Git 守护进程

对于提供公共的，非授权的只读访问，我们可以抛弃 HTTP 协议，改用 Git 自己的协议，这主要是出于性能和速度的考虑。Git 协议远比 HTTP 协议高效，因而访问速度也快，所以它能节省很多用户的时间。

重申一下，这一点只适用于非授权的只读访问。如果建在防火墙之外的服务器上，那么它所提供的服务应该只是那些公开的只读项目。如果是在防火墙之内的服务器上，可用于支撑大量参与人员或自动系统（用于持续集成或编译的主机）只读访问的项目，这样可以省去逐一配置 SSH 公钥的麻烦。

但不管哪种情形，Git 协议的配置设定都很简单。基本上，只要以守护进程的形式运行该命令即可：

git daemon --reuseaddr --base-path=/opt/git/ /opt/git/

这里的 --reuseaddr 选项表示在重启服务前，不等之前的连接超时就立即重启。而 --base-path 选项则允许克隆项目时不必给出完整路径。最后面的路径告诉 Git 守护进程允许开放给用户访问的仓库目录。假如有防火墙，则需要为该主机的 9418 端口设置为允许通信。

以守护进程的形式运行该进程的方法有很多，但主要还得看用的是什么操作系统。在 Ubuntu 主机上，可以用 Upstart 脚本达成。编辑该文件：

/etc/event.d/local-git-daemon

加入以下内容：

start on startup

stop on shutdown

exec /usr/bin/git daemon \

--user=git --group=git \

--reuseaddr \

--base-path=/opt/git/ \

/opt/git/

respawn

出于安全考虑，强烈建议用一个对仓库只有读取权限的用户身份来运行该进程 — 只需要简单地新建一个名为 git-ro 的用户（译注：新建用户默认对仓库文件不具备写权限，但这取决于仓库目录的权限设定。务必确认 git-ro 对仓库只能读不能写。），并用它的身份来启动进程。这里为了简化，后面我们还是用之前运行 Gitosis 的用户 'git'。

这样一来，当你重启计算机时，Git 进程也会自动启动。要是进程意外退出或者被杀掉，也会自行重启。在设置完成后，不重启计算机就启动该守护进程，可以运行：

initctl start local-git-daemon

而在其他操作系统上，可以用 xinetd，或者 sysvinit 系统的脚本，或者其他类似的脚本 — 只要能让那个命令变为守护进程并可监控。

接下来，我们必须告诉 Gitosis 哪些仓库允许通过 Git 协议进行匿名只读访问。如果每个仓库都设有各自的段落，可以分别指定是否允许 Git 进程开放给用户匿名读取。比如允许通过 Git 协议访问 iphone\_project，可以把下面两行加到 gitosis.conf 文件的末尾：

[repo iphone\_project]

daemon = yes

在提交和推送完成后，运行中的 Git 守护进程就会响应来自 9418 端口对该项目的访问请求。

如果不考虑 Gitosis，单单起了 Git 守护进程的话，就必须到每一个允许匿名只读访问的仓库目录内，创建一个特殊名称的空文件作为标志：

$ cd /path/to/project.git

$ touch git-daemon-export-ok

该文件的存在，表明允许 Git 守护进程开放对该项目的匿名只读访问。

Gitosis 还能设定哪些项目允许放在 GitWeb 上显示。先打开 GitWeb 的配置文件 /etc/gitweb.conf，添加以下四行：

$projects\_list = "/home/git/gitosis/projects.list";

$projectroot = "/home/git/repositories";

$export\_ok = "git-daemon-export-ok";

@git\_base\_url\_list = ('git://gitserver');

接下来，只要配置各个项目在 Gitosis 中的 gitweb 参数，便能达成是否允许 GitWeb 用户浏览该项目。比如，要让 iphone\_project 项目在 GitWeb 里出现，把 repo 的设定改成下面的样子：

[repo iphone\_project]

daemon = yes

gitweb = yes

在提交并推送过之后，GitWeb 就会自动开始显示 iphone\_project 项目的细节和历史。