# 调用链系统架构设计与实现

**课程概要：**

1. 整体架构
2. 插桩机制设计
3. 埋点与采集机制设计

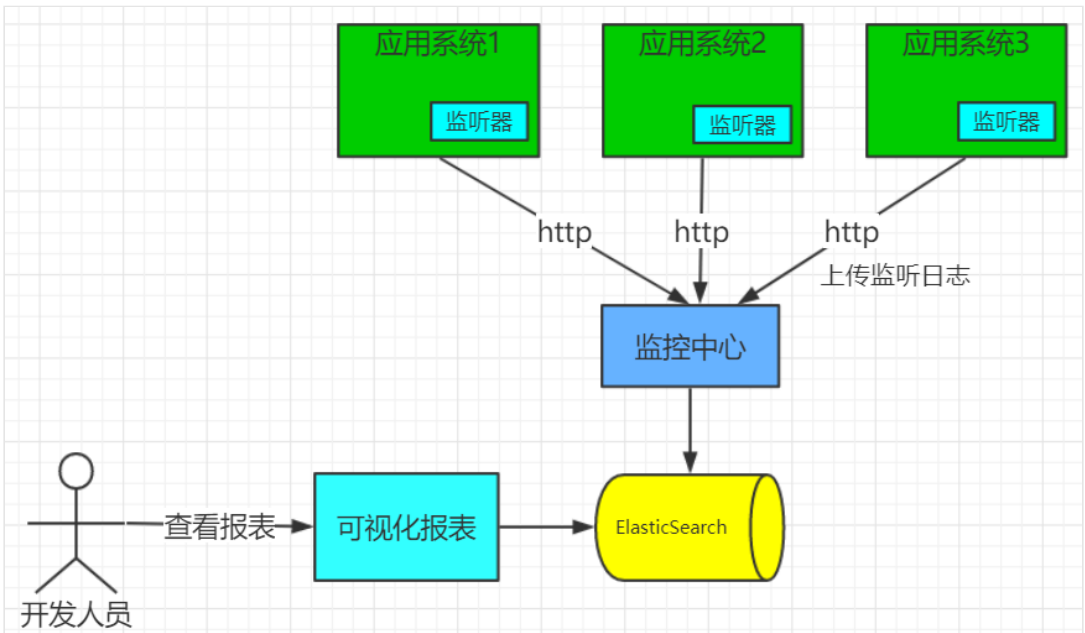
## 一、整体架构

---

**架构目标：**

1. **基本功能：**实现监控数据采集上报与展示
2. **可扩展性：**能够非常轻松灵活的添加新的采集器
3. **可配置**：可灵活配置
4. **可运维：**Agent自动更新、异常日志输出

### 1、基本功能实现架构

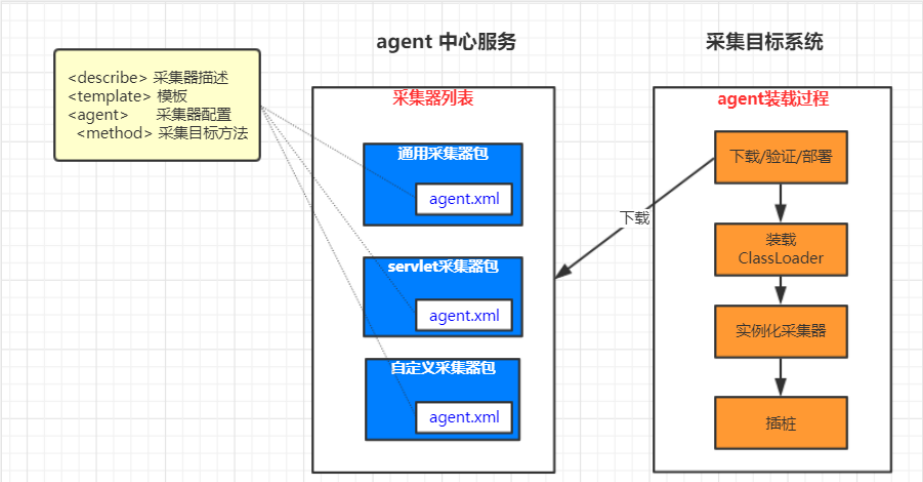



全局架构说明：

1. 监听器基于埋点采集，接口、SQL、Servlet等响应信息
2. 采集的数据基于后台线程 以Http的方式发送至监控中心
3. 监控中心实时上传至ElasticSearch
4. 可视化展示调用链详情

**2、可扩展性：**

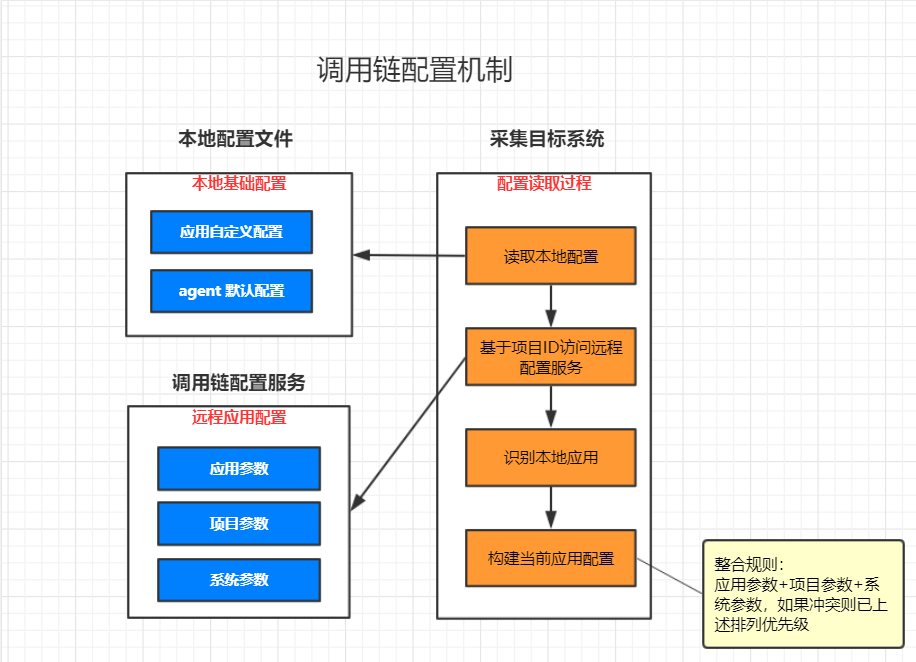
插件机制、灵活自定义扩展，其过程见下图。




☐ 演示已构建好的项目包

3、**可配置性：**

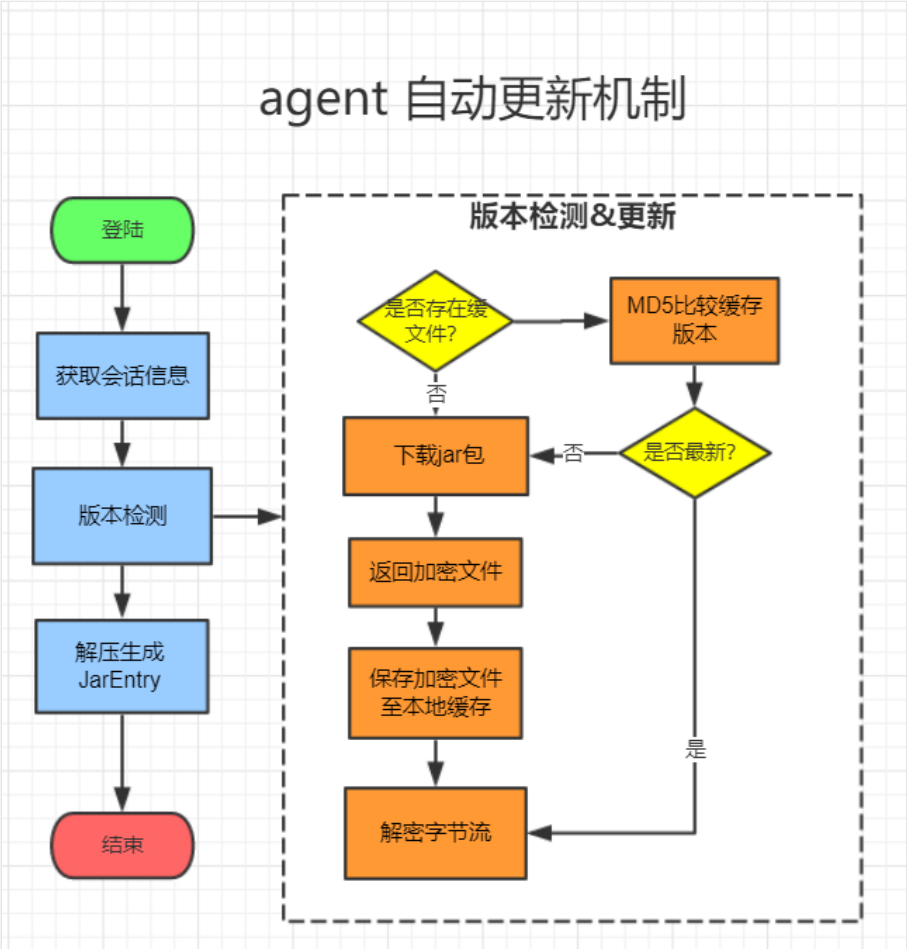
灵活的配置方式：




☐ 自动化配置演示

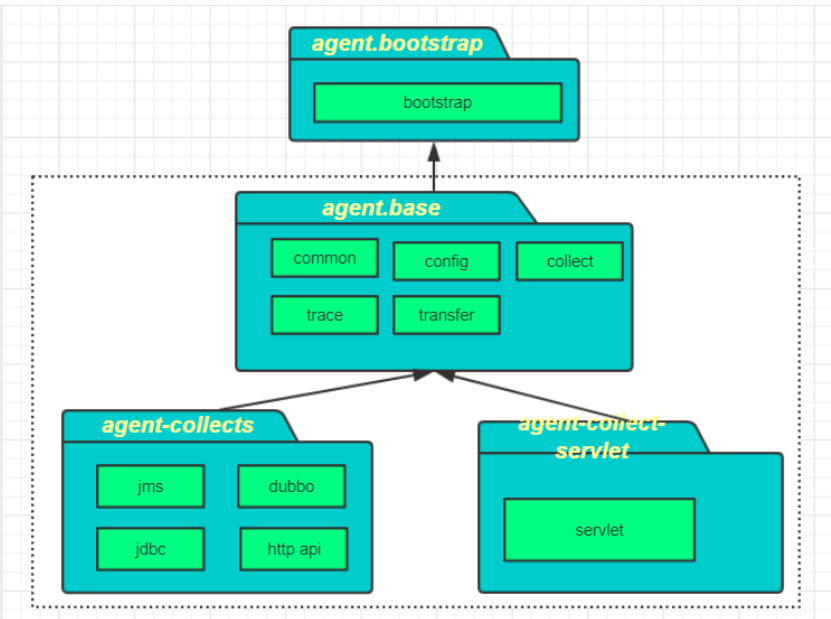
**4、可运维:**

自动更新:




☐ 自动更新演示

## 工程结构：




## 总结：

1. 方便扩展的插件机制
2. 提高体验的中心化配置
3. 可运维的自动化更新

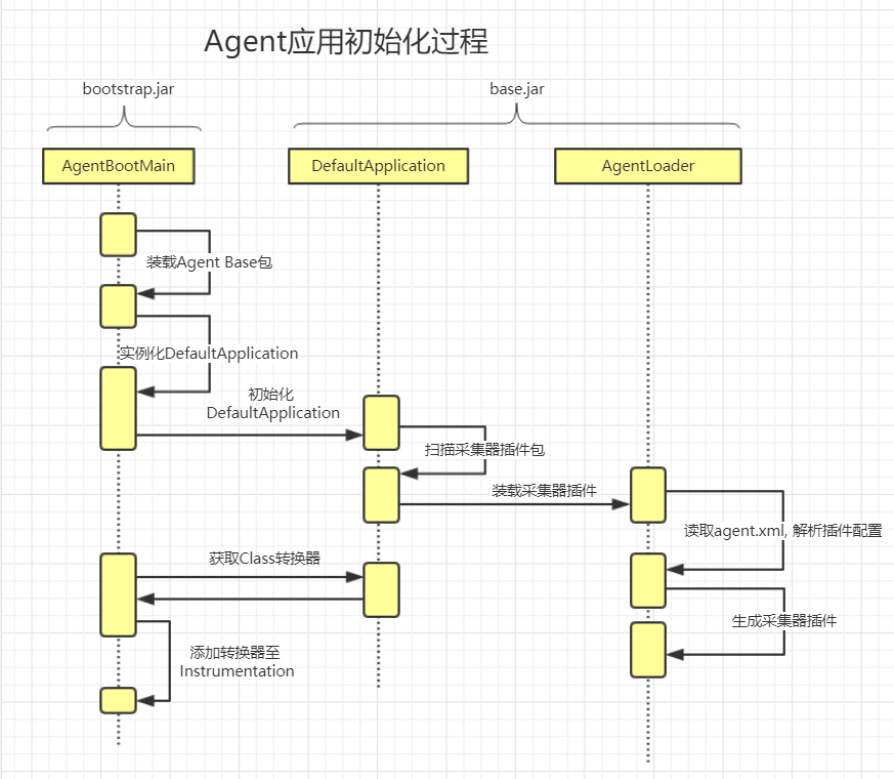
## 二、插桩机制设计

**插桩机制设计要点：**

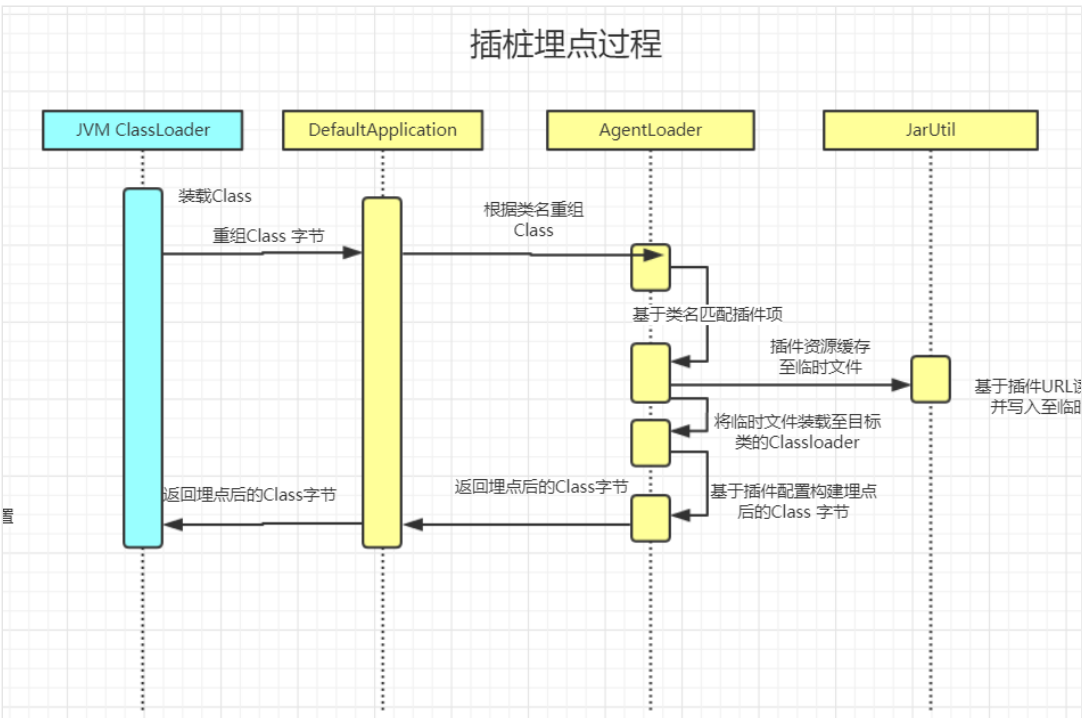
1、插桩前环境准备

2、插桩具体过程

**插桩前准备：**

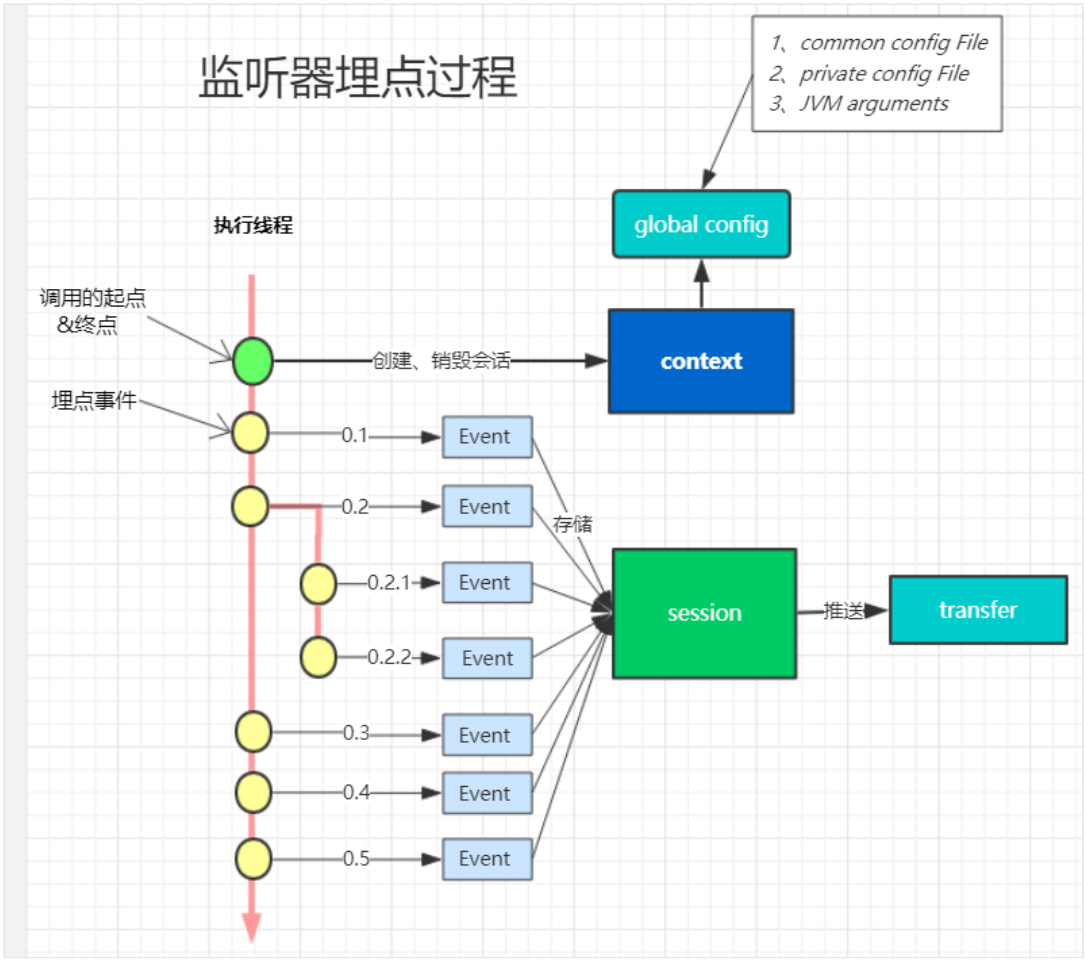



**埋点过程**



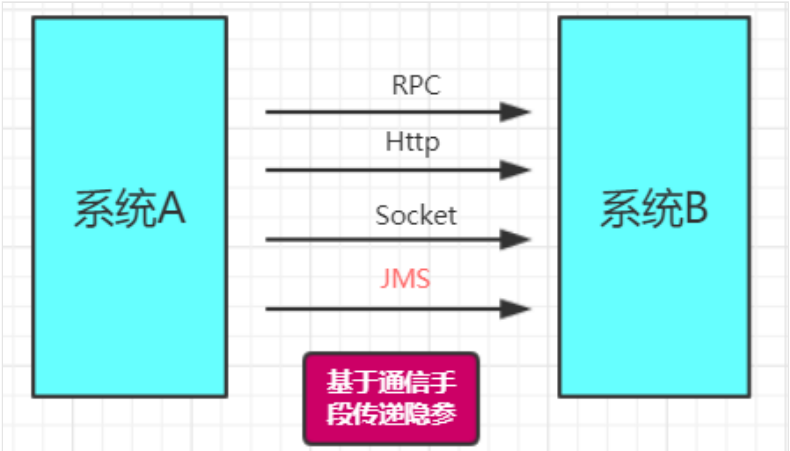
## 三、埋点与采集机制

上节课我们知道，调用链其实就把系统间的调用关系整合成一个链条，那怎么捕获这些调用信息呢？就是在方法的关键调用节点进行埋点捕获调用信息（参数、用时、结果、异常等）。就是埋点，即把监控逻辑动态的埋在业务方法里，并监控该方法的执行情况，具体流程如下图：




**开关跟踪会话:**

通常是在系统的入口处，开启会话，在等结束调用后关闭会话，类似JDBC事物。那么一个系统入口处理有哪些呢?




提问：如果基于Http 可以在哪里埋点用于开启关闭追踪会话?

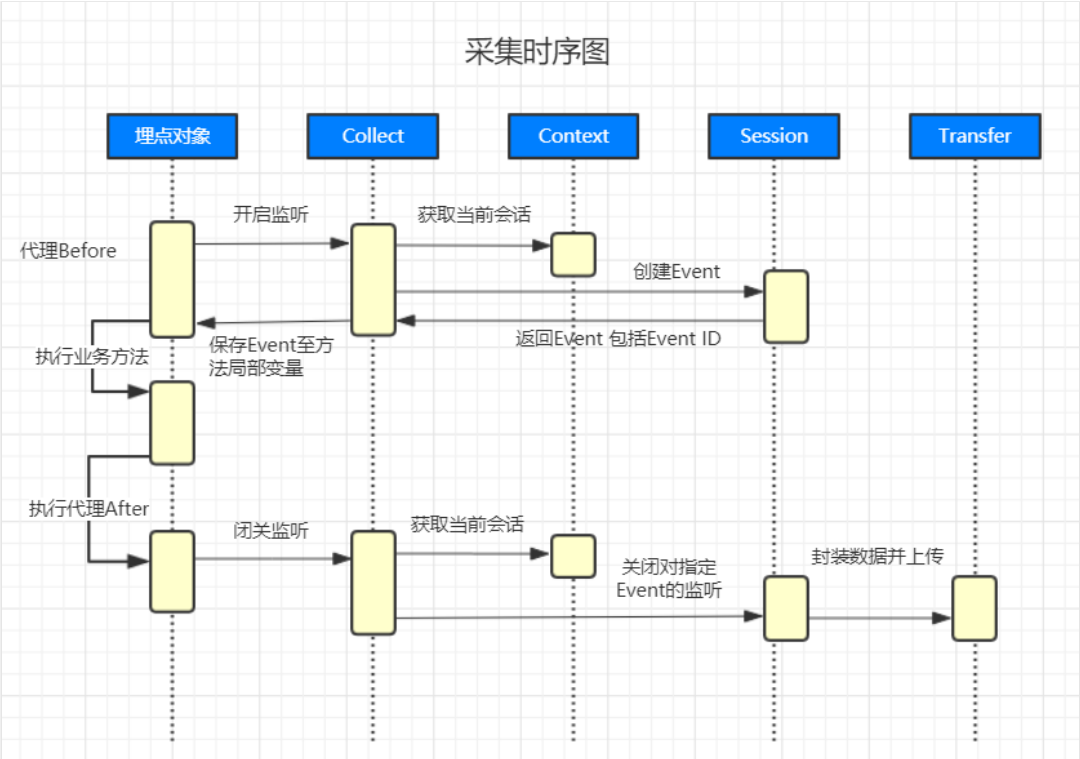
**开关监控事件:**

在监控会话期内发生的事件进行埋点捕捉。

**对像说明:**

1. **Context** ： 开启关闭监控会话和存储配置信息。
2. **Session**：存储监控会话信息（traceId、parentId、当前rpcID）
3. **Event**：存储事件即单次调用信息
4. **request**：上游调用节点信息(parentId,属性)

**具体时序图:**




**演示埋点采集过程:**

- ☐ 演示会话开启与关闭
- ☐ 演示Duubbo 调用埋点

**作业项目部署:**

在内部环境搭建一调用链服务，并将自己的系统与其进行对接.

https://www.elastic.co/downloads/past-releases/elasticsearch-1-7-3

1 、数据库导入数据

2、安装Elasticesearch 1.7.3

2、部署，要修改配置 启动