**Redis**

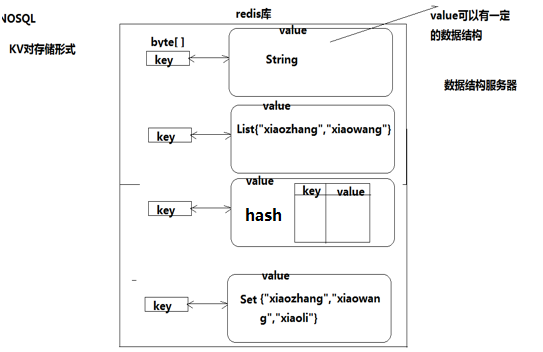
1. 安装
2. wget http://download.redis.io/releases/redis-4.0.9.tar.gz
3. tar -xzvf redis包名，并将解压后的文件夹移动到/usr/local目录下
4. cd /usr/local/redis-4.0.9; make && make install
5. 将redis目录下redis.config复制到/etc/redis下
6. 配置vim /etc/redis/redis.config
7. 绑定ip：需要远程访问，将此行注释或绑定真实ip

* bind 127.0.0.1

1. 端口，默认6379
2. 守护进程：

* daemonize yes
* yes表示以守护进程运行，在命令行不阻塞，类似服务
* no表示以非守护进程运行，当前终端被阻塞

1. 数据文件：dbfilename dump.rdb
2. 数据文件存储路径：dir /var/lib/redis
3. 日志文件：logfile “/var/log/redis/redis-server.log”
4. 数据库：默认16个：database 16
5. 主从复制，类似双机备份：slaveof
6. 客户端命令
7. 启动服务端：redis-server /etc/redis/redis.config
8. **另开命令行**启动客户端：redis-cli
9. 切换数据库：select 10
10. 数据操作
11. redis数据结构



1. 字符串

* 设置set key value
* 定时设置setex key second value
* 批量设置mset key1 value1 key2 value2 …
* 获取get key
* 批量获取mget key1 key2

1. 键命令

* 查找键：keys pattern
* 键是否存在：exists key
* 查看键对应value的类型：type key
* 删除键：del key1 key2
* 键定时：expire key
* 查看键值有效时间：ttl key

1. hash

* 设置：hset key field value(hset user name zhangsan)
* 批量设置：hmset key field1 value1 field2 value2
* 获取键的所有属性（field）：hkeys key
* 获取键的所有属性的值（value）：hvals key
* 获取键的属性值：hget key field
* 获取键的多个属性值：hmget key field1 field2
* 删除属性：hdel key field1 field2

1. list**（列表的值只能存字符串，可将字典转换成字符串存）**

* 列表左侧插入数据：lpush key value1 value2
* 列表右侧插入数据：rpush key value1 value2
* 获取列表值：lrange key start stop
* 删除列表值：lrem key count value
  + count=0：重复数据全删
  + count>0：从左开始删除count个重复数据
  + count<0：从右开始删除count个重复数据

1. set**（无序，无重复，只存字符串，无修改操作）**

* 增加数据：sadd key member1 member2…
* 获取数据：smembers key
* 删除集合：srem key
* 删除特点数据：srem key member

1. zset**（有序，无重复，只存字符串，无修改操作）**

* 增加数据：zadd key score1 member1 score2 member2
* 获取数据：zrange key start stop
* 删除集合：zrem key
* 删除特定数据：zrem key member1 member2

1. 与Python交互
2. 进入Python的虚拟环境，pip install redis
3. 操作redis

* Redis类继承自Object
* 实例化一个redis-cli：redis-cli=Redis(host=”localhost”, port=6379, db=0)
* result=redis-cli.set(‘name’, ‘zhangsan’)**（result是set方法的返回值，True表示创建成功，False表示创建失败）**
* name=redis-cli.get(‘name’)
* redis-cli.delete(‘name’)

**Git**

1. 安装
2. Linux：yum install git
3. windows：官网或者镜像网站下载Git
4. Git简介



1. 工作区：对文件进行增加，修改，删除都在工作区进行
2. 暂存区：对阶段工作进行缓存，属于版本库
3. 仓库区：属于个人开发的小阶段的完成，支持版本回退
4. 服务器：前三个属于本地仓库，可提交远程仓库，也可将远程仓库拉取到本地
5. Git单人本地仓库操作
6. 创建项目文件夹test：

* 进入文件夹中通过git bash执行git init
* 此时test文件夹内出现.git隐藏文件夹，对其中的config文件配置个人信息（通过git bash）：
  + git config user.name xxx
  + git config user.email xxx@xxx

1. 在test目录下新增文件，通过git status命令查看文件状态

* 红色表示文件新建或者修改，都在工作区
  + git checkout filename：**拿暂存区代码（最近版本）替代工作区代码**
  + git add filename：提交工作区代码
* 绿色表示文件通过git add命令被添加到了暂存区

1. 将暂存区代码提交到本地仓库区

* git commit -m ‘备注’：将暂存区代码提交本地仓库
* git rm –cached filename：将提交到暂存区代码退回到工作区

1. **本地仓库区版本管理**

* 查看历史版本：git reflog（**前面一段数字代表版本号**）
* 回退版本（**回退仓库区代码**）：git reset --hard HEAD
  + HEAD：表示最新版本
  + HEAD^：回退一个版本
  + HEAD^^：回退两个版本
  + git reset --hard HEAD~1：回退一个版本
  + git reset --hard HEAD~10：回退十个版本
  + as git reset --hard 版本号：回退到指定版本
* 撤销修改（**对add到暂存区代码撤销到工作区且丢弃工作区代码**，只撤销暂存区与工作区）
  + git reset HEAD filename
  + git checkout filename

1. 远程仓库管理（以github为例）
2. 创建远程仓库
3. 配置SSH
4. clone项目

* git clone 项目git地址
* 配置个人信息
  + git config user.name xxx
  + git config user.email xxx@xxx

1. 本地代码推送到远程仓库：git push
2. 从远程仓库拉取代码：git pull
3. 分支（区分生产环境代码与开发环境代码）：

* 常用公用分支：master，dev
* master：默认分支，用于发布，需要发布时将dev分支整合到master分支
* dev：用于开发阶段性代码，开发完成后整合到master分支

1. 分支操作常用命令：

* git branch：查看当前分支
* git checkout -b dev：创建并切换到dev分支
* git push -u origin dev：设置本地分支跟踪远程分支
* git checkout master：切换到master分支
* git merge dev：master分支整合dev分支**（必须先切换到master分支）**
* **本地整合分支后需要push到远程仓库**

1. 版本回退方法总结
2. 本地仓库版本回退

* git reflog：查看版本号
* git reset --hard 版本号：回退特定版本
* git push：修改完成后push到远程仓库

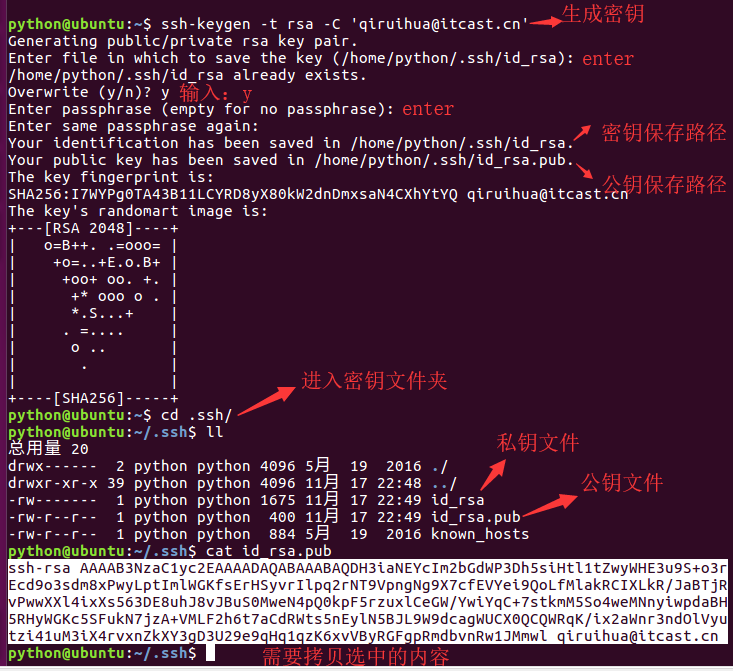
1. 自己的远程分支版本回退

* git pull：从远处仓库拉取最新版本
* 本地仓库回退到指定版本（方法如1所示）
* git push -f：将回退的版本强制push到远程仓库**（必须加-f，否则无法推送）**

1. 公共远程分支版本回退**（远程master分支：A1——A2——B1）**

* A发现A2这次的提交出现了问题，**直接reset回滚了远程master分支到A1**
* 此时B会发现他的**本地master分支并没有回退**，且**比远程master分支超前了两次提交**（远程master分支回退）
* 因此B需要**找回B1**那次提交并将自己的本地master分支重新**同步远程master分支**，然后**将B1重新提交**远程master分支**（远程master分支已回退，无B1）：**
  + git checkout B的dev分支：回到B自己的分支
  + git reflog：**查看最前端和B1的commit id（版本号）**
  + git reset --hard B1版本号：回退到B1
  + git checkout -b 备份分支：**新建一个备份分支，保存B1提交的代码**
  + git checkout B的dev分支：回到B自己的dev分支
  + git reset --hard 最前端版本号：B自己dev分支的本地仓库重新回到最新版本
  + git reset --hard origin/master：同步自己的本地master分支与远程master分支
  + git checkout master：B回到本地master分支
  + git merge 备份分支：本地master分支整合备份分支B1代码**（此时B的本地master分支是A1——B1）**
  + git push：push到远程master分支
  + 对于**A操作后没有丢失提交的队友**来说（例如，C1-D1-A1-A2-B1），**只需**同步自己的本地master分支与远程master分支即可**（git reset origin/master）**
* 公共远程分支版本回退**（git revert HEAD）**
  + HEAD：**表示撤销最近一次提交，回到上上次版本**
  + HEAD~1：表示撤销上上次提交，回到上上上次版本
  + git revert 版本号：撤销特定版本号提交，回退到此版本前一个版本
  + **revert方法虽然是回退代码，但其实版本依然向前**
  + **git revert HEAD也会产生commit id**，因此执行两次revert等于什么也没发生
  + **git revert HEAD~1**：当前面版本中有revert操作也会算在里面
  + 若不是撤销最近一次提交，就**一定会产生代码冲突**，**删除当前所有的代码（被HEAD标记的代码）**，只保留之前版本的代码
  + **revert**方法**不会丢弃其他人的提交**，即便撤销了其他人的提交，他也可以**通过在本地master分支（从撤销后的远程master分支拉取的）reset找回之前提交的代码**，通过**新建分支并整合到最新版本本地master分支上**，最后**push到远程master分支，恢复提交**
* 若是要回退版本太远，建议直接本地回退到此版本之前，并将代码另存，然后删除master所有代码，将新代码整合再重新push到远程master

1. 配置远程仓库ssh（以GitHub为例）
2. 在git bash中输入命令（Linux可直接在终端中输入）：ssh-keygen -t rsa -C “[wangjian\_usr@163.com](mailto:wangjian_usr@163.com)”



1. win10秘钥文件保存在C:\Users\wangjian\.ssh中
2. 将公钥文件中内容拷贝到GitHub的SSH key中

