**华为公有云实习日记**

第五/六周July

**Task：**

Prometheus 监控系统调研与实践

任务输入：

1）了解Prometheus监控系统

2）github、google等官网社区，调研kafka和zookeeper 2个 exporter

3）重点掌握2-3个exporter

任务输出：

1) expoter测试验证，代码归档

2）本地搭建Prometheus demo，对2个Exporter做监控验证，demo实践代码归档

3）做一次汇报，讲解exporter具体代码。

4）输出实践总结，归档

验证环境：

Azure 虚拟机,：

系统Linux (ubuntu 18.04)

Java -version:

openjdk version "1.8.0\_262"

OpenJDK Runtime Environment (Zulu 8.48.0.51-linux64)-Microsoft-Azure-restricted (build 1.8.0\_262-b19)

OpenJDK 64-Bit Server VM (Zulu 8.48.0.51-linux64)-Microsoft-Azure-restricted (build 25.262-b19, mixed mode)

**一. 1. 了解Prometheus监控系统**

**Prometheus 架构**

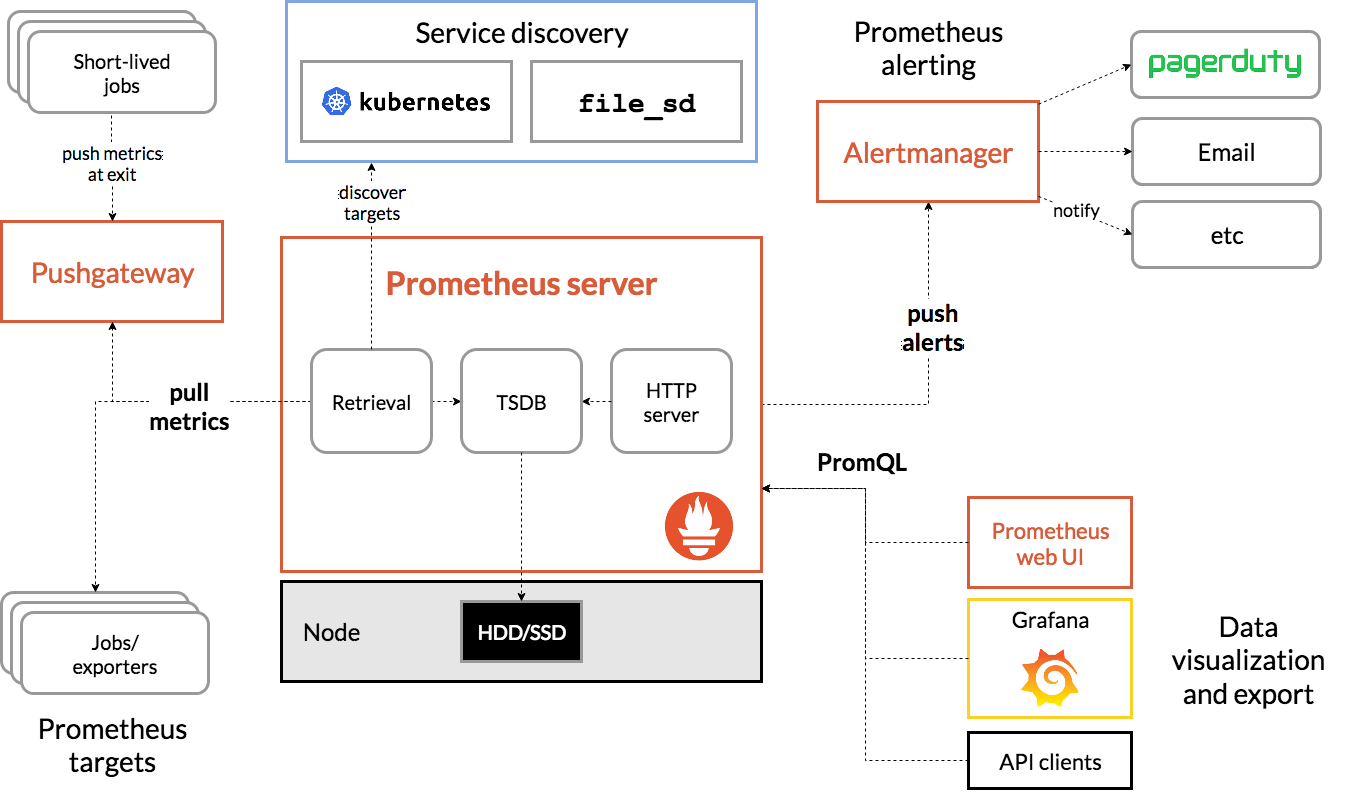
Prometheus joined the Cloud Native Computing Foundation in 2016 as the second hosted project, after Kubernetes.

For more elaborate overviews of Prometheus, see the resources linked from the media section.

**Features**

Prometheus's main features are:

* a multi-dimensional data model with time series data identified by metric name and key/value pairs
* PromQL, a flexible query language to leverage this dimensionality
* no reliance on distributed storage; single server nodes are autonomous
* time series collection happens via a pull model over HTTP
* pushing time series is supported via an intermediary gateway
* targets are discovered via service discovery or static configuration
* multiple modes of graphing and dashboarding support



**核心组件**

* Prometheus Server， 主要用于抓取数据和存储时序数据，另外还提供查询和 Alert Rule 配置管理。
* client libraries，用于对接 Prometheus Server, 可以查询和上报数据。
* push gateway ，用于批量，短期的监控数据的汇总节点，主要用于业务数据汇报等。
* 各种汇报数据的 exporters ，例如汇报机器数据的 node\_exporter, 汇报 MongoDB 信息的 MongoDB exporter 等等。
* 用于告警通知管理的 alertmanager 。

**2. github、google等官网社区，调研kafka和zookeeper 2个 exporter**

**相关概念：**

**Exporter:**

*An exporter is a binary running alongside the application you want to obtain metrics from. The exporter exposes Prometheus metrics, commonly by converting metrics that are exposed in a non-Prometheus format into a format that Prometheus supports.【1】*

**（一个Exporter本质就是将收集到的数据转换为文本格式并提供http请求即可）**

**注释**：

* 以 # HELP 开头表示 metric 帮助说明。
* 以 # TYPE 开头表示定义 metric 类型，含 counter, gauge, histogram, summary, 和 untyped 类型。
* 其他表示一般注释，供阅读使用，将被 Prometheus 忽略。

**采样数据：**

内容如果不以 # 开头，表示采样数据。它通常紧挨着类型定义行，满足以下格式：

metric\_name [

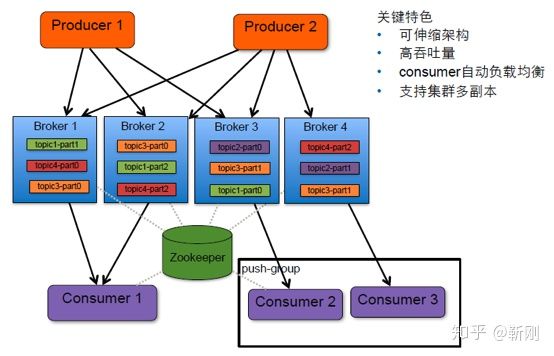
"{" label\_name "=" `"` label\_value `"` { "," label\_name "=" `"` label\_value `"` } [ "," ] "}"] value [ timestamp ]

**Tip**:

假设采样数据 metric 叫做 x, 如果 x 是 histogram 或 summary 类型必需满足以下条件：

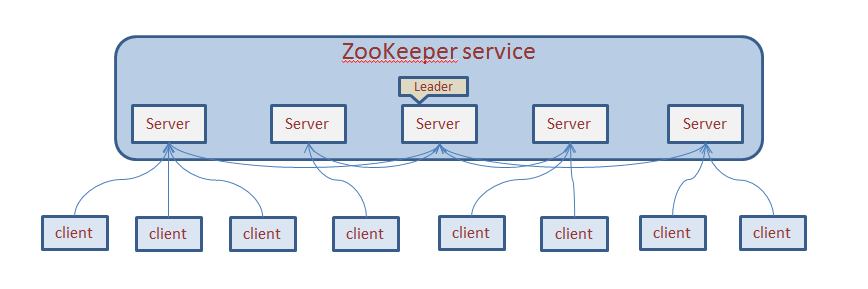
* 采样数据的总和应表示为 x\_sum。
* 采样数据的总量应表示为 x\_count。
* summary 类型的采样数据的 quantile 应表示为 x{quantile="y"}。
* histogram 类型的采样分区统计数据将表示为 x\_bucket{le="y"}。
* histogram 类型的采样必须包含 x\_bucket{le="+Inf"}, 它的值等于 x\_count 的值。
* summary 和 historam 中 quantile 和 le 必需按从小到大顺序排列。

**Kafka 架构：**



* Topic：消息的类别。Kafka中可以将Topic从物理上划分成一个或多个分区（Partition）
* Broker：代理，用来存储消息，Kafka集群中的每一个服务器都是一个代理（Broker），消费者将从broker拉取订阅的消息
* Producer向Kafka发送消息的进程，生产者会根据topic分布消息
* Consumer实例可以是独立的进程，负责订阅和消费消息
* Consumer Group：同一个Consumer Group中的Consumers，Kafka将相应Topic中的每个消息只发送给其中一个Consumer

**Zookeeper 架构：**



Zookeeper 有三种运行模式：单机模式、伪集群模式和集群模式。（容灾性）

**（只要集群中存在超过一半的机器能够正常工作，那么整个集群就能够正常对外服务。）**

参考link：

Kafka 官网：<https://kafka.apache.org/>

Zookeeper 官网：<https://zookeeper.apache.org/>

Prometheus 官网：<https://prometheus.io/docs/introduction/glossary/>

Prometheus 实战：<https://songjiayang.gitbooks.io/prometheus/content/>

Tutorialspoint Kafka教程:<https://www.tutorialspoint.com/apache_kafka/>

知乎Kafka介绍：<https://zhuanlan.zhihu.com/p/43843796>

知乎大白话 + 13 张图解 Kafka：<https://zhuanlan.zhihu.com/p/103276657>

YouTube: Kafka and Zookeeper get started:

<https://www.youtube.com/watch?v=VbSRS7kG5Cw>

知乎不懂 Zookeeper？没关系，看这篇就够了：<https://zhuanlan.zhihu.com/p/98070354>

知乎Kafka\_exporter 部署实战：<https://zhuanlan.zhihu.com/p/57704357>

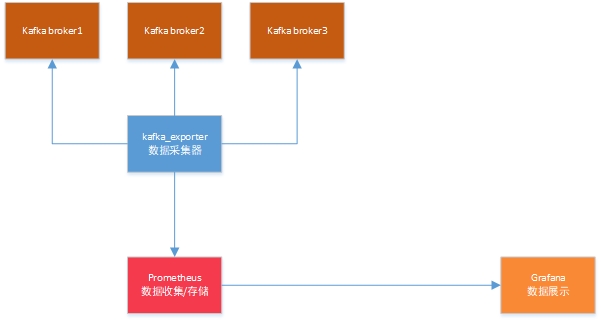
CNBlog构建高可用Zookeeper集群：<https://www.cnblogs.com/cyfonly/p/5626532.html>

Prometheus with hot reload: <http://www.songjiayang.com/posts/2016-09-19-prometheuswith-hot-reload/>

Kafka性能测试实例<https://www.cnblogs.com/smartloli/p/10093838.html>

**二. 1. 测试验证，代码归档 Expoter**

**结构：**



**需要组件：**Prometheus：时序数据库，按时间保存监控历史数据。语言：Go

Grafana：metrics 可视化系统

Kafka Exporter: jmx\_exporter-0.13.1

**验证环境：**Prometheus: prometheus-2.20.0.linux-amd64

Grafana: grafanna\_7.1.1\_amd\_64

Kafka: kafka\_2.12-2.5.0

Zookeeper: apache-zookeeper-3.6.1

Kafka Exporter:

JMX（官方）：GitHub: <https://github.com/prometheus/jmx_exporter>

GOLang：GitHub：<https://github.com/danielqsj/kafka_exporter#run-binary>

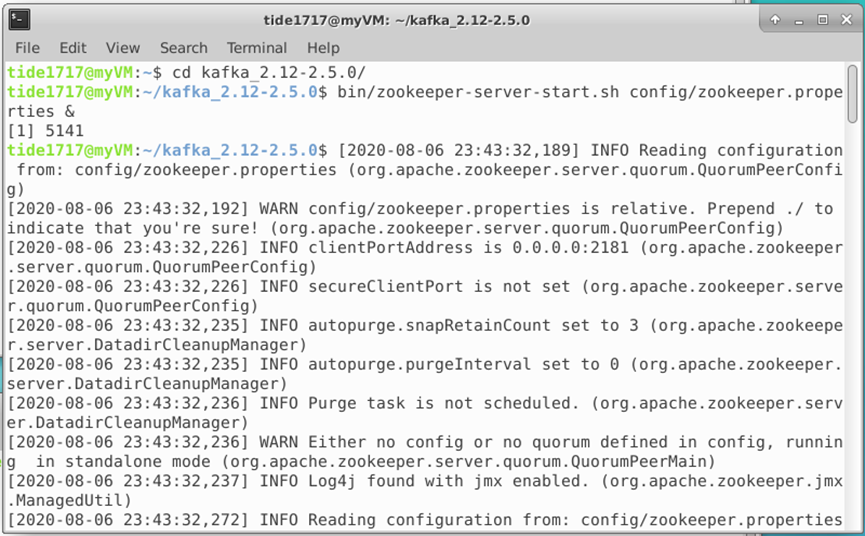
下载Zookeeper 及 Kafka后 版本如上

进入Kafka

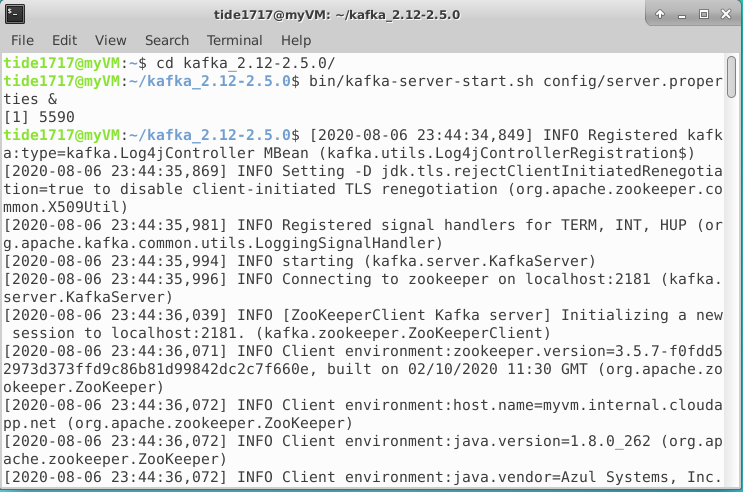
cd kafka\_2.13-2.6.0

运行 Zookeeper及Kafka

> bin/zookeeper-server-start.sh config/zookeeper.properties



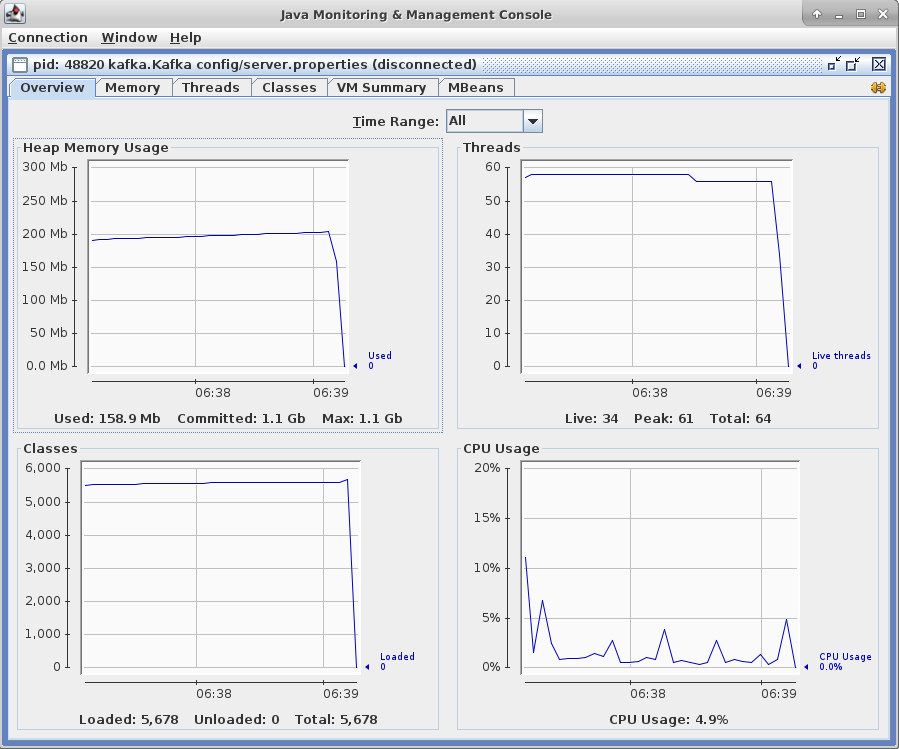
> bin/kafka-server-start.sh config/server.properties



通过 JConsole 得到如下监控画面

运行

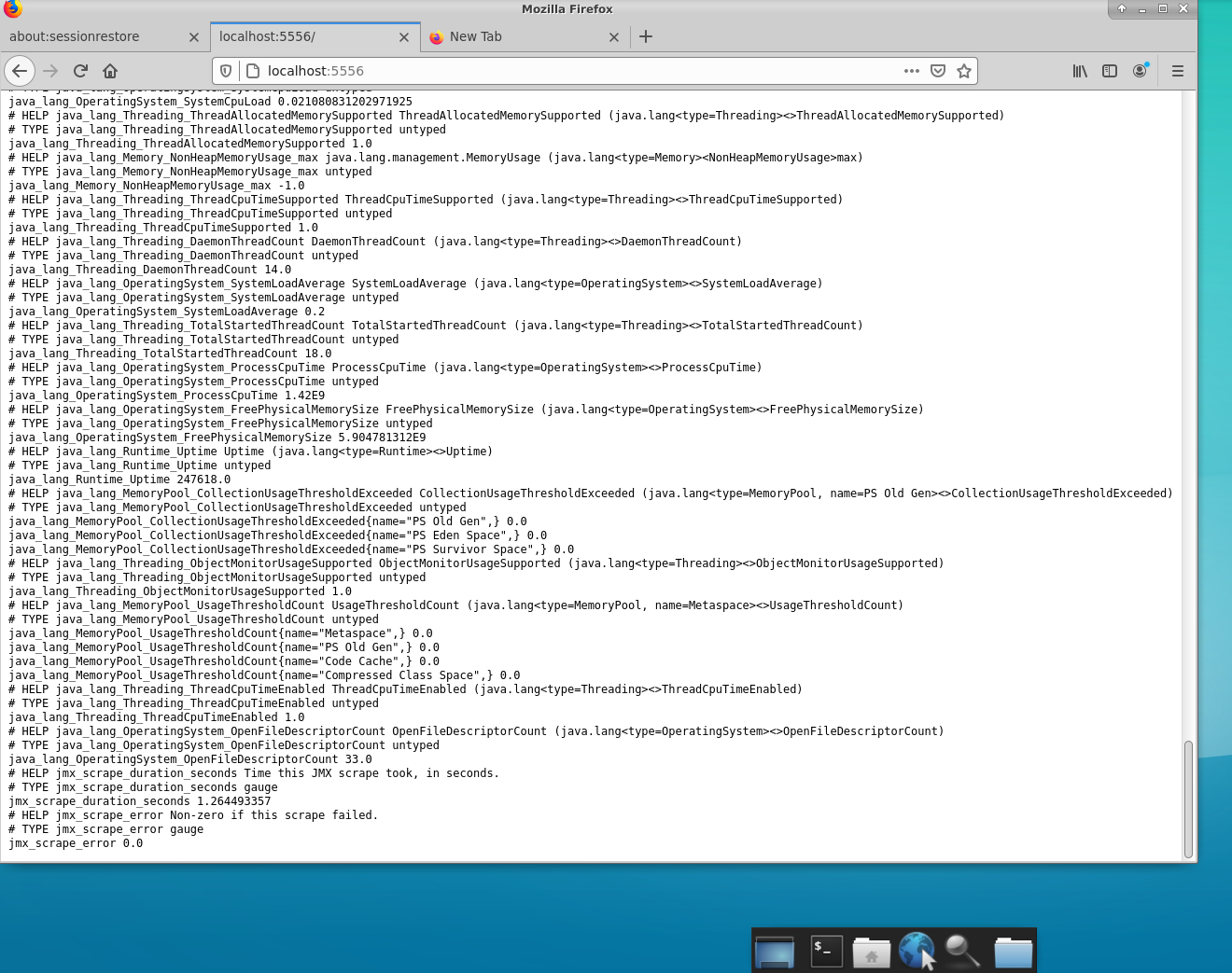
JConsole



运行

java -Dcom.sun.management.jmxremote.ssl=false -Dcom.sun.management.jmxremote.authenticate=false -Dcom.sun.management.jmxremote.port=5556-jar jmx\_prometheus\_httpserver/target/jmx\_prometheus\_httpserver-${version}-jar-with-dependencies.jar 5556 example\_configs/httpserver\_sample\_config.yml

通过localhost：5556 验证 jmx\_exporter运行成功

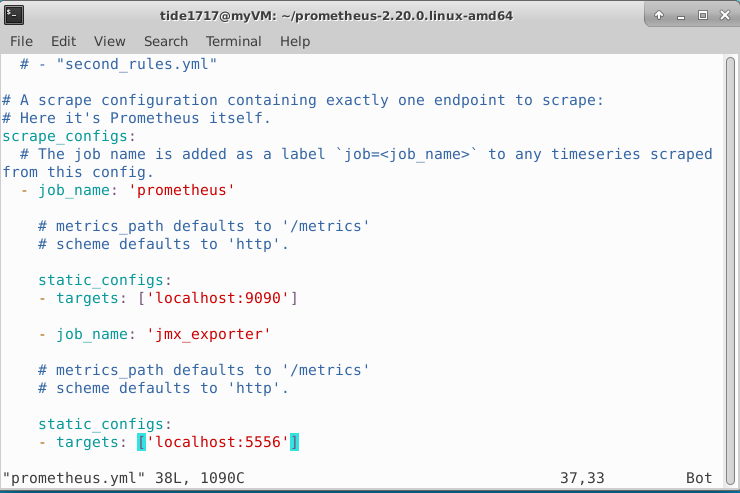


运行Prometheus

cd prometheus-2.20.0.linux-amd64/

vi prometheus.yml

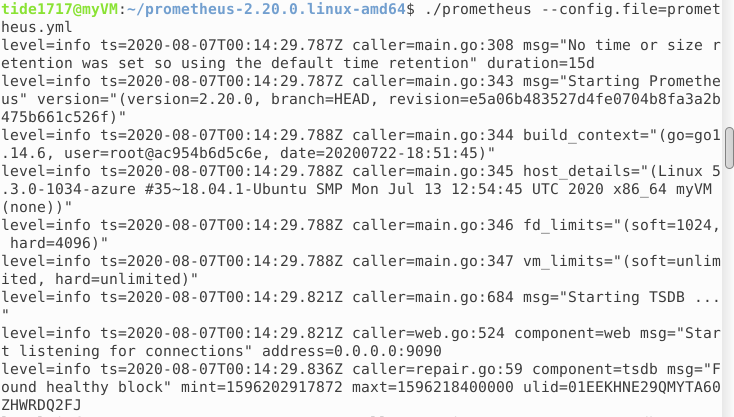
修改 prometheus.yml文件



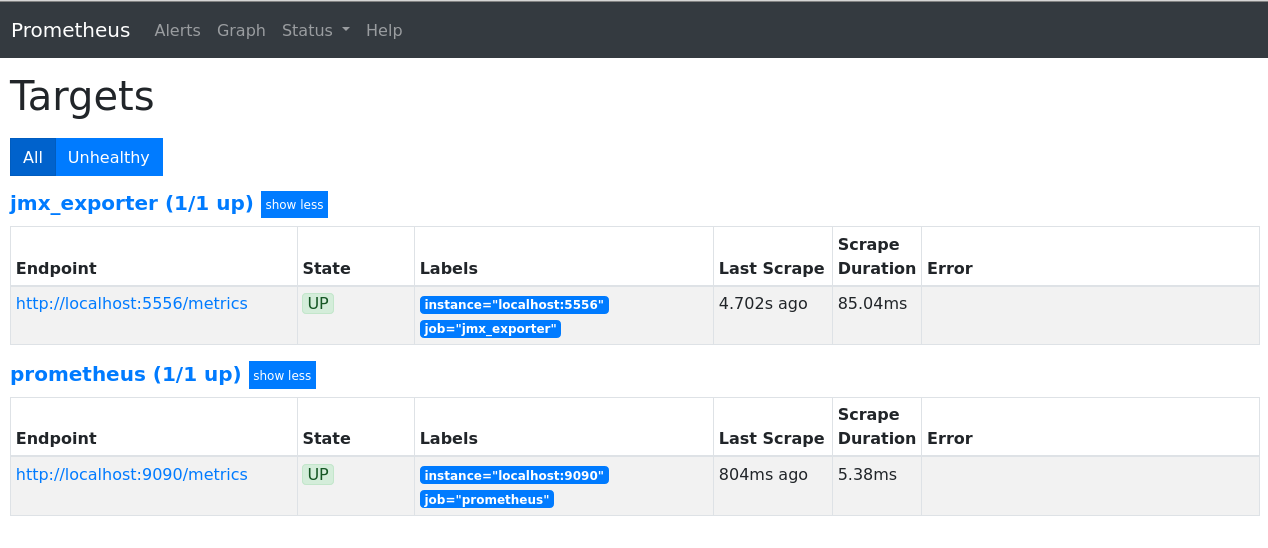
运行Prometheus

./prometheus --config.file=prometheus.yml

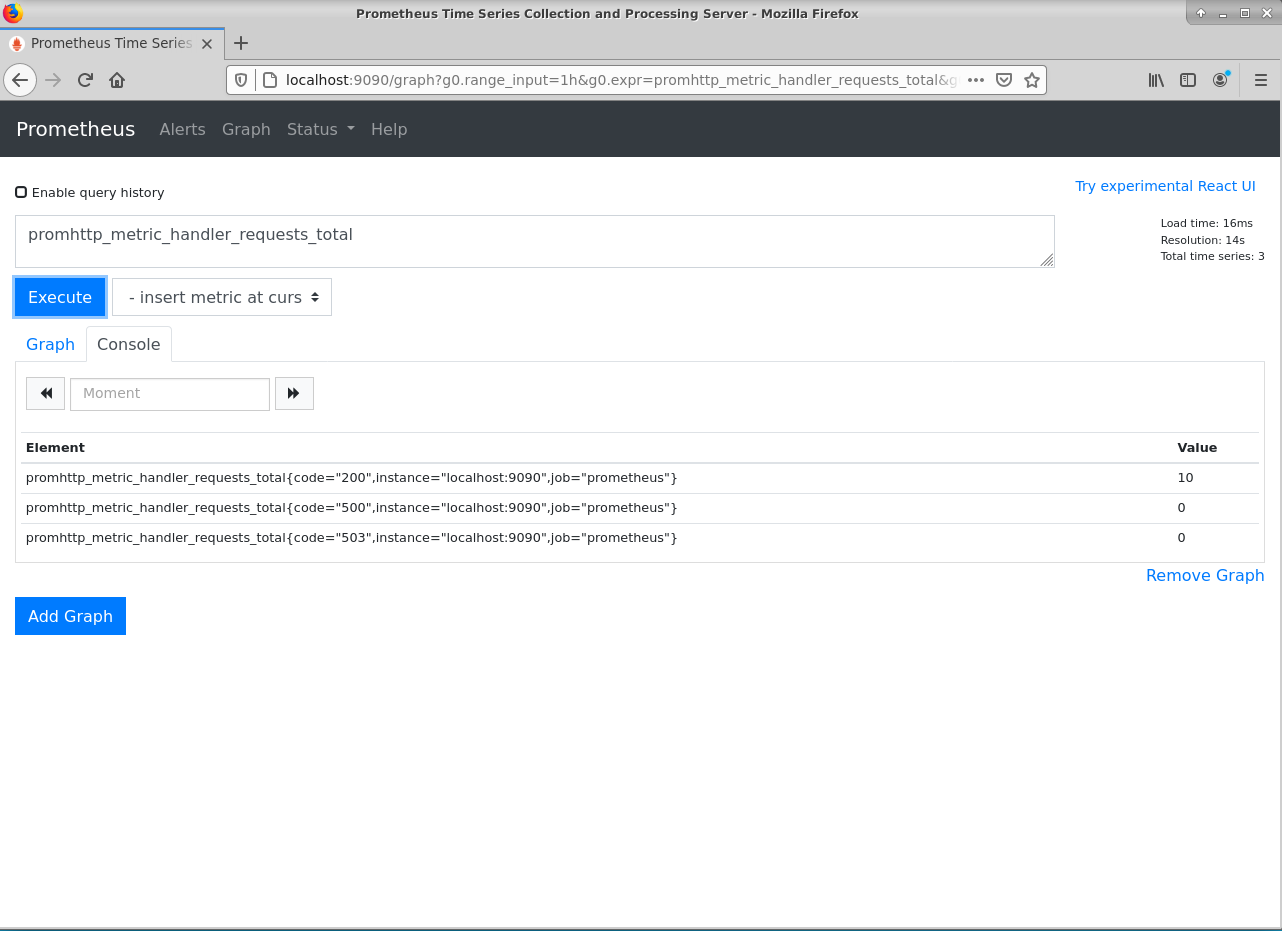
得到



通过 localhost:9090 验证



在Targets中可以发现有两个在运行 为jmx\_exporter 和 Prometheus自身



关于:

Prometheus Query:<https://prometheus.io/docs/prometheus/latest/querying/basics/>

下载Grafana

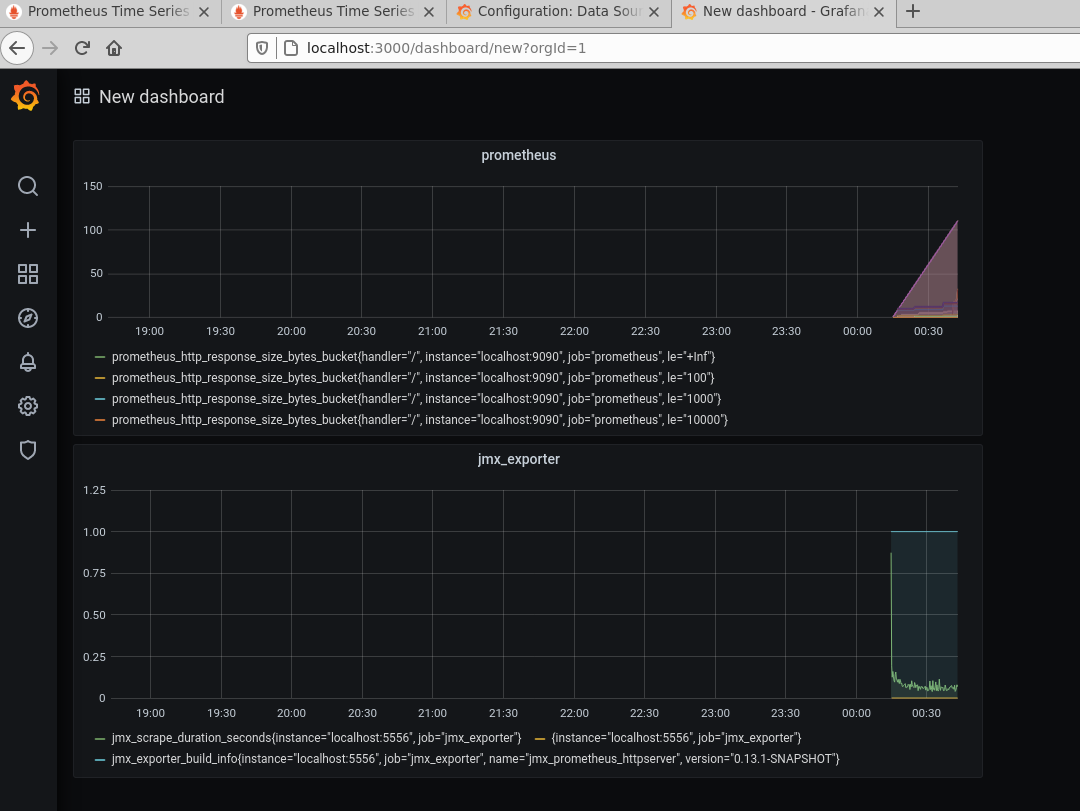
Link：<https://grafana.com/docs/grafana/latest/installation/debian/#2-start-the-server>

sudo apt-get install -y adduser libfontconfig1

wget https://dl.grafana.com/oss/release/grafana\_7.1.3\_amd64.deb

sudo dpkg -i grafana\_7.1.3\_amd64.deb

通过 localhost:3000验证



验证成功。