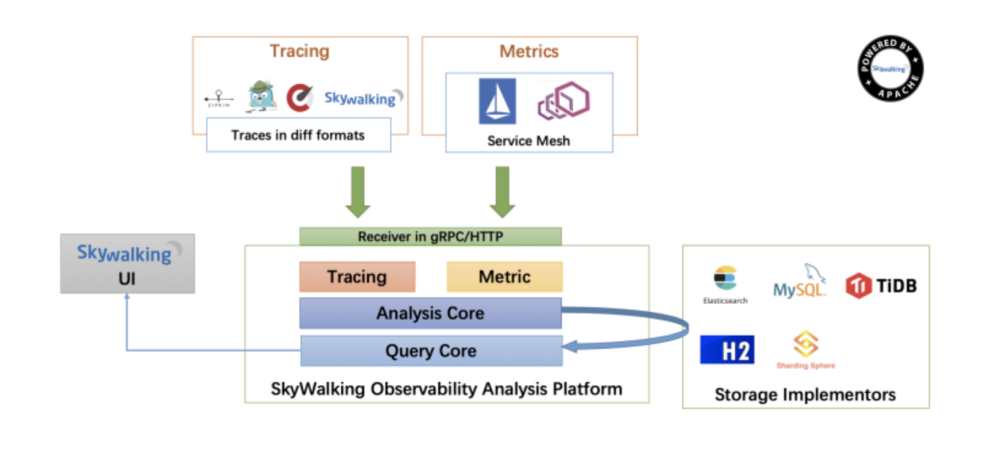
# Skywalking初体验

#### 一、是什么？

SkyWalking是apache基金会下面的一个开源APM项目，为微服务架构和云原生架构系统设计。它通过探针自动收集所需的指标，并进行分布式追踪。通过这些调用链路以及指标，Skywalking APM会感知应用间关系和服务间关系，并进行相应的指标统计。Skywalking支持链路追踪和监控应用组件基本涵盖主流框架和容器，如国产RPC Dubbo和motan等，国际化的spring boot，spring cloud。官方网站：<http://skywalking.apache.org/>

APM (Application Performance Management) 即应用性能管理系统，是对企业系统即时监控以实现对应用程序性能管理和故障管理的系统化的解决方案。应用性能管理，主要指对企业的关键业务应用进行监测、优化，提高企业应用的可靠性和质量，保证用户得到良好的服务，降低IT总拥有成本。APM系统是可以帮助理解系统行为、用于分析性能问题的工具，以便发生故障的时候，能够快速定位和解决问题。

#### 二、整体架构是怎样的？



整体架构包含如下三个组成部分：

1. 探针(agent)负责进行数据的收集，包含了Tracing和Metrics的数据，agent会被安装到服务所在的服务器上，以方便数据的获取。

1. 可观测性分析平台OAP(Observability Analysis Platform)，接收探针发送的数据，并在内存中使用分析引擎（Analysis Core)进行数据的整合运算，然后将数据存储到对应的存储介质上，比如Elasticsearch、MySQL数据库、H2数据库等。同时OAP还使用查询引擎(Query Core)提供HTTP查询接口。

1. Skywalking提供单独的UI进行数据的查看，此时UI会调用OAP提供的接口，获取对应的数据然后进行展示。

Tracing的最大特点就是，它在单次请求的范围内，处理信息。 任何的数据、元数据信息都被绑定到系统中的单个事务上。 例如：一次调用远程服务的RPC执行过程；一次实际的SQL查询语句；一次HTTP请求的业务性ID。总结，Metric主要用来进行数据的统计，比如HTTP请求数的计算。Tracing主要包含了某一次请求的链路数据。

#### 三、能为我们做什么？

**场景1:**

随着分布式系统和微服务架构的出现，一次用户的请求会经过多个系统，不同服务之间的调用关系十分复杂，任何一个系统出错都可能影响整个请求的处理结果。以往的监控系统往往只能知道单个系统的健康状况、一次请求的成功失败，无法快速定位失败的根本原因。

**场景2:**

性能分析：一个服务依赖很多服务，被依赖的服务也依赖了其他服务。如果某个接口耗时突然变长了，那未必是直接调用的下游服务慢了，也可能是下游的下游慢了造成的，如何快速定位耗时变长的根本原因呢？

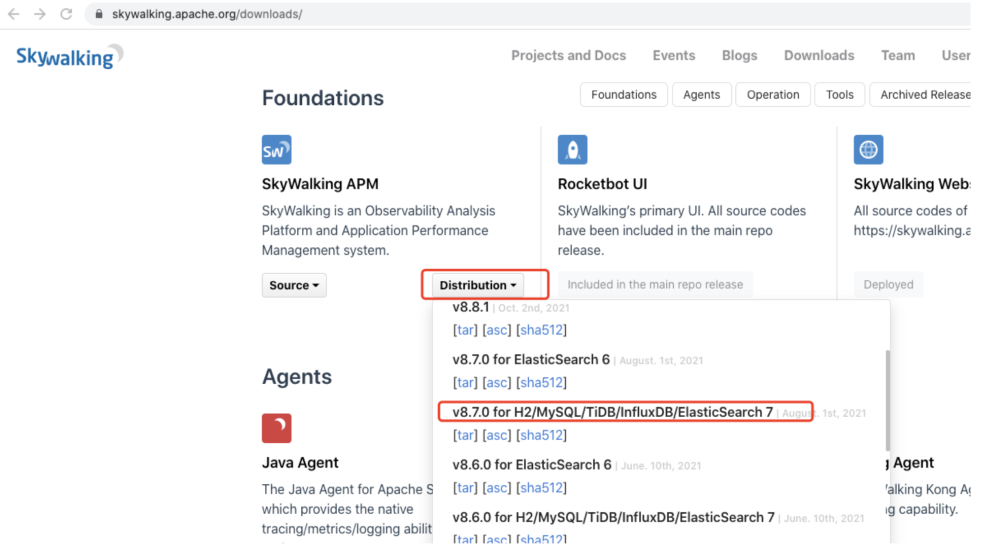
**订单下单链路问题。。。**

详细的介绍：参考博文

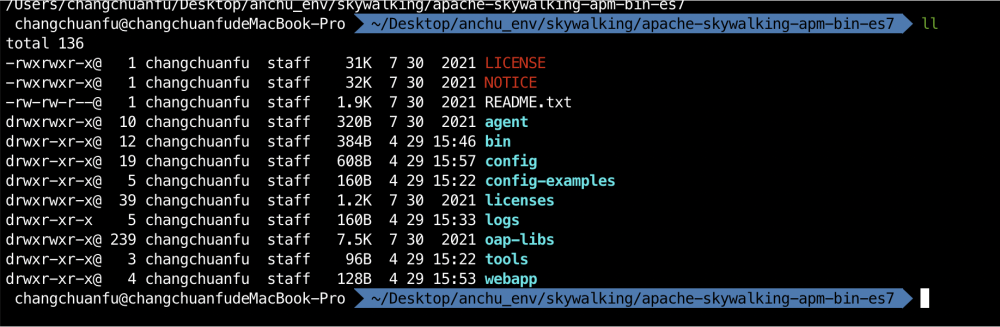
<https://blog.csdn.net/qq_29860591/article/details/106010968>

#### 四、初体验？

下载支持ES版本：<https://skywalking.apache.org/downloads/>



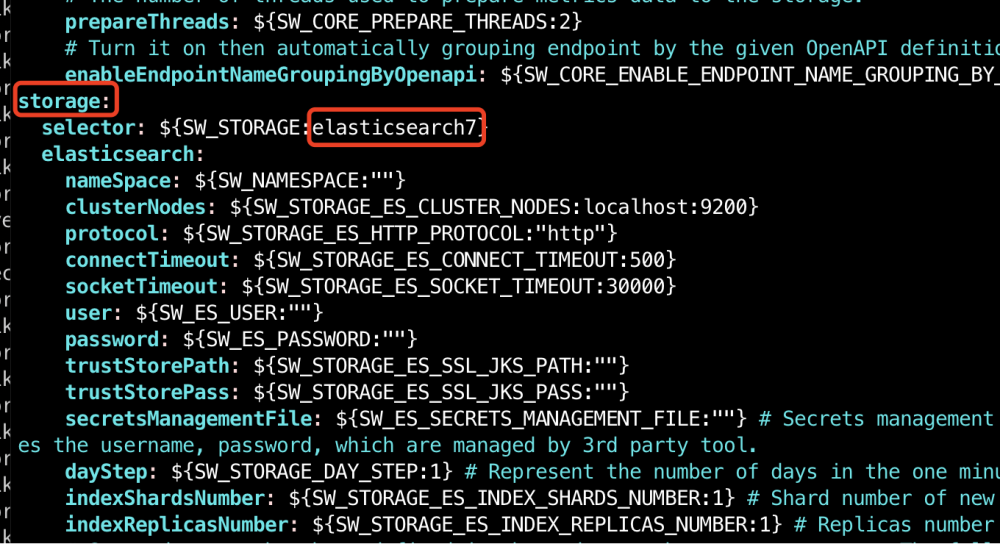
下载解压后的目录结构：



修改相应的配置：

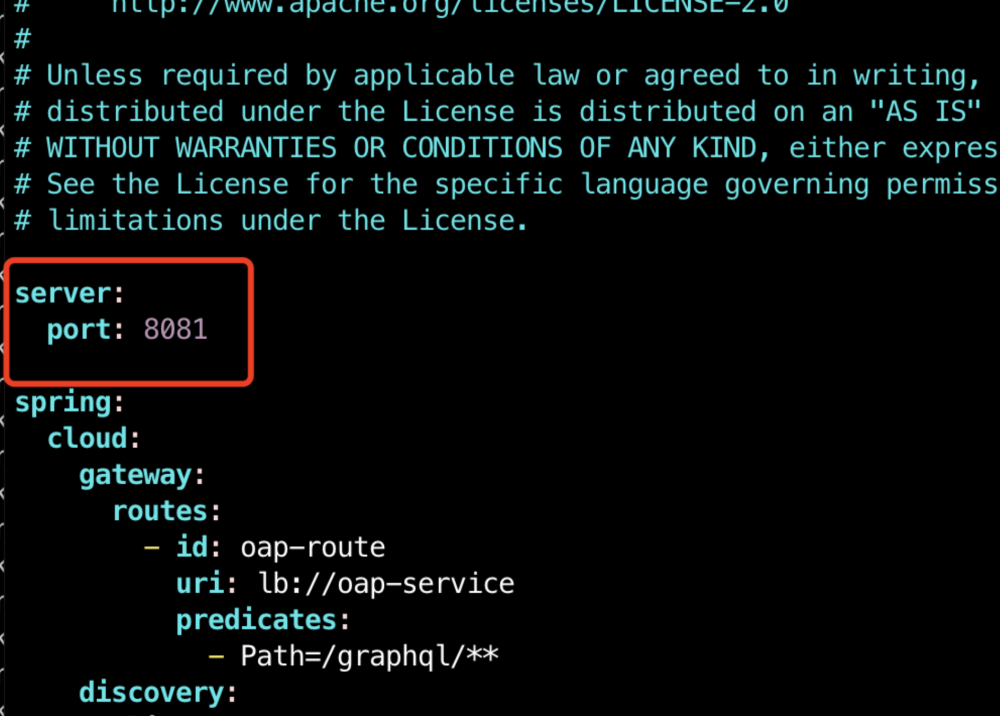
1.修改config/application.yml

主要是调整持久化的方式，这里我们使用es。



2.如有需要调整skywalking web的端口:

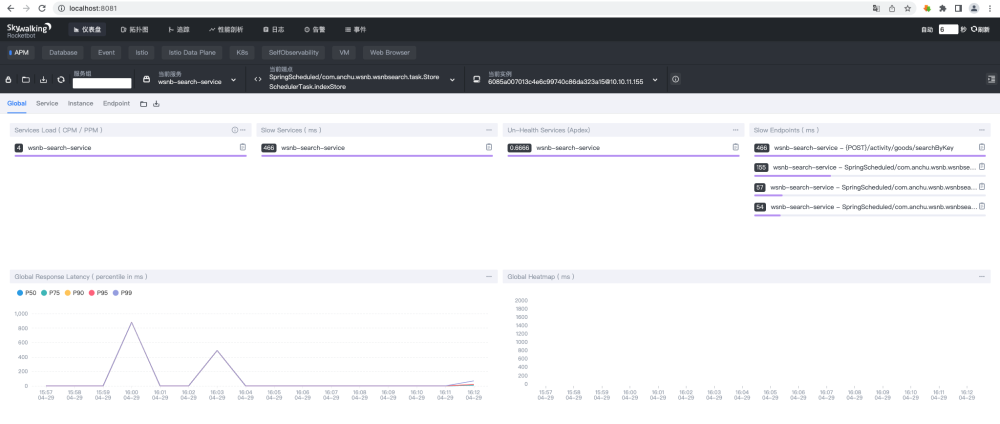
webapp/webapp.yml



3.启动

bin/startup.sh.

查看页面： <http://localhost:8081>



#### 五、效果展示

**将wsnb-service这个项目集成skywalking。**

因为skywalking使用的是agent的方式，所以对我们的代码是令侵入。只要在启动的时候指定agent的路径即可。

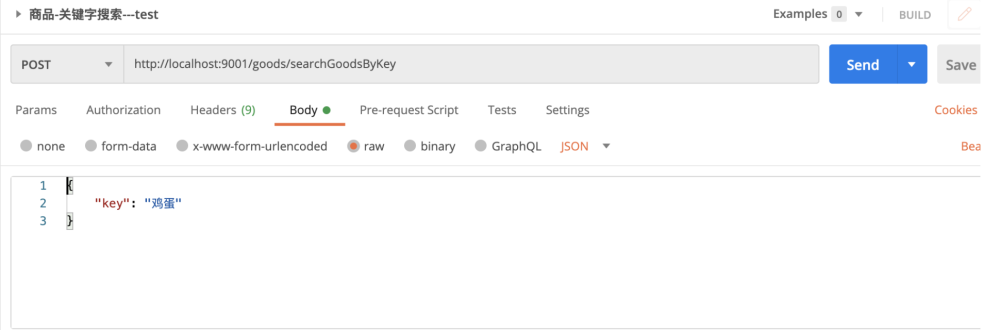
java -javaagent:/Users/changchuanfu/Desktop/anchu\_env/skywalking/skywalking-test/agent/skywalking-agent.jar   
-Dskywalking.agent.service\_name=wsnb-search-service  
-Dskywalking.collector.backend\_service=127.0.0.1:11800   
-jar wsnb-search-0.0.1-SNAPSHOT.jar &

参数解释：

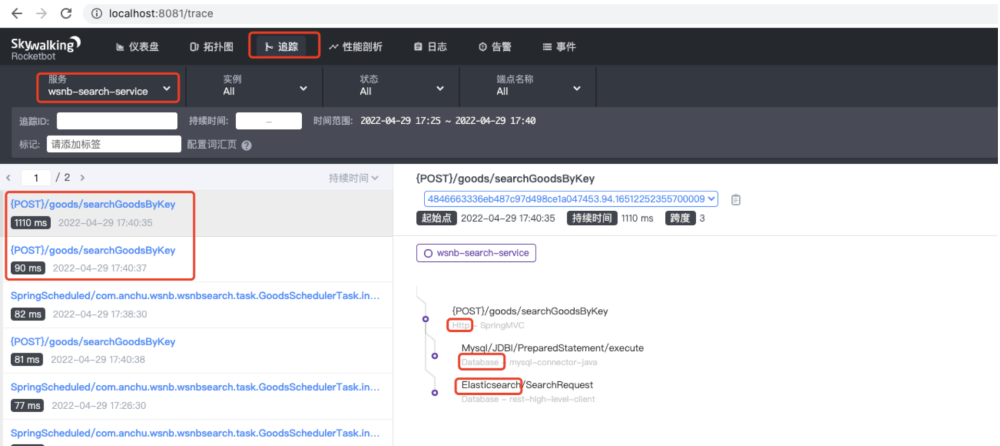
-javaagent:/Users/changchuanfu/Desktop/anchu\_env/skywalking/skywalking-test/agent/skywalking-agent.jar     # 指定agent-jar的位置

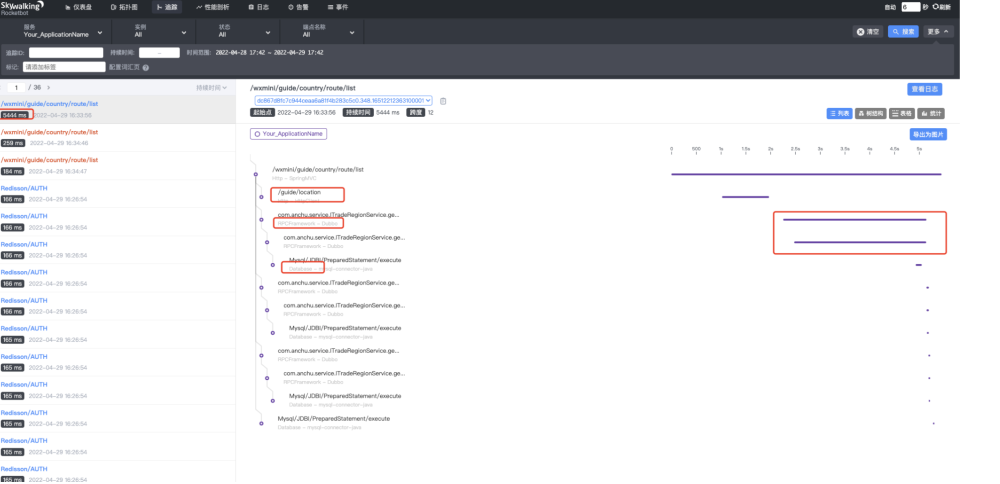
-Dskywalking.agent.service\_name=wsnb-search-service   # 服务名称

-Dskywalking.collector.backend\_service=127.0.0.1:11800 # 指定skywalking的地址

请求： wsnb-service这个服务的商品搜索接口

效果：可以看到一次请求经过了http、db、es等链路。以及相应的执行时间。





#### 六、日志增加traceId

为什么要在日志中增加traceId ？

skywalking的ui中是支持traceId进行检索的，所以当出现性能问题的时候，打印出traceId，可以准确找到我们要定位的地方。

官网配合log4j2地址： <https://skywalking.apache.org/docs/main/v8.6.0/en/setup/service-agent/java-agent/application-toolkit-log4j-2.x/>

具体步骤如下：

**步骤1**: 项目中集成如下依赖

<dependency>  
 <groupId>org.apache.skywalking</groupId>  
 <artifactId>apm-toolkit-log4j-2.x</artifactId>  
 <version>{project.release.version}</version>  
 </dependency>

**步骤2**: 更改log4j2.xml 打印日志的pattern，增加traceId



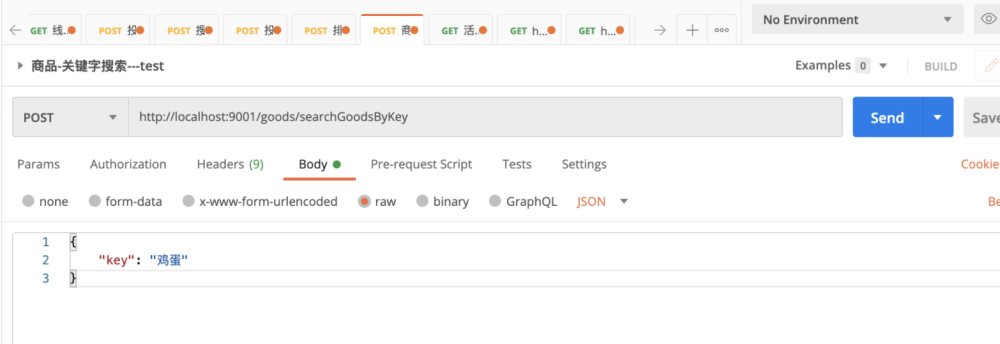
就完事了。。。。。几乎无侵入吧。

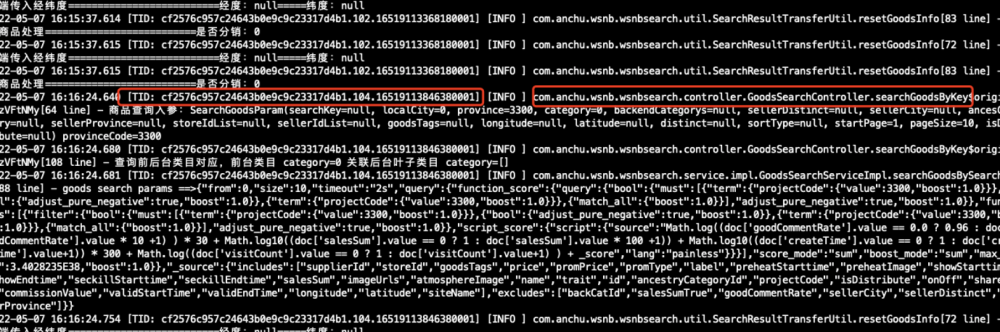
**步骤3**: 查看效果

非接口请求时，打印N/A



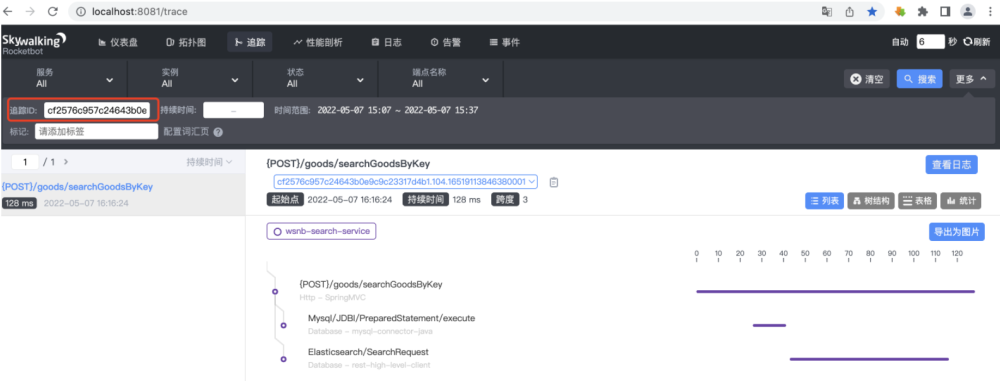
当我们请求接口：可以看到已经打印了TID





**步骤4**: 当我们有了TID之后，我们就可以到web界面去分析了。

TID： cf2576c957c24643b0e9c9c23317d4b1.104.16519113846380001



#### 七、源码解析

todo