1 .tos安装mysql

1.1下载rpm包 wget <http://repo.mysql.com/mysql-community-release-el6-5.noarch.rpm>

1.2安装rpm包 rpm -ivh mysql-community-release-el6-5.noarch.rpm

1.3通过yum命令安装mysql服务以及其依赖关系 yum install mysql-server 出现提示一直Y

1.4检测是否成功 进入etc/yum.repos.d文件多出 mysql-community-source.repo mysql-community.repo即为成功

1.5首次登录mysql mysql -u root回车显示 mysql> 即为成功

1.6修改mysql登录密码 set password=password('123456');

1.7开启远程连接 登录mysql

1.7.1 grant all privileges on \*.\* to 'root'@'%' identified by '123456' with grant option;

root是用户名，%代表任意主机，'123456'指定的登录密码（这个和本地的root密码可以设置不同的，互不影响）

1.7.2 flush privileges; # 重载系统权限

1.7.3 exit;

1.7.4 iptables -I INPUT -p tcp -m state --state NEW -m tcp --dport 3306 -j ACCEPT

查看规则是否生效

1.7.5 iptables -L -n

1.8关闭远程连接 iptables -D INPUT -p tcp -m state --state NEW -m tcp --dport 3306 -j ACCEPT

2.开放8080端口供本机访问

2.1 cd /etc/sysconfig/

2.2 vi iptables

2.3在基本相同的命令换行添加-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp --dport 8080 -j ACCEPT

2.4将iptables服务重启。 service iptables restart

（参考链接 <https://blog.csdn.net/yueaini10000/article/details/52918943>）

3. 安装nginx

3.1 nginx下载地址（http://nginx.org/download/）

3.2 nginx的编译需要c++，同时prce（重定向支持）和openssl（https支持）也需要安装。

yum install gcc-c++

yum -y install pcre\*

yum -y install openssl\*

3.3将压缩文件放在/usr/local下解压

3.4进入[root@admin local]# cd nginx-1.9.9

3.5设置安装目录 [root@admin nginx-1.9.9]# ./configure --prefix=/usr/local/nginx （直接打）

3.6编译安装

[root@admin nginx-1.9.9]# make

[root@admin nginx-1.9.9]# make install

3.7打开80端口防火墙

3.8启动服务

[root@admin ~]# cd /usr/local/nginx

[root@admin sbin]# ./nginx

3.9 ip+80访问出现（welcaome to nginx）成功

4启动多个tomcate

<http://www.360doc.com/content/15/1208/17/9552892_518792749.shtml>

第一个不用修改其余的service.xml全部修改

<Connector port="18009" protocol="AJP/1.3" redirectPort="18443" />

<Connector port="8081" protocol="HTTP/1.1"

connectionTimeout="20000"

redirectPort="18443" />

<Server port="18005" shutdown="SHUTDOWN">

基本为所有端口不要相同，上述案例在说有端口全加了1 如：8443->18443

5负载均衡ngnix配置文件

#user nobody;

worker\_processes 1;

#error\_log logs/error.log;

#error\_log logs/error.log notice;

#error\_log logs/error.log info;

#pid logs/nginx.pid;

events {

worker\_connections 1024;

}

http {

include mime.types;

default\_type application/octet-stream;

#log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '

# '$status $body\_bytes\_sent "$http\_referer" '

# '"$http\_user\_agent" "$http\_x\_forwarded\_for"';

#access\_log logs/access.log main;

sendfile on;

#tcp\_nopush on;

#keepalive\_timeout 0;

keepalive\_timeout 65;

#gzip on;

upstream tomcats{

server 127.0.0.1:8080 weight=1;

server 127.0.0.1:8081 weight=1;

}

server {

listen 80;

server\_name localhost;

#charset koi8-r;

#access\_log logs/host.access.log main;

location / {

root html;

index index.html index.htm;

proxy\_pass http://tomcats;

proxy\_redirect default;

}

#location /status{

# stub\_status on;

# access-log off;

#}

#error\_page 404 /404.html;

# redirect server error pages to the static page /50x.html

#

error\_page 500 502 503 504 /50x.html;

location = /50x.html {

root html;

}

# proxy the PHP scripts to Apache listening on 127.0.0.1:80

#

#location ~ \.php$ {

# proxy\_pass http://127.0.0.1;

#}

# pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000

#

#location ~ \.php$ {

# root html;

# fastcgi\_pass 127.0.0.1:9000;

# fastcgi\_index index.php;

# fastcgi\_param SCRIPT\_FILENAME /scripts$fastcgi\_script\_name;

# include fastcgi\_params;

#}

# deny access to .htaccess files, if Apache's document root

# concurs with nginx's one

#

#location ~ /\.ht {

# deny all;

#}

}

# another virtual host using mix of IP-, name-, and port-based configuration

#

#server {

# listen 8000;

# listen somename:8080;

# server\_name somename alias another.alias;

# location / {

# root html;

# index index.html index.htm;

# }

#}

# HTTPS server

#

#server {

# listen 443 ssl;

# server\_name localhost;

# ssl\_certificate cert.pem;

# ssl\_certificate\_key cert.key;

# ssl\_session\_cache shared:SSL:1m;

# ssl\_session\_timeout 5m;

# ssl\_ciphers HIGH:!aNULL:!MD5;

# ssl\_prefer\_server\_ciphers on;

# location / {

# root html;

# index index.html index.htm;

# }

#}

}

6打包springboot项目

6.1在pom.xml里设置

packaging>war</packaging>

6.2在pom.xml里找到spring-boot-starter-web依赖节点，在其中添加如下代码，

dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

<!-- 移除嵌入式tomcat插件 -->

<exclusions>

<exclusion>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-tomcat</artifactId>

</exclusion>

</exclusions>

</dependency>

6.3**加servlet-api的依赖** version信息改成对应的tomcat版本信息

dependency>

<groupId>org.apache.tomcat</groupId>

<artifactId>tomcat-servlet-api</artifactId>

<version>8.5.20</version>

<scope>provided</scope>

</dependency>

6.4修改启动类，修改启动类，并重写初始化方法

/\*\*

\* 修改启动类，继承 SpringBootServletInitializer 并重写 configure 方法

\*/

public class SpringBootStartApplication extends SpringBootServletInitializer {

public static void main( String[] args ){

SpringApplication.run(SpringBootStartApplication .class, args);

}

/\*\*

\*新增此方法

\*/

@Override

protected SpringApplicationBuilder configure(SpringApplicationBuilder builder) {

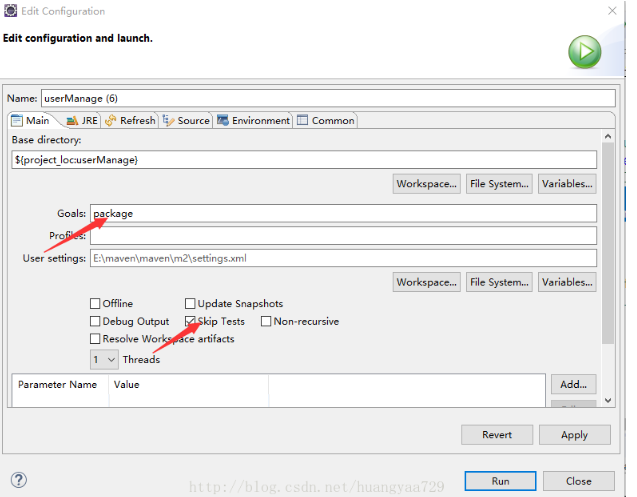
// 注意这里要指向原先用main方法执行的Application启动类

return builder.sources(Application.class);

}

}

6.5点击项目右键—> run as—>maven build：



注：访问时要添加项目名称 即war包的名字