**UPDATE：**这里记录的“Git 工作流程”比较适合于团队大部分人没有使用过 Git 的情况，能够简单无障碍入门；以及，适合于系统比较大，不同的人工作于不同的部分，彼此进度不一的情况。日志写作时，我们正好符合这两个情况，所以采用了所述的流程。随着时间的推移，团队成员使用 Git 已经比较熟练，而我们系统的开发也逐渐成熟，现已转用了另一种流程。大体而言，跟这个思路比较接近：http://nvie.com/posts/a-successful-git-branching-model/ 。

——

注：

Git 的强大、灵活、好用，毋庸置疑。

但也正是 Git 的灵活性，在公司推行时，如何执行统一的流程成为一个问题。我想了不少时间才制订出一个办法。

目的是规范、统一。还有就是，Windows 下的同事，特别是不熟悉命令行的同事，怎样才能使他们好理解，并且觉得简单（之前大家觉得概念太多，难以理解；步骤多，记不住，不小心就搞错，冲突频发）。

说到 Windows，Git 在 Windows 下不如 Linux 下好使，这也是一个需要考虑的问题。

同样是在公司 Wiki 上写的，再次拿到 Blog 来凑数呵呵～。

**版本控制**

——————  
简单地说，就是将在本地开发的代码，定时推送到服务器。每一次修改，记录下它的作者、时间及修改说明等。

相对的，我们也可以从服务器下拉其他人推送的代码，并了解它的作者、时间、修改说明及其具体的修改内容。

这样，版本控制给团队协作开发提供了极大的方便。即使是一个人开发，因为它记录下了我们整个的开发历史，也是极有帮助和价值的。

比如，如果某次修改甚至整个系统出现问题，它也能帮助找回我们珍贵的代码。

**分布式版本控制**

——————————–  
更进一步，分布式版本控制工具使得我们在本机上即拥有完整的功能，不依赖于服务器，使用更为方便。它们往往也提供其它更好用或更强大的功能，比如灵活的分支管理。

**Git**

——–  
Git 是 Linux 之父 Linus Trovalds，为管理 Linux 内核代码而建立的，被认为是分布式版本控制工具中的顶级水准。智能、友好、强健、高效。

**Git 工作流程**

—————————-  
1、使用中央服务器辅助协作；

2、每人在服务器拥有一个以自己 id 为名称的分支；

3、各人只许推送更新到自己的分支，不允许推送到别人的分支或者 master；

4、master 由专人管理，在合适时 merge 其它分支（开发初期每日自动 merge 各人分支，生产化后则由人工 merge 经过 review 的分支）；

5、代码修改 merge 到 master 后，将同步到所有终端。

【TIP】：熟悉之后，你可以创建类似 myId\_branchName 的其它分支。

【TIP】：以上只是概念介绍，至于具体的操作，请参考：[《Windows 下 Git 配置与使用指南》](http://wangcongming.info/2010/07/31/git-%E7%B3%BB%E5%88%97%E4%B9%8B%E4%B8%89%EF%BC%9Awindows-%E4%B8%8B-git-%E9%85%8D%E7%BD%AE%E4%B8%8E%E4%BD%BF%E7%94%A8%E6%8C%87%E5%8D%97/)、 [《Git 进阶功能》](http://wangcongming.info/2010/07/31/git-%E7%B3%BB%E5%88%97%E4%B9%8B%E5%9B%9B%EF%BC%9Agit-%E8%BF%9B%E9%98%B6%E5%8A%9F%E8%83%BD/)。