

對比 Git 與 SVN，這篇講的很易懂

2018-11-04 由 Java 的小本家 發表於[程式開發](#)

一、Git vs SVN

Git 和 SVN 孰優孰好，每個人有不同的體驗。

Git 是分佈式的，SVN 是集中式的

這是 Git 和 SVN 最大的區別。若能掌握這個概念，兩者區別基本搞懂大半。因為 Git 是分佈式的，所以 Git 支援離線工作，在本地可以進行很多操作，包括接下來將要重磅推出的分支功能。而 SVN 必須聯網才能正常工作。

Git 複雜概念多，SVN 簡單易上手

所有同時掌握 Git 和 SVN 的開發者都必須承認，Git 的命令實在太多了，日常工作需要掌握 add, commit, status, fetch, push, rebase 等，若要熟練掌握，還必須掌握 rebase 和 merge 的區別，fetch 和 pull 的區別等，除此之外，還有 cherry-pick, submodule, stash 等功能，僅是這些名詞聽著都很繞。

在易用性這方面，SVN 會好得多，簡單易上手，對新手很友好。但是從另外一方面看，Git 命令多意味著功能多，若我們能掌握大部分 Git 的功能，體會到其中的奧妙，會發現再也回不去 SVN 的時代了。

Git 分支廉價，SVN 分支昂貴

在版本管理裡，分支是很常使用的功能。在發布版本前，需要發布分支，進行大需求開發，需要 **feature** 分支，大團隊還會有開發分支，穩定分支等。在大團隊開發過程中，常常存在創建分支，切換分支的需求。

Git 分支是指針指向某次提交，而 **SVN** 分支是拷貝的目錄。這個特性使 **Git** 的分支切換非常迅速，且創建成本非常低。

而且 **Git** 有本地分支，**SVN** 無本地分支。在實際開發過程中，經常會遇到有些代碼沒寫完，但是需緊急處理其他問題，若我們使用 **Git**，便可以創建本地分支存儲沒寫完的代碼，待問題處理完後，再回到本地分支繼續完成代碼。

二、Git 核心概念

Git 最核心的一個概念就是工作流。

- 工作區(Workspace)是電腦中實際的目錄。
- 暫存區(Index)類似於緩存區域，臨時保存你的改動。
- 倉庫區(Repository)，分為本地倉庫和遠程倉庫。

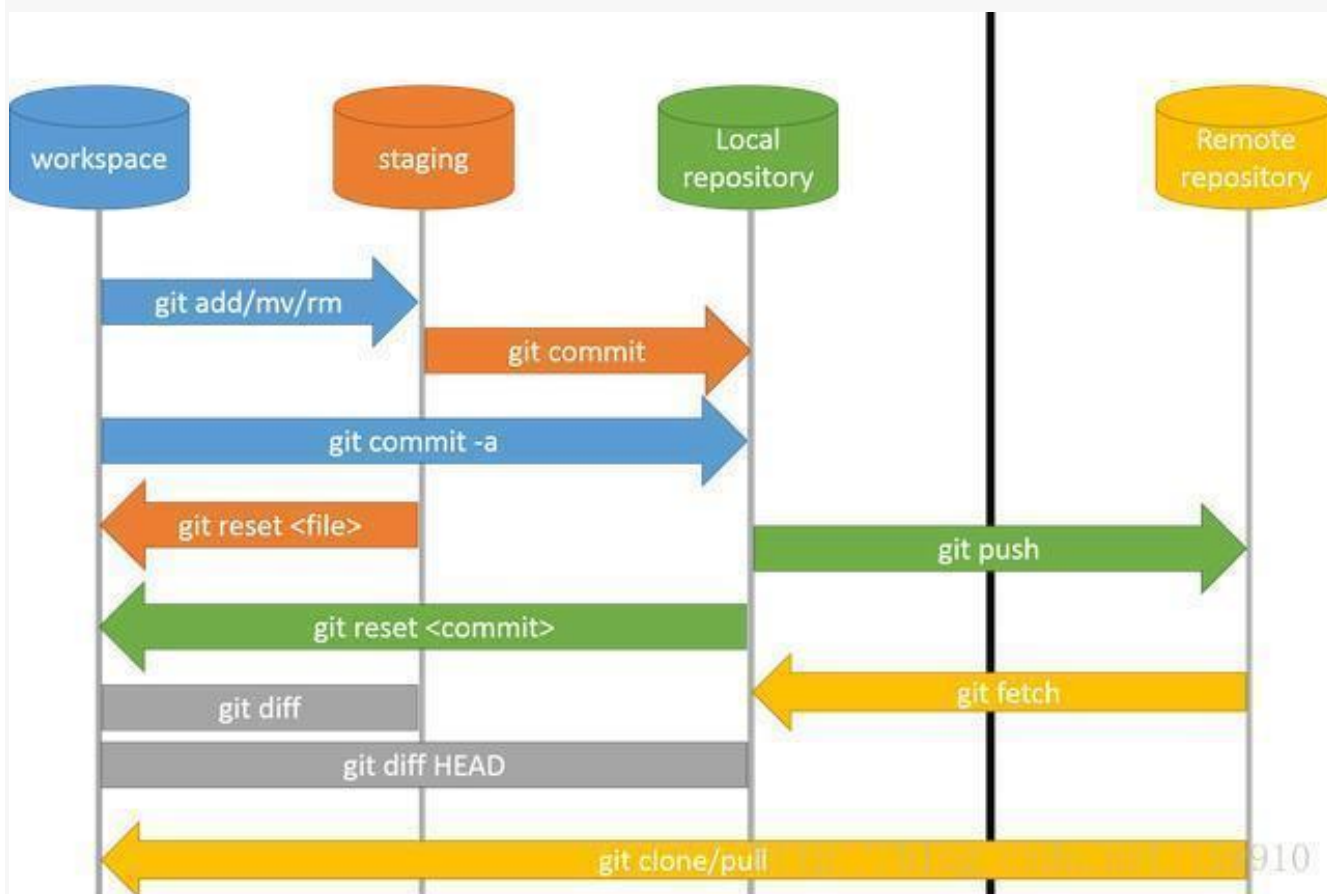
從 **SVN** 切換到 **Git**，最難理解並且最不能理解的是暫存區和本地倉庫。熟練使用

Git 後，會發現這簡直是神設計，由於這兩者的存在，使許多工作變得易管理。

通常提交代碼分為幾步：

1. `git add` 從工作區提交到暫存區
2. `git commit` 從暫存區提交到本地倉庫
3. `git push` 或 `git svn dcommit` 從本地倉庫提交到遠程倉庫

一般來說，記住以下命令，便可進行日常工作了（圖片來源於網絡）：



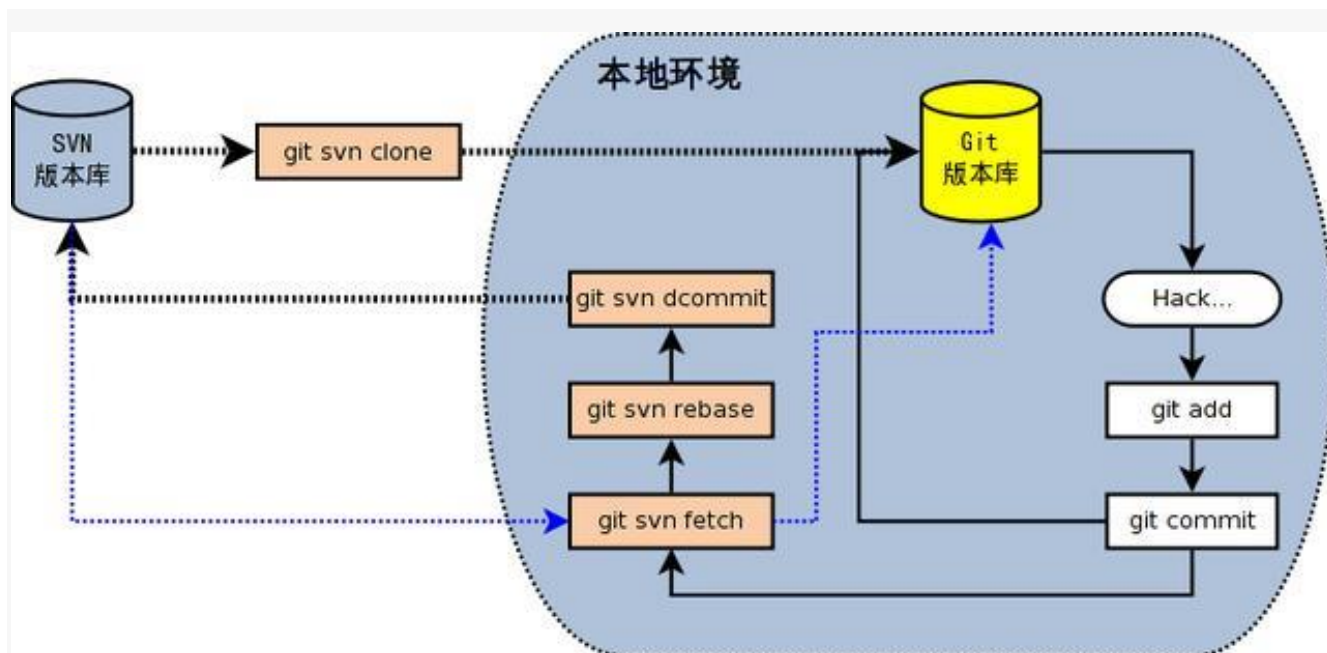
[Git 命令]

三、Git-SVN 常用命令

本節命令針對使用 Git-SVN 的開發者，請務必掌握。

若伺服器使用的 SVN，但是本地想要體驗 Git 的本地分支，離線操作等功能，可以使用 Git-SVN 功能。

常用操作如下（圖片來源於網絡）：



[Git-SVN]

下載一個 SVN 項目和它的整個代碼歷史，並初始化為 Git 代碼庫

```
$ git svn clone -s [repository]
```

查看當前版本庫情況

```
$ git svn info
```

取回遠程倉庫所有分支的變化

```
$ git svn fetch
```

取回遠程倉庫當前分支的變化，並與本地分支變基合併

```
$ git svn rebase
```

上傳當前分支的本地倉庫到遠程倉庫

```
$ git svn dcommit
```

拉取新分支，並提交到遠程倉庫

```
$ svn copy [remote_branch] [new_remote_branch] -m [message]
```

創建遠程分支對應的本地分支

```
$ git checkout -b [local_branch] [remote_branch]
```

四、初始化

從本節開始，除特殊說明，以下命令均適用於 Git 與 Git-SVN。

```
# 在當前目錄新建一個 Git 代碼庫

$ git init

# 下載一個項目和它的整個代碼歷史 [Git only]

$ git clone [url]
```

五、組態

```
# 列舉所有組態

$ git config -l

# 為命令組態別名

$ git config --global alias.co checkout
$ git config --global alias.ci commit
$ git config --global alias.st status
$ git config --global alias.br branch

# 設置提交代碼時的用戶資訊

$ git config [--global] user.name "[name]"
$ git config [--global] user.email "[email address]"
```

Git 用戶的組態檔案位於 ~/.gitconfig

Git 單個倉庫的組態檔案位於 ~/\$PROJECT_PATH/.git/config

六、增刪檔案

```
# 新增當前目錄的所有檔案到暫存區

$ git add .

# 新增指定檔案到暫存區

$ git add <file1> <file2> ...

# 新增指定目錄到暫存區，包括其子目錄

$ git add <dir>
```

刪除工作區檔案，並且將這次刪除放入暫存區

```
$ git rm [file1] [file2] ...
```

停止追蹤指定檔案，但該檔案會保留在工作區

```
$ git rm --cached [file]
```

改名檔案，並且將這個改名放入暫存區

```
$ git mv [file-original] [file-renamed]
```

把檔案名稱 `file1` 新增到 `.gitignore` 檔案裡，Git 會停止追蹤 `file1` 的狀態。

七、分支

列出所有本地分支

```
$ git branch
```

列出所有本地分支和遠程分支

```
$ git branch -a
```

新建一個分支，但依然停留在當前分支

```
$ git branch [branch-name]
```

新建一個分支，並切換到該分支

```
$ git checkout -b [new_branch] [remote-branch]
```

切換到指定分支，並更新工作區

```
$ git checkout [branch-name]
```

合併指定分支到當前分支

```
$ git merge [branch]
```

選擇一個 `commit`，合併進當前分支

```
$ git cherry-pick [commit]
```

刪除本地分支，`-D` 參數強制刪除分支

```
$ git branch -d [branch-name]

# 刪除遠程分支

$ git push [remote] :[remote-branch]
```

八、提交

```
# 提交暫存區到倉庫區

$ git commit -m [message]

# 提交工作區與暫存區的變化直接到倉庫區

$ git commit -a

# 提交時顯示所有 diff 資訊

$ git commit -v

# 提交暫存區修改到倉庫區，合併到上次修改，並修改上次的提交資訊

$ git commit --amend -m [message]

# 上傳本地指定分支到遠程倉庫

$ git push [remote] [remote-branch]
```

九、拉取

```
# 下載遠程倉庫的所有變動 (Git only)

$ git fetch [remote]

# 顯示所有遠程倉庫 (Git only)

$ git remote -v

# 顯示某個遠程倉庫的資訊 (Git only)

$ git remote show [remote]

# 增加一個新的遠程倉庫，並命名 (Git only)

$ git remote add [remote-name] [url]

# 取回遠程倉庫的變化，並與本地分支合併，(Git only)，若使用 Git-SVN，請查看第三節

$ git pull [remote] [branch]
```

取回遠程倉庫的變化，並與本地分支變基合併，(Git only)，若使用 Git-SVN，請查看第三節

```
$ git pull --rebase [remote] [branch]
```

十、撤銷

恢復暫存區的指定檔案到工作區

```
$ git checkout [file]
```

恢復暫存區當前目錄的所有檔案到工作區

```
$ git checkout .
```

恢復工作區到指定 commit

```
$ git checkout [commit]
```

重設暫存區的指定檔案，與上一次 commit 保持一致，但工作區不變

```
$ git reset [file]
```

重設暫存區與工作區，與上一次 commit 保持一致

```
$ git reset --hard
```

重設當前分支的指針為指定 commit，同時重設暫存區，但工作區不變

```
$ git reset [commit]
```

重設當前分支的 HEAD 為指定 commit，同時重設暫存區和工作區，與指定 commit 一致

```
$ git reset --hard [commit]
```

新建一個 commit，用於撤銷指定 commit

```
$ git revert [commit]
```

將未提交的變化放在儲藏區

```
$ git stash
```

將儲藏區的內容恢復到當前工作區

```
$ git stash pop
```


十一、查詢

查看工作區檔案修改狀態

```
$ git status
```

查看工作區檔案修改具體內容

```
$ git diff [file]
```

查看暫存區檔案修改內容

```
$ git diff --cached [file]
```

查看版本庫修改記錄

```
$ git log
```

查看某人提交記錄

```
$ git log --author=someone
```

查看某個檔案的歷史具體修改內容

```
$ git log -p [file]
```

查看某次提交具體修改內容

```
$ git show [commit]
```

十二、其他

寫在後面

從 SVN 到 Git，除本文列舉的基礎概念和常用命令，包括但不限於如何從 SVN 伺服器切換到 Git 伺服器，分支模型管理等也非常重要。