為什麼Git把SVN拍在了沙灘上

程序員自修室 今天

Git vs SVN

> Git 和SVN 孰優孰好，每個人有不同的體驗。

一、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 支持離線工作，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。

二、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。

三、Git分支廉價，SVN分支昂貴

在版本管理里，分支是很常使用的功能。在发布版本前，需要发布分支，进行大需求开发，需要 feature 分支，大团队还会有开发分支，稳定分支等。在大团队开发过程中，常常存在创建分支，切换分支的需求。

Git 分支是指针指向某次提交，而 SVN 分支是拷贝的目录。这个特性使 Git 的分支切换非常迅速，且创建成本非常低。

而且 Git 有本地分支，SVN 无本地分支。在实际开发过程中，经常会遇到有些代码没写完，但是需紧急处理其他问题，若我们使用 Git，便可以创建本地分支存储没写完的代码，待问题处理完后，再回到本地分支继续完成代码。

Git核心思想

Git 最核心的一个概念就是**工作流**。

- **工作区(Workspace)** 是电脑中实际的目录。
- **暂存区(Index)** 类似于缓存区域，临时保存你的改动。

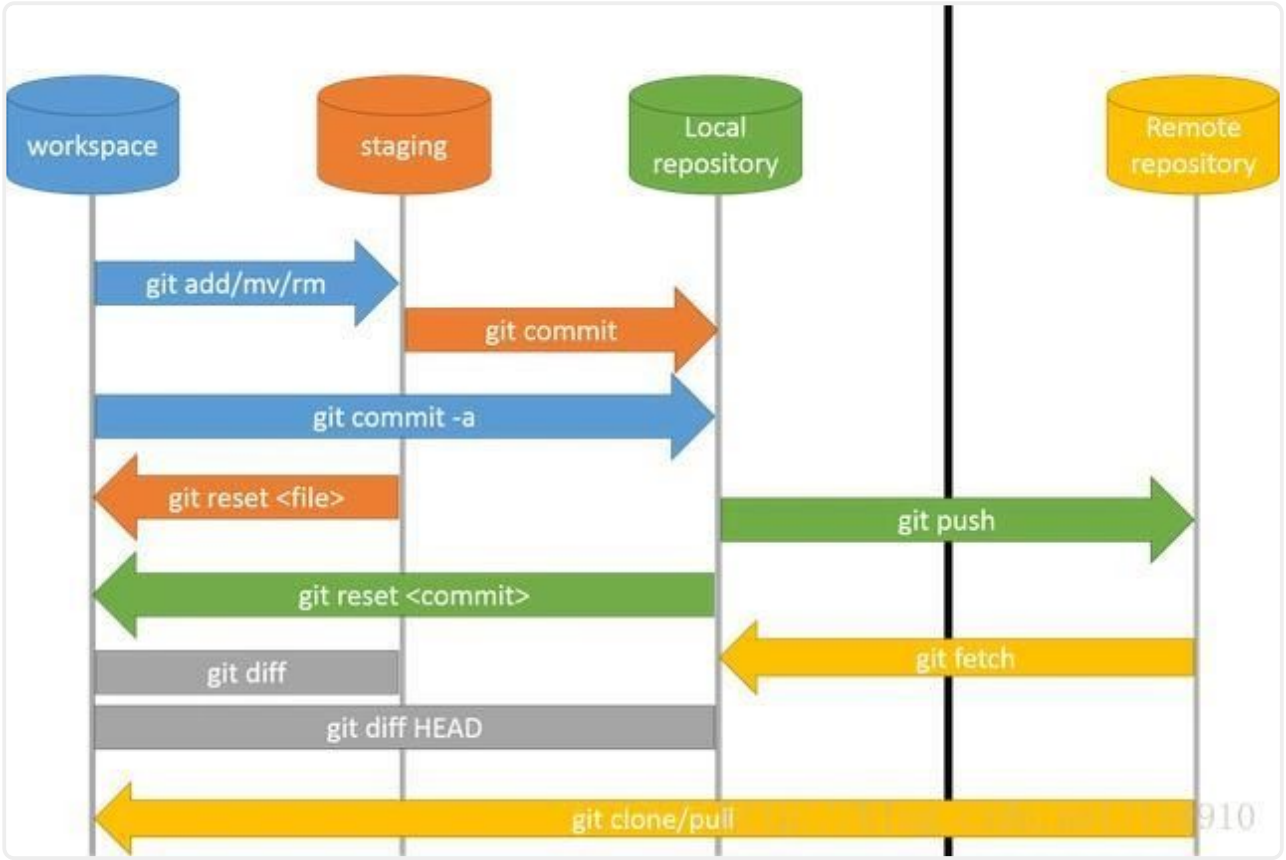
- **仓库区(Repository)**，分为本地仓库和远程仓库。

从 SVN 切换到 Git，最难理解并且最不能理解的是暂存区和本地仓库。熟练使用 Git 后，会发现这简直是神设计，由于这两者的存在，使许多工作变得易管理。

通常提交代码分为几步：

- `git add` 从工作区提交到暂存区
- `git commit` 从暂存区提交到本地仓库
- `git push`或 `git svn dcommit` 从本地仓库提交到远程仓库

一般来说，记住以下命令，便可进行日常工作了（图片来源于网络）：

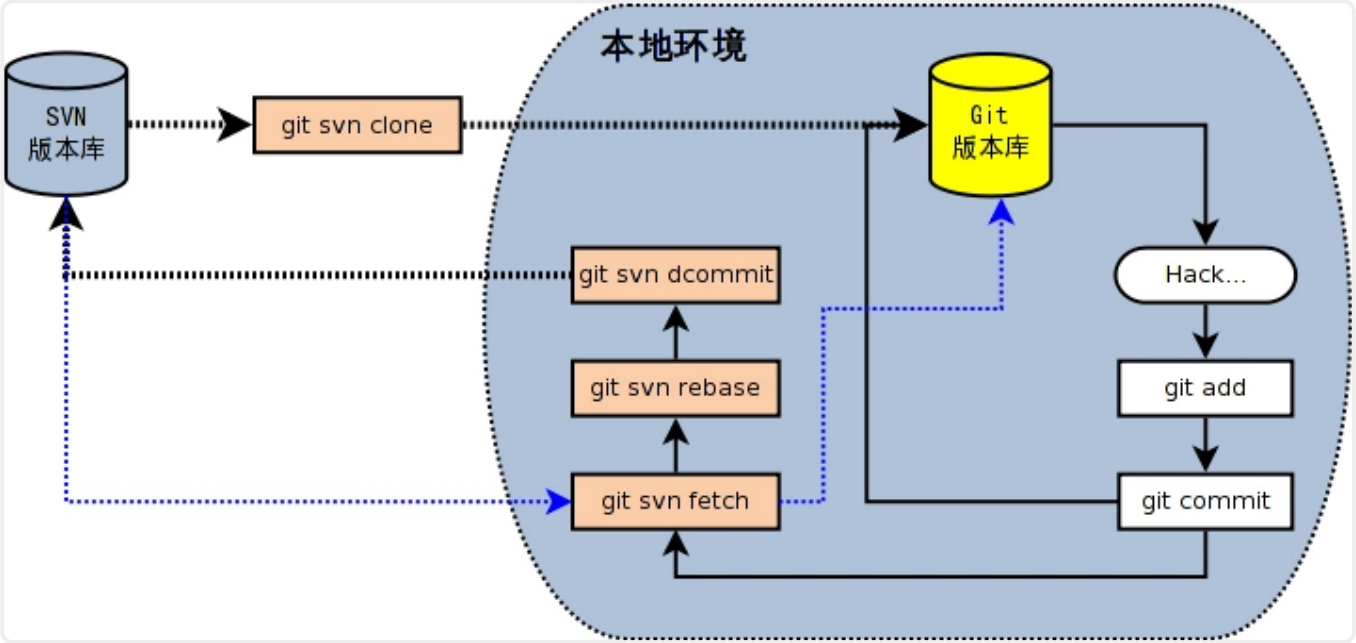


Git-SVN常用命令

> 看到Git-SVN不要懵！是的，在Git中甚至有命令可以直接和SVN进行桥接，Git 中所有与SVN桥接的基础命令就是 `git svn`

若服务器使用的 SVN，但是本地想要体验 Git 的本地分支，离线操作等功能，可以使用 Git-SVN功能。

常用操作如下（图片来源于网络）：




```
# 下载一个 SVN 项目和它的整个代码历史，并初始化为 Git 代码库
$ git svn clone -s [repository]

# 查看当前版本库情况
$ git svn info

# 取回远程仓库所有分支的变化
$ git svn fetch

# 取回远程仓库当前分支的变化，并与本地分支变基合并
$ git svn rebase

# 上传当前分支的本地仓库到远程仓库
$ git svn dcommit
```

```
# 拉取新分支，并提交到远程仓库
$ svn copy [remote_branch] [new_remote_branch] -m [message]

# 创建远程分支对应的本地分支
$ git checkout -b [local_branch] [remote_branch]
```

初 始 化

> 从本节开始，除特殊说明，以下命令均适用于 Git 与 Git-SVN。

```
# 在当前目录新建一个Git代码库
$ git init

# 下载一个项目和它的整个代码历史 [Git only]
$ git clone [url]
```

配　置

```
# 列举所有配置
$ git config -l
```

```
# 为命令配置别名
$ git config --globalalias.co checkout
$ git config --globalalias.ci commit
$ git config --globalalias.st status
$ git config --globalalias.br branch

# 设置提交代码时的用户信息
$ git config [--global] user.name "[name]"
$ git config [--global] user.email "[email address]"
```

Git 用户的配置文件位于 `~/.gitconfig`

Git 單個倉庫的配置文件位於 `~/$PROJECT_PATH/.git/config`

增刪文件

```
# 添加当前目录的所有文件到暂存区
$ git add .

# 添加指定文件到暂存区
$ git add <file1><file2>...

# 添加指定目录到暂存区，包括其子目录
$ git add <dir>

# 删除工作区文件，并且将这次删除放入暂存区
$ git rm [file1] [file2] ...

# 停止追踪指定文件，但该文件会保留在工作区
$ git rm --cached [file]
```

```
# 改名文件，并且将这个改名放入暂存区
$ git mv [file-original] [file-renamed]
```

> 把文件名file1 添加到.gitignore 文件裡，Git 會停止跟踪file1 的狀態。

分 支

```
# 列出所有本地分支
$ git branch

# 列出所有本地分支和远程分支
$ git branch -a

# 新建一个分支，但依然停留在当前分支
$ git branch [branch-name]

# 新建一个分支，并切换到该分支
$ git checkout -b [new_branch] [remote-branch]

# 切换到指定分支，并更新工作区
$ git checkout [branch-name]

# 合并指定分支到当前分支
$ git merge [branch]

# 选择一个 commit，合并进当前分支
$ git cherry-pick [commit]

# 删除本地分支，-D 参数强制删除分支
$ git branch -d [branch-name]
```

```
# 删除远程分支
$ git push [remote] :[remote-branch]
```

提 交

```
# 提交暂存区到仓库区
$ git commit -m [message]

# 提交工作区与暂存区的变化直接到仓库区
$ git commit -a

# 提交时显示所有 diff 信息
$ git commit -v

# 提交暂存区修改到仓库区，合并到上次修改，并修改上次的提交信息
$ git commit --amend -m [message]

# 上传本地指定分支到远程仓库
$ git push [remote] [remote-branch]
```

拉 取

```
# 下载远程仓库的所有变动 (Git only)
```

```
$ git fetch [remote]

# 显示所有远程仓库 (Git only)
$ git remote -v

# 显示某个远程仓库的信息 (Git only)
$ git remote show [remote]

# 增加一个新的远程仓库，并命名 (Git only)
$ git remote add [remote-name] [url]

# 取回远程仓库的变化，并与本地分支合并，(Git only)，若使用 Git-SVN，请查看第三节
$ git pull [remote] [branch]

# 取回远程仓库的变化，并与本地分支变基合并，(Git only)，若使用 Git-SVN，请查看第三节
$ git pull --rebase [remote] [branch]
```

撤銷

```
# 恢复暂存区的指定文件到工作区
$ git checkout [file]

# 恢复暂存区当前目录的所有文件到工作区
$ git checkout .

# 恢复工作区到指定 commit
$ git checkout [commit]

# 重置暂存区的指定文件，与上一次 commit 保持一致，但工作区不变
$ git reset [file]

# 重置暂存区与工作区，与上一次 commit 保持一致
```

```
$ git reset --hard

# 重置当前分支的指针为指定 commit，同时重置暂存区，但工作区不变
$ git reset [commit]

# 重置当前分支的HEAD为指定 commit，同时重置暂存区和工作区，与指定 commit 一致
$ git reset --hard [commit]

# 新建一个 commit，用于撤销指定 commit
$ git revert [commit]

# 将未提交的变化放在储藏区
$ git stash

# 将储藏区的内容恢复到当前工作区
$ git stash pop
```

查 詢

```
# 查看工作区文件修改状态
$ git status

# 查看工作区文件修改具体内容
$ git diff [file]

# 查看暂存区文件修改内容
$ git diff --cached [file]

# 查看版本库修改记录
$ git log

# 查看某人提交记录
```

```
$ git log --author=someone

# 查看某个文件的历史具体修改内容
$ git log -p [file]

# 查看某次提交具体修改内容
$ git show [commit]
```

喜歡此內容的人還喜歡

把Spring Cloud 給拆了！詳解每個組件的作用

快學Java

為什麼阿里巴巴禁止使用存儲過程?

Java基基

記一次線上請求偶爾變慢的排查

Java知音