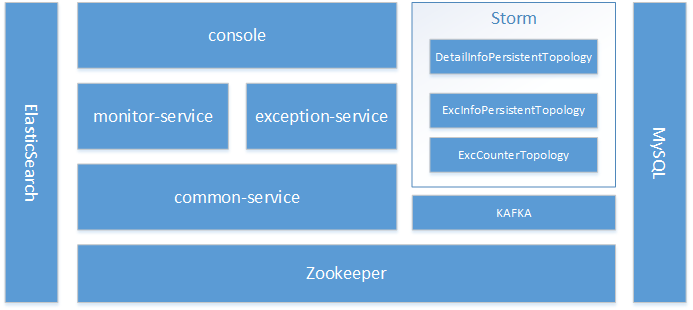
# 前言

IT监控平台主要用于向IT技术人员提供应用监控、异常监控等功能的平台，即监控其他系统的系统。通过探针，对已有java项目进行简单配置，即能够介入监控平台，对应用性能指标、异常情况进行全方面的监控，主要包括：CPU使用率、吞吐率、错误率、响应时间、线程使用、GC、SQL耗时等。

项目使用Maven 3作为项目管理工具，GIT作为版本管理工具，源码及编译基于Oracle JDK 1.8。

应用架构如下：



* zookeeper：根据微服务的思想，依赖zookeeper实现服务注册/发现功能，实现各个模块间的有效解耦。
* Kafka：监控数据允许少量的丢失，采用Kafka作为异步传输工具。
* ElasticSearch：由于监控数据庞大，一般的关系型数据库不足以满足要求，使用ElasticSearch用于数据存储；
* MqSQL：用户存储基本的字典数据及统计结果；
* storm：根据监控数据的情况，采用实时的流失计算方式对数据进行统计；
* common-service：通用服务模块，用于为其他模块提供web服务，比如报警、通用参数获取等；
* exception-service：异常数据服务模块，用于提供异常（警告、错误）信息相关服务；
* monitor-service：监控服务模块，用于提供监控相关服务；
* console：控制台模块，提供前端界面、权限控制、以及与其他服务模块打交互。

# 基础配置

## 数据表创建

执行docs/sql/table.2016.sql创建项目所用基础表。

执行docs/sql/resource-data.sql进行基础数据的初始化。

## 分环境配置

IT监控平台根据实际项目考虑，分为开发、测试、预生产、生产四个不同环境，记住Maven的profile方式，对四种不同的环境进行配置。四种配置环境分别位于src/main目录中的resources-1-development、resources-2-integration、resources-3-staging、resources-4-production，需要根据实际情况进行配置。

### common模块

IT监控平台是一个有机的整体，大多数公用配置放置在common模块中，减少配置变化需要改动多处配置项，能够有效的避免因为人为失误造成的配置错误。

* dataSource.properties：项目中采用阿里开源组件druid数据源连接池，该文件主要用于提供mysql数据库基础配置。需要注意的是，在monitor-service、console中所使用的数据源配置也是使用该文件。
* it-monitor-common.properties：由于IT监控配置是一个有机的整体，所以通用配置放置在该文件中。由于配置项大多输入固定数据，需要注意的有以下几项：
* zk.address：zookeeper地址，用于服务注册、服务发现的基础zookeeper地址
* zk.namespace：在zookeeper中的通用命名空间，可以将IT监控平台所有使用zookeeper的znode节点约束在同一个命名空间中，不会与其他项目发生冲突。
* omni.monitor.zk.address：IT监控平台对其他项目的监控采用zookeeper中的znode节点作为注册依据，由于监控数据请求资源比较大，所以与业务所需的zookeeper地址进行分离，也可以与zk.address使用相同的zookeeper集群环境。
* omni.monitor.zk.root： IT监控平台对其他项目的监控采用zookeeper中的znode节点作为注册依据，各个接入IT监控平台的项目创建各自的znode节点，统一存储在omni.monitor.zk.root定义的路径下。
* es.nodes：前文提到过，监控数据存储在elasticsearch中，该配置项即为存储数据的elasticsearch集群地址
* es.clusterName：根据需要，使用elasticsearch集群需项目提供elasticsearch集群名称，该配置项即为其集群名称

### stash模块

该模块是单独用于storm中topology的模块，配置项主要用于topology的正常运行所用。由于topology中读取配置的方式，所以部分基础配置与common模块中的配置重复。

* hikari.properties：在storm使用hikari作为数据源组件，其性能更优于druid，且没有不必要的监控功能。该文件中的配置项主要用户提供mysql基础配置。配置内容与common模块中dataSource.properties文件内容一致
* stash-common.properties：IT监控平台中有3个有效topology，该配置文件主要提供公用的配置项，需要注意的主要有下面几项：
* topology.brokerZkStr：topology需要与kafka交互，根据kafka的客户端要求，需要提供kafka依赖的zookeeper集群地址；
* topology.brokerZkPath：kafka消费消息是需要提供broker在zookeeper注册znode地址；
* topology.brokerList：该配置提供kafka的broker服务列表；
* es.transport.address：该配置项提供存储监控数据、监控统计结果的elasticsearch集群地址，配置内容与common模块中it-monitor-common.properties文件的es.nodes一致；
* es.cluster.name：该配置项提供elasticsearch集群的集群名，配置内容与common模块中it-monitor-common.properties文件的es.clusterName一致；

## elasticsearch中mapping定义

由于IT监控平台对elasticsearch中存储的数据结构、类型有特殊要求，所以需要在elasticsearch中预先定义mapping格式。配置内容在docs/mappings中，分为exc-data.json、monitor-data.json，分别对应异常信息数据、监控信息数据。定义方式是以POST的方式请求elasticsearch地址http://{{host}}:{{port}}/{{indexName}}，其中host为elasticsearch集群地址，port为服务端口（默认是9200），indexName为索引名称，即文件名称，POST方式的请求体数据为文件内容。

## 服务注册/发现

由于IT监控平台采用服务注册/发现的方式有机的结合在一起，需要在应用启动前配置所在服务器的IP地址，通过环境变量方式注入。需要在系统中提前配置环境变量：LOCAL\_IP\_ADDRESS，值为可以被外部访问的IP地址。如果在单机中部署所有实例，可以将值设置为127.0.0.1，如：export LOCAL\_IP\_ADDRESS=127.0.0.1。

# 部署

在本章节中主要介绍IT监控平台及依赖环境的配置，根据依赖关系，建议按照顺序安装并启动。

## JDK环境

* 版本：JDK8，建议使用oracle官网中提供的jdk8中最新的更新
* 要求：环境变量JAVA\_HOME、CLASSPATH，在PATH中引入${JAVA\_HOME}/bin。

## zookeeper集群

* 版本：3.4.9
* 要求：由于zookeeper集群用于提供众多服务的监控配置，需要集群吞吐量、性能较优，所以需要根据需要调整集群大小。

## kafka集群

* 版本：kafka\_2.10-0.8.2.2
* 要求：由于kafka的版本间差异较大，建议版本按照要求使用，避免因为kafka版本间差异、API变化、客户端差异引起的未知异常

## elasticsearch集群

* 版本：2.4.1
* 要求：可以使用2.2、2.3、2.4几个版本，由于项目中需要用到groovy脚本功能，所以需要在elasticsearch集群配置过程中打开groovy脚本可运行开关。

## strom集群

* 版本：0.10.0
* 要求：storm版本间存在较大差异，建议使用0.10.0。

## 应用打包

使用maven打包，由于有不同的环境，需要在打包是指定所用环境。其中common-service、monitor-servie、exception-service、console四个模块结果为war包，stash模块结果为jar包。以测试环境（integration）为例,命令如下：

* common-service模块：maven clean package -pl common-service -am -rf common -P common,integration -Dmaven.test.skip=true
* monitor-service模块：maven clean package -pl monitor-service -am -rf common -P common,integration -Dmaven.test.skip=true
* exception-service模块：maven clean package -pl exception-service -am -rf common -P common,integration -Dmaven.test.skip=true
* console模块：maven clean package -P common,integration -Dmaven.test.skip=true
* stash模块中的topology：maven clean package -pl stash -am -rf common -P common,integration -Dmaven.test.skip=true

# [web应用接入监控](http://172.17.254.246:8090/pages/viewpage.action?pageId=5965427)

该章节主要介绍web应用项目如何接入IT监控平台，前文已经提过，监控方式使用探针，使用极少配置即能接入监控，而且对本应用影响极小。

## 依赖jar包

在WEB-INF/lib目录中添加依赖jar包：

* 监控依赖：servlet-monitor-1.3.4.jar
* 基础依赖：commons-beanutils-1.8.3.jar、jackson-mapper-asl-1.9.13.jar、snappy-java-1.1.1.7.jar、commons-lang-2.6.jar、jline-0.9.94.jar、lz4-1.2.0.jar、zookeeper-3.4.6.jar、jackson-core-asl-1.9.13.jar、kafka-clients-0.8.2.2.jar
* 日志依赖：jcl-over-slf4j-1.7.12.jar、logback-classic-1.1.3.jar、logback-core-1.1.3.jar、slf4j-api-1.7.12.jar

## 定义filter

在web.xml中增加filter配置：

<filter>

<filter-name>servlet-monitor</filter-name>

<filter-class>com.wfj.netty.servlet.filter.Monitor</filter-class>

</filter>

<filter-mapping>

<filter-name>servlet-monitor</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

## 配置文件wfj-netty.properties

在classes目录中增加监控配置文件，无该文件启动异常，文件名为wfj-netty.properties：

# 系统contextRoot路径，与定义名相同

system.setting.context-root=/ocm-info-webin

# 系统contextName，与定义名相同

system.setting.context-name=ocm-info-webin

# 系统码，由监控平台分配

system.setting.context-code=76

# 自定义描述内容

system.setting.context-desc='{'"version"\:"1.2","name"\:"wfj-omni-channel-ocm-info-webin","desc":"ocm-info-webin","status"\:"{0}"}

# 是否调试模式，true为调试模式，不进行监控

system.seeting.monitor.isDebug=false

# 服务监控地址，随环境不同，配置不同

system.seeting.monitor.url=10.6.2.49:2181

system.seeting.monitor.timeout=5000

# sql监控地址，随环境不同，配置不同

system.setting.kafka.bootstrap.servers=10.6.2.56:9092,10.6.2.57:9092,10.6.2.58:9092

system.setting.kafka.acks=1

system.setting.kafka.retries=0

system.setting.kafka.batch.size=16384

# sql监控写入位置，随环境不同，配置不同

system.setting.kafka.sql.topic=sql-info-test

## 数据源监控配置

目前仅支持C3P0、druid两种方式数据源，由于C3P0效率远低于druid，所推荐尽可能使用druid作为数据源。

### 第一种方式（推荐）

<bean id="springDataSourceBeanPostProcessor" class="com.wfj.netty.servlet.handler.spring.SpringDataSourceBeanPostProcessor"/>

### 第二种方式

<bean id="wrappedDataSource" class="com.wfj.netty.servlet.handler.factory.SpringDataSourceFactoryBean">

<property name="targetName" value="dataSource"/>

</bean>

其中dataSource的值为spring配置数据源的名称