

SVN 及 Git、SVN 协作学习笔记

Author: Marlous

E-mail: goonecat@foxmail.com

Date: 2018/10/25

一 SVN 与 Git 区别联系

[参考博文](#)

SVN 比 Git 概念更简单。

SVN 为集中式版本控制系统，从中央服务器取得最新的版本后开始本地工作，然后推到中央服务器，必须联网才能工作。适合小团队协作，其他部门协作使用。

Git 为分布式，不需要中央服务器，每个人本地都有一份完整的版本库，无需联网（实际会有一台“中央服务器”来让交换代码更方便）。适合大团队协作。

二 SVN

1 SVN 概念

SVN 服务端，客户端软件教程：[参考](#)

1. SVN 即 Subversion（分为服务端、客户端）。

Windows 平台：服务端软件有 VisualSVN server，客户端软件有 TortoiseSVN。

Linux 平台：服务端软件有 CollabNet Subversion Edge（GUI），客户端软件有 RapidSVN。

2. 服务端与客户端使用流程：

- 服务端使用流程：建立仓库；添加用户；建立组并添加用户（可同时设置权限）。
- 客户端使用流程：checkout（拷贝到本地，链接在服务器端获得）；update（更新本地仓库）；commit（提交）。

2 SVN 使用（命令行与图形界面）

服务端：

- 初次配置，创建版本库根目录：`mkdir /opt/svn`
- 创建一个仓库：`svnadmin create /opt/svn/demo`，具体文件 SVN 会以自己的格式存储。
- 此创建的仓库目录中，修改默认配置文件配置，包括 `svnserve.conf`、`passwd`、`authz` 配置相关用户和权限。[配置教程](#)

svnserve.conf 配置：

```
[general]
anon-access = none
auth-access = write
password-db = /opt/svn/demo/conf/passwd
authz-db = /opt/svn/demo/conf/authz
realm = Ubuntu_SVN
```

passwd 配置:

```
[users]
admin = admin
mo = 12345
```

authz 配置 1: 一个 svnserve 可以为一个仓库工作 `svnserve -d -r /opt/svn/demo`

```
[groups]
admin = mo
dev = user1

[/]
@admin = rw
user1 = r
```

authz 配置 2: 一个 svnserve 可以为多个仓库工作 `svnserve -d -r /opt/svn/`

```
[groups]
admin = mo
dev = user1

# [库名:路径] , 服务器运行多个仓库, 需写出仓库名。
# 此时还用 [/], 则表示所有库的根目录, 同理, [/src] 表示所有库的根目录下的 src 目录。
[demo:/]
@admin = rw
user1 = r

[demo2:/tags]
mo = rw
user1 = r
```

客户端通过前缀为 `svn://` 或 `svn+ssh://` 的 URL 访问版本库有效, 而对通过前缀 `http://`、`https://` 或 `file:///` 的 URL 无效。

- 启动服务: `svnserve -d -r 目录 --listen-port 端口号`, 如 `svnserve -d -r /opt/svn`, (此为启动多个仓库; 启动某一个仓库, 目录为下一级 `/opt/svn/demo`, 不加端口号默认 3690)。
- 重启服务: 查找后 `kill $(ps -ef|grep svn)`, 然后 `svnserve -d -r 目录`。

客户端:

[SVN 常用命令参考](#)

- 补充常用目录结构：可以在 demo 仓库文件夹中建立多个文件夹区分功能，如 scr 文件夹等。
标准目录为 /trunk 主分支，是日常开发进行的地方； /branches 分支，一些阶段性的 release 版本； /tags 一般是只读的，这里存储阶段性的发布版本，作为一个里程碑的版本进行存档。
- 检出操作，初次从服务器拉代码：
在文件夹（SVN 根目录）上面右键 checkout（使用 TortoiseSVN 客户端）。服务端运行多个仓库，填入 svn 服务器及目标仓库 `svn://192.168.50.214/demo`，服务器运行一个仓库填 `svn://192.168.50.214/`；。
命令行为 `svn checkout svn://192.168.50.214/demo --username=mo`。
- 补充检入操作，初次提交服务器：文件夹右键 Import。
命令行为 `svn import -m "init import" http://192.168.50.214/svn/test/`
- 更新：`svn update`。
- 解决冲突。
- 做成版本：`svn add`，`svn status`，`svn diff -r 旧修订版序号:新修订版序号`，`svn commit -m "xxx"`。
- 对提交但未推送的回滚：`svn revert`。对已经推送的回滚：`svn merge -r 22:21 readme`（22 为当前版本，21 为么恢复的，后面为文件名）。
- 退回与更新：`svn log`，`svn merge -r 版本号`。更新到某处 `svn update -r 版本号`。
- 推送：`svn update`。
- 删除文件：`svn delete [path] -m "注释"`。
- 创建分支：`svn copy trunk/ branches/my_branch`，然后 commit，update。
- 合并分支：先进入要合并的分支 trunk 目录，`svn merge ../branches/my_branch/`。
- 标签：`svn copy trunk/ tags/v1.0`，然后 commit。

三 Git 与 SVN 协作

[官方参考文档](#) [参考博文](#)

1 概念

git-svn 工具把 Git 变成了 Subversion 服务的客户端，在本地享受到 Git 所有的功能，而后直接向 Subversion 服务器推送内容。Git 中所有 Subversion 桥接命令的基础是 `git svn`。所有的命令都从它开始，它实际是在与 Subversion 交互。

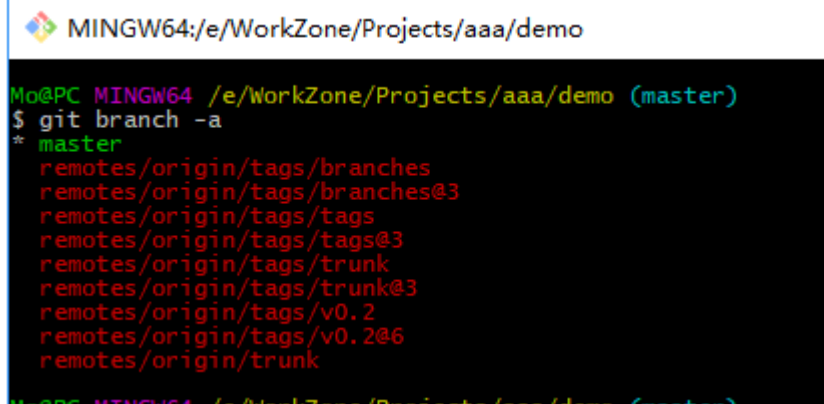
2 流程

软件工程师本地库使用 Git，通过 `git-svn ->` 公司 SVN 库（软件工程师直接与公司的 SVN 库打交道）。

3 git-svn 工作流（Windows 使用方法）

- 克隆远程 SVN 的仓库变为 git-svn 协作仓库。Windows 下使创建要使用的文件夹，Git Bash Here，`git svn clone svn://192.168.50.214/demo -T trunk -b branches -t tags`，或用 -s 参数。注：SVN 服务器该仓库

authz 配置目录权限下一定要加上 `* = r`，否则无法被克隆会报错！下图表明成功克隆。



```
MINGW64:/e/WorkZone/Projects/aaa/demo
Mo@PC MINGW64 /e/WorkZone/Projects/aaa/demo (master)
$ git branch -a
* master
remotes/origin/tags/branches
remotes/origin/tags/branches@3
remotes/origin/tags/tags
remotes/origin/tags/tags@3
remotes/origin/tags/trunk
remotes/origin/tags/trunk@3
remotes/origin/tags/v0.2
remotes/origin/tags/v0.2@6
remotes/origin/trunk
```

项目太大也可以同 `init`、`fetch` 代替 `clone`

初始化仓库

```
git svn init <svn-url>
```

拉取指定版本之后的记录

```
git svn fetch -r <svn-version>
```

拉取最新版本

```
git svn fetch -r HEAD
```

- 创建本地工作分支并在其上工作：`git checkout -b work`。
- 正常使用 Git 做开发，`add`、`commit`。完成阶段性工作。
- 更新 master 分支：

```
git checkout master
```

```
git svn rebase # 同步远程代码
```

- 更新 work 分支：

```
git checkout work
```

```
git rebase master # 手工解决可能的冲突
```

- 将 master、work 分支合并：`git checkout master`，`git merge work`。
- 向 Subversion 服务器推送：`git svn dcommit`。

注：

- 将在主线分支外进行的开发通通行合回主线，避免直接合并。离线状态下进行多次提交然后一次性的推送到 Subversion 的服务器上。
- 不要单独建立和使用一个 Git 服务器来搞合作。可以为了加速新开发者的克隆进程建立一个，但是不要向它提供任何不包含 `git-svn-id` 条目的内容。甚至可以添加一个 `pre-receive` 挂钩来在每一个提交信息中查找 `git-svn-id` 并拒绝提交那些不包含它的 `commit`。

补充：需要克隆多个远程 SVN 分支情况，参考博文配置 [博文](#)。修改 `.git/config` 文件，添加新的远程分支；

```
[svn-remote "svn-trunk"]
  url = <trunk-svn-url>
  fetch = :refs/remotes/git-svn-trunk
```

然后使用 `git svn fetch svn-trunk -r HEAD` 来获取。切换到本地创建的分支工作：

```
# 切换到远程分支git-svn-trunk
git checkout git-svn-trunk
# 创建本地分支
git checkout -b master-trunk
```