微服务链路追踪SkyWalking

1. SkyWalking简介

skywalking是一个国产开源框架，2015年由吴晟开源 ， 2017年加入Apache孵化器。skywalking是分布式系统的应用程序性能监视工具，专为微服务、云原生架构和基于容器（Docker、K8s、Mesos）架构而设计。它是一款优秀的APM（Application Performance Management）工具，包括了分布式追踪、性能指标分析、应用和服务依赖分析等。

官网：http://skywalking.apache.org/

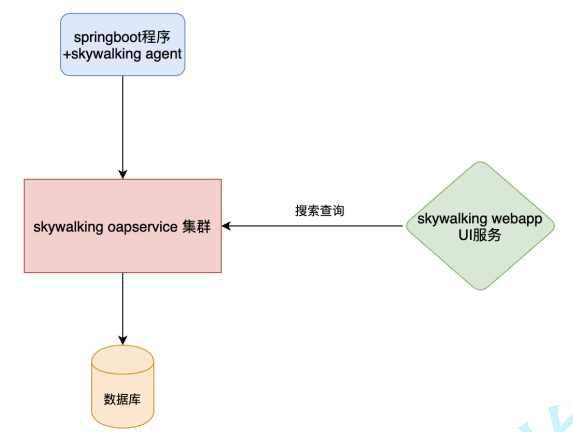
下载：http://skywalking.apache.org/downloads/

Github：https://github.com/apache/skywalking

文档： https://skywalking.apache.org/docs/main/v8.4.0/readme/

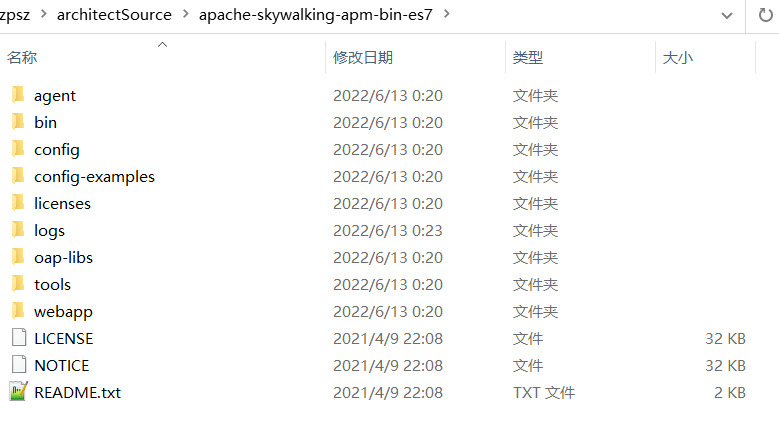
中文文档： <https://skyapm.github.io/document-cn-translation-of-skywalking/>

1. SkyWalking特性
2. 多种监控手段，可以通过语言探针和service mesh获得监控的数据；
3. 支持多种语言自动探针，包括 Java，.NET Core 和 Node.JS；
4. 轻量高效，无需大数据平台和大量的服务器资源；
5. 模块化，UI、存储、集群管理都有多种机制可选；
6. 支持告警；
7. 优秀的可视化解决方案；
8. SkyWalking 环境搭建部署



* 1. Skywalking agent和业务系统绑定在一起，负责收集各种监控数据。
  2. Skywalking oapservice是负责处理监控数据的，比如接受skywalking agent的监控数据，并存储在数据库中;接受skywalking webapp的前端请求，从数据库查询数据，并返回数据给前端。Skywalking oapservice通常以集群的形式存在。
  3. Skywalking webapp，前端界面，用于展示数据。
  4. 数据库则用于存储监控数据，数据库可选mysql、elasticsearch等。
  5. SkyWalking 目录介绍

下载地址：<http://skywalking.apache.org/downloads/>， 以apache-skywalking-apm-bin-es7 v8.0.4 for H2/MySql/TiDB/InfluxDB/ElasticSearch7 版本为例。



Webapp：UI前端（web监控页面）的jar和配置文件。

Oap-libs：后台应用jar，以及依赖jar。

Config：各种配置文件。

Bin：启动脚本。

Agent：包含skywalking-agent.jar代理服务jar、config代理服务配置文件、plugins插件和optional-plugins可选插件。

* 1. 搭建SkyWalking OAP 服务，以windows环境下本地工程为例

在工程中配置jvm参数，如下所示为例：

# skywalking‐agent.jar的本地磁盘的路径

-javaagent:.\apache-skywalking-apm-bin-es7\agent\skywalking-agent.jar

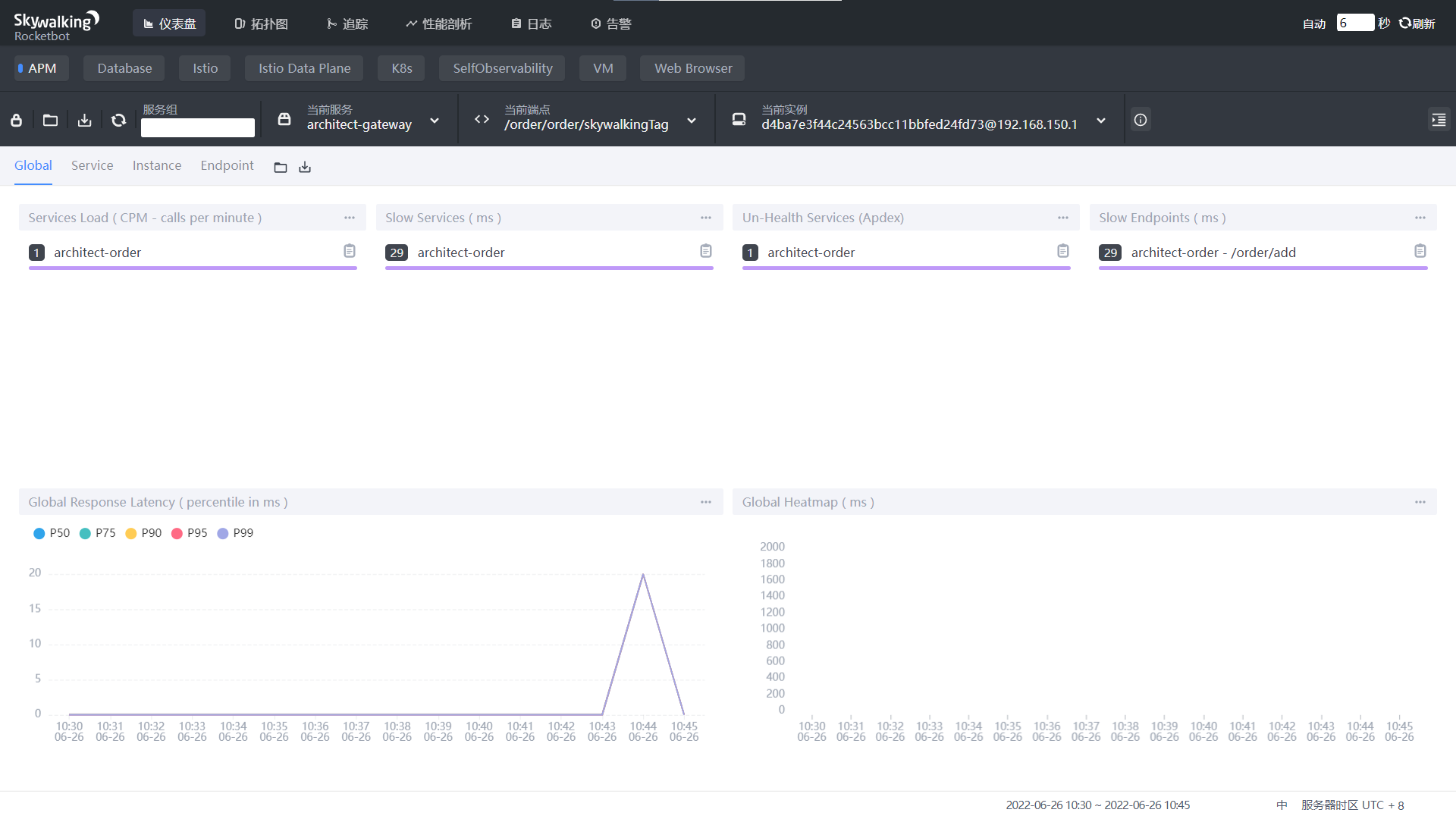
# 在skywalking上显示的服务名

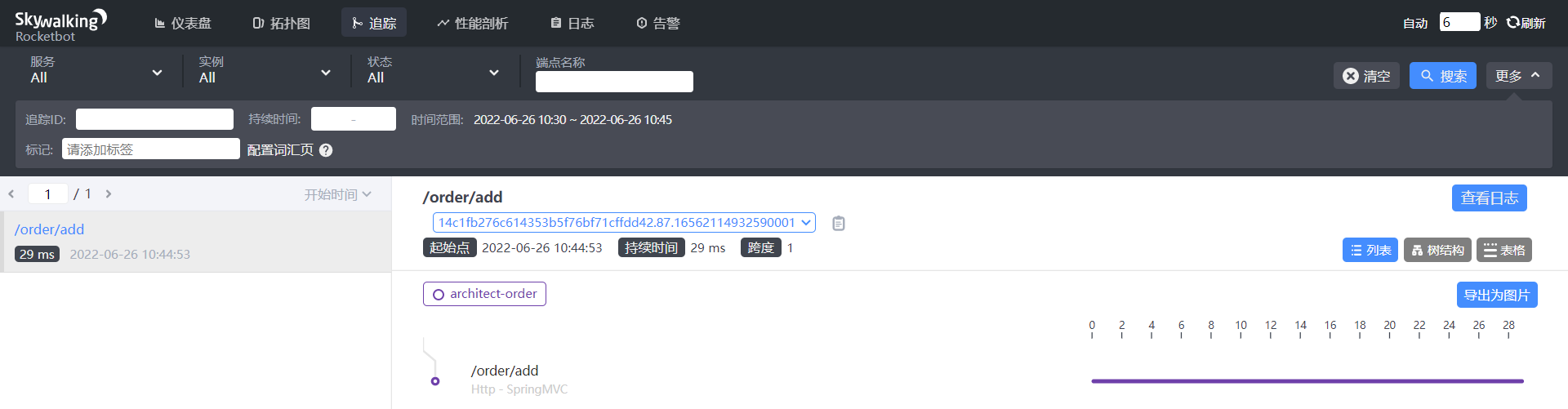
-DSW\_AGENT\_NAME=architect-order

# skywalking的collector服务的IP及端口

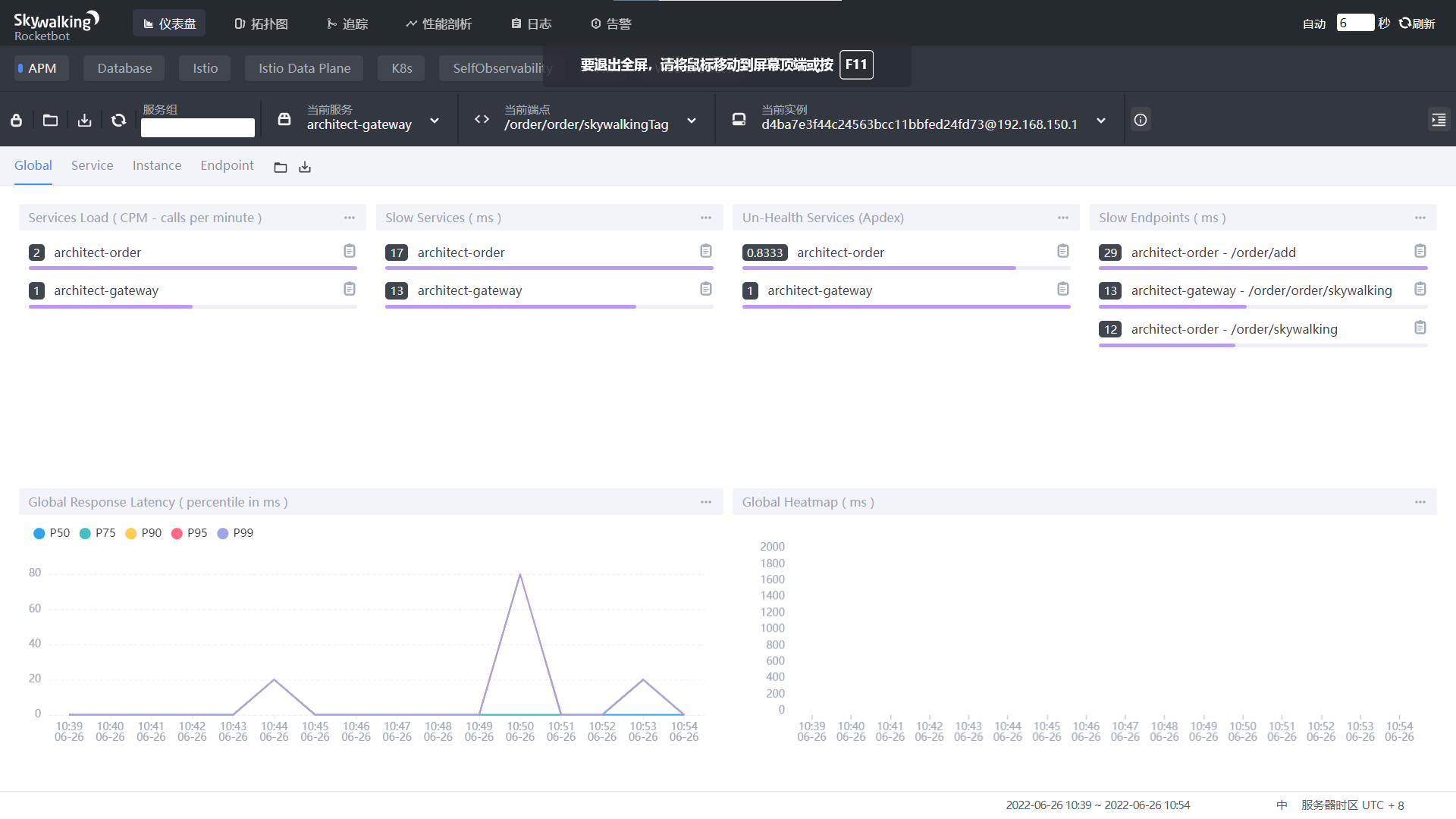
-DSW\_AGENT\_COLLECTOR\_BACKEND\_SERVICES=127.0.0.1:11800

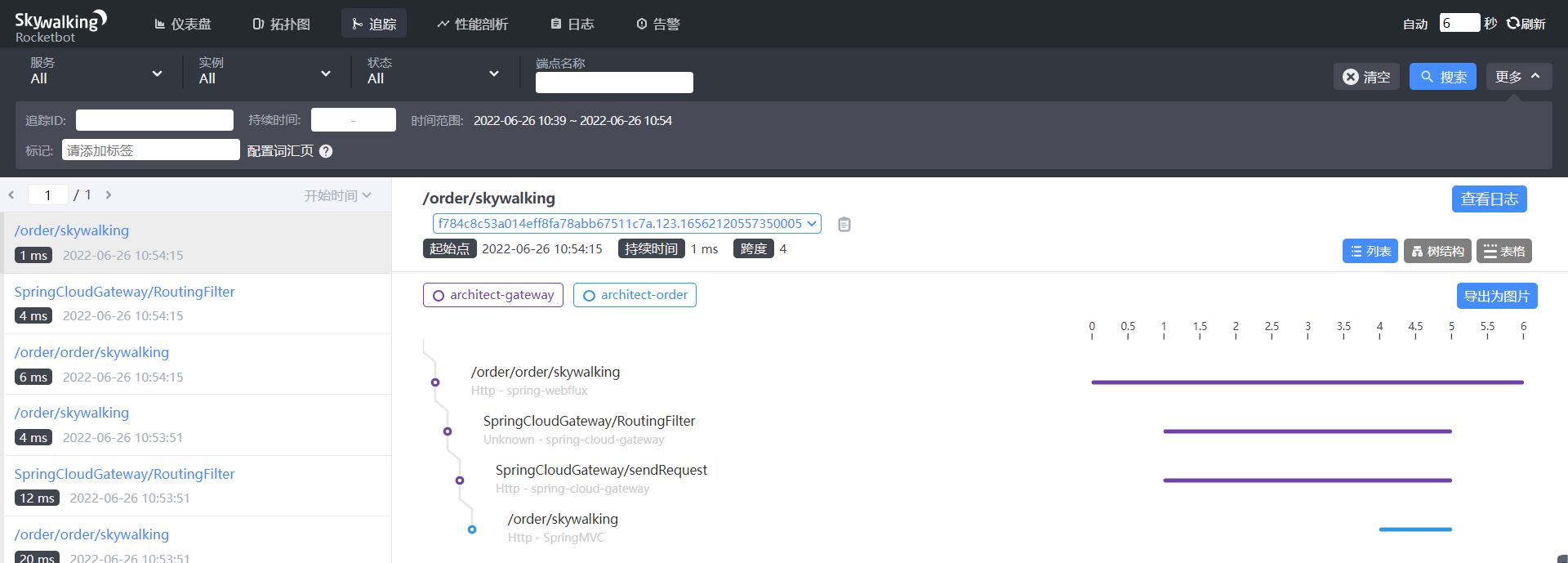
启动脚本bin/startup.cmd，会默认启动2个服务，一个是skywalking-oap-server，一个是skywalking-web-ui（webapp/webapp.yml配置端口8868）。然后启动本地工程，调用随意一个接口，即可在localhost:8868/中看到





配合网关、多个微服务，链路会更明显。





1. 自定义SkyWalking链路追踪

如果我们希望对项目中的业务方法，实现链路追踪，方便我们排查问题，可以配置如下。

添加依赖

<dependency>

<groupId>org.apache.skywalking</groupId>

<artifactId>apm-toolkit-trace</artifactId>

<version>8.4.0</version>

</dependency>

* 1. 如果一个业务方法想在ui界面的跟踪链路上显示出来，只需要在业务方法上加上@Trace注解即可。
  2. 我们还可以为追踪链路增加其他额外的信息，比如记录参数和返回信息。实现方式：在方法上增加@Tag或者@Tags。

@Tag注解中key=标记；value= returnedObj（返回值）、arg[0]（入参参数）

若方法如下配置@Trace和@Tag

@Service

public class OrderService {

@Trace

@Tags({

@Tag(key="rtnVal", value = "returnedObj"),

@Tag(key = "name", value = "arg[0]")

})

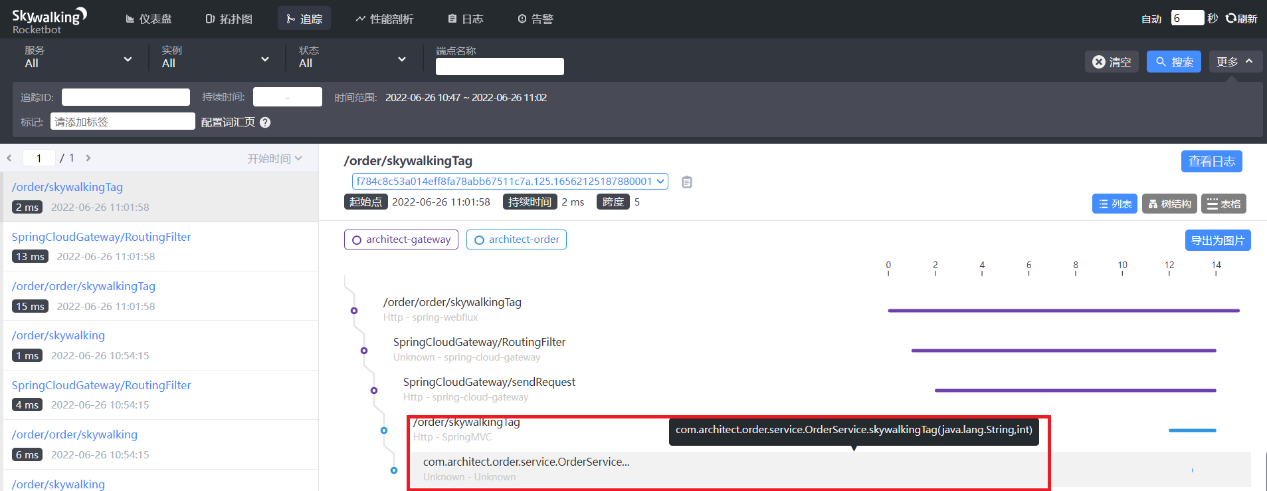
public String skywalkingTag(String name, int id) {

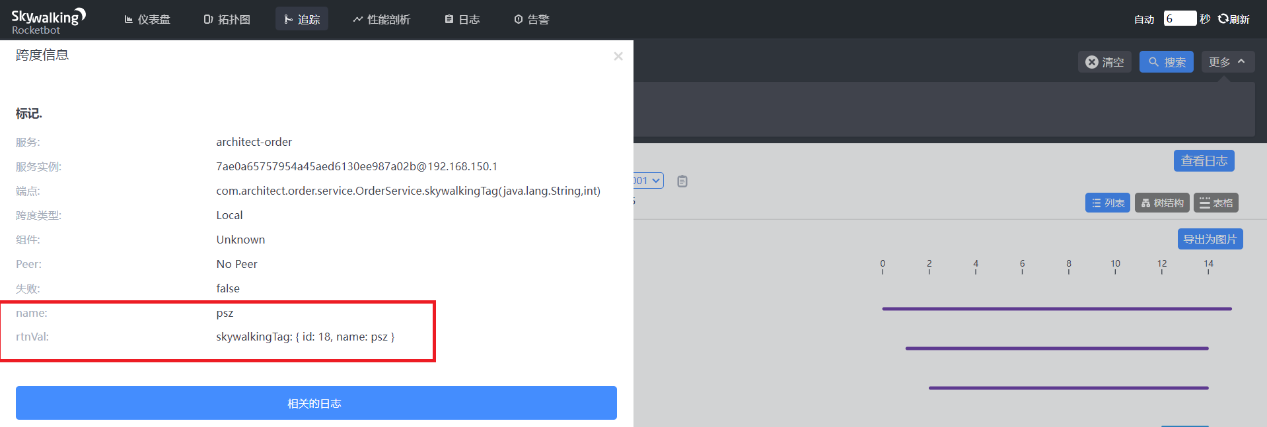
return "skywalkingTag: " + "{ id: " + id + ", name: " + name + " }";

}

}

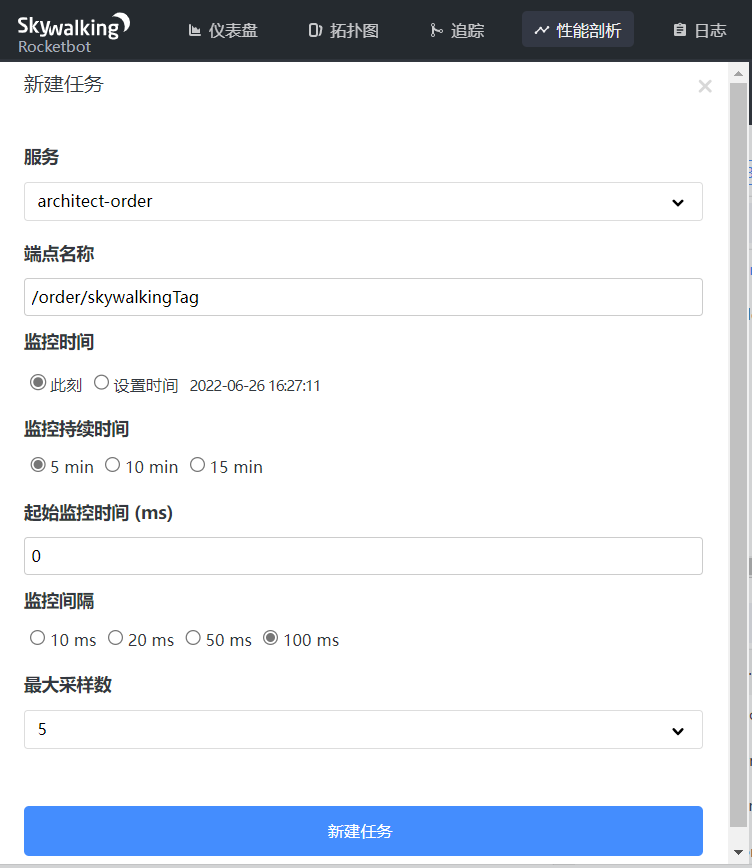
链路截图如下

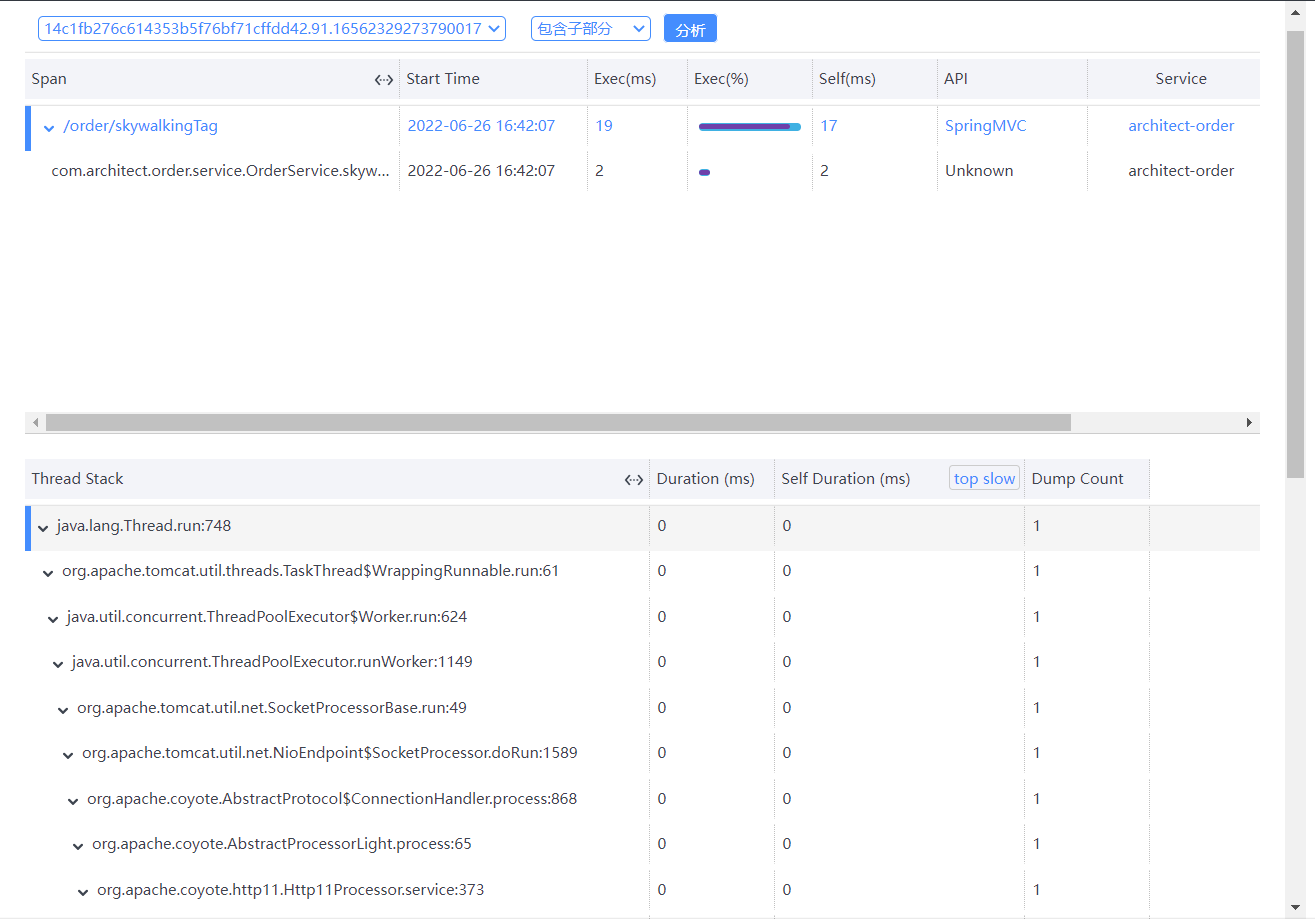




1. 性能剖析

如下建立性能剖析任务，多次调用/order/add1方法后，可以看到该端点的剖析结果





1. 日志

6.1 控制台tid

添加依赖

<dependency>

<groupId>org.apache.skywalking</groupId>

<artifactId>apm-toolkit-logback-1.x</artifactId>

<version>8.5.0</version>

</dependency>

配置logback-spring.xml，并配置%tid占位符

<?xml version="1.0" encoding="UTF-8"?>

<configuration>

<!-- 引入 Spring Boot 默认的 logback XML 配置文件 ‐-->

<include resource="org/springframework/boot/logging/logback/defaults.xml"/>

<appender name="console" class="ch.qos.logback.core.ConsoleAppender">

<!-- 日志的格式化 -->

<encoder class="ch.qos.logback.core.encoder.LayoutWrappingEncoder">

<layout class="org.apache.skywalking.apm.toolkit.log.logback.v1.x.TraceIdPatternLogbackLayout">

<Pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%tid] [%thread] %-5level %logger{36} -%msg%n</Pattern>

</layout>

</encoder>

</appender>

<!-- 设置 Appender -->

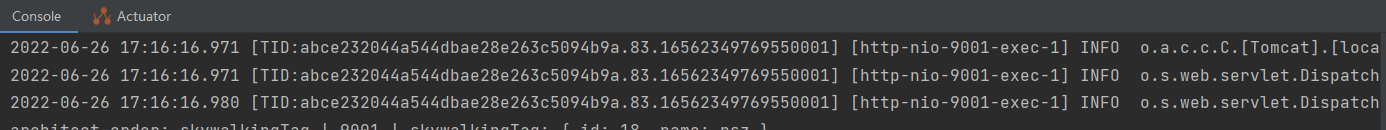
<root level="INFO">

<appender-ref ref="console"/>

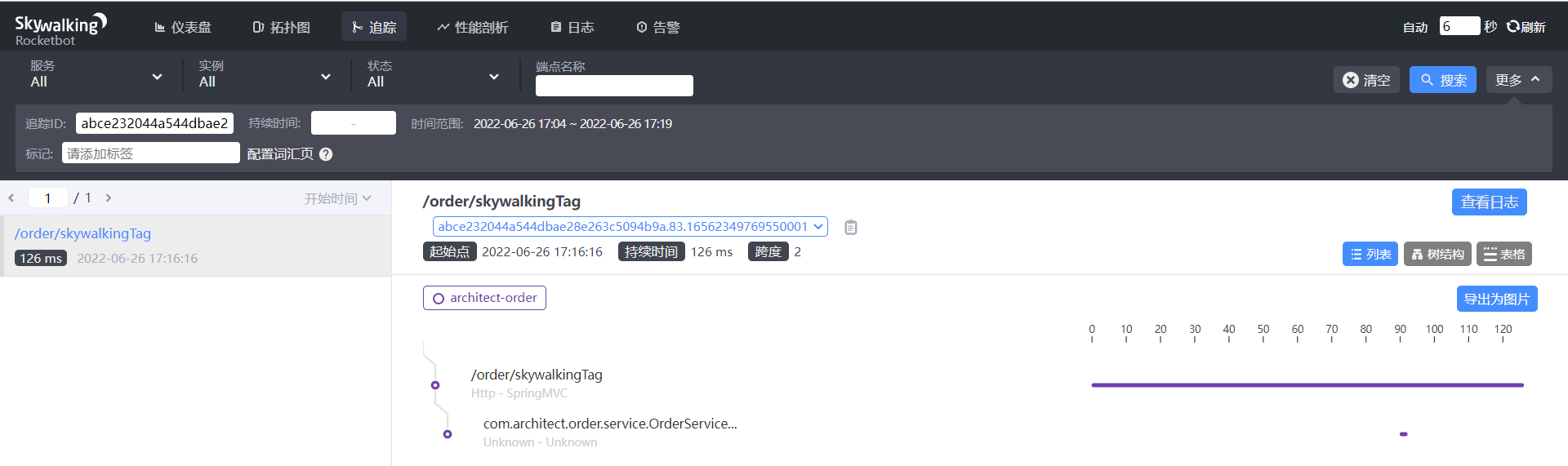
</root>

</configuration>

在调用任意接口后，控制台会显示出%tid对应的值



Skywalking里的追踪可以凭借tid查询



6.2 Skywalking通过grpc上报日志 （需要v8.4.0+）

Logback-spring.xml配置如下

<?xml version="1.0" encoding="UTF-8"?>

<configuration>

<!-- 引入 Spring Boot 默认的 logback XML 配置文件 ‐-->

<include resource="org/springframework/boot/logging/logback/defaults.xml"/>

<appender name="console" class="ch.qos.logback.core.ConsoleAppender">

<!-- 日志的格式化 -->

<encoder class="ch.qos.logback.core.encoder.LayoutWrappingEncoder">

<layout class="org.apache.skywalking.apm.toolkit.log.logback.v1.x.TraceIdPatternLogbackLayout">

<Pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%tid] [%thread] %-5level %logger{36} -%msg%n</Pattern>

</layout>

</encoder>

</appender>

<appender name="grpc-log" class="org.apache.skywalking.apm.toolkit.log.logback.v1.x.log.GRPCLogClientAppender">

<encoder class="ch.qos.logback.core.encoder.LayoutWrappingEncoder">

<layout class="org.apache.skywalking.apm.toolkit.log.logback.v1.x.mdc.TraceIdMDCPatternLogbackLayout">

<Pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%X{tid}] [%thread] %-5level %logger{36} -%msg%n</Pattern>

</layout>

</encoder>

</appender>

<!-- 设置 Appender -->

<root level="INFO">

<appender-ref ref="console"/>

<appender-ref ref="grpc-log"/>

</root>

</configuration>

在调用任意接口后，可直接在Skywalking里的日志中看到

