99使用 OpenTelemetry 和 Grafana Tempo 升级您的 Tracing平台

如果您希望改进跟踪平台，增加跟踪采样，同时保持成本优化，特别是如果您是开源爱好者，那么本文适合您。

Traces 是一些最重要的遥测信号，现在是我们给予它们应有的重视的时候了。维护和改进 tracing 平台可能会非常繁重，而且很快就会变得昂贵，因此我想向您介绍强大的二人组：OpenTelemetry 和 Grafana Tempo。这种强大的组合允许您从各种来源收集 trace 信号，并将它们有效地存储在 Azure Blob 存储、S3 或 GCS 支持的后端中。凭借其令人印象深刻的查询功能，该解决方案将您的 tracing 平台提升到一个新的水平。

在本文中，我将提供一些见解，并为您指出有关以下主题的正确文档：

1. 如何部署 OpenTelemetry Collector
2. 如何部署Grafana Tempo
3. 充分利用这两个工具

部署 OpenTelemetry 收集器非常简单。您可以在此处(https://github.com/open-telemetry/opentelemetry-helm-charts)找到社区 chart 和说明。为了利用 OpenTelemetry 收集器的最佳功能，一般来说，我建议使用 opentelemetry-collector-contrib 镜像。您可以在此处（https://github.com/open-telemetry/opentelemetry-collector-contrib）找到可与 contrib 镜像一起使用的接收器、导出器和处理器的列表。但要将 tempo 用作导出器，您也可以使用 opentelemetry-collector 镜像，因为 otlp/http 导出器是默认导出器之一。

要接收 traces ，您需要配置正确的接收器，在本示例中，我将收集器配置为接收来自 jaeger、otlp 和 kafka 的日志。

```
receivers:
    jaeger:
      protocols:
        grpc:
          endpoint: ${env:MY_POD_IP}:14250
        thrift_http:
          endpoint: ${env:MY_POD_IP}:14268
        thrift_compact:
          endpoint: ${env:MY_POD_IP}:6831
    otlp:
      protocols:
        grpc:
          endpoint: ${env:MY_POD_IP}:4317
        http:
          endpoint: ${env:MY_POD_IP}:4318

   kafka:
    protocol_version: 2.0.0
    brokers: broker:9092
    topic: tracing_kafka
```

其次，我们需要配置一个导出器以将 trace 发送到 Tempo。在本例中，我选择了 otlp/http 导出器

```
 otlphttp:
    endpoint: https://tempo-gateway:4318
```

最后，将接收者和导出者添加到 trace 管道中

```
pipelines:
      logs:
        exporters:
          - logging
        processors:
          - memory_limiter
          - batch
        receivers:
          - otlp
      metrics:
        exporters:
          - logging
        processors:
          - memory_limiter
          - batch
        receivers:
          - otlp
          - prometheus
      traces: # here, we add the receivers and exporters accordingly
        exporters:
          - otlphttp
        processors:
          - memory_limiter
          - batch
        receivers:
          - otlp
          - jaeger
          - kafka
```

就像这样，您就可以接收 trace 并将其发送到 Tempo 后端。现在开始 Tempo 的实现.....

什么是 Grafana Tempo？它是一个分布式追踪后端，只需要对象存储即可运行。它与 Azure 存储帐户、AWS S3 存储桶、GCS 等兼容。Tempo 具有高度可扩展性和高性能。部署 Grafana Tempo 也非常简单。您可以在此处(https://github.com/grafana/helm-charts/tree/main/charts/tempo-distributed)找到社区 chart 以及说明。我推荐使用 tempo-distributed chart，也就是微服务模式。但如果您想使用单一二进制 tempo，您可以在此处(https://github.com/grafana/helm-charts/tree/main/charts/tempo)找到chart。

首先，出于测试目的，您可以使用 chart 中默认配置的本地存储。但如果您希望拥有更持久的存储、更高的保留率，那么您需要启用您最喜欢的云提供商的存储。

```
#To configure a different storage backend instead of local storage:
storage:
  trace:
    backend: azure #can be s3 or gcs as well
    azure:
      container_name:
      storage_account_name:
      storage_account_key:
```

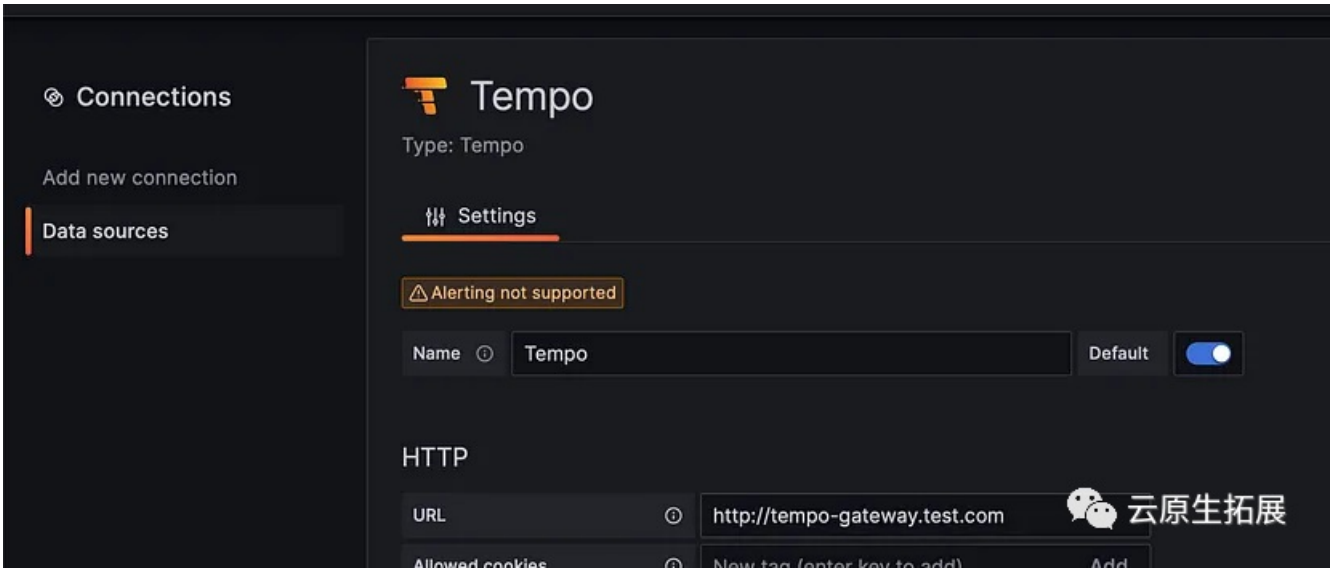
其次，我强烈建议启用网关，您可以将其用作 trace 的入口点，还可以将 Tempo 添加为 Grafana 中的数据源。

```
gateway:
  # -- Specifies whether the gateway should be enabled
  enabled: true
```

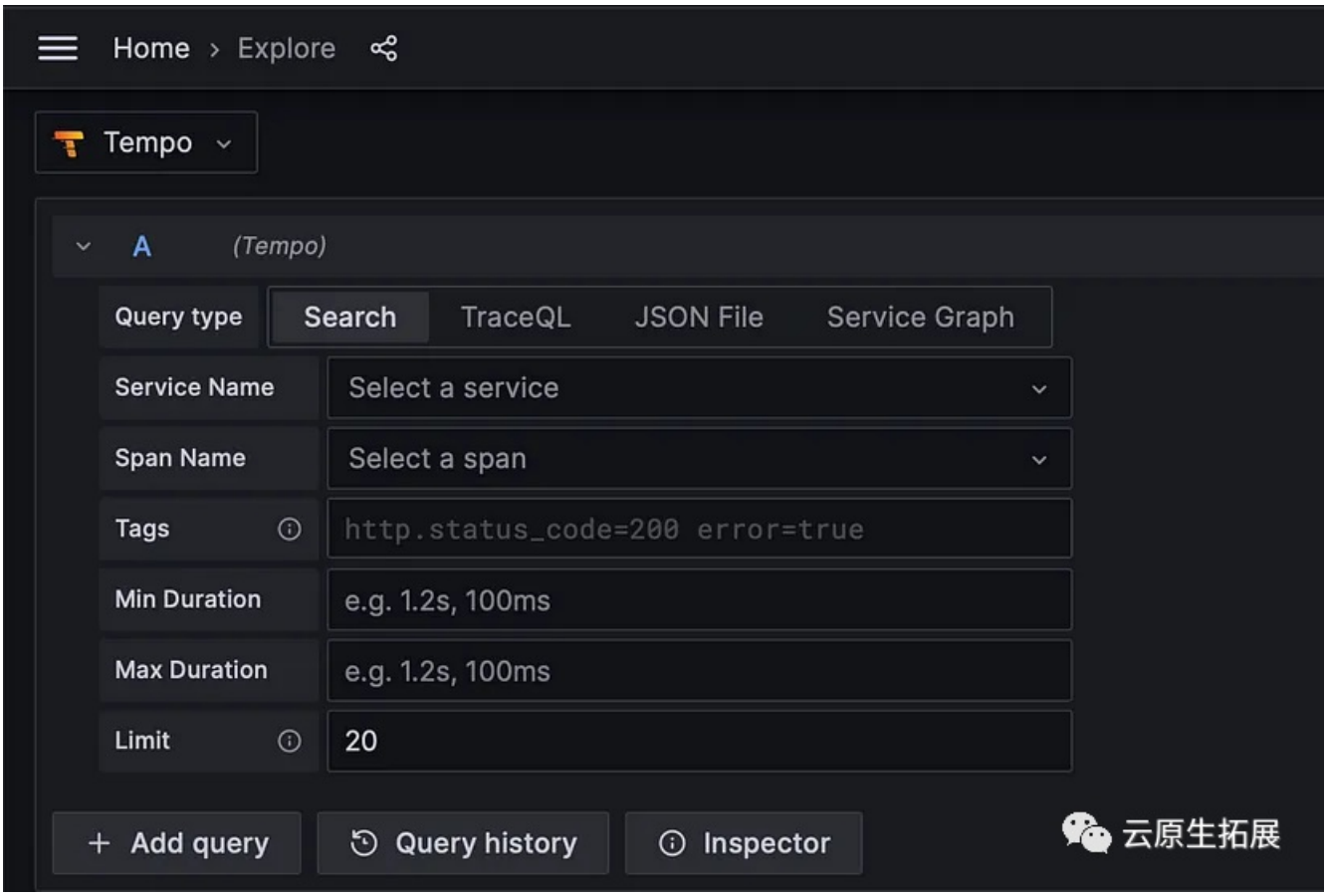
解决方法是使用分发器作为入口点来接收 trace，并使用 queryFrontend 将 Tempo 添加为 Grafana 中的数据源。

```
queryFrontend:
  query:
    # -- Required for grafana version <7.5 for compatibility with jaeger-ui. Doesn't work on ARM arch
    enabled: true
distributor:
  enabled: true
```

一旦您启动并运行 Tempo，您就可以将其添加为 Grafana 实例中的数据源。可以手动完成，如下图所示。或者您可以通过图表添加它，如下所示。



保存数据源后，您可以通过 Grafana 中的资源管理器查看它。



我通过使用这两个工具的经验学到了一些东西，我相信这些东西确实对跟踪平台产生了影响。

1. OpenTelemetry 收集器具有“内置”缓冲区。它们以队列的形式出现，其大小取决于分配的内存。不仅如此，您还可以选择将数据保存在本地存储中，例如持久卷。如果您的任何一个或多个收集器发生故障，也不会丢失数据。

2. OpenTelemetry 收集器具有内置批处理处理器，可对遥测信号进行批处理。此外，使用 Tempo 时，系统接收和处理数据的能力依赖于计算能力。只要您分配了足够的资源或实现了有效的自动缩放系统，您就可以发送大量的跨度，而不需要诸如Kafka或其他排队系统之类的中介技术。

3. 在 Grafana Tempo 中启用metrics_generator 将允许您获取 Prometheus 格式的一些关键指标数据。你可以在这(https://grafana.com/docs/tempo/latest/metrics-generator/span_metrics/)里读更多关于它的内容。

Metric	Type	Labels	Description
traces_spanmetrics_latency	Histogram	Dimensions	Duration of the span
traces_spanmetrics_calls_total	Counter	Dimensions	Total count of the span
traces_spanmetrics_size_total	Counter	Dimensions	Total size of spans inge

云原生拓展

4. trace 中的 Prometheus 指标也用于生成服务图。这些图表检查跟踪，寻找具有父子关系的跨度，以提供系统的高级概述。这就是为什么代码的检测和处理的数据的质量非常重要。这进一步强化了 OpenTelemetry 和 Tempo 形成强大组合的原因。您可以在此处(https://grafana.com/docs/tempo/latest/metrics-generator/service_graphs/)了解有关服务图的更多信息。

5. 虚拟和水平自动扩展对于您充分利用这两种技术非常重要。

6. 追踪所使用的存储并相应地调整保留量非常重要。

我真诚地希望本文能为您提供宝贵的见解，并且为您的可观测平台提供一些新的想法，为什么不呢？我们非常感谢您对内容的反馈，因此请随时与我们联系。我很想听听你的消息。

在我的下一篇文章中，我将更详细地介绍通过 Opentelemetry 收集器收集追踪和指标的信息，因此请确保您密切关注这些内容。