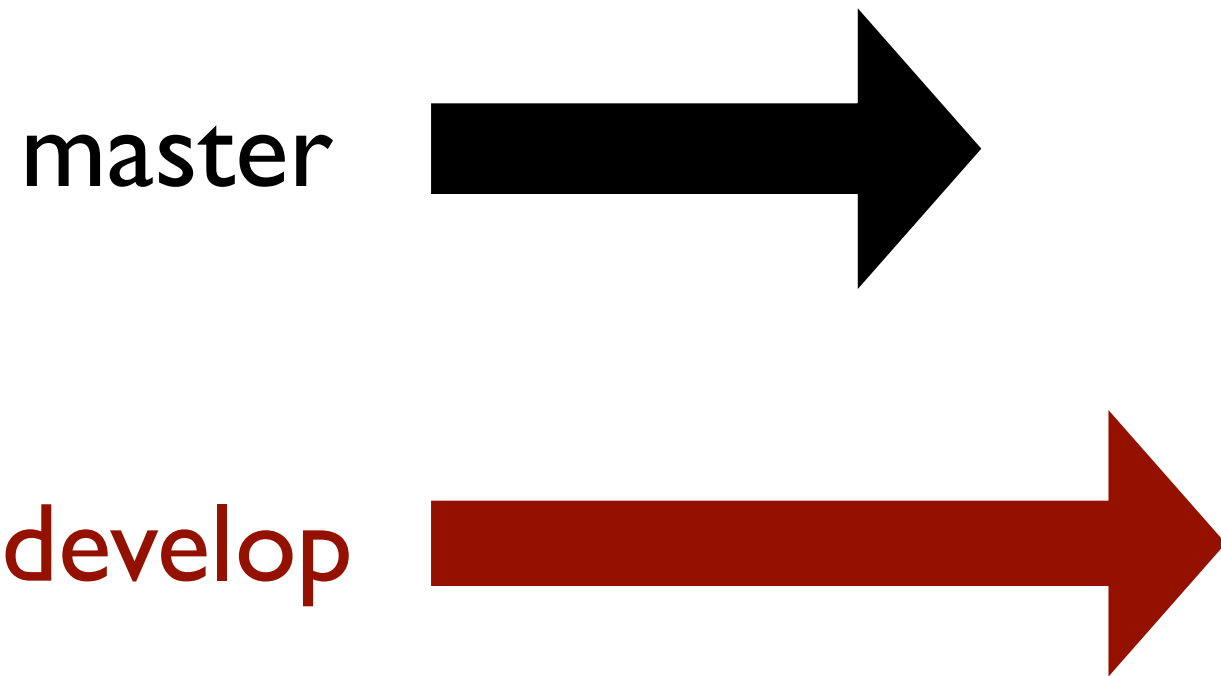
协作开发模型



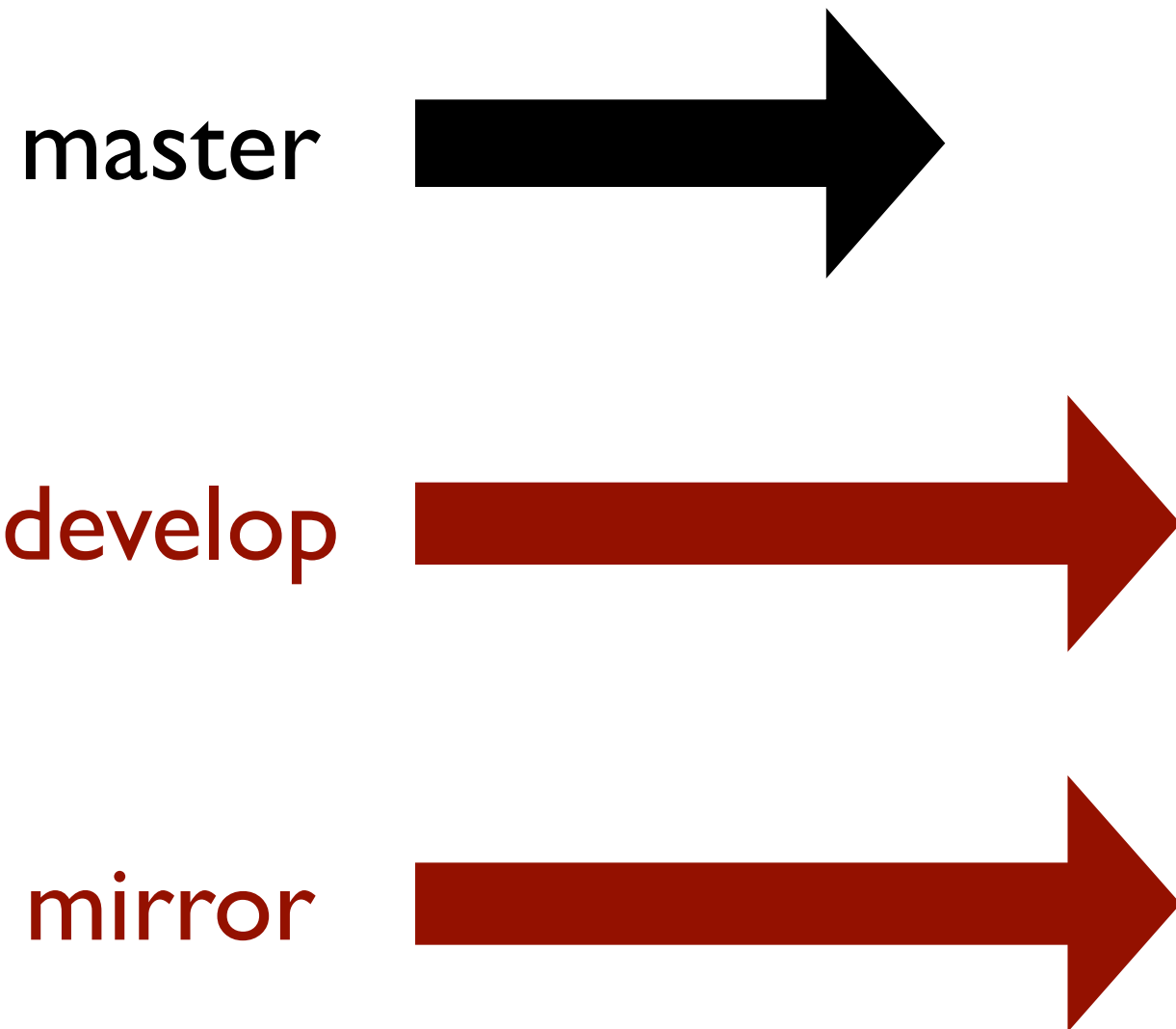
更快节奏的开发 workflow



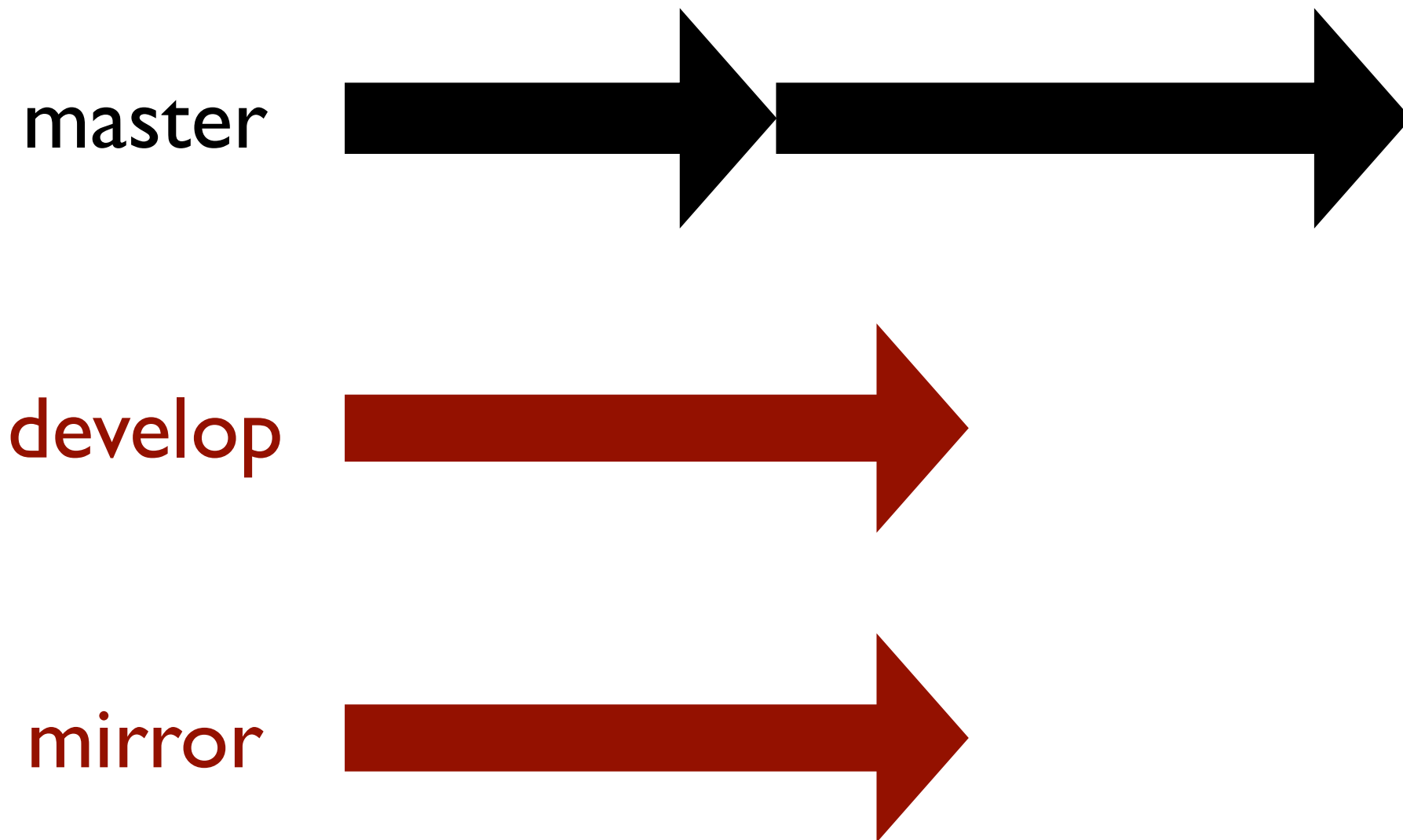
更快节奏的开发 workflow



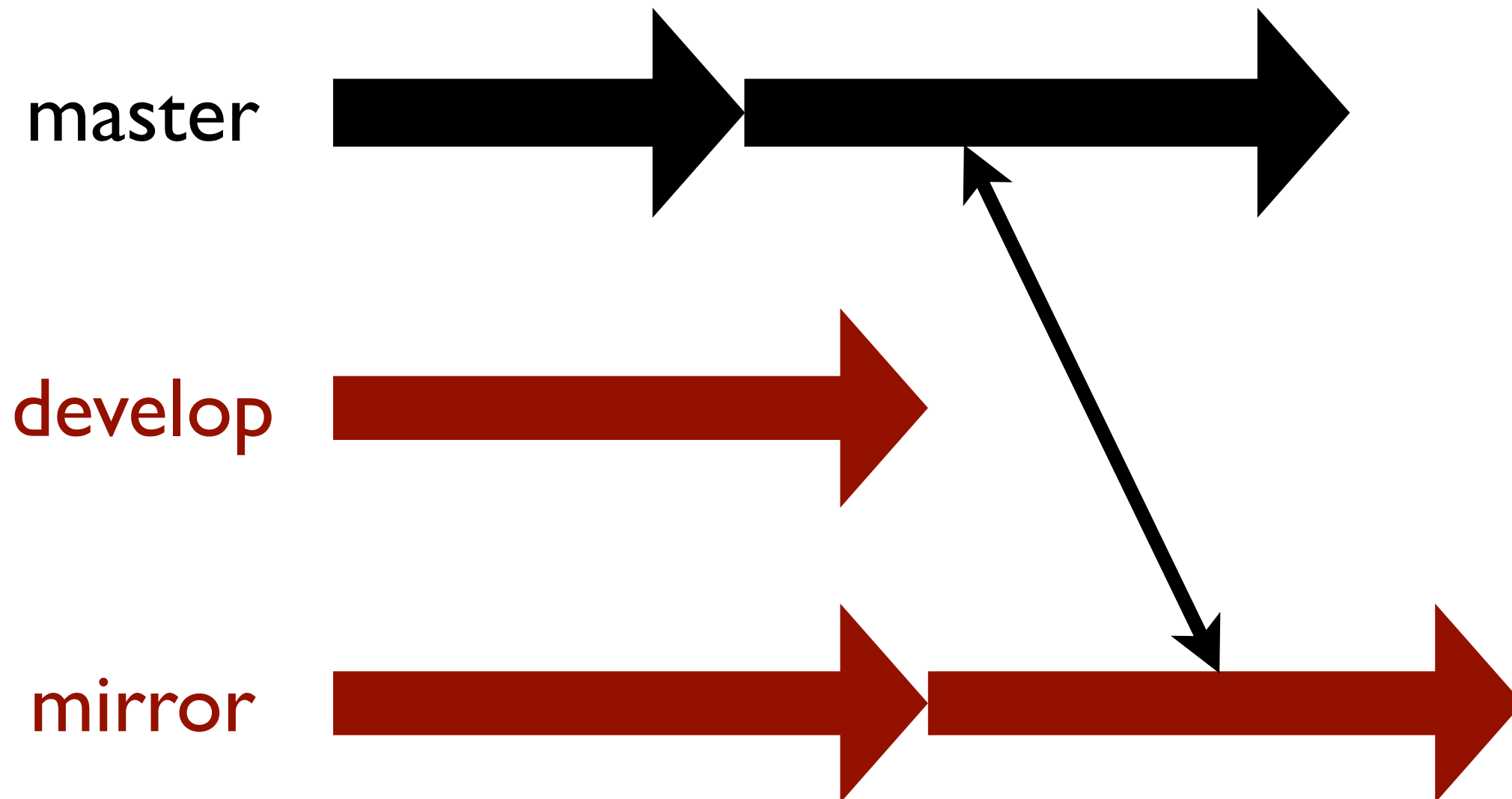
更快节奏的开发 workflow



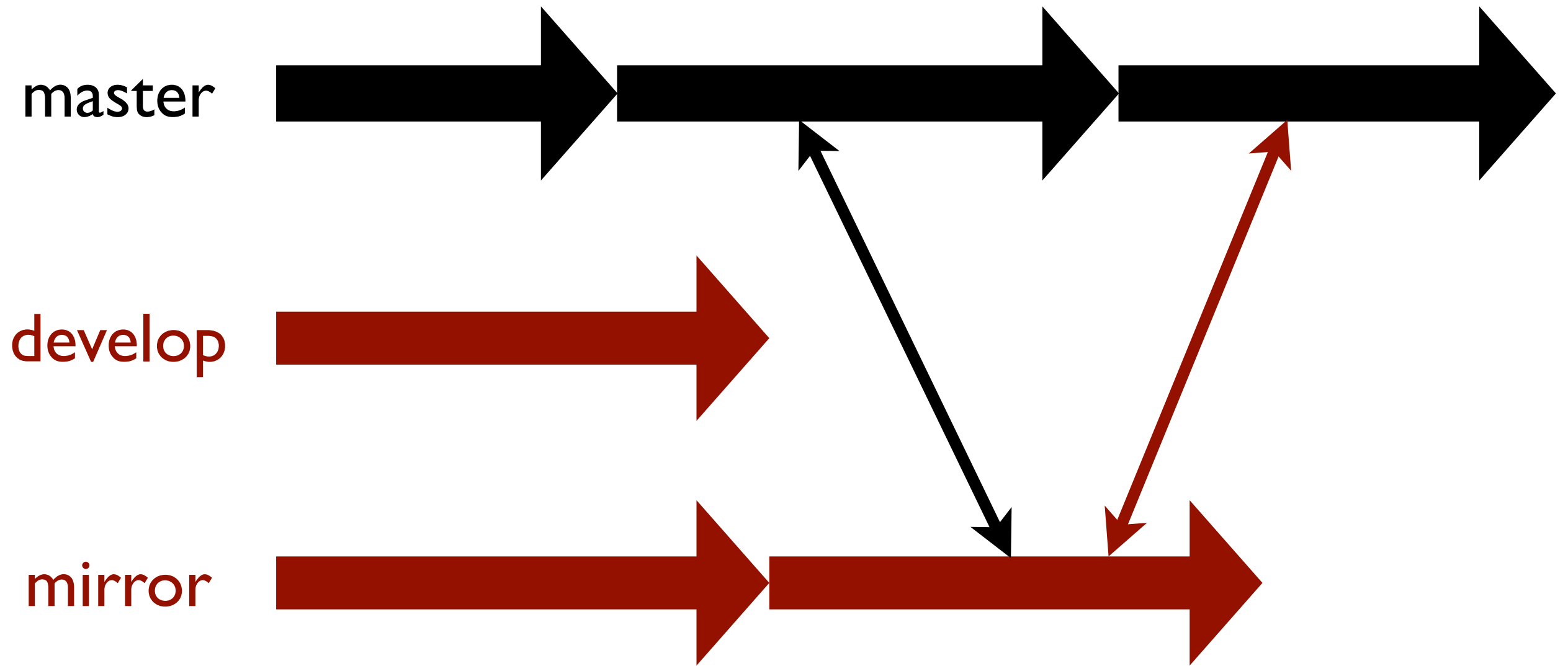
更快节奏的开发 workflow



更快节奏的开发 workflow



更快节奏的开发 workflow







master是一座桥梁

杂项

- 一些改变生活的小技巧

常用设置

- `git config --global color.ui auto`
- `git config --global alias.st status`
- `git config core.filemode false`
- `git config -l`
- `/etc/gitconfig`
`~/.gitconfig`
`.git/config`

简短的HASH

- `git show c2bd66ee2f5ebef7dac171fca57875bd99f47318`
- `git show c2bd66ee2f5ebef7`
- `git show c2bd66ee`
- `git show c2bd`

储藏代码

- `git stash`
- `git stash apply`
- `git stash list`
- `git stash branch abc`

快速查找代码树

- `git grep keyword`
- `git grep keyword -i -A10 -B10`

查看代码是谁修改的

- `git blame filename`

修改最后一次提交

- `git commit -m “new mobile”`
- `git add forgotten_file`
- `git commit --amend`

修改本地某次提交

- `git rebase -i 32cb53^`
- `git commit --amend`
- `git rebase --continue`

修改本地某次提交

- `git rebase -i 32cb53^`
- `git commit --amend`
- `git rebase --continue`

警告： 不要修改已经push到远程仓库的提交

将diff复写到相关文件

- `git diff > ~/diff` # 备份代码到diff文件
- `git apply ~/diff` # 将diff应用到当前代码树
- `patch -p1 < ~/diff` # 如果没有安装git

命令行提示当前分支

```
[panweizeng@dev:mt(master)]$ git checkout develop
```

```
Switched to branch 'develop'
```

```
[panweizeng@dev:mt(develop)]$ git checkout mis
```

```
Switched to branch 'mis'
```

```
[panweizeng@dev:mt(mis)]$
```

命令行提示当前分支

在 ~/.bashrc 中添加

```
function parse_git_branch {  
    git branch --no-color 2> /dev/null | sed -e '/^[^*]/d' -e  
's/* \(.*\)/(\1)/' -e 's/((/(/' -e 's/)))/)/'  
}
```

```
function prom1 {  
    local GREEN="\033[0;32m\  
    local COLOR_END="\033[0m"  
    PS1="[ \u@\h:\w$GREEN\$(parse_git_branch)$COLOR_END]\$ "  
}
```

prom1

命令行自动补全

```
$cd ~/bin
```

```
$curl -O https://github.com/git/git/raw/master/  
contrib/completion/git-completion.bash
```

```
$curl -O https://github.com/bobthecow/git-flow-  
completion/raw/master/git-flow-completion.bash
```

编辑~/.bashrc增加两行

```
source ~/bin/git-completion.bash
```

```
source ~/bin/git-flow-completion.bash
```

命令行自动补全

```
$cd ~/bin
```

```
$curl -O https://github.com/git/git/raw/master/  
contrib/completion/git-completion.bash
```

```
$curl -O https://github.com/bobthecow/git-flow-  
completion/raw/master/git-flow-completion.bash
```

编辑~/.bashrc增加两行

```
source ~/bin/git-completion.bash
```

```
source ~/bin/git-flow-completion.bash
```

也可以放到/etc/bash_completion.d/并编辑/etc/profile

```
for i in /etc/bash_completion.d/* ; do
```

```
    if [ -r "$i" ]; then
```

```
        . $i
```

```
    fi
```

```
done
```

如何学习

如何学习

- 多查，手册是你的朋友

如何学习

- 多查，手册是你的朋友
- 多问，同事是你的朋友

如何学习

- 多查，手册是你的朋友
- 多问，同事是你的朋友
- 多练，git也是你的朋友

The End

- <http://progit.org/book/zh>

The End

- <http://progit.org/book/zh>

