用苹果系统已经有一段时间了，越来越喜欢这个系统，不管是一般的使用、做设计、做开发，都非常适合。最近也对系统做了一些调整，使得开发环境更加的舒服，记录下来，免得忘了。

实际的过程是漫长而曲折的，跟下面所写的顺序完全没有关系。如果按照下面的步骤操作，中间难免会遇到一些问题（大多是由于墙导致的），那时再自行Google吧。

[**基础设施**](http://www.gocalf.com/blog/make-mac-better-for-development.html#id9)

[**iTerm**](http://www.gocalf.com/blog/make-mac-better-for-development.html#id10)

对于几乎离不开shell的开发者来说，一款优秀的终端程序是基础的基础。

毫无疑问，我用的是[iTerm2](http://www.iterm2.com/)。如果说访问互联网世界的入口是Chrome，那么访问程序世界的入口就是iTerm了。



字体选择一款好看的等宽字体即可，比如常用的Consolas、 Curier New等。我用的是[Source Code Pro](https://github.com/adobe/source-code-pro)。中文使用Microsoft YaHei字体。

配色当然首选[Solarized](https://github.com/altercation/solarized)的暗色系，在Github上可以找到专门提供给iTerm2用的配色文件[Solarized Dark.itemcolors](https://github.com/altercation/solarized/blob/master/iterm2-colors-solarized/Solarized%20Dark.itermcolors)。

[**Homebrew**](http://www.gocalf.com/blog/make-mac-better-for-development.html#id11)

[Homebrew](http://brew.sh/) installs the stuff you need that Apple didn't.

我电脑上的大部分工具都是通过homebrew安装和管理的，非常方便。虽然提供类似功能的还有[Fink](http://www.finkproject.org/)和[macports](http://www.macports.org/)，但我认为Homebrew是最方便的。具体的就不在这里比较了，大家可以自行调研。当然，至少选择一个来帮助自己安装盒管理软件包，会让很多事情变得更容易。

安装Homebrew非常方便，首先要安装[Command Line Tools for Xcode](https://developer.apple.com/downloads/index.action)，然后运行

ruby -e "$(curl -fsSL https://raw.github.com/Homebrew/homebrew/go/install)"

在使用Homebrew的过程中，要注意经常更新它。在brew的世界中，24小时就已经是非常久了。所以在安装某个包或者做其他操作之前，一般都要运行`brew update`和`brew doctor`，前者用于更新Homebrew自身和各个软件包，后者用于排查可能会遇到的问题。我之前没注意这个（也是因为那时候还没太依赖于Homebrew），有一次安装一个程序，总是提示我有一个依赖没有装，但那个依赖明明就在那儿，百思不得其解，还跑到github上发issue去问，被拍了一顿。其实只要brew update一下就知道，我安装的那个依赖包太老了，升级了就好了。

在安装某个程序前，我一般也要先brew info $FORMULA一下看看，了解一下有没有什么值得注意的参数，安装后有什么需要手动进行的后续操作。

Homebrew默认会掌控系统中的/usr/local目录，目前我这个目录也就只是给Homebrew用了，其他的东西都不往里放。在使用Homebrew的过程中，一般都避免使用sudo进行操作，实际上现在的版本用了sudo也就没法使了。

[**Python**](http://www.gocalf.com/blog/make-mac-better-for-development.html#id12)

苹果系统自带了好几个版本的Python，装在/Library/Python中。不过版本都不新，为了便于维护，还是自己装一个Python吧。用brew install python可以安装最新的Python 2.x，用brew install python3可以安装最新的Python 3.x。

Homebrew的Python已经安装了pip，用于管理Python的软件包。根据brew info python提供的提示，运行下列命令对pip进行更新：

|  |  |
| --- | --- |
| 1  2 | pip install --upgrade setuptools  pip install --upgrade pip |

我现在使用[virtualenv](http://virtualenv.readthedocs.org/en/latest/)来管理Python的环境，用[virtualenvwrapper](http://virtualenvwrapper.readthedocs.org/en/latest/)来方便地使用virtualenv。分别用pip进行安装：

|  |  |
| --- | --- |
| 1  2 | pip install virtualenv  pip install virtualenvwrapper |

virtualenvwrapper提供了很多方便的命令，还支持命令的tab completion，这些都包含在virtualenvwrapper.sh文件中。在~/.bash\_profile中引入该文件来激活相关的命令和功能：

|  |  |
| --- | --- |
| 1  2  3  4 | export VIRTUAL\_ENV\_DISABLE\_PROMPT=1  export WORKON\_HOME=$HOME/.virtualenvs  export PROJECT\_HOME=$HOME/projects  [ -f "/usr/local/bin/virtualenvwrapper.sh" ] && source "/usr/local/bin/virtualenvwrapper.sh" |

准备好后就可以用mkvirtualenv ENVNAME来创建一个新的virtualenv，用workon命令来切换环境，用deactive退出虚拟环境。更多的命令可以查看[virtualenvwrapper](http://virtualenvwrapper.readthedocs.org/en/latest/)的文档。

另外，[ipython](http://ipython.org/)是非常好用的Python的交互式终端，比Python自身的命令行提供了更丰富和方便的功能，建议使用。通过pip install ipython即可安装。不过目前我还没想清楚要不要把它装在某个虚拟环境中。按理说应该是要在任何一个虚拟环境中都能用ipython的，否则就要给每个虚拟环境都装一次，岂不是很浪费空间？这个问题以后再考虑吧。

ipython除了shell console外，还提供Qt console，详细的信息查看官方的介绍吧。

[**科学上网**](http://www.gocalf.com/blog/make-mac-better-for-development.html#id13)

由于一些众所周不知的原因，这个世界上存在着一些不存在的网站。本来不应该为不存在的事物所烦恼，但对于开发人员来说，不存在的世界中却存在着一些非常有价值的资源。所以，需要用科学的方法访问互联网。

我目前主要用到了[goagent](https://code.google.com/p/goagent/)、SSH tunnel、[proxychains-ng](https://github.com/rofl0r/proxychains-ng/)、[dnscrypt-proxy](https://github.com/jedisct1/dnscrypt-proxy)和[unbound](http://unbound.net/)。

goagent需要在Google App Engine上用自己的账号安装服务端，在本地用python运行客户端。具体的安装方法参见官网介绍。我创建一个virtualenv给它使用，在这个虚拟环境中安装相关的Python依赖。

|  |  |
| --- | --- |
| 1  2  3  4 | mkvirtualenv goagent  pip install pyopenssl  pip install pycrypto  pip install gevent |

用goagent访问HTTPS网站的时候，需要安装证书。现在的goagent已经可以自动安装证书了（需要用sudo权限运行）。如果是第一次使用goagent，可以先将goagent的local目录中的ca.cer、ca.key和certs目录内的文件都删除，删除浏览器或系统中的goagent ca证书，然后用sudo权限启动goagent，它会自行安装证书到系统中。我建议一直使用sudo权限运行goagent。

在Mac系统中，利用系统的launchd来控制goagent的随系统（以root权限）启动。可以在/Library/LaunchDaemons中创建一个扩展名为.plist的文件，内容为（需要根据你的实际环境进行调整）：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | <?xml version="1.0" encoding="UTF-8"?>  <!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">  <plist version="1.0">  <dict>  <key>Label</key>  <string>com.github.calfzhou.goagent.local</string>  <key>ProgramArguments</key>  <array>  <string>YOUR\_OWN\_PATH/.virtualenvs/goagent/bin/python</string>  <string>proxy.py</string>  </array>  <key>RunAtLoad</key>  <true/>  <key>ServiceDescription</key>  <string>Goagnet proxy</string>  <key>StandardErrorPath</key>  <string>/dev/null</string>  <key>StandardOutPath</key>  <string>/dev/null</string>  <key>WorkingDirectory</key>  <string>YOUR\_OWN\_PATH/goagent/local</string>  </dict>  </plist> |

给系统的网络连接设置代理，进入System Preferences -> Network，选择使用的网络，点击Advanced...，在Proxies页中勾选“Automatic Proxy Configuration”，在URL内填入“<http://127.0.0.1:8086/proxy.pac>”，保存生效。

对于Chrome浏览器，推荐使用[Proxy SwichySharp](https://chrome.google.com/webstore/detail/proxy-switchysharp/dpplabbmogkhghncfbfdeeokoefdjegm)插件。鉴于Google所有的服务都访问不顺畅，这个链接可能不太容易访问到，可以考虑使用[chrome extension downloader](http://chrome-extension-downloader.com/)网站来直接下载插件的.crx文件。用非Chrome浏览器打开该网站，输入Proxy SwichySharp的ID（dpplabbmogkhghncfbfdeeokoefdjegm），下载保存，然后在Chrome的extensions页面中把.crx文件拖进去即可。goagent里也提供了该插件的.crx文件以及配置文件，可以直接使用（作者想的真周到啊）。

系统的网络连接代理和Chrome的代理插件基本能解决大部分网页访问的需求，比如苹果自带的Safari就会默认通过系统的代理，也就能科学地上网了。不过wget、curl等shell命令却无法直接使用这些代理，它们会根据环境变量http\_proxy和https\_proxy来访问网络。为了方便，在~/.bash\_profile中添加：

|  |  |
| --- | --- |
| 1  2  3 | # Setup or dismiss (goagent) proxy for curl, wget, etc.  alias gaproxy='export http\_proxy=http://127.0.0.1:8087 https\_proxy=http://127.0.0.1:8087'  alias noproxy='unset http\_proxy https\_proxy' |

在需要wget或者curl某个不存在的网页前，通过gaproxy命令开启代理，使用完毕后通过noproxy关闭代理即可。

有的时候goagent会抽疯，一个备选的代理是必需的。我一般会利用SSH隧道，通过gocalf网站所在的主机建立socks代理。如果你也有一台在国外的服务器，可以通过这个命令在本地开启socks5代理服务：

|  |  |
| --- | --- |
| 1 | ssh -D LOCAL\_PORT(7070) -p REMOTE\_SSH\_PORT(22) USER\_NAME@SERVER\_ADDRESS |

对于不支持http\_proxy和https\_proxy的程序，我会使用[proxychains-ng](https://github.com/rofl0r/proxychains-ng/)。通过brew install proxychains-ng即可安装，运行的命令是proxychains4。这个有点儿像Windows里的SocksCap，但是更强大些，比如想从一个不存在的svn站点下载代码，可以用proxychains4 svn checkout xxxx实现。

最近发现[Dropbox](https://www.dropbox.com/)的客户端即使设置上goagent代理也不好使（网页倒是没问题），所幸Dropbox（还有Facebook等）不存在的原因只是域名解析被人为破坏了，只要能解析出正确的ID地址，不用代理也能够访问。为了防止域名解析被恶意破坏，我又祭出了[dnscrypt-proxy](https://github.com/jedisct1/dnscrypt-proxy)这个法宝。不幸的是，dnscrytp-proxy的下载站点本身就是不存在的，要用前面提到的gaproxy激活代理后才能下载成功：

|  |  |
| --- | --- |
| 1  2  3 | gaproxy  brew install dnscrypt-proxy  noproxy |

安装后根据提示设置成开机自动启动即可。默认的话它会监听127.0.0.1的53端口提供DNS服务，上游使用OpenDNS服务（可自行配置），并使用加密通信来防止DNS污染。将网络连接的DNS设置为127.0.0.1（System Preferences -> Network -> 当前使用的网络 -> Advanced... -> DNS -> DNS Servers），就会发现即使没有goagent，Dropbox、Facebook等网站也变得存在了。

dnscrypt-proxy有个缺点就是没有缓存功能，每次来个域名都要去远程服务器上解析一次，速度很慢，非常影响上网的体验，建议配合具备DNS缓存的工具一起使用，比如[unbound](http://unbound.net/)、[dnsmasq](http://www.thekelleys.org.uk/dnsmasq/doc.html)等。二者都可以通过Homebrew安装，非常方便。当然要配合使用，就需要一些配置，在性能方面也需要做一些优化，这里就不再仔细说了。

[**友好的Shell**](http://www.gocalf.com/blog/make-mac-better-for-development.html#id14)

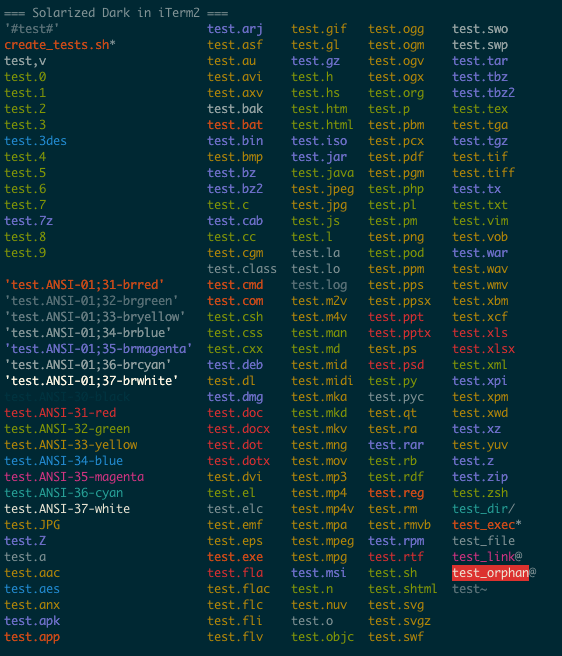
有了iTerm还不够，要让shell变得好用，还需要再做一些配置。

[**Bash Profile**](http://www.gocalf.com/blog/make-mac-better-for-development.html#id15)

~/.bash\_profile的作用就不用我多说了，在这里可以对shell进行很多的个性化配置。参考[mathiasbynens的dotfiles](https://github.com/mathiasbynens/dotfiles)项目，我也为自己打造了舒适的shell环境，相关的配置保存在[GitHub - calfzhou - dotfiles - bash](https://github.com/calfzhou/dotfiles/tree/master/bash)里面，对Mac和Linux都是可以的，让我在不同的服务器上也有相同的操作体验。涉及到的内容很多，就不逐一介绍了，比较重要的几点下面会提到。

[**Dir Colors**](http://www.gocalf.com/blog/make-mac-better-for-development.html#id16)

虽然iTerm本身已经设置好了Solarized配色，但是ls的时候并不一定有颜色。ls没有颜色绝对是让人难以忍受的，在[bash\_inc/alias](https://github.com/calfzhou/dotfiles/blob/master/bash/bash_inc/aliases)里除了根据系统中ls支持的参数让ls的输出显示出颜色外，还通过配置LS\_COLOR让色彩更丰富，可以让不同类型的文件有不同的颜色，看起来非常的清晰。推荐使用[dircolors-solarized](https://github.com/seebi/dircolors-solarized)提供的配色文件，在Mac OS X的采用了Solarized Dark配色的iTerm2里看起来会是这个样子：



iTerm2中Solarized Dark系的dircolors效果

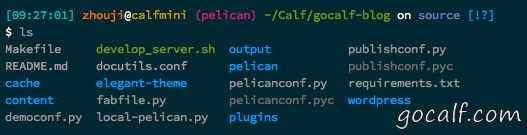
[**Bash Prompt**](http://www.gocalf.com/blog/make-mac-better-for-development.html#id17)

在使用shell的过程中，命令提示符会一直陪伴着我们，是时候扔掉默认的提示符了。[我的命令提示符](https://github.com/calfzhou/dotfiles/blob/master/bash/bash_inc/bash_prompt)用不同的颜色分别显示出当前时刻、当前用户、当前主机（通过颜色标识是否通过是通过SSH登录的）、当前使用的Python virtualenv（如果有的话）、当前目录、当前目录所在的git分支和状态（如果是git项目的目录的话）。

关于主机名，Mac系统下默认应该是localhost，可以通过scutil命令修改成想要的值：

|  |  |
| --- | --- |
| 1  2  3 | $ sudo scutil --set HostName MYNAME  $ hostname  MYNAME |

配合上iTerm的配色、ls的颜色等，我的shell看起来是这样的：



我的iTerm2的效果

[**Bash Completion**](http://www.gocalf.com/blog/make-mac-better-for-development.html#id18)

在shell里面输入命令的时候，如果只记得开头几个字母，后面的记不清楚了也没关系，输入几个字母后，按TAB键就可以自动补全或者提示出所有可行的命令。在输入文件名的时候也可以通过TAB键自动补全或者提示出有效的文件文来，这个功能是非常方便的。Homebrew又额外提供了一些bash completion功能，可以通过brew install bash-completion进行安装，并会生成/usr/local/etc/bash\_completion文件，在~/.bash\_profile中source一下这个文件，就可以把Homebrew提供的命令补全包含进来。如果通过Homebrew安装了别的工具包，比如git、svn等，它们也会有各自相应的命令补全文件，存放在/usr/local/etc/bash\_completion.d目录中，都会被刚才那个文件自动引入。

[**Shell Commands**](http://www.gocalf.com/blog/make-mac-better-for-development.html#id19)

用惯了Linux（CentOS）里面的shell命令，就无法忍受Mac系统中那些落后的shell命令了，像ls、date、ps、echo、grep等等，功能都特别少。在忍无可忍之后，终于决定用[GNU Coreutils](http://www.gnu.org/software/coreutils/)替换它们。用brew install coreutils就可以搞定了，安装完成后，根据提示，将/usr/local/opt/coreutils/libexec/gnubin和/usr/local/opt/coreutils/libexec/gnuman分别添加到$PATH和$MANPATH中即可。

grep命令不在coreutils，可以通过brew install grep --default-names搞定（如果说找不到grep可以先brew tap homebrew/dupes）。当然还有很多其他特别有用的命令，就不一一细说了，反正想到什么，只要用brew info或者brew search找找看就行。

[**编辑器和IDE**](http://www.gocalf.com/blog/make-mac-better-for-development.html#id20)

[**VIM**](http://www.gocalf.com/blog/make-mac-better-for-development.html#id21)

其实我也不是VIM重度使用者，现在写超过两个文件的Python代码、Java代码等都会使用专门的IDE。我看很多人都喜欢用VIM来写复杂的项目代码，但他们的VIM都没有做任何额外的设置，自身对VIM的快捷键和命令也不熟悉，只是把VIM当成一个连移动光标都很费劲的编辑器来用，写代码的效率可想而知。而且像Python这种脚本语言，很多错误只有到运行到那句话的时候才会有效果，VIM没有足够的只能针对某一个语言做太多的静态分析。

~/.vimrc用于对VIM进行各种设置，如果没有这个文件，赶紧创建一个吧。复杂的语法高亮、配色、插件等都可以放在~/.vim里面。VIM自身对插件没有很好地管理，我选择了[vim-pathogen](https://github.com/tpope/vim-pathogen)来管理所需要的VIM扩展。只要把[pathogen.vim](https://github.com/tpope/vim-pathogen/blob/master/autoload/pathogen.vim)放在~/.vim/autoload目录中，把所需的扩展包放在~/.vim/bundle目录下即可。

同样地，我也把自己的VIM设置和依赖放在GitHub上（[GitHub - calfzhou - dotfiles - vim](https://github.com/calfzhou/dotfiles/tree/master/vim)），在别的server上直接clone下来保持一致的操作体验。

我现在使用[riv.vim](https://github.com/Rykka/riv.vim)扩展来写reStructuredText文件（.rst），操作起来非常方便。

如果不满意苹果系统自带的较低版本的VIM，或者想用GUI界面的VIM，可以利用Homebrew进行安装，Formula是vim和macvim。

[**Sublime Text**](http://www.gocalf.com/blog/make-mac-better-for-development.html#id22)

[Sublime Text](http://www.sublimetext.com/)是我现在主要使用的GUI编辑器，但看其首页的动画演示就会觉得非常cool。以前还经常用Ultra Edit和Notepad++，现在基本都不用了。在GUI程序中，我一般会使用Solarized Light配色方案，总感觉大部分GUI程序用暗色系就很丑（PyCharm系列除外）。

具体的配置也就不多说了。注意Sublime Text原生不支持GBK编码的文件，需要安装扩展包，即使这样，在编辑GBK编码的文件时，它会生成一个临时文件进行操作，保存的时候再写回去，体验上还是有些不爽。

还有一个地方我也一直没搞清楚，就是中文字体不是刚刚好跟两个英文字母一样宽。在编辑reStructuredText文件的时候，还是会比较麻烦的。

[**Python IDE**](http://www.gocalf.com/blog/make-mac-better-for-development.html#id23)

以前一直用VIM做Python开发，是因为没找到好用的Python IDE，现在我用[JetBrains](http://www.jetbrains.com/)出的[PyCharm](http://www.jetbrains.com/pycharm/)。不要跟我争，这绝对是世界上最好用的Python IDE，没有之一。

PyCharm里强烈推荐自带的Darcula配色，感觉在GUI界面中，这个配色比Solarized Dark要舒服一些。PyCharm的默认配置基本就很好用了，它对Python代码的可读性检测方面要求还是比较严格的，空行、空格不合适都会有提示。如果Python项目中包含其他类型的文件（如shell脚本、html页面等），它也有相应的插件可以对这些文件进行语法高亮。

一直觉得如果JetBrains出一个通用的编辑器，应该能把Sublime Text甩出好几条街去。

[**Java IDE**](http://www.gocalf.com/blog/make-mac-better-for-development.html#id24)

好多人都用Eclipse写Java代码（包括Android开发），真想不通为什么那么难用的IDE还那么受欢迎。可能学校里交Java的时候都用的Eclipse吧，就像直到现在都还有很多人用VS6做C++开发一样。对于Android开发，可能跟之前Google推Eclipse with ADT有关吧。不过现在Google也认识到了Eclipse的不足，转身投入[JetBrains](http://www.jetbrains.com/)的[IntelliJ IDEA](http://www.jetbrains.com/idea/)的怀抱了。JetBrains出品的个个都是精品啊，微软里面很多项目组也都在使用JetBrains出的[ReSharper](http://www.jetbrains.com/resharper/)。

我现在也是用IntelliJ IDEA来做Java开发，使用体验跟PyCharm类似。Java项目的依赖管理用Maven，Homebrew里也提供了安装。建议目前使用Maven 3.0.\*：

|  |  |
| --- | --- |
| 1  2 | brew tap homebrew/versions  brew install maven30 |

当然如果要用最新的3.2.\*，可以直接brew install maven。

[**版本控制**](http://www.gocalf.com/blog/make-mac-better-for-development.html#id25)

[**Git**](http://www.gocalf.com/blog/make-mac-better-for-development.html#id26)

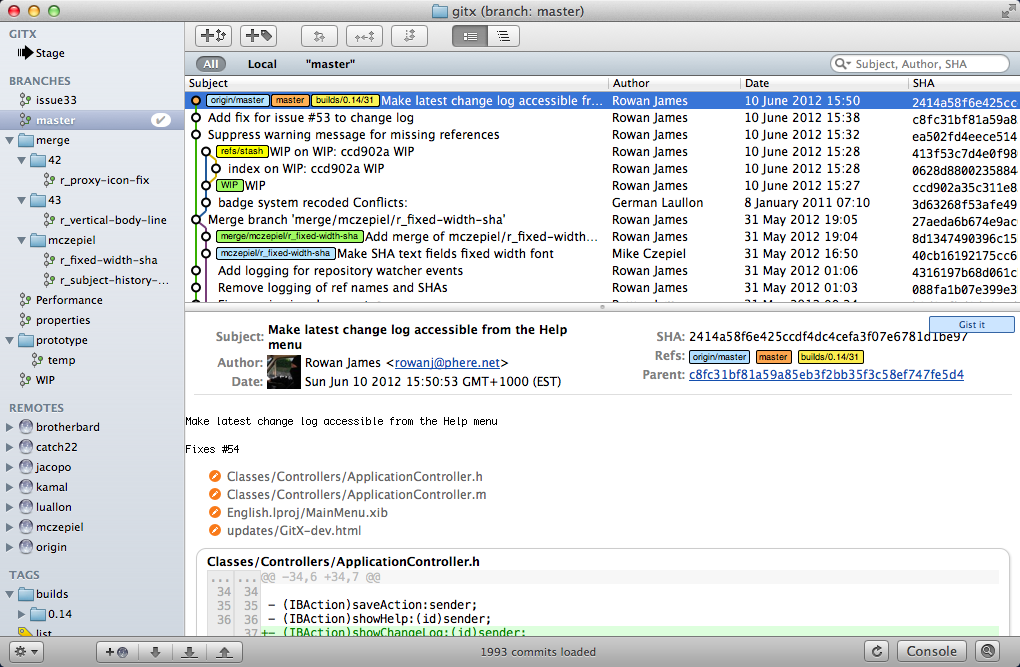
用Homebrew可以方便地安装最新的Git。如果像上面介绍的那样，用Homebrew安装并使用了bash-completion，那么装好git后，git指令也会按TAB键补全了，再也不动担心记不住git指令。

~/.gitconfig可以用来定义很多个性化的设置，可以直接编辑或者通过git config --global进行设置。类似的，我也把我的配置文件保存在GitHub上（[GitHub - calfzhou - dotfiles - git](https://github.com/calfzhou/dotfiles/tree/master/git)），以便在不同的地方有同样的操作体验。

顺便提一下，用git管理项目版本的话，推荐使用[git-flow](https://github.com/nvie/gitflow)管理分支和版本，通过Homebrew可以直接安装它，同样也会带有TAB自动补全功能。关于这种分支管理的模型，可以阅读[A successful Git branching model](http://nvie.com/posts/a-successful-git-branching-model/)。

如果想在公开的git仓库中保存一些比较隐私的信息，可以利用[git-encrypt](https://github.com/shadowhand/git-encrypt)来加密其中的一部分文件。这个也可以直接通过Homebrew安装。大致的介绍在之前的文章[用Git和云存储保存隐私信息](http://www.gocalf.com/blog/git-encrypt.html)中略有介绍。

如果还想用个GUI的Git客户端，我用的是[GitX-dev](http://rowanj.github.io/gitx/)，主要用来直观地观察分支的演化情况。



GitX-dev界面演示

[**SVN**](http://www.gocalf.com/blog/make-mac-better-for-development.html#id27)

用了Git，就再也不想用SVN了。不过有时候也难免会需要用，用Homebrew安装个新版本的SVN吧，同样也会有TAB键补全哦。