[Linux笔记](#_Toc515176666)

[Git笔记](#_Toc515176667)

[MongoDB笔记](#_Toc515176668)

[Nginx笔记](#_Toc515176669)

[NPM笔记](#_Toc515176670)

[Gulp笔记](#_Toc515176671)

[Bower笔记](#_Toc515176672)

[Gradle笔记](#_Toc515176673)

[JSON笔记](#_Toc515176674)

# Linux笔记

1. 指令

|  |  |  |
| --- | --- | --- |
| sudo | 以系统管理者的身份执行指令 |  |
| systemctl poweroff | 系统关机 |  |
| systemctl reboot | 重新启动 |  |
| systemctl suspend | 进入暂停模式 |  |
| systemctl hibernate | 进入休眠模式 |  |
| systemctl rescue | 强制进入救援模式 |  |
| systemctl emergency | 强制进入紧急救援模式 |  |
| systemctl set-default graphical.target |  |  |
| ps –l | 仅观察自己的 bash 相关进程： |  |
| ps aux | 观察系统所有进程： |  |
| cd .. | 返回上级目录 |  |
| yum search vim | 查找源中的VIM包 |  |
| yum -y install vim-enhanced | 安装vim-enhanced这个包 | -y，就会自动选择y, 跳过询问你Is this OK[y/d/N] |
| which vim | 已安装VIM的系统中查看VIM属于哪个软件包 |  |
| rpm -q < package name> | 查询一个包是否被安装 |  |
| rpm -qa | sort | 查询系统安装的所有软件包 |  |
| rpm -ql< package name> | 列出软件包安装的文件**和位置** |  |
| rm –rf /home/test | 删除test目录下所有文件 | 包括test目录 |
| rm -rf /home/test/\* | 删除test目录下所有文件 | **保留**test目录  -f：强制删除;-r：递归的删除目录下面的文件以及子目录下文件 |
| rm –rf !(a|b) | 删除除a和b外所有文件，必须在当前目录下执行 | 如果提示“-bash: !: event not found ” shopt -s extglob 来解决 |
| find a/\* [-not] –name b|xargs rm –rf | 可选参数[-not]存在时，删除a文件下所有除b外的所有其他文件 | b可为文件夹名，若b为文件夹时，b底下的文件也会被删除 |
| find /usr/sam/\* -path  /usr/sam/dir1 –prune -o –print | 在/usr/sam目录下查找不在dir1子目录之内的所有文件 |  |
| mkdir 文件夹名称 | 新建文件夹 |  |
| vi文件名称 | 修改此文件 | 按i键进入编辑模式，按esc退出编辑模式；  在浏览模式中：:qw 保存退出，:q直接退出，u撤销上一步的操作 |
| vi 新文件名 | 创建文本文件 |  |

2. 功能提示

* cd .. 返回上级目录

.表示当前目录；..表示上级目录；./aaa/ 表明的就是当前目录下面的aaa目录

* vncserver启动vncserver服务

vncpasswd修改vnc远程登录密码

systemctllist-unit-files | grep vnc查看vnc服务

Systemctlenablevncserver@serivce启用服务

* **Centos 安装vim**
* vim编辑器需安装三个包：

    vim-enhanced-7.0.109-7.el5

   vim-minimal-7.0.109-7.el5

   vim-common-7.0.109-7.el5

* rpm -qa|grep vim，查看本机已经存在的包，确认VIM是否已经安装
* 如果缺少某个，比如说：vim-enhanced，执行：yum -y install vim-enhanced 命令，它会自动下载安装。

-y，就会自动选择y, 跳过询问你Is this OK[y/d/N]

* **vi 和vim 的区别**
* vim对vi的完全兼容,可视为vi的升级版本
* vi有3个模式：插入模式、命令模式、低行模式。

插入模式：在此模式下可以输入字符，按ESC将回到命令模式。

命令模式：可以移动光标、删除字符等。

低行模式：可以保存文件、退出vi、设置vi、查找等功能(低行模式也可以看作是命令模式里的)。

* vim的配置文件 /etc/vimrc

|  |  |
| --- | --- |
| **以下命令在命令模式下使用** | |
| vi filename | 打开filename文件 |
| :w | 保存文件 |
| :w vpser.net | 保存至vpser.net文件 |
| :q | 退出编辑器，如果文件已修改请使用下面的命令 |
| :q! | 退出编辑器，且不保存 |
| :wq | 退出编辑器，且保存文件 |
| **以下命令在命令模式下使用，执行下面命令后将进入插入模式，按ESC键可退出插入模式** | |
| a | 当前光标位置的右边添加文本 |
| i | 在当前光标位置的左边添加文本 |
| A | 在当前行的末尾位置添加文本 |
| I | 在当前行的开始处添加文本(非空字符的行首) |
| O | 在当前行的上面新建一行 |
| o | 在当前行的下面新建一行 |
| R | 替换(覆盖)当前光标位置及后面的若干文本 |
| J | 合并光标所在行及下一行为一行(依然在命令模式) |

3. 常用系统文件

|  |  |  |
| --- | --- | --- |
| /etc/yum.repos.d/ | Yum仓库配置文件夹 | 可在下面添加软件安装源配置文件 |
|  |  |  |

* bash环境下通配符：



* bash环境下特殊符号：



* 管线命令





* 管线命令仅会处理 standard output，对于 standard error output 会予以忽略
* 管线命令必须要能够接受来自前一个指令的数据成为 standard input 继续处理才行。

# Git笔记

1. 指令

|  |  |  |
| --- | --- | --- |
| git config --global http.sslVerify false | 设置全局变量http.sslVerify为false |  |
| git pull | = git fetch + git merge |  |
| git push --force origin | 强制推送 |  |
| git reset --hard 8618818（版本号） | 回退到某一版本 |  |

1. 功能提示



* webstorm删除github上的文件
* 先commit到本地仓库，再push到GitHub上
* 强制覆盖本地文件（本地文件有更改）
* 本地先revert到上一个版本，然后git pull

# MongoDB笔记

下载地址参考：<https://www.mongodb.org/dl/win32/i386>

1. 安装 MongoDB服务
2. 新建数据库根文件夹mkdir c:\data
3. 新建存放数据和日志文件的目录mkdir c:\data\dbmkdir c:\data\log
4. 创建一个配置文件位于 C:\mongodb\mongod.cfg，内容如下

systemLog:

destination: file

path: c:\data\log\mongod.log

storage:

dbPath: c:\data\db

1. 通过执行mongod.exe，使用--install选项来安装服务，使用--config选项来指定之前创建的配置文件。

C:\mongodb\bin\mongod.exe --config "C:\mongodb\mongod.cfg" –install

1. 启动MongoDB服务net start MongoDB

关闭MongoDB服务net stop MongoDB

移除 MongoDB服务C:\mongodb\bin\mongod.exe --remove

1. Centos上安装 MongoDB

1. centos官方yum源，其中是没有mongodb包的，所以要额外增加些配置，编辑Mongodb安装源如下：

* vim /etc/yum.repos.d/mongodb-org-3.6.repo
* 编辑内容：

[mongodb-org-3.6]

name=MongoDB Repository

#baseurl=https://repo.mongodb.org/yum/redhat/$releasever/mongodb-org/3.6/x86\_64/

#阿里云上的镜像

baseurl=http://mirrors.aliyun.com/mongodb/yum/redhat/6/mongodb-org/3.2/x86\_64/

gpgcheck=1

enabled=1

gpgkey=https://www.mongodb.org/static/pgp/server-3.6.asc

2. 安装

yum install -y mongodb-org

3. 启动前修改配置

* （推荐）修改ulimit，改高一点就好了，方法参考其他博客
* （推荐）关闭transparent\_hugepage

vim /etc/rc.local，在最后添加以下代码，重启系统生效

if test -f /sys/kernel/mm/transparent\_hugepage/enabled; then

echo never > /sys/kernel/mm/transparent\_hugepage/enabled

fi

if test -f /sys/kernel/mm/transparent\_hugepage/defrag; then

echo never > /sys/kernel/mm/transparent\_hugepage/defrag

fi

4. 修改mongodb配置文件

* vim /etc/mongod.conf, 修改内容为如下配置:

其中端口被修改为21111（随意），并且允许远程访问，不允许则修改bandIp为：127.0.0.1

已经启用了用户身份验证，并关闭了javascript

systemLog:

destination: file

logAppend: true

path: /var/log/mongodb/mongod.log

storage:

dbPath: /var/lib/mongo

journal:

enabled: true

processManagement:

fork: true

pidFilePath: /var/run/mongodb/mongod.pid

net:

port: 21111

bindIp: 0.0.0.0

security:

authorization: enabled

javascriptEnabled: false

* 保存后重启系统 reboot

5. 修改mongodb配置文件

* 启动，以下方式会自动使用/etc/mongod.conf配置文件

service mongod start

* 新建管理员用户

mongo --port 21111

> use admin

>db.createUser({user:"userAdmin",pwd:"123456",roles:[{role:"userAdminAnyDatabase",db:"admin"}]})

> db.auth("userAdmin","123456")

* 新建普通用户

> use test

> db.createUser({user:"test",pwd:"123456",roles:[{role:"readWrite",db:"test"}]})

* 退出mongo命令行 exit
* 停止mongodb

mongod -f /etc/mongod.conf --shutdown

3. 指令

|  |  |  |
| --- | --- | --- |
| mongod –dbpath数据库存放目录 | 启动数据库 | 在系统Shell下 |
| mongodb://[username:password@]host1[:port1] [/[database][?options]] | 连接数据库 | 在系统Shell下 |
| mongo | 连接数据库进入后台管理Shell | 在系统Shell下 |
| mongodb://[username:password@]host1[:port1] [/[database][?options]] | 连接数据库 | 以下指令在后台管理Shell下 |
| db.auth("name","pwd") | 验证账户，密码 |  |
| use DATABASE\_NAME | DATABASE\_NAME不存在新建数据库；存在则使用此数据库 |  |
| show dbs | 查看所有数据库 |  |
| db.createCollection(name) | 新建集合 |  |
| db.dropDatabase() | 删除数据库 | 需先use切换入要删的数据库 |
| db.collection.drop() | 删除集合 |  |
| db.原集合名.renameCollection(新集合名) | 修改集合名称 |  |
| show collections | 查看所有集合 |  |
| db.auth(‘name’,’pwd’) | 进行用户授权 | 运行此指令必须要先用use xxx切换到响应数据库里；之下的创建用户，需先授权管理员 |
| db.system.users.find() | 查看数据库的用户表和权限 | 需先use admin |
| db.createUser({user:'test',pwd:'test',roles:[{role:'readWrite',db:'testDB'}]}) | 创建新用户 | 需要有管理员权限的授权下 |
| db.dropUser('用户名') | 删除某一用户对某一数据库的权限 | 需先use admin，在db.auth验证后进行 |

3. 功能提示

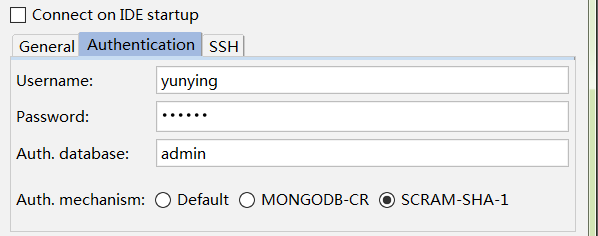
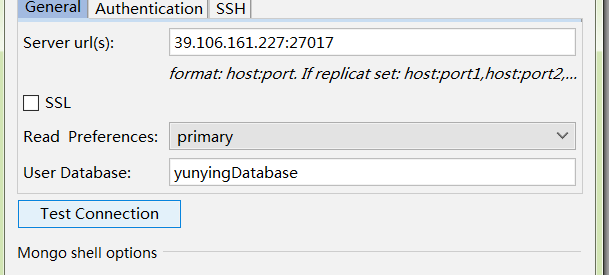
* mongodb 修改用户密码
* 修改密码不能直接更新表数据，这样的话，存到表里的密码是明文的
* 利用db.changeUserPassword

> db.changeUserPassword('tank2','test');

* Intellij使用mongdb插件连接远程服务器时General填写要连接的实际数据库

Authentication必须选择SCRAM-SHA-1，Auth.database填写mongDB的用户库

另外，远程服务器的响应端口必须要打开



* mongo shell里所有的db.auth命令必须要在use admin后，在这个用户库里进行（因为所有的用户数据都保存在这个库里），完成验证后，即可转到相应的其他库进行工作
* mongo 设置权限的一些示例
* 授予角色：db.grantRolesToUser( "userName" , [ { role: "<role>", db: "<database>" } ])
* 取消角色：db.grantRolesToUser( "userName" , [ { role: "<role>", db: "<database>" } ])
* 创建角色db.createRole
* 删除角色db.dropRole("testRole")
* 更新角色

db.updateRole("testRole",{ roles:[{ role: "readWrite",db: "mydb"}]},{ w:"majority" })

* 添加Privileges给角色

db.grantPrivilegesToRole ("testRole",

[{ resource: { db: "mydb", collection: "" },actions: [ "update", "insert", "remove" ]}

])

|  |  |  |
| --- | --- | --- |
| **角色分类** | **角色** | **权限及角色**  **（本文大小写可能有些变化，使用时请参考官方文档）** |
| Database User Roles | read | CollStats,dbHash,dbStats,find,killCursors,listIndexes,listCollections |
| readWrite | CollStats,ConvertToCapped,CreateCollection,DbHash,DbStats,  DropCollection,CreateIndex,DropIndex,Emptycapped,Find,  Insert,KillCursors,ListIndexes,ListCollections,Remove,  RenameCollectionSameDB,update |
| Database Administration Roles | dbAdmin | collStats,dbHash,dbStats,find,killCursors,listIndexes,listCollections,  dropCollection 和 createCollection 在 system.profile |
| dbOwner | 角色：readWrite, dbAdmin,userAdmin |
| userAdmin | ChangeCustomData,ChangePassword,CreateRole,CreateUser,  DropRole,DropUser,GrantRole,RevokeRole,ViewRole,viewUser |
| Cluster Administration Roles | clusterAdmin | 角色：clusterManager, clusterMonitor, hostManager |
| clusterManager | AddShard,ApplicationMessage,CleanupOrphaned,FlushRouterConfig,  ListShards,RemoveShard,ReplSetConfigure,ReplSetGetStatus,  ReplSetStateChange,Resync,    EnableSharding,MoveChunk,SplitChunk,splitVector |
| clusterMonitor | connPoolStats,cursorInfo,getCmdLineOpts,getLog,getParameter,  getShardMap,hostInfo,inprog,listDatabases,listShards,netstat,  replSetGetStatus,serverStatus,shardingState,top  collStats,dbStats,getShardVersion |
| hostManager | applicationMessage,closeAllDatabases,connPoolSync,cpuProfiler,  diagLogging,flushRouterConfig,fsync,invalidateUserCache,killop,  logRotate,resync,setParameter,shutdown,touch,unlock |
| Backup and Restoration Roles | backup | 提供在admin数据库mms.backup文档中insert,update权限  列出所有数据库：listDatabases  列出所有集合索引：listIndexes    对以下提供查询操作：find  \*非系统集合  \*系统集合：system.indexes, system.namespaces, system.js  \*集合：admin.system.users 和 admin.system.roles |
| restore | 非系统集合、system.js，admin.system.users 和 admin.system.roles 及2.6 版本的system.users提供以下权限：  collMod,createCollection,createIndex,dropCollection,insert    列出所有数据库：listDatabases  system.users ：find,remove,update |
| All-Database Roles | readAnyDatabase | 提供所有数据库中只读权限：read  列出集群所有数据库：listDatabases |
| readWriteAnyDatabase | 提供所有数据库读写权限：readWrite  列出集群所有数据库：listDatabases |
| userAdminAnyDatabase | 提供所有用户数据管理权限：userAdmin  Cluster：authSchemaUpgrade,invalidateUserCache,listDatabases  admin.system.users和admin.system.roles：  collStats,dbHash,dbStats,find,killCursors,planCacheRead  createIndex,dropIndex |
| dbAdminAnyDatabase | 提供所有数据库管理员权限：dbAdmin  列出集群所有数据库：listDatabases |
| Superuser Roles | root | 角色：dbOwner，userAdmin，userAdminAnyDatabase  readWriteAnyDatabase, dbAdminAnyDatabase,  userAdminAnyDatabase，clusterAdmin |
| Internal Role | \_\_system | 集群中对任何数据库采取任何操作 |

* Mongo shell

当一行以{或[或(结束时，可进入多行模式；连续回车可退出多行模式

# Nginx笔记

1. CentOS 7上安装Nginx

1.添加CentOS 7 EPEL仓库

sudo yum install epel-release

2.安装Nginx

sudo yum install nginx

3.启动Nginx

sudo systemctl start nginx

4.如果您正在运行防火墙，请运行以下命令以允许HTTP和HTTPS通信：

sudo firewall-cmd --permanent --zone=public --add-service=http

sudo firewall-cmd --permanent --zone=public --add-service=https

sudo firewall-cmd --reload

5.如果想在系统启动时启用Nginx。请输入以下命令：

sudo systemctl enable nginx

2. 指令

|  |  |  |
| --- | --- | --- |
| nginx –s reload | 重启命令 |  |
| nginx –t | 检查配置文件是否正确 |  |

3.功能提示

* nginx 更改配置文件后需要重启，重启命令：nginx –s reload

# NPM笔记

1. 指令

|  |  |  |
| --- | --- | --- |
| npm config get prefix | 查看npm全局仓库路径 |  |
| npm config set prefix | 设置npm全局仓库路径 |  |
| npm view jquery versions | 查看jquery模块全部版本 |  |
| npm install gulp-sourcemaps --save | 安装gulp-sourcemaps模块并写入package.json |  |

2.功能提示

* package.json版本(^和~区别)
* 波浪符号（~）：他会更新到当前minor version（也就是中间的那位数字）中最新的版本
* 插入符号（^）：将会把当前库的版本更新到当前major version（也就是第一位数字）中最新的版本

# Bower笔记

1. 指令

|  |  |  |
| --- | --- | --- |
| bower list | 列出当前安装的版本和最新版本，以及各个安装包的依赖 |  |
| bower info jquery | 查询jQuery有哪些版本 |  |
| bower install bootstrap --save | 安装bootstrap并更新至bower.json中的Dependencies |  |
| bower install bootstrap –save-dev | 安装bootstrap并更新至bower.json中的devDependencies |  |
| bower install --force | 强制安装 |  |

2.功能提示

* bower.json文件解释

{

"name":"", //必须，如果需要注册包，则该包名唯一。

"description":"", //可选，包描述

"main":[], //可选，入口文件，bower本身不使用，供第三方构建工具会使用,每种文件类型只能有一个。

"ignore":[], //可选，文件或目录列表。bower安装的时候将忽略该列表中的文件,bower是从git仓库或压缩包下载一个包，里面的文件并不一定全部需要。

"dependencies":[], //依赖包，name:value，value可以是包的semverrange(版本号范围)，也可以直接是一个包的git地址或压缩包地址。

"devDependencies":[], //开发依赖包，仅仅在开发过程中测试或者编译文档用，部署生产环境是不需要。

"resolutions":[], //包引用冲突自动使用该模块指定的包版本,格式和dependencies 相同

"overrides" :[//可以覆盖一个包中的默认设置，比如main里面设定的入口文件

"package-name":{ //这样可以根据需要，让第三方工具只打包需要的文件。

"main":[]

}

],

"moduleType":"", //可选，指定包采用那种模块化方式(globals,amd,node,es6,yui)

"private":Boolean, //是否公开发布当前包,如果只是使用bower来管理项目的包，设置为true.

"license":"", //授权方式(GPL-3.0,CC-BY-4.0.....)

"keywords":[], //可选，方便注册后容易被其他人搜索到。

"authors":[], //作者列表

"homepage":[], //主页，包介绍页

"repository":{ //包所在仓库。

"type": "git",

"url": "git://github.com/foo/bar.git"

},

｝

# Gulp笔记

1. 语法

* gulp.src(globs[, options])
* gulp.src (['./src/\*\*', '**!**\*\*/node\_modules/\*\*'])**→**感叹号表示不包括第二个参数代表的内容
* gulp.src ('./src/\*\*/\*.js', {base:'./src/'})**→**base作用：'./src/\*\*/\*.js'路径去除'./src/'得到的路径加到gulp.dest里的路径后面得出最终的目的路径

# Gradle笔记

1. 目录文件解释

|  |  |
| --- | --- |
| **目录文件** | **作用** |
| .gradle | gradle项目产生文件（自动编译工具产生的文件） |
| .idea | IDEA项目文件（开发工具产生的文件） |
| app | 其中一个module，复用父项目的设置，可与父项目拥有相同的配置文件  （里面也能包含build.gradle、gradle.properties、setting.gradle 等相关gradle文件，其实每一层都是一个module，整个项目是一个大的 module 而已） |
| build | 自动构建时生成文件的地方 |
| gradle | 自动完成gradle环境支持文件夹 |
| .gitignore | git源码管理文件 |
| build.gradle | gradle 项目自动编译的配置文件 |
| gradle.properties | gradle 运行环境配置文件  比如配置gradle运行模式，运行时jvm虚拟机的大小 |
| gradlew | 自动完成 gradle 环境的linux mac 脚本，配合gradle 文件夹使用 |
| gradlew.bat | 自动完成 gradle 环境的windows 脚本，配合gradle 文件夹使用 |
| local.properties | Android SDK NDK 环境路径配置 |
| \*.iml | IDEA 项目文件 |
| setting.gradle | gradle 项目的子项目包含文件 |

2. 提示

* 在项目构建时选择use default gradle warpper
* IDEA会调用项目根目录 gradlew 或者 gradlew.bat （根据linux，windows，osx自动选型）代替原生的 gradle 方法做自动构建
* 解析 gradle/wrapper/gradle-wrapper.properties 文件，获取项目需要的gradle版本下载地址
* 判断本地用户目录下的 ./gradle 目录下是否存在该版本，不存在该版本，下载，存在跳过下载
* 下载 gradle-wrapper.properties 指定版本，并解压到用户目录的下 ./gradle 文件下
* 利用 ./gradle 目录下对应的版本的 gradle 进行相应自动编译操作
* gradle-wrapper.
* gradle-wrapper.jar是Gradle Wrapper的主体功能包。在Intellij安装过程中产生gradle-wrapper.jar，然后每次新建项目，会将gradle-wrapper.jar拷贝到你的项目的gradle/wrapper目录中
* gradle-wrapper.properties文件主要指定了该项目需要什么版本的Gradle，从哪里下载该版本的Gradle；此文件中的GRADLE\_USER\_HOME一般指~/.gradle
* Gradle对应版本下载完成之后，Gradle Wrapper的使命基本完成了

# JSON笔记

1.注意事项

* 基本规则
* JSON官网规定，不管是键或值都用**双引号**引起来
* json文件里的换行符\r\n
* json文件里不允许加注释
* 要描述的一个对象, (里面是这个对象的属性/函数!!) 那么用大括号, 表示对象; 如果你要描述的东西是多个并列的值, 那么用数组用中括号的方式!!!