

### 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) exporter 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

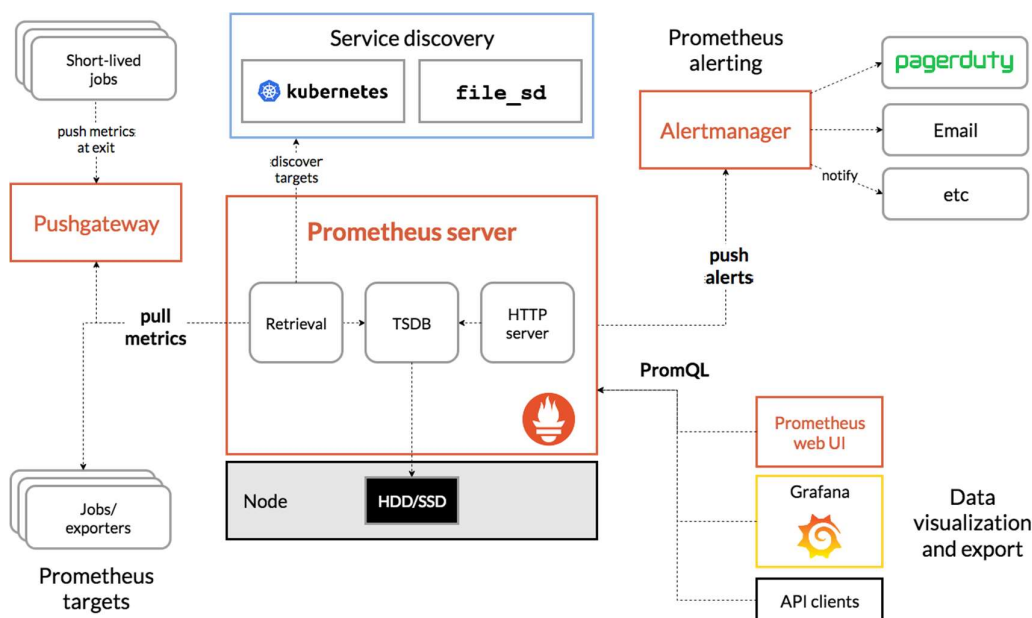
Prometheus joined the Cloud Native Computing Foundation in 2016 as the second hosted project, after Kubernetes.

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 [
```

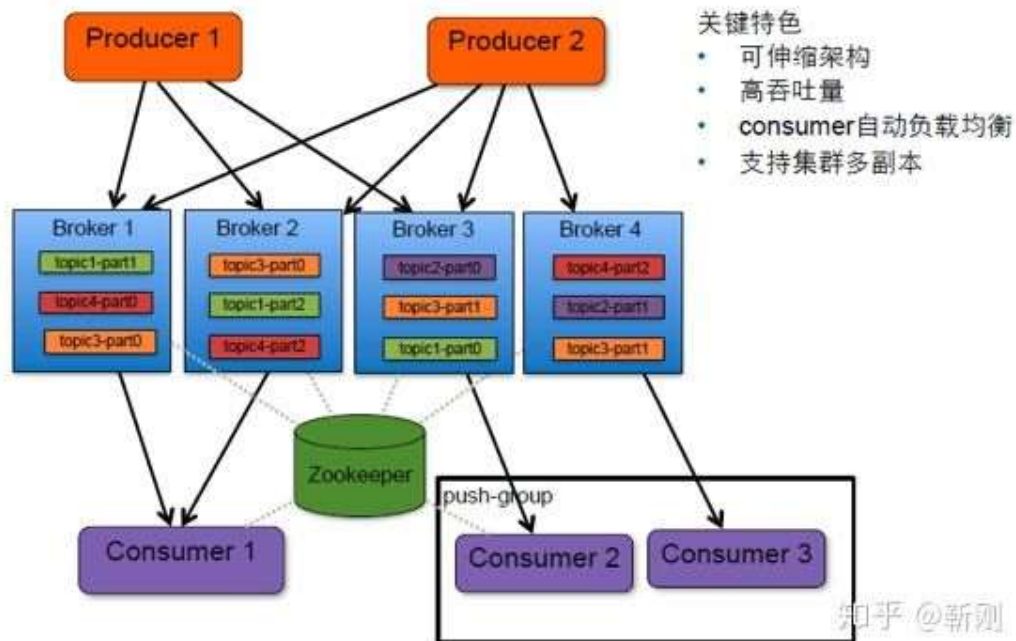
```
"{" label_name "=" "" label_value "" { "," label_name "=" "" label_value "" }
[ "," ] "]" value [ timestamp ]
```

#### 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:

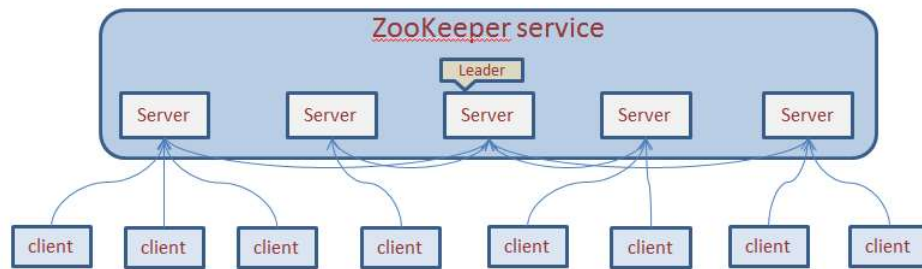
- 采样数据的总和应表示为 `x_sum`。
- 采样数据的总量应表示为 `x_count`。
- summary 类型的采样数据的 quantile 应表示为 `x{quantile="y"}`。
- histogram 类型的采样分区统计数据将表示为 `x_bucket{le="y"}`。
- histogram 类型的采样必须包含 `x_bucket{le="+Inf"}`, 它的值等于 `x_count` 的值。
- summary 和 histogram 中 quantile 和 le 必需按从小到大顺序排列。

#### 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/](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>

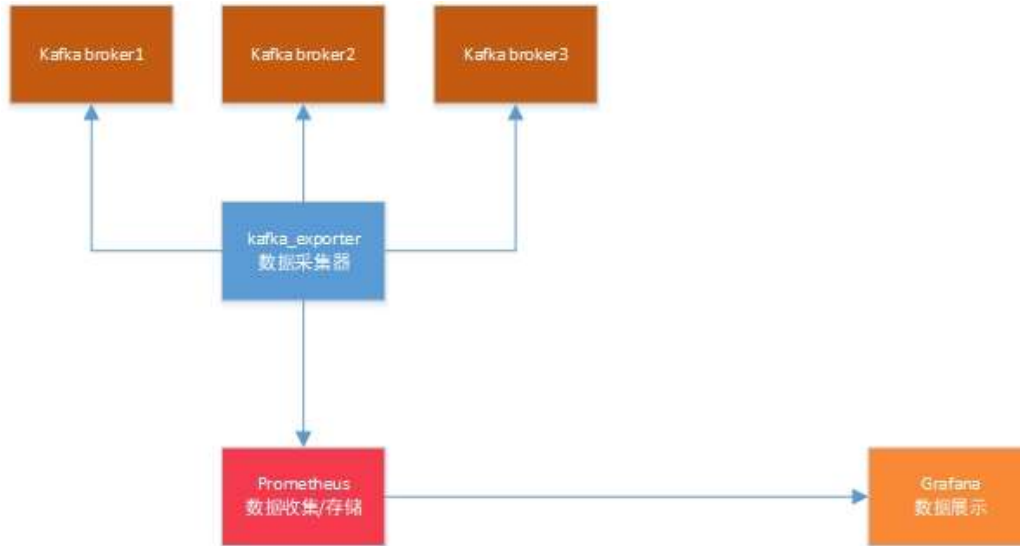
CNBlog 构建高可用 Zookeeper 集群:  
<https://www.cnblogs.com/cyfonly/p/5626532.html>

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/smartlooli/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](https://github.com/prometheus/jmx_exporter)

GOLang：GitHub: [https://github.com/danielqsj/kafka\\_exporter#run-binary](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
```

```
tide1717@myVM: ~/kafka_2.12-2.5.0
File Edit View Search Terminal Help
tide1717@myVM:~$ cd kafka_2.12-2.5.0/
tide1717@myVM:~/kafka_2.12-2.5.0$ bin/zookeeper-server-start.sh config/zookeeper.properties &
[1] 5141
tide1717@myVM:~/kafka_2.12-2.5.0$ [2020-08-06 23:43:32,189] INFO Reading configuration from: config/zookeeper.properties (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2020-08-06 23:43:32,192] WARN config/zookeeper.properties is relative. Prepend ./ to indicate that you're sure! (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2020-08-06 23:43:32,226] INFO clientPortAddress is 0.0.0.0:2181 (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2020-08-06 23:43:32,226] INFO secureClientPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2020-08-06 23:43:32,235] INFO autopurge.snapRetainCount set to 3 (org.apache.zookeeper.server.DatadirCleanupManager)
[2020-08-06 23:43:32,235] INFO autopurge.purgeInterval set to 0 (org.apache.zookeeper.server.DatadirCleanupManager)
[2020-08-06 23:43:32,236] INFO Purge task is not scheduled. (org.apache.zookeeper.server.DatadirCleanupManager)
[2020-08-06 23:43:32,236] WARN Either no config or no quorum defined in config, running in standalone mode (org.apache.zookeeper.server.quorum.QuorumPeerMain)
[2020-08-06 23:43:32,237] INFO Log4j found with jmx enabled. (org.apache.zookeeper.jmx.ManagedUtil)
[2020-08-06 23:43:32,272] INFO Reading configuration from: config/zookeeper.properties
```

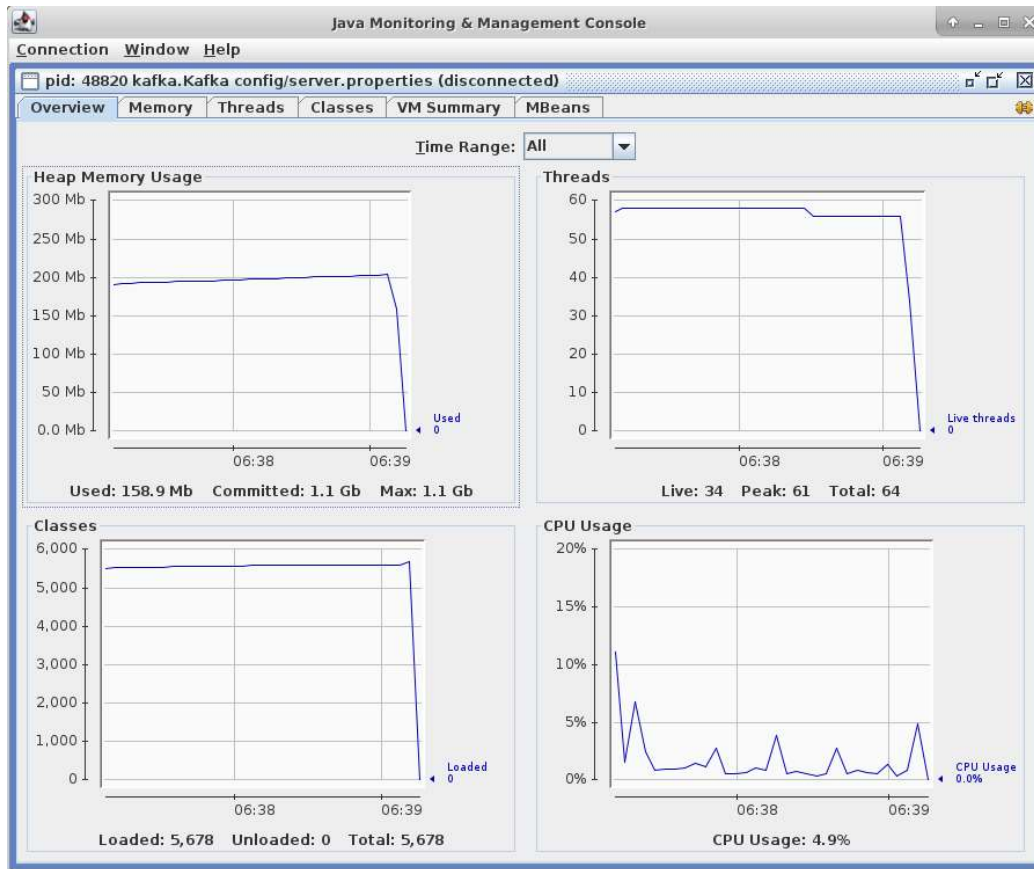
> bin/kafka-server-start.sh config/server.properties

```
tide1717@myVM: ~/kafka_2.12-2.5.0
File Edit View Search Terminal Help
tide1717@myVM:~$ cd kafka_2.12-2.5.0/
tide1717@myVM:~/kafka_2.12-2.5.0$ bin/kafka-server-start.sh config/server.properties &
[1] 5590
tide1717@myVM:~/kafka_2.12-2.5.0$ [2020-08-06 23:44:34,849] INFO Registered kafka:type=kafka.Log4jController MBean (kafka.utils.Log4jControllerRegistration$)
[2020-08-06 23:44:35,869] INFO Setting -D jdk.tls.rejectClientInitiatedRenegotiation=true to disable client-initiated TLS renegotiation (org.apache.zookeeper.common.X509Util)
[2020-08-06 23:44:35,981] INFO Registered signal handlers for TERM, INT, HUP (org.apache.kafka.common.utils.LoggingSignalHandler)
[2020-08-06 23:44:35,994] INFO starting (kafka.server.KafkaServer)
[2020-08-06 23:44:35,996] INFO Connecting to zookeeper on localhost:2181 (kafka.server.KafkaServer)
[2020-08-06 23:44:36,039] INFO [ZooKeeperClient Kafka server] Initializing a new session to localhost:2181. (kafka.zookeeper.ZooKeeperClient)
[2020-08-06 23:44:36,071] INFO Client environment:zookeeper.version=3.5.7-f0fdd52973d373ffd9c86b81d99842dc2c7f660e, built on 02/10/2020 11:30 GMT (org.apache.zookeeper.ZooKeeper)
[2020-08-06 23:44:36,072] INFO Client environment:host.name=myvm.internal.cloudapp.net (org.apache.zookeeper.ZooKeeper)
[2020-08-06 23:44:36,072] INFO Client environment:java.version=1.8.0_262 (org.apache.zookeeper.ZooKeeper)
[2020-08-06 23:44:36,072] INFO Client environment:java.vendor=Azul Systems, Inc.
```

通过 JConsole 得到如下监控画面

运行

JConsole

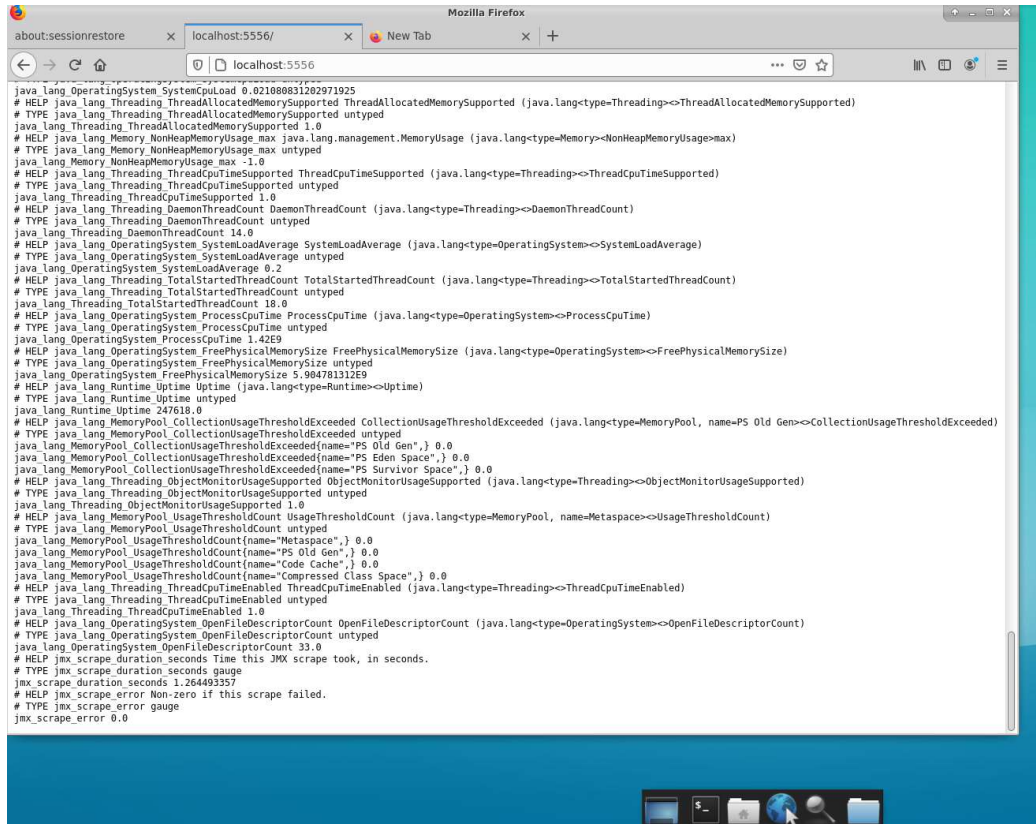


## 运行

```
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



```
tide1717@myVM: ~/prometheus-2.20.0.linux-amd64
File Edit View Search Terminal Help
# - "second_rules.yml"

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped
  # from this config.
  - job_name: 'prometheus'

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
      - targets: ['localhost:9090']

      - job_name: 'jmx_exporter'

        # metrics_path defaults to '/metrics'
        # scheme defaults to 'http'.

        static_configs:
          - targets: ['localhost:5556']

"prometheus.yml" 38L, 1090C 37,33 Bot
```

运行 Prometheus

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

得到

```
tide1717@myVM:~/prometheus-2.20.0.linux-amd64$ ./prometheus --config.file=prometheus.yml
level=info ts=2020-08-07T00:14:29.787Z caller=main.go:308 msg="No time or size retention was set so using the default time retention" duration=15d
level=info ts=2020-08-07T00:14:29.787Z caller=main.go:343 msg="Starting Prometheus" version="(version=2.20.0, branch=HEAD, revision=e5a06b483527d4fe0704b8fa3a2b475b661c526f)"
level=info ts=2020-08-07T00:14:29.788Z caller=main.go:344 build_context="(go=go1.14.6, user=root@ac954b6d5c6e, date=20200722-18:51:45)"
level=info ts=2020-08-07T00:14:29.788Z caller=main.go:345 host_details="(Linux 5.3.0-1034-azure #35~18.04.1-Ubuntu SMP Mon Jul 13 12:54:45 UTC 2020 x86_64 myVM (none))"
level=info ts=2020-08-07T00:14:29.788Z caller=main.go:346 fd_limits="(soft=1024, hard=4096)"
level=info ts=2020-08-07T00:14:29.788Z caller=main.go:347 vm_limits="(soft=unlimited, hard=unlimited)"
level=info ts=2020-08-07T00:14:29.821Z caller=main.go:684 msg="Starting TSDB ..."
level=info ts=2020-08-07T00:14:29.821Z caller=web.go:524 component=web msg="Start listening for connections" address=0.0.0.0:9090
level=info ts=2020-08-07T00:14:29.836Z caller=repair.go:59 component=tsdb msg="Found healthy block" mint=1596202917872 maxt=1596218400000 ulid=01EEKHNE29QMYTA60ZHWRDQ2FJ
```

通过 localhost:9090 验证

Prometheus Alerts Graph Status Help					
Targets					
All Unhealthy					
jmx_exporter (1/1 up) show less					
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:5556/metrics	UP	instance="localhost:5556" job="jmx_exporter"	4.702s ago	85.04ms	
prometheus (1/1 up) show less					
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9090/metrics	UP	instance="localhost:9090" job="prometheus"	804ms ago	5.38ms	

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

prometheus Time Series Collection and Processing Server - Mozilla Firefox

localhost:9090/graph?g0.range\_input=1h&g0.expr=promhttp\_metric\_handler\_requests\_total&...

Prometheus Alerts Graph Status Help

Enable query history Try experimental React UI

promhttp\_metric\_handler\_requests\_total

Execute - Insert metric at curs

Graph Console

Moment

Element	Value
promhttp_metric_handler_requests_total{code="200",instance="localhost:9090",job="prometheus"}	10
promhttp_metric_handler_requests_total{code="500",instance="localhost:9090",job="prometheus"}	0
promhttp_metric_handler_requests_total{code="503",instance="localhost:9090",job="prometheus"}	0

Load time: 16ms  
Resolution: 14s  
Total time series: 3

Remove Graph

Add Graph

关于:

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

下载 Grafana

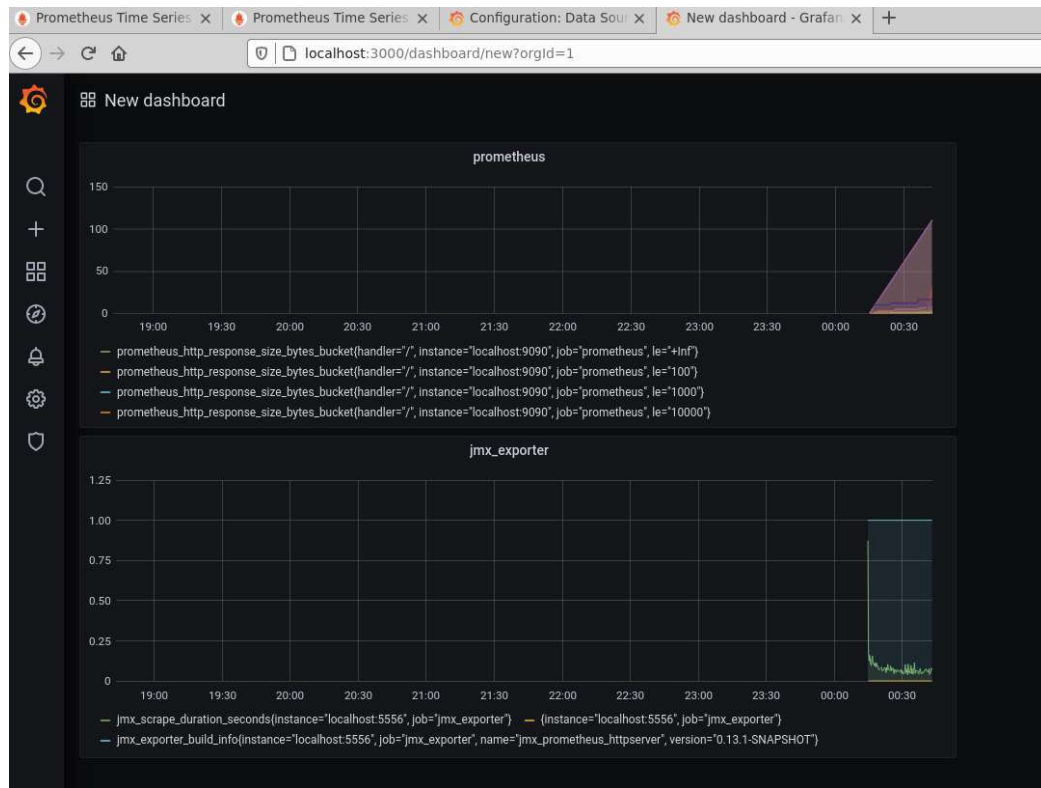
Link: <https://grafana.com/docs/grafana/latest/installation/debian/#2-start-the-server>

```
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。