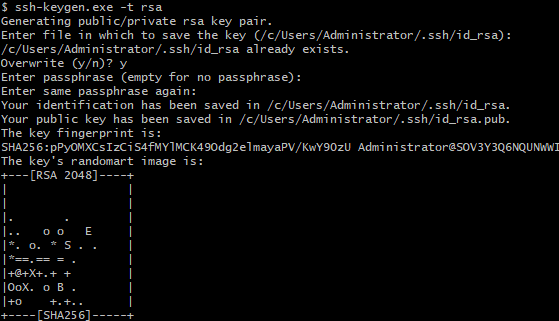
**Git详解**

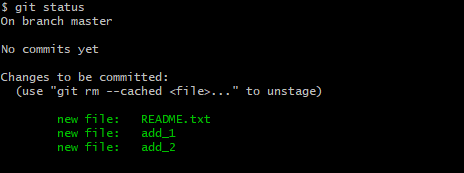
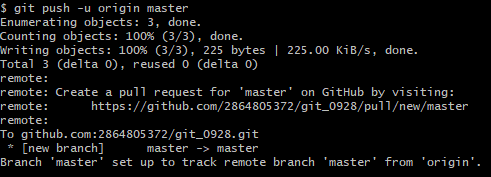
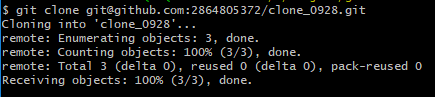
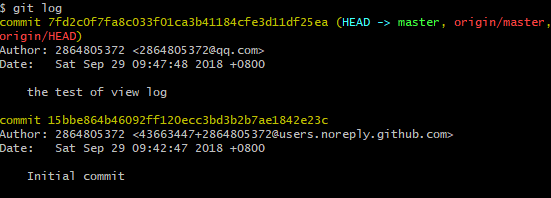
1. 版本控制
   1. 概念：在软件开发过程中，对各种程序代码、配置文件、说明文档等文件的变更进行管理。
   2. 功能
      1. 追踪文件的变更，将文件修改信息(修改人、修改时间、修改内容)记录下来。
      2. 实现并行开发
   3. 特点：通过版本号进行识别
   4. 内容：
      1. 输入输出控制
      2. 分支与合并
      3. 历史记录
   5. 常用工具
      1. 集中式：SVN
      2. 分布式：GIT
2. 集中式与分布式
   1. 集中式(SVN)
      1. 代码存放在中央服务器，需要自己每次从服务器拉取，完成之后再进行推送给中央。
      2. 集中式需要联网
   2. 分布式(git)
      1. 可以在自己机器上clone一个和服务器中完全相同的副本。
      2. 在不能联网的情况下，仍然可以进行提交文件，查看日志，创建分支等操作。
3. Git详解
   1. 下载安装
      1. 下载地址：<https://git-scm.com/downloads>
      2. 安装
         1. Windows ： 直接下载安装包点击运行就可以
         2. Mac：通过homebrew安装
         3. Ubuntu：sudo apt-get install git
      3. 查看安装结果
         1. git version：查看当前git版本号
         2. git help：查看指令的帮助信息
   2. 使用
      1. 配置用户名和邮箱
         1. 用户名：git config --global user.name <username>
         2. 邮箱：git config --global user.email <email>
         3. 查看：git config <要查看的字段>
      2. 仓库
         1. Git中，仓库就是一个文件夹，文件夹中的所有文件都受到版本控制管理，对文件夹中任何一个文件的修改都会被追踪到，还可以在任何时刻去查询历史记录，还原指定版本。
         2. 创建仓库
            1. 新建并进行目录：mkdir git\_dir && cd git\_dir
            2. 创建：git init(在windows中，路径尽量不要包含中文)
            3. 
            4. 通过查看 .git 来查看仓库信息
         3. 配置ssh-key
            1. 生成：ssh-keygen.exe -t rsa
            2. 
            3. 路径：

Windows: C:\Users\Administrator\.ssh

Ubuntu&mac:cd ~/.ssh



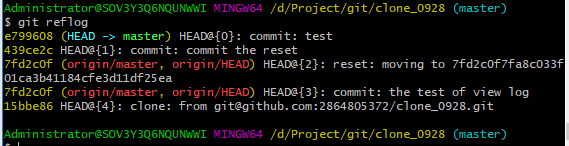
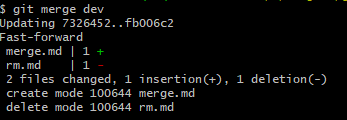
打开公钥文件:id\_rsa.pub(在windows下面，尽量不要用记事本打开，推荐notepad++)

* + - * 1. 把公钥文件的内容存入git的ssh-key中
    1. 文件操作
       1. 添加文件：
          1. 添加单个文件：git add <filename>
          2. 添加所有被修改的文件：git add .
          3. 通过git status可以查看当前仓库的文件状态
          4. 
       2. 提交文件：
          1. git commit -m ‘the des of current commit’
          2. -m 后面输入的是本次提交的说明
       3. 解释一下add和commit
          1. 工作区：电脑中的目录
          2. 暂存区：.git中的index文件
          3. git add 实际上就是把文件修改添加到暂存区中
          4. git commit 就是把暂存区中的内容添加到本地仓库的当前分支中
          5. git commit 只负责提交已经保存到暂存区中的修改
    2. 远程仓库
       1. 远程地址:github.com
       2. 远程仓库地址：
          1. ssh :git@github.com:2864805372/git\_0928.git
          2. https :https://github.com/2864805372/git\_0928.git
       3. 将本地仓库和远程仓库进行关联
          1. git remote add origin <https://github.com/2864805372/git_0928.git>
          2. 查看远程仓库的关联：git remote -v
          3. 
       4. 推送：git push -u origin master
          1. 
       5. 从远程仓库克隆代码等文件
          1. 在远程服务器上新建一个仓库
          2. 克隆到本地：git clone [git@github.com:2864805372/clone\_0928.git](mailto:git@github.com:2864805372/clone_0928.git)
          3. 
    3. 版本管理其它命令
       1. 查看提交日志
          1. git log 显示从第一次到最近一次提交的日志
          2. 
          3. 优化输出:git log --pretty=oneline
          4. commit\_id：通过sha1计算出来的(在多人协作开发的过程中不会产生冲突)
       2. 版本回退
          1. 在暂存区中回退：

回退单个文件：git reset HEAD <filename>

回退所有文件：git reset --hard HEAD^

回退到指定提交：git commit --hard commit\_id

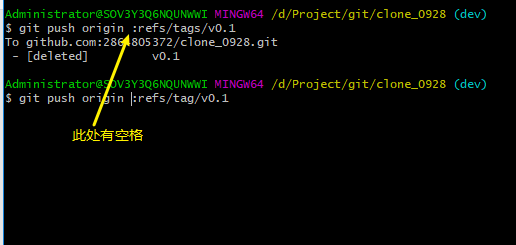
* + - * 1. 在工作区中的回退(撤销修改)：git checkout -- <filename>
        2. 记录git命令：git reflog
        3. 
      1. 删除文件：git rm <filename>
    1. 分支管理
       1. 查看当前所有分支：git branch
       2. 新建分支
          1. git branch <branch\_name>:仅仅是新建一个分支
          2. git checkout -b <branch\_name>:新建分支并且切换到新建的分支上面
       3. git checkout <branch\_name>:切换分支
       4. 分支合并：git merge dev：把分支dev上面的修改合并到当前分支下
       5. 
          1. 也可以先把自己的修改push到远程自己的分支上，再进行merge
       6. 分支删除
          1. git branch -d <branch\_name>
          2. git branch -D <branch\_name>:强制删除
       7. 分支冲突：如果产生同时在不同分支上对相同的文件进行了不同内容的修改，就会产生冲突，一般情况下，通过手动解决冲突内容，再重新提交。
       8. 分支管理原则
          1. 工作中，用到的分支，一般有master,dev,bug,feature
          2. master:保持稳定，只用来发布新的版本，不要直接修改master上面的代码
          3. dev:进行开发工作的分支，在需要发币新版本的时候，再将dev上的修改合并到master
          4. bug:在程序出现bug时，最好是专门新建一个用于修改bug的分支
          5. feature:在程序出现新功能需求时，最好是专门新建一个用于功能实现的分支
       9. 工作现场的保存
          1. git stash：存储当前的工作现场
          2. git stash list: 查看已保存的工作现场
       10. 工作现场恢复
           1. git stash apply:在恢复之后，不会删除stash中的内容
           2. git stash pop: 在恢复之后会删除此次stash的内容
           3. git stash apply stash@{n}可以恢复第n次存储的工作
           4. 
       11. 抓取分支：从服务器上或者其他电脑中获取分支
           1. git clone [git@github.com:2864805372/clone\_0928.git](mailto:git@github.com:2864805372/clone_0928.git)
           2. git pull：可以把最新提交从分支中抓取下来

git pull origin dev

* + 1. 标签(tags)管理
       1. 作用：方便识别，相比commit\_id来说，更加可读性
       2. 创建标签
          1. 仅仅创建一个标签：git tag <tag\_name>
          2. 创建带有说明的标签：git tag -a <tag\_name> -m <”需要说明的内容”>
       3. 查看所有标签：git tag
       4. 查看只定的标签信息：git show <tag\_name>
       5. 删除标签：
          1. 本地：git tag -d <tag\_name>
          2. 远程：

先删除本地标签

从远程删除标签：git push origin :refs/tags/<tag\_name>



* + - 1. 推送标签到远程：
         1. 推送指定的标签：git push origin <tag\_name>
         2. 推送所有标签：git push origin --tags