目录

[第一讲 课程内容概述与准备 3](#_Toc36108759)

[第四讲 Git重要命令详解 4](#_Toc36108760)

[初始化仓库： 4](#_Toc36108761)

[工作区操作： 4](#_Toc36108762)

[设置用户名和用户邮箱： 4](#_Toc36108763)

[注： 4](#_Toc36108764)

[第五讲 Git添加，删除，修改与日志 5](#_Toc36108765)

[添加： 5](#_Toc36108766)

[删除： 5](#_Toc36108767)

[恢复删除： 5](#_Toc36108768)

[修改文件名： 5](#_Toc36108769)

[修改提交注释： 5](#_Toc36108770)

[提交日志： 6](#_Toc36108771)

[help： 6](#_Toc36108772)

[第六讲 .gitignore与分支 7](#_Toc36108773)

[.gitignore: 7](#_Toc36108774)

[分支： 7](#_Toc36108775)

[第七讲 分支重要操作 8](#_Toc36108776)

[分支基础操作： 8](#_Toc36108777)

[分支合并操作： 8](#_Toc36108778)

[第八讲 分支进阶与版本回退 9](#_Toc36108779)

[分支进阶： 9](#_Toc36108780)

[版本回退： 10](#_Toc36108781)

[提交历史和操作历史： 10](#_Toc36108782)

[第九讲 checkout进阶与stash 11](#_Toc36108783)

[checkout-reset命令进阶： 11](#_Toc36108784)

[stash命令： 11](#_Toc36108785)

[重命名branch： 11](#_Toc36108786)

[第十讲 标签与diff 12](#_Toc36108787)

[标签： 12](#_Toc36108788)

[diff： 12](#_Toc36108789)

第一讲 课程内容概述与准备

ls 显示当前目录下的文件

ls -al 以列表形式显示当前目录下文件

cd 进入当前目录下的文件夹

cd – 返回上一个路径

vi 编辑文件

cp 拷贝文件

rm 删除文件

rm -rf 删除文件夹

mv 重命名

cat 显示文件内容

clear 清除屏幕信息

ctrl + l 快速清屏

get –version 获取版本号

pwd 获取当前路径

mkdir 建立文件夹

在编辑器中：

:wq保存并退出

:set number 显示行号

:数字 跳转到指定行数

新建文件：

touch ‘text.text’ 创建一个新的文件，无其他动作

vi ‘text.text’ 编辑文件，如果没有就直接新建此文件名的一个文件

each ‘words’ > ‘text.text’ 向一个文件中以覆盖的形式写入数据

第四讲 Git重要命令详解

初始化仓库：

**git init** 初始化一个空的仓库

工作区操作：

**git status** 查看工作区的状态

**git add ‘text.text’** 将在工作区依旧修改的文件加入暂存区中

**git add .** 将工作区中所有更改的文件加入暂存区

**git commit -m ‘commit message’** 将在暂存区的文件提交到版本库

**git commit -am ‘commit message’** 将在工作区的文件直接提交到版本库中

//**git -rm –catch ‘text.text’** 将此文件从暂存区删除，还原到工作区

**git reset HEAD ‘text.text’** 将此文件的变更从暂存区删除，还原到工作区

**git checkout -- ‘text.text’** 将工作区中的文件还原成未更改的状态

设置用户名和用户邮箱：

users.name && users.email：

1. /etc/gitconfig(几乎不会被使用) **git config --system**
2. ~/.gitconfig(针对用户) **git config --global**
3. .git/config(针对项目) **git config –local**

注：

优先级3>2>1

注：

在命令行中按**上箭头下箭头**可以直接使用之前使用过的命令

在命令行中按**Tab**可以获得当前未打完的命令的提示信息

在命令行中输入**pwd**可以获取当前目录位置

在vi编辑器中按**shift + a**可以直接切换到行尾并进入写入状态

在命令行中**ctrl + a**可以跳到行首

在命令行中**ctrl + e**可以跳到行末

第五讲 Git添加，删除，修改与日志

****添加：****

**git add ‘text.text’** 将在工作区依旧修改的文件加入暂存区中

**git add .** 将工作区中所有更改的文件加入暂存区

**git commit -m ‘commit message’** 将在暂存区的文件提交到版本库

**git commit -am ‘commit message’** 将在工作区的文件直接提交到版本库中

新创建的文件不能直接以此方式提交至版本库中，只有已存在被修改的文件可

****删除：****

**rm ‘text.text’** 在本地删除此文件

*手动将此变更在提交到暂存区在提交版本库*

**git rm ‘text.text’** 在本地删除此文件且将此次动作提交到暂存区

*git rm = rm + add*

*文件必须已经被提交在版本库中*

*删除操作实质上也是先删除后提交（先执行删除命令，再将删除记录保存在暂存区的等待提交，后需要手动提交且写注释）*

****恢复删除：****

**git reset HEAD ‘text.text’** 将此文件的变更从暂存区删除，还原到工作区

将删除此文件的动作从暂存区退回到工作区

**git checkout -- ‘text.text’** 将工作区中的文件还原成未更改的状态

将此文件在工作区所做的删除动作撤销回未被更改的状态’

****修改文件名：****

**mv** **“text.text” “text1.text”** 将text重命名为text1

*删除原文件并生成新文件*

**git mv “text.text” “text1.text”** 将text重命名为text1

*git mv = mv + add*

*重命名文件并将此文件的更改纳入暂存区*

*重命名实质上是删除原文件并生成新文件（先执行删除命令并生成新文件，并将记录保存在暂存区等待提交，后需要手动提交且写注释）*

*rename = new + delete*

****修改提交注释：****

**git commit –amend -m ”message”** 修正上一次的提交

****提交日志：****

**git log** 获取commit提交历史

**git log -num** 可以获得最新的num条提交历史

**git log --graph** 可以以图表的形式展示提交历史

**git log ­--stat** 显示简要的增改行数的统计

**git log --pretty=oneline** 以一行的形式来显示提交历史

**git log --graph --abbrev-commit** 简写提交

****help：****

**git help ‘command’** 显示帮助

**git** **‘command’ --help** 显示帮助

第六讲 .gitignore与分支

****.gitignore:****

将不需要加入版本库的系统配置文件忽略在版本库之外

1. 在目录下新建.gitignore文件
2. 将想要忽略的文件名写入.gitignore中
3. 将.gitignore提交至版本库

注：只能是没有加入版本库中的文件

支持通配符\*和运算表达式！

/\*/text.txt 下层目录下的text.txt

/\*\*/text.txt 所有的目录下的text.txt

****分支：****

**git branch** 查全拥有多少分支

**git branch** ‘分支名’ 建立新的分支

**git checkout** ‘分支名’ 转到指定分支

第七讲 分支重要操作

****分支基础操作：****

**git branch** 查看分支列表

**git branch ‘my\_branch’** 新建my\_branch分支

**git checkout ‘branch’** 切换到branch分支

**git branch -d ‘my\_branch’** 删除分支

不能删除当前所在分支

有改动且在未被合并的分支上使用git branch -D ‘my\_branch’强制删除

**git checkout -b ‘new\_branch’** 创建新的分支并进入此分支

**git checkout -v** 显示最近一次此分支上的提交

****分支合并操作：****

**git merge ‘my\_branch’** 将my\_branch合并到当前分支

当未冲突时，‘快进’直接完成合并

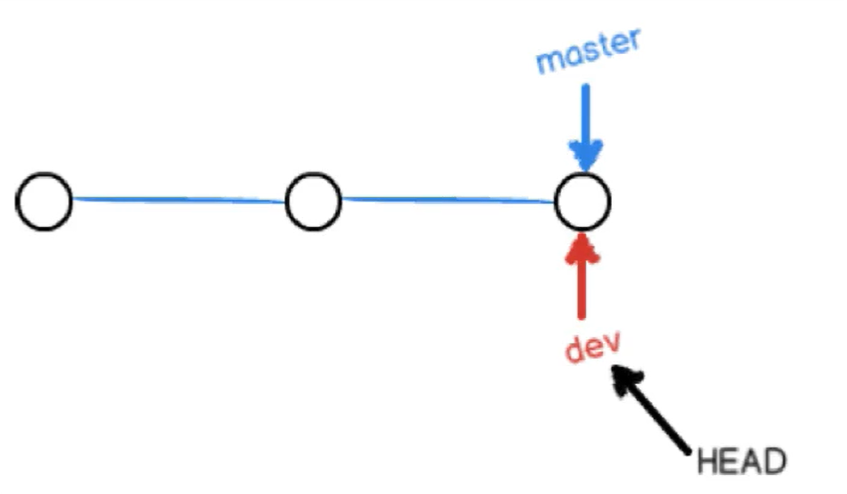
当发生冲突的时候，手动解决有冲突的文件，并运行add，commit

第八讲 分支进阶与版本回退

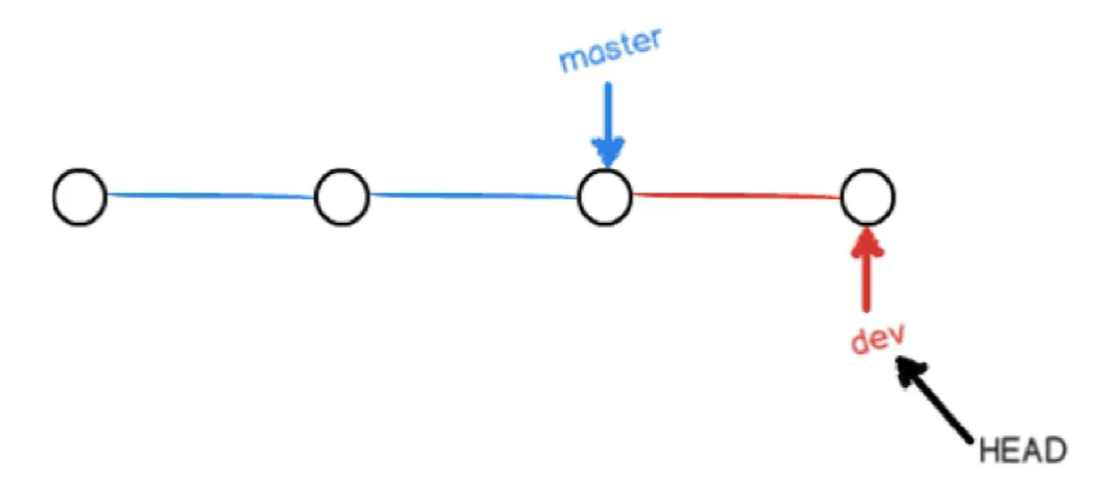
****分支进阶：****

* HEAD指向的是当前分支（可变）
* master指的是最新提交（理论不变）

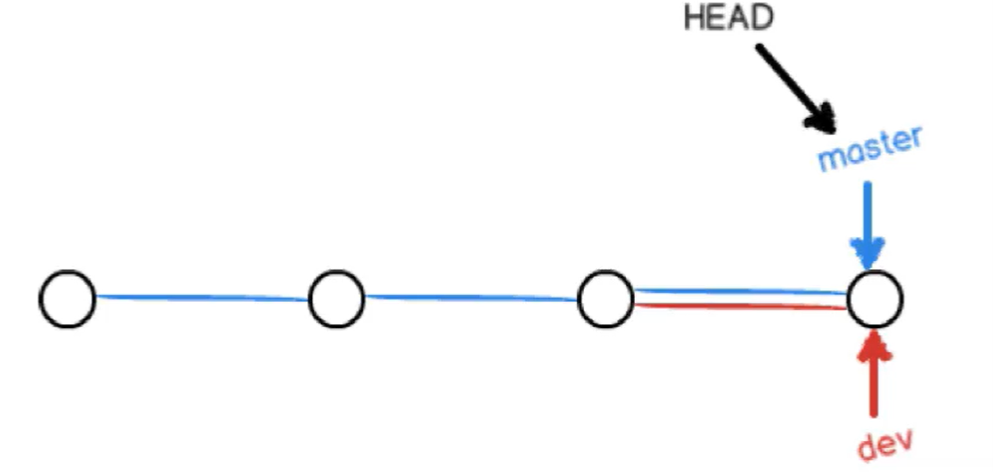
**新建分支：**



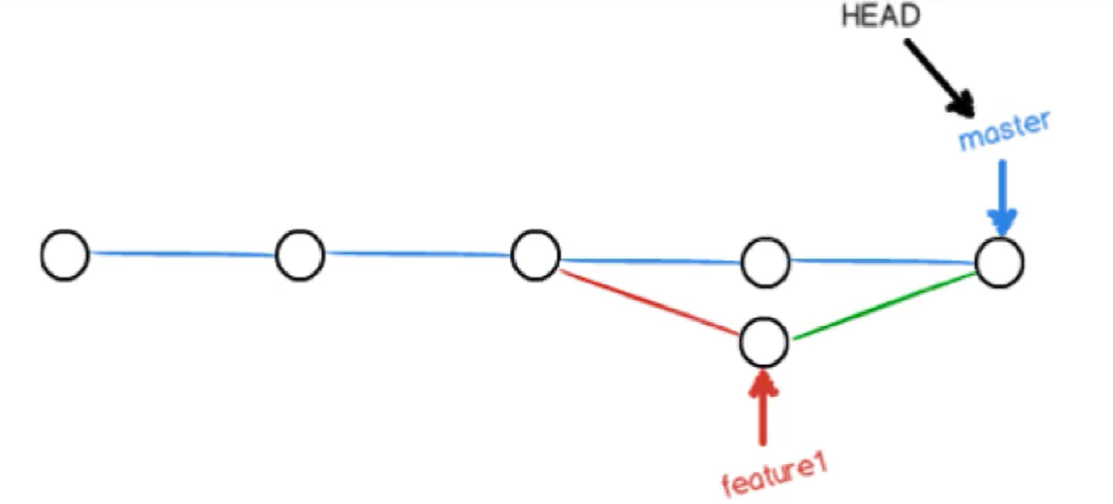
**分支更新：**



**合并更新（fast-forward）：**



**分支冲突：**



注：

fast-forward模式在删除分支的时候会丢失分支信息

--no-ff会禁用fast-forward模式（会手动提交一个commit id）

会显示一步合并操作。

****版本回退：****

**git reset --hard HEAD^**  回退^数量个版本

**git reset --hard HEAD~num** 回退num个版本

**git reset --hard commit\_id**  回退到指定版本

****提交历史和操作历史：****

**git log** 提交历史记录

**git reflog** 操作历史记录

第九讲 checkout进阶与stash

checkout-reset命令进阶：

**git checkout -- ‘text.txt’** 丢弃掉相对于暂存区最后一个添加的文件内容所做的变更

（对其掉所有没有提交到暂存区的所有内容）

但是需要在提交暂存区之前执行

**git checkout ‘branch’** 切换到master分支的最新的一个提交点

**git checkout -b ‘branch’** 新建并切换到branch分支

**git checkout ‘commit\_id’** 切换到某一提交点

在此节点提交变更内容不会改变历史提交内容，但做出修改时需要新建分支或者使用将所做的修改保存起来

**git reset HEAD ‘text.txt’**  将之前添加到暂存区的内容从暂存区移出到工作区。

stash命令：

**保存：**

**git stash**  将此分支在工作区所做的改动保存在本地，使得文件退回到未被编辑的状态，以来可以顺利切换到其他分支进行工作

**git stash save**提交时添加说明

**查询：**

**git stash list** 显示已经被保存数据

**读取：**

**git statsh pop** 恢复最近的一个存档并且删除此文档。

**git stash apply** 恢复最近的一个存档并且删除此文档，需要手动删除

**git stash apply stash@{num}** 恢复成指定的存档

**删除存档：**

**git stash drop stash@{num}** 删除id为num的那个存档

重命名branch：

**git cranch -m ‘branch’ ‘branch2’**  将branch重命名为branch2

第十讲 标签与diff

标签：

**标签：**

在版本的发布或者里程碑时打上标签。

标签有轻量级标签和带有附注标签两种。

针对整个系统而非分支。

**标签的创建：**

**git tag ‘V1.0.1’** 创建轻量级标签

**git tag -a ‘V1.0.1’ -m ‘release version’**  创建带有附注的标签。

**查看标签：**

**git tag** 查看标签

**查找标签：**

**git tag -l ‘tag\_name‘**

可以使用通配符\*’

**标签的删除：**

**git tag -d ‘tag\_name’**  删除标签

diff：

**blame ：**可以展示摸个文件的修改历史

**git blame ‘text.txt’**  展示文件的修改历史

**diff：**

**diff ‘text1.txt’ ‘text2.txt**

比较源文件与目标文件的区别，使用加减来表示不同。

**diff -u ‘text1.txt’ ‘text2.txt**

**git diff：**

文件区与暂存区比较：

**git diff** 默认比较文件在文件区和在暂存区的差异  
 将暂存区的文件当做源文件，文件区的为目标文件

工作区与版本库比较：

**git diff ‘commit\_id‘**  比较指定版本库与当前文件区的文件的差异。

**git diff HEAD** 比较最新的版本库中的文件与工作区文件的差异

暂存区与版本库比较：

**git diff --cached [commit\_id]** 比较指定版本库和当前暂存区的文件的差异

第十一讲 远程与Github