# Linux

## 查看端口占用情况

netstat -lnp | grep 4141

lsof -i tcp:4141

## 设置开机启动

chkconfig mysqld on

## yum

yum install psmisc

yum install net-tools

## RMP包安装

rpm -ivh mysql-community-release-el7-5.noarch.rpm

## 修改linux swap空间的swappiness，降低对硬盘的缓存

linux 会使用硬盘的一部分做为SWAP分区，用来进行进程调度--进程是正在运行的程序--把当前不用的进程调成‘等待（standby）‘，甚至‘睡眠 （sleep）’，一旦要用，再调成‘活动（active）’，睡眠的进程就躺到SWAP分区睡大觉，把内存空出来让给‘活动’的进程。 　　如果内存够大，应当告诉 linux 不必太多的使用 SWAP 分区， 可以通过修改 swappiness 的数值。swappiness=0的时候表示最大限度使用物理内存，然后才是 swap空间，swappiness＝100的时候表示积极的使用swap分区，并且把内存上的数据及时的搬运到swap空间里面。 　　在ubuntu 里面，默认设置swappiness这个值等于60。 　　 　　!!!! 如果内存较小，而进程调度频繁，硬盘的响动就会大了 !!!! 　　 　　现在一般1个G的内存可修改为10， 2个G的可改为5， 甚至是0。具体这样做： 　　1.查看你的系统里面的swappiness 　　$ cat /proc/sys/vm/swappiness 　　不出意外的话，你应该看到是 60 　　2.修改swappiness值为10 　　$ sudo sysctl vm.swappiness=10 　　但是这只是临时性的修改，在你重启系统后会恢复默认的60，为长治久安，还要更进一步： 　　$ sudo gedit /etc/sysctl.conf 　　在这个文档的最后加上这样一行: 　　vm.swappiness=10 　　然后保存，重启。ok，你的设置就生效了。

## shell bash判断文件或文件夹是否存在

#shell判断文件夹是否存在

#如果文件夹不存在，创建文件夹

if [ ! -d "/myfolder" ]; then

mkdir /myfolder

fi

#shell判断文件,目录是否存在或者具有权限

folder="/var/www/"

file="/var/www/log"

# -x 参数判断 $folder 是否存在并且是否具有可执行权限

if [ ! -x "$folder"]; then

mkdir "$folder"

fi

# -d 参数判断 $folder 是否存在

if [ ! -d "$folder"]; then

mkdir "$folder"

fi

# -f 参数判断 $file 是否存在

if [ ! -f "$file" ]; then

touch "$file"

fi

# -n 判断一个变量是否有值

if [ ! -n "$var" ]; then

echo "$var is empty"

exit 0

fi

# 判断两个变量是否相等

if [ "$var1" = "$var2" ]; then

echo '$var1 eq $var2'

else

echo '$var1 not eq $var2'

fi

## [Linux shell 之 提取文件名和目录名的一些方法](http://blog.csdn.net/ljianhui/article/details/43128465)

很多时候在使用Linux的shell时，我们都需要对文件名或目录名进行处理，通常的操作是由路径中提取出文件名，从路径中提取出目录名，提取文件后缀名等等。例如，从路径/dir1/dir2/file.txt中提取也文件名file.txt，提取出目录/dir1/dir2，提取出文件后缀txt等。

下面介绍两种常用的方法来进行相关的操作。

一、使用${}

1、${var##\*/}

该命令的作用是去掉变量var从左边算起的最后一个'/'字符及其左边的内容，返回从左边算起的最后一个'/'（不含该字符）的右边的内容。使用例子及结果如下：



从运行结果可以看到，使用该命令，可以提取出我们需要的文件名file.txt。

若使用时在shell程序文件中，可以使用变量来保存这个结果，再加以利用，如file=${var##\*/}

2、${var##\*.}

该命令的作用是去掉变量var从左边算起的最后一个'.'字符及其左边的内容，返回从左边算起的最后一个'.'（不含该字符）的右边的内容。使用例子及结果如下：



从运行结果可以看到，使用该命令，可以提取出我们需要的文件后缀。

如果文件的后缀不仅有一个，例如，file.tar.gz，命令${var##\*.}仅能提取最后一个后缀，而我想提取tar.gz时该怎么办？那么就要用下面所说的${var#\*.}命令了。

3、${var#\*.}

该命令的作用是去掉变量var从左边算起的第一个'.'字符及其左边的内容，返回从左边算起第一个'.'（不含该字符）的右边部分的内容。使用例子及结果如下：



从运行结果可以看到，使用该命令，可以提取出文件的多个后缀。

4、${var%/\*}

该命令的使用是去掉变量var从右边算起的第一个'/'字符及其右边的内容，返回从右边算起的第一个'/'（不含该字符）的左边的内容。使用例子及结果如下：



从运行的结果可以看到，使用该命令，可以提取出我们需要的文件所在的目录

5、${var%%.\*}

该命令的使用是去掉变量var从右边算起的最后一个'.'字符及其右边的内容，返回从右边算起的最后一个'.'（不含该字符）的左边的内容。使用例子及结果如下：



当我们需要建立一个与文件名相同名字（没有后缀）的目录与对应的文件相对应时，就可以使用该命令来进行操作。例如，解压文件的情况就与此类似，我们压缩文件file.zip时，会在与file.zip同级目录下建立一个名为file的目录。

6、${}总结

其实${}并不是专门为提取文件名或目录名的，它的使用是变量的提取和替换等等操作，它可以提取非常多的内容，并不一定是上面五个例子中的'/'或'.'。也就是说，上面的使用方法只是它使用的一个特例。

看到上面的这些命令，可能会让人感到非常难以理解和记忆，其实不然，它们都是有规律的。

#：表示从左边算起第一个

%：表示从右边算起第一个

##：表示从左边算起最后一个

%%：表示从右边算起最后一个

换句话来说，＃总是表示左边算起，％总是表示右边算起。

＊：表示要删除的内容，对于#和##的情况，它位于指定的字符（例子中的'/'和'.'）的左边，表于删除指定字符及其左边的内容；对于%和%%的情况，它位于指定的字符（例子中的'/'和'.'）的右边，表示删除指定字符及其右边的内容。这里的'\*'的位置不能互换，即不能把\*号放在#或##的右边，反之亦然。

例如：${var%%x\*}表示找出从右边算起最后一个字符x，并删除字符x及其右边的字符。

看到这里，就可以知道，其实该命令的用途非常广泛，上面只是指针文件名和目录名的命名特性来进行提取的一些特例而已。

二、basename和dirname

${}并不是专门为提取文件名和目录名设计的命令，那么basename和dirname命令就是专门为做这一件事而已准备的了。

1、basename

该命令的作用是从路径中提取出文件名，使用方法为basename NAME [SUFFIX]。

1）从路径中提出出文件名（带后缀），例子如下：



2）从上面命令的用法中可以看到，后缀（SUFFIX）是一个可选项。所以，若只想提取出文件名file，而不带有后缀，还可以在变量的后面加上后缀名，例子如下：



2、dirname

该命令的作用是从路径中提取出目录名，使用方法为 dirname NAME

使用例子如下：



这样就提取出了file.txt文件所在的目录。

注：该命令不仅能提取出普通文件所的目录，它能提取出任何文件所在的目录，例如目录所在的目录，如下：



它提取出了目录dir2所在的目录dir1.

## 时间函数

a=`date "+%Y-%m-%d"`

b=`date -d "$a - 10 days"` #前10天的日期

得到上一个小时

a=`date "+%H"`

b=`date -d "$a-1 hour" "+%H"`

b=`date -d "$a-65 minutes" "+%H"`

cur\_time=`date "+%s"` #时间戳

### pre\_hour=`date -d "- 65 minutes" "+%H"`

### linux在shell中获取时间

<http://www.cnblogs.com/mfryf/archive/2012/03/23/2413362.html>

## Firewall

检查防火墙状态

firewall-cmd --stat

临时开放ftp服务

firewall-cmd --add-service=ftp

永久开放ftp服务

firewall-cmd --add-service=ftp --permanent

关闭ftp服务

firewall-cmd --remove-service=ftp --permanent

配置防火墙在public区域永久开放http服务

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

加入指定开放端口

firewall-cmd --add-port=1324/tcp

为了让之前的设定生效当然要重启服务咯

systemctl restart firewalld

或者使用下面的命令免去重启服务(防火墙策略配置后重新载入)

firewall-cmd --complete-reload

firewall-cmd --reload   (这两句功能相同)

检查ftp服务的21端口是否开放

iptables -L -n | grep 21 ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0 tcp dpt:21 ctstate NEW

查询ftp服务启用状态

firewall-cmd --query-service ftp

查看当前规则

firewall-cmd --list-all

仅允许部分IP访问本机服务配置

firewall-cmd --permanent --zone=public --add-rich-rule="rule family="ipv4" source address="192.168.0.4/24" service name="http" accept"

仅允许部分IP访问本机端口配置

firewall-cmd --permanent --zone=public --add-rich-rule="rule family="ipv4" source address="192.168.0.4/24" port protocol="tcp" port="8080" accept"

firewall-cmd --zone=public --add-port=80/tcp –permanent

## Linux NFS服务器的安装与配置

<http://www.cnblogs.com/mchina/archive/2013/01/03/2840040.html>

## [Linux下如何查看分区文件系统类型](http://freearth.blog.chinaunix.net/uid-20769015-id-3478861.html)

<http://freearth.blog.chinaunix.net/uid-20769015-id-3478861.html>

mount 也可查看到

# JAVA

java 库介绍

==========================Guava===================

Guava工程包含了若干被Google的 Java项目广泛依赖 的核心库，例如：集合 [collections] 、缓存 [caching] 、原生类型支持 [primitives support] 、并发库 [concurrency libraries] 、通用注解 [common annotations] 、字符串处理 [string processing] 、I/O 等等。 所有这些工具每天都在被Google的工程师应用在产品服务中。

==========================Jackson==================

Jackson可以轻松的将Java对象转换成json对象和xml文档，同样也可以将json、xml转换成Java对象。

前面有介绍过json-lib这个框架，在线博文：http://www.cnblogs.com/hoojo/archive/2011/04/21/2023805.html

相比json-lib框架，Jackson所依赖的jar包较少，简单易用并且性能也要相对高些。而且Jackson社区相对比较活跃，更新速度也比较快。

Gson

解析和生成jsxon工具类

snappy

Snappy 是一个 C++ 的用来压缩和解压缩的开发包。其目标不是最大限度压缩或者兼容其他压缩格式，而是旨在提供高速压缩速度和合理的压缩率。Snappy 比 zlib 更快，但文件相对要大 20% 到 100%。在 64位模式的 Core i7 处理器上，可达每秒 250~500兆的压缩速度。

Curator

Curator是Netflix开源的一套ZooKeeper客户端框架. Netflix在使用ZooKeeper的过程中发现ZooKeeper自带的客户端太底层, 应用方在使用的时候需要自己处理很多事情, 于是在它的基础上包装了一下, 提供了一套更好用的客户端框架.

xmlenc

The xmlenc library is a fast stream-based XML output library for Java. Main design goals are performance, simplicitity and pureness. As far as known, xmlenc is the fastest XML output library for Java.

Paranamer

Paranamer是一个Java开源类库能够在运行期读取Java类中非私有方法与构造函数中包含的参数名。

Jersey

Jersey是JAX-RS（JSR311）开源参考实现用于构建RESTful Web service，它包含三个部分：

　　核心服务器（Core Server） 通过提供JSR 311中标准化的注释和API标准化，可以用直观的方式开发RESTful Web服务。

　　核心客户端（Core Client） Jersey客户端API能够帮助开发者与RESTful服务轻松通信；

　　集成（Integration） Jersey还提供可以轻松继承Spring、Guice、Apache Abdera的库

<http://www.cnblogs.com/xinsheng/p/3897289.html>

asn1

该工具可根据ASN.1协议描述文件,生成对应的java类

JAXB

JAXB（Java API for XML Binding），提供了一个快速便捷的方式将Java对象与XML进行转换。在JAX-WS（Java的WebService规范之一）中，JDK1.6 自带的版本JAX-WS2.1，其底层支持就是JAXB。

commons-math3

Section 1 linear 线性代数（矩阵为主）

1） Vector 向量

2） Matrix 矩阵

3） Matrix Decomposition 矩阵分解

Section 2 analysis 数学分析（函数为主）

1） Function 函数

2） Polynomial 多项式函数

3） Interpolation 插值

4） Integration 积分

5） Solver 求解

Section 3 Probabilityand Statistics 概率和统计

1）distribution 分布

2）fraction and complex 分数和复数

3）random and statistics 随机生成和统计初步

4）cluster and regression聚类和回归

Mockito：

一个强大的用于 Java 开发的模拟测试框架

<http://www.oschina.net/translate/mockito-a-great-mock-framework-for-java-development>

Xalan

xalan是一套xslt处理器（有C和JAVA语言两种版本），用来将XML文件转换为HTML,TEXT和XML等其他类型文件格式。支持XSLT1.0和XPATH 1.0版。开发人员可以通过命令行方式或在JAVA APPLET和SERVLET中使用，并可以作为自己开发的应用程序的类库使用。xalan-java实现的是transformation API for XML(TRaX)接口，此接口为jaxp1.2标准中的一部分。

spark 编译

build/mvn -T 8 -Pyarn -Phadoop-2.6 -Dhadoop.version=2.6.2 -DskipTests clean package

./make-distribution.sh —name spark1.5.2-hadoop-2.7-bin —tgz -Phadoop-2.7 –Pyarn

g++ --std=c++11 -o client \*.cpp –lthrift

### memcache

https://github.com/gwhalin/Memcached-Java-Client/downloads

### Zero Copy

http://www.ibm.com/developerworks/linux/library/j-zerocopy

### Spring

<http://docs.spring.io/spring/docs/current/spring-framework-reference/htmlsingle/>

http://docs.spring.io/spring-framework/

#### spring3mvc框架开发中resin和tomcat默认servelt配置处理静态资源

<http://blog.csdn.net/5iasp/article/details/8017918>

#### Spring Web Flow 2.0 入门详解

http://docs.spring.io/spring-webflow/docs/2.3.x/reference/htmlsingle/

http://www.cnblogs.com/xwdreamer/archive/2011/11/10/2296939.html

### API

<http://www.java2s.com/Tutorials/Java_Lambda/java.util.function/Function/Function_andThen_example.htm>

<http://docs.oracle.com/javase/8/docs/api/>

<https://segmentfault.com/a/1190000000664023>

IBM

<http://www.ibm.com/developerworks/cn/java/>

### concurrent 包使用介绍

<http://my.oschina.net/u/1185331/blog/502350>

线程池

<http://zy116494718.iteye.com/blog/1704344>

<http://wenku.baidu.com/link?url=uN_WeM4pMmha6kVytYsxRiQpjPvQcUqcxrDnjWK3DccTE6a89Bxl3TMF2jUhUau5eygb3ywZYYUOFGENzYdosYbu9PBCqrz-1ya7YzKgDkW>

### java 分析工具

<http://www.cnblogs.com/nexiyi/p/java_thread_jstack.html>

强引用

<http://www.cnblogs.com/mengdd/archive/2013/09/03/3298852.html>

### jvm调优

<http://www.360doc.com/content/12/0113/08/1073512_179088229.shtml>

<http://www.cnblogs.com/dartagnan/archive/2011/09/25/2190507.html>

### 详细分析Java中断机制

<http://www.infoq.com/cn/articles/java-interrupt-mechanism>

### Java Platform, Standard Edition Tools Reference

<http://docs.oracle.com/javase/8/docs/technotes/tools/unix/index.html>

### 将 java 做成服务

http://wrapper.tanukisoftware.com/doc/english/download.jsp

# MAC

chflags的详解

用法：chflags [参数] 关键词 档案

常用关键词：

schg 档案完全无法被修改（super-user only）。

sappend 档案內容只能用附加（append）的方式修改（super-user only）。

sunlink 档案无法被刪除或者重命名（super-user only）。

uchg 档案完全无法被修改（super-user only 或者一般使用者）。

uappend 档案內容只能用附加（append）的方式修改（super-user only 或者一般使用者）。

uunlink 档案无法被刪除或者重命名（super-user only 或者一般使用者）。

uchg 用于文件时，owner 不能修改文件，也不能删除文件或者对文件重命名；uappend 用于文件时，owner 只能以 append 的方式修改文件，不能删除文件或者对文件重命名；uunlink 用于文件时，owner 可以对文件进行修改，但不能删除文件或者对文件重命名。

uchg 用于目录时，owner 不能删除目录或者重命名目录，可以修改目录中已有的文件，但不能在目录中添加文件，也不能删除目录中的文件或者对其重命名；uappend 用于目录时，owner 不能删除目录或者重命名目录，可以修改目录中已有的文件和在目录中添加文件，不能删除目录中的文件或者对其重命名；uunlink 用于目录时，owner 不能删除目录或者重命名目录，可以修改目录中已有的文件和在目录中添加文件，也可以删除目录中的文件或者对其重命名。

结论：uchg>uappend>uunlink。

示例：要在文件 file1 上应用系统禁删标志， 应使用下述命令：

# chflags sunlink file1

　　要禁用系统禁删标志， 只需在前述命令中的

sunlink

标志前加 “no”。 例如：

# chflags nosunlink file1

　　要显示文件上的标志， 应使用命令 [ls(1)](http://www.freebsd.org/cgi/man.cgi?query=ls&sektion=1) 的

-lo

参数：

# ls -lo file1

　　输出结果应类似于：

-rw-r--r-- 1 trhodes trhodes sunlnk 0 Mar 1 05:54 file1

# MAVEN

eclipse导出Maven依赖的Jar

1、 从Maven项目中导出项目依赖的jar包：进入工程pom.xml 所在的目录下，执行如下命令：

|  |  |
| --- | --- |
|  | mvn dependency:copy-dependencies |

2、从Maven仓库中导出jar包：进入工程pom.xml 所在的目录下，

输入以下命令：mvn dependency:copy-dependencies -DoutputDirectory=lib

更简单的  mvn dependency:copy-dependencies    会导出到target\dependency 下面

mvn clean dependency:copy-dependencies package  未试过 复制依赖的jar 非常有用

mvn dependency:copy-dependencies -DoutputDirectory=lib   -DincludeScope=compile 这样jar包都会copy到工程目录下的lib里面  mvn package 然后从被打包好的文件里找出jar来。

3、maven打包的时候有些jar包是不在mavencenter的。需要在pom中引入相关的本地jar包，那么相关的depency应该像如下更改

http://my.oschina.net/zimingforever/blog/266191

<dependency>

<groupId>org.apache.flueme</groupId>

<artifactId>flume-tools</artifactId>

<version>1.6.0</version>

<scope>system</scope>

<systemPath>${project.basedir}/libs/flume-tools-1.6.0.jar</systemPath>

</dependency>

<http://mvnrepository.com>

<http://mvnrepository.com/open-source/core-utilities>

<http://maven.apache.org/shared/maven-archiver/examples/classpath.html>

# 大数据

<http://www.cloudera.com/content/www/en-us/documentation.html>

<http://archive.cloudera.com/cdh5/cdh/5/>

http://archive.cloudera.com/cm5/cm

http://www.cloudera.com/documentation/enterprise/latest/topics/introduction.html

## Hadoop

### webHDFS设置和使用说明

<http://blog.csdn.net/iloveyin/article/details/28264027>

curl -i <http://192.168.21.34:50070/webhdfs/v1/user/hive?op=LISTSTATUS>

curl -i -L <http://192.168.21.34:50070/webhdfs/v1/user/flume/Games/tzsg/pay/2016-01-16/01/event_2016-01-16_01.1453694419464.log?op=OPEN>

### hadoop 权威指南

http://shop.oreilly.com/product/0636920021773.do

### HDFS

### 利用JavaAPI访问HDFS的文件

<http://blog.csdn.net/zhangzhaokun/article/details/5597433>

<http://www.cnblogs.com/xuqiang/archive/2011/06/03/2042526.html>

## NOSQL

<http://baike.baidu.com/link?url=O3pT6n1QXrbKc9RgbA2TVGjNcQQEWdJwizE4Aa0LDvC0hYj3CYJ8CrgVK4xlKBFdSebrSxWntWmDSpuFQxql6K>

cdh5

<http://archive.cloudera.com/cdh5/cdh/5/>

<http://www.cloudera.com/content/www/en-us/downloads.html>

## flume

https://cwiki.apache.org/confluence/display/Hive/GettingStarted

启动

bin/flume-ng agent --conf conf --conf-file conf/thrift-logger.conf --name agent1 -Dflume.root.logger=INFO,console

curl -X POST -d ‘[{ "headers" : { "timestamp" : "434324343", "host" :"random\_host.example.com", "field1" : "val1" }, "body" : "random\_body" }]‘ localhost:9000

### thrift 到 hdfs

tier1.sources = r1

tier1.channels = c1

tier1.sinks = k1

#tier1.sources.r1.interceptors = i1

#tier1.sources.r1.interceptors.i1.type = org.apache.flume.interceptor.TimestampInterceptor$Builder

# For each one of the sources, the type is defined

tier1.sources.r1.type = thrift

tier1.sources.r1.channels = c1

tier1.sources.r1.bind = slave1

tier1.sources.r1.port = 9922

# Each sink's type must be defined

tier1.sinks.k1.channel = c1

tier1.sinks.k1.type = hdfs

tier1.sinks.k1.hdfs.path = hdfs://master1:8020/user/flume

tier1.sinks.k1.hdfs.filePrefix = event\_%y-%m-%d

tier1.sinks.k1.hdfs.fileSuffix = .log

tier1.sinks.k1.hdfs.rollSize = 0

tier1.sinks.k1.hdfs.rollCount = 0

#tier1.sinks.k1.hdfs.batchSize = 1

tier1.sinks.k1.hdfs.round = true

tier1.sinks.k1.hdfs.roundValue=24

tier1.sinks.k1.hdfs.roundUnit = hour

#tier1.sinks.k1.hdfs.threadsPoolSize = 25

tier1.sinks.k1.hdfs.useLocalTimeStamp = true

tier1.sinks.k1.hdfs.minBlockReplicas = 1

tier1.sinks.k1.fileType = DataStream

tier1.sinks.k1.writeFormat = TEXT

tier1.sinks.k1.rollInterval = 60

# Each channel's type is defined.

tier1.channels.c1.type = memory

# Other config values specific to each type of channel(sink or source)

# can be defined as well

# In this case, it specifies the capacity of the memory channel

tier1.channels.memoryChannel.capacity = 100

### flume-hdfs 按照时间关闭并新开文件

<http://www.bubuko.com/infodetail-1086730.html>

### flume-ng依据源文件名输出到HDFS指定文件名

<http://www.dzend.com/forum.php?mod=viewthread&tid=611>

### 【Flume】flume中sink到hdfs，文件系统频繁产生文件，文件滚动配置不起作用？

<http://doc.okbase.net/chiweitree/archive/126197.html>

### flume学习（八）：自定义source

http://blog.csdn.net/xiao\_jun\_0820/article/details/38312091

## Hive

<https://cwiki.apache.org/confluence/display/Hive/GettingStarted>

这里罗列常用操作，更多参考 <https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL#LanguageManualDDL-Create%2FDrop%2FTruncateTable>

### 简单的创建表

create table table\_name (

id int,

dtDontQuery string,

name string

)

### 创建有分区的表

create table table\_name (

id int,

dtDontQuery string,

name string

)

partitioned by (date string)

一个表可以拥有一个或者多个分区，每个分区以文件夹的形式单独存在表文件夹的目录下。

分区是以字段的形式在表结构中存在，通过describe table命令可以查看到字段存在，但是该字段不存放实际的数据内容，仅仅是分区的表示。

在Hive Select查询中一般会扫描整个表内容，会消耗很多时间做没必要的工作。有时候只需要扫描表中关心的一部分数据，因此建表时引入了partition概念。表中的一个 Partition 对应于表下的一个目录,Partition 就是辅助查询，缩小查询范围，加快数据的检索速度和对数据按照一定的规格和条件进行管理。

### 典型的默认创建表

[](javascript:void(0);)

CREATE TABLE page\_view(

viewTime INT,

userid BIGINT,

page\_url STRING,

referrer\_url STRING,

ip STRING COMMENT 'IP Address of the User')

COMMENT 'This is the page view table'

PARTITIONED BY(dt STRING, country STRING)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY '\001'

COLLECTION ITEMS TERMINATED BY '\002'

MAP KEYS TERMINATED BY '\003'

STORED AS TEXTFILE;

[](javascript:void(0);)

这里创建了表page\_view,有表的注释，一个字段ip的注释，分区有两列,分别是dt和country。

[ROW FORMAT DELIMITED]关键字，是用来设置创建的表在加载数据的时候，支持的列分隔符。不同列之间用一个'\001'分割,集合(例如array,map)的元素之间以'\002'隔开,map中key和value用'\003'分割。

[STORED AS file\_format]关键字是用来设置加载数据的数据类型,默认是TEXTFILE，如果文件数据是纯文本，就是使用 [STORED AS TEXTFILE]，然后从本地直接拷贝到HDFS上，hive直接可以识别数据。

### 常用的创建表

[](javascript:void(0);)

CREATE TABLE login(

userid BIGINT,

ip STRING,

time BIGINT)

PARTITIONED BY(dt STRING)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY '\t'

STORED AS TEXTFILE;

[](javascript:void(0);)

 LOAD DATA LOCAL INPATH './examples/files/kv1.txt' OVERWRITE INTO TABLE pokes;

### 创建外部表

如果数据已经存在HDFS的'/user/hadoop/warehouse/page\_view'上了，如果想创建表，指向这个路径，就需要创建外部表:

[](javascript:void(0);)

CREATE EXTERNAL TABLE page\_view(

viewTime INT,

userid BIGINT,

page\_url STRING,

referrer\_url STRING,

ip STRING COMMENT 'IP Address of the User',

country STRING COMMENT 'country of origination')

COMMENT 'This is the staging page view table'

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\054'

STORED AS TEXTFILE

LOCATION '/user/hadoop/warehouse/page\_view';

[](javascript:void(0);)

创建表，有指定EXTERNAL就是外部表，没有指定就是内部表，内部表在drop的时候会从HDFS上删除数据，而外部表不会删除。

外部表和内部表一样，都可以有分区，如果指定了分区，那外部表建了之后，还要修改表添加分区。

外部表如果有分区，还可以加载数据，覆盖分区数据，但是外部表删除分区，对应分区的数据不会从HDFS上删除，而内部表会删除分区数据。

### 指定数据库创建表

如果不指定数据库，hive会把表创建在default数据库下，假设有一个hive的数据库mydb,要创建表到mydb,如下:

CREATE TABLE mydb.pokes(foo INT,bar STRING);

或者是

use mydb; --把当前数据库指向mydb

CREATE TABLE pokes(foo INT,bar STRING);

### 复制表结构

CREATE TABLE empty\_table\_name LIKE table\_name;

根据table\_name创建一个空表empty\_table\_name,empty\_table\_name没有任何数据。

**create-table-as-selectt (CTAS)**

CTAS创建的表是原子性的，这意味着，该表直到所有的查询结果完成后，其他用户才可以看到完整的查询结果表。

CTAS唯一的限制是目标表，不能是一个有分区的表，也不能是外部表。

简单的方式

CREATE TABLE new\_key\_value\_store

AS

SELECT (key % 1024) new\_key, concat(key, value) key\_value\_pair FROM key\_value\_store;

复杂的方式

CREATE TABLE new\_key\_value\_store

ROW FORMAT SERDE "org.apache.hadoop.hive.serde2.columnar.ColumnarSerDe"

STORED AS RCFile AS

SELECT (key % 1024) new\_key, concat(key, value) key\_value\_pair

FROM key\_value\_store

SORT BY new\_key, key\_value\_pair;

### 删除表

DROP TABLE table\_name;

DROP TABLE IF EXISTS table\_name;

删除表会移除表的元数据和数据，而HDFS上的数据，如果配置了Trash，会移到.Trash/Current目录下。

删除外部表时，表中的数据不会被删除。

### 截断表

TRUNCATE TABLE table\_name;

TRUNCATE TABLE table\_name PARTITION (dt='20080808');

从表或者表分区删除所有行，不指定分区，将截断表中的所有分区，也可以一次指定多个分区，截断多个分区。

### Hive之partition

时间 2014-06-25 11:01:32 [ITeye-博客](http://www.tuicool.com/sites/jaeeAn)

原文

[http://snv.iteye.com/blog/2085030](http://snv.iteye.com/blog/2085030?utm_source=tuicool&utm_medium=referral)

主题 [Hive](http://www.tuicool.com/topics/11020065) [HDFS](http://www.tuicool.com/topics/11030091) [Nosql](http://www.tuicool.com/topics/11000065)

概述

hive的 partition可以认为是 RMDB中的分区，目的是query时减少全表扫描。

使用：

创建分区：

create EXTERNAL TABLE IF NOT EXISTS p(

id STRING COMMENT 'id'

) partitioned by (seq int)

STORED AS SEQUENCEFILE

LOCATION 'hdfs:///hive/hdfs/p/';

当数据插入的时候会以hdfs:///hive/hdfs/p/为基础构建以分区=值为名的目录

使用分区：

insert into table p partition(seq=1) select count(1) from dual;

此时会形成hdfs:///hive/hdfs/p/seq=1/数据 的文件目录格式，在insert into的时候需要指定分区否则会报如下错误：

Need to specify partition columns because the destination table is partitioned

查询操作：

select \* from p where seq=1

可以使用多个分区：

create EXTERNAL TABLE IF NOT EXISTS pp(

id STRING COMMENT 'id'

) partitioned by (seq int,num int)

STORED AS SEQUENCEFILE

LOCATION 'hdfs:///hive/hdfs/pp/';

此时如果写入数据指定了分区那么就会形成 hdfs:///hive/hdfs/pp/seq=值/num=值/数据 的目录结构

查询分区：

可以查询某表具有的分区：

show partitions pp;

 可以基于分区删除数据：

ALTER TABLE pp DROP PARTITION (seq=1,num=1);

pp为外部表那么其元数据被删除，真实数据扔存在，可以通过hadoop fs查看到

ps：

1.执行insert overwrite 的时候指定分区中的数据会被删除，其他分区数据不受影响

2.一个表可以有多个分区，分区列是伪劣不是真实存在的列以目录的形式存在

### 创建索引

create index user\_index on table user(id)

alter index user\_index on user rebuild;

### 在HIVE中使用AVRO

create EXTERNAL table avrotest(gameId string, gameName string)

ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.avro.AvroSerDe'

STORED AS INPUTFORMAT 'org.apache.hadoop.hive.ql.io.avro.AvroContainerInputFormat'

OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.avro.AvroContainerOutputFormat'

LOCATION '/user/hive/avro/'

TBLPROPERTIES ('avro.schema.literal'='{

"name": "my\_record",

"type": "record",

"fields": [

{"name":"gameId", "type":"string"},

{"name":"gameName", "type":"string"}]}');

CREATE TABLE test2(name1 STRING,name2 STRING)

ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' COLLECTION ITEMS TERMINATED BY ':' MAP KEYS TERMINATED BY '#' LINES TERMINATED BY '\n'

STORED AS TEXTFILE;

LOCATION '/user/hive/avro/'

## Impala

<http://www.cloudera.com/content/www/en-us/documentation/enterprise/latest/topics/impala_tutorial.html>

<http://www.cloudera.com/content/www/en-us/documentation/archive/impala/2-x/2-1-x/topics/impala_jdbc.html>

impala-shell -i localhost –quiet

导入数据直接通过hive来导入比较好，现在hive中建张表，制定分隔符， 然后再 load data local inpath ......overwrite into table .......

然后impala在 refresh 那张表 或者 invalidate metadata 后。 查询。

### 导入数据

<http://blog.itpub.net/10037372/viewspace-1411981/>

### 性能调优

http://my.oschina.net/weiqingbin/blog/192612#OSC\_h3\_12

采集表的统计信息用于查询优化

在impala 在使用

compute stats tablename;

在hive中使用

为整个表或特定分区在 Hive 中执行 ANALYZE TABLE 语句：

ANALYZE TABLEtablename[PARTITION(partcol1[=val1],partcol2[=val2], ...)] COMPUTE STATISTICS [NOSCAN];

例如，为非分区表采集统计信息：

ANALYZE TABLE customer COMPUTE STATISTICS;

为以 state 和 city 分区列的分区表 store 表采集所有分区的统计信息：

ANALYZE TABLE store PARTITION(s\_state, s\_county) COMPUTE STATISTICS;

只采集分区表 store 中 California 分区的统计信息：

ANALYZE TABLE store PARTITION(s\_state='CA', s\_county) COMPUTE STATISTICS;

### Impala SQL 语言元素

http://my.oschina.net/weiqingbin/blog/189413?fromerr=E9WBdcGh#OSC\_h2\_28

### Impala 表使用 Avro 文件格式（翻译）

<http://my.oschina.net/weiqingbin/blog/194411?fromerr=HVOsFIBf>

### CM配置impala使用本地时间

**Clusters** > **Services** > **Impala** > **Configuration** > **Impala Daemon > Impala Daemon 命令行参数高级配置代码段（安全阀）**

-use\_local\_tz\_for\_unix\_timestamp\_conversions=true

### Cloudera Impala 常见问题（翻译）

http://my.oschina.net/weiqingbin/blog/196143#OSC\_h3\_29

## HBase

<http://javacrazyer.iteye.com/blog/1186881>

### Impala与HBase整合实践

<http://shiyanjun.cn/archives/526.html>

## Kafka

### Kafka+Storm+HDFS整合实践

<http://shiyanjun.cn/archives/934.html>

## AVRO

<https://github.com/sidshetye/HelloAvro/tree/master/Avro/Lib>

### 在Hive中使用Avro

http://www.iteblog.com/archives/1007

### Hue

<http://demo.gethue.com>

### Sqoop

#### Sqoop中文手册

<http://www.zihou.me/html/2014/01/28/9114.html>

## Storm

### 基于HBase做Storm 实时计算指标存储

<http://ifeve.com/getting-started-with-stom-index/>

### 《Storm入门》中文版

<http://ifeve.com/getting-started-with-stom-index/>

### 使用Storm实现实时大数据分析

<http://qq85609655.iteye.com/blog/2035717>

### 基于HBase做Storm 实时计算指标存储

<http://developer.51cto.com/art/201510/493422.htm>

### RealTimeTraffic

<https://github.com/whughchen/RealTimeTraffic/tree/master/src/storm/realTraffic/spout>

### Storm 常见模式

#### 批处理

http://www.cnblogs.com/panfeng412/archive/2012/06/19/storm-common-patterns-of-batching.html

# MYSQL

## centos7下使用yum安装mysql

CentOS7的yum源中默认好像是没有mysql的。为了解决这个问题，我们要先下载mysql的repo源。

**1. 下载mysql的repo源**

$ wget http://repo.mysql.com/mysql-community-release-el7-5.noarch.rpm

**2. 安装mysql-community-release-el7-5.noarch.rpm包**

$ sudo rpm -ivh mysql-community-release-el7-5.noarch.rpm

安装这个包后，会获得两个mysql的yum repo源：/etc/yum.repos.d/mysql-community.repo，/etc/yum.repos.d/mysql-community-source.repo。

**3. 安装mysql**

$ sudo yum install mysql-server

根据步骤安装就可以了，不过安装完成后，没有密码，需要重置密码。

**4. 重置密码**

重置密码前，首先要登录

$ mysql -u root

登录时有可能报这样的错：ERROR 2002 (HY000): Can‘t connect to local MySQL server through socket ‘/var/lib/mysql/mysql.sock‘ (2)，原因是/var/lib/mysql的访问权限问题。下面的命令把/var/lib/mysql的拥有者改为当前用户：

$ sudo chown -R openscanner:openscanner /var/lib/mysql

然后，重启服务：

$ service mysqld restart

接下来登录重置密码：

$ mysql -u root

mysql > use mysql;

mysql > update user set password=password(‘123456‘) where user=‘root‘;

mysql > exit;

**5. 开放3306端口**

$ sudo vim /etc/sysconfig/iptables

添加以下内容：

-A INPUT -p tcp -m state --state NEW -m tcp --dport 3306 -j ACCEPT

保存后重启防火墙：

$ sudo service iptables restart

这样从其它客户机也可以连接上mysql服务了。

impala-shell -i localhost --quiet

# SVN

## SVN使用import导入数据到版本库 (2010-11-09 14:34:52)

|  |  |
| --- | --- |
| [转载](javascript:;)  [▼](javascript:;)  标签： [拷贝](http://search.sina.com.cn/?c=blog&q=%BF%BD%B1%B4&by=tag) [版本](http://search.sina.com.cn/?c=blog&q=%B0%E6%B1%BE&by=tag) [目录](http://search.sina.com.cn/?c=blog&q=%C4%BF%C2%BC&by=tag) [服务器](http://search.sina.com.cn/?c=blog&q=%B7%FE%CE%F1%C6%F7&by=tag) [组文件](http://search.sina.com.cn/?c=blog&q=%D7%E9%CE%C4%BC%FE&by=tag) [it](http://search.sina.com.cn/?c=blog&q=it&by=tag) | 分类： [c/linux/server](http://blog.sina.com.cn/s/articlelist_1267930890_8_1.html) |

正文开始

有两种方法可以将新文件引入Subversion版本库：svn import和svn add，现在讨论svn import

svn import是将未版本化文件导入版本库的最快方法，会根据需要创建中介目录。svn import不需要一个工作拷贝，你的文件会直接提交到版本库，这通常用在你希望将一组文件加入到Subversion版本库时，例如：

1)首先，服务器管理员在SVN服务器上创建项目版本库newrepos

$ svnadmin create /usr/local/svn/newrepos

2)然后，在客户端使用svn命令把本地文件导入到服务器上的项目版本库

$ svn import mytree <file:///usr/local/svn/newrepos/some/project>  -m "Initial import"

Adding         mytree/foo.c

Adding         mytree/bar.c

Adding         mytree/subdir

Adding         mytree/subdir/quux.h

Committed revision 1.

在上一个例子里，将会拷贝目录mytree到版本库的some/project下：

$ svn list file:///usr/local/svn/newrepos/some/project

bar.c

foo.c

subdir/

注意，在导入之后，原来的目录树并没有转化成工作拷贝，为了开始工作，你还是需要运行svn checkout导出一个工作拷贝。

推荐的版本库布局

尽管Subversion的灵活性允许你自由布局版本库，但我们有一套推荐的方式，创建一个trunk目录来保存开发的“主线”，一个branches目录存放分支拷贝，tags目录保存标签拷贝，例如：

$ svn list file:///usr/local/svn/repos

/trunk

/branches

/tags

svn import cloudbase svn://192.168.21.8/hadoop/cloud/cloudbase -m "导入项目"

# Git

## 解决Git 每次提交时都要输入用户名和密码的缓存机制的设置

在使用 Git 的时候，经常会遇到需要频繁输入密码的情况，在每次 commit 的时候都要求输入你的用户名和密码，如果提交频繁的话甚是不便。

GNU Linux 用户可以使用 ‘cache’ 认证助手包来缓存认证信息，运行下面的命令来启用凭据缓存，启用后每次输入密码将保存一小时（3600秒）

git config --global  credential.helper  'cache --timeout 3600'

查看 ‘cache’ 认证助手的所有配置:

git helpcredential-cache

# 消息队列协议

## RabbitMQ基础概念详细介绍

<http://blog.csdn.net/whycold/article/details/41119807>

## RabbitMq、ActiveMq、ZeroMq、kafka之间的比较,资料汇总

<http://blog.csdn.net/linsongbin1/article/details/47781187>

**java 消息机制 ActiveMQ入门实例**

http://heisetoufa.iteye.com/blog/1908335

## Rabbitmq

创建用户

rabbitmqctl add\_user cmz pwd

设置权限

rabbitmqctl set\_permissions -p "/" cmz ".\*" ".\*" ".\*"

# Eclipse

## Eclipse将引用了第三方jar包的Java项目打包成jar文件的两种方法

http://www.cnblogs.com/lanxuezaipiao/p/3291641.html

# Ant

## 使用ant获取文件列表

<http://blog.sina.com.cn/s/blog_605f5b4f0100xnij.html>

# Maven

<http://mvnrepository.com>

http://maven.apache.org

# Netty

## 《Netty 权威指南》—— AIO创建的TimeClient源码分析

<http://ifeve.com/netty-4-2/>

http://ifeve.com/netty5-user-guide/

## Netty Best Practices a.k.a Faster == Better

http://normanmaurer.me/presentations/2014-facebook-eng-netty/slides.html#1.0

# WEB技术

## 实时更新网页

基于[Express](http://expressjs.com/)和[Socket.io](http://socket.io/)两个框架实现了向浏览器推送内容更新

# Redis

## 安装EPEL

### CentOS6

的请编辑/etc/yum.repos.d/epel.repo输入以下内容

|  |
| --- |
| [epel]  name=Extra Packages **for** Enterprise Linux 6 - $basearch  #baseurl=http://download.fedoraproject.org/pub/epel/6/$basearch  mirrorlist=https://mirrors.fedoraproject.org/metalink?repo=epel-6&arch=$basearch  failovermethod=priority  enabled=1  gpgcheck=1  gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-6    [epel-debuginfo]  name=Extra Packages **for** Enterprise Linux 6 - $basearch - Debug  #baseurl=http://download.fedoraproject.org/pub/epel/6/$basearch/debug  mirrorlist=https://mirrors.fedoraproject.org/metalink?repo=epel-debug-6&arch=$basearch  failovermethod=priority  enabled=0  gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-6  gpgcheck=1    [epel-source]  name=Extra Packages **for** Enterprise Linux 6 - $basearch - Source  #baseurl=http://download.fedoraproject.org/pub/epel/6/SRPMS  mirrorlist=https://mirrors.fedoraproject.org/metalink?repo=epel-source-6&arch=$basearch  failovermethod=priority  enabled=0  gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-6  gpgcheck=1 |

然后执行yum clean all

yum makecache即可

### centos7的epel源

编辑/etc/yum.repos.d/epel.repo

|  |
| --- |
| [epel]  name=Extra Packages **for** Enterprise Linux 7 - $basearch  #baseurl=http://download.fedoraproject.org/pub/epel/7/$basearch  mirrorlist=https://mirrors.fedoraproject.org/metalink?repo=epel-7&arch=$basearch  failovermethod=priority  enabled=1  gpgcheck=0  gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-7  [epel-debuginfo]  name=Extra Packages **for** Enterprise Linux 7 - $basearch - Debug  #baseurl=http://download.fedoraproject.org/pub/epel/7/$basearch/debug  mirrorlist=https://mirrors.fedoraproject.org/metalink?repo=epel-debug-7&arch=$basearch  failovermethod=priority  enabled=0  gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-7  gpgcheck=1  [epel-source]  name=Extra Packages **for** Enterprise Linux 7 - $basearch - Source  #baseurl=http://download.fedoraproject.org/pub/epel/7/SRPMS  mirrorlist=https://mirrors.fedoraproject.org/metalink?repo=epel-source-7&arch=$basearch  failovermethod=priority  enabled=0  gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-7  gpgcheck=1 |

**方法一：使用命令安装（前提是已经安装了EPEL）。**

## 安装redis:

### yum -y install redis

启动/停止/重启 Redis

启动服务：

1

systemctl start redis.service

停止服务：

systemctl stop redis.service

重启服务：

systemctl restart redis.service

检查状态：

[root@idoseek ~]# systemctl status redis.service

redis.service - Redis persistent key-value database

   Loaded: loaded (/usr/lib/systemd/system/redis.service; enabled)

   Active: active (running) since 二 2014-10-21 21:37:22 EDT; 5h 26min ago

 Main PID: 30413 (redis-server)

   CGroup: /system.slice/redis.service

           └─30413 /usr/bin/redis-server 127.0.0.1:6379

10月 21 21:37:22 idoseek.com systemd[1]: Started Redis persistent key-value database.

随系统启动服务：

[root@idoseek ~]# systemctl enable redis.service

ln -s '/usr/lib/systemd/system/redis.service' '/etc/systemd/system/multi-user.target.wants/redis.service'

关闭随机启动：

[root@idoseek ~]# systemctl disable redis.service

rm '/etc/systemd/system/multi-user.target.wants/redis.service'

### 方法二：编译安装

下载安装编译:

# wget http://download.redis.io/releases/redis-2.8.17.tar.gz

# tar xzf redis-2.8.17.tar.gz

# cd redis-2.8.17

# make

# make install

设置配置文件路径:

# mkdir -p /etc/redis && cp redis.conf /etc/redis

修改配置文件：

# vim /etc/redis/redis.conf

修改为： daemonize yes

启动Redis:

# /usr/local/bin/redis-server /etc/redis/redis.conf

#关闭服务

redis-cli shutdown

或者在cli中执行shutdown

redis 127.0.0.1:6379> shutdown

清除缓存

redis-cli flushall

更多文档请参考软件包内的“README”文件。

查看状态 ：

# ss -nlp|grep redis

或者

# [ps](http://www.111cn.net/fw/photo.html) -ef | grep redis

### 下面介绍为PHP添加redis插件。

从官网下载最新的拓展，地址：http://pecl.php.net/package/redis或者https://github.com/phpredis/phpredis

#wget http://pecl.php.net/get/redis-2.2.5.tgz

#phpize

#./configure --prefix=/opt/redis --enable-redis --with-php-config=/opt/php/bin/php-config

#make && make install

把拓展添加至php.ini，重启php-fpm：

service php-fpm restart

# CentOS7安装配置redis-3.0.0

**一.安装必要包**

yum install gcc

**二.linux下安装**

#下载

wget http://download.redis.io/releases/redis-3.0.0.tar.gz

tar zxvf redis-3.0.0.tar.gz

cd redis-3.0.0

#如果不加参数,linux下会报错

make MALLOC=libc

**安装好之后,启动文件**

#启动redis

src/redis-server &

#关闭redis

src/redis-cli shutdown

**测试redis**

$ src/redis-cli

127.0.0.1:6379> set foo bar

OK

127.0.0.1:6379> get foo

"bar"

$

测试成功

**3.redis cluster集群搭建**

**建立本机测试环境**

建立运行目录

#建立redis运行目录

mkdir -p redis-server/7000/

#复制默认的配置文档

cp redis-3.0.0/redis.conf redis-server/redis.default.conf

#把编译好的server复制到运行目录

cp redis-3.0.0/src/redis-server redis-server/7000/

建立独立配置文件

#在7000目录下建立redis的配置文档

vim redis-server/7000/redis.conf

文件内容

[](javascript:void(0);)

#redis-server/7000/redis.conf

include /root/redis-server/redis.default.conf

pidfile /var/run/redis-7000.pid

port 7000

cluster-enabled yes

cluster-config-file redis-node-7000.conf

cluster-node-timeout 5000

appendonly yes

[](javascript:void(0);)

复制运行目录(模拟集群环境)

#复制目录

cp -R 7000/ 7001/

cp -R 7000/ 7002/

cp -R 7000/ 7003/

cp -R 7000/ 7004/

cp -R 7000/ 7005/

修改相应配置文件的端口和文件名

建立启动脚本 redis-server/redis-start.sh

[](javascript:void(0);)

#!/bin/sh

/root/redis-server/7000/redis-server /root/redis-server/7000/redis.conf &

/root/redis-server/7001/redis-server /root/redis-server/7001/redis.conf &

/root/redis-server/7002/redis-server /root/redis-server/7002/redis.conf &

/root/redis-server/7003/redis-server /root/redis-server/7003/redis.conf &

/root/redis-server/7004/redis-server /root/redis-server/7004/redis.conf &

/root/redis-server/7005/redis-server /root/redis-server/7005/redis.conf &

[](javascript:void(0);)

**配置集群**

安装ruby

yum install ruby-devel.x86\_64

安装redis gem

# gem install redis

Fetching: redis-3.2.1.gem (100%)

Successfully installed redis-3.2.1

Parsing documentation for redis-3.2.1

Installing ri documentation for redis-3.2.1

1 gem installed

使用脚本建立集群机制

*在create的时候,加上参数--replicas 1 表示为每个master分配一个salve,如例子,则是3个master 3个salve*

[](javascript:void(0);)

# ./redis-trib.rb create 127.0.0.1:7000 127.0.0.1:7001 127.0.0.1:7002 127.0.0.1:7003 127.0.0.1:7004 127.0.0.1:7005

>>> Creating cluster

Connecting to node 127.0.0.1:7000: OK

Connecting to node 127.0.0.1:7001: OK

Connecting to node 127.0.0.1:7002: OK

Connecting to node 127.0.0.1:7003: OK

Connecting to node 127.0.0.1:7004: OK

Connecting to node 127.0.0.1:7005: OK

>>> Performing hash slots allocation on 6 nodes...

Using 6 masters:

127.0.0.1:7000

127.0.0.1:7001

127.0.0.1:7002

127.0.0.1:7003

127.0.0.1:7004

127.0.0.1:7005

M: f3dd250e4bc145c8b9f864e82f65e00d1ba627be 127.0.0.1:7000

slots:0-2730 (2731 slots) master

M: 1ba602ade59e0770a15128b193f2ac29c251ab5e 127.0.0.1:7001

slots:2731-5460 (2730 slots) master

M: 4f840a70520563c8ef0d7d1cc9d5eaff6a1547a2 127.0.0.1:7002

slots:5461-8191 (2731 slots) master

M: 702adc7ae9caf1f6702987604548c6fc1d22e813 127.0.0.1:7003

slots:8192-10922 (2731 slots) master

M: 4f87a11d2ea6ebe9caf02c9dbd827a3dba8a53cf 127.0.0.1:7004

slots:10923-13652 (2730 slots) master

M: 216bbb7da50bd130da16a327c76dc6d285f731b3 127.0.0.1:7005

slots:13653-16383 (2731 slots) master

Can I set the above configuration? (type 'yes' to accept): yes

>>> Nodes configuration updated

>>> Assign a different config epoch to each node

>>> Sending CLUSTER MEET messages to join the cluster

Waiting for the cluster to join...

>>> Performing Cluster Check (using node 127.0.0.1:7000)

M: f3dd250e4bc145c8b9f864e82f65e00d1ba627be 127.0.0.1:7000

slots:0-2730 (2731 slots) master

M: 1ba602ade59e0770a15128b193f2ac29c251ab5e 127.0.0.1:7001

slots:2731-5460 (2730 slots) master

M: 4f840a70520563c8ef0d7d1cc9d5eaff6a1547a2 127.0.0.1:7002

slots:5461-8191 (2731 slots) master

M: 702adc7ae9caf1f6702987604548c6fc1d22e813 127.0.0.1:7003

slots:8192-10922 (2731 slots) master

M: 4f87a11d2ea6ebe9caf02c9dbd827a3dba8a53cf 127.0.0.1:7004

slots:10923-13652 (2730 slots) master

M: 216bbb7da50bd130da16a327c76dc6d285f731b3 127.0.0.1:7005

slots:13653-16383 (2731 slots) master

[OK] All nodes agree about slots configuration.

>>> Check for open slots...

>>> Check slots coverage...

[OK] All 16384 slots covered.

[](javascript:void(0);)

 如果需要全部重新自动配置,则删除所有的配置好的cluster-config-file,重新启动所有的redis-server,然后重新执行配置命令即可

# Centos开机自启动redis

一：原理介绍

先看下当前系统已设置的开机自启动程序有哪些：

[root@localhost init.d]# chkconfig --list

Note: This output shows SysV services only and does not include native

      systemd services. SysV configuration data might be overridden by native

      systemd configuration.

      If you want to list systemd services use 'systemctl list-unit-files'.

      To see services enabled on particular target use

      'systemctl list-dependencies [target]'.

netconsole     0:off1:off

2:off3:off

4:off5:off

6:off

network         0:off1:off

2:on3:on

4:on5:on

6:off

[root@localhost init.d]# ls /etc/init.d/ -lrt

total 40

-rwxr-xr-x. 1 root root  6470 Jan 15  2015 network

-rwxr-xr-x. 1 root root  2989 Jan 15  2015 netconsole

-rw-r--r--. 1 root root 13430 Jan 15  2015 functions

-rw-r--r--. 1 root root  1160 Mar  6  2015 README

-rw-r--r--. 1 root root    37 Nov 30 05:43 dump.rdb

从上面可以看出系统已设置的开机自启动程序有network和netconsole，新增开机自启动程序，基本原理为：

系统开机启动时会去加载/etc/init.d/下面的脚本，通常而言每个脚本文件会自定义实现程序的启动；若想将新的程序开机自启动，只需在该目录下添加一个自定义启动程序的脚本，然后设置相应规则即可(如我在/etc/init.d/下新建一个redis的脚本，开机启动时会去加载执行该脚本；当然这只是开机设置自启动的一种方法)

二：实践

废话少说，直接来步骤：

1、设置redis.conf中daemonize为yes,确保守护进程开启。

2、编写开机自启动脚本

vi /etc/init.d/redis

脚本内容如下：

# chkconfig: 2345 10 90

# description: Start and Stop redis

PATH=/usr/local/bin:/sbin:/usr/bin:/bin

REDISPORT=6379

EXEC=/usr/local/redis-3.0.0-beta5/src/redis-server

REDIS\_CLI=/usr/local/redis-3.0.0-beta5/src/redis-cli

PIDFILE=/var/run/redis.pid

CONF="/usr/local/redis-3.0.0-beta5/redis.conf"

AUTH="1234"

case "$1" in

start)

if [ -f $PIDFILE ]

then

echo "$PIDFILE exists, process is already running or crashed."

else

echo "Starting Redis server..."

$EXEC $CONF

fi

if [ "$?"="0" ]

then

echo "Redis is running..."

fi

;;

stop)

if [ ! -f $PIDFILE ]

then

echo "$PIDFILE exists, process is not running."

else

PID=$(cat $PIDFILE)

echo "Stopping..."

$REDIS\_CLI -p $REDISPORT SHUTDOWN

sleep 2

while [ -x $PIDFILE ]

do

echo "Waiting for Redis to shutdown..."

sleep 1

done

echo "Redis stopped"

fi

;;

restart|force-reload)

${0} stop

${0} start

;;

\*)

echo "Usage: /etc/init.d/redis {start|stop|restart|force-reload}" >&2

exit 1

esac

3、写完后保存退出VI

4、设置权限

chmod 755 redis

5、启动测试

/etc/init.d/redis start

启动成功会提示如下信息：

Starting Redis server...

2358:M 30 Nov 05:34:16.991 \* Increased maximum number of open files to 10032 (it was originally set to 1024).

                \_.\_

           \_.-``\_\_ ''-.\_

      \_.-``    `.  `\_.  ''-.\_           Redis 2.9.54 (00000000/0) 64 bit

  .-`` .-```.  ```\/    \_.,\_ ''-.\_

 (    '      ,       .-`  | `,    )     Running in stand alone mode

 |`-.\_`-...-` \_\_...-.``-.\_|'` \_.-'|     Port: 6379

 |    `-.\_   `.\_    /     \_.-'    |     PID: 2358

  `-.\_    `-.\_  `-./  \_.-'    \_.-'

 |`-.\_`-.\_    `-.\_\_.-'    \_.-'\_.-'|

 |    `-.\_`-.\_        \_.-'\_.-'    |           http://redis.io

  `-.\_    `-.\_`-.\_\_.-'\_.-'    \_.-'

 |`-.\_`-.\_    `-.\_\_.-'    \_.-'\_.-'|

 |    `-.\_`-.\_        \_.-'\_.-'    |

  `-.\_    `-.\_`-.\_\_.-'\_.-'    \_.-'

      `-.\_    `-.\_\_.-'    \_.-'

          `-.\_        \_.-'

              `-.\_\_.-'

2358:M 30 Nov 05:34:16.993 # Server started, Redis version 2.9.54

2358:M 30 Nov 05:34:16.993 # WARNING overcommit\_memory is set to 0! Background save may fail under low memory condition. To fix this issue add 'vm.overcommit\_memory = 1' to /etc/sysctl.conf and then reboot or run the command 'sysctl vm.overcommit\_memory=1' for this to take effect.

2358:M 30 Nov 05:34:16.993 \* The server is now ready to accept connections on port 6379

使用redis-cli测试：

[best\_husband@localhost src]$ /usr/local/redis-3.0.0-beta5/src/redis-cli

127.0.0.1:6379> set name usbdrivers

OK

127.0.0.1:6379> get name

"usbdrivers"

127.0.0.1:6379> quit

[best\_husband@localhost src]$

6、设置开机自启动

chkconfig redis on

7、关机重启测试

reboot

然后在用redis-cli测试即可。

# Python

## 开发工具

<http://www.jetbrains.com/pycharm/>

# Tomcat

## 乱码解决

在server.xml 中找到Connector

添加 uRIEncoding=*"utf-8"* useBodyEncodingForURI=*"true"*

例如:

<Connector connectionTimeout=*"20000"* port=*"8080"* protocol=*"HTTP/1.1"* redirectPort=*"8443"* uRIEncoding=*"utf-8"* useBodyEncodingForURI=*"true"*/>

# Resin

Eclipse 配置 resin

服务器启动默认会加上项目名作为二级目录，用以下方法去掉二级目录

<!-- the default host, matching any host name -->

<host id=*""* root-directory=*"."*>

<!--

- webapps can be overridden/extended in the resin.xml

-->

<web-app id=*"/"* root-directory=*"webapps/datacenter"*>

<character-encoding>UTF-8</character-encoding>

</web-app>

</host>

# Disruptor

<http://ifeve.com/disruptor-getting-started/>

# 高可用

## haproxy+keepalived实现高可用负载均衡

<http://www.cnblogs.com/dkblog/archive/2011/07/06/2098949.html>

# 缓存

Ignite

# 前端技术

## bootstrap

<http://www.runoob.com/bootstrap/bootstrap-forms.html>

# 序列化框架

## MessagePack, Protocol Buffers和Thrift序列化框架原理和比较说明

http://jimmee.iteye.com/blog/2042420