# Springboot笔记

目录

# liunx 命令:

**（可执行文件是绿色的）**

Ifconfig 查看ip地址

ps -ef|grep redis 检查是否启动服务

kill -s 9 1827 强制杀死进程 （1827是进程的意思）

ll查看目录、cd 进入文件夹

cd空格点点 cd .. 返回上一级  
cd ~ 返回默认[根目录](https://www.baidu.com/s?wd=%E6%A0%B9%E7%9B%AE%E5%BD%95&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)  
cd ../.. 返回上级的上级

cp 拷贝

vi 文本编辑器

tar -zxvf 文件名 （解压文件）

退出vi，i 开始编辑 esc键退出编辑 但不保存更改 :q! ， :wq! 强制退出并保存，:w 保存，但不退出vi。

mkdir 创建文件夹

rm 删除

mv 移动 （mv redis-trib.rb ./redis-cluster/）

dd 删除一行（文本编辑器）

chmod （赋予权限）

ps -ef|grep redis 检查是否启动服务

Redis 前台启动:(不推荐)：

解压命令：

yum 在线安装 yum install安装的意思

wget 下载

tar tar -xzvf redis-3.2.8.tar.gz

rpm 类似windows中的.exe

make 编译文件

make install PREFIX=/usr/local/redis15 开始安装 prefix前缀指定安装到什么地方

./redis.server 启动redis服务

Ctrl+C 退出服务

Redis 后台启动 （推荐）：

cp /root/redis-3.2.8/redis.conf ./ 拷贝redis.conf文件至当前文件夹 ./(当前文件夹)

vi 文本编辑器 进入文本编辑器

i 进入编辑

/ 查找

daemonize no 改成yes

#bind 127.0.0.1 注释本机 用来实现远程访问

# requirepass foobared 修改密码 requirepass root

:wq! 强制退出保存

./redis-server redis.conf 启动服务

ps -ef|grep redis 检查是否启动服务

./redis-cli -a root连接服务进入客户端

auth root 输入密码登录

ping 测试心跳 返回pong表示成功

keys \* 查看所有键值

set key1 value1

get key1

flushall（清空所有键值对）

select 1(切换默认库，redis默认16个库，select 用来切换库)

quit 退出客户端

service iptables stop 关闭防火墙（用来开放6379端口,

下次打开虚拟机还会重新打开防火墙）

chkconfig iptables off 永久关闭防火墙

下面是red hat/CentOs7关闭防火墙的命令!

1:查看防火墙状态

systemctl status firewalld

service iptables status

2:暂时关闭防火墙

systemctl stop firewalld

service iptables stop

3:永久关闭防火墙

systemctl disable firewalld

chkconfig iptables off

4:重启防火墙

systemctl enable firewalld

service iptables restart

# Redis 集群搭建

yum install ruby (安装ruby)

yum install rubygems (安装rubygems )

将redis-3.0.0.gem 文件传输至liunx



gem install redis-3.0.0.gem 安装

进入安装的reids目录 cd /usr/local/redis15/

mkdir 创建文件夹 redis-cluster(cluster集群的意思)

cp /root/redis-3.2.8/src/redis-trib.rb ./ （复制redis-trib.rb到当前目录）

cp -r bin redis-cluster/7001 复制 bin目录及子目录下的所有文件到redis-cluster的7001目录

rm -f \*.rdb （删除所有的.rdb文件）(.rm -f 其中的，f参数 （f --force ） 忽略不存在的文件，不显示任何信息)

vi redis.conf (编辑redis.conf文件)

port 7001(改端口号)

cluster-enabled yes（开启集群）

拷贝5台redis（一共6台）（为什么拷贝5台，采用3主3从，”投票机制”超过半数连接不上则表示redis宕机，2台无法使用投票机制。50:50）

[root@192 redis-cluster]# cp -r 7001 7002

[root@192 redis-cluster]# cp -r 7001 7003

[root@192 redis-cluster]# cp -r 7001 7004

[root@192 redis-cluster]# cp -r 7001 7005

c[root@192 redis-cluster]# cp -r 7001 7006

修改5台redis的端口号

新建开启服务脚本： vi startall.sh (用来一次性修改6个reids的配置，开启服务)

脚本内容：

cd 7001

./redis-server redis.conf

cd ..

cd 7002

./redis-server redis.conf

cd ..

cd 7003

./redis-server redis.conf

cd ..

cd 7004

./redis-server redis.conf

cd ..

cd 7005

./redis-server redis.conf

cd ..

cd 7006

./redis-server redis.conf

chmod u+x startall.sh给脚本赋予权限（x权限 u=当前用户 x=x权限 没权限是白色的，赋予权限之后为绿色）

新建关闭服务脚本：vi stopall.sh

脚本内容：（-p 端口号的意思 shutdown 停止服务）

7001/redis-cli -p 7001 -a root shutdown

7002/redis-cli -p 7002 -a root shutdown

7003/redis-cli -p 7003 -a root shutdown

7004/redis-cli -p 7004 -a root shutdown

7005/redis-cli -p 7005 -a root shutdown

7006/redis-cli -p 7006 -a root shutdown

./startall.sh(启动服务)

由于配置了密码登录，而该命令在执行时默认没有密码，因此登录不上各个Redis实例，

此时用vi编辑器打开redis-trib.rb文件，搜索到如下一行：

@r =Redis.new(:host => @info[:host], :port => @info[:port], :timeout=>60)

修改成

@r =Redis.new(:host => @info[:host], :port => @info[:port], :timeout=>60,:password=>”root”)

执行如下命令创建Redis集群：

./redis-trib.rb create --replicas 1 192.168.132.129:7001 192.168.132.129:7002 192.168.132.129:7003 192.168.132.129:7004 192.168.132.129:7005 192.168.132.129:7006

（其中，replicas表示每个主节点的slave数量。在集群的创建过程中会分配主机和 从机，每个集群在创建的过程中都将分配一个唯一的id并分配到一段slot）

进入reids实例中输入命令登录：

./redis-cli -h 192.168.132.129 -p 7001 -a root -c（-h 主机名 -p 端口号 -a 密码 -c 以集群的 方式登录）

Cluster info 命令可以查询集群状态信息

Cluster nodes命令可以查询集群节点信息

# MongoDB安装

ps -ef|grep mongod检查是否启动服务

**1.下载MongoDB**

wget [https://fastdl.mongodb.org/linux/mongodb-liunx-x86\_64-4.0.0.tgz](https://fastdl.mgngodb.org/linux/mongodb-liunx-x86_64-4.0.0.tgz)

解压并改名

tar -zxvf mongodb-liunx-x86\_64-4.0.0.tgz

mv mongodb-liunx-x86\_64-4.0.0 mongodb

1. **配置MongoDB**

进入mongodb目录下，创建两个文件夹db和logs

Cd mongodb

Mkdir db

Mkdir logs

然后进入bin目录，创建一个新的mongoDB的配置文件mongo.conf,文件内容如下：

dbpath=/root/mongodb/db 配置表示数据存储目录

logpath=/root/mongodb/logs/mongodb.log 表示日志文件位置

port=27017 启动端口

fork=true 允许mongodb后台运行

1. **mongoDB的启动和关闭**

./mongod -f mongo.conf --bind\_ip\_all (-f表示配置文件位置，--bind\_ip\_all则允许所有远程地址连接该MongoDB实例)

启动成功后，在bin目录下执行mongo命令，进入控制台

./mongo

db.version()

如果看到MongoDB的版本号就表示安装成功：4.0.12

**关闭**

默认情况下。启动后连接的是test库，而关闭mongoDB的命令需要在admin 库，因此切换库，然后执行db.shutdownServer()

use admin

db.shutdownServer()

exit

1. **安全管理**

默认情况下，启动mongoDB没有登录密码，在生产环境中这是非常不安全的。

但是不同于mysql，oracle数据库，mongoDB中每个库都有独立的密码，在哪个库

中创建密码，就要验证哪个库的密码，要配置密码，首先要创建用户，例如在admin库中创建一个用户：

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

readWrite：权限为读写，为test数据库的权限

创建成功后退出实例重新启动，

./mongod -f mongo.conf --auth --bind\_ip\_all

进入到实例后切换数据库为admin

Use admin

./mongo

验证密码，返回1为认证成功！

Db.auth(“root”,”root”)

yum -y install zlib zlib-devel openssl openssl-devel pcre pcre-devel

# Liunx在线安装jdk

1.执行命令yum -y list java\*查看可安装java版本。

yum install -y java-1.8.0-openjdk-devel.x86\_64

查看安装是否完成 检查jdk环境 java -version

liunx安装卸载完整步骤

<https://www.cnblogs.com/ysgcs/p/9849263.html>

<https://www.cnblogs.com/wenbinshen/p/9049447.html>

创建文件夹 mkdir /usr/local/java

解压文件:

tar -zxvf jdk-8u171-linux-x64.tar.gz -C /usr/local/java

配置环境变量 vim /etc/profile

export JAVA\_HOME=/usr/local/java/jdk1.8.0\_171

export JRE\_HOME=${JAVA\_HOME}/jre

export CLASSPATH=.:${JAVA\_HOME}/lib:${JRE\_HOME}/lib

export PATH=${JAVA\_HOME}/bin:$PATH

执行profile文件 source /etc/profile

使配置文件生效

6.检查新安装的jdk java -version

结果如下

java version "1.8.0\_171"

Java(TM) SE Runtime Environment (build 1.8.0\_171-b11)

Java HotSpot(TM) 64-Bit Server VM (build 25.131-b11, mixed mode)

# ActiveMQ消息服务安装

网站：

<https://blog.csdn.net/Dream_ya/article/details/84570199>

# Docker容器

Docker 是一个开源的应用容器引擎，让开发者可以打包他们的应用以及依赖包到一个可移植的容器中,然后发布到任何流行的Linux机器或Windows 机器上,也可以实现虚拟化,容器是完全使用沙箱机制,相互之间不会有任何接口。

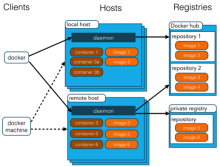
一个完整的Docker有以下几个部分组成：

DockerClient客户端

Docker Daemon守护进程

Docker Image镜像

DockerContainer容器 [2]



## 默认路径

Centos7安装docker之后，默认的镜像及容器存储路径为/var/lib/docker，可以使用命令docker info查看。

## Docker操作

### 1.启动docker：

systemctl stop docker.service  停止docker进程

systemctl start  docker.service  启动docker服务

systemctl restart docker 重启docker服务

systemctl enable docker 【设置开机自启】

### 查看docker版本：

docker -v

### 查找镜像:

docker search 镜像名称

例如

docker search zookeeper

docker search nginx

docker search tomcat

docker search redis

### 使用docker安装镜像：

docker pull 镜像名称

例如：

docker pull mysql

docker pull nginx

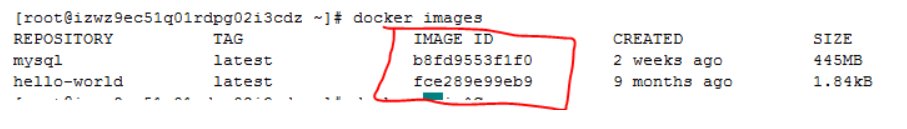
docker pull redis

### 查看当前系统docker中已经安装的镜像：

docker images

### 删除docker中的安装镜像

docker rmi 镜像id



### 运行docker容器中指定的镜像：

docker run --name container-name -d images-name[:tag]

说明：

--name：自定义容器名称参数

container-name：自定义的容器名称

-d:后台运行

images-name:指定镜像模板

[:tag]:可选参数，标签tag

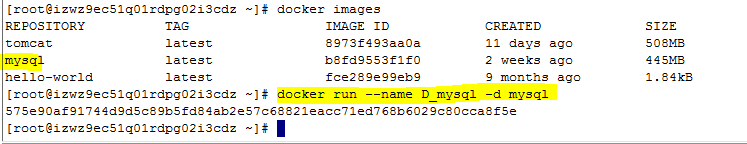
例如：

docker run --name myTomcat -d tomcat:8.0

docker run --name myTomcat -d tomcat:latest

docker run --name myTomcat -d tomcat

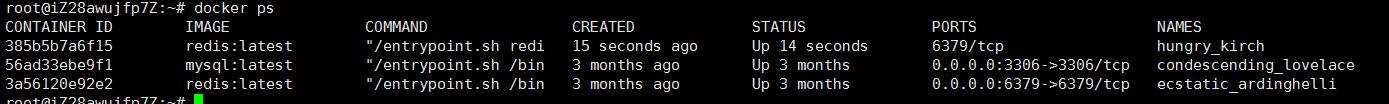
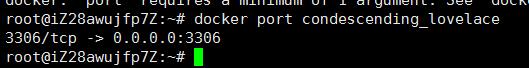
效果图：



### docker端口映射：

为什么要端口映射？  
​  
在启动容器时，如果不配置宿主机器与虚拟机的端口映射，外部程序是无法访问虚拟机的，因为没有端口。  
​  
端口映射的指令是什么？  
​  
docker指令：docker run  -p  ip:hostPort:containerPort  redis

例子：

启动容器并添加映射  
 docker run --privileged=true -d --name zookeeper --publish 2181:2181 -d zookeeper:latest  
​  
使用-p参数  会分配宿主机的端口映射到虚拟机。   
​  
IP表示主机的IP地址。  
​   
hostPort表示宿主机的端口。  
​   
containerPort表示虚拟机的端口。  
​  
支持的格式有三种：  
​  
ip:hostPort:containerPort：映射指定地址的指定端口到虚拟机的指定端口（不常用）   
​  
如：127.0.0.1:3306:3306，映射本机的3306端口到虚拟机的3306端口。   
​  
ip::containerPort：映射指定地址的任意端口到虚拟机的指定端口。（不常用）   
​  
如：127.0.0.1::3306，映射本机的3306端口到虚拟机的3306端口。   
​  
hostPort:containerPort：映射本机的指定端口到虚拟机的指定端口。（常用）   
​  
如：3306:3306，映射本机的3306端口到虚拟机的3306端口。  
​  
如何查看是否映射成功？  
​  
使用docker ps命令查看  
  
​  
出现6379/tcp的表示宿主机并没有打开与虚拟机的端口映射。   
​  
出现0.0.0.0:3306->3306/tcp表示宿主机的3306端口映射到了虚拟机的3306端口。  
​  
也可以使用docker port NAME查看端口映射情况。  
  
​  
  
表示虚拟机的3306端口映射到了主机的3306端口。   
如果没有任何回复，表示端口映射失败。

### 查看运行中的容器：

docker ps [参数]

说明：

参数：

-a :显示所有的容器，包括未运行的。

-f :根据条件过滤显示的内容。

--format :指定返回值的模板文件。

-l :显示最近创建的容器。

-n :列出最近创建的n个容器。

--no-trunc :不截断输出。

-q :静默模式，只显示容器编号。

-s :显示总的文件大小。

例如：

docker ps -a

### 9.停止运行中的容器:

docker stop 容器id或容器名称

例如：

docker stop myTomcat

### 10.启动容器中的镜像：

docker start 容器id或容器名称

例如：

docker start myTomcat

### 删除容器，前提是容器必须是停止状态：

docker rm 容器的id

### docker 中的Tomcat进行端口映射，设置后可访问Tomcat：

docker run -d -p 服务器端口:镜像端口 容器名称

例如：

docker中安装了Tomcat，默认是8080.

此时启动docker中的Tomcat并不能通过8080端口直接访问到docker中的Tomcat，

此时需要将docker中Tomcat的8080端口进行映射。

8080端口与服务器的端口进行映射。

-p：表示进行端口映射。

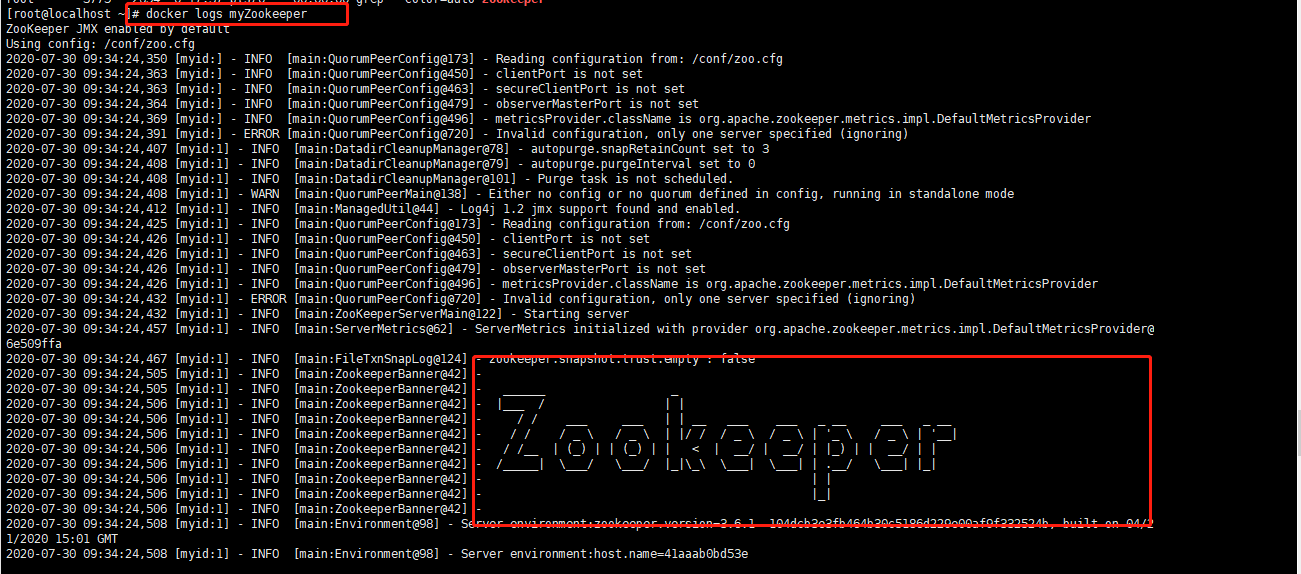
注意：服务器要开放8081端口。

docker run -d -p 8081:8080 tomcat

### 查看容器日志：

docker logs 容器id

效果图：



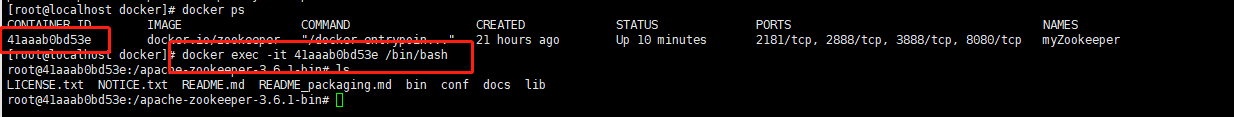
### **14进入容器，并修改其配置 ：**

docker exec -it 容器ID /bin/bash

示例：

 docker exec -it 41aaab0bd53e /bin/bash

效果图



### 15.退出容器：

exit

# Docker中Zookeeper的使用

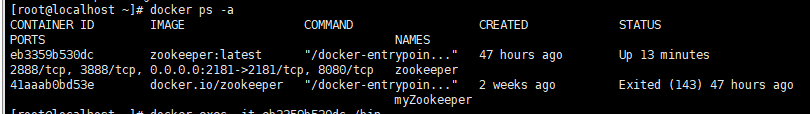
Docker安装Zookeeper

下载Zookeeper镜像  
docker pull zookeeper

启动容器并**添加映射（不添加映射无法使用容器）**docker run --privileged=true -d --name zookeeper --publish 2181:2181 -d zookeeper:latest

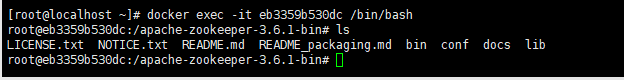
（莫名奇妙我为什么使用这句话就不会报错，而使用docker start myZookeeper这个就会报错。。可能是因为没添加映射吗）（主要还是我菜）

查看容器是否启动  
 1.docker ps



### 进入容器，并进行操作

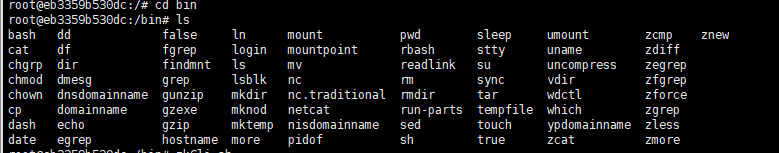
 2.docker exec -it 41aaab0bd53e /bin/bash



### 进入bin目录

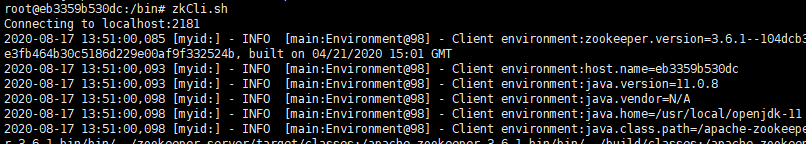
1. cd bin

ls 查看



### 启动zookeeper客户端

4.zkCli.sh



### 查看根节点

ls / （此时我已经开了一个8004的服务，所以里面有两个节点、如果没开则只有zookeeper节点）

1597672501(1)

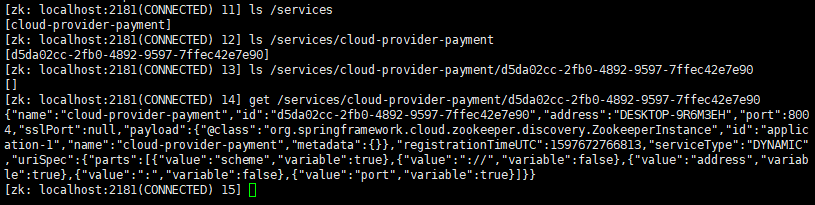
### 查看services节点

ls /services

colud-provider-payment 这就是我们的微服务

1597673659(1)

### 查看微服务信息



### 退出客户端

quit

### 退出容器

exit