一、在 Windows 上安装 Git

msysgit 是 Windows 版的 Git，从 http://msysgit.github.io/下载，然后按默认选项安装即可。

安装完成后，在开始菜单里找到“Git”->“Git Bash”，蹦出一个类似命令行窗口的东西，就说明 Git 安装成功!

安装完成后，在命令行输入：

```
$ git config --global user.name "Your Name"
$ git config --global user.email "email@example.com"
```

二、Git 的使用

1、初始化仓库

```
git init
```

此时会在当前目录＊假如为($WORK）生成一个.git 的目录文件. 这个.git 目录就是Git 仓库. 其中存放的是我们所提交的文档索引内容，Git 可基于文档索引内容对其所管理的文档进行内容追踪，从而实现文档的版本控制。工作树是包含 .git 的目录$WORK.如图所示

2、生成快照，并将快照存放到索引区域

```
git add .
```

此命令将当前目录下所有文件生成快照. 此命令通常在忘记哪些文档更新，编辑或者删除的情况下使用.也可以指定一个或者多个文件，多个文件之间用空格隔开.

```
git add a.txt b.java
```

在我们使用的过程中. 有时候编码器产生的一些临时文件是没有必要提交 git 管理的，可以用以下方法来进行忽略.

```
echo 'zh/' >> .gitignore
```

这样对于 zh 目录及内容是不会被 git 所管理的.

注意如果这里使用 > 定向符的话，则.gitignore 文件只能保留最后一次的内容的.

3、提交索引到仓库中

```
git commit
```

此时系统会自动调用系统默认的文本编辑器，让用户输入版本更新备注信息，对于简单的备注信息要，可以用

```
git commit -m "这里是版本更新信息"
```

另外对于 git commit 命令还有一个-a 的参数.此选项可以将所有被修改的文档或者删除的文档的当前状态提交到仓库中.

4、查看版本历史

```
git log
```

如果想看一下每一次版本的大致变动情况，可以使用

```
git log --stat --summary
```

撤销与恢复

git-reset 命令有三个选项：--mixed 、 --soft 和 --hard 。我们在日常使用中仅使用前两个选项；第三个选项由于杀伤力太大，容易损坏项目仓库，需谨慎使用。 --mixed 是

git-reset 的默认选项，它的作用是重置索引内容，将其定位到指定的项目版本，而不改变你的工作树中的所有内容，只是提示你有哪些文件还未更新 --soft 选项既 不触动索引的位置，也不改变工作树中的任何内容，但是会要求它们处于一个良好的次序之内。该选项会保留你在工作树中的所有更新并使之处于待提交状态。 如果执行过上面的命令，想查看实际效果，可以使用命令
git log
git reset --hard SHA1_HASH
git-status

三、BUG 管理系统
Git 提供了一个 stash 功能，可以把当前工作现场保存起来，等恢复现场后继续工作：
$ git stash
Saved working directory and index state WIP on dev: 6224937 add merge
HEAD is now at 6224937 add merge

现在，用 git status 查看工作区，就是干净的（除非有没有被 Git 管理的文件），因此可以放心地创建分支来修复 bug。
首先确定要在哪个分支上修复 bug，假定需要在 master 分支上修复，就从 master 创建临时分支：
$ git checkout master
Switched to branch 'master'
Your branch is ahead of 'origin/master' by 6 commits.
$ git checkout -b issue-101
Switched to a new branch 'issue-101'

现在修复 bug，需要把“Git is free software ...”改为“Git is a free software ...”，然后提交：
$ git add readme.txt
$ git commit -m "fix bug 101"
[issue-101 cc17032] fix bug 101
1 file changed, 1 insertion(+), 1 deletion(-)

修复完成后，切换到 master 分支，并完成合并，最后删除 issue-101 分支：
$ git checkout master
Switched to branch 'master'
Your branch is ahead of 'origin/master' by 2 commits.
$ git merge --no-ff -m "merged bug fix 101" issue-101
Merge made by the 'recursive' strategy.
readme.txt | 2 +-
1 file changed, 1 insertion(+), 1 deletion(-)
$ git branch -d issue-101Deleted branch issue-101 (was cc17032).

用 git stash list 命令查看保存的工作现场：

```
$ git stash list
stash@{0}: WIP on dev: 6224937 add merge
```

工作现场还在，Git 把 stash 内容存在某个地方了，但是需要恢复一下，有两个办法：

一是用 git stash apply 恢复，但是恢复后，stash 内容并不删除，你需要用 git stash drop 来删除；

另一种方式是用 git stash pop，恢复的同时把 stash 内容也删了：

```
$ git stash pop
#Dropped refs/stash@{0} (f624f8e5f082f2df2bed8a4e09c12fd2943bdd40)
```

再用 git stash list 查看，就看不到任何 stash 内容了：

```
$ git stash list
```

你可以多次 stash，恢复的时候，先用 git stash list 查看，然后恢复指定的 stash，用命令：

```
$ git stash apply stash@{0}
```