尚硅谷电商数仓项目--实时计算

版本：V 1.6

张晨

# 实时需求概览

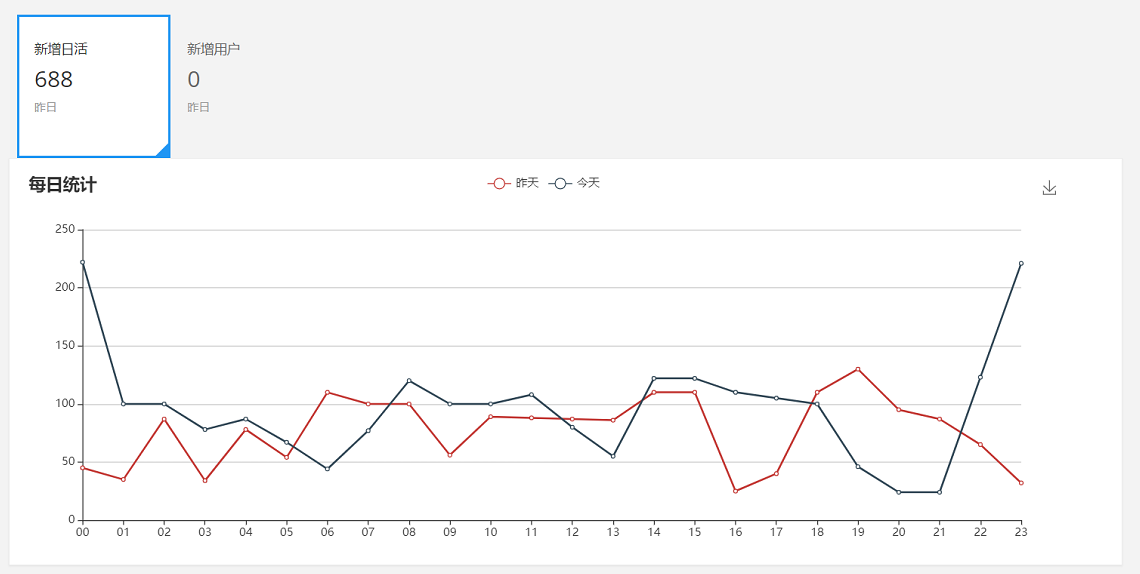
## 1 实时需求与离线需求的比较

**离线需求，**一般是根据前一日的数据生成报表，虽然统计指标、报表繁多，但是对时效性不敏感。

**实时需求，**主要侧重于对当日数据的实时监控，通常业务逻辑相对离线需求简单一下，统计指标也少一些，但是更注重数据的时效性，以及用户的交互性。

## 2 需求明细

### 2.1当日用户首次登录（日活）分时趋势图，昨日对比

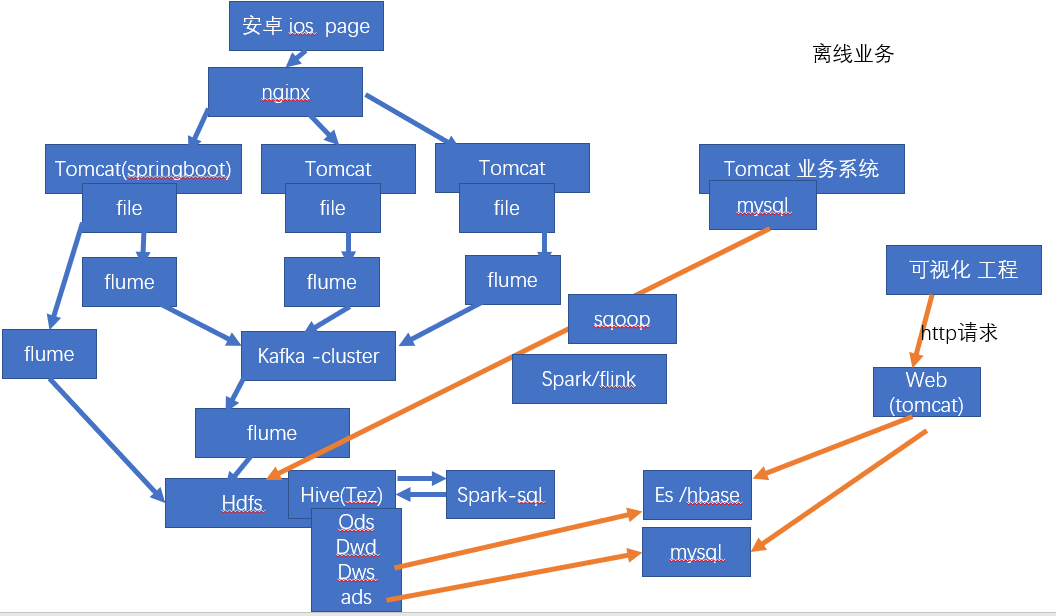


### 2.2当日首单交易额及分时趋势图，昨日对比

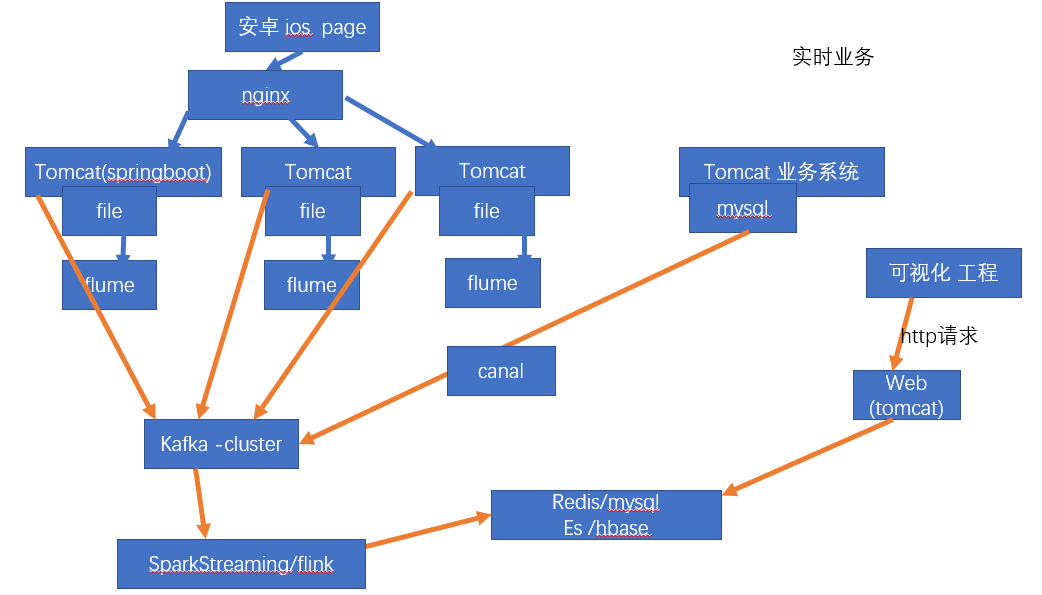
### 2.3活动相关统计

# 第二章 实时统计架构

## 1 离线：

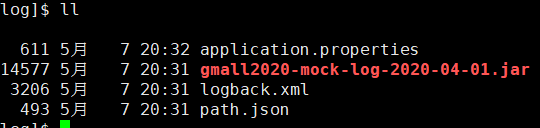


## 2 实时：

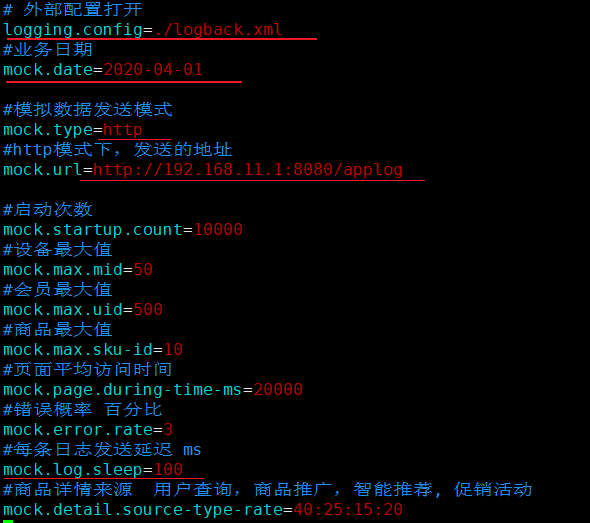


# 第三章 模拟日志生成器的使用

## 1 拷贝日志生成jar包到虚拟机的某个目录



## 2 修改application.properties



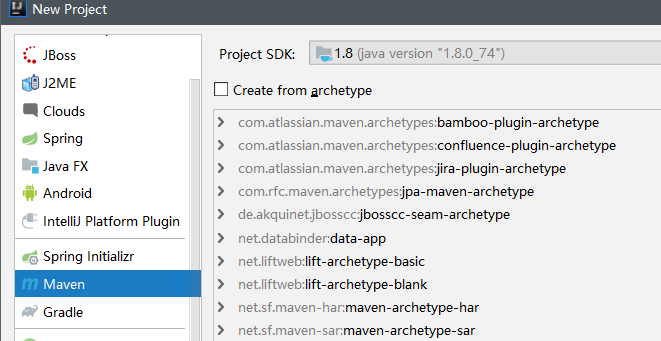
## 3、使用模拟日志生成器的jar 运行

|  |
| --- |
| java -jar gmall2020-mock-log-2020-04-01.jar |

# 第四章 父工程：gmall2020-parent

### 1 建立工程

(课堂中命名可能与课件中不一致)



### 2 pom.xml

|  |
| --- |
| *<?*xml version="1.0" encoding="UTF-8"*?>* <project xmlns="http://maven.apache.org/POM/4.0.0"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>   <groupId>com.atguigu.gmall2020</groupId>  <artifactId>gmall2020-parent</artifactId>  <version>1.0-SNAPSHOT</version>    </project> |

# 第五章 搭建日志采集系统的集群

## 1、子模块：日志采集模块 logger --（单机开发调试）

### 1.1 springboot简介

Spring Boot 是由 Pivotal 团队提供的全新框架，其设计目的是用来简化新 Spring 应用的初始搭建以及开发过程。 该框架使用了特定的方式来进行配置，从而使开发人员不再需要定义样板化的配置。

#### 1.1.1有了springboot 我们就可以…

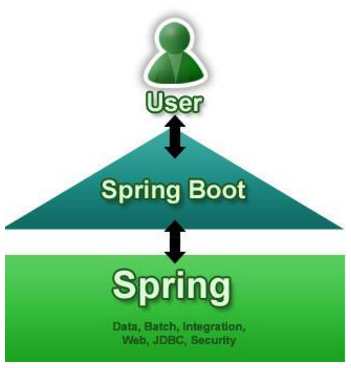
1 不再需要那些千篇一律，繁琐的xml文件。

2 内嵌Tomcat,不再需要外部的Tomcat

3 更方便的和各个第三方工具（mysql,redis,elasticsearch,dubbo,kafka等等整合），而只要维护一个配置文件即可。

#### 1.1.2 springboot和ssm的关系

springboot整合了springmvc ，spring等核心功能。也就是说本质上实现功能的还是原有的spring ,springmvc的包，但是springboot单独包装了一层，这样用户就不必直接对springmvc， spring等，在xml中配置。



#### 1.1.3 没有xml，我们要去哪配置

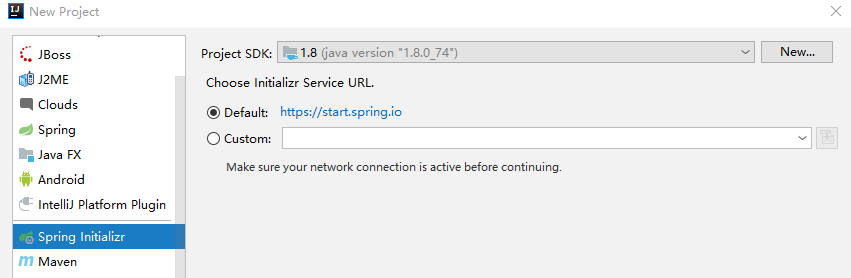
1) springboot实际上就是把以前需要用户手工配置的部分，全部作为默认项。除非用户需要额外更改不然不用配置。这就是所谓的：“约定大于配置”

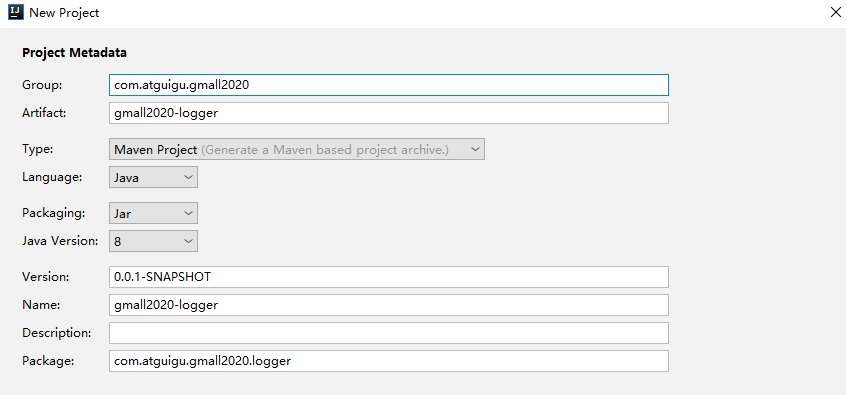
2) 如果需要特别配置的时候，去修改application.properties （application.yml）

### 1.2 快速搭建

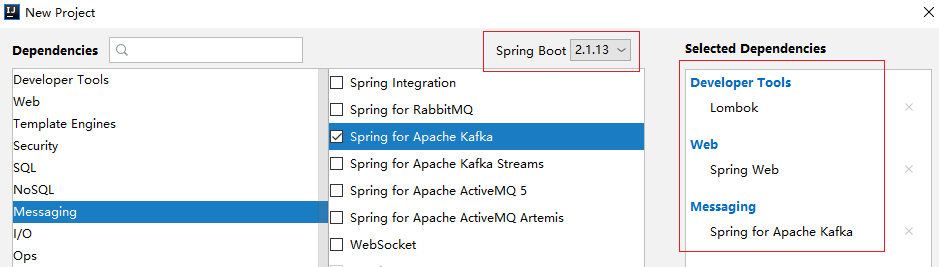
#### 1.2.1 新建module

在project下增加一个Module，选择Spring Initializr





目前企业中普遍选择2.1.3，不推荐选择2.2.x



#### 1.2.2 pom.xml

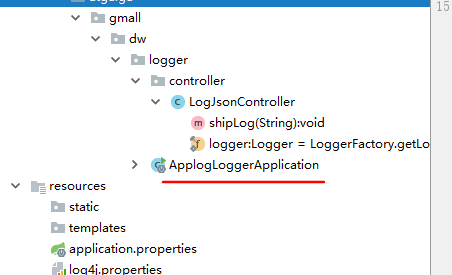
|  |
| --- |
| *<?*xml version="1.0" encoding="UTF-8"*?>* <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>  <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>2.1.13.RELEASE</version>  <relativePath/> *<!-- lookup parent from repository -->* </parent>  <groupId>com.atguigu.gmall2020</groupId>  <artifactId>gmall2020-logger</artifactId>  <version>0.0.1-SNAPSHOT</version>  <name>gmall2020-logger</name>  <description>Demo project for Spring Boot</description>   <properties>  <java.version>1.8</java.version>  </properties>   <dependencies>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.kafka</groupId>  <artifactId>spring-kafka</artifactId>  </dependency>   <dependency>  <groupId>org.projectlombok</groupId>  <artifactId>lombok</artifactId>  <optional>true</optional>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-test</artifactId>  <scope>test</scope>  </dependency>  <dependency>  <groupId>com.alibaba</groupId>  <artifactId>fastjson</artifactId>  <version>1.2.56</version>  </dependency>   <dependency>  <groupId>org.springframework.kafka</groupId>  <artifactId>spring-kafka-test</artifactId>  <scope>test</scope>  </dependency>  </dependencies>   <build>  <plugins>  <plugin>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-maven-plugin</artifactId>  </plugin>  </plugins>  </build>  </project> |

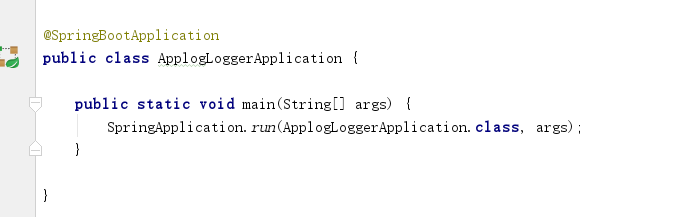
#### 1.2.3 controller

|  |
| --- |
| @Controller **public class** Demo1Controller {  @ResponseBody  @RequestMapping(**"testDemo"**)  **public** String testDemo(){  **return "hello demo"**;  } } |

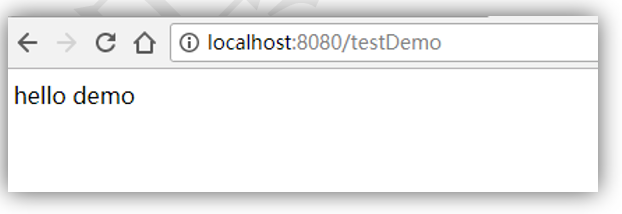
#### 1.2.4 用main方法启动tomcat

在程序中直接找到XXXXXApplication这个类



这个类只有一个main函数 ，直接执行就可以

#### 1.2.5 用浏览器测试



#### 1.2.6 修改端口号

在 resources目录下的application.properties 加入

|  |
| --- |
| server.port=80 |

### 1.3 springboot整合kafka

#### 1.3.1 application.propeties

|  |
| --- |
| *#============== kafka =================== # 指定kafka 代理地址，可以多个* spring.kafka.bootstrap-servers=hadoop1:9092   *# 指定消息key和消息体的编解码方式* spring.kafka.producer.key-serializer=org.apache.kafka.common.serialization.StringSerializer spring.kafka.producer.value-serializer=org.apache.kafka.common.serialization.StringSerializer |

#### 1.3.2 LogJsonController

|  |
| --- |
| **import** com.alibaba.fastjson.JSONObject; **import** lombok.extern.slf4j.Slf4j; **import** org.springframework.beans.factory.annotation.Autowired; **import** org.springframework.kafka.core.KafkaTemplate; **import** org.springframework.web.bind.annotation.RequestBody; **import** org.springframework.web.bind.annotation.RequestMapping; **import** org.springframework.web.bind.annotation.RequestParam; **import** org.springframework.web.bind.annotation.RestController;  @RestController @Slf4j **public class** LoggerController {   @Autowired  KafkaTemplate **kafkaTemplate**;    @RequestMapping(**"/applog"**)  **public** String applog(@RequestBody JSONObject jsonObject){  String logJson = jsonObject.toJSONString();  ***log***.info(logJson);  **if**(jsonObject.getString(**"start"**)!=**null**){  **kafkaTemplate**.send(**"GMALL\_START"**,logJson);  }**else**{  **kafkaTemplate**.send(**"GMALL\_EVENT"**,logJson);   }    **return "success"**;  }   } |

#### 1.3.3 logback.xml

|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**configuration**>  <**property name="LOG\_HOME" value="/applog/gmall2020"** />  <**appender name="console" class="ch.qos.logback.core.ConsoleAppender"**>  <**encoder**>  <**pattern**>%msg%n</**pattern**>  </**encoder**>  </**appender**>   <**appender name="rollingFile" class="ch.qos.logback.core.rolling.RollingFileAppender"**>  <**file**>${LOG\_HOME}/app.log</**file**>  <**rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy"**>  <**fileNamePattern**>${LOG\_HOME}/app.%d{yyyy-MM-dd}.log</**fileNamePattern**>  </**rollingPolicy**>  <**encoder**>  <**pattern**>%msg%n</**pattern**>  </**encoder**>  </**appender**>    *<!-- 将某一个包下日志单独打印日志 -->* <**logger name="com.atguigu.gmall2020.logger.controller.LoggerController"  level="INFO" additivity="false"**>  <**appender-ref ref="rollingFile"** />  <**appender-ref ref="console"** />  </**logger**>   <**root level="error" additivity="false"**>  <**appender-ref ref="console"** />   </**root**> </**configuration**> |

#### 1.3.4创建对应的topic

|  |
| --- |
| kafka-topics.sh --create --topic GMALL\_START --zookeeper hadoop1:2181,hadoop2:2181,hadoop3:2181 --partitions 4 --replication-factor 1 |

#### 1.3.5 启动LoggerApplication ， 启动日志Mock的生成类

用kafka进行测试

|  |
| --- |
| /bigdata/kafka\_2.11-0.11.0.2/bin/kafka-console-consumer.sh --bootstrap-server hadoop1:9092,hadoop2:9092,hadoop3:9092 --topic **GMALL\_START** |

## 2、 日志采集模块打包部署--- (部署到服务器)

### 2.1 修改logback中的配置文件

|  |
| --- |
| *<?*xml version="1.0" encoding="UTF-8"*?>* <configuration>  <property name="LOG\_HOME" value="/opt/applog/gmall2020" />  <appender name="console" class="ch.qos.logback.core.ConsoleAppender">  <encoder>  <pattern>%msg%n</pattern>  </encoder>  </appender>   <appender name="rollingFile" class="ch.qos.logback.core.rolling.RollingFileAppender">  <file>${LOG\_HOME}/app.log</file>  <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">  <fileNamePattern>${LOG\_HOME}/app.%d{yyyy-MM-dd}.log</fileNamePattern>  </rollingPolicy>  <encoder>  <pattern>%msg%n</pattern>  </encoder>  </appender>    *<!-- 将某一个包下日志单独打印日志 -->* <logger name="com.atgugu.gmall2020.mock.log.Mocker"  level="INFO" additivity="true">  <appender-ref ref="rollingFile" />  <appender-ref ref="console" />  </logger>   <root level="error" additivity="true">  <appender-ref ref="console" />   </root> </configuration> |

### 2.2 把打好的jar包拷贝到Linux 路径下

### 2.3 启动jar包

|  |
| --- |
| java -jar /app/gmall2019/dw-logger-0.0.1-SNAPSHOT.jar |

如果出现权限问题，是因为Linux默认不允许非root账号使用1024以下的端口，所以改换为8080端口

|  |
| --- |
| java -jar /app/gmall2019/dw-logger-0.0.1-SNAPSHOT.jar –server.port=8080 >/dev/null 2>&1 & |

### 2.4 再次测试kafka消费

|  |
| --- |
| /bigdata/kafka\_2.11-0.11.0.2/bin/kafka-console-consumer.sh --bootstrap-server hadoop1:9092,hadoop2:9092,hadoop3:9092 --topic **GMALL\_STARTUP** |

## 3 搭建日志采集集群---(集群部署)

### 3.1 Nginx 入门

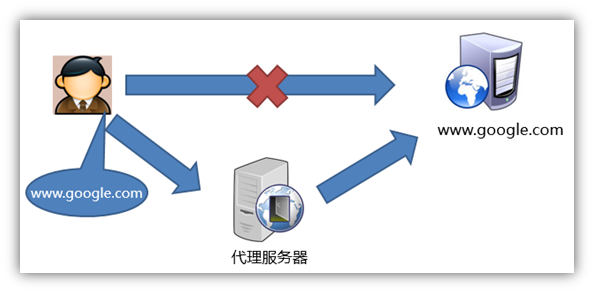
#### 3.1.1简介

*Nginx* ("engine x") 是一个**高性能的HTTP和反向代理服务器**,特点是占有内存少，并发能力强，事实上nginx的并发能力确实在同类型的网页服务器中表现较好，中国大陆使用nginx网站用户有：百度、京东、新浪、网易、腾讯、淘宝等。

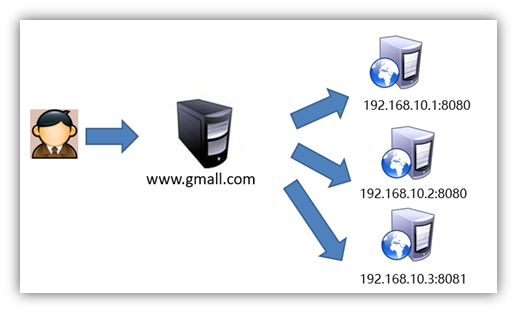
#### 3.1.2 Nginx 功能

1. 反向代理

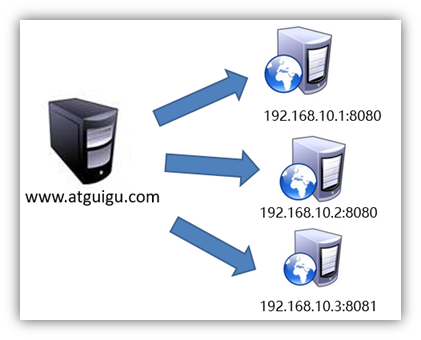
什么是反向代理？先看什么是正向代理



再看什么是反向代理



1. 负载均衡

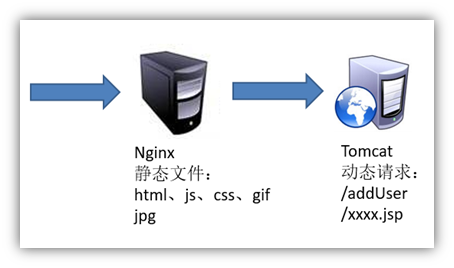


负载均衡策略： 轮询

权重

备机

1. 动静分离



#### 3.1.3 安装

##### 1） yum安装依赖包

|  |
| --- |
| sudo yum -y install openssl openssl-devel pcre pcre-devel zlib zlib-devel gcc gcc-c++ |

##### 2） 安装依赖包

|  |
| --- |
| 解压缩nginx-xx.tar.gz包。  进入解压缩目录，执行  ./configure --prefix=/opt/module/nginx  make && make install |

--prefix=要安装到的目录

##### 3） 启动、关闭命令nginx

|  |
| --- |
| 启动命令: 在/usr/local/nginx/sbin目录下执行 ./nginx  关闭命令: 在/usr/local/nginx/sbin目录下执行 ./nginx -s stop  重新加载命令: 在/usr/local/nginx/sbin目录下执行 ./nginx -s reload |

**如果启动时报错：**

|  |
| --- |
| ln -s /usr/local/lib/libpcre.so.1 /lib64 |

#### 3.1.4 赋权限

nginx占用80端口，默认情况下非root用户不允许使用1024以下端口

|  |
| --- |
| sudo setcap cap\_net\_bind\_service=+eip /bigdata/nginx/sbin/nginx |

#### 3.1.5 修改/bigdata/nginx/conf/nginx.conf

|  |
| --- |
| http{  ..........  upstream logserver{  server hadoop1:8080 weight=1;  server hadoop2:8080 weight=1;  server hadoop3:8080 weight=1;    }  server {  listen 80;  server\_name logserver;    location / {  root html;  index index.html index.htm;  proxy\_pass http://logserver;  proxy\_connect\_timeout 10;    }  ..........  } |

### 3.2 集群脚本

|  |
| --- |
| #!/bin/bash  JAVA\_BIN=/bigdata/jdk1.8.0\_152/bin/java  PROJECT=gmall2019  APPNAME=xxxxx.jar  SERVER\_PORT=8080    case $1 in  "start")  {    for i in hadoop1 hadoop2 hadoop3  do  echo "========: $i==============="  ssh $i "$JAVA\_BIN -Xms32m -Xmx64m -jar /applog/$PROJECT/$APPNAME --server.port=$SERVER\_PORT >/dev/null 2>&1 &"  done  echo "========NGINX==============="  /usr/local/nginx/sbin/nginx  };;  "stop")  {  echo "======== NGINX==============="  /usr/local/nginx/sbin/nginx -s stop  for i in hadoop1 hadoop2 hadoop3  do  echo "========: $i==============="  ssh $i "ps -ef|grep $APPNAME |grep -v grep|awk '{print \$2}'|xargs kill" >/dev/null 2>&1  done    };;  esac |