Loki & Promtail 详解

发布于 2021-10-18 11:04:24 7.6K 0

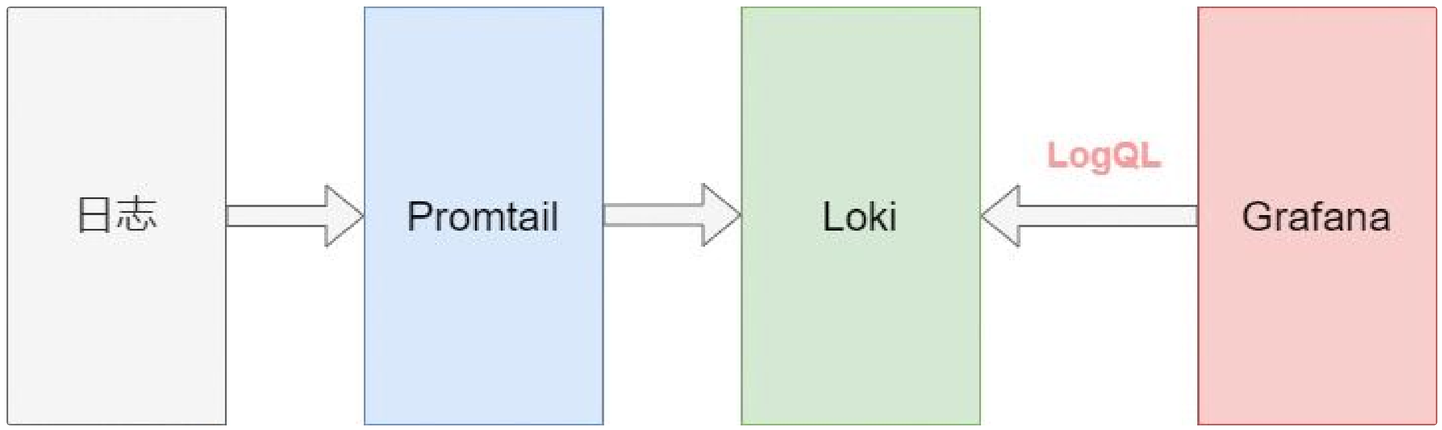
> “主流的日志收集架构一般采用 ELK/EFK/EFLK ，但是这些都比较适合在重量级、需要日志数据清理的场景下使用。
> 云原生环境下， Grafana + Loki + Promtail 横空出世。
> “Like Prometheus, but for logs.”
> ”

整体介绍

Loki 是受 Prometheus 启发的水平可扩展、高可用、多租户日志聚合系统。非常适合采集 Kubernetes Pod 的日志，关键 Loki 还易于操作且更加轻量级（相比 ELK/EFK/EFLK ）。

在 Loki 架构中有以下几个概念：

- Grafana：相当于 EFK 中的 Kibana ， 用于 UI 的展示。
- Loki：相当于 EFK 中的 ElasticSearch ， 用于存储日志和处理查询。
- Promtail：相当于 EFK 中的 Filebeat/Fluentd ， 用于采集日志并将其发送给 Loki 。
- LogQL：Loki 提供的日志查询语言，类似 Prometheus 的 PromQL，而且 Loki 支持 LogQL 查询直接转换为 Prometheus 指标。



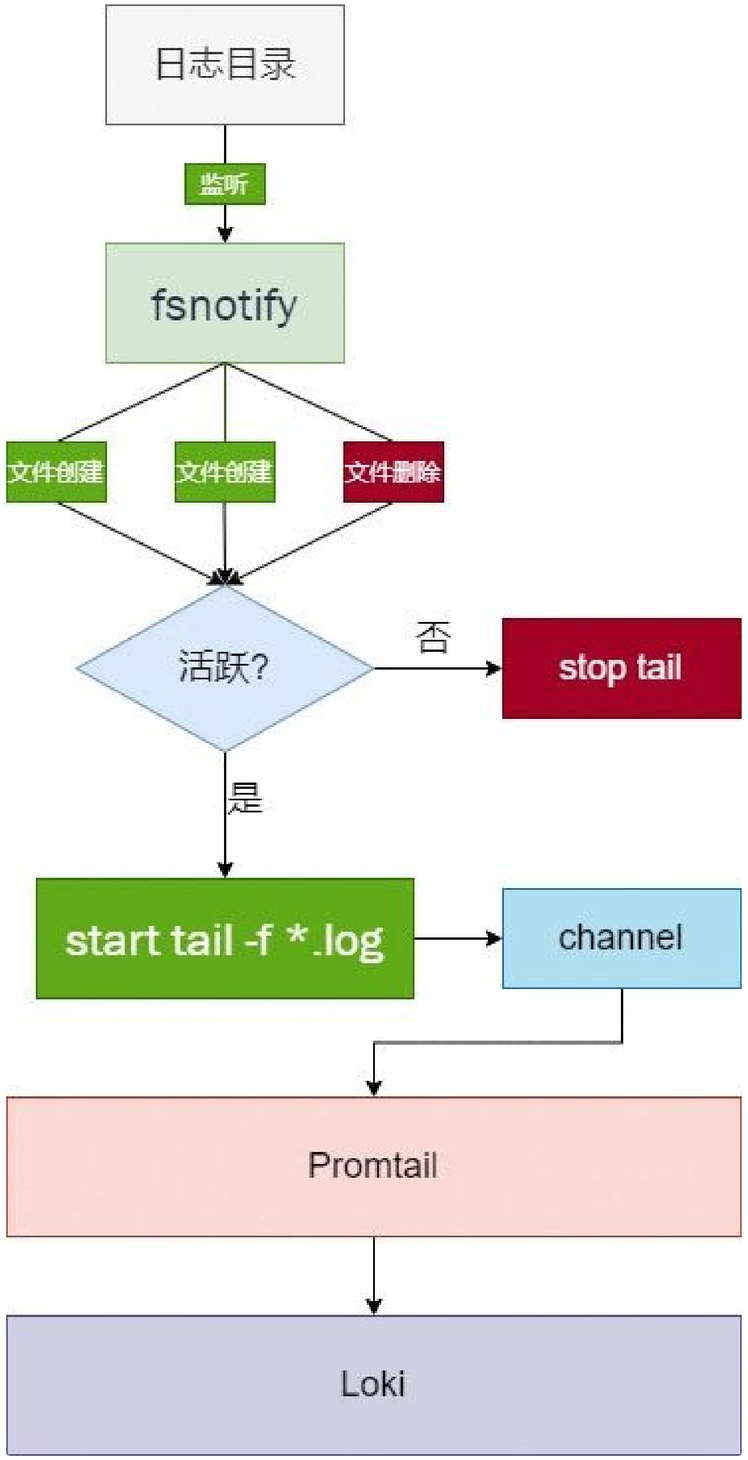
Loki整体架构

Promtail 介绍

Promtail 将本地日志内容传送到 Loki 实例。需要监控的应用程序的机器上都需要部署该组件。

它的主要工作流程可以划分为：

- 使用 fsnotify 监听指定目录下（例如：/var/log/*.log）的文件创建与删除
- 对每个活跃的日志文件起一个 goroutine 进行类似 tail -f 的读取，读取到的内容发送给 channel
- 有一个单独的 goroutine 会读取 channel 中的日志行，分批并附加上标签后推送给 Loki



promtail原理

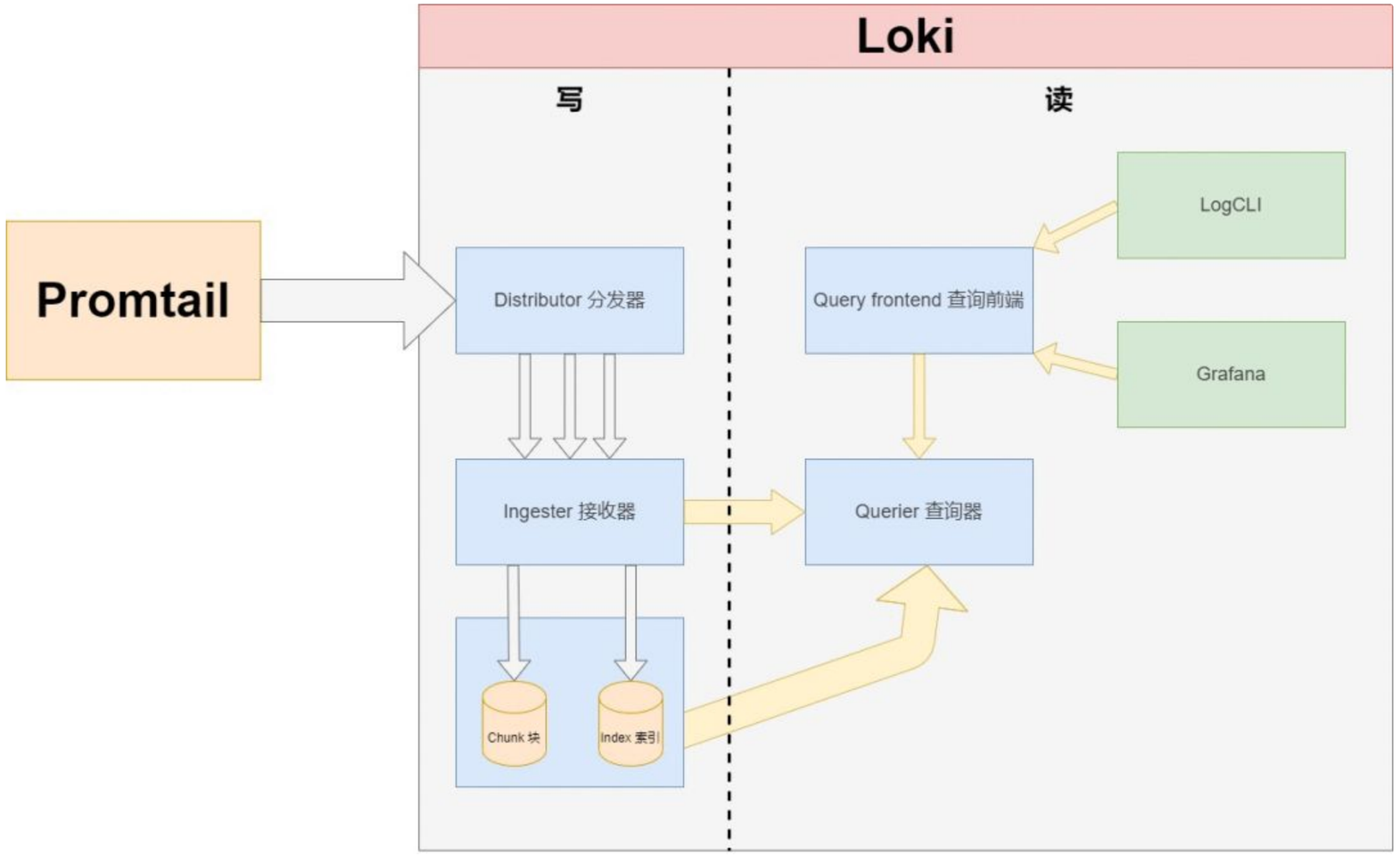
Loki 介绍

Loki 是用来接受、存储、处理、查询日志的集合体。

Loki 采用读写分离架构，关键组件有：

- Distributor 分发器：日志数据传输的“第一站”，Distributor 分发器接收到日志数据后，根据元数据和 hash 算法，将日志分批并行地发送到多个 Ingester 接收器上
- Ingester 接收器：接收器是一个有状态的组件，在日志进入时对其进行 gzip 压缩操作，并负责构建和刷新 chunck 块，当 chunk 块达到一定的数量或者时间后，就会刷新 chunk 块和对应的 Index 索引存储到数据库中
- Querier 查询器：给定一个时间范围和标签选择器，Querier 查询器可以从数据库中查看 Index 索引以确定哪些 chunck 块匹配，并通过 greps 将结果显示出来，它还会直接从 Ingester 接收器获取尚未刷新的最新数据

- Query frontend 查询前端：查询前端是一个可选的组件，运行在 Querier 查询器之前，起到缓存，均衡调度的功能，用于加速日志查询



loki组件通信

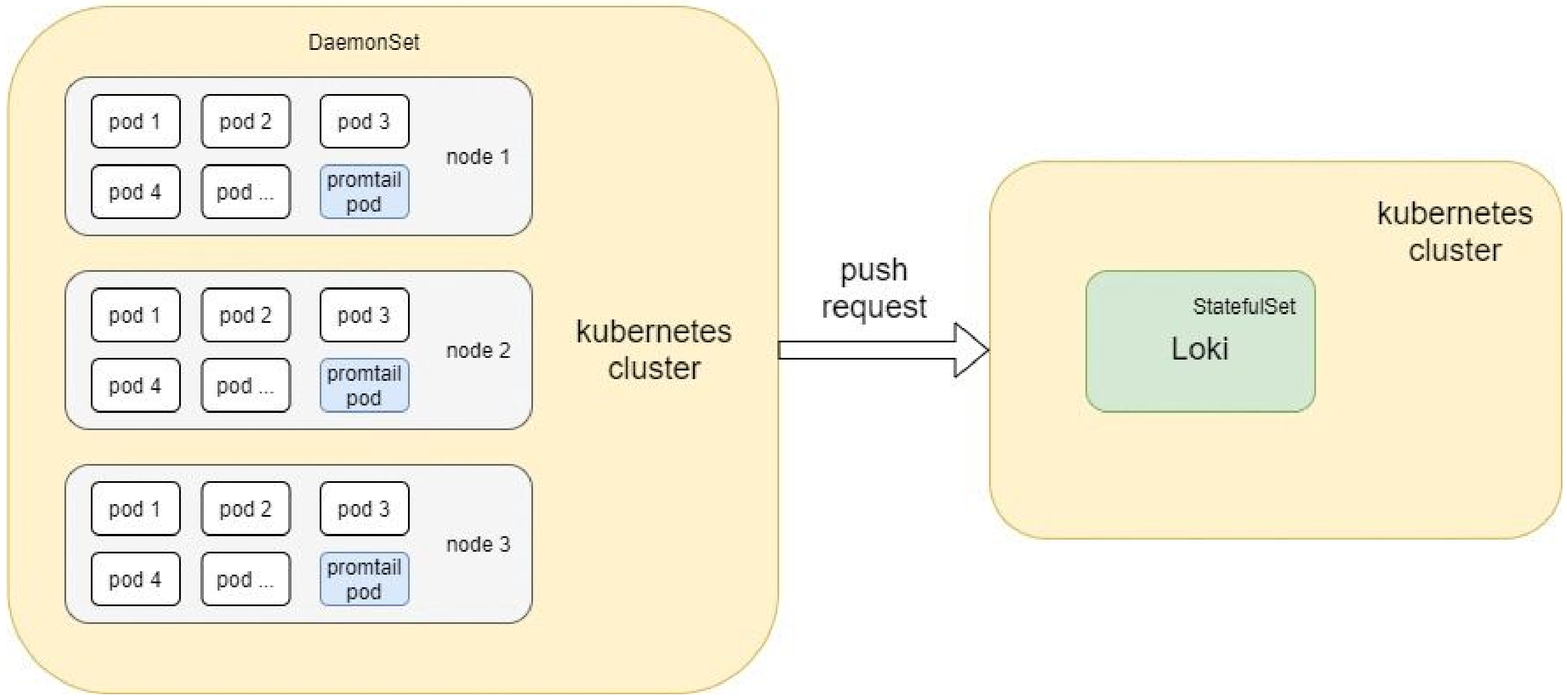
Loki 提供了两种部署方式：

- 单体模式，ALL IN ONE：Loki 支持单一进程模式，可在一个进程中运行所有必需的组件。单进程模式非常适合测试 Loki 或以小规模运行。不过尽管每个组件都以相同的进程运行，但它们仍将通过本地网络相互连接进行组件之间的通信（grpc）。使用 Helm 部署就是采用的该模式。
- 微服务模式：为了实现水平可伸缩性，Loki 支持组件拆分为单独的组件分开部署，从而使它们彼此独立地扩展。每个组件都产生一个用于内部请求的 gRPC 服务器和一个用于外部 API 请求的 HTTP 服务，所有组件都带有 HTTP 服务器，但是大多数只暴露就绪接口、运行状况和指标端点。

Loki组件架构

使用 Helm 部署

以 Helm 部署 Loki (StatefulSet 方式) 和 Promtail（DaemonSet 方式）采集 k8s pod 应用的日志为例




```shell
# 添加 grafana 源
helm repo add grafana https://grafana.github.io/helm-charts

# 创建命名空间
kubectl create ns grafana
kubectl create ns loki

# 部署 grafana，并开启 NodePort 访问
# 用户名 admin
# 密码 kubectl get secret --namespace grafana grafana -o jsonpath="{.data.admin-password}" | base64 --decode ; echo
helm install grafana grafana/grafana --set "service.type=NodePort" -n grafana

# 部署 loki
helm install loki grafana/loki -f loki-config.yaml -n loki

# 部署 promtail
helm install promtail grafana/promtail -f promtail-config.yaml -n loki
```

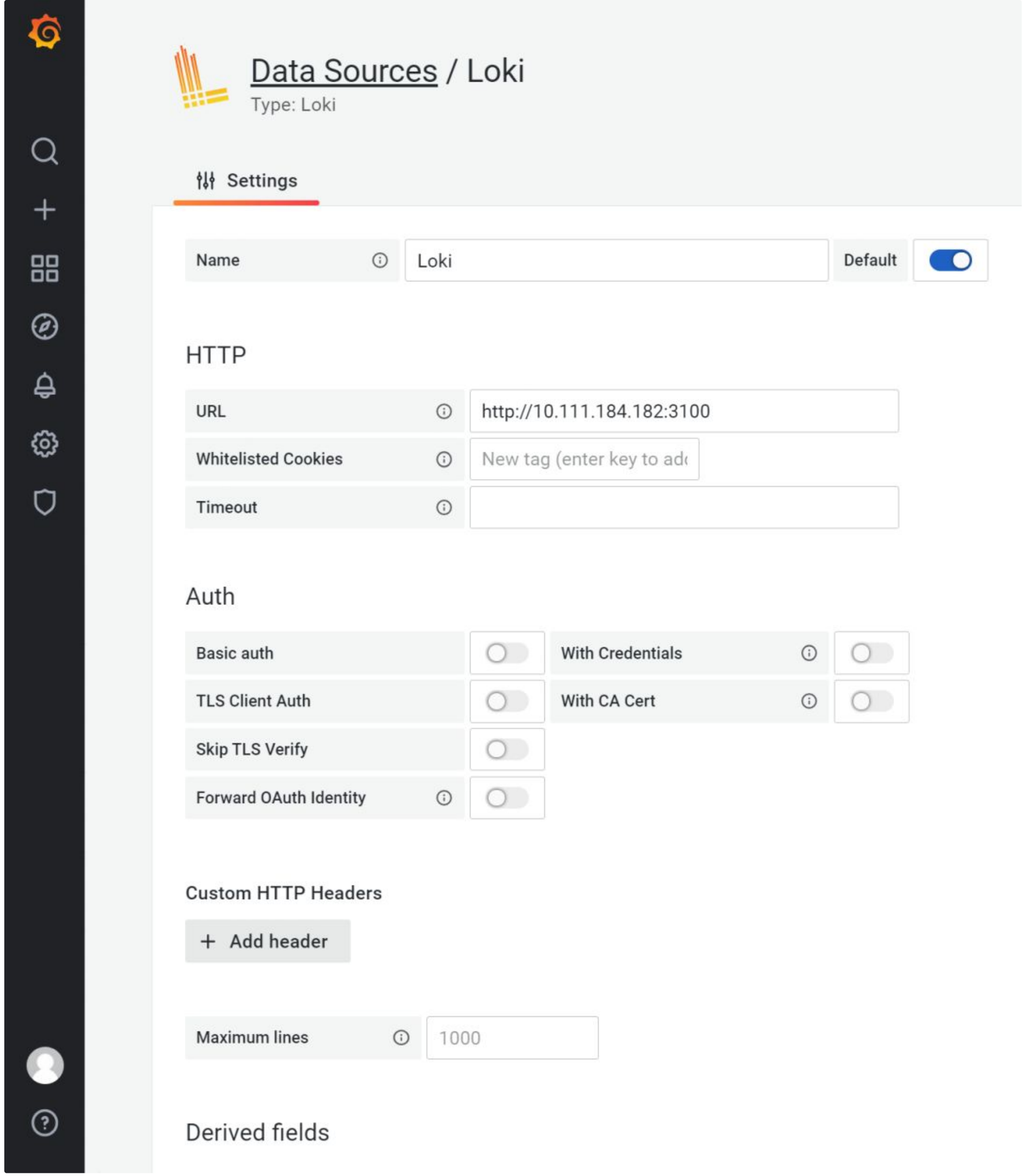
`loki-config.yaml` 配置：

```yaml
# loki 配置
config:
  limits_config:
   # Distributor 分发器的日志接收速率限制
    ingestion_rate_mb: 8
# 实例数
replicas: 1
# 限制使用资源
resources:
  limits:
    cpu: 2000m
    memory: 2048Mi
# 挂载宿主机时间
extraVolumeMounts:
  - name: host-time
    mountPath: /etc/localtime
extraVolumes:
  - name: host-time
    hostPath:
      path: /etc/localtime
```

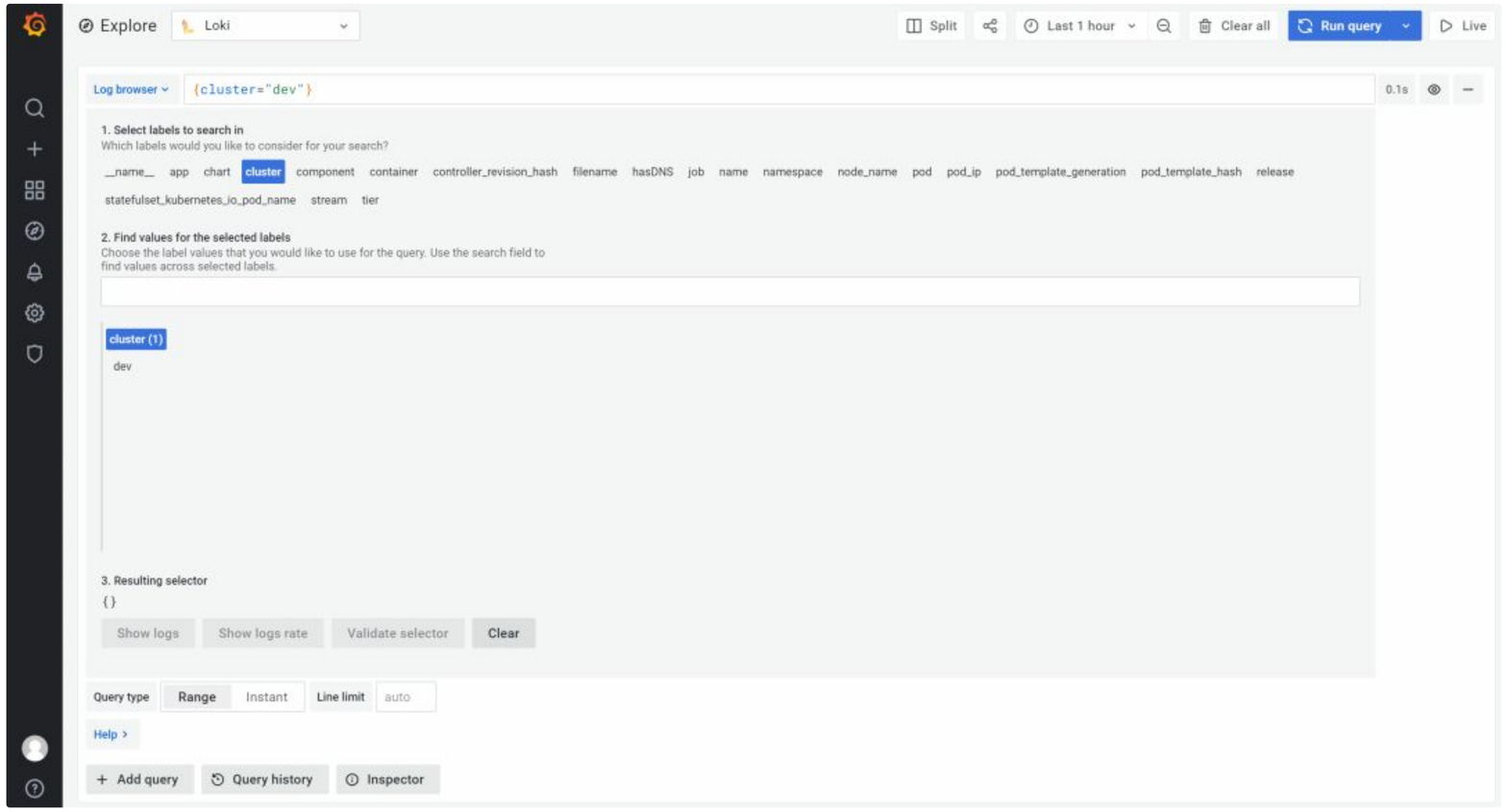
`promtail-config.yaml` 配置：

```yaml
extraArgs:
  # 添加全局静态标签 cluster:dev
  - -client.external-labels=cluster=dev
# 限制使用资源
resources:
  limits:
    cpu: 512m
    memory: 512Mi
# 挂载宿主机时间
extraVolumeMounts:
  - name: host-time
    mountPath: /etc/localtime
extraVolumes:
  - name: host-time
    hostPath:
      path: /etc/localtime
# promtail 配置
config:
  lokiAddress: http://loki-ip:3100/loki/api/v1/push
  snippets:
    # 清除默认配置
    scrapeConfigs: ""
    # 自定义配置
    extraScrapeConfigs: |
      # 通过 kubernetes_sd_configs:pod 配置 pod 日志，参考 https://grafana.com/docs/loki/latest/clients/promtail/configuration/#kubernetes_sd_config
      - job_name: kubernetes-pods-app
        # 流水线
        pipeline_stages:
          {{- toYaml .Values.config.snippets.podPipelineStages | nindent 4 }}
        kubernetes_sd_configs:
          - role: pod
        relabel_configs:
          # 把 pod 所有的标签暴露出来
          - action: labelmap
            regex: __meta_kubernetes_pod_label_(.+)
            replacement: $1
            target_label: $1
          - action: drop
            regex: .+
            source_labels:
              - __meta_kubernetes_pod_label_app_kubernetes_io_name
          - action: replace
            source_labels:
              - __meta_kubernetes_pod_ip
            target_label: pod_ip
          - action: replace
            source_labels:
              - __meta_kubernetes_pod_label_app
            target_label: app
          - action: drop
            regex: ''
            source_labels:
              - app
          - action: replace
            source_labels:
              - __meta_kubernetes_pod_label_component
            target_label: component
          {{- if .Values.config.snippets.addScrapeJobLabel }}
          - action: replace
            replacement: kubernetes-pods-app
            target_label: scrape_job
          {{- end }}
          {{- toYaml .Values.config.snippets.common | nindent 4 }}
    podPipelineStages:
      - docker: {}
```

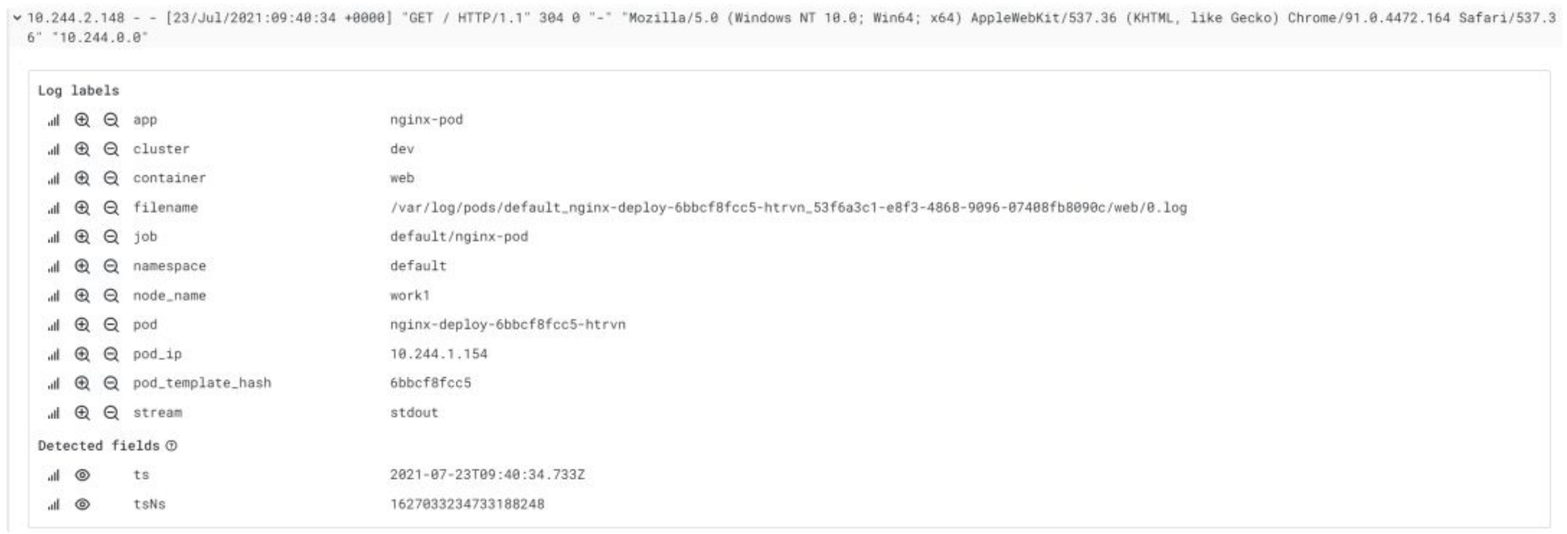
Grafana 添加 Loki 数据源：


Data Sources / Loki
Type: Loki
Settings
Name
Loki
Default
HTTP
URL
http://10.111.184.182:3100
Whitelisted Cookies
New tag (enter key to ad
Timeout
Auth
Basic auth
With Credentials
TLS Client Auth
With CA Cert
Skip TLS Verify
Forward OAuth Identity
Custom HTTP Headers
+ Add header
Maximum lines
1000
Derived fields

Grafana 中按照标签查询日志：



nginx 日志示例：



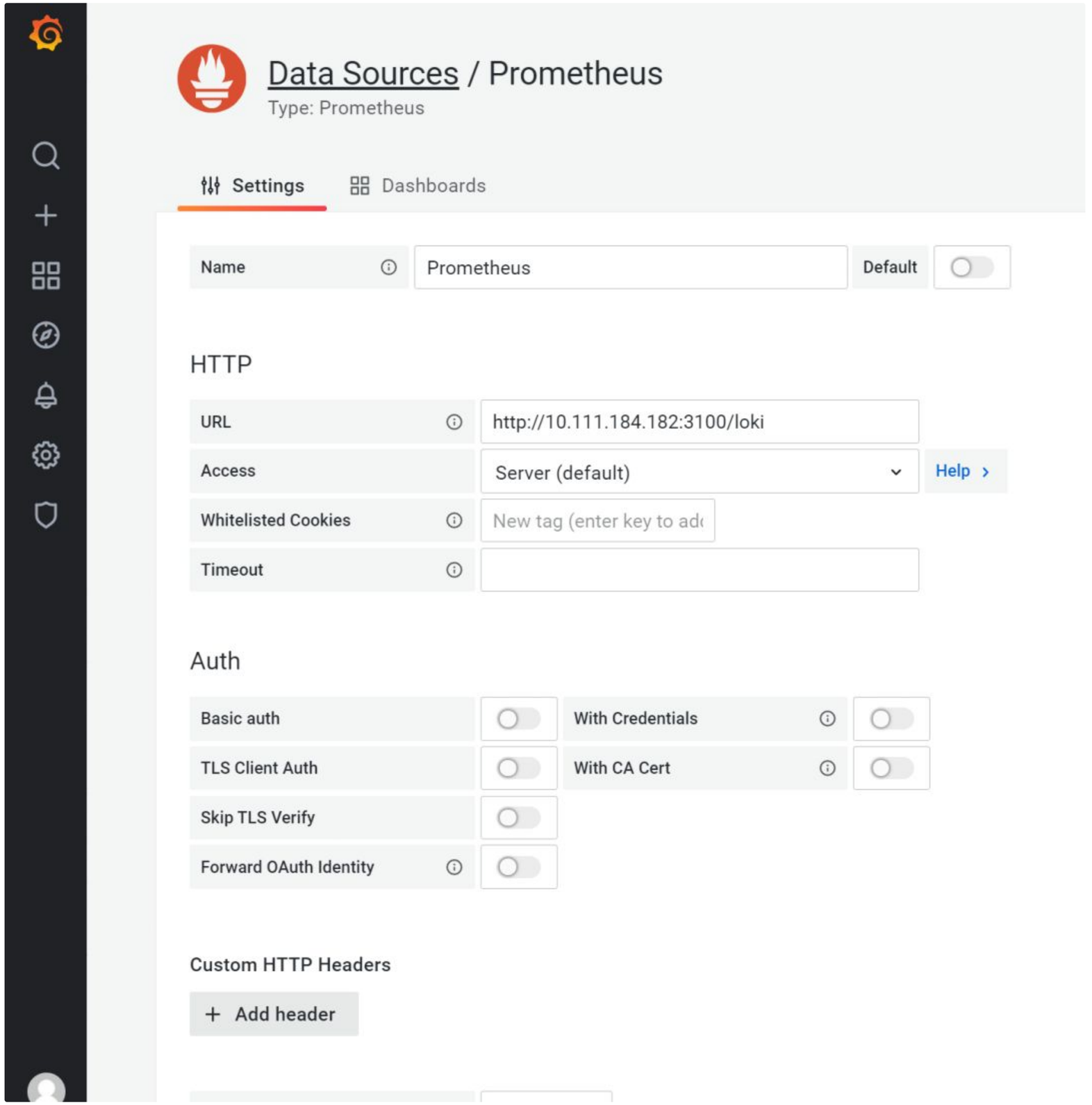
日志告警

Loki 支持三种模式创建日志告警：

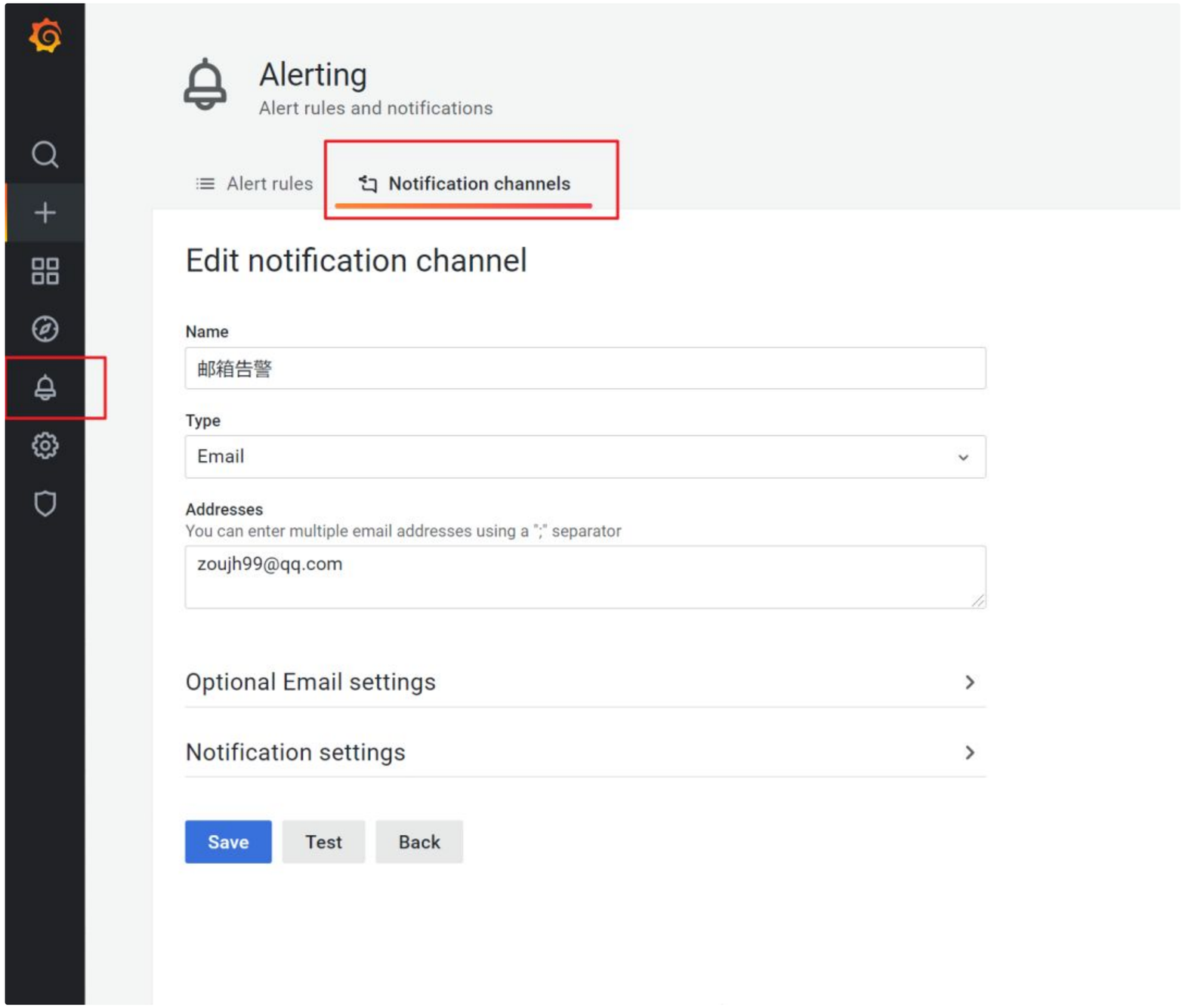
- 在 Promtail 中的 pipeline 管道的 metrics 的阶段，根据需求增加一个监控指标，然后使用 Prometheus 结合 Alertmanager 完成监控报警。
- 通过 Loki 自带的报警功能（ Ruler 组件）可以持续查询一个 rules 规则，并将超过阈值的事件推送给 AlertManager 或者其他 Webhook 服务。
- 将 LogQL 查询转换为 Prometheus 指标。可以通过 Grafana 自带的 Alert rules & notifications，定义有关 LogQL 指标的报警，推送到 Notification channels（ Prometheus Alertmanager， Webhook 等）。

以下主要介绍 LogQL 转化为 Prometheus 指标的方式实现告警。

首先，在 Grafana 添加 Prometheus 数据源，URL 只需要填入 http://loki-ip:3100/loki 即可将 LogQL 查询转换为 Prometheus 指标。



在告警规则配置中配置 Notification channels

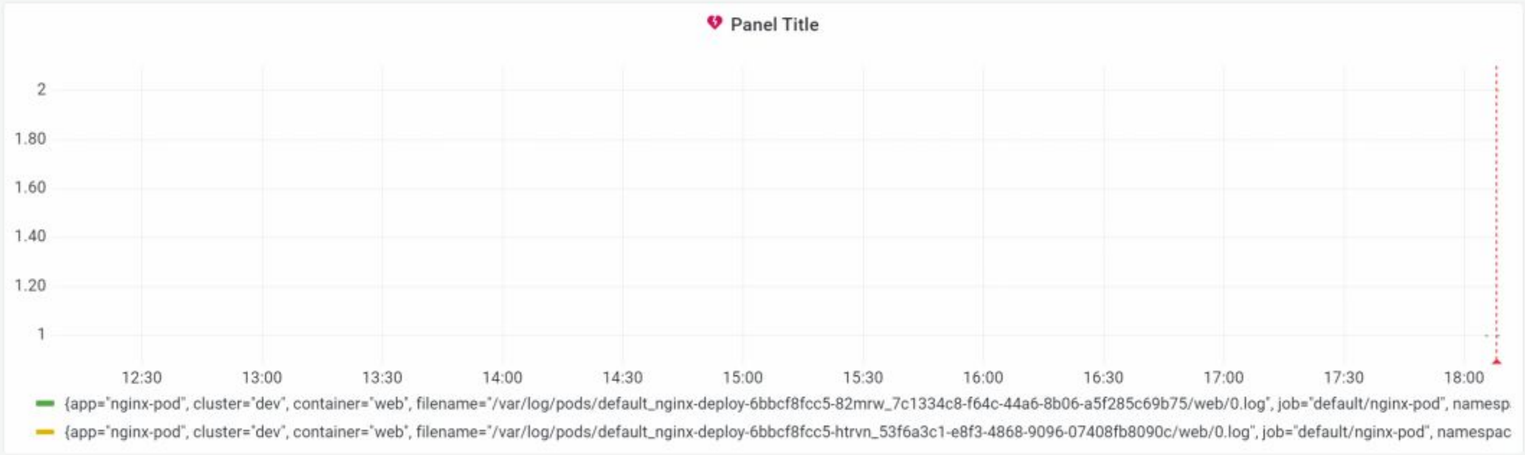


新建一个 Dashboard ，配置一个面板，例如：当 nginx 出现 404 状态码，触发告警：

```
count_over_time({app="nginx-pod"} |= "404" [1m])
```

告警 / Edit Panel

Table view | Fill | Actual | Last 6 hours



Query 1 | Transform 0 | Alert 1

Data source: Prometheus | Query options | MD = auto = 1166 | Interval = 20s | Query inspector

A (Prometheus)

(No metrics found) > `count_over_time({app="nginx-pod"} |= "404" [1m])`

Legend: legend format | Min step: | Resolution: 1/1

Format: Time series | Instant | Prometheus | Exemplars

Using Loki as a Prometheus data source is no longer supported. You must use the Loki data source for your Loki instance.

+ Query | + Expression

Query 1 | Transform 0 | Alert 1

Rule

Name: Nginx 出现 404 | Evaluate every: 1m | For: 0s

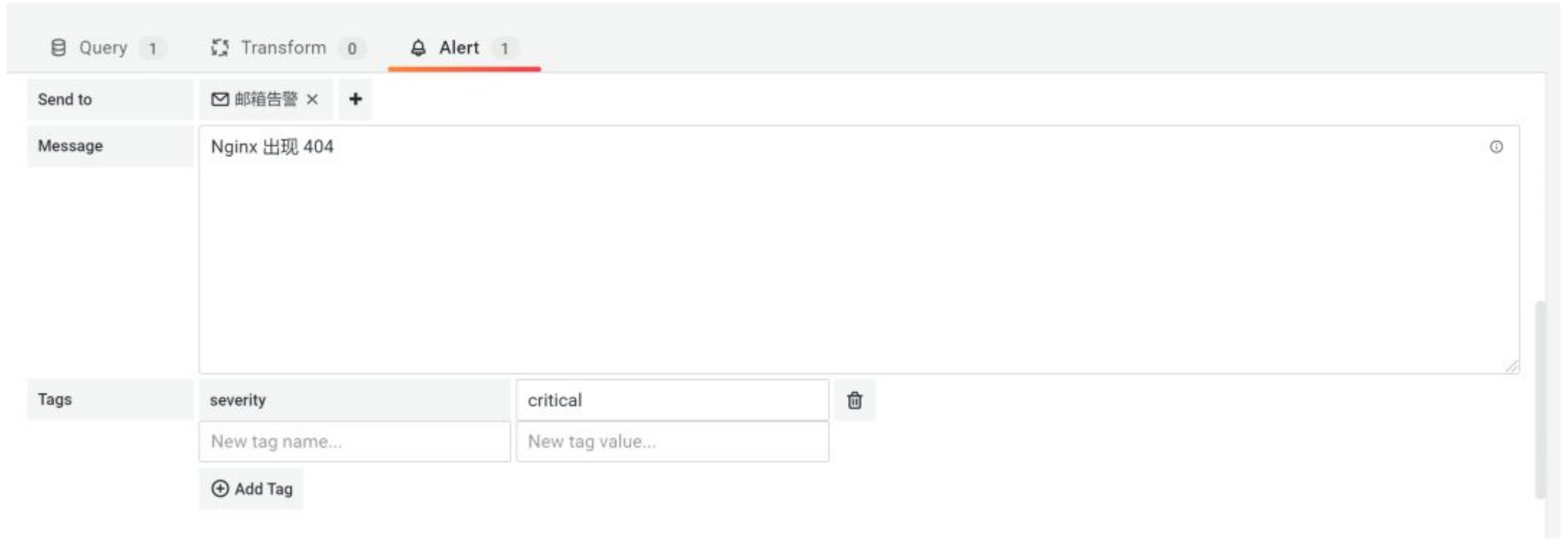
Conditions

WHEN sum () OF query (A, 5m, now) IS ABOVE 0

No data and error handling

If no data or all values are null — set state to: No Data

If execution error or timeout — set state to: Alerting



手动访问 nginx 404 页面，可以看到日志已经产生告警：

关于更多 Loki 和 Promtail 配置，以及日志的告警，推荐直接看官方文档，已经很详细了。这里不过多介绍。