Git

一、配置文件存储路径

windows上Git Bash配置文件在：`C:\\'Program Files'\\Git\\etc\\bash.bashrc`

Linux上Git配置：

- 系统配置

  `git config --system --list`查看

  配置文件存储在`/etc/gitconfig`

- 本地配置

  `git config --local --list`查看

  配置文件存储在仓库的`.git/config`

- 全局配置

  `git config --global --list`查看

  配置文件存储在`~/.gitconfig`

二、配置多色彩输出

```
git config --global color.status auto ; \
git config --global color.diff auto ; \
git config --global color.branch auto ; \
git config --global color.interactive auto;
```

三、自定义git log

给`git log`取一个别名`gl`

```
git config --global alias.lg "log --color --graph --
pretty=format:'%Cred%h%Creset -%C(yellow)%d%Creset %s %Cgreen(%cr) %C(bold
blue)<%an>%Creset' --abbrev-commit"
```

四、打tag提交tag

```
git tag                     # 查看tag
git tag v1.0                # 给当前HEAD指向的commint打tag
git push origin v1.0        # 推送单个tag
git push origin --tags      # 推送所有tag
```

五、子模块

克隆时把子模块一并克隆

```
git clone --recursive https://github.com/xboot/xboot.git
```

添加子模式

```
git submodule add <repository_url> <path>
                  仓库地址        下载到本地哪个路径
```

初始化子模块

```
git submodule update --init
```

更新子模块（递归更新子模块的子模块）

```
git submodule update --recursive --remote
```

移除子模块

- 删除`.gitmodules`文件中的相应子模块条目。
- 执行以下命令以从Git配置中删除子模块相关的配置信息：

  ```
  git config -f .git/config --remove-section submodule.<path>
  ```

- 删除子模块的目录：

  ```
  git rm --cached <path>
  rm -rf .git/modules/<path>
  ```

- 提交更改（添加、更新或删除子模块后，需要提交这些更改到父存储库中）

  ```
  git commit -am "Updated submodules"
  ```

六、生成ssh密钥

```
ssh-keygen -t rsa -C "              gmail.com"
```

查看公钥

```
cat ~/.ssh/id_rsa.pub
```

七、git pull设置默认拉取分支

```
git branch --set-upstream-to=origin/master master
```

八、git不用每次输入用户名和密码

```
git config --global credential.helper store
```

九、删除已提交却不想跟踪的文件

```
# 删除文件 -r表示删除文件夹 --cached表示保留本地文件
git rm -r --cached 文件

# 重新添加提交
git add .
git commit -m XXX
git push origin main
```

十、配置代理

```
# 配置VPN 可以顺利拉取github仓库
git config --global http.proxy http://127.0.0.1:7890
git config --global https.proxy https://127.0.0.1:7890

# 取消
git config --global --unset http.proxy
git config --global --unset https.proxy
```

十一、拉取远程非master分支

```
# 同步远程所有改动
git fetch

# 查看远程分支
git branch -r
===============>
  origin/HEAD -> origin/master
  origin/develop
  origin/master

# 切换到要合并的分支
git checkout -b develop

# 拉取远程分支
git pull origin develop
```

十一、克隆仓库

```
# 克隆某个仓库的某个分支
git clone <repository_url> -b <branch>
            仓库地址              分支名
eg: git clone                                  -b dev

# 克隆某个分支的主分支的第几层commit,这样只克隆最新最完整的工程，历史修改记录丢失，体积小下载极快
git clone --depth <depth> <repository_url>
                    深度        仓库地址
eg: git clone --depth 1                                    .git
```

十二、经验

- git创建分支的成本极低，多切换分支，尽量不要丢失任何代码，哪怕你觉得像屎一样的代码。
- 回退代码时：

```
# 1. 先创建一个新的分支
git checkout -b fenzhi

# 2. 此时在新分支上
git log --graph
或使用自定义的 gl

# 3. 切换commit
git reset --hard <commit id>

# 4. 最后又想回到操作之前，直接切回之前分支即可
git checkout fenzhi_prev
```

- 删除分支:

  慎用，一般不要删除分支，分支名字起的合适一些，只要不混乱，不要删除

```
git branch -D <要删除的分支名>
```

- 写了一大堆代码，没提交，发现想重写，又担心这大段代码后面又想用。那就切换分支备份，再回来写。

```
# 将大批未add的代码存入缓存
git stash

# 创建新分支
git checkout -b <新分支名>

# 从缓存中弹出之前没提交的代码
git stash pop

# 可以看下是否弹出了
git status

```

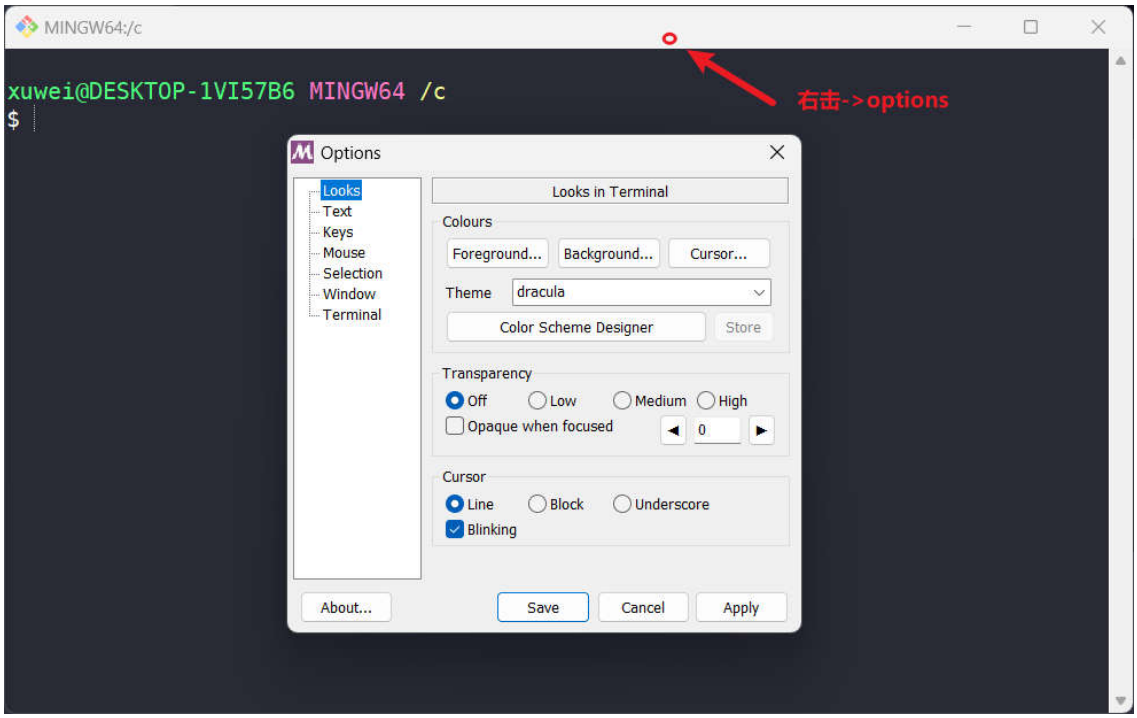
```
# 提交代码
git add .
git commit -m "提交日志：备份一些代码，准备重构。"

# 再切换回之前分支
git checkout <之前分支>
```

附、新装的Git快速配置

windows

1. 配置bash的主题



2. 配置bash

   打开配置文件，在最后增加一些自定义命令

```
vi C:\\'Program Files'\\Git\\etc\\bash.bashrc
==============================================
# some ls error aliases
alias lks='ls'
alias sl='ls'
alias ks='ls'
alias kls='ls'

# some cd .. aliases
alias c='cd ..'
alias cc='cd ../..'
alias ccc='cd ../../..'
alias cccc='cd ../../../..'
alias ccccc='cd ../../../../..'

# some git aliases
alias gs='git status'
```

```
alias gb='git branch'
alias gcb='git checkout -b'
alias gl='git lg'
alias gc='git checkout'
alias gcm='git commit -m'
alias ga='git add .'
```

3. 配置Git，复制下面代码到git bash中执行一下，注意更改最后的用户名和邮箱

```
git config --global color.status auto ; \
git config --global color.diff auto ; \
git config --global color.branch auto ; \
git config --global color.interactive auto; \
git config --global alias.lg "log --color --graph --
pretty=format:'%Cred%h%Creset -%C(yellow)%d%Creset %s %Cgreen(%cr)
%C(bold blue)<%an>%Creset' --abbrev-commit"; \
git config --global credential.helper store; \
git config --global http.proxy http://127.0.0.1:7890; \
git config --global https.proxy https://127.0.0.1:7890; \
git config --global user.name "     ; \
git config --global user.email "          @gmail.com"
```

Linux

2. 配置bash

   打开配置文件，在最后增加一些自定义命令

```
vi ~/.bashrc
==============================================
# some ls error aliases
alias lks='ls'
alias sl='ls'
alias ks='ls'
alias kls='ls'

# some cd .. aliases
alias c='cd ..'
alias cc='cd ../..'
alias ccc='cd ../../..'
alias cccc='cd ../../../..'
alias ccccc='cd ../../../../..'

# some git aliases
alias gs='git status'
alias gb='git branch'
alias gcb='git checkout -b'
alias gl='git lg'
alias gc='git checkout'
alias gcm='git commit -m'
alias ga='git add .'
```

3. 配置Git

```
git config --global color.status auto ; \
git config --global color.diff auto ; \
git config --global color.branch auto ; \
git config --global color.interactive auto; \
git config --global alias.lg "log --color --graph --
pretty=format:'%Cred%h%Creset -%C(yellow)%d%Creset %s %Cgreen(%cr)
%C(bold blue)<%an>%Creset' --abbrev-commit"; \
git config --global credential.helper store; \
git config --global http.proxy http://127.0.0.1:7890; \
git config --global https.proxy https://127.0.0.1:7890; \
git config --global user.name "     "; \
git config --global user.email "           @gmail.com"
```