

GIT 从入门到精通

Git 从入门到精通

张长青
第一研究所
2013-01-10

目录

- Git简介
- Git常用命令
- TortoiseGit
- Repo的使用
- Git实战
- 总结

Git 简介

Git 是一个快速、可扩展的分布式版本控制系统，它具有极为丰富的命令集，对内部系统提供了高级操作和完全访问。

Git --- The stupid content tracker, 傻瓜内容跟踪器。Linus Torvalds 是这样给我们介绍 Git 的。

Git简介——历史

2005年，Torvalds开始着手开发Git是为了作为一种过渡方案来替代BitKeeper，后者之前一直是Linux内核开发人员在使用的版本控制工具，当时由于自由软件社区中的许多人觉得BitKeeper的使用许可证并不适合自由软件社区的工作，因此Linus决定着手开发许可证更为自由灵活的版本控制系统。

尽管最初Git的开发是为了辅助Linux内核开发的过程，但是现在很多其他自由软件项目中也使用了Git实现代码版本管理，譬如，Android项目、X.org项目、许多Freedesktop.org的项目、Ruby项目等。

Git简介——初衷

为什么使用版本控制系统？

版本控制是**管理数据变更**的艺术，无论数据变更时来自同一人，还是来自不同的人（一个团队）。版本控制系统不但要忠实地**记录**数据的每一次变更，还要能够帮助**还原**任何一次历史变更，以及实现**团队的协同工作**等。Git就是版本控制系统中的佼佼者。

版本控制系统是为懒人准备的，它让懒人们比那些善于备份文档的勤劳人拥有更干净的文件系统，以及更多的可以活着的时间。

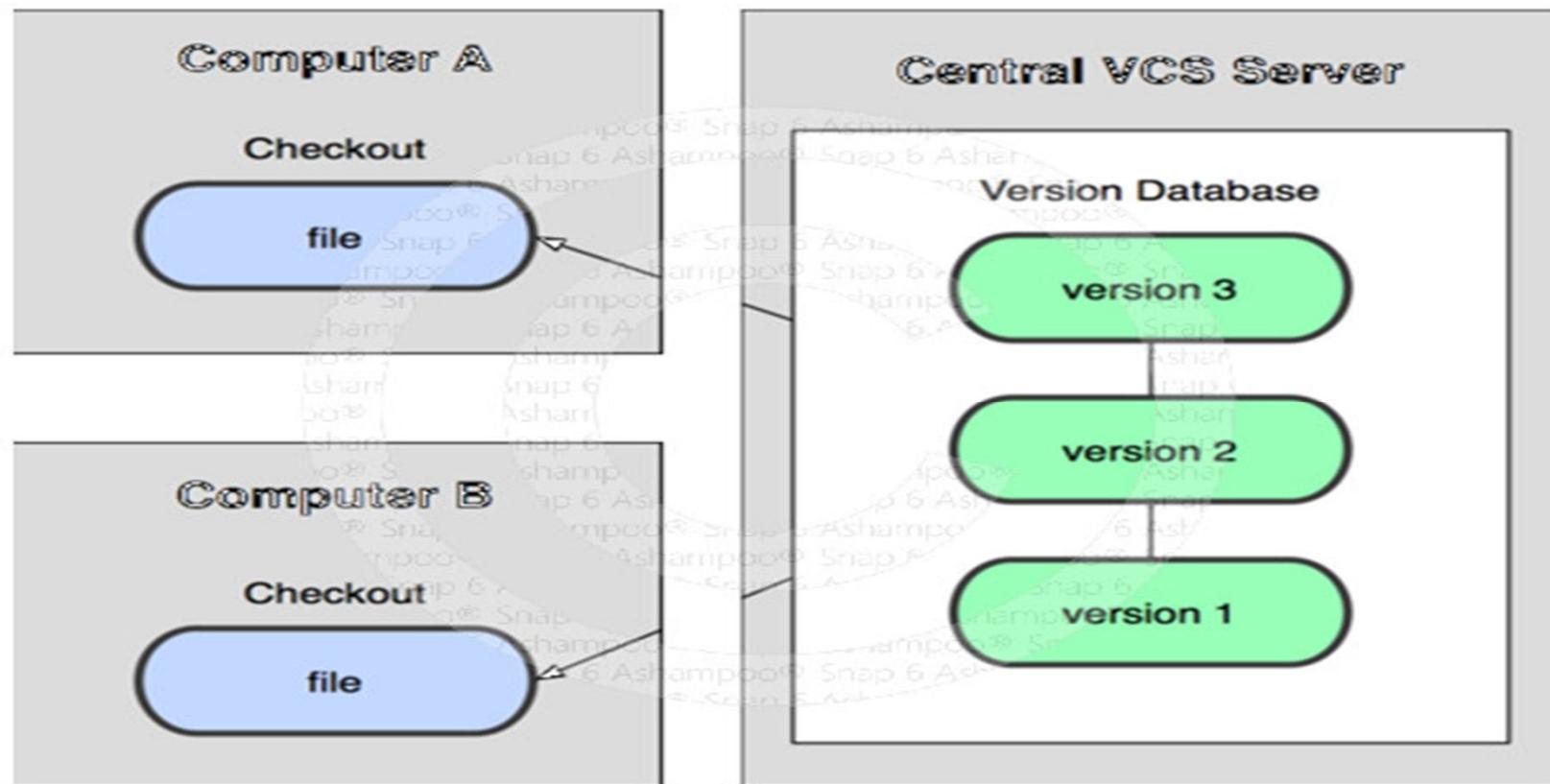
Git简介——原理

集中式管理 & 分布式管理

GIT和SVN最大的差异在于**GIT是分布式的**管理方式，而**SVN是集中式的**管理方式。如果不习惯使用代码管理工具，可能比较难理解分布式管理和集中式管理的概念。下面介绍两种工具的工作流程（团队开发），通过阅读下面的工作流程，你将会很好的理解以上两个概念。

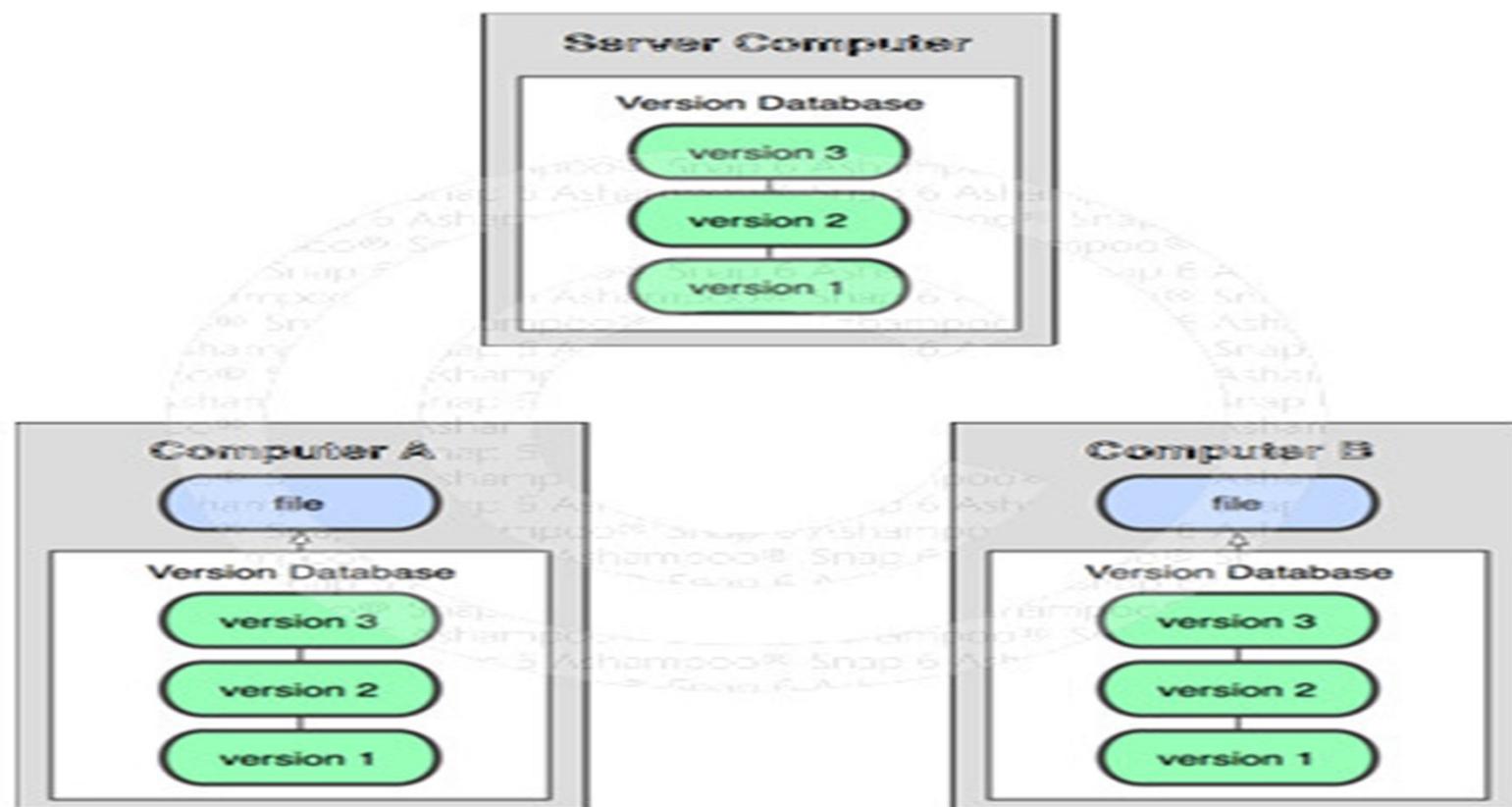
Git简介——集中式管理

集中式管理—只有服务器上拥有完整的历史记录



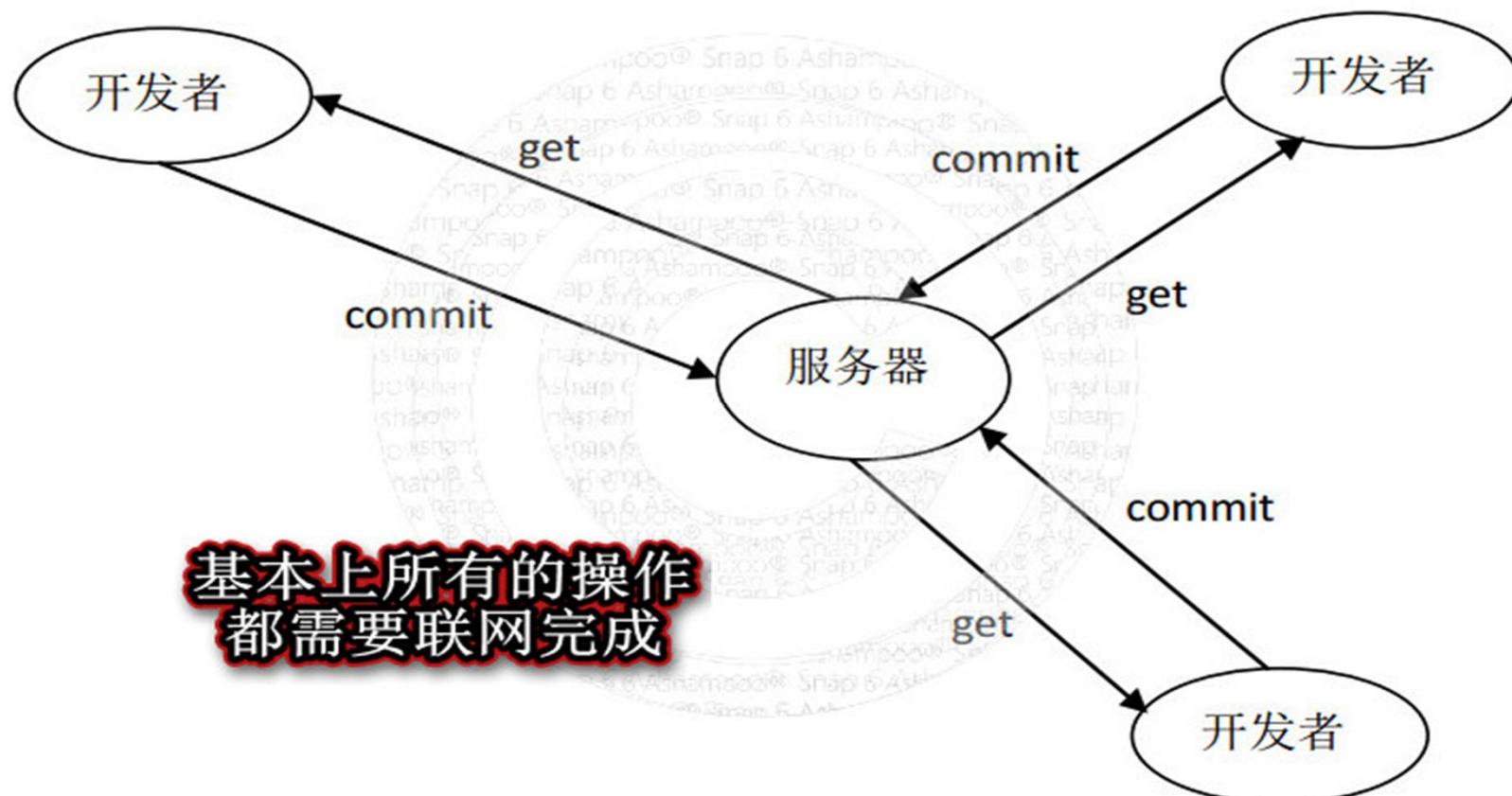
Git简介——分布式原理

分布式管理—每个人都拥有一个完整的仓库



Git简介——集中式管理

集中式管理的工作方式



Git简介——集中式管理优缺点

优点：

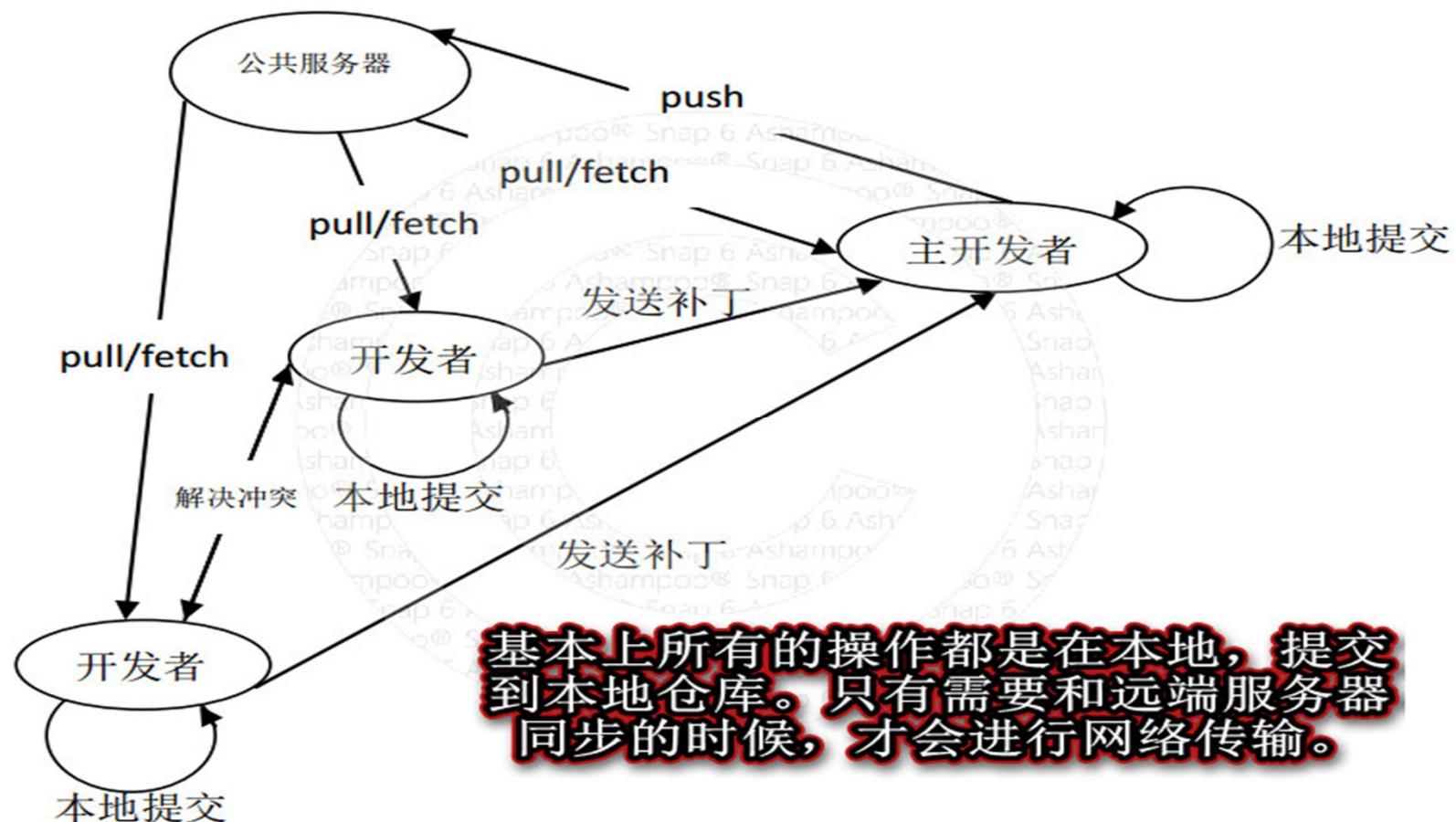
- 1、管理方便，逻辑明确，符合一般人思维习惯；
- 2、易于管理，集中式服务器更能保证安全性；
- 3、代码一致性非常高；
- 4、适合开发人数不多的项目开发。

缺点：

- 1、服务器压力太大，数据库容量暴增；
- 2、不能离线工作；
- 3、不适合多人并行开发。

Git简介——分布式管理

分布式管理的工作方式



Git简介——分布式管理优缺点

优点：

- 1、适合分布式开发，强调个体；
- 2、公共服务器压力和数据量都不会太大；
- 3、速度快、灵活；
- 4、任意两个开发者之间可以很容易的解决冲突；
- 5、支持离线工作。

缺点：

- 1、学习资料少（起码中文资料很少）；
- 2、学习周期相对而言比较长；
- 3、不符合常规思维；
- 4、代码保密性差，一旦开发者把整个库克隆下来就可以完全公开所有代码和版本信息。

Git简介——Git工作目录的结构

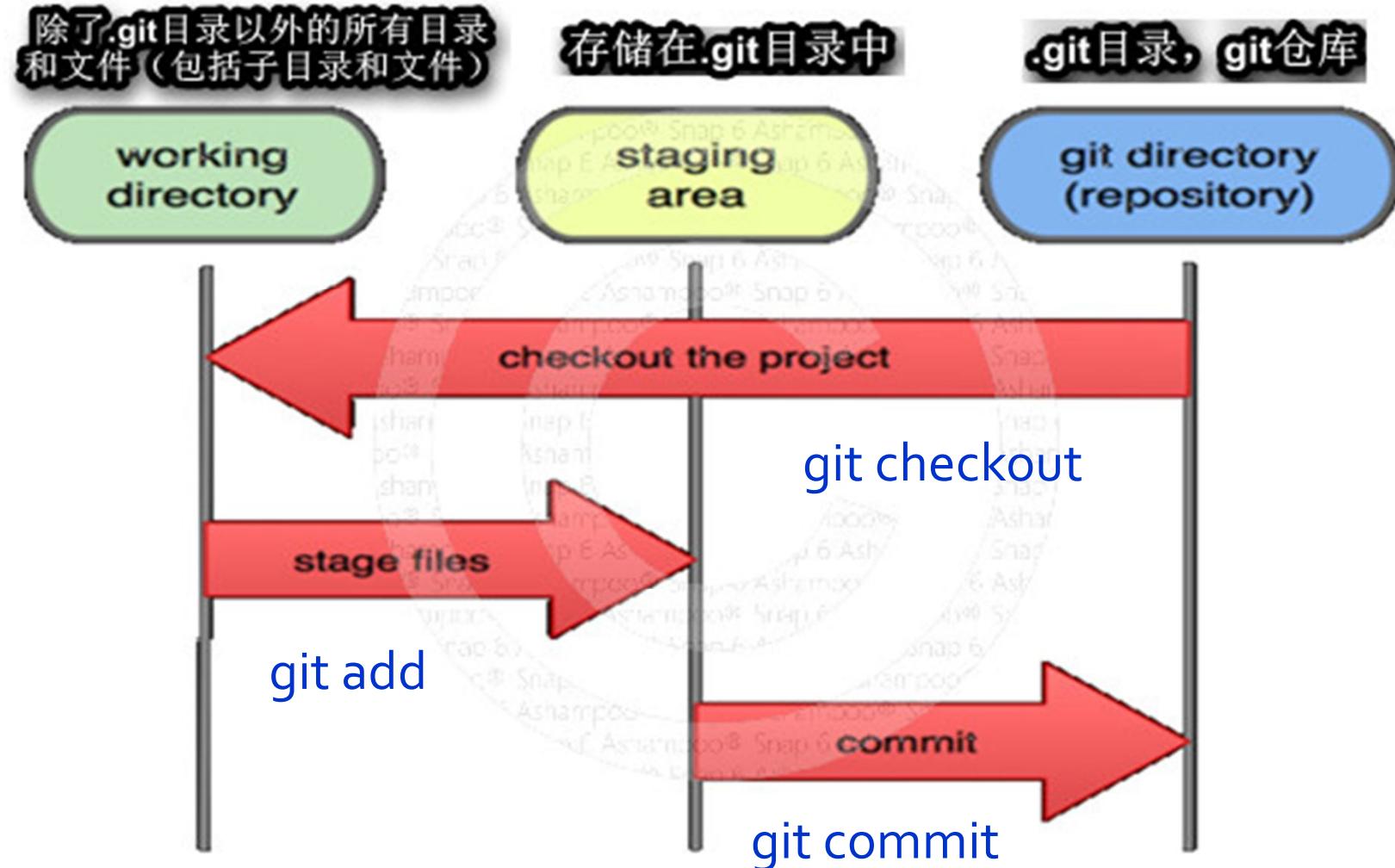


Git简介——Git仓库的结构

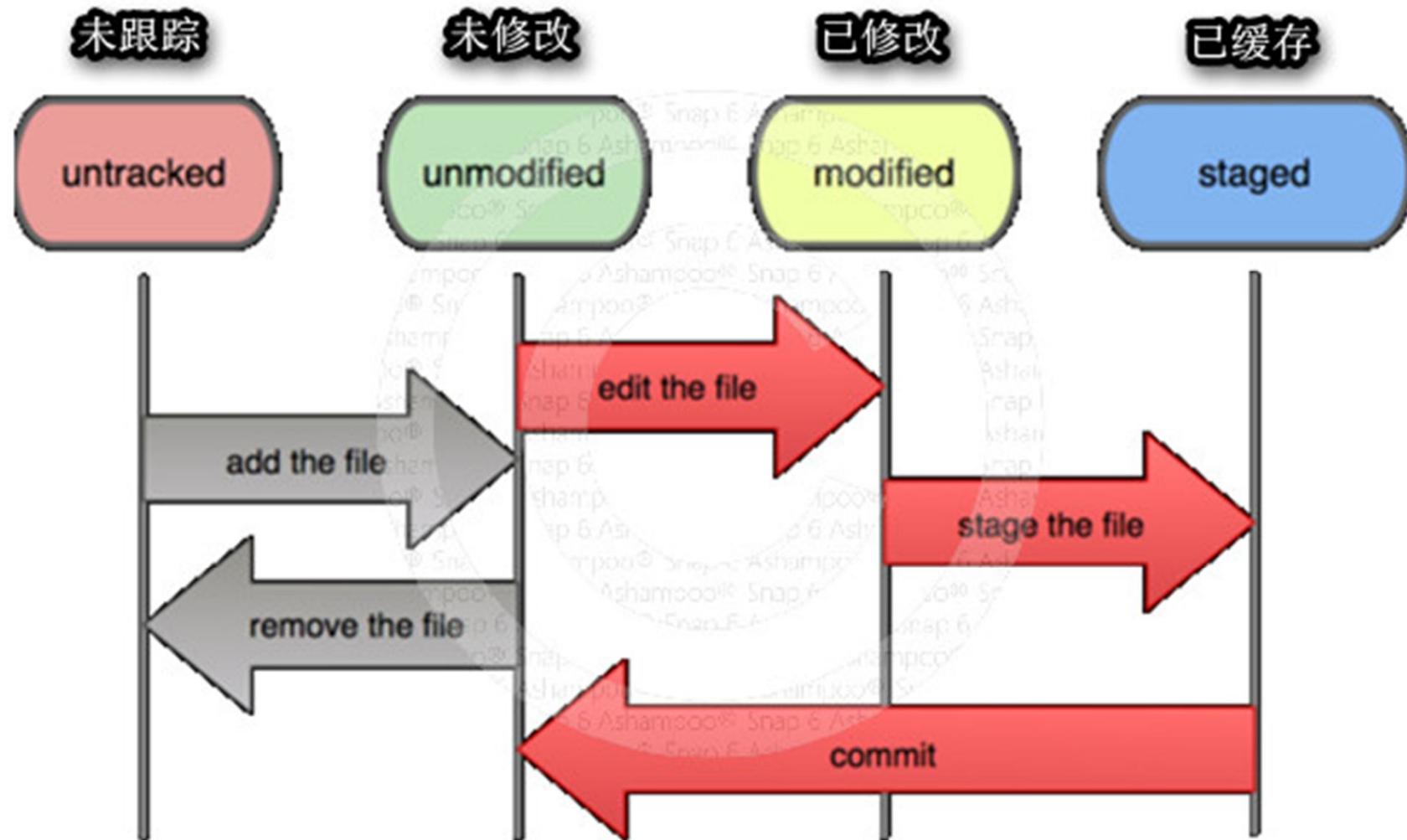
```
.git
  ├── branches
  ├── hooks
  ├── info
  ├── objects
  │   ├── info
  │   └── pack
  ├── refs
  │   ├── heads
  │   └── tags
  └── remotes
```

git 将代码库和工作拷贝的版本信息统一放在工作目录下的.git 中

Git 简介——Git 的存储区



Git简介——Git文件的状态迁移



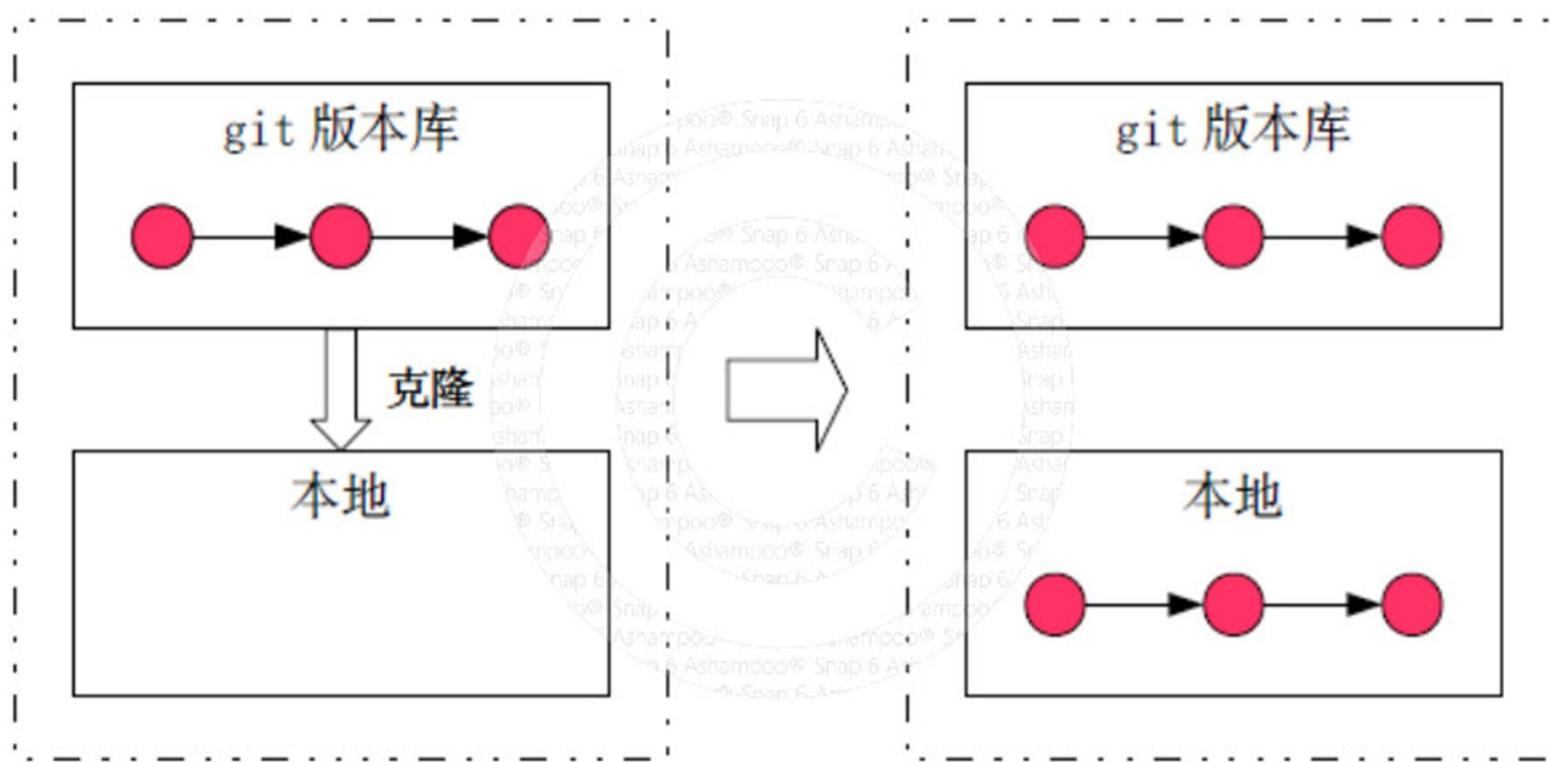
Git常用操作命令

- 1、同步操作
- 2、提交操作
- 3、查询操作
- 4、撤销操作

- 5、分支操作
- 6、标签操作
- 7、补丁操作
- 8、冲突解决

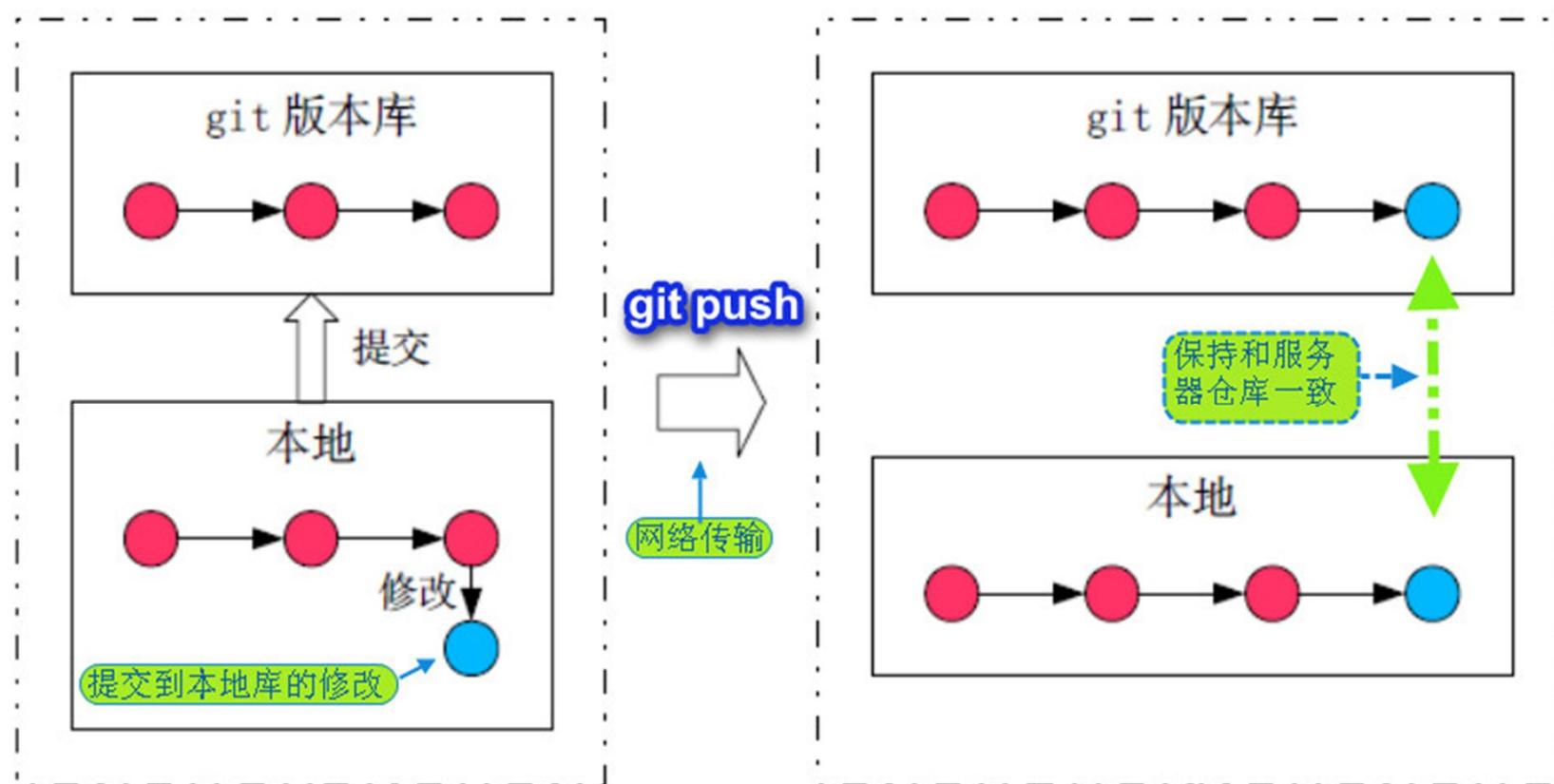
Git常用操作命令——同步操作

- `git clone` 复制一个版本库到指定文件夹



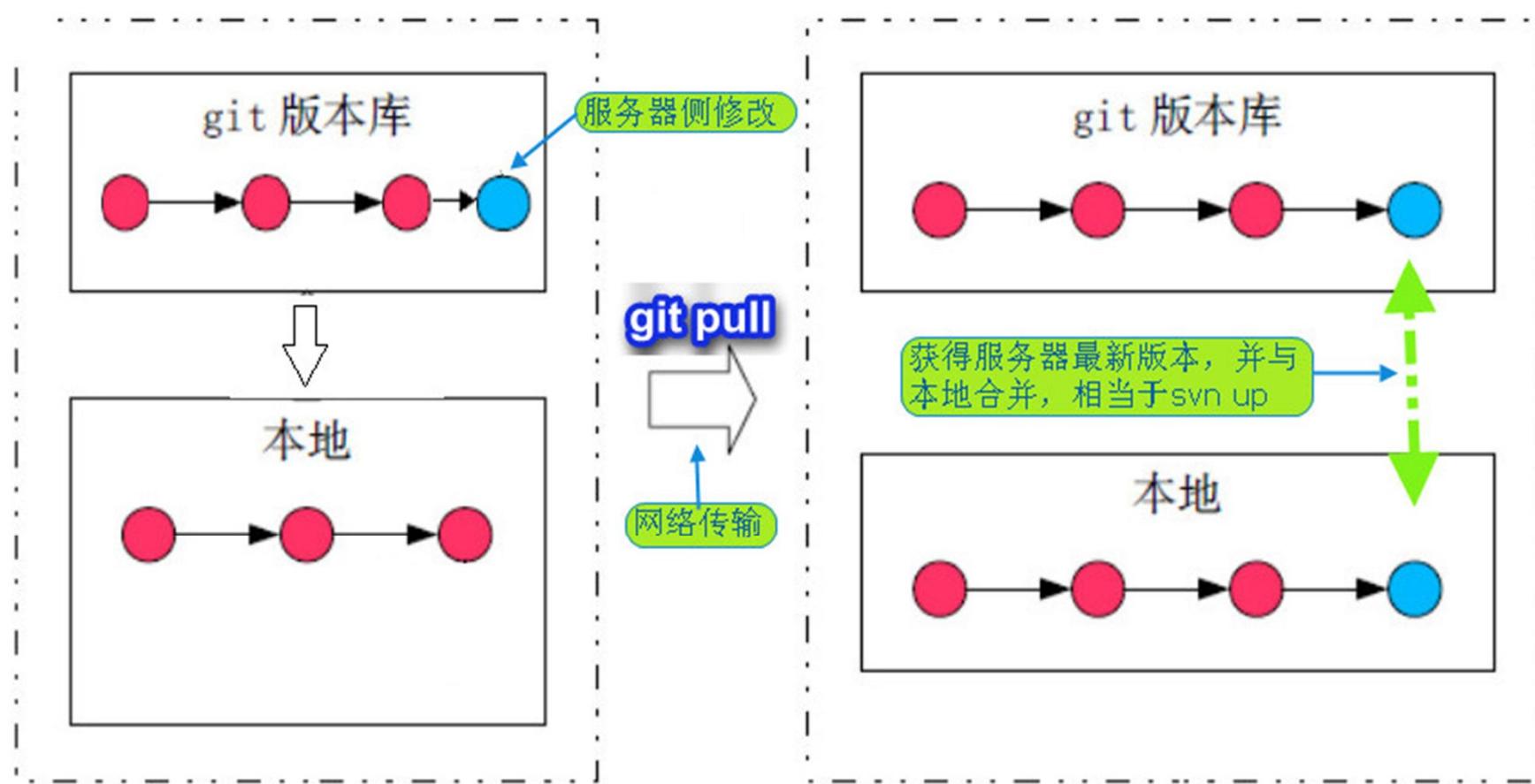
Git常用操作命令——同步操作

- git push 将本地内容推送到服务器版本库



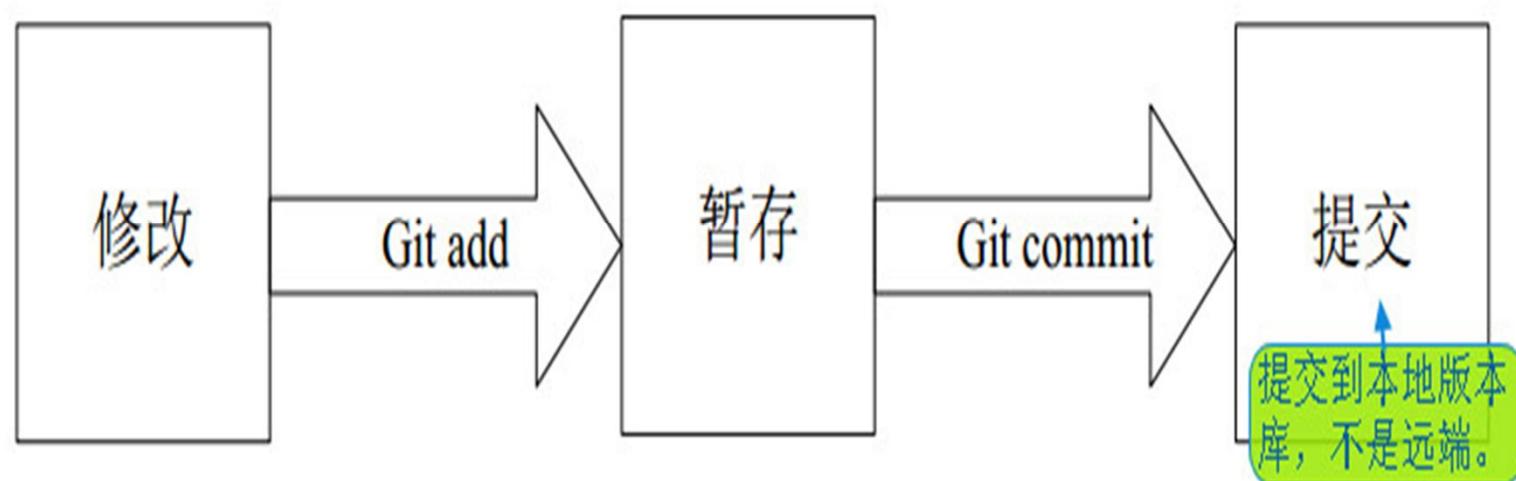
Git常用操作命令——同步操作

- git pull从服务器更新版本，并本地合并



Git常用操作命令——提交操作

- 修改代码后，本地版本库会有三种状态：
修改、暂存（标记）、提交。



Git常用操作命令——提交操作

- git add . //标记当前文件夹中所有内容
- git add filename //标记修改的文件
- git rm filename //标记删除的文件
- git commit -m “提交信息” [file]

备注：

- 1、git add操作是将修改或新增的文件放入暂存区，待提交。
- 2、git commit后面的文件名是可选，如果不写文件名，默认将暂存区的所有文件提交上去。如果写了文件名，则只针对该文件进行提交。

Git常用操作命令——查询操作

- git log 可以查看当前项目的日志

```
zhangchangqing@ZHANGCHANGQING /D/Git/test <master>
$ git log
commit 539d3e9b6c15100612403b778ceb92e93c3adf78
Author: zhangchangqing <zhangchangqing@hisensecom.com>
Date:   Thu Dec 13 19:09:03 2012 +0800
dd

commit 44f446a5fc59dc7e7c4f6cffa6e5c870fe2cf7f5
Author: zhangchangqing <zhangchangqing@hisensecom.com>
Date:   Thu Dec 13 19:06:44 2012 +0800
The directory

commit 52cd9be4c8ab9789f9c9d49e52dba14aa2881f51
Author: zhangchangqing <zhangchangqing@hisensecom.com>
Date:   Thu Dec 13 19:02:46 2012 +0800
test
```

作者和日期信息

上传记录的Log信息

SHA1 Hash码格式的Revision号，全球唯一

Git常用操作命令——查询操作

■ git log日志的导航

```
commit cfefbc16dc2a9e19213f073adbbf9487e298c4817
Author: zhangchangqing <zhangchangqing@hisensecom.com>
Date:   Tue Dec 11 18:01:29 2012 +0800

    Test branch

commit 686f71ec5fd57d04c038007f4557d5ef2b0900ca
Merge: 146724e f44f868
Author: zhangchangqing <zhangchangqing@hisensecom.com>
Date:   Tue Dec 11 17:56:04 2012 +0800

    On master: iii

commit f44f86871a763c62454091547a601f3206270cf2
Author: zhangchangqing <zhangchangqing@hisensecom.com>
Date:   Tue Dec 11 17:56:03 2012 +0800

    index on master: 146724e add D file

commit 146724e3efd4b84a8de0ce76827bd688e118b71c
Author: zhangchangqing <zhangchangqing@hisensecom.com>
Date:   Tue Dec 11 16:02:34 2012 +0800
```

如果历史记录较多的话，可以通过上下方向键浏览。
如果要退出，从键盘输入按键“q”

这里的冒号代表命令模式，输入 q 就可以退出查看哦

Git常用操作命令——查询操作

- git status 查询状态，包含分支情况和提交信息

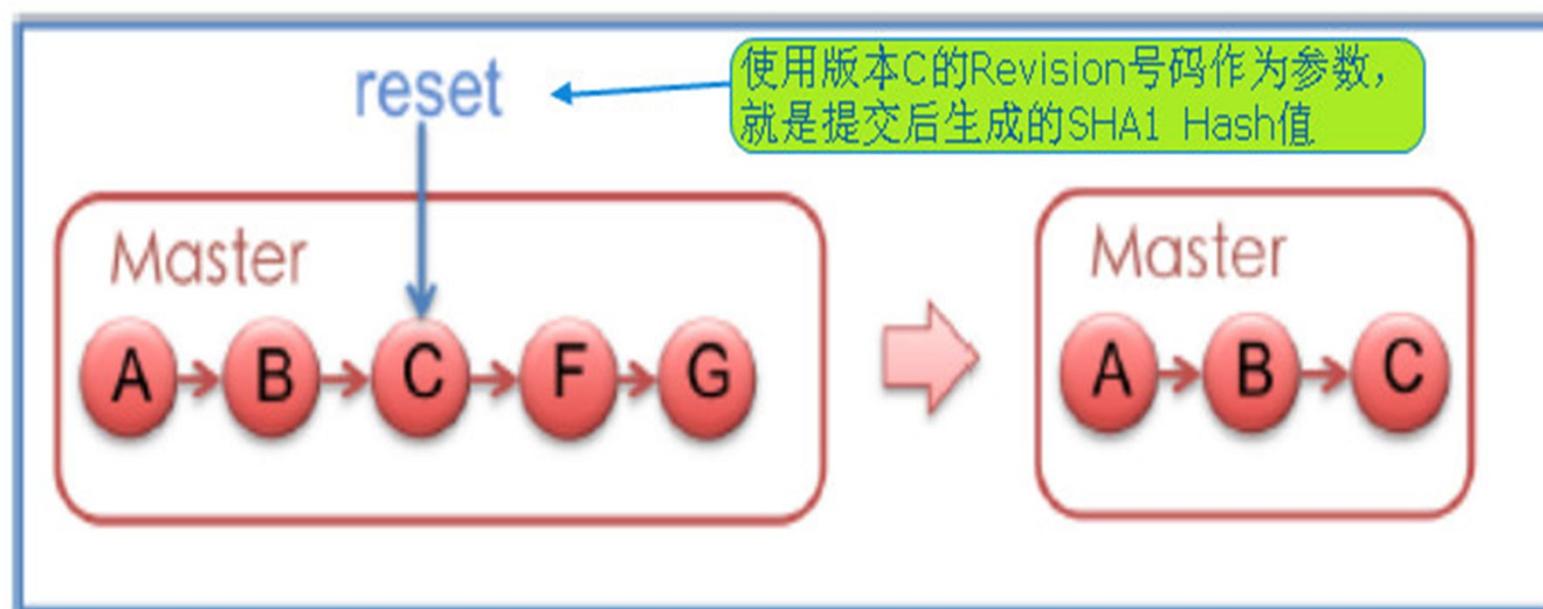
```
zhangchangqing@ZHANGCHANGQING /D/Git/test <master>
$ git status
# On branch master
# Your branch is ahead of 'origin/master' by 7 commits.
#
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   d.txt
#       new file:  e.txt
#
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   e.txt
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       f.txt
```

The diagram shows the output of a `git status` command with annotations explaining the different types of changes:

- 待提交的内容，已经存在暂存区**: Points to the first section of the output, which contains files `d.txt` and `e.txt` under the heading `Changes to be committed`.
- 已修改但未放入暂存区的内容**: Points to the second section, which contains file `e.txt` under the heading `Changes not staged for commit`.
- 未加入跟踪的文件**: Points to the third section, which contains file `f.txt` under the heading `Untracked files`.

Git常用操作命令——撤销操作

- git checkout [filename] //撤销修改的文件
- git reset //恢复到之前的版本



Git常用操作命令——分支操作

- git branch //查看当前分支
- git branch -D 分支名 //删除分支
- git checkout -b 分支名 //新建分支
- git checkout 分支名 //切换分支

commit	Author	Date
master modify hello.h	Jiang Xin <jiangxin@ossp.com>	2010-12-12 12:11:00
• add hello.h	Jiang Xin <jiangxin@ossp.com>	2010-12-07 19:39:10
• ignore object files.	Jiang Xin <jiangxin@ossp.com>	2010-12-07 19:21:39
• F modify hello.h	Jiang Xin <jiangxin@ossp.com>	2010-12-12 12:11:00
• E add hello.h	Jiang Xin <jiangxin@ossp.com>	2010-12-07 19:39:10
• D move .gitignore outside also works.	Jiang Xin <jiangxin@ossp.com>	2010-12-07 19:34:37
• C ignore object files.	Jiang Xin <jiangxin@ossp.com>	2010-12-07 19:21:39
• B hello_1.0 Hello world initialized.	Jiang Xin <jiangxin@ossp.com>	2010-12-07 18:33:45
• A README is from welcome.txt.	Jiang Xin <jiangxin@ossp.com>	2010-12-07 14:50:02
• restore file: welcome.txt	Jiang Xin <jiangxin@ossp.com>	2010-12-07 14:32:57

Git常用操作命令——标签操作

- git tag -a name -m "name" //添加新标签
- git checkout name //恢复某一标签版本
- git show tag-name //查看具体标签信息

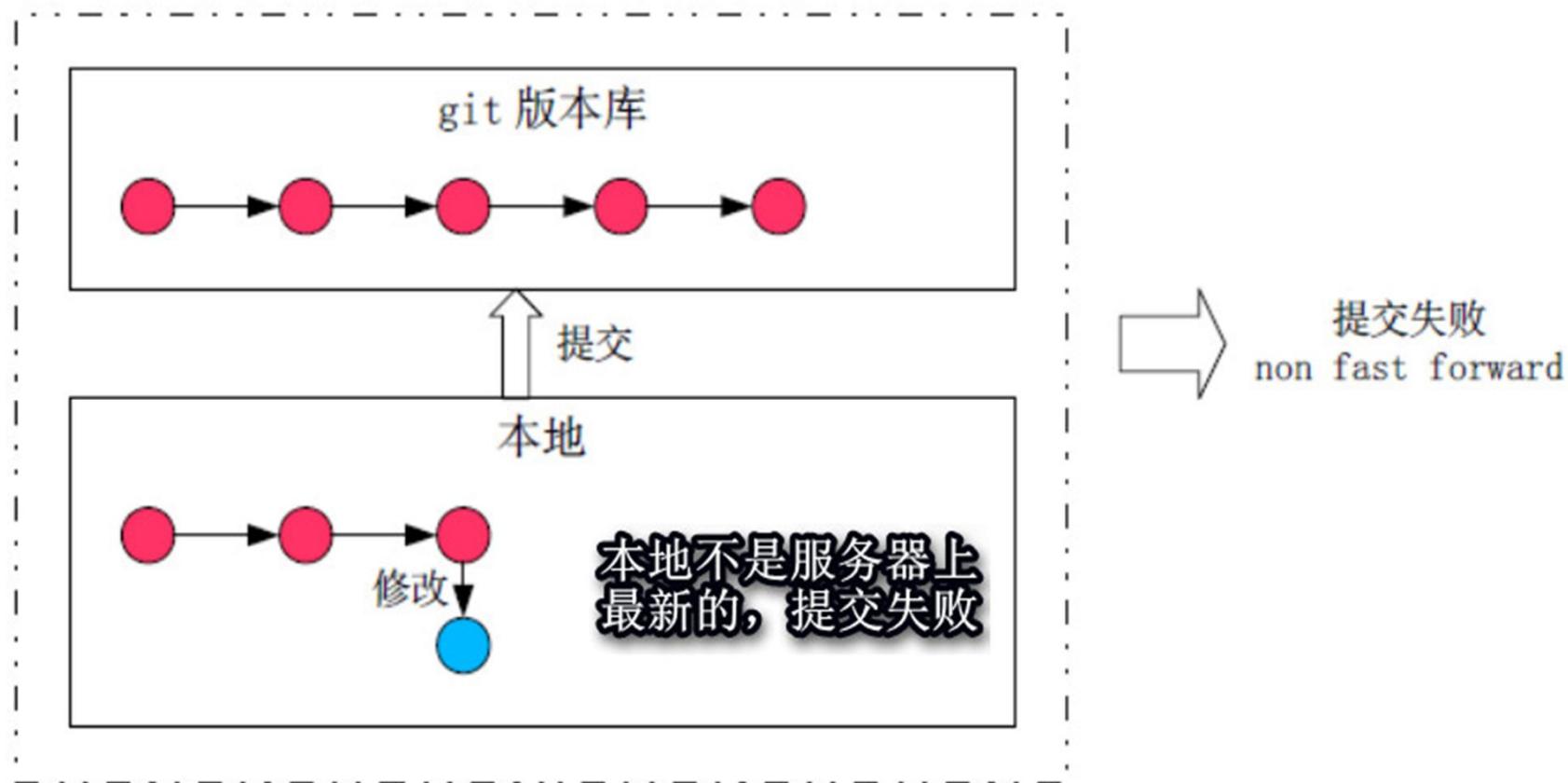


Git常用操作命令——补丁操作

- git diff 版本号 > patch
- git diff 分支名 > patch
- git format-patch 2e4182bad9ccad688cc...
- git apply *.patch //打上*.patch 补丁

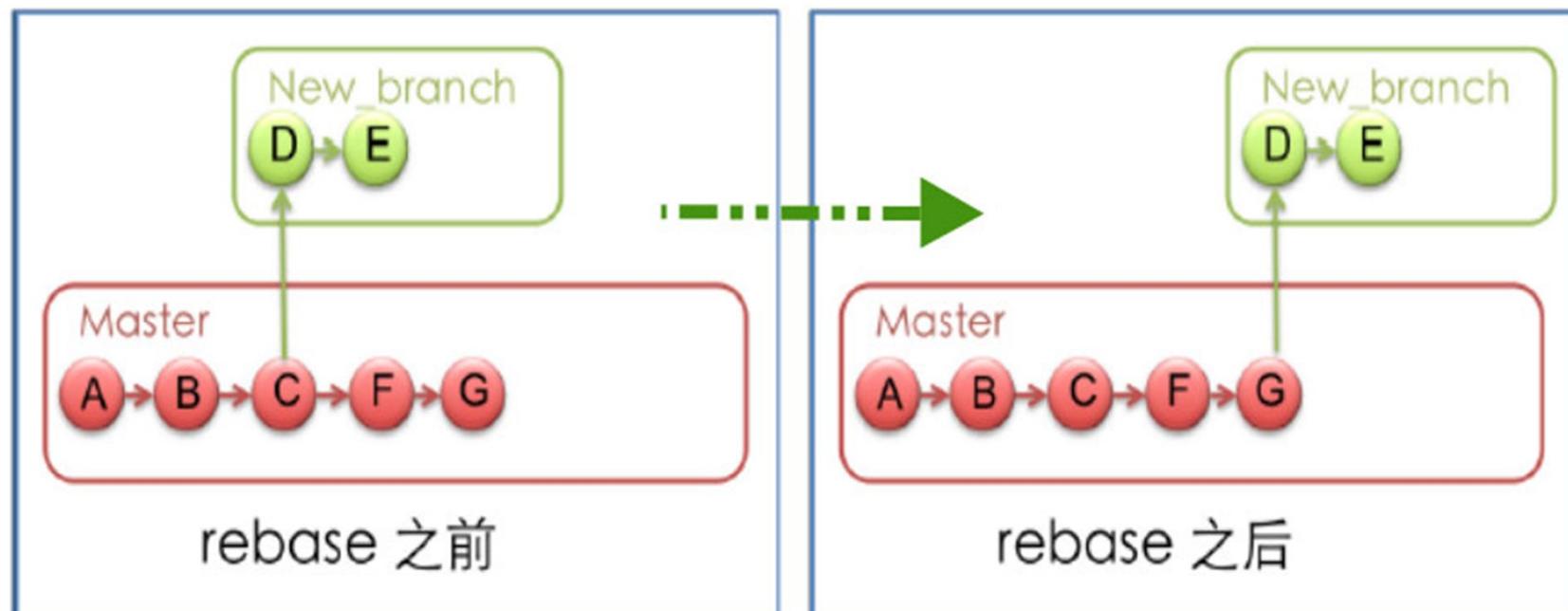
Git常用操作命令——冲突解决

- git rebase //重新定位一个基点



Git常用操作命令——冲突解决

- git rebase //重新定位一个基点



Git常用操作命令——常用命令

■ git help 【cmd】 //查看具体命令帮助

```
$ git help
usage: git [--version] [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p|--paginate|--no-pager] [--no-replace-object] [--depth <n>]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           [-c name=value] [--help]
           <command> [<args>]
```

The most commonly used git commands are:

add	Add file contents to the index
bisect	Find by binary search the change that introduced a bug
branch	List, create, or delete branches
checkout	Checkout a branch or paths to the working directory
clone	Clone a repository into a new directory
commit	Record changes, create a new commit object, and update refs
diff	Show changes between commits, commit and working tree, or worktree and index
fetch	Download objects and refs from another repository
grep	Print lines matching a pattern
init	Create an empty git repository or reinitialize an existing one
log	Show commit logs
merge	Join two or more development histories together
mv	Move or rename a file, a directory, or a ref
pull	Fetch from and merge with another repository's tip
push	Update remote refs along with associated objects
rebase	Forward-port local commits to the updated upstream head
reset	Reset current HEAD to the specified state

最常用的几个命令

git clone
git add
git commit
git push
git pull
git log
git status

TortoiseGit

Windows下的
图形化Git工
具，几乎和
TortoiseSVN
一样好用



TortoiseGit——安装步骤

➤ 安装Git工具 msysgit

下载：<http://code.google.com/p/msysgit>

文件：Git-1.7.11-preview20120710.exe

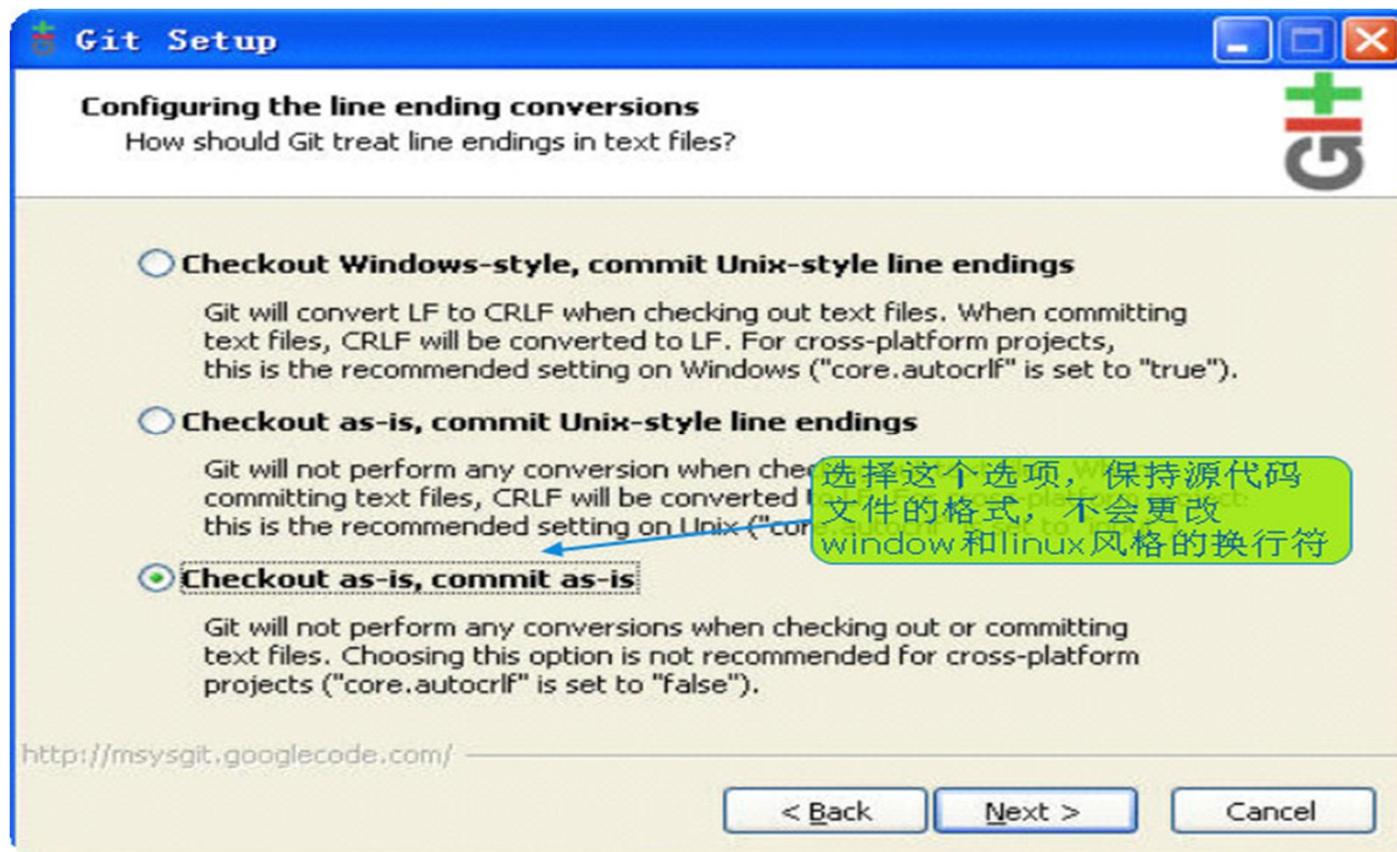
➤ 安装图形化界面 TortoiseGit

下载：<http://code.google.com/p/tortoisegit>

文件：Tortoisegit-1.6.5.0-64bit.msi

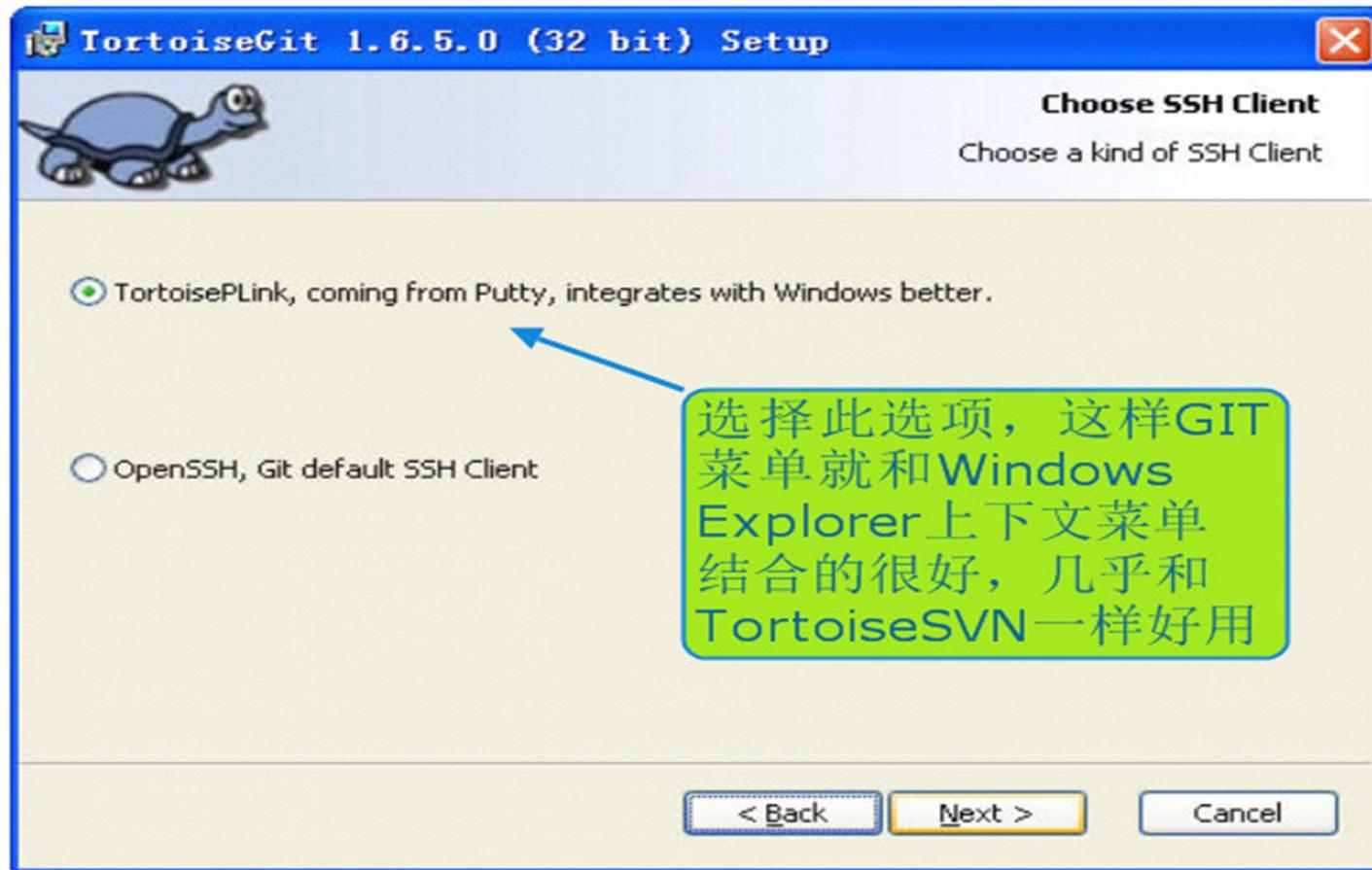
TortoiseGit——安装msysgit

安装过程中要注意，在设置行结束转换时，选择Checkout as-is, commit as-is，这样Git就不会修改换行风格了。其他用缺省设置即可。



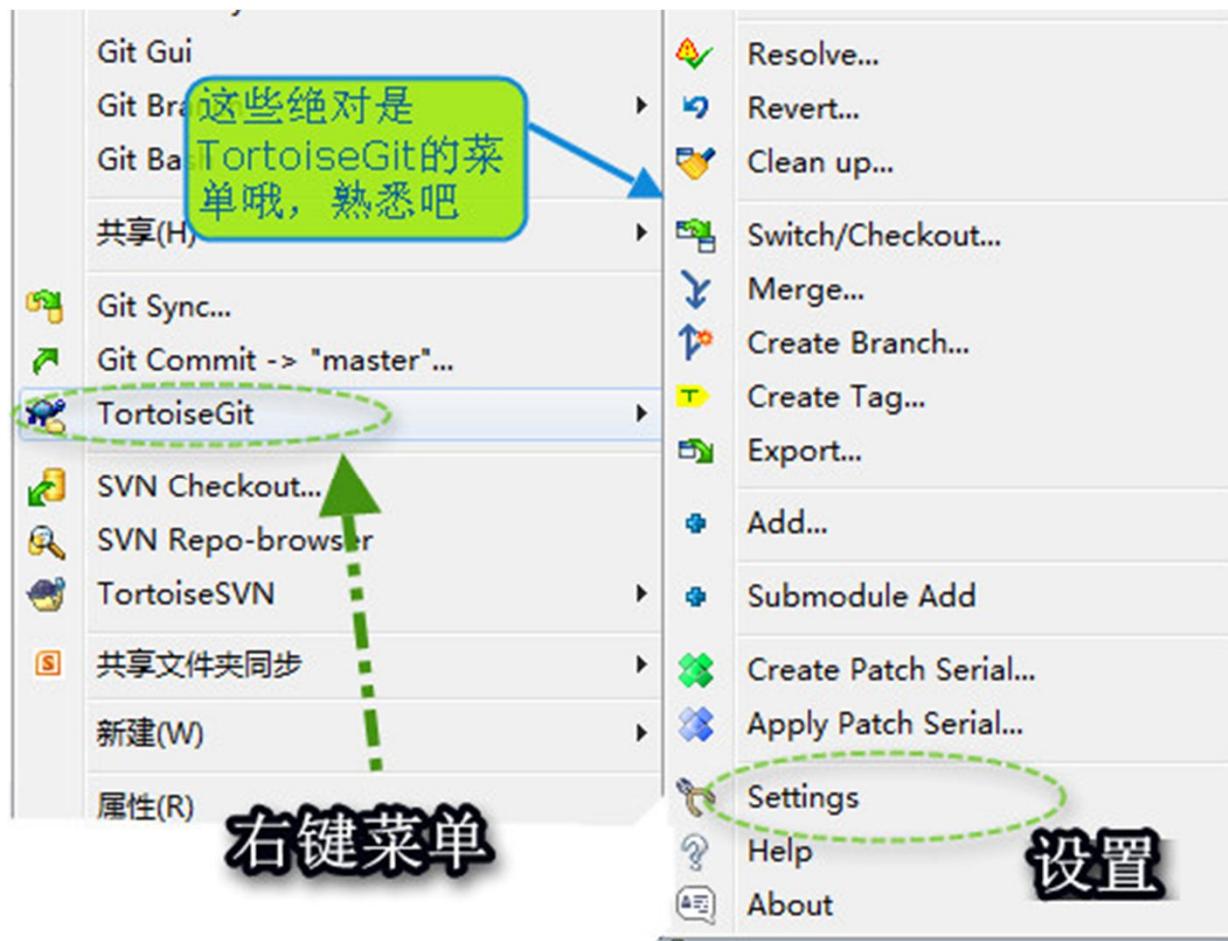
TortoiseGit——安装图形容户端

安装的过程中，选择TortoisePLink



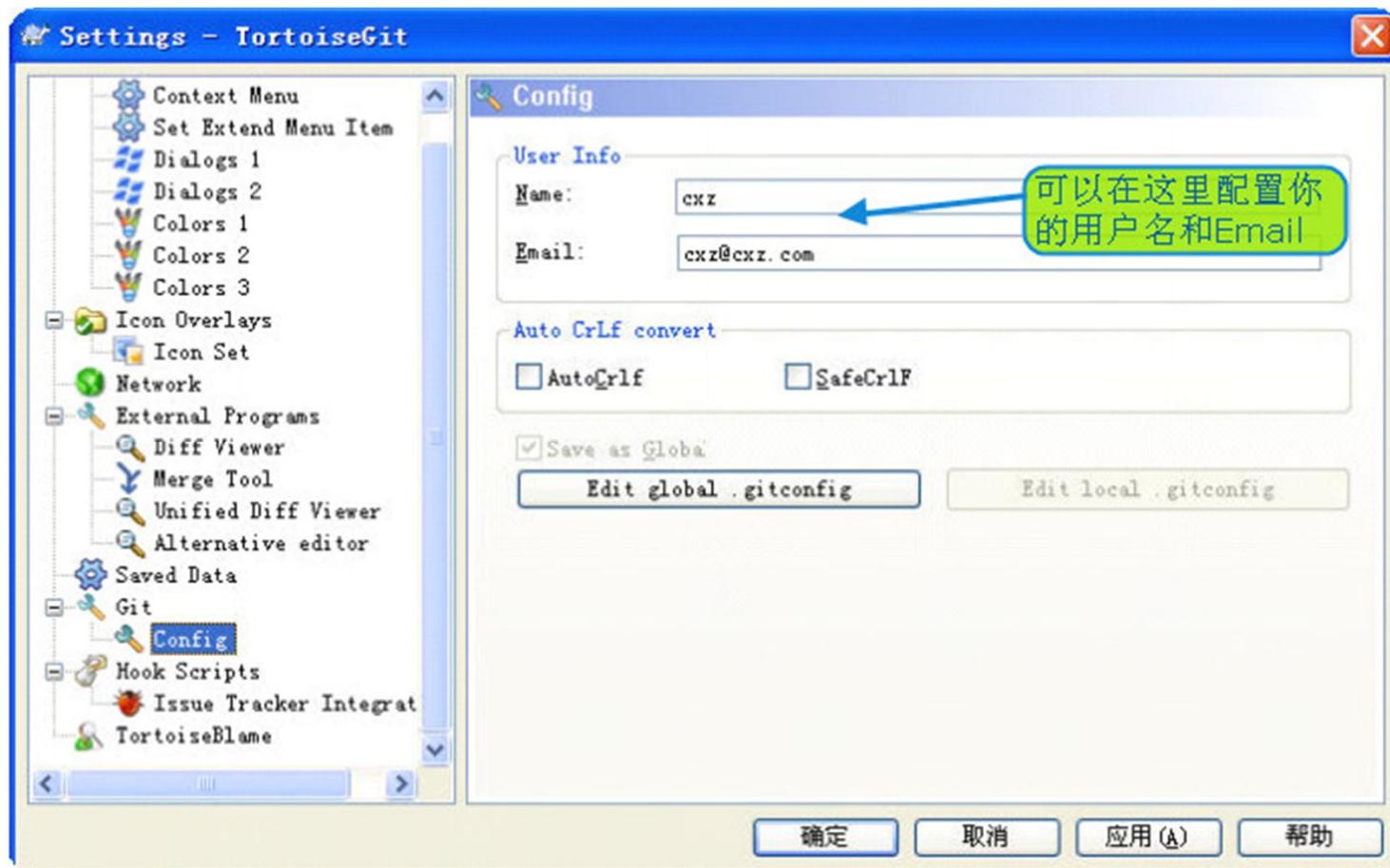
TortoiseGit——右键菜单

安装完毕后，TortoiseGit的入口在Windows的文件管理器中，

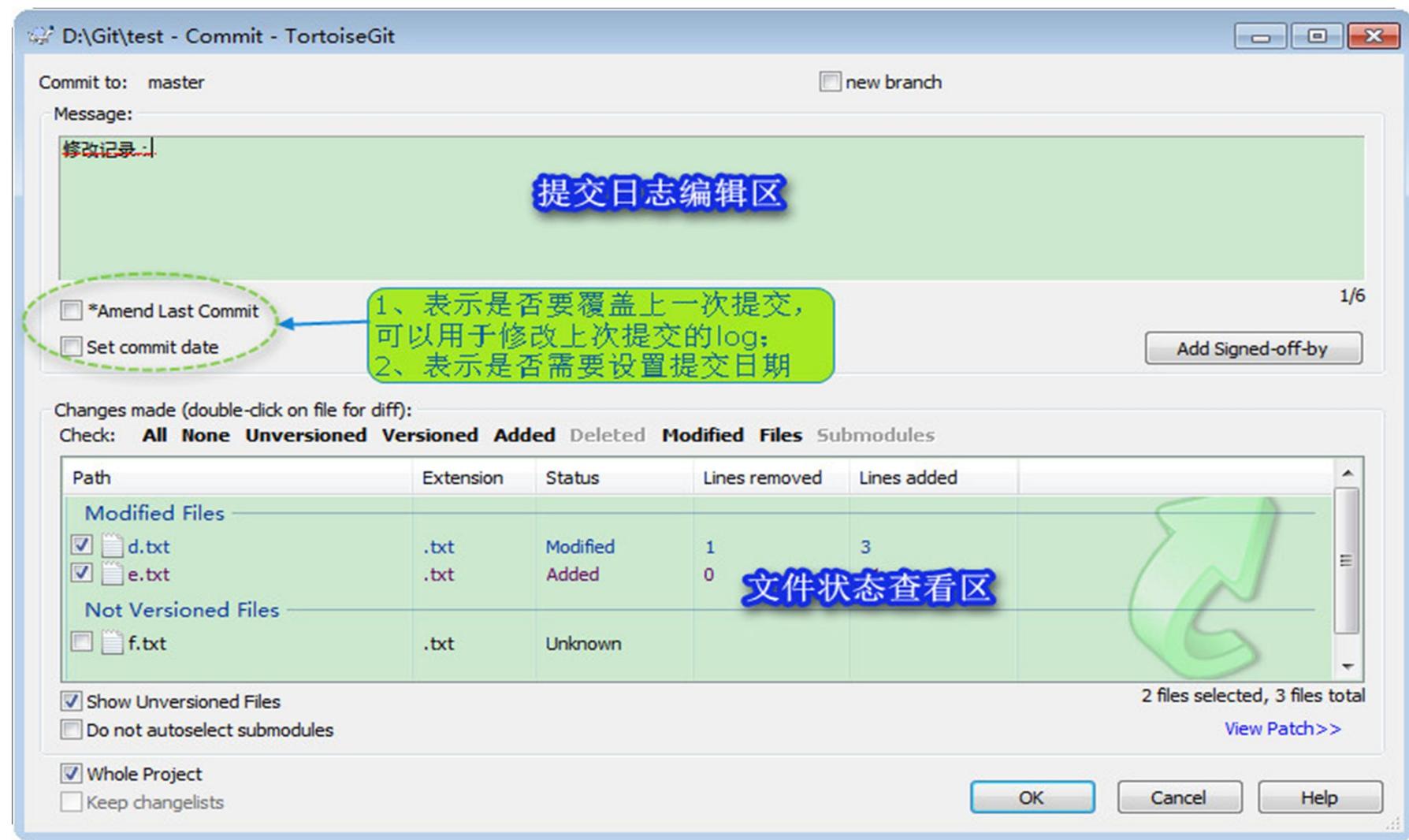


TortoiseGit——设置

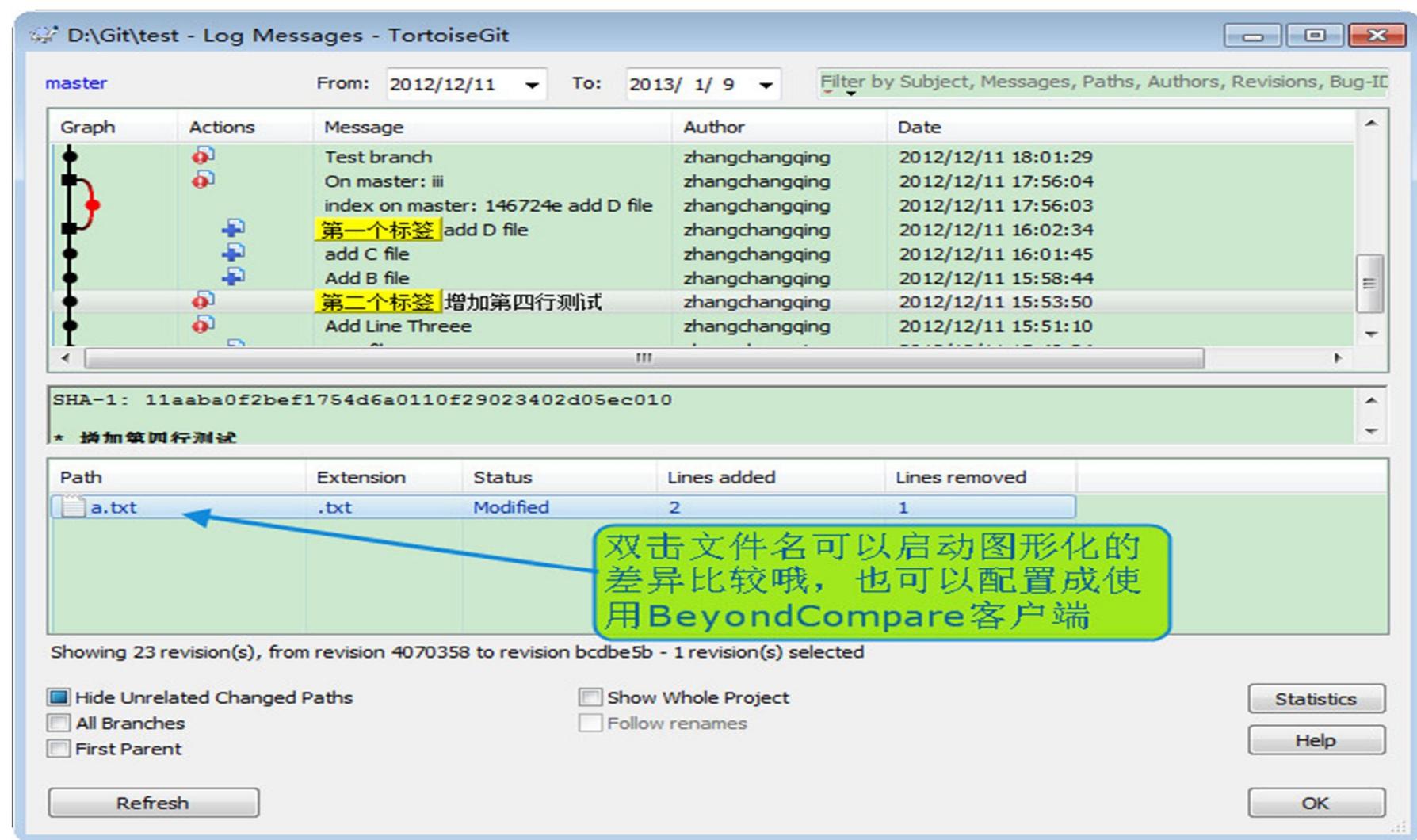
先到 TortoiseGit 程序组中调用 Settings 进行设置



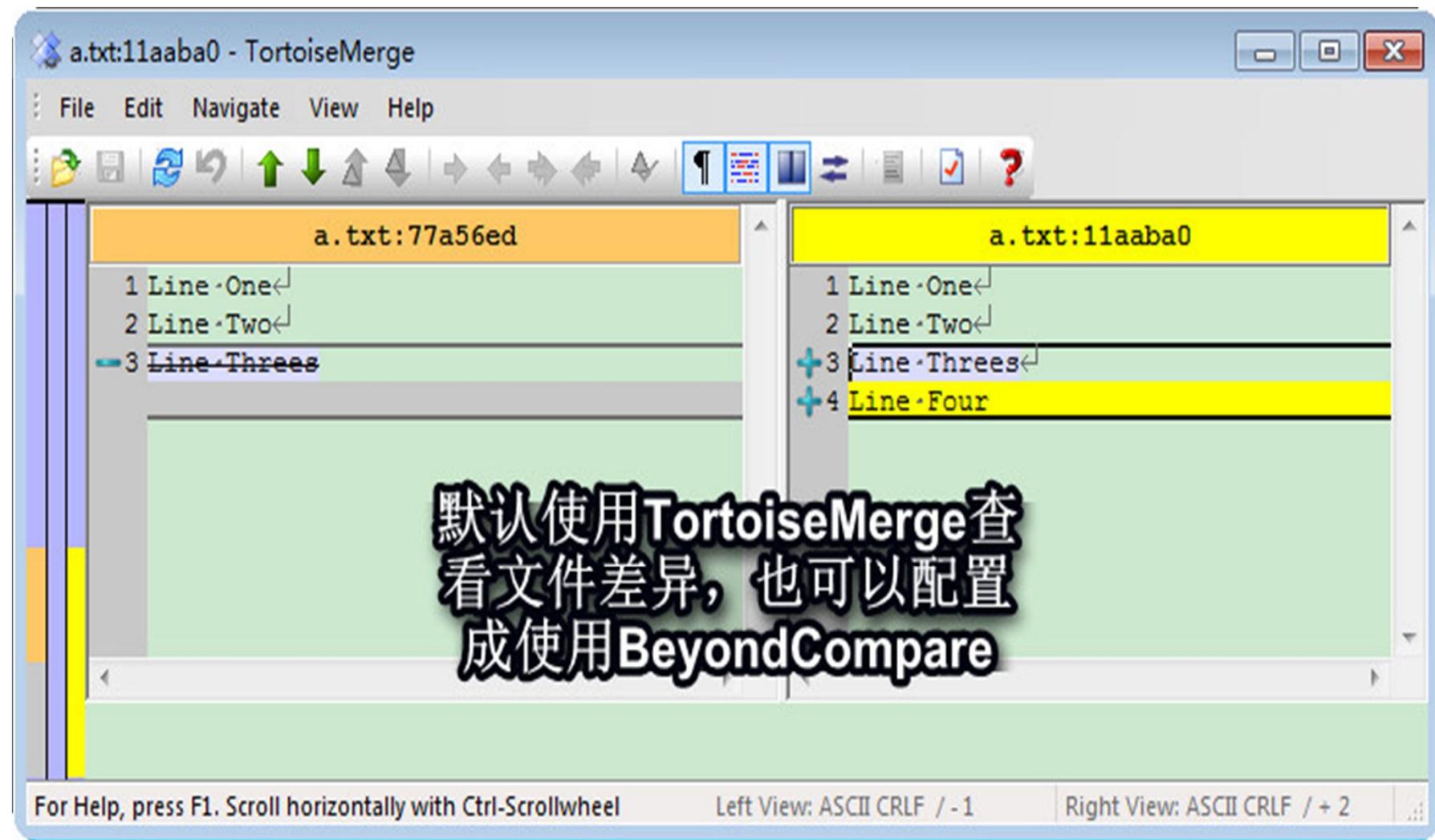
TortoiseGit——提交Commit



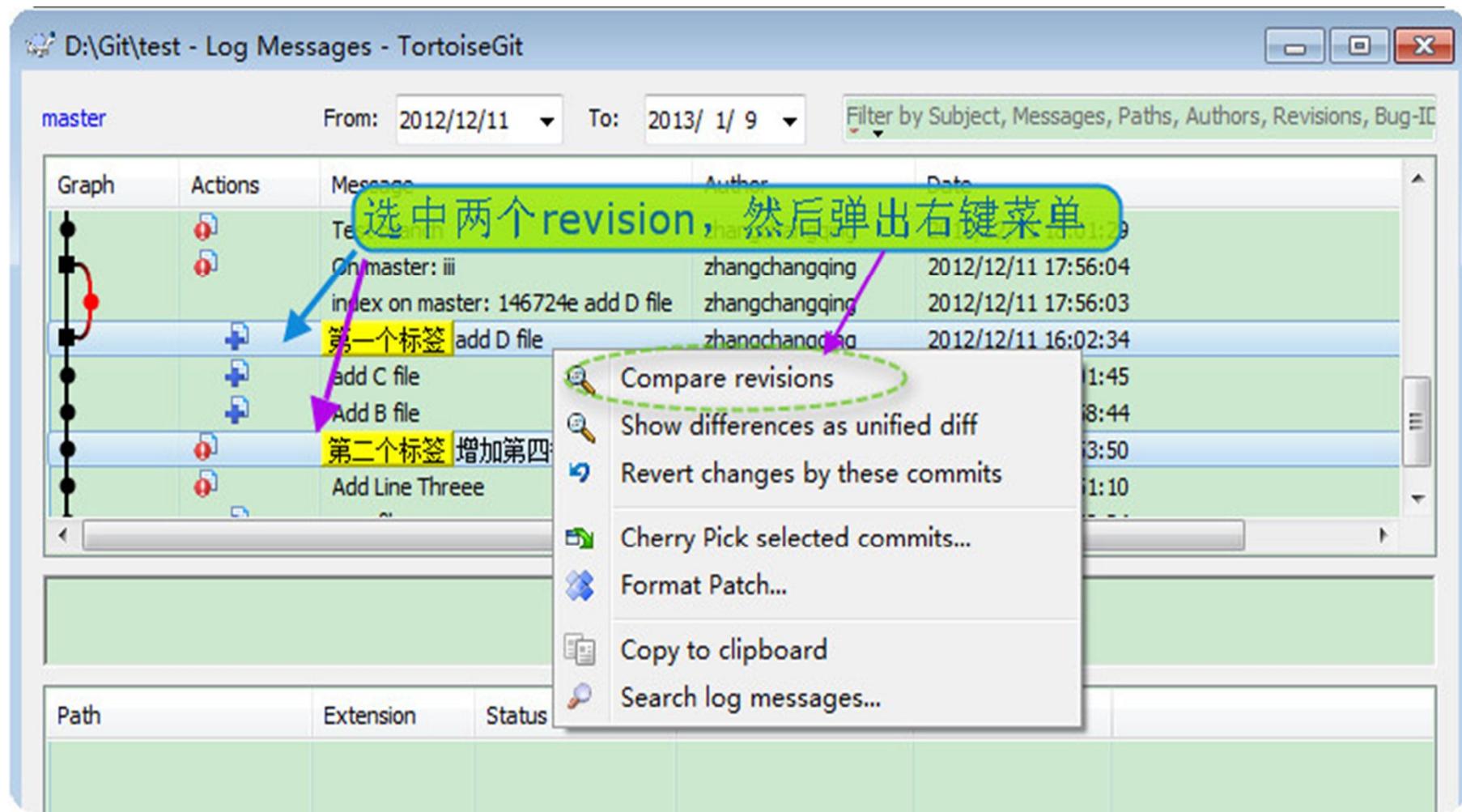
TortoiseGit——查看Log



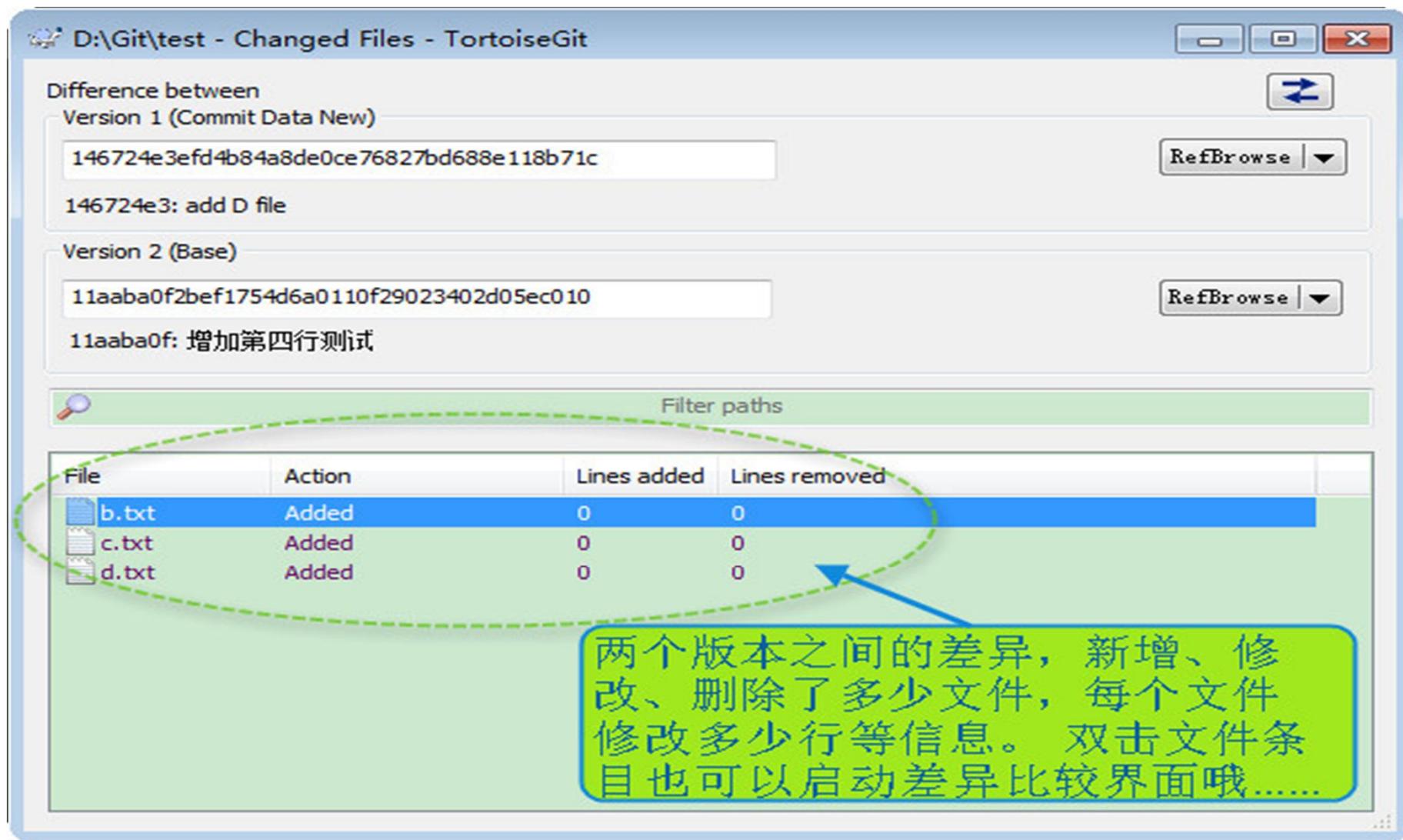
TortoiseGit——文件差异比较



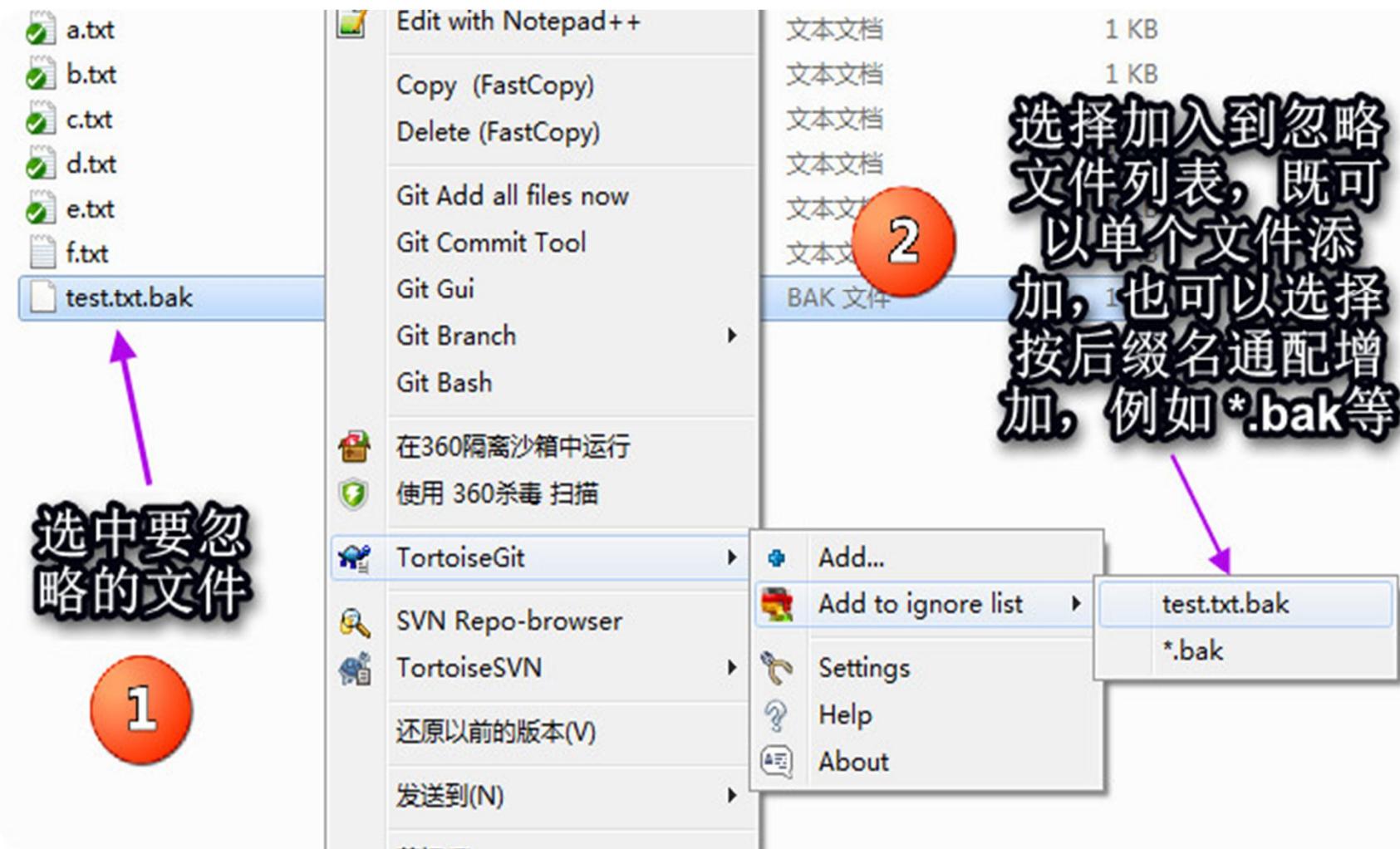
TortoiseGit——版本差异比较入口



TortoiseGit——版本差异比较界面



TortoiseGit——设置忽略文件



TortoiseGit——使用总结

- 图形化操作还是非常方便和高效的；
- 可以用于本地Git库的各种操作，例如看Log；
- 但是对于需要和远端服务器同步的操作，例如clone, push和pull等，尽量使用命令行执行。不仅仅因为效率比较高，还牵扯到使用SSH协议认证的问题。因为大家的SSH Key都是在Linux下生成的，如果在Windows下使用，需要重新配置一下。

Repo的使用

Repo是Google开发的用与管理Android版本库的工具，它是一个使用Python语言开发的命令行工具。Repo并不是用于取代Git，而是用Python对Git进行了一定的封装，简化了对多个Git版本库的管理。对于repo管理的任何一个版本库，都需要使用Git命令进行操作。

PS：我们最新下载的Android 4.2代码总共有339个独立的git库。
(每一个.git目录都是一个git仓库)

Repo的使用——更新repo库的步骤

- 1、下载repo库(如果知道工程对应的xml名，也可以不下载)

```
$git clone git@192.168.110.12/manifest.git
```

```
root@hisense-virtual-machine:~/work1/manifest/manifest# ls  
default.xml  y1test.xml  
root@hisense-virtual-machine:~/work1/manifest/manifest# █
```

- 2、下载对应的repo工程

查看manifest库中已经存在的工程，每个xml文件代表一个repo工程。使用下面的方式在当前目录下创建工程clone。

```
$repo init -u ssh://git@192.168.110.12/manifest.git -m y1test.xml
```

- 3、同步工程代码

```
$repo sync
```

Repo的使用——更新repo库的步骤

➤ 4、repo分支准备

刚下的repo工程不要急着用，先运行下面的语句，将代码与服务器的master分支同步

```
$repo start master -all
```

➤ 5、git库的递归操作

使用repo forall -c xxxx 可以遍历当前repo工程下所有的git库，并执行语句xxxx，例如：

✓ 遍历添加

```
$repo forall -c git add .
```

✓ 遍历提交log

```
$repo forall -c git commit -m "log xxx"
```

✓ 遍历push

```
$repo forall -c git push
```

Repo的使用——其他

➤ Q: 为什么要使用repo库?

A: 因为要和高通的库同步，高通就是按照repo的方法进行管理独立的git库，如果我们不一致，将无法把代码patch同步给他们。

➤ Q: 8x30 Repo库的地址和使用方法?

A: 目前我们的git服务器已经搭建完毕，repo也建设好。正在将所有的代码往服务器上同步，等建好并测试完成后，会发布给大家Step By Step的操作说明。

Git 实战

- 建立Git账户的步骤
- 配置你的用户名和Email
- 如何将代码发给高通？

Git使用——建立Git账户的步骤

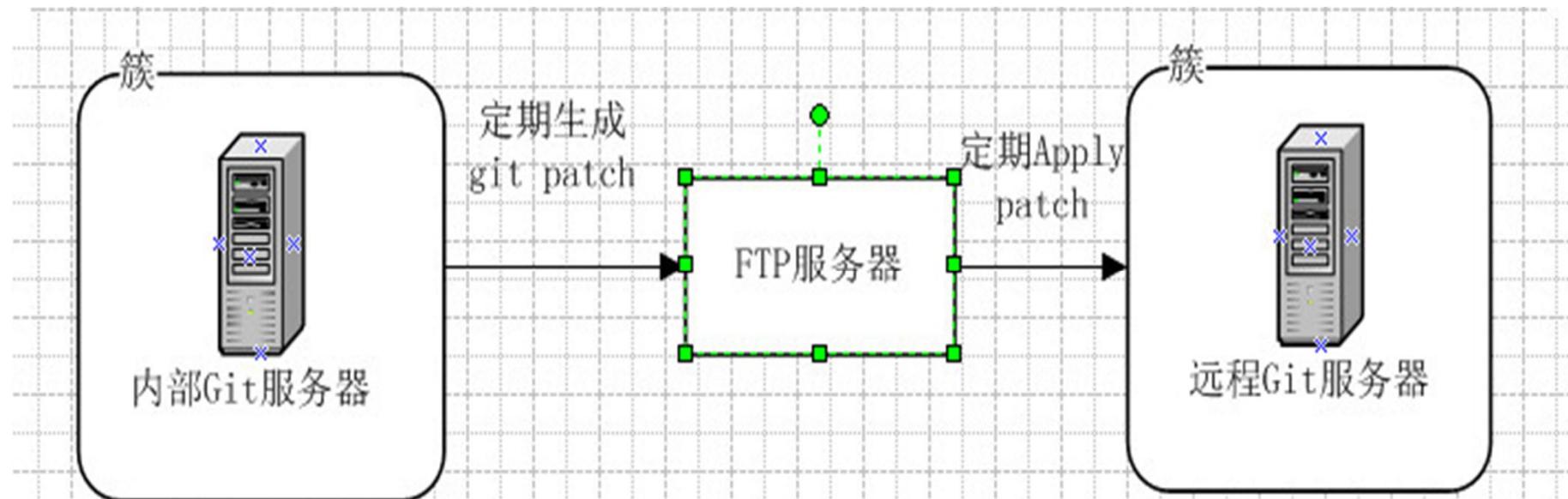
- 1、创建ssh密钥；
- 2、打开命令行终端，执行ssh-keygen，一路回车，中间不要改名；
- 3、在/root/.ssh下会生成两个文件id_rsa（私钥）和id_rsa.pub（公钥），将公钥文件**id_rsa.pub**文件发送给管理员即可，私钥文件id_rsa不要改动。如果丢失或损坏，则需要重新生成密钥并提交给管理员，建议保留一份私钥的备份。
- 4、管理员收到公钥文件后，在服务器上配置好账户，即可使用git与服务器交互。

Git使用——配置用户名和Email

Git 不喜欢不愿透漏姓名的人， 因为它要求每个人在向仓库提交数据时， 都应当承担一定的责任。要向 Git 进行自我介绍，请使用以下命令：

```
$ git config --global user.name "zhangchangqing"  
$ git config --global user.email zhangchangqing@hisensecom.com
```

Git使用——如何与高通同步代码



说明：以后发布给高通的都是一个一个独立
git库的patch文件，前期基本上一天一次，后
期两天一次。如果每个人都做，一定很繁琐，
幸运的是我们将开发脚本，自动执行该操作。

总结

- 对于用惯SVN来说，Git是一种思维方式的转变——“每个人都是一个仓库”；
- 很显然，这次培训不可能让你成为Git专家，但希望这是一个好的起点；
- 亲自去实践一下，尝试一下，才会有感觉；

親：亲眼所见才会亲，经常见面才会亲

Q & A

Thank you for your time !