大标题

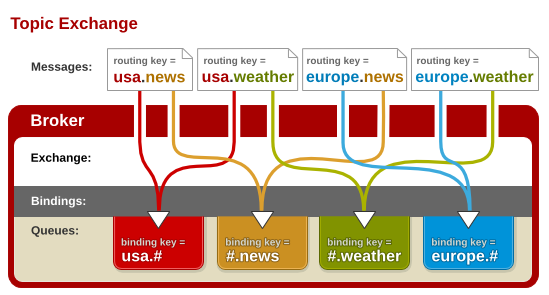
# 监控方案

## Prometheus

### 参考

使用Prometheus+Grafana快速打造高逼格监控平台 <https://juejin.im/post/6844903897220415501>

#### 4



正文

Sample Java Agent and Bytecode manipulation

Sample maven project containing a Java agent and examples of bytecode manipulation with ASM and Javassist.

# 知识储备

## JavaAgent入门，基于Javassist的demo

<https://zhuanlan.zhihu.com/p/139756708>，如何在pom文件设置Premain-Class打包

## JavaAgent原理与实践，为何Javassist的加载必须放置在其他（如ASM Code或者ByteBuddy）之前

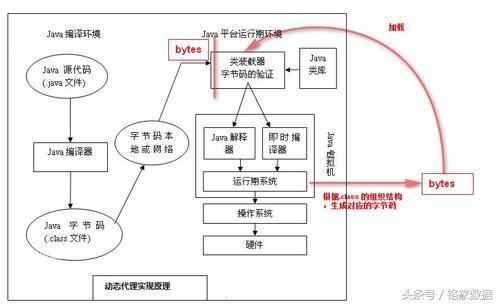
<https://www.infoq.cn/article/fh69pypqzpf6cj1ujy7x>，当两个JavaAgent的载入顺序为：Javassist-ByteBuddy，这两个JavaAgent对于同一个类的同一个方法的增强都生效了；当两个JavaAgent的载入顺序为：ByteBuddy-Javassist，Javassist对应的增强生效了，而ByteBuddy对应的增强却没生效；当使用三个JavaAgent（两个Javassist，一个ByteBuddy）的载入顺序为:Javassist-ByteBuddy-Javassist，两个Javassist的增强都生效了，而ByteBuddy对应的增强却没生效

## 基于javaAgent和ASM字节码技术跟踪java程序调用链

<https://www.jianshu.com/p/88be1658f26e>，-javaagent这个参数的个数是没有限制的，所以可以添加任意多个javaagent；所有的javaagent会按照运行时的参数顺序执行；此外，javaagent需要放在包含main方法的jar包之前，否则javaagent不会起作用；每一个javaagent都可以接收一个字符串类型的参数，也就是premain中的agentArgs

## Java探针-Java Agent技术-阿里面试题

<https://www.cnblogs.com/aspirant/p/8796974.html>，如何启动代理和设置热部署+基于javaAgent和Java字节码注入技术的java探针工具技术原理图；利用javaAgent和ASM字节码技术，在JVM加载class二进制文件的时候，利用ASM动态的修改加载的class文件



## javaagent使用指南

<https://www.cnblogs.com/rickiyang/p/11368932.html>，源码解析（带中文注释）+detach方法的解释

## class is frozen（冻结class）的解决

<https://my.oschina.net/momisabuilder/blog/1845775>，如果一个CtClass对象通过writeFile，toClass或者toByteCode转换成class文件，那么javassist会冻结这个CtClass对象。后面就不能修改这个CtClass对象了。这样是为了警告开发者不要修改已经被JVM加载的class文件，因为JVM不允许重新加载一个类

## JVM源码分析之javaagent原理完全解读

<https://www.infoq.cn/article/javaagent-illustrated>

### javaagent 的主要功能

在加载 class 文件之前做拦截，对字节码做修改；在运行期对已加载类的字节码做变更，但是这种情况下会有很多的限制

### 多个javaagent的情况

如果同时使用基于Javassist和基于其他字节码工具的JavaAgent去增强同一个类，Javassist的加载顺序一定要在其他字节码（如ASM Code或者ByteBuddy）的JavaAgent之前，这样才能保证两个字节码工具都可以进行完整的增强。如果基于Javassist的JavaAgent最后增强，那么之前的非Javassist的JavaAgent对于字节码的增强都会被丢弃掉，这也能会带来不小的麻烦

# 实战

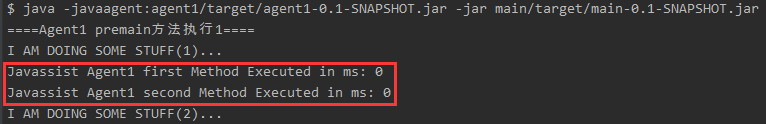
## Build & Run

From the root dir

$ mvn clean install -DskipTests

执行单个代理

$ java -javaagent:agent1/target/agent1-0.1-SNAPSHOT.jar -jar main/target/main-0.1-SNAPSHOT.jar



执行多个代理

$ java -javaagent:agent1/target/agent1-0.1-SNAPSHOT.jar -javaagent:agent2/target/agent2-0.1-SNAPSHOT.jar -jar main/target/main-0.1-SNAPSHOT.jar

