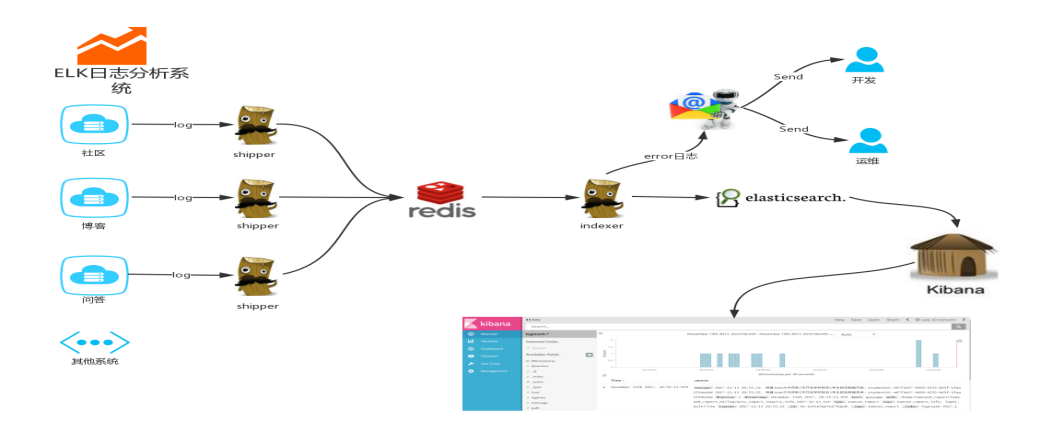
**统一Nlog采集日志格式**

分两大类：采集日志信息、采集指标信息

1. **采集日志信息**
2. 业务日志
3. Web业务日志
4. 业务日志
5. **采集指标信息**
6. 服务器指标
7. 消息队列指标
8. Redis数据库指标
9. Postgres指标

**日志采集框架图**



详解采集信息

环境要求 Dotnet core 3.0以上版本

1. **采集日志信息**

使用方法 NuGet包引用 Bkyl.Log

源码文件放置在tfs上

BKYL.ComputingProgram\_Core3.1/BKYL.Log

**介绍：**

解压后有两个项目一个控制台demo 还有一个web站demo

**使用：**

使用方法 NuGet包引用 Bkyl.Log(最新版)

**Web项目使用方法**

Program文件中CreateHostBuilder方法中添加 .UseNLog();管道中间件

Startup中配置扩展方法

//加载log中间件 内部有全局捕获异常建议放在注册捕获异常特性最后

services.AddWebLog();

同时将两个.config文件的属性配置成始终复制

只需要根据程序情况配置一下信息即可

<!--nodeName 请配置你的项目名字 例如web\_test\_log 这个名将会成为es索引前缀-->

<variable name="nodeName" value="web\_test\_log"/>

<!--tcpAddress 关于Web服务日志发送的logstash地址和端口 例如39.96.160.43:8102 -->

<variable name="webtcpAddress" value="39.96.160.43:8102"/>

<!--tcpAddress 预警发送的电子邮箱 -->

<variable name="notice" value="XXXXXXXXX@xx.com"/>

<!--tcpAddress 环境变量 -->

<variable name="environment" value="dev"/>

**控制台项目使用方法**

//加载配置文件

Bkyl.Log.LogExtension.AddProgramLog();

同时将.config文件的属性配置成始终复制

<!--nodeName 请配置你的项目名字 例如program\_test\_log 这个名将会成为es索引前缀-->

<variable name="nodeName" value="program\_test\_log"/>

<!--tcpAddress 关于程序日志发送的logstash地址和端口 例如39.96.160.43:8103 -->

<variable name="programtcpAddress" value="39.96.160.43:8103"/>

<!--tcpAddress 预警发送的电子邮箱 -->

<variable name="notice" value="XXXXXXX@xx.com"/>

<!--tcpAddress 环境变量 -->

<variable name="environment" value="dev"/>

**业务日志**

**以下接口都继承Bkyl.Log.IBaseGateway接口**

/// <summary>

/// Info级别日志

/// </summary>

/// <param name="message">记录的消息</param>

/// <param name="is\_warning">是否预警</param>

/// <param name="is\_console">是否打印控制台</param>

void Info(string message, bool is\_warning = false, bool is\_console = false);

/// <summary>

/// Info级别日志

/// </summary>

/// <param name="format">包含格式项的字符串</param>

/// <param name="is\_warning">是否预警</param>

/// <param name="is\_console">是否打印控制台</param>

/// <param name="args">要格式化的参数</param>

void Info(string format, bool is\_warning = false, bool is\_console = false, params object[] args);

/// <summary>

/// Info级别日志

/// </summary>

/// <param name="message">日志信息</param>

/// <param name="exception">异常对象</param>

/// <param name="is\_warning">是否预警</param>

/// <param name="is\_console">是否打印控制台</param>

void Info(string message, Exception exception, bool is\_warning = false, bool is\_console = false);

/// <summary>

/// Debug级别日志

/// </summary>

/// <param name="message">记录的消息</param>

/// <param name="is\_warning">是否预警</param>

/// <param name="is\_console">是否打印控制台</param>

void Debug(string message, bool is\_warning = false, bool is\_console = false);

/// <summary>

/// Debug级别日志

/// </summary>

/// <param name="format">包含格式项的字符串</param>

/// <param name="is\_warning">是否预警</param>

/// <param name="is\_console">是否打印控制台</param>

/// <param name="args">要格式化的参数</param>

void Debug(string format, bool is\_warning = false, bool is\_console = false, params object[] args);

/// <summary>

/// Debug级别日志

/// </summary>

/// <param name="message">日志信息</param>

/// <param name="exception">异常对象</param>

/// <param name="is\_warning">是否预警</param>

/// <param name="is\_console">是否打印控制台</param>

void Debug(string message, Exception exception, bool is\_warning = false, bool is\_console = false);

/// <summary>

/// Warn级别日志

/// </summary>

/// <param name="message">记录的消息</param>

/// <param name="is\_warning">是否预警</param>

/// <param name="is\_console">是否打印控制台</param>

void Warn(string message, bool is\_warning = false, bool is\_console = false);

/// <summary>

/// Warn级别日志

/// </summary>

/// <param name="format">包含格式项的字符串</param>

/// <param name="is\_warning">是否预警</param>

/// <param name="is\_console">是否打印控制台</param>

/// <param name="args">要格式化的参数</param>

void Warn(string format, bool is\_warning = false, bool is\_console = false, params object[] args);

/// <summary>

/// Warn级别日志

/// </summary>

/// <param name="message">日志信息</param>

/// <param name="exception">异常对象</param>

/// <param name="is\_warning">是否预警</param>

/// <param name="is\_console">是否打印控制台</param>

void Warn(string message, Exception exception, bool is\_warning = false, bool is\_console = false);

/// <summary>

/// Error级别日志

/// </summary>

/// <param name="message">记录的消息</param>

/// <param name="is\_warning">是否预警</param>

/// <param name="is\_console">是否打印控制台</param>

void Error(string message, bool is\_warning = false, bool is\_console = false);

/// <summary>

/// Error级别日志

/// </summary>

/// <param name="format">包含格式项的字符串</param>

/// <param name="is\_warning">是否预警</param>

/// <param name="is\_console">是否打印控制台</param>

/// <param name="args">要格式化的参数</param>

void Error(string format, bool is\_warning = false, bool is\_console = false, params object[] args);

/// <summary>

/// Error级别日志

/// </summary>

/// <param name="message">日志信息</param>

/// <param name="exception">异常对象</param>

/// <param name="is\_warning">是否预警</param>

/// <param name="is\_console">是否打印控制台</param>

void Error(string message, Exception exception, bool is\_warning = false, bool is\_console = false);

/// <summary>

/// Fatal级别日志

/// </summary>

/// <param name="message">记录的消息</param>

/// <param name="is\_warning">是否预警</param>

/// <param name="is\_console">是否打印控制台</param>

void Fatal(string message, bool is\_warning = false, bool is\_console = false);

/// <summary>

/// Fatal级别日志

/// </summary>

/// <param name="format">包含格式项的字符串</param>

/// <param name="is\_warning">是否预警</param>

/// <param name="is\_console">是否打印控制台</param>

/// <param name="args">要格式化的参数</param>

void Fatal(string format, bool is\_warning = false, bool is\_console = false, params object[] args);

/// <summary>

/// Fatal级别日志

/// </summary>

/// <param name="message">日志信息</param>

/// <param name="exception">异常对象</param>

/// <param name="is\_warning">是否预警</param>

/// <param name="is\_console">是否打印控制台</param>

void Fatal(string message , Exception exception, bool is\_warning = false, bool is\_console = false);

1. **Web业务日志 【使用Bkyl.Log.IWebGateway接口】**

**使用**:\_log.Info(描述，异常对象，是否预警，是否打印控制台)

**格式**：时间#项目名称#环境变量#服务器名称#进程id#日志等级#记录器#请求url#控制器#请求动作#http请求类型#客户端ip#请求body#请求quwey#http状态码#记录的消息#堆栈报错信息#

例：

2020-08-06 11:17:52.8686#服务名称web\_test\_log#计算机主机名称Host#进程10001#环境变量dev#日志级别INFO#记录器Bkyl.Log.LogGateway.Imp.WebGateway#预警邮箱793087382@qq.com#请求URLhttp://localhost/api/log/web#控制器Test#方法WebLogTest#Http请求类型GET#Http请求Ip39.96.160.43#Http请求Body{a:"a"}#请求Querya=a#a=a#响应状态码200#日志信息我是info级别日志#0#报错的堆栈信息#

1. **业务日志【使用Bkyl.Log.ILogGateway接口】**

**使用:**\_log.Info(描述，异常对象，是否预警，是否打印控制台)

**格式：**时间#项目名称#服务器名称#进程id#环境变量#日志等级#记录器#预警邮箱#记录的消息#是否预警#堆栈报错信息#

**关于日志格式注意事项：**

1. **时间必须是yyyy-MM-dd HH:mm:ss ffff**
2. **日志等级必须【INFO、ERROR、DEUBG、WARN、FATAL】**
3. **其他参数统一当作字符串处理**
4. **严格按照#号分隔。不能为空值，空置请用\_代替**
5. **日志记录必须是一行，日志按一行一行读取匹配，需要过滤掉\r \n**

**记录器：当时调用log方法时的对口对象**

详细使用日志方法和注入方法，和组件的引用请参考两个Demo文件

日志将发送到es中存储登录<http://62.234.155.90:5601/>

kibana按照定义的项目名称进行检索 即可查到

1. **采集指标信息**
2. 服务器指标
3. 消息队列指标
4. Redis数据库指标
5. Postgres指标
6. **服务器指标（定时获取）**

Json:Cpu利用率、内存使用率、磁盘使用率

**格式：**时间#服务名称#节点名称#Json数据#预警邮箱#是否预警#预警信息#

1. **消息队列指标（定时获取）**

Json:任务堆积数量

**格式：**时间#服务名称#节点名称#Json数据#预警邮箱#是否预警#预警信息#

1. **Redis数据库指标（定时获取）**

Json:内存使用大小、总内存大小、Key键数量、客户端连接数

**格式：**时间#服务名称#节点名称#Json数据#预警邮箱#是否预警#预警信息#

1. **Postgres数据库指标（定时获取）**

Json:日志占用空间，数据占用空间，

**格式：**时间#服务名称#节点名称#Json数据#预警邮箱#是否预警#预警信息#

**开发人员无需关系具体格式情况，只需要调用log封装的方法，底层将会自动封装日志格式存储**