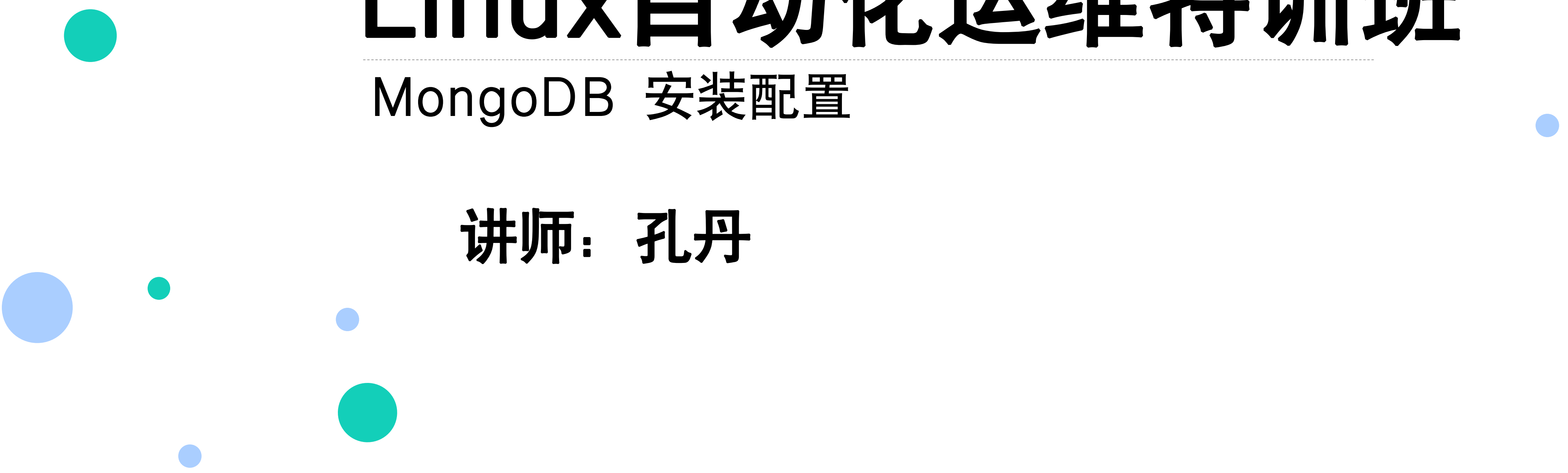




Linux自动化运维特训班

Git分布式版本控制系统

讲师：孔丹



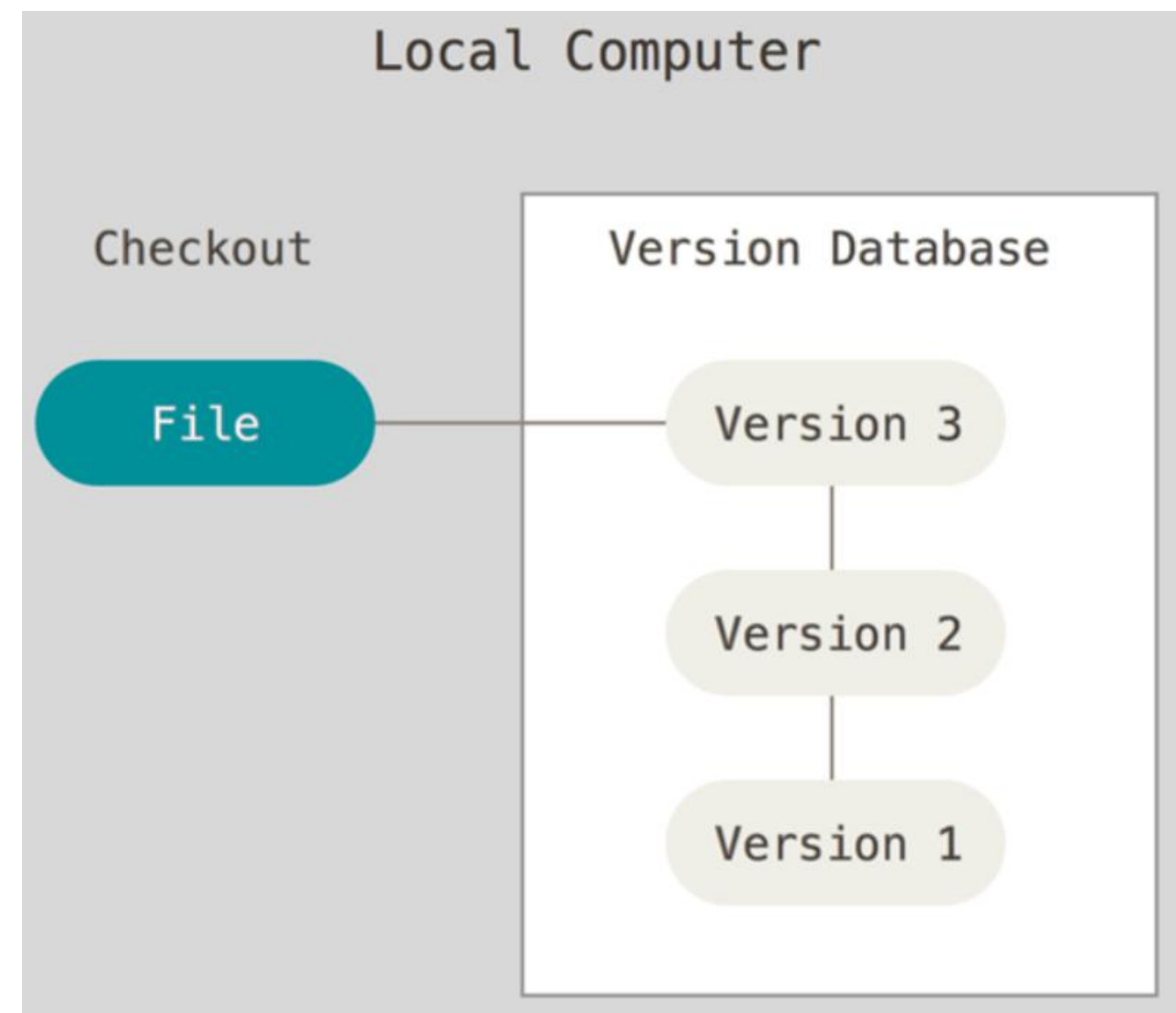
大纲

- 版本控制简介
- Git简介
- Git安装
- Git基本配置
- Git常用操作
- windows上使用git

版本控制简介

□ 本地版本控制

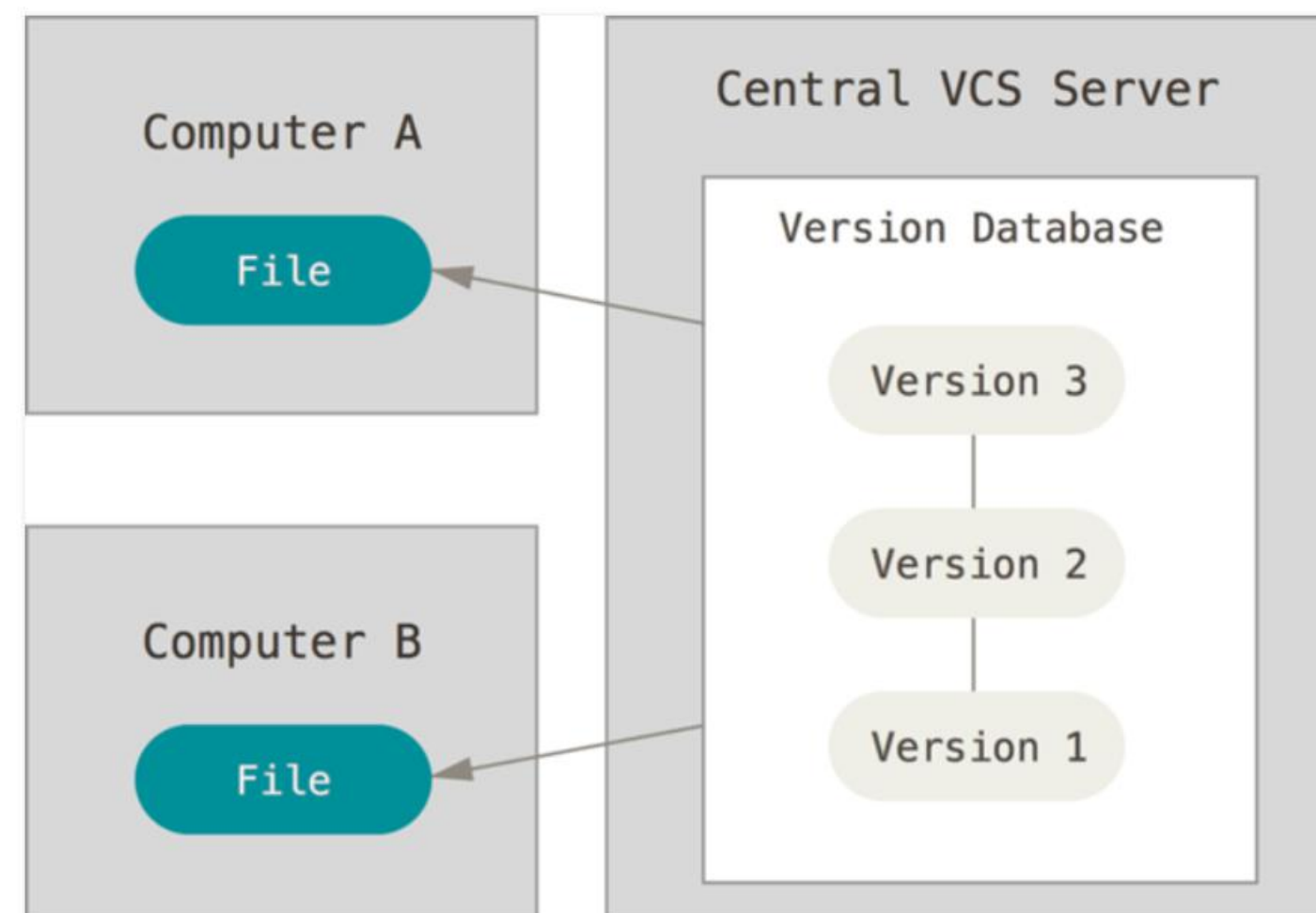
本地版本控制系统 许多人习惯用复制整个项目目录的方式来保存不同的版本，或许还会改名加上备份时间以示区别。这么做唯一的 好处就是简单，但是特别容易犯错。有时候会混淆所在的工作目录，一不小心会写错文件或者覆盖意想不到的文件。



版本控制简介

□ 集中化的版本控制系统

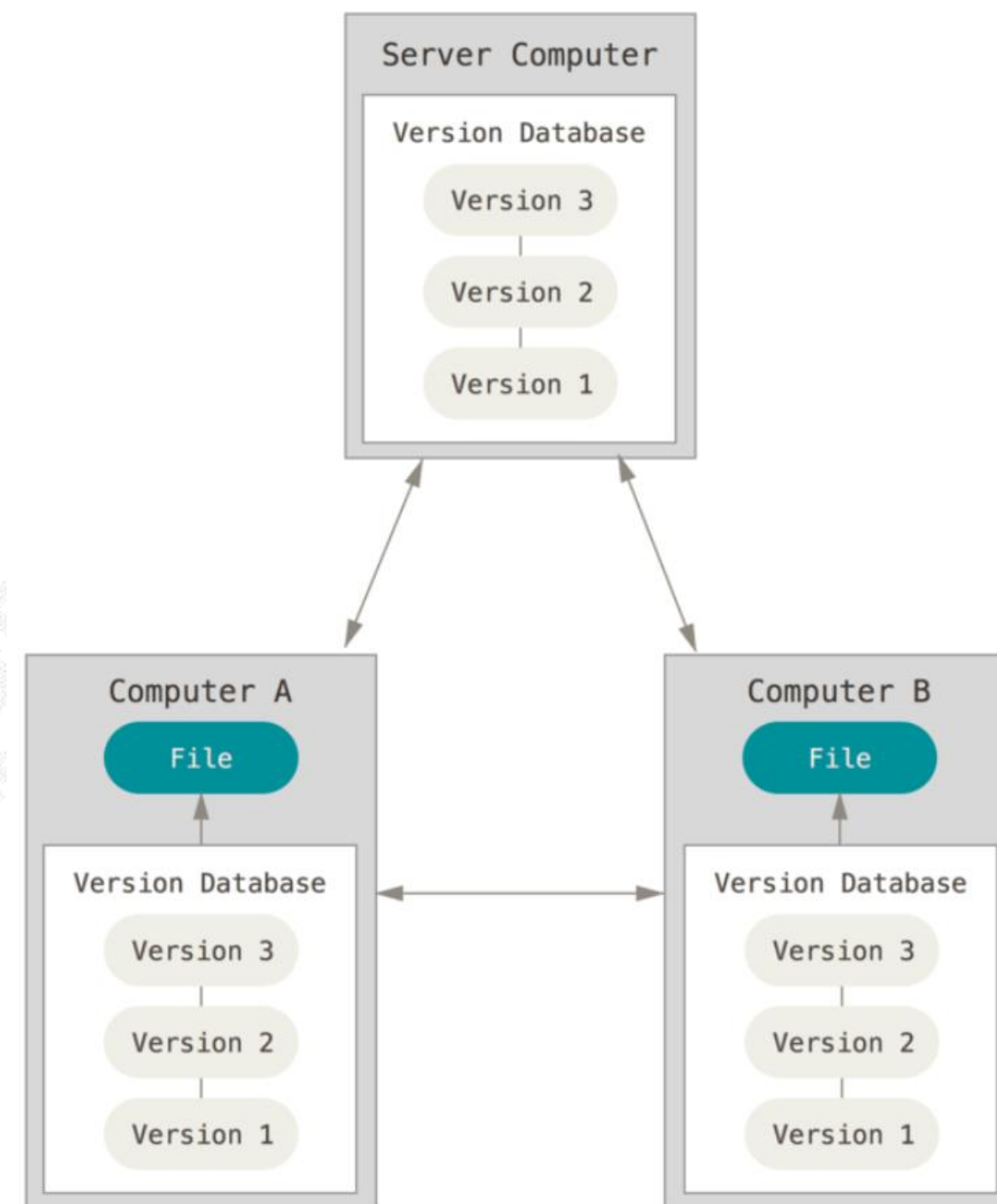
如何让在不同系统上的开发者协同工作？于是，集中化的版本控制系统（Centralized Version Control Systems, 简称 CVCS）应运而生。这类系统，诸如 CVS、Subversion 以及 Perforce 等，都有一个单一的集中管理的服务器，保存所有文件的修订版本，而协同工作的人们都通过客户端连到这台服务器，取出最新的文件或者提交更新。多年以来，这已成为版本控制系统的标准做法。



版本控制简介

□ 分布式版本控制系统

在这类系统中，像Git、Mercurial、Bazaar 以及 Darcs 等，客户端并不只提取最新版本的文件快照，而是把代码仓库完整地镜像下来。这么一来，任何一处协同工作用的服务器发生故障，事后都可以用任何一个镜像出来的本地仓库恢复。因为每一次的克隆操作，实际上都是一次对代码仓库的完整备份。



Git简介

□ git产生

git是一个分布式版本控制软件，最初由林纳斯·托瓦兹（Linus Torvalds）创作，于2005年以GPL发布。最初目的是为更好地管理Linux内核开发而设计。

自2002年开始，林纳斯·托瓦兹决定使用BitKeeper作为Linux内核主要的版本控制系统用以维护代码。因为BitKeeper为专有软件，这个决定在社区中长期遭受质疑。在Linux社区中，特别是理查德·斯托曼与自由软件基金会的成员，主张应该使用开放源代码的软件来作为Linux核心的版本控制系统。林纳斯·托瓦兹曾考虑过采用现成软件作为版本控制系统（例如Monotone），但这些软件都存在一些问题，特别是性能不佳。现成的方案，如CVS的架构，受到林纳斯·托瓦兹的批评。

2005年，安德鲁·垂鸠写了一个简单程序，可以连接BitKeeper的存储库，BitKeeper著作权拥有者拉里·麦沃伊认为安德鲁·垂鸠对BitKeeper内部使用的协议进行逆向工程，决定收回无偿使用BitKeeper的授权。Linux内核开发团队与BitMover公司进行磋商，但无法解决他们之间的歧见。林纳斯·托瓦兹决定自行开发版本控制系统替代BitKeeper，以十天的时间，编写出第一个git版本

Git简介

□ 为什么用Git

1. 开源的代码管理软件
2. 分布式版本控制系统
3. 分散式协作存储工具

和svn cvs相比有下面的好处:

- 如遇服务器宕机, 整个协同工作无法进行, 因为此时无法进行代码更新提交, 当然也不能checkout最新代码
- 如果服务器数据丢失, 整个版本数据也将丢失, 除非刻意版本管理服务器备份

而git可以方便地在本地进行版本管理, 就如同在你本地有一个版本管理服务器一样。你可以选择在合适的时候将本地版本推送到统一的版本管理服务器

Git简介

□ Git和svn比较

代码管理

- ▶ svn是集中式管理
- ▶ 版本库是集中存放在中央服务器的， 每次必须联网获取最新的代码， 工作完成后再推送到中央服务器。
- ▶ 这个的弊端就是必须联网， 局域网内还可以， 如果是互联网的话， 在带宽不够的情况下， 上传一份大文件就会很慢， 体验很差
- ▶ git是分布式管理
- ▶ 每个人的电脑上都有一个版本库， 这样就不需要联网了， 因为版本库就在自己电脑上（本地库）。
- ▶ 中央服务器是用来交换大家修改的内容（远程库）， 即使中央服务器宕机了， 也可以立即从本地版本库中恢复代码
- ▶ 文件存储
- ▶ git是保存历史版本的完整文件， svn是保存文件差异

Git安装

□ Git官网

<https://git-scm.com>



中文手册: <https://git-scm.com/book/zh/v2>

Git简介

□ yum安装

本课程主要介绍Linux下安装

```
yum install -y git
```

□ 源码包安装

git官网下载源码包编译安装。

安装完毕后，可以使用git --version验证

Git基本配置

□ 配置使用者信息

git config --global user.name zhangm

git config --global user.email 12345678@gq.com

□ local、 global、 system三种级别

local 当前项目有效 (工作目录/.git/config)

global 当前用户有效 (用户目录/.gitconfig)

system 所有用户有效 (git目录/etc/gitconfig)

□ 删除使用者信息

git config --unset user.name

git config --unset user.email

默认删除local下的, 如果需要删除指定级别下的, 则
添加具体参数

Git基本配置

□ 查询使用者信息

查看所有配置: `git config — list`

查看指定的key : `git config username`

□ git帮助

`git help`

`git --help`

`man git-key (linux)`

比如要查看config的用法

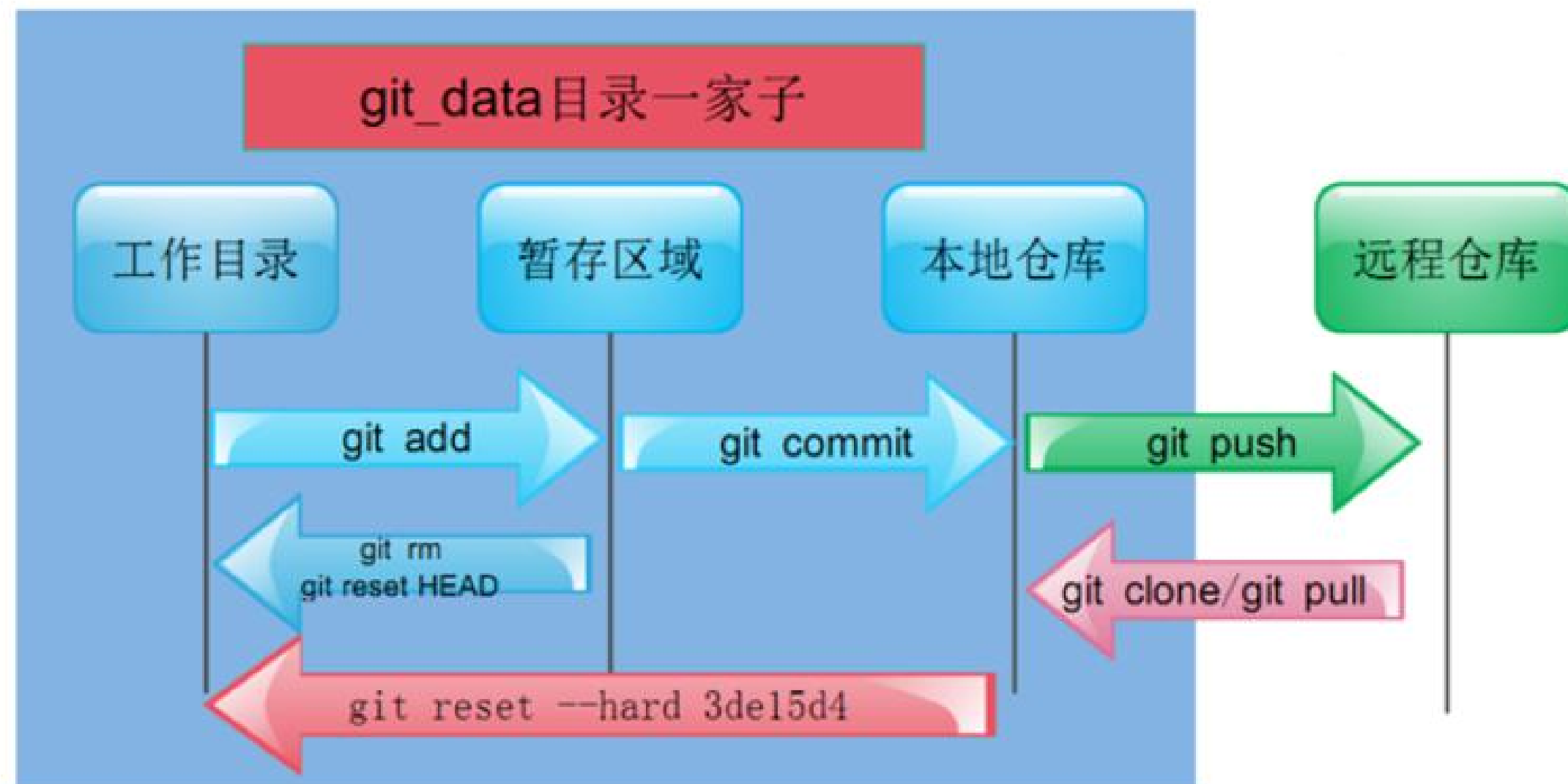
`git help config`

`git --help config`

`man git-config`

Git工作区

- git有三个常用的工作区：
 - 工作区： 编码、 编写文档
 - 暂存区： 标记文件等待提交
 - 历史仓库： 文件托管



Git文件状态

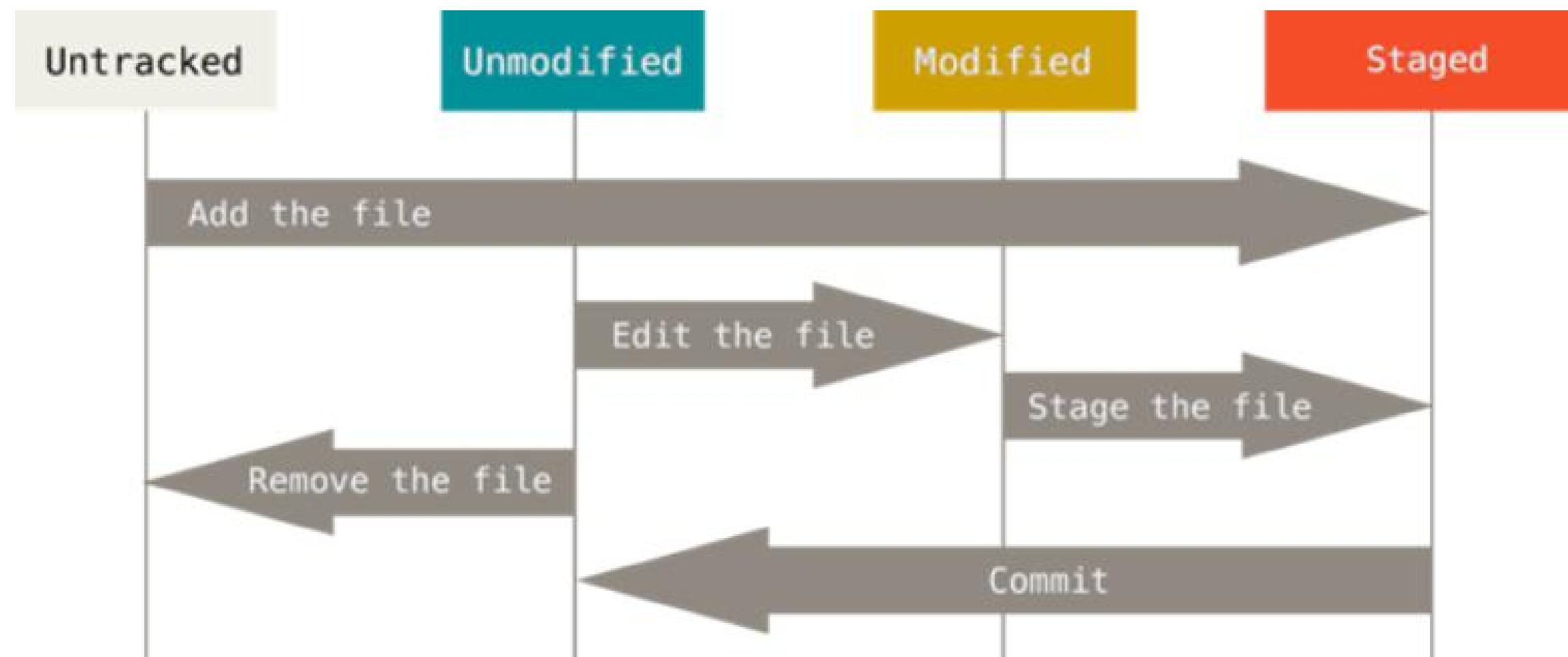
□ git文件有以下四种状态

unstaged: 历史仓库中没有文件记录

modified: 历史仓库中存在文件记录, 但是此文件被修改过

staged: 文件被保存到暂存但并没有提交到历史仓库

committed: 文件被提交到历史仓库



Git仓库初始化

创建目录

```
mkdir git_data
```

进入目录

```
cd git_data/
```

初始化

```
git init
```

查看工作区状态

```
git status
```

默认处于master分支， HEAD上指针指向mgster分支

Git目录

HEAD : git项目当前处于哪个分支

config:项目的配置信息 (git config --local)

description : 项目描述信息

index : 索引文件

hooks/:系统默认钩子脚本目录

logs/: 各个refs的历史信息

objects/:git仓库的所有对象 (commit、 tree、 blob、 tag)

refs/:标识项目中每个分支指向了哪个commit

Git对象

git使用40个 16进制的SHA-1 Hash来 唯一标识对象

git对象可以分为四种

blob: 二进制文件

tree: 目录 (指针)

commit: 提交 (快照)

tag: 固定的提交

tag -> commit -> tree -> tree * * *-> blob

Git命令常规操作

命令	命令说明
add	添加文件内容至索引
bisect	通过二分查找定位引入 bug 的变更
branch	列出、创建或删除分支
checkout	检出一个分支或路径到工作区
clone	克隆一个版本库到一个新目录
commit	记录变更到版本库
diff	显示提交之间、提交和工作区之间等的差异
fetch	从另外一个版本库下载对象和引用
grep	输出和模式匹配的行
init	创建一个空的
Git	版本库或重新初始化一个已存在的版本库
log	显示提交日志
merge	合并两个或更多开发历史
mv	移动或重命名一个文件、目录或符号链接

Git命令常规操作

命令	命令说明
pull	获取并合并另外的版本库或一个本地分支
push	更新远程引用和相关的对象
rebase	本地提交转移至更新后的上游分支中
reset	重置当前HEAD到指定状态
rm	从工作区和索引中删除文件
show	显示各种类型的对象
status	显示工作区状态
tag	创建、列出、删除或校验一个GPG签名的 tag 对象

Git命令常规操作

1 创建文件

```
[root@gitlab git_data]# touch README
```

```
[root@gitlab git_data]# git status
```

```
# 位于分支 master
```

```
# 初始提交
```

```
#
```

```
# 未跟踪的文件:
```

```
#   (使用 "git add <file>..." 以包含要提交的内容)
```

```
#   README
```

```
提交为空, 但是存在尚未跟踪的文件 (使用 "git add" 建立跟踪)
```

```
  添加文件跟踪
```

```
[root@gitlab git_data]# git add ./*
```

```
[root@gitlab git_data]# git status
```

```
# 位于分支 master
```

```
# 初始提交
```

```
#
```

```
# 要提交的变更:
```

```
#   (使用 "git rm --cached <file>..." 撤出暂存区)
```

```
#
```

```
#   新文件:      README
```


Git命令常规操作

1 创建文件

文件会添加到.git的隐藏目录

```
[root@gitlab git_data]# tree .git/
```

查看git的状态

```
[root@gitlab git_data]# git status
```

位于分支 master
无文件要提交，干净的工作区

2 添加新文件

git add * 添加到暂存区域

git commit 提交git仓库 -m 后面接上注释信息，内容关于本次提交的说明，方便自己或他人查看

Git命令常规操作

3 文件删除

删除工作区、暂存区文件，最终历史仓库中文件也会删除

```
git rm -f hello.java
```

删除暂存区中的文件，不删除工作区

```
git rm --cached hello.java
```

4 重命名暂存区数据

- 没有添加到暂存区的数据直接mv/rename改名即可。
- 已经添加到暂存区数据：

```
git mv README NOTICE
```

Git命令常规操作

5 查看历史记录

- `git log` #→查看提交历史记录
- `git log -2` #→查看最近几条记录
- `git log -p -1` #→-p显示每次提交的内容差异,例如仅查看最近一次差异
- `git log --stat -2` #→--stat简要显示数据增改行数, 这样能够看到提交中修改过的内容, 对文件添加或移动的行数, 并在最后列出所有增减行的概要信息
- `git log --pretty=oneline` #→--pretty根据不同的格式展示提交的历史信息
- `git log --pretty=fuller -2` #→以更详细的模式输出提交的历史记录
- `git log --pretty=format:"%h %cn"` #→查看当前所有提交记录的简短SHA-1哈希字符串与提交者的姓名。

Git命令常规操作

6 git show

git show 查看最后一次提交

git show HEAD 查看HEAD指向的分支的commit内容

git show e0783 根据hash查看内容

git show --pretty=raw e0783根据hash查看完整内容

git ls-tree 5fb56 查看tree对象内容

7 Git 撤销

恢复本地内容、提交错误等等是开发过程中常常遇到的情况，Git也给我们提供了丰富的撤销操作

git checkout -- Demo.java 从暂存区恢复到工作区、暂存区

git checkout -- test Demo.java 从test分支恢复到工作区、暂存区

git reset 还原暂存区

git revert HEAD 产生一个新的提交撤销之前的提交

Git命令常规操作

8 Git 对比差异

对比工作区、暂存区、历史仓库中内容是常用的技巧

- ▶ `git diff Demo.java` 对比工作区和暂存区中的差异
- ▶ `git diff --cached Demo.java` 对比暂存区和历史仓库中的差异
- ▶ `git diff HEAD*` 对比上一个版本的差异
- ▶ `git diff origin/master:master` 对比远程master和本地master的差异

9 标签使用

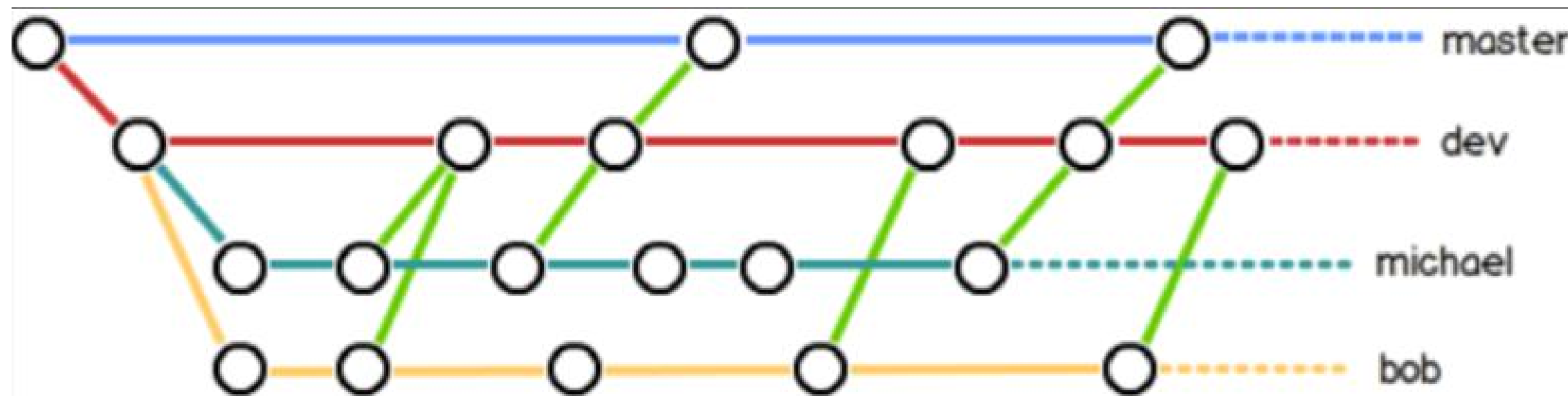
git 标签可以理解为用户里程碑。

▶ 可以这样设想一下，当项目上线了，当前版本是一个稳定版本。这时项目需要二期开发，我们依然会在当前分支下进行管理。如果没有为稳定版本打Tag，后续想要切换回这个版本就需要去查看提交历史，这是一个很麻烦的过程，这个就是git标签的好处。

- ▶ `git tag "v-0.0.1"`
- ▶ `git tag -a "v-0.0.2" -m "project released on v-0.0.2"`
- ▶ `git tag -d v-0.0.1` #删除之前的v-0.0.1

Git分支结构

- 在实际的项目开发中，尽量保证master分支稳定，仅用于发布新版本，平时不要随便直接修改里面的数据文件。
- 那在哪干活呢？干活都在dev分支上。每个人从dev分支创建自己个人分支，开发完合并到dev分支，最后dev分支合并到master分支。所以团队的合作分支看起来会像下图那样。



Git分支结构

□ 分支切换

- ▶ `git branch test` 创建test分支
- ▶ `git checkout test` 切换到test分支

□ 分支合并

在test分支进行修改

```
echo "Creating a new branch is quick." >> readme.txt
```

```
git add readme.txt
```

```
git commit -m "new branch"
```

然后切换到master进行合并

```
git checkout master
```

查看文件

```
cat readme.txt
```

合并分支

```
git merge test
```

Git分支结构

□ 合并失败解决

模拟冲突，在文件的同一行做不同修改
在test分支修改

```
echo "test in test" >> readme.txt  
git add readme.txt  
# git commit -m "test in test "
```

在master 分支进行修改

```
git checkout master  
echo "test in master" >> readme.txt  
git add readme.txt  
git commit -m "test in master "
```

合并: git merge test

冲突后首先手工处理，Git用< <<<<<<, =====, >>>>>>分割开了各个分支冲突的内容，我们需要手工的删除这些符号，并将内容修改在提交

```
git add readme.txt  
# git commit -m "conflict fixed"
```

Git分支结构

□ 删除分支

分支名字前没有 * 号的分支通常可以使用 `git branch -d` 删除掉
查看所有包含未合并工作的分支，可以运行 `git branch --no-merged:`
`git branch -d test`

如果真的想要删除分支并丢掉那些工作，如同帮助信息里所指出的，可以使用 `-D` 选项强制删除它。

windows上git使用

□ 安装

- 1.首先打开 Git 的官方网站: <http://git-scm.com/>
- 2.然后找到下载页面: <http://git-scm.com/downloads>
- 3.找到Windows版本的下载页面: <http://git-scm.com/download/win>
- 4.因为准备使用TortoiseGit做图形客户端,所以就不选择Git GUI 版本

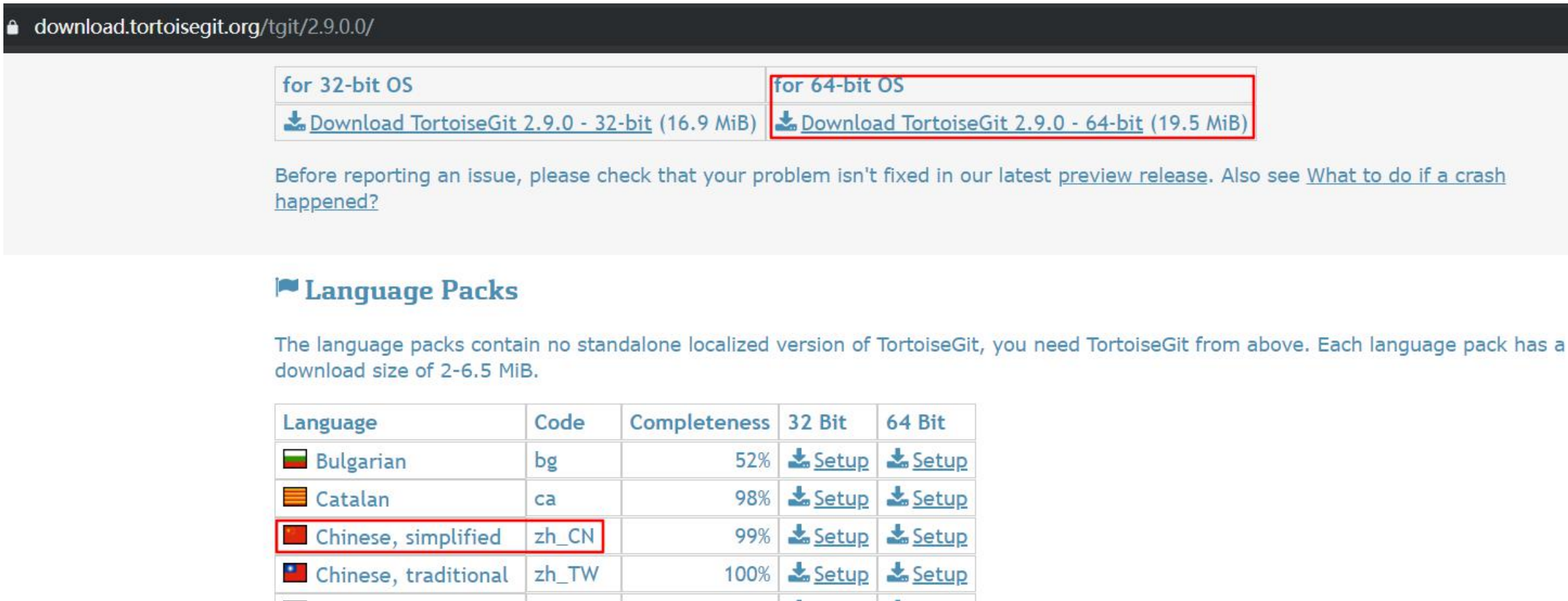


windows上git使用

□ TortoiseGit安装与配置

TortoiseGit 简称 tgit, 中文名海龟Git. 海龟Git只支持神器 Windows 系统, 有一个前辈海龟SVN, TortoiseSVN和TortoiseGit都是非常优秀的开源的版本库客户端. 分为32位版与64位版.并且支持各种语言,包括简体中文 (Chinese, simplified; zh_CN).

- 1.TortoiseGit下载页面: <http://download.tortoisegit.org/tgit/>
- 2.打开下载页面后, 找到对应的 Latest stable release (最新稳定版) 目录
- 3.进入具体版本页面后,根据Windows操作系统版本选择相应的程序安装包和中文语言包



windows上git使用

□ TortoiseGit安装与配置

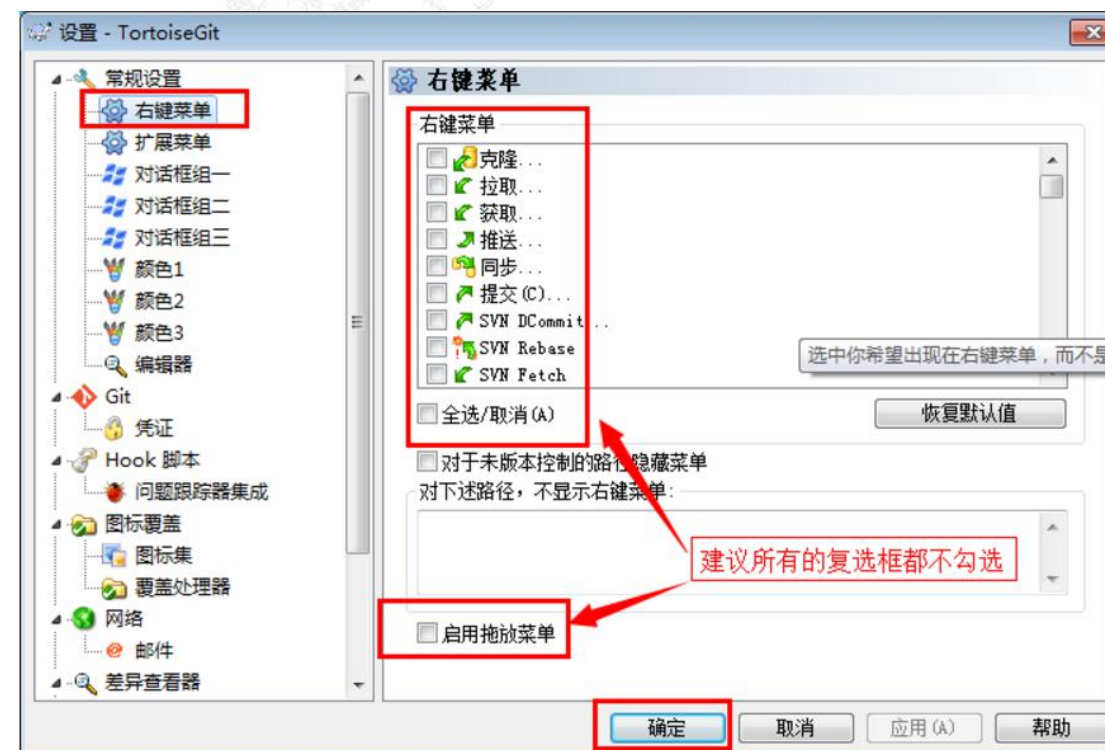
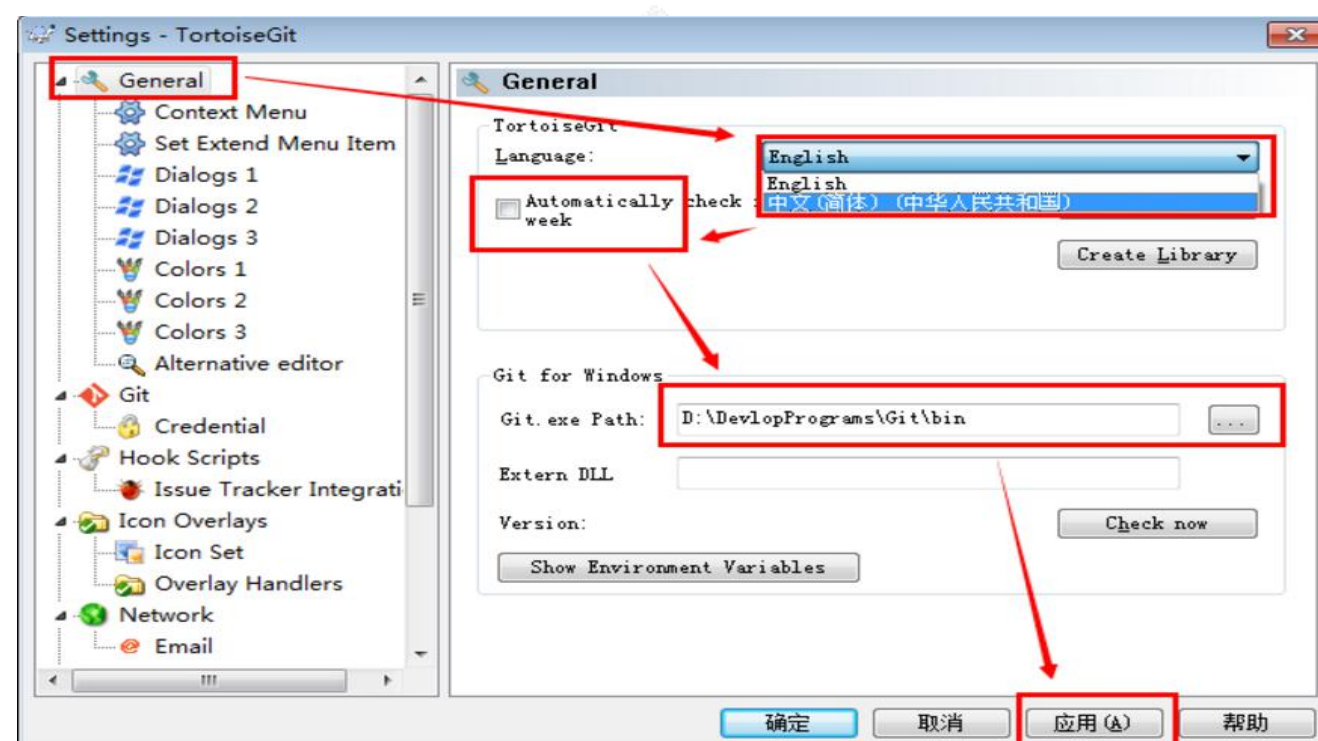
4. 我们需要先安装程序包,然后安装语言包(LanguagePack). 因为 TortoiseGit 只是一个程序壳,必须依赖一个 Git Core,也就是上一节我们安装的 Git.



windows上git使用

□ TortoiseGit安装与配置

1. 首先,请选定一个存放Git项目的目录,这样管理方便. 如 : E:\STUDY\GIT_STUDY , 然后在资源管理器中打开.
2. 在空白处点击鼠标右键, 选择 --> TortoiseGit --> Settings, 然后就可以看到配置界面:
3. 选中General,在右边的 Language中选择中文. 不勾选自动升级的复选框,可能还需要指定 Git.exe 文件的路径,如 "D:\Program Files\Git\bin". 完成后,点击应用,确定关闭对话框.
4. 再次点击鼠标右键,可以看到弹出菜单中已经变成中文. 原来的 Settings 变成 设置; Clone 变为 克隆.
5. 配置右键菜单. 在设置对话框中,点选左边的"右键菜单",然后在右边将所有的复选框都去掉,这样右键菜单显得比较干净:



windows上git使用

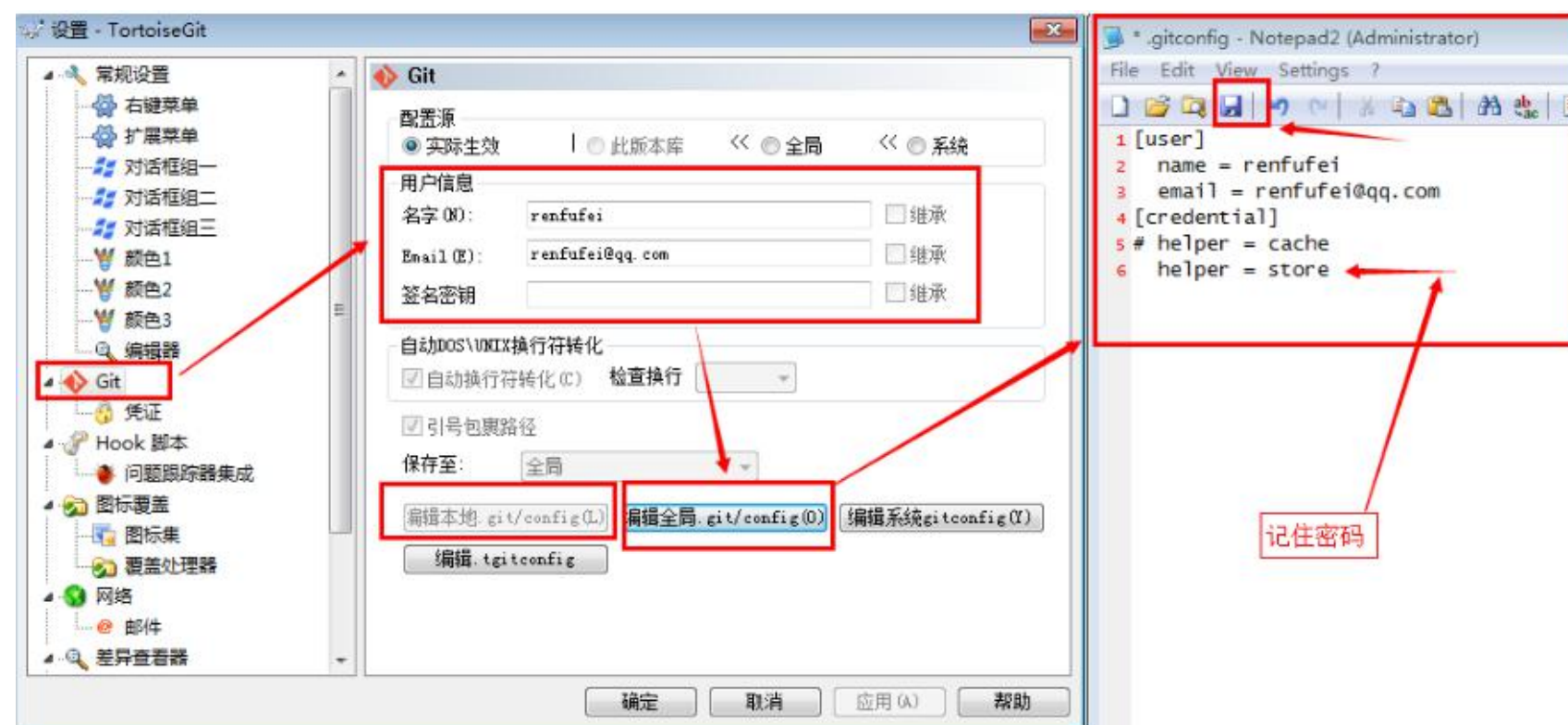
□ TortoiseGit安装与配置

配置

6.设置记住密码

!!!!!! 密码会明文保存在 C:\Users\Administrator\.git-credentials 这种文件中, 请小心使用.

进入设置, 点选左边的Git标签. 可以发现, 右边可以配置用户的名字与Email信息. 如下图所示:



上面的内容:

[credential]

helper = store

工作中git使用

□ 项目经理

- (1)项目经理搭建项目的框架。
- (2)搭建完项目框架之后，项目经理把项目框架代码放到服务器。

□ 普通员工

- (1)在自己的电脑上，生成ssh公钥，然后把公钥给项目经理，项目经理把它添加的服务器上面。
- (2)项目经理会给每个组员的项目代码的地址，组员把代码下载到自己的电脑上。
- (3)创建本地的分支dev,在dev分支中进行每天的开发。
每一个员工开发完自己的代码之后，都需要将代码发布远程的dev分支上

Master:用户保存发布的项目代码。V1.0,V2.0

Dev:保存开发过程中的代码。

总结

- 版本控制简介
- Git简介
- Git安装
- Git基本配置
- Git常用操作
- windows上使用git



谢谢观看

更多好课，请关注[万门大学APP](#)

