Subversion 使用

SVN 从入门到出家

罗流毅

xluoly@gmail.com

2018-7-26

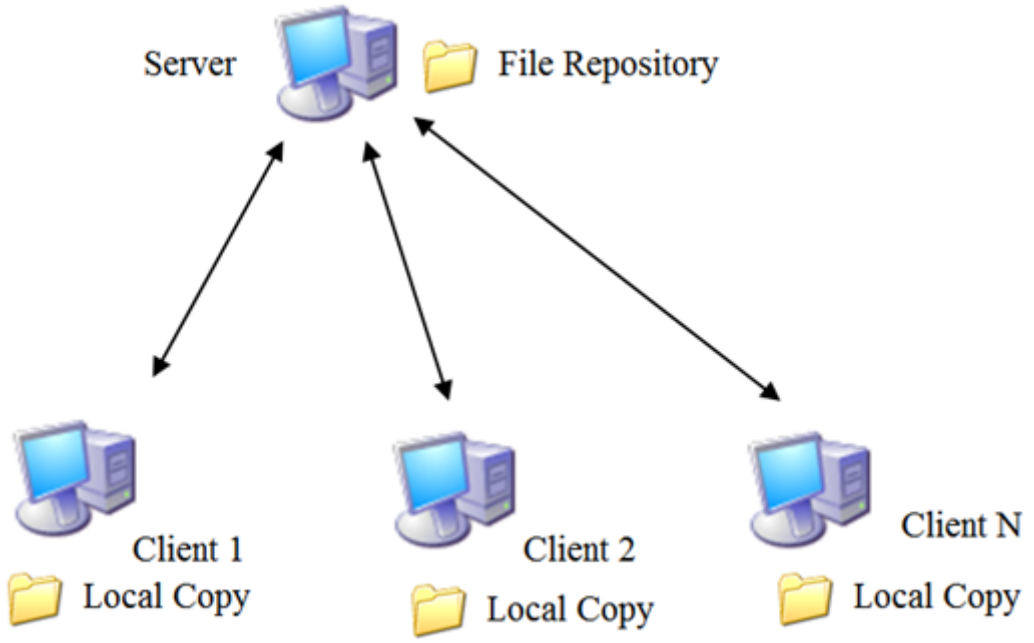
Agenda

基本概念

- Subversion 简称 SVN
- Subversion 被设计为 CVS 的替代产品
- 一个开放源代码的版本控制系统 (VCS)
- Apache 软件基金会的一个顶级项目

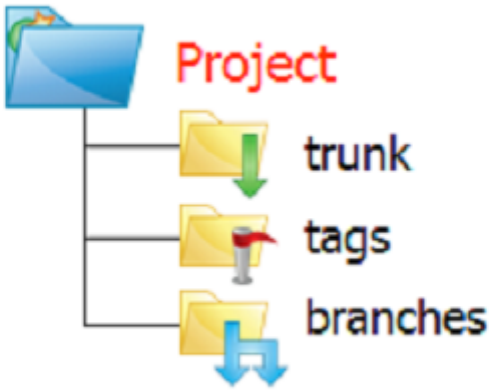
SVN Architecture

- 集中式版本控制系统
- Repository - 版本库
- Working Copy - 工作副本

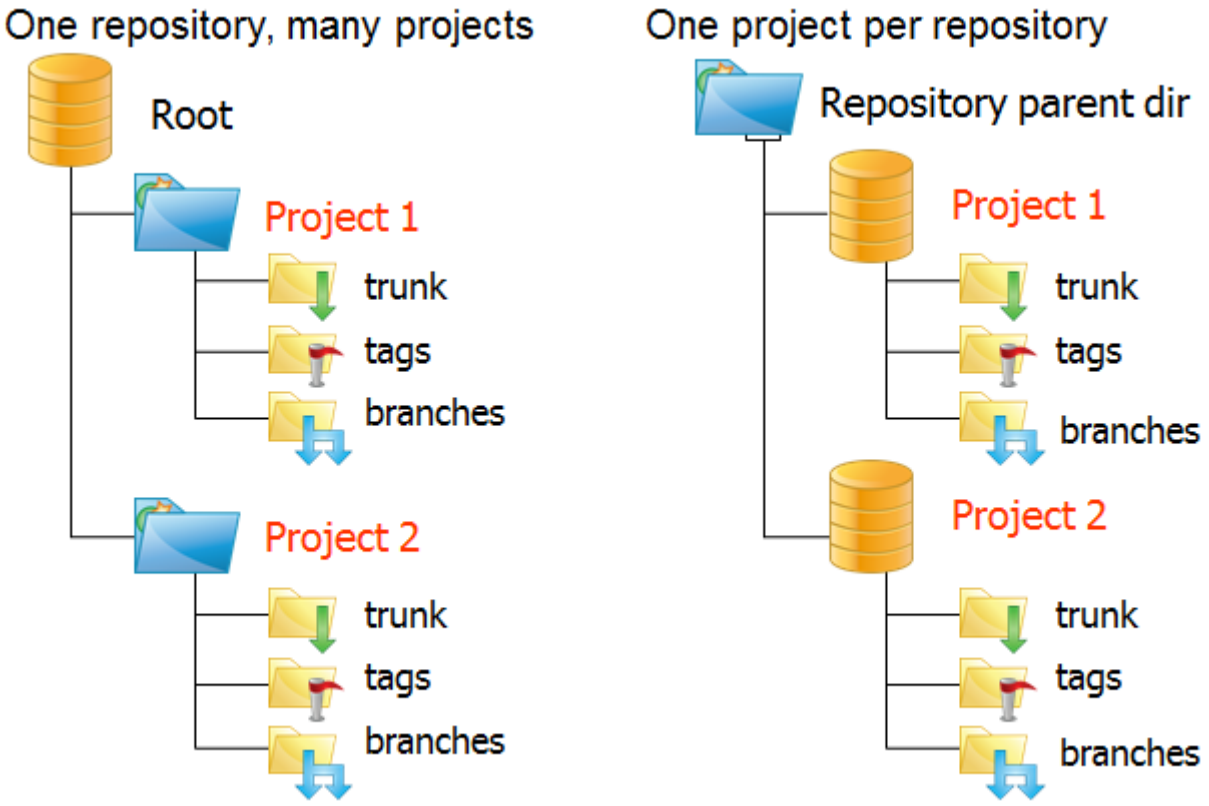


SVN Directories

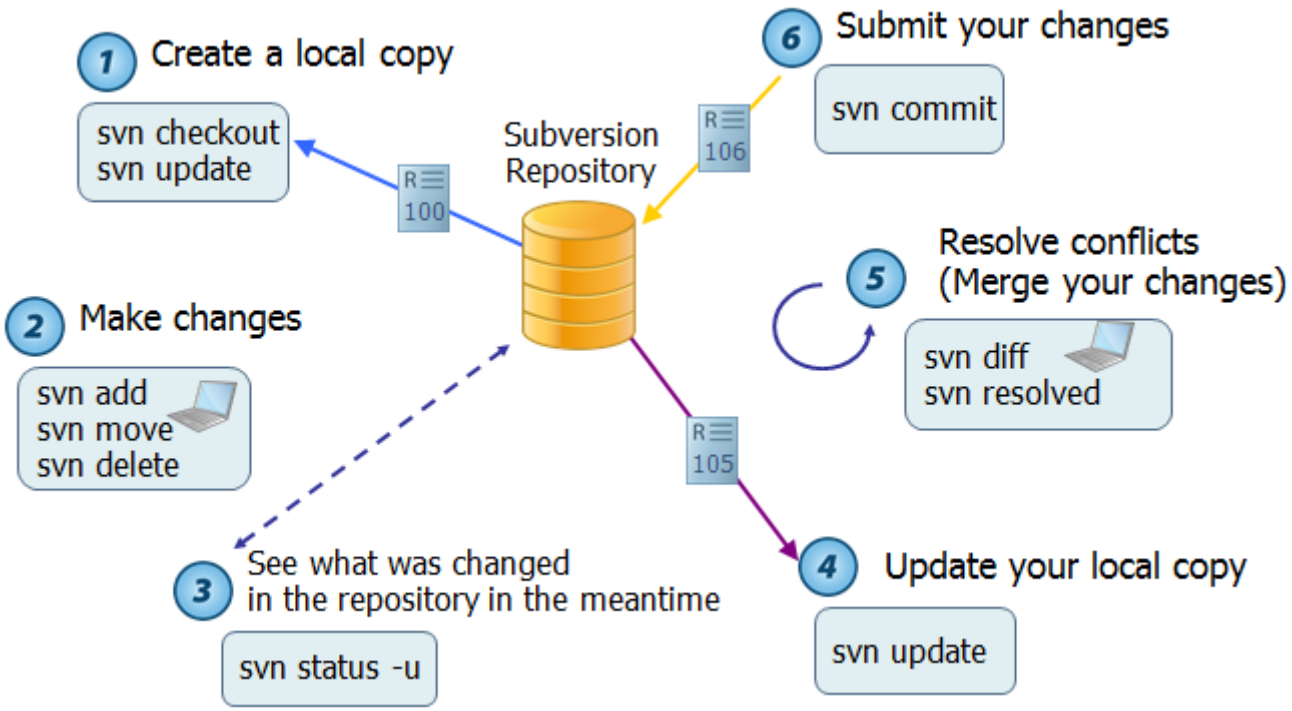
- trunk: 主工作分支
- branches: 放置从主分支分离出来的其他工作分支
- tags: 放置版本发布标签，不应该向这里提交修改



SVN Repository Layout



Work Cycle

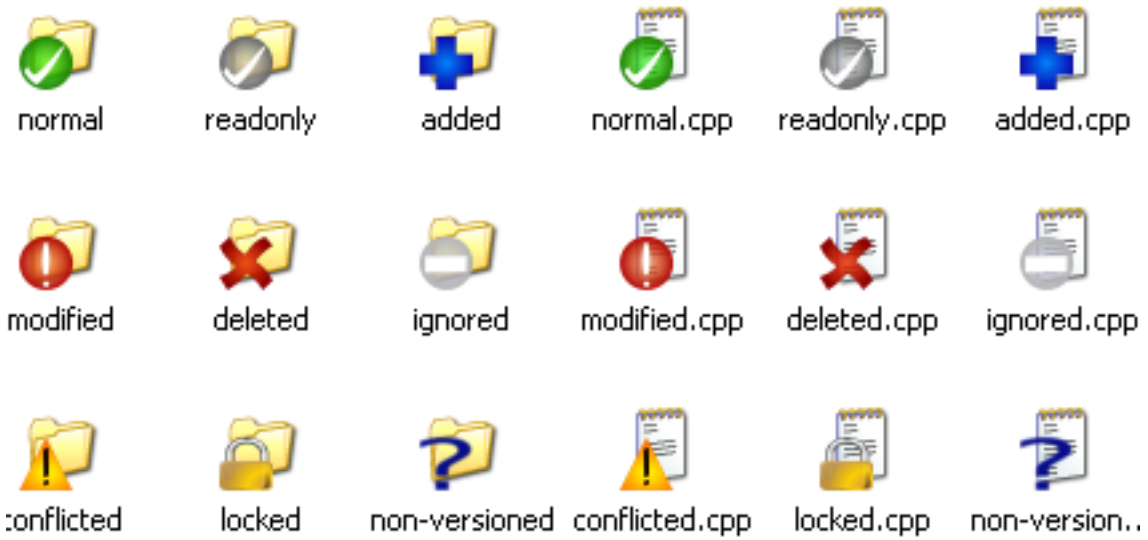

1 Create a local copy
svn checkout
svn update
2 Make changes
svn add
svn move
svn delete
3 See what was changed
in the repository in the meantime
svn status -u
4 Update your local copy
svn update
5 Resolve conflicts
(Merge your changes)
svn diff
svn resolved
6 Submit your changes
svn commit
Subversion
Repository
R 100
R 105
R 106

客户端工具

- Command Line
- Graphical
  - TortoiseSVN

- IDE Integration
  - IntelliJ IDEA
  - Eclipse

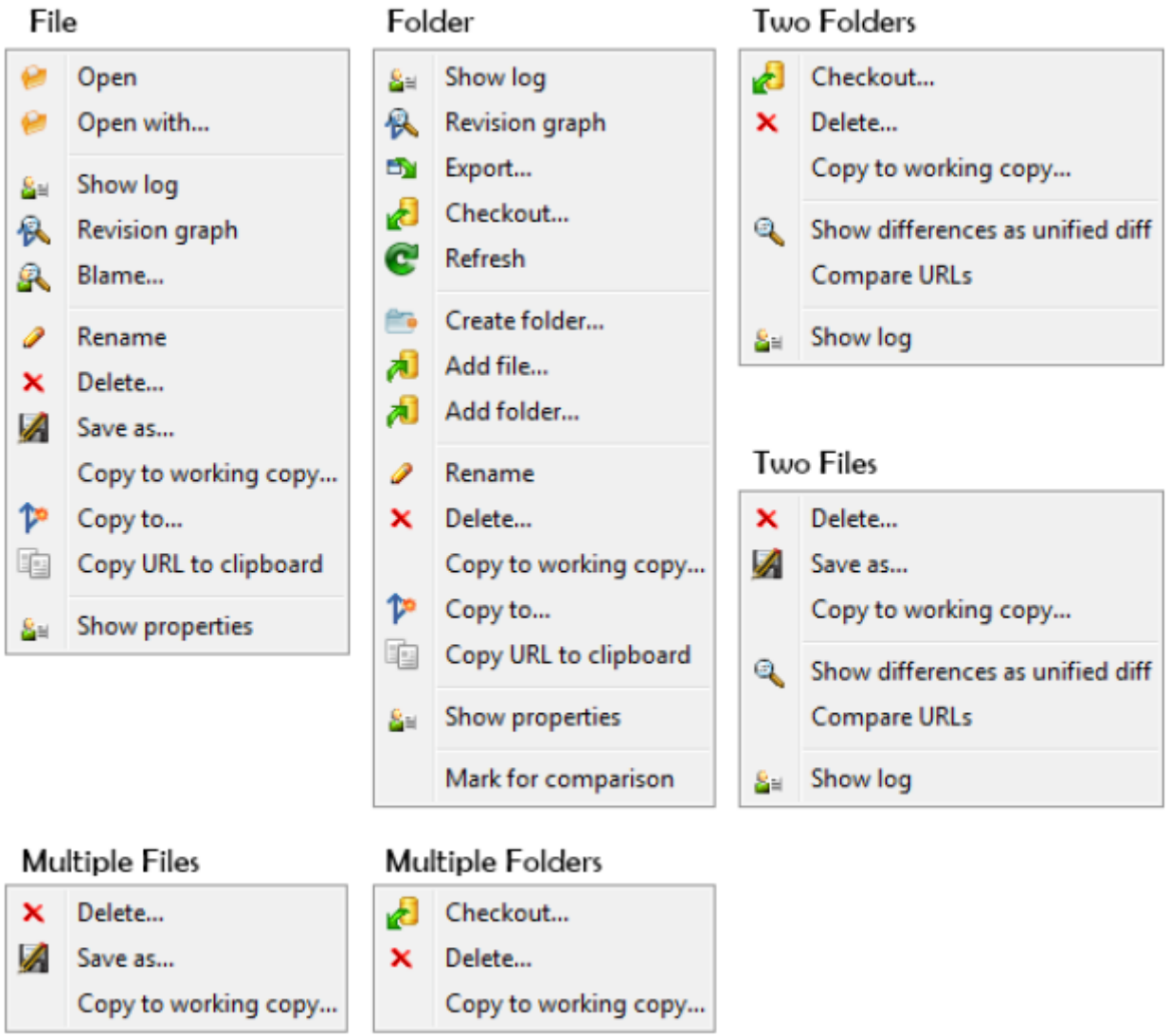
TortoiseSVN Icon Overlay



Agenda

TortoiseSVN 的右键菜单

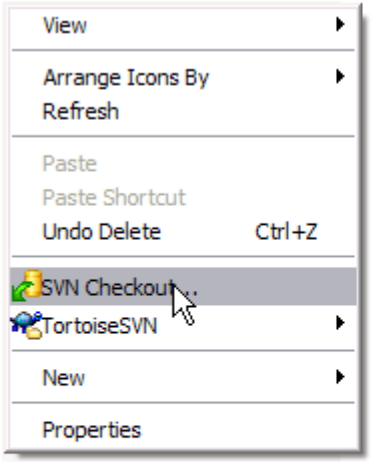
- TortoiseSVN 的所有操作都是从右键菜单启动

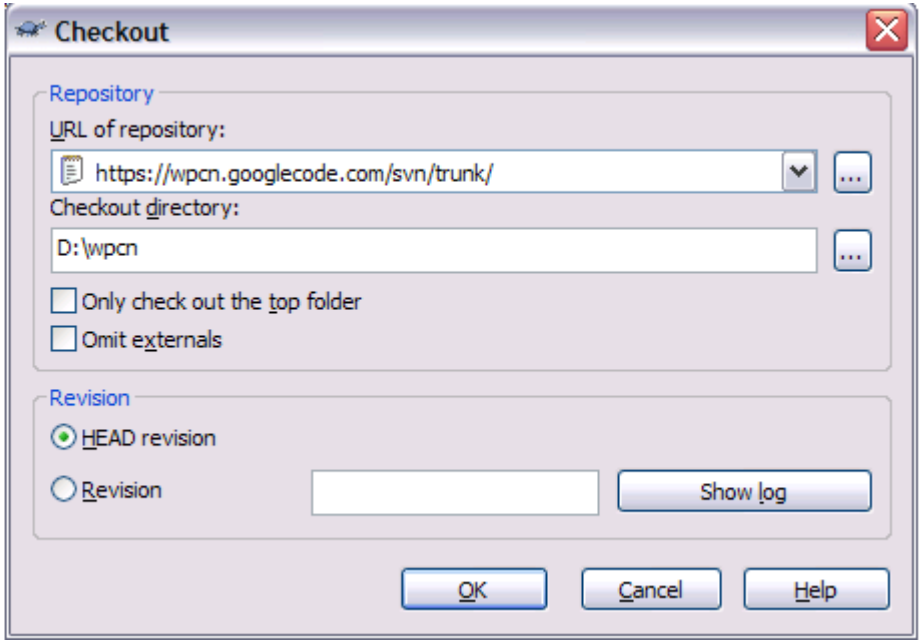


Checkout

- 从版本库中检出文件到当前目录创建工作副本

```
$ svn checkout https://server/svn/project/trunk
$ svn co https://server/svn/project/trunk
```





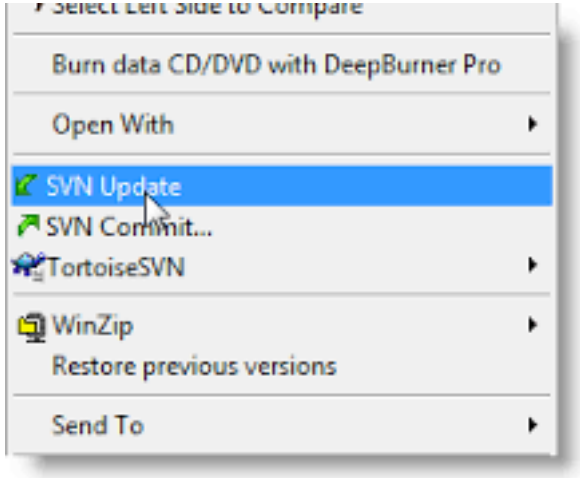
Update

- 将工作副本同步到版本库的最新状态

```
$ svn update
$ svn up
```

- 将工作副本同步到指定版本

```
$ svn update -r 100
```



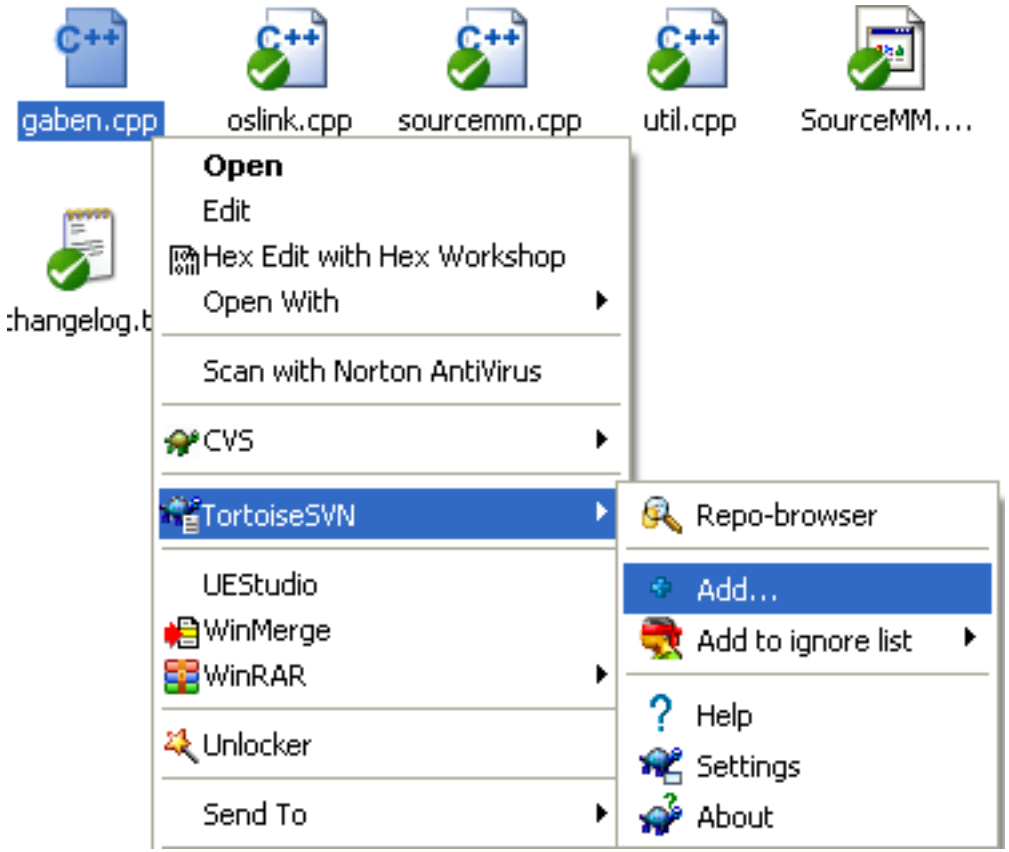
Add

- 向版本库添加单个文件

```
$ svn add gaben.cpp
```

- 添加目录下的所有文件

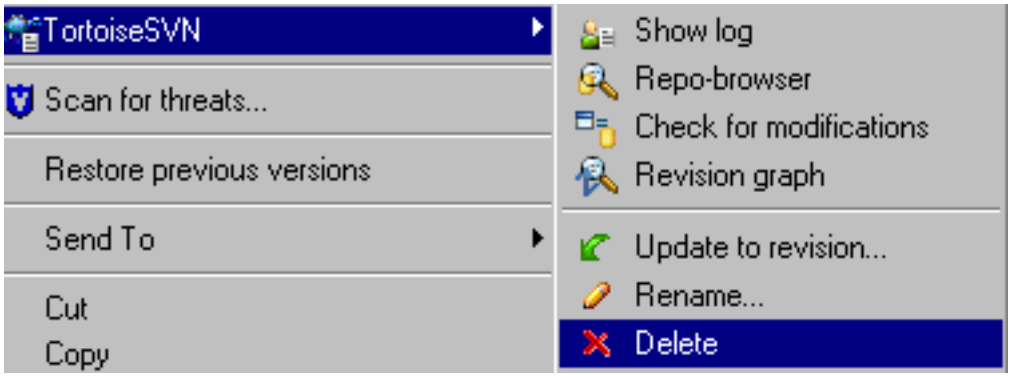
```
$ svn add dir
```



Delete

- 将文件从版本库中移除

```
$ svn delete foo.c
$ svn del foo.c
$ svn remove foo.c
$ svn rm foo.c
```



Copy

- 复制文件

```
$ svn copy sourcefile destfile
$ svn cp sourcefile destfile
```

- TortoiseSVN Copy-Paste
  1. Windows 文件浏览器右键菜单 -> 复制 (Copy)
  2. TortoiseSVN 右键菜单 -> Paste
- TortoiseSVN 右键拖动
  1. 鼠标右键选中要复制的文件
  2. 按住鼠标右键的同时将文件拖动到目标文件夹
  3. 在弹出的右键菜单中选择 `SVN Copy versioned item(s) here` 或者 `SVN Copy and rename versioned item here`

Move

- 移动文件

```
$ svn move sourcefile destfile
$ svn mv sourcefile destfile
$ svn rename sourcefile destfile
$ svn ren sourcefile destfile
```

- TortoiseSVN
  1. Windows 文件浏览器右键菜单 -> 剪切 (Cut)
  2. TortoiseSVN 右键菜单 -> Paste
- TortoiseSVN 右键拖动
  1. 鼠标右键选中要复制的文件
  2. 按住鼠标右键的同时将文件拖动到目标文件夹
  3. 在弹出的右键菜单中选择 `SVN Move versioned item(s) here` 或者 `SVN Move and rename versioned item here`

Status

- 查看版本库相对于工作副本的最新修改

```
$ svn status -u
```

- 查看工作副本的修改状态

```
$ svn status
$ svn st
```

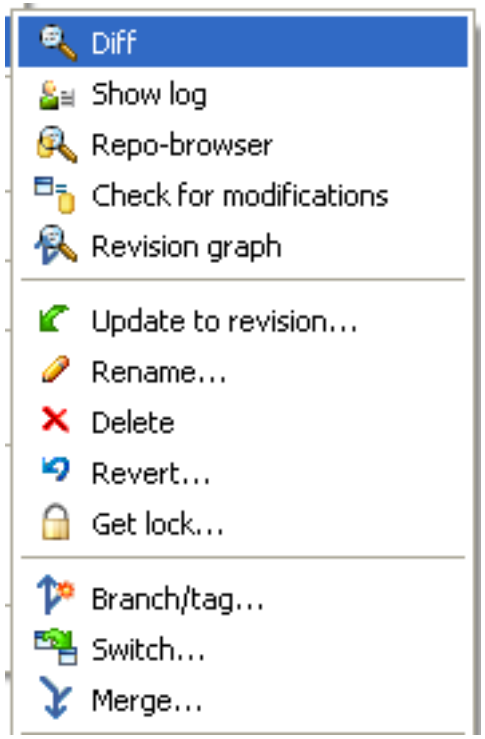
- 输出状态含义

```
A - Add
C - Conflict
D - Delete
M - Modify
? - Unversioned
! - Missing
```

Diff

- 显示文件的本地修改差异

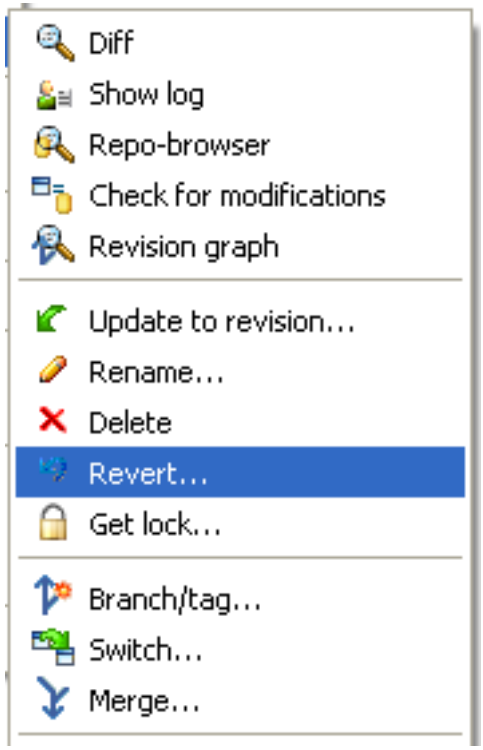
```
$ svn diff foo.cpp
$ svn di foo.cpp
```



Revert

- 撤销/放弃工作副本所做的修改
- 放弃新增加文件

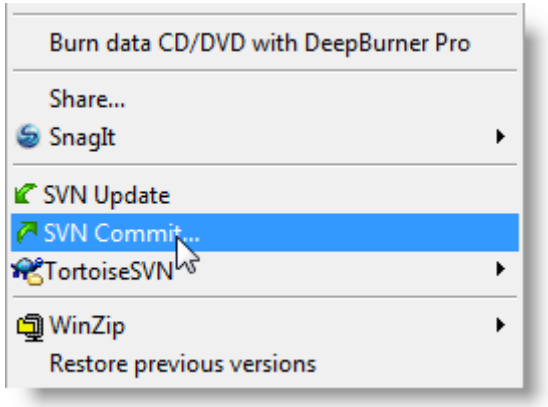
```
$ svn revert foo.cpp
```



Commit

- 提交工作副本的修改

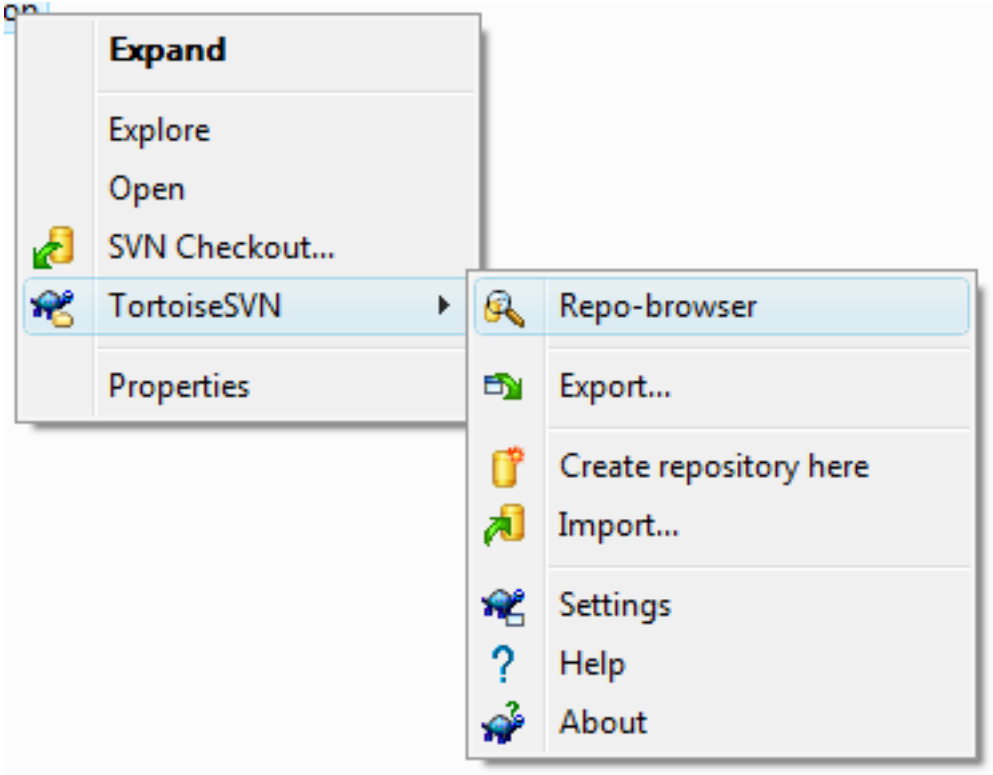
```
$ svn commit -m "write your commit log here"
$ svn ci -m "write your commit log here"
$ svn ci -m "write your commit log here" foo.cpp
```



查看版本库

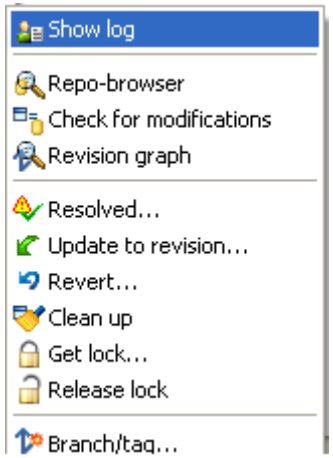
- 浏览版本库中的文件

```
$ svn list https://example.com/svn/project/trunk
```



查看提交记录

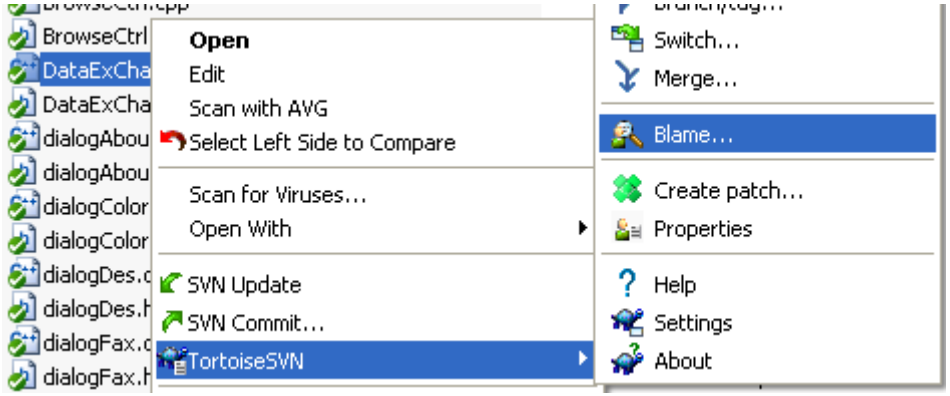
```
$ svn log
$ svn log -r 10   # Display logs for reversion 10 only
$ svn log -r 5:10 # Display logs for reversion 5 through 10
$ svn log -l 3    # Display maximum 3 entries of log
```



Blame

- 查看文件每一行的最后修改版本号和责任人

```
$ svn blame foo.cpp
$ svn annotate foo.cpp
$ svn ann foo.cpp
```



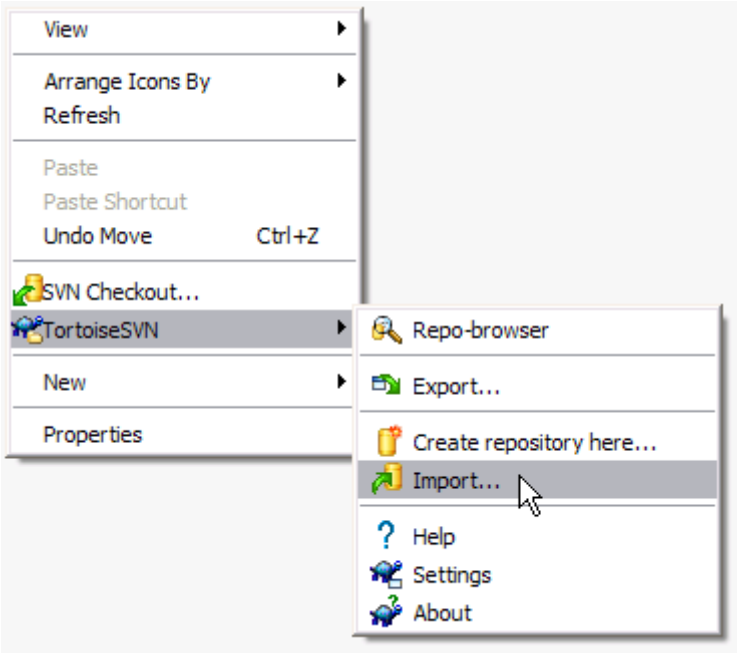
Import

- 将文件导入版本库，第一次初始化版本库

```
$ svn import /path/to/mytree https://example.com/svn/project/trunk \
-m "First import"
```

- 等效于

```
$ svn co https://example.com/svn/project/trunk
$ cd project
$ cp -r /path/to/mytree/* .
$ svn add .
$ svn ci -m "First import"
```



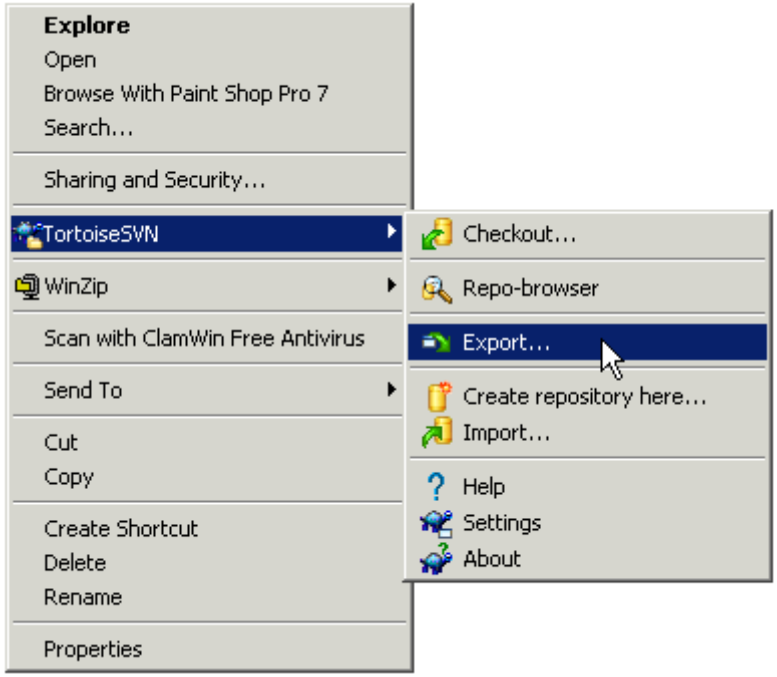
Export

- 从版本库中导出一份干净的文件包，里面不包含版本库的信息，通常用于发布

```
$ svn export https://example.com/svn/project/trunk project
```

- 导出指定版本

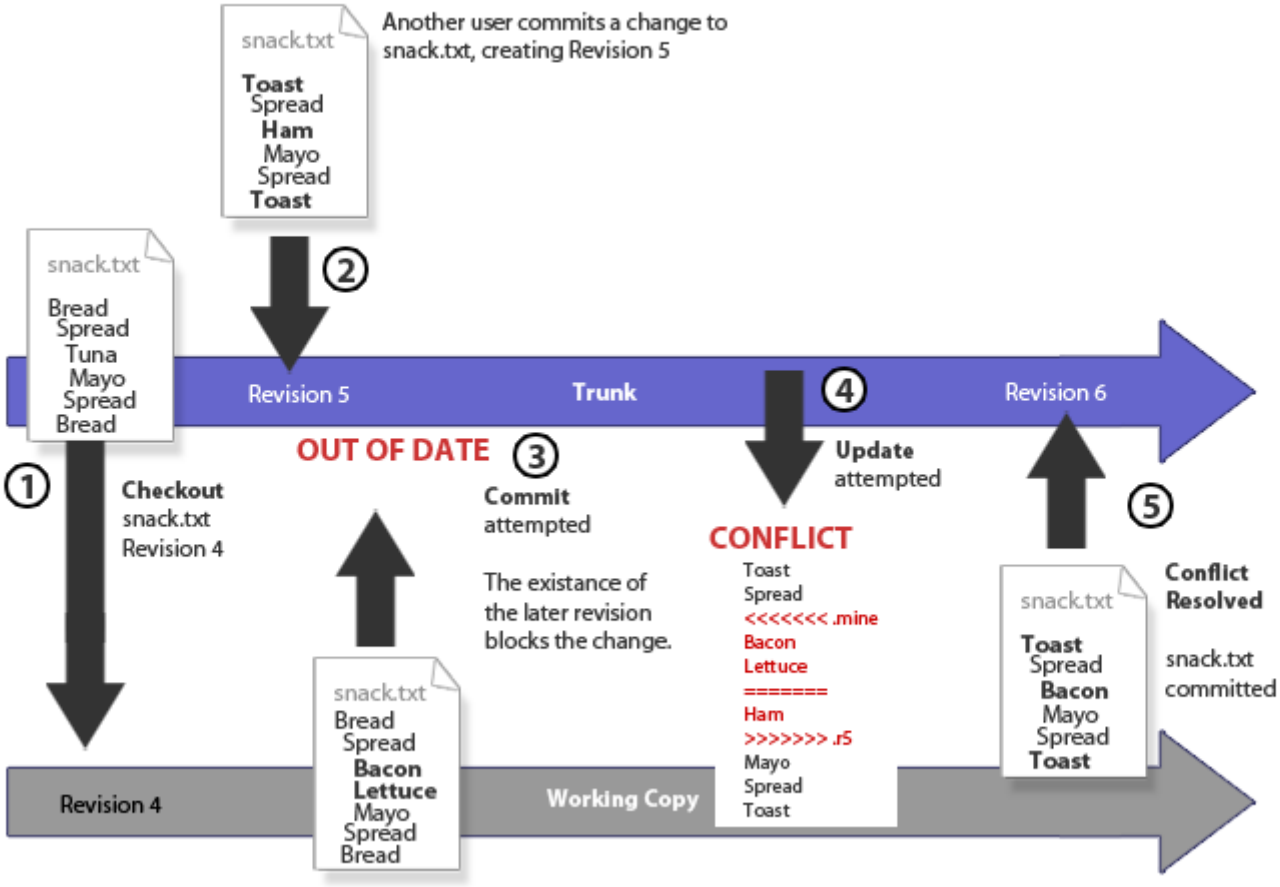
```
$ svn export https://example.com/svn/project/trunk -r 100 project-r100
```



Agenda

冲突产生的原因

- 不同的修改给机器造成了困惑，不知道怎么合并，将决定权留给人类
  - 两个人同时修改同一文件的同一行
  - 一个修改了某个文件，另一个人删除该文件



合并冲突

- 冲突文件内容类似下面这样，用文本编辑器直接打开，编辑成自己想要的结果即可

```
$ cat snack.txt
Toast
Spread
<<<<<<< .mine
Bacon
Lettuce
=======
Ham
>>>>>>> .r5
Mayo
Spread
Toast
```

- 使用 TortoiseMerge 进行合并

标记冲突解决

- 选择自己的修改（完全放弃或者覆盖别人的修改）

```
$ svn resolve --accept mine-full sandwich.txt
```

- 选择别人的修改（完全放弃或者覆盖自己的修改）

```
$ svn resolve --accept theirs-full sandwich.txt
```

- 手工编辑文件，合并冲突，然后应用合并后的修改

```
$ svn resolve --accept working sandwich.txt
```

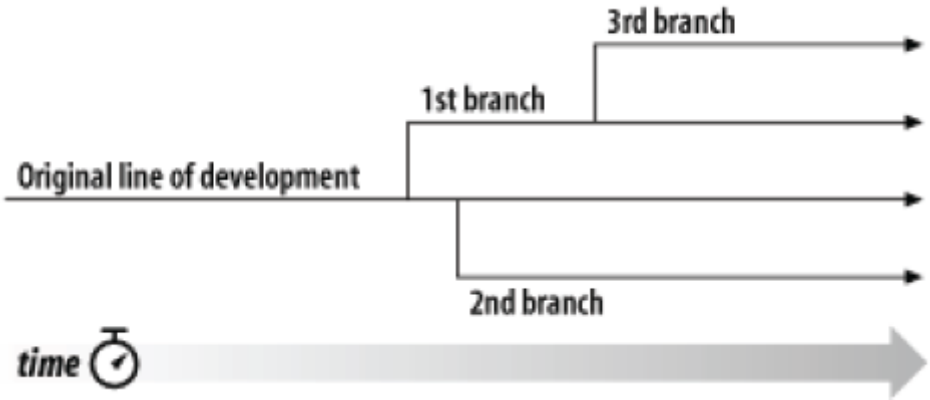
如何避免冲突

- 尽量保持工作副本与版本库同步（经常 `update`），每天开始工作前先执行 `update`
- 完成一项工作后及时提交修改，每天下班前清理手上的工作，尽量提交入库
- 二进制文件（图片、Office 文档等）无法进行合并，尽量避免多人同时修改同一个文件，可以采用 `lock-modify-unlock` 方式进行修改提交

Agenda

什么是分支

- 从一条开发线上分叉出来
- 可以维护自己独立的修改历史
- 各个分支上的修改不会相互干扰



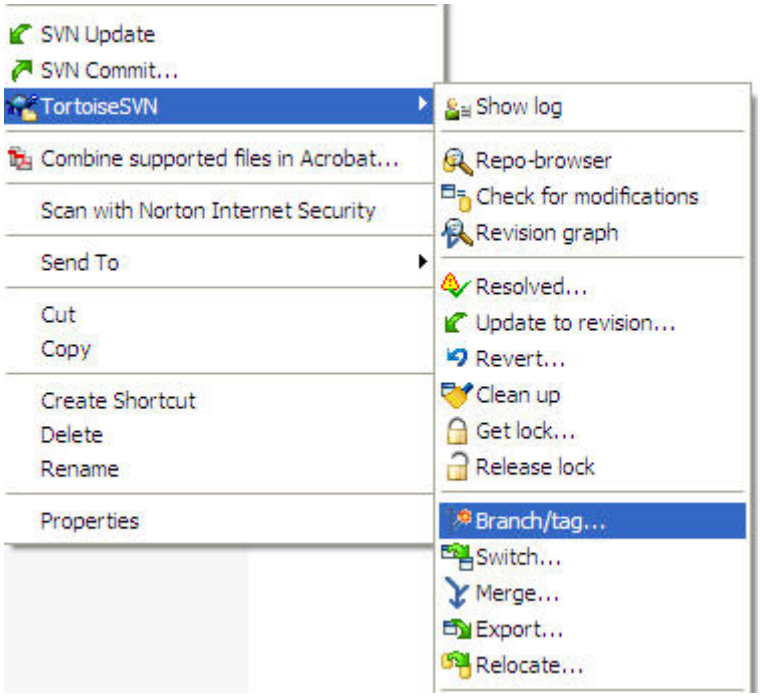
什么时候需要分支

- 开发一个相对比较大的功能，开发时间可能比较长（例如 3 天以上），开发期间不能影响其他人的工作
- 修改一个 Bug，改动比较大，或者修改时间比较长，改动需要经过严谨的测试才能应用到项目中
- 多人合作完成一项工作，需要同步个人的修改，但是工作完成之前又不能影响到主线上的开发
- 基于原有的项目启动一个新的项目

创建分支

- 基于 trunk 的最新版本创建一个功能分支 feature-xxx

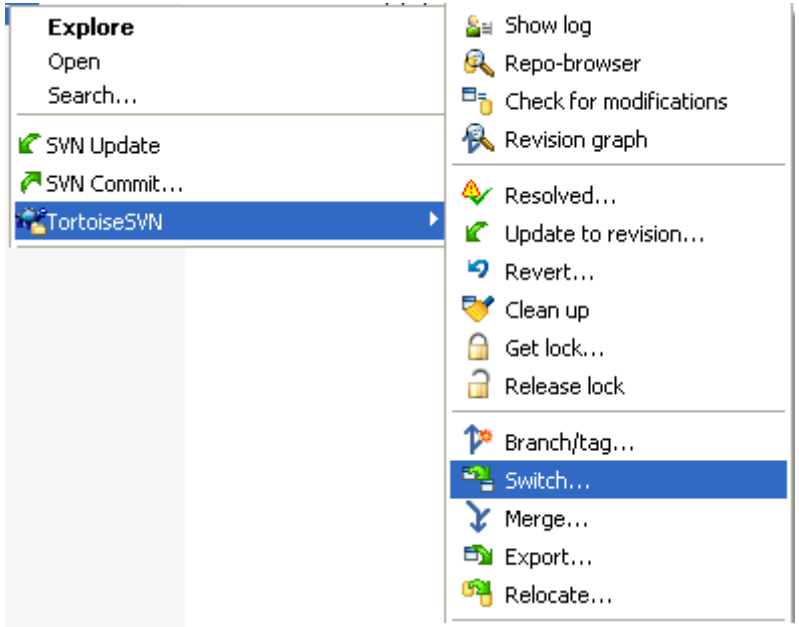
```
$ svn copy https://example.com/svn/project/trunk \
https://example.com/svn/project/branches/feature-xxx \
-m "Create feature branch for developing ..."
```



切换分支

- 将工作副本关联到 feature-xxx 分支

```
$ svn switch ^/project/branches/feature-xxx
```



合并分支

1. 保持分支与 trunk 同步，在分支的工作副本中执行

```
$ svn merge ^/project/trunk
```

2. Build and test
3. 切换到 trunk

```
$ svn switch ^/project/trunk
```

4. 将 feature 分支合并回 trunk

```
$ svn merge --reintegrate ^/project/branches/feature-xxx
```

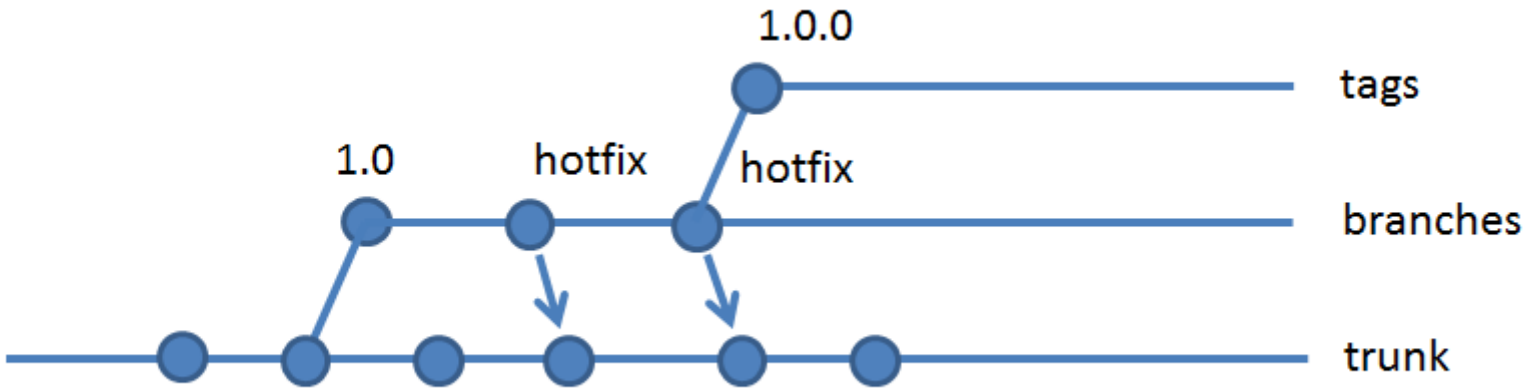
Tag

- 创建标签

```
$ svn copy https://example.com/svn/project/trunk \
https://example.com/svn/project/tags/release-1.0 \
-m "Tagging the 1.0 release"
```

推荐的 Release Branche

1. trunk 作为开发主线，每天的修改提交到/trunk
2. 版本发布前，从/trunk 复制出一个分支到/branches 下，例如：/branches/1.0
3. 测试团队的工作在/branches/1.0 上进行，开发团队的工作依旧可以在/trunk 上进行。
4. 如果测试发现 Bug，必须要在这一次解决的话，开发团队在/trunk 上修改、提交，然后合并到/branches/1.0
5. 测试全部通过后，将/branches/1.0 复制到/tags/1.0.0 作为这一次的发布标签，/tags/1.0.0 就可以打包发布出去了



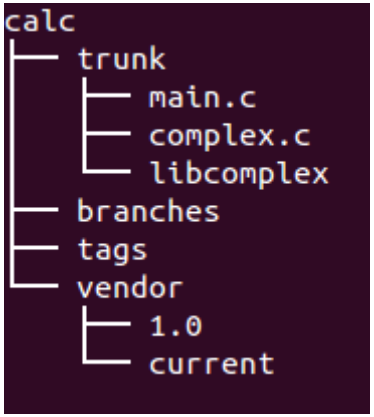
Vendor Branch

1. 第一次拿到厂商提供的代码，导入版本库并打上 tag

```
$ svn import /path/to/libcomplex-1.0 \
http://example.com/svn/calc/vendor/libcomplex/current \
-m "importing initial 1.0 vendor drop"
$ svn copy http://example.com/svn/calc/vendor/libcomplex/current \
http://example.com/svn/calc/vendor/libcomplex/1.0 \
-m "tagging libcomplex-1.0"
```

2. 从厂商代码开出一个分支形成 trunk，作为后续开发的基础

```
$ svn copy http://example.com/svn/calc/vendor/libcomplex/1.0 \
http://example.com/svn/calc/trunk/libcomplex \
-m "bringing libcomplex-1.0 into the main branch"
```



Vendor Branch

3. 厂商代码有新的版本发布，导入版本库

```
$ svn_load_dirs.pl http://example.com/svn/calc/vendor/libcomplex \
current \
/path/to/libcomplex-2.0
$ svn commit -m "import 2.0 vendor drop"
$ svn copy http://example.com/svn/calc/vendor/libcomplex/current \
http://example.com/svn/calc/vendor/libcomplex/2.0 \
-m "tagging libcomplex-2.0"
```

4. 将厂商的修改应用到开发分支

```
$ svn merge ^/vendor/libcomplex/2.0 \
^/vendor/libcomplex/1.0 \
libcomplex
… # resolve all the conflicts between their changes and our changes
$ svn commit -m "merging libcomplex-2.0 into the main branch"
```

Agenda

提交日志的填写

好的提交日志

- 描述清楚这次提交解决了什么问题，或者实现了什么功能
- 日志可以分成多行，第一行是概况行的描述，然后空一行，再写上详细的内容
- 如果修改有对应的 Redmine Issue，应该附上 Issue 的链接，把 Issue 的 ID 和标题也写上

不好的提交日志

- update
- fix
- 更新代码
- 修改问题

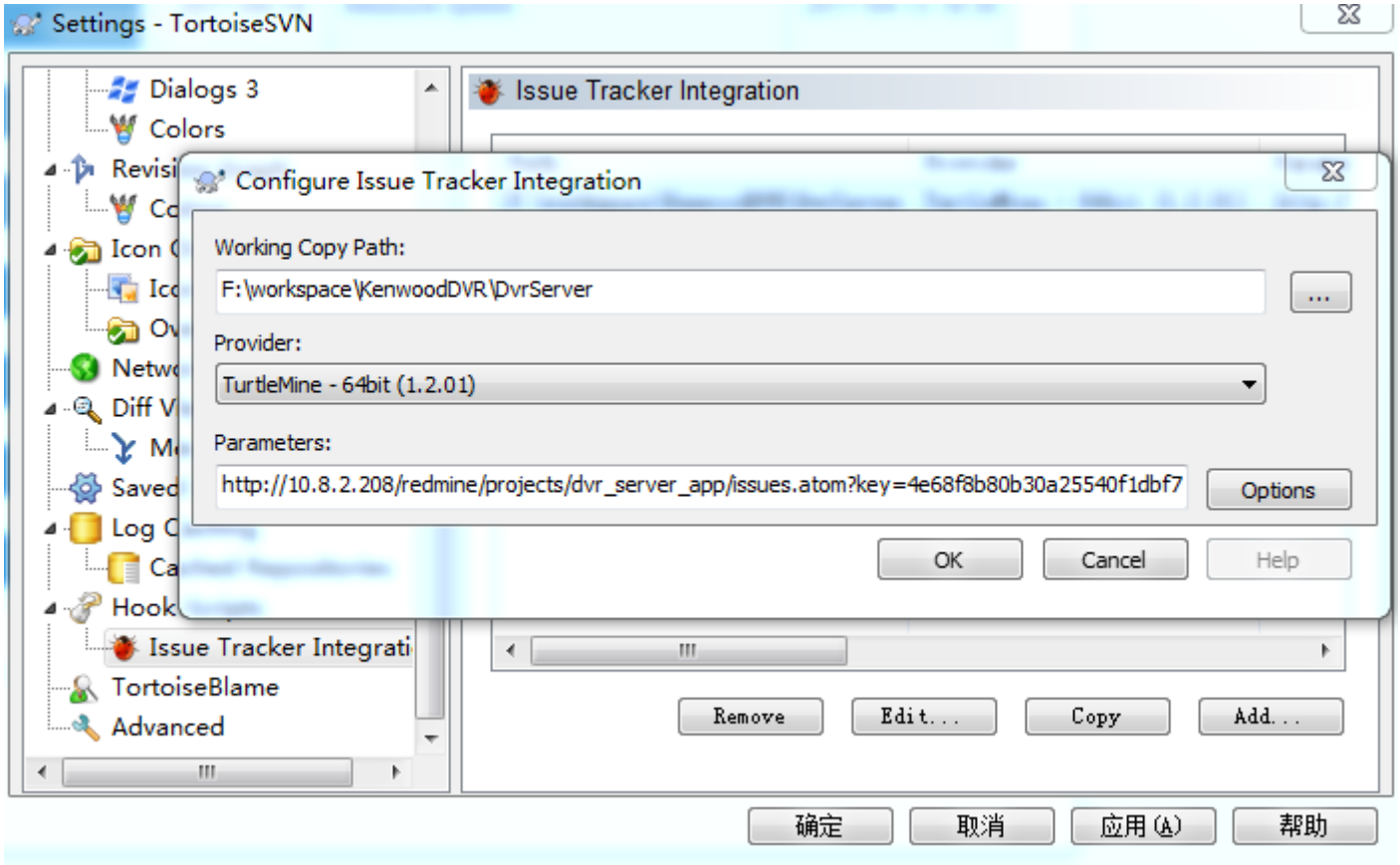
提交规范

- 频繁提交、原子提交，一次提交只完成一项工作（问题解耦），例如：实现一个功能、解决一个 Bug
- 提交前必须仔细检查修改，确认所有修改都是需要提交的，确认没有遗漏的文件，特别是新增的文件容易遗漏
- 以下内容不应该提交：
  - 开发中的调试代码，比如临时的、用于调试的 Log 输出语句
  - 注释掉的代码，不用的代码应该直接删除
  - 不必要的空行
  - 临时文件、编译中间文件、能编译生成的库文件、个人配置文件等
- 让 SVN 知道你的意图，例如：复制文件、移动文件有没有使用正确的操作

Agenda

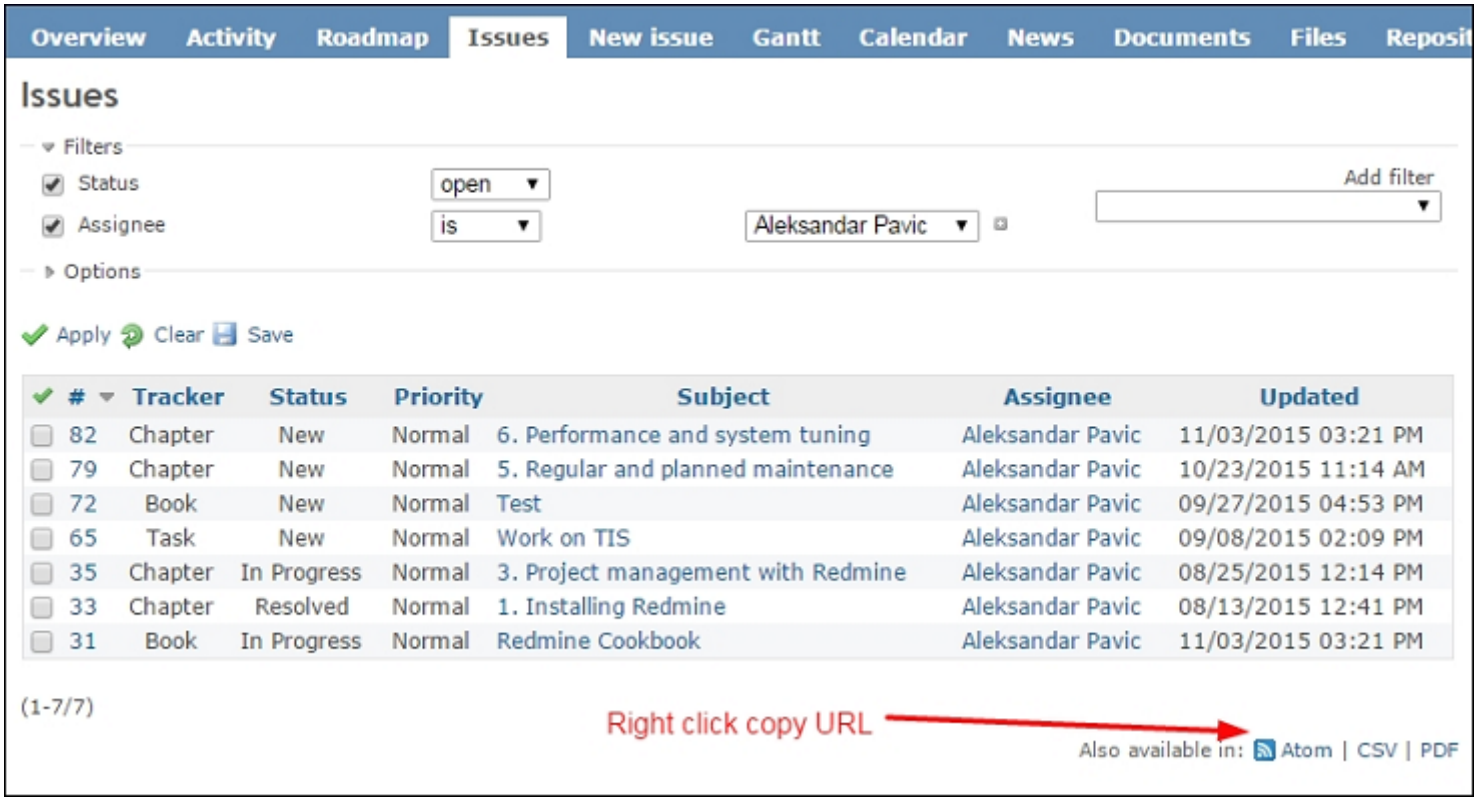
TortoiseSVN 与 Redmine 集成

1. 安装 TurtleMine plugin
2. Tortoise SVN | Settings -> Hook Scripts -> Issue Tracker Integration
3. 设置 Working Copy Path 和 Parameters



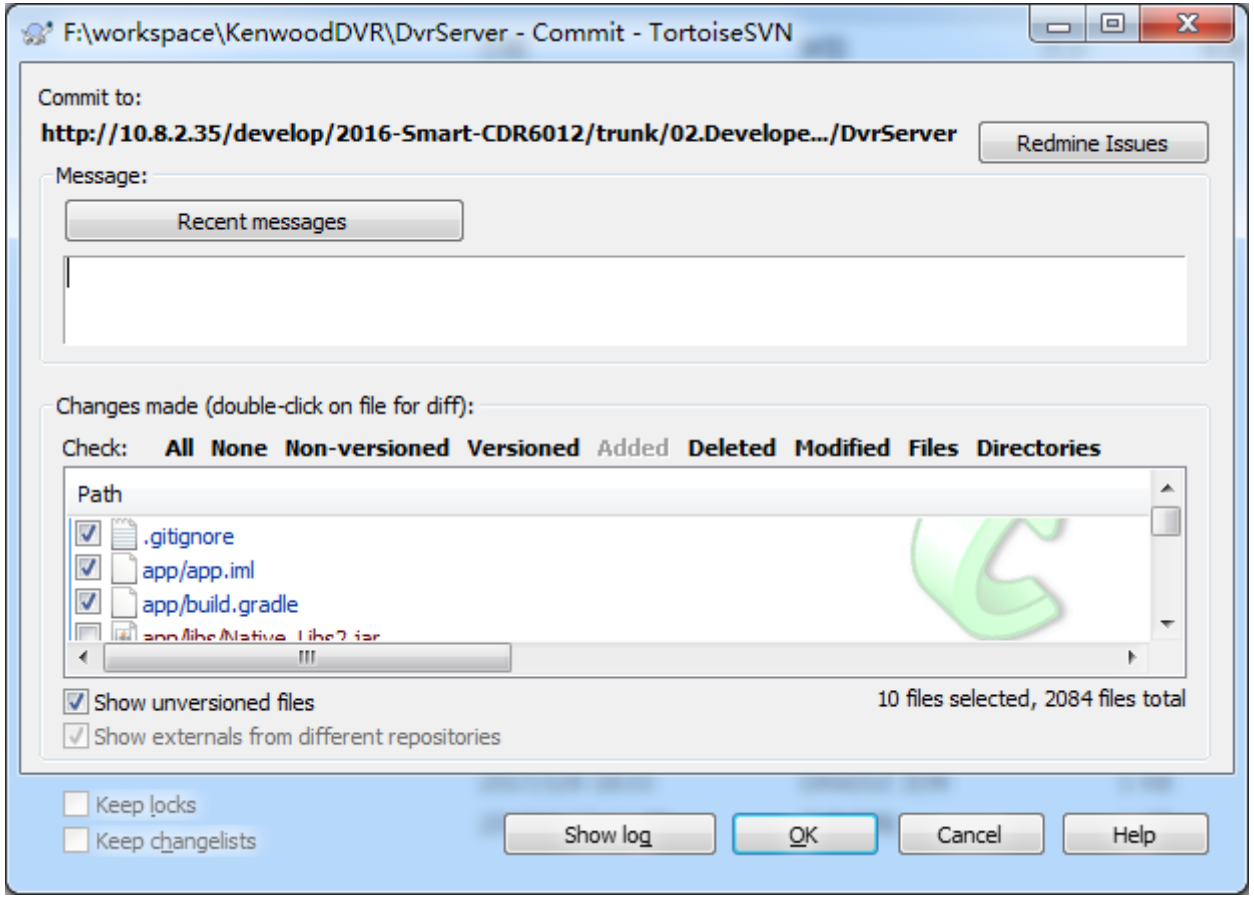
Parameters 的获取

- 登陆 Redmine
- Project Issue 页面右下角 `Atom` 点右键复制 URL



提交

1. 通过 TortoiseSVN 提交的时候，点击对话框的右上角的 `Redmine Issues` 按钮
2. 在出现一个 Issues 列表中选择关联的 Issue，可以多选



The End