



2025インターン

prometheusでメトリクス取と
ダッシュボード作成

せっかく高速化したのでどれくらいリクエストを短期間にどれくらい
捌いているかを可視化させてみよう
今回はPrometheus と grafanaを用いてダッシュボードを作ります

Prometheusとは?

Prometheus(プロメテウス)はイベント監視とアラート通知に利用される無料のアプリケーションソフトウェア。時系列データベースをHTTPプル方式でリアルタイムに構築して柔軟なクエリとアラート機能を提供する

[Wikipedia](#)より

Yahooでもかつて使われていました

[導入事例](#)



Prometheusでメトリクス取得

Prometheusでメトリクスを取るための実装を行う

/auctionで1秒間にどれだけリクエストを捌けているかのメトリクスを取得する

例えばラベル名を`auction_requests_total`にする

(ここのラベル名は各自好きに設定して構いませんが、後ほど使います)

ドキュメントを読んで実装しよう

- pythonの方は[こちら](#)
- goの方は[こちら](#)
- C++の方は[こちら](#)(C++だけ非公式のthird party製)

他のサイトで調べても構いません.

Prometheusでメトリクス取得

メトリクスを公開するポート番号は8888でお願いします

make checkをして動いている時に

うまく行けば次のcurlコマンドを投げると

curl -v <http://localhost:8888/metrics>

右のようになります

200 OKなら成功です

(必ずしも画像のように完全には一致しません)

```
naoto-fujiwara@J0124 ~ % curl -v http://localhost:8888/metrics
* Trying [::1]:8888...
* connect to ::1 port 8888 failed: Connection refused
* Trying 127.0.0.1:8888...
* Connected to localhost (127.0.0.1) port 8888
> GET /metrics HTTP/1.1
> Host: localhost:8888
> User-Agent: curl/8.4.0
> Accept: */*
>
* HTTP 1.0, assume close after body
< HTTP/1.0 200 OK
< Date: Sun, 25 May 2025 09:00:45 GMT
< Server: WSGIServer/0.2 CPython/3.12.8
< Content-Type: text/plain; version=0.0.4; charset=utf-8
< Content-Length: 3783
<
# HELP python_gc_objects_collected_total Objects collected during gc
# TYPE python_gc_objects_collected_total counter
python_gc_objects_collected_total{generation="0"} 274.0
python_gc_objects_collected_total{generation="1"} 6.0
python_gc_objects_collected_total{generation="2"} 0.0
# HELP python_gc_objects_uncollectable_total Uncollectable objects found during
```

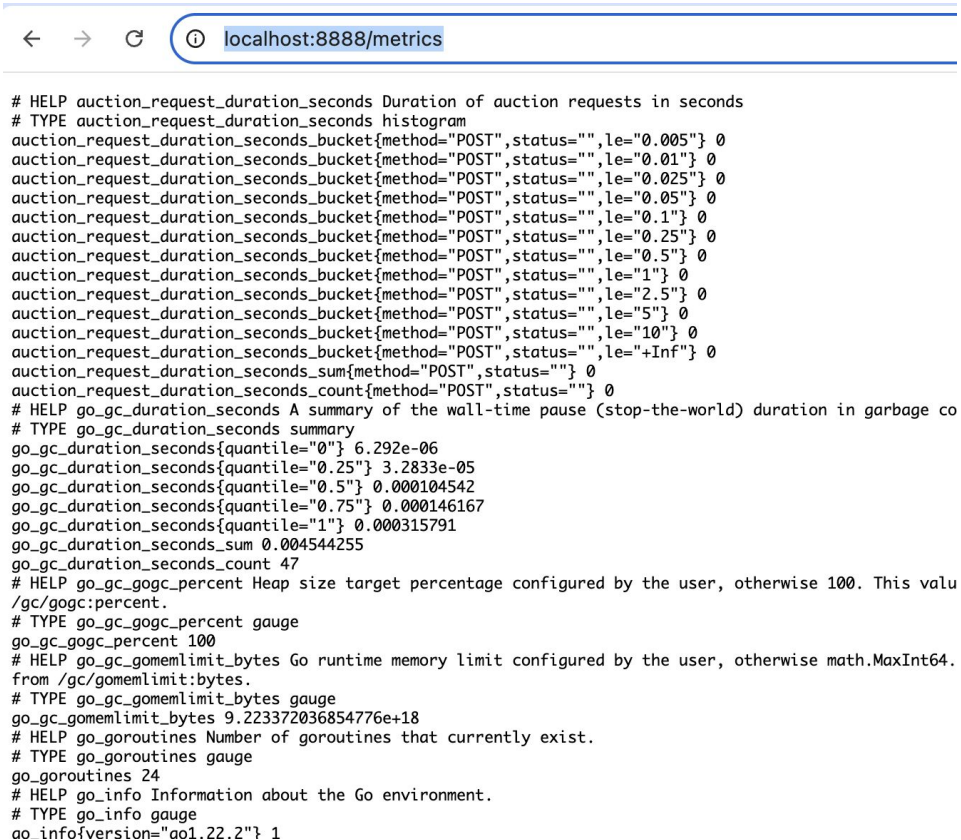
Prometheusでメトリクス取得

<http://localhost:8888/metrics>

へアクセスする

右のようになればOK

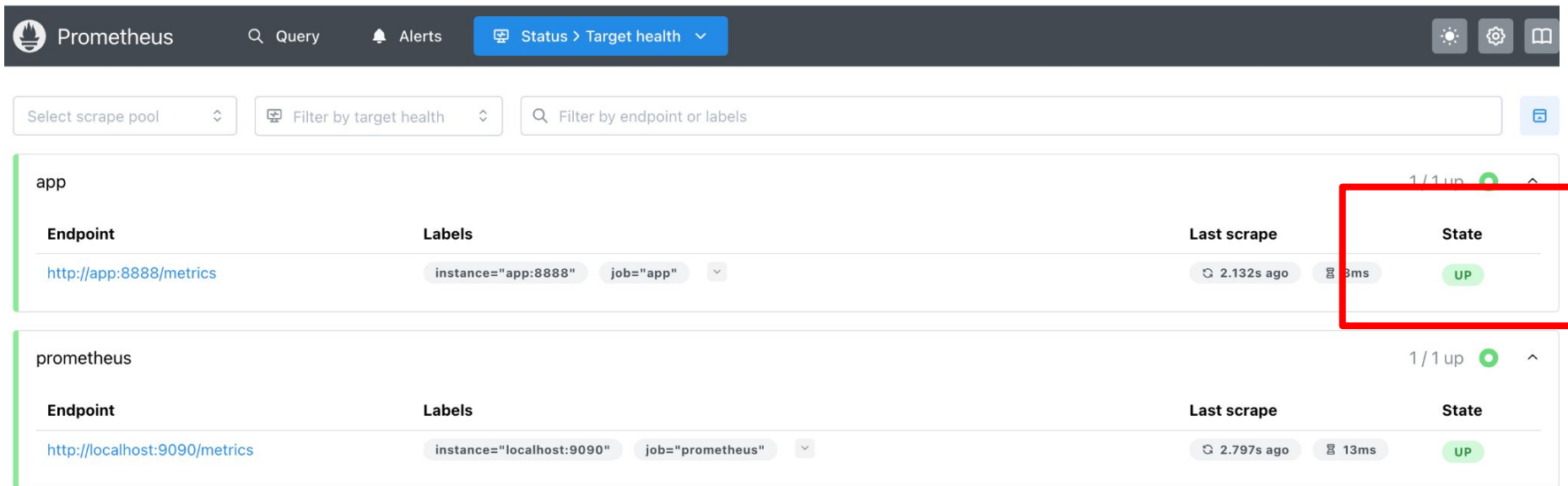
画像とは必ずしも一致しません



```
# HELP auction_request_duration_seconds Duration of auction requests in seconds
# TYPE auction_request_duration_seconds histogram
auction_request_duration_seconds_bucket{method="POST",status="",le="0.005"} 0
auction_request_duration_seconds_bucket{method="POST",status="",le="0.01"} 0
auction_request_duration_seconds_bucket{method="POST",status="",le="0.025"} 0
auction_request_duration_seconds_bucket{method="POST",status="",le="0.05"} 0
auction_request_duration_seconds_bucket{method="POST",status="",le="0.1"} 0
auction_request_duration_seconds_bucket{method="POST",status="",le="0.25"} 0
auction_request_duration_seconds_bucket{method="POST",status="",le="0.5"} 0
auction_request_duration_seconds_bucket{method="POST",status="",le="1"} 0
auction_request_duration_seconds_bucket{method="POST",status="",le="2.5"} 0
auction_request_duration_seconds_bucket{method="POST",status="",le="5"} 0
auction_request_duration_seconds_bucket{method="POST",status="",le="10"} 0
auction_request_duration_seconds_bucket{method="POST",status="",le="+Inf"} 0
auction_request_duration_seconds_sum{method="POST",status=""} 0
auction_request_duration_seconds_count{method="POST",status=""} 0
# HELP go_gc_duration_seconds A summary of the wall-time pause (stop-the-world) duration in garbage co
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 6.292e-06
go_gc_duration_seconds{quantile="0.25"} 3.2833e-05
go_gc_duration_seconds{quantile="0.5"} 0.000104542
go_gc_duration_seconds{quantile="0.75"} 0.000146167
go_gc_duration_seconds{quantile="1"} 0.000315791
go_gc_duration_seconds_sum 0.004544255
go_gc_duration_seconds_count 47
# HELP go_gc_gogc_percent Heap size target percentage configured by the user, otherwise 100. This valu
/gc/gogc:percent.
# TYPE go_gc_gogc_percent gauge
go_gc_gogc_percent 100
# HELP go_gc_gomemlimit_bytes Go runtime memory limit configured by the user, otherwise math.MaxInt64.
from /gc/gomemlimit:bytes.
# TYPE go_gc_gomemlimit_bytes gauge
go_gc_gomemlimit_bytes 9.223372036854776e+18
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
go_goroutines 24
# HELP go_info Information about the Go environment.
# TYPE go_info gauge
go_info{version="go1.22.2"} 1
```

Prometheusでメトリクス取得

make check で動いている時に <http://localhost:9090/targets> にアクセス
appがup状態かを確認



The screenshot shows the Prometheus web interface at the 'Status > Target health' page. It displays two targets:

Target	Endpoint	Labels	Last scrape	State
app	http://app:8888/metrics	instance="app:8888" job="app"	2.132s ago 3ms	UP
prometheus	http://localhost:9090/metrics	instance="localhost:9090" job="prometheus"	2.797s ago 13ms	UP

The 'app' target's state is highlighted with a red box, indicating it is 'UP'.

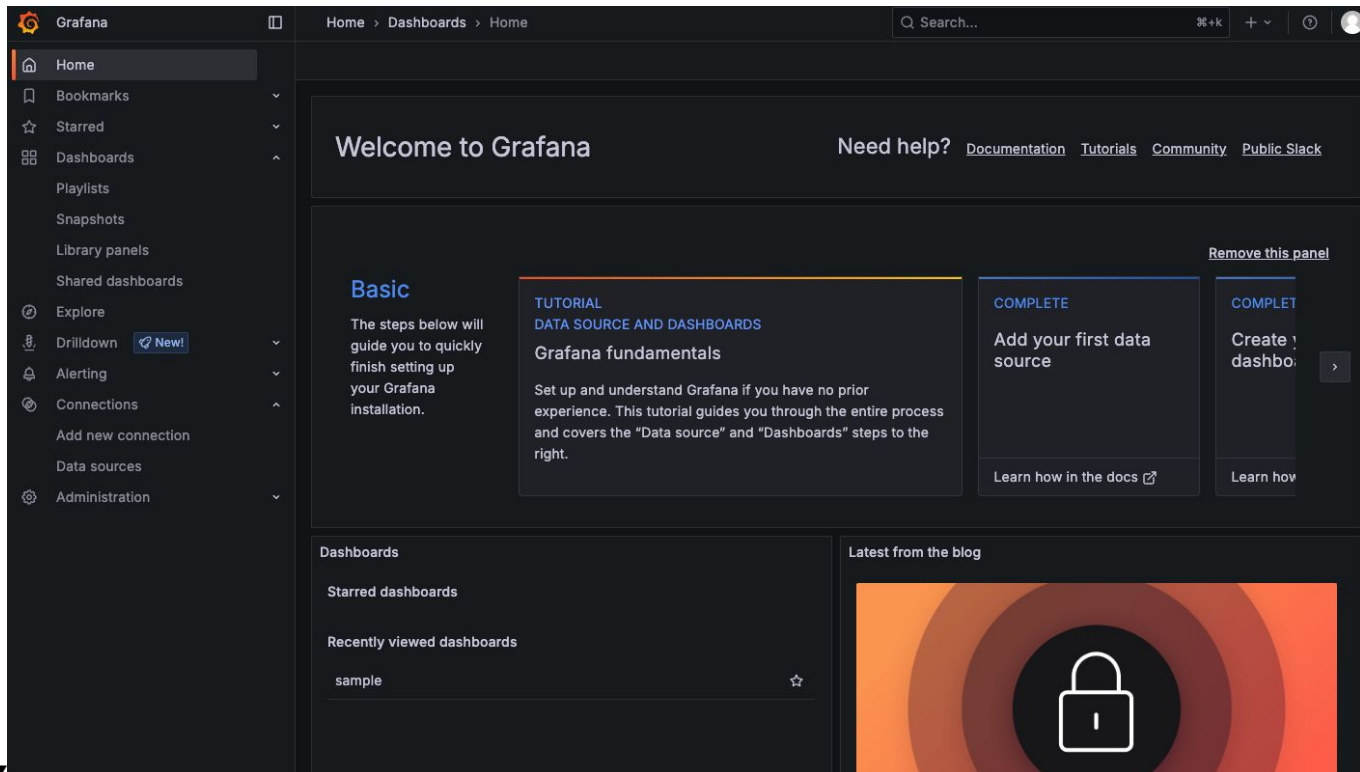
Grafanaとは?

Grafana(グラフィナ)は分析およびインタラクティブな視覚化を可能にする、マルチプラットフォームで動作するオープンソースのWebアプリケーションである

[Wikipedia](#)より

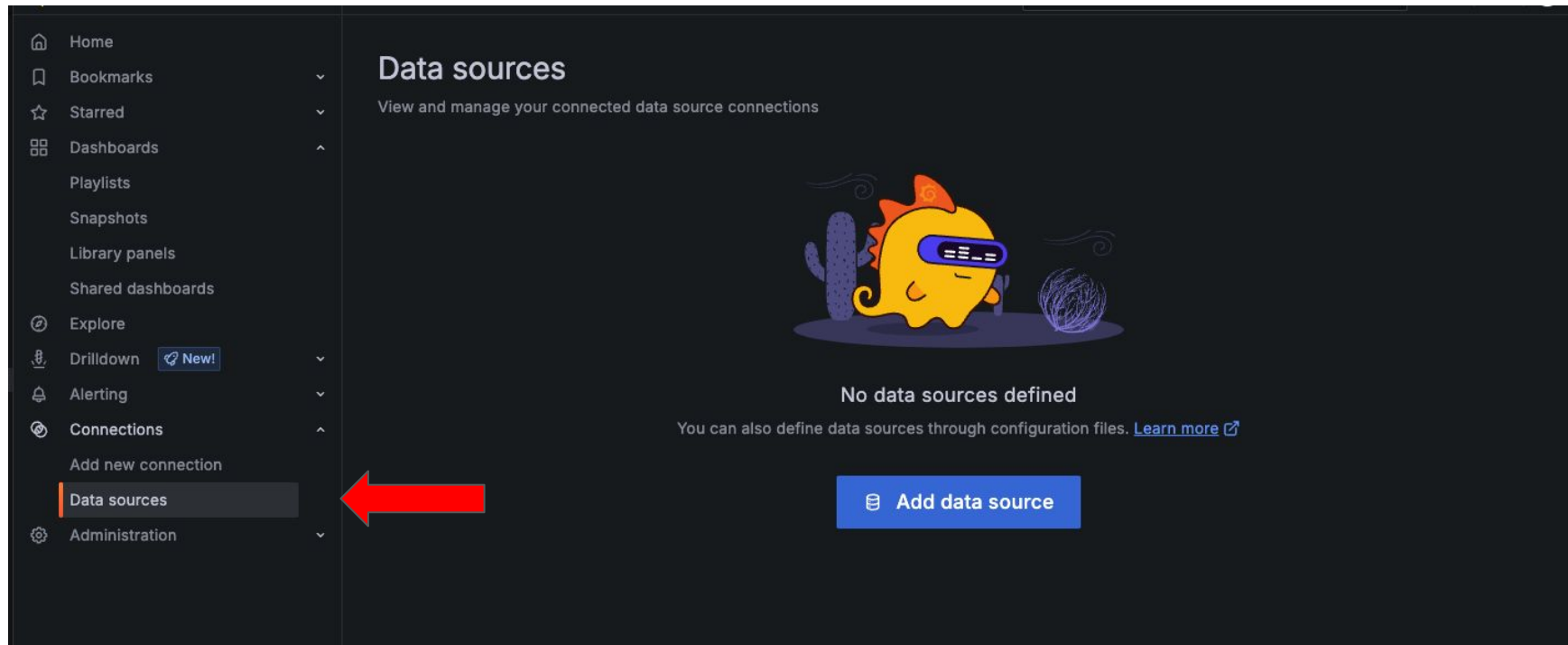
grafanaでダッシュボードを作成

続いて <http://localhost:3000> にアクセス。ユーザ名とパスワードはadmin



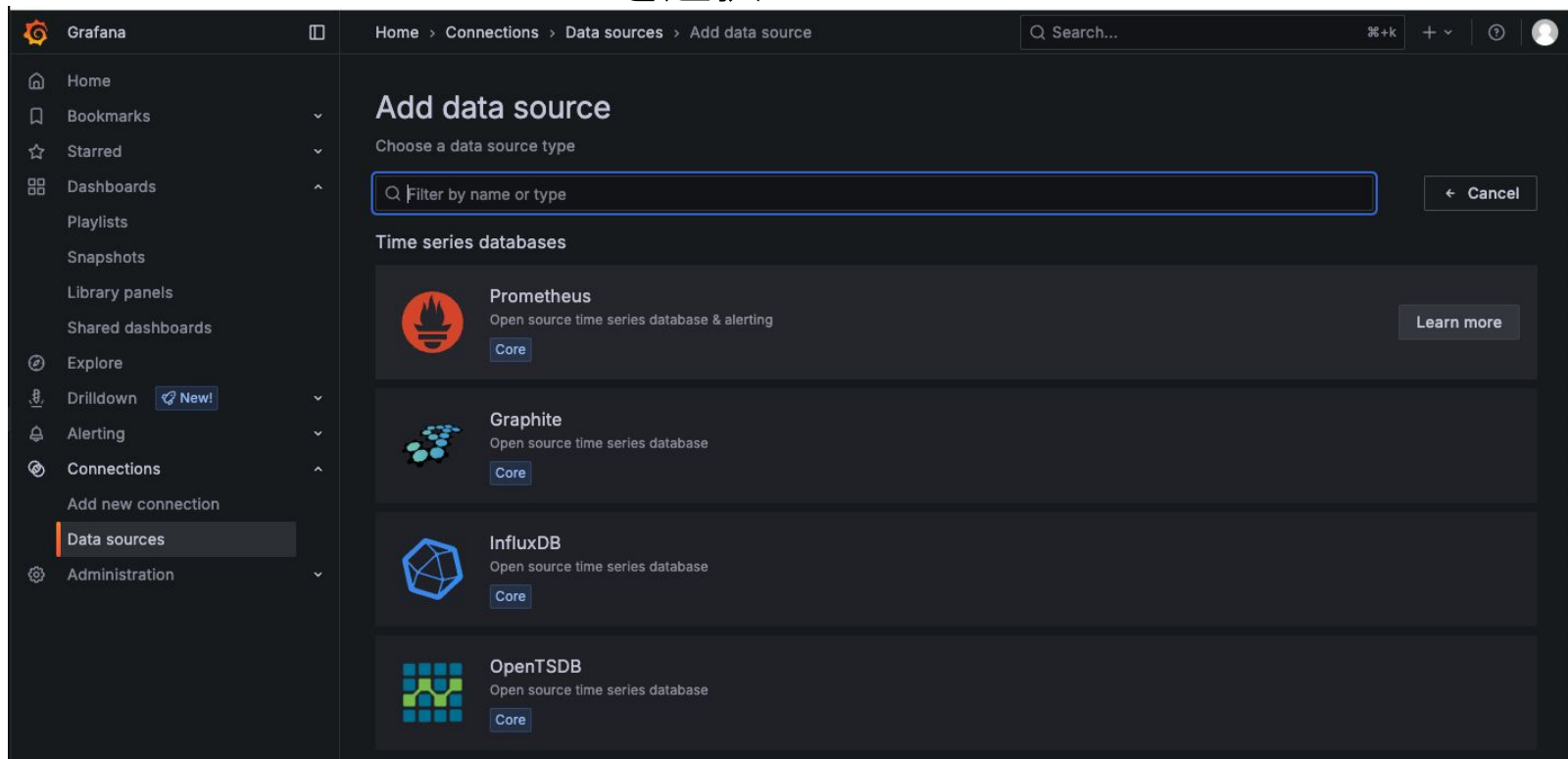
grafanaでダッシュボードを作成

Data sources に行くとこんな感じ Add Data sourceをクリック

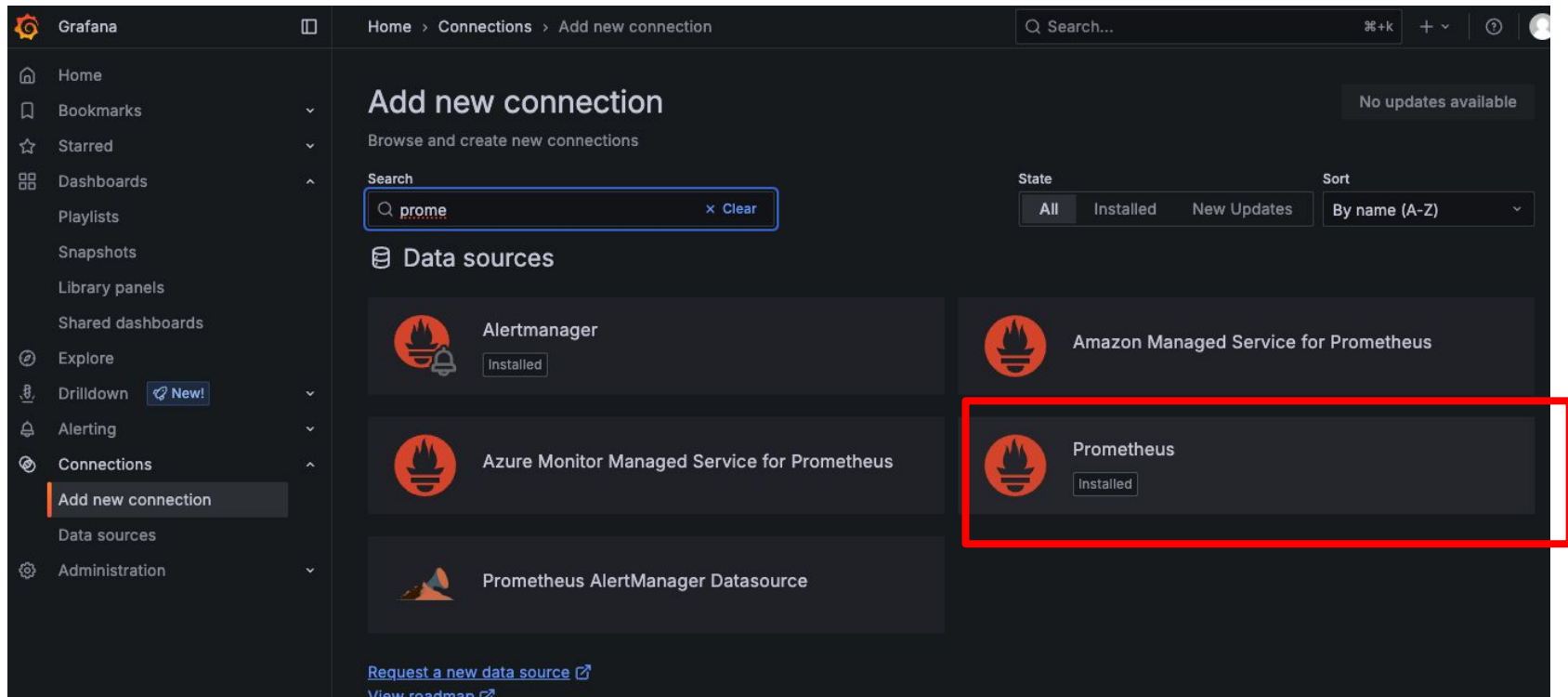


grafanaでダッシュボードを作成

Data sources からPrometheusを選択

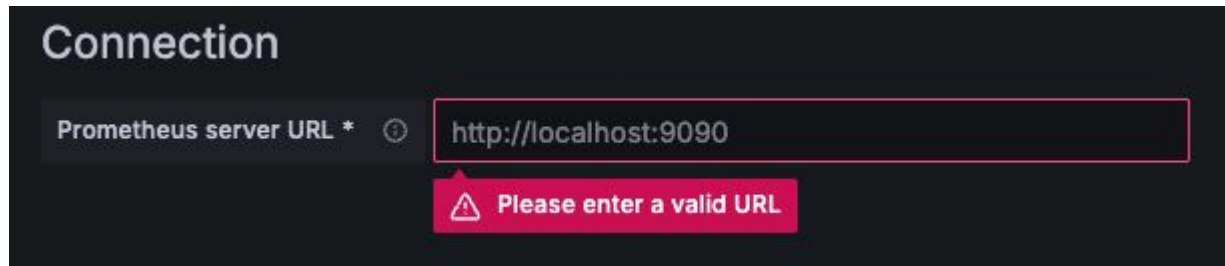


Add new connection からPrometheusを選択



grafanaでダッシュボードを作成

Connection に <http://prometheus:9090> と打ち込む

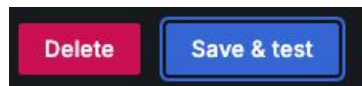


Connection

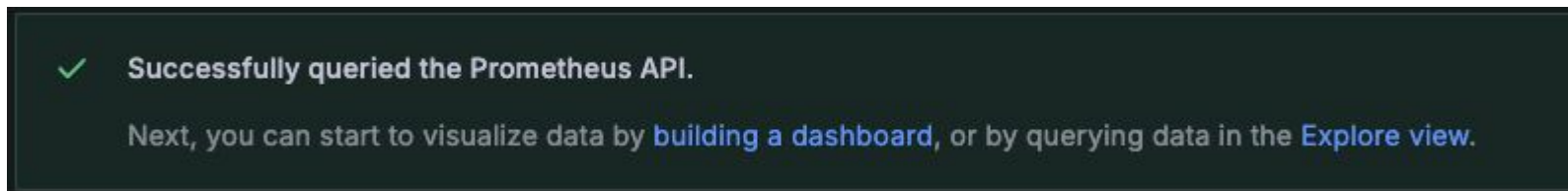
Prometheus server URL *

Please enter a valid URL

下にスクロールしてsave & test

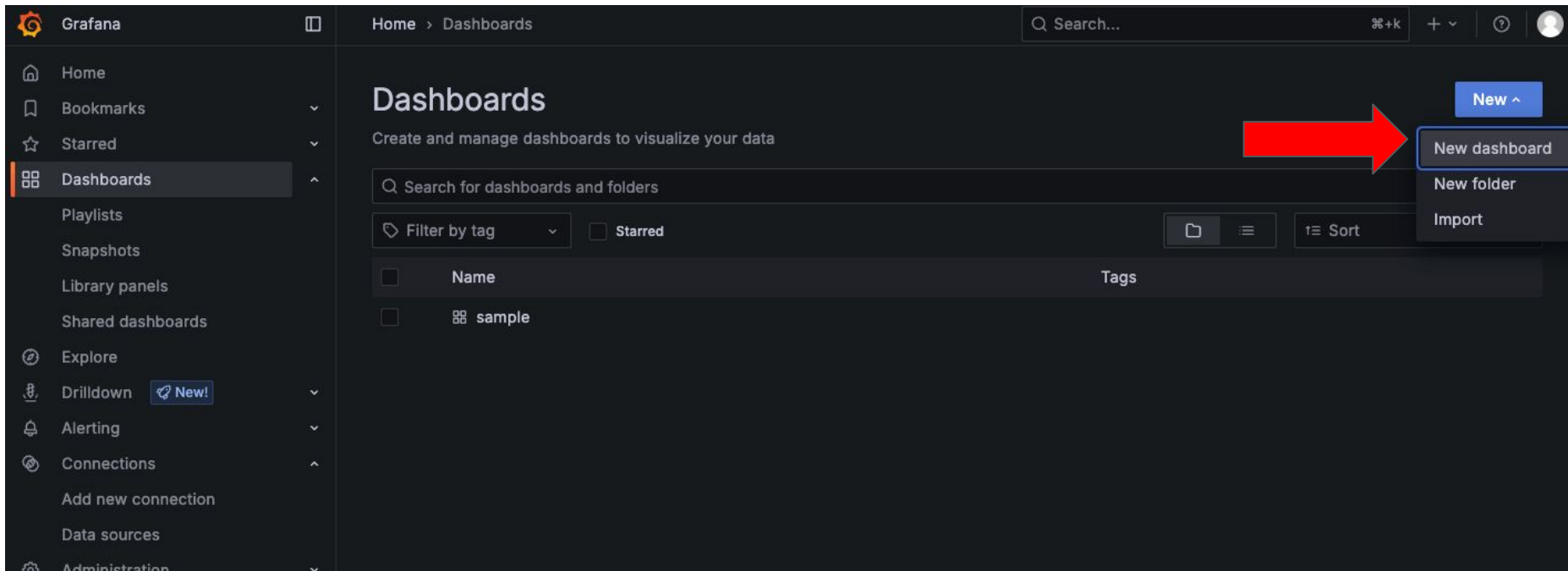


以下のようなものが出ればOK



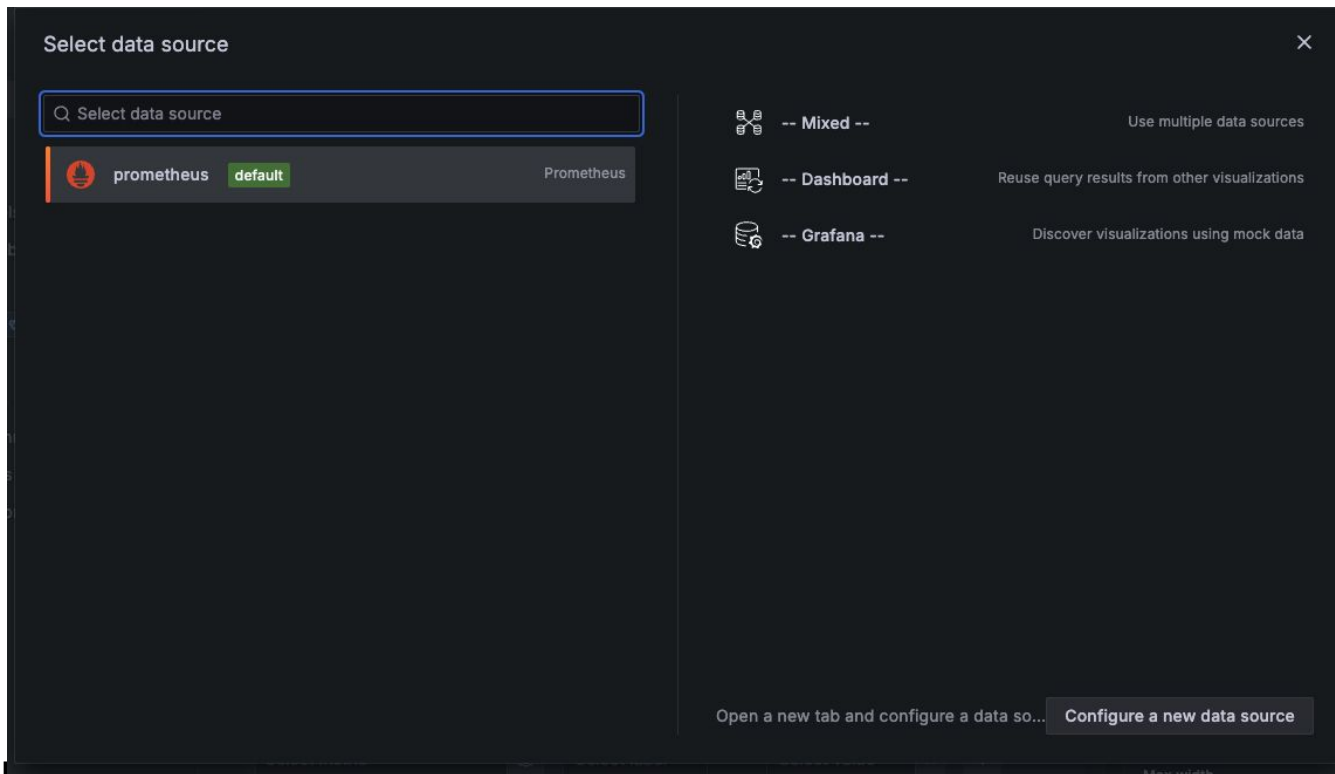
grafanaでダッシュボードを作成

DashboardsからNew dashboardを選択 (sampleは皆さんの中には存在しない)



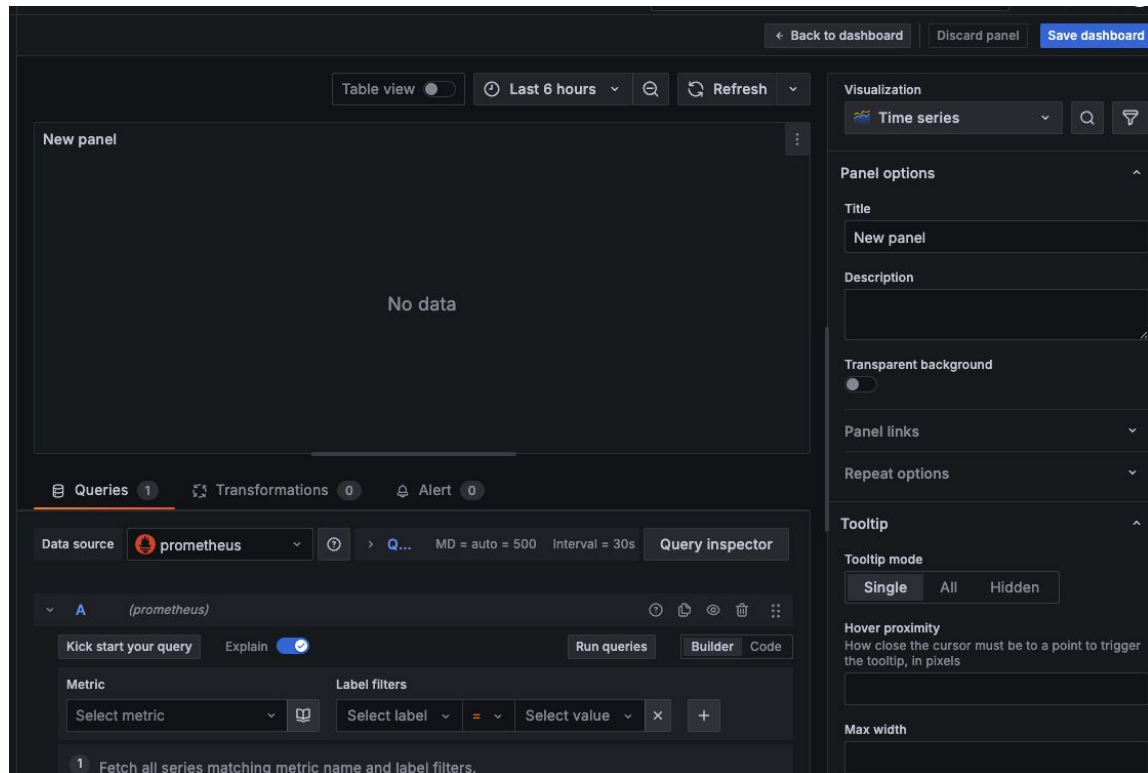
grafanaでダッシュボードを作成

先ほど選択したprometheusを選択



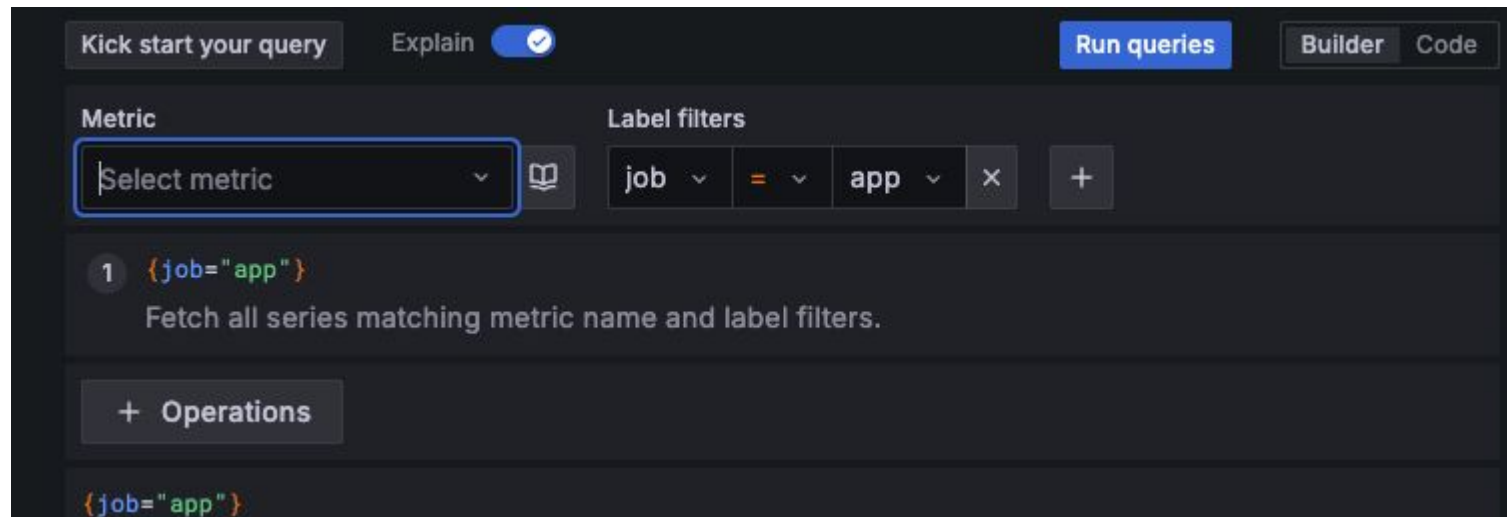
grafanaでダッシュボードを作成

以下のようなダッシュボード編集画面になる



grafanaでダッシュボードを作成

Label filtersでjob, appをそれぞれ選ぶ。下のようになればOK

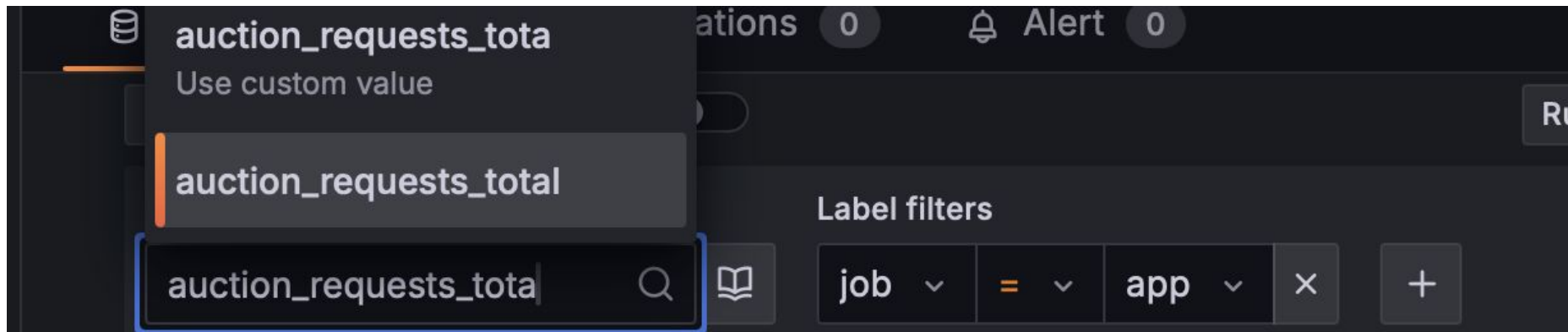


grafanaでダッシュボードを作成

Metricを色々選んでみる

自分でつけたラベル名(例auction_requests_total)

とかで入力すると候補が出てくる

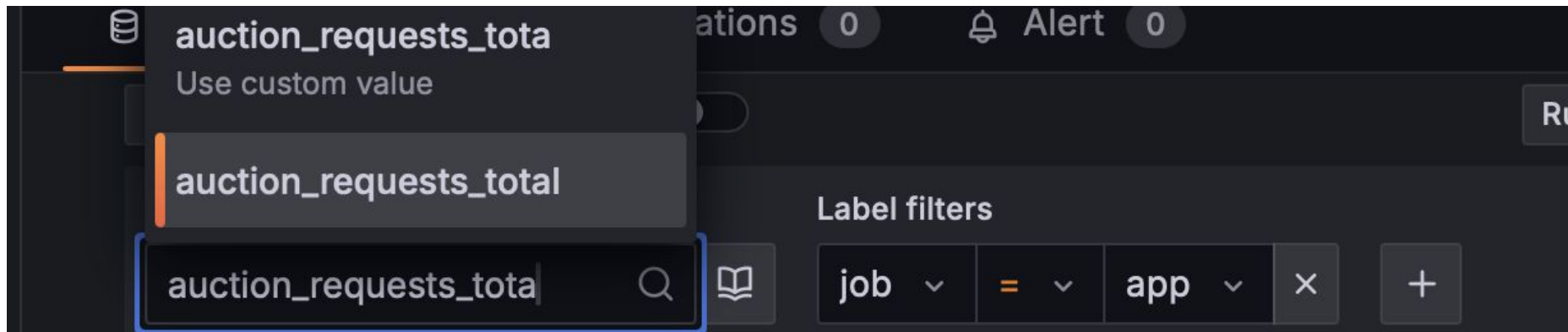


grafanaでダッシュボードを作成

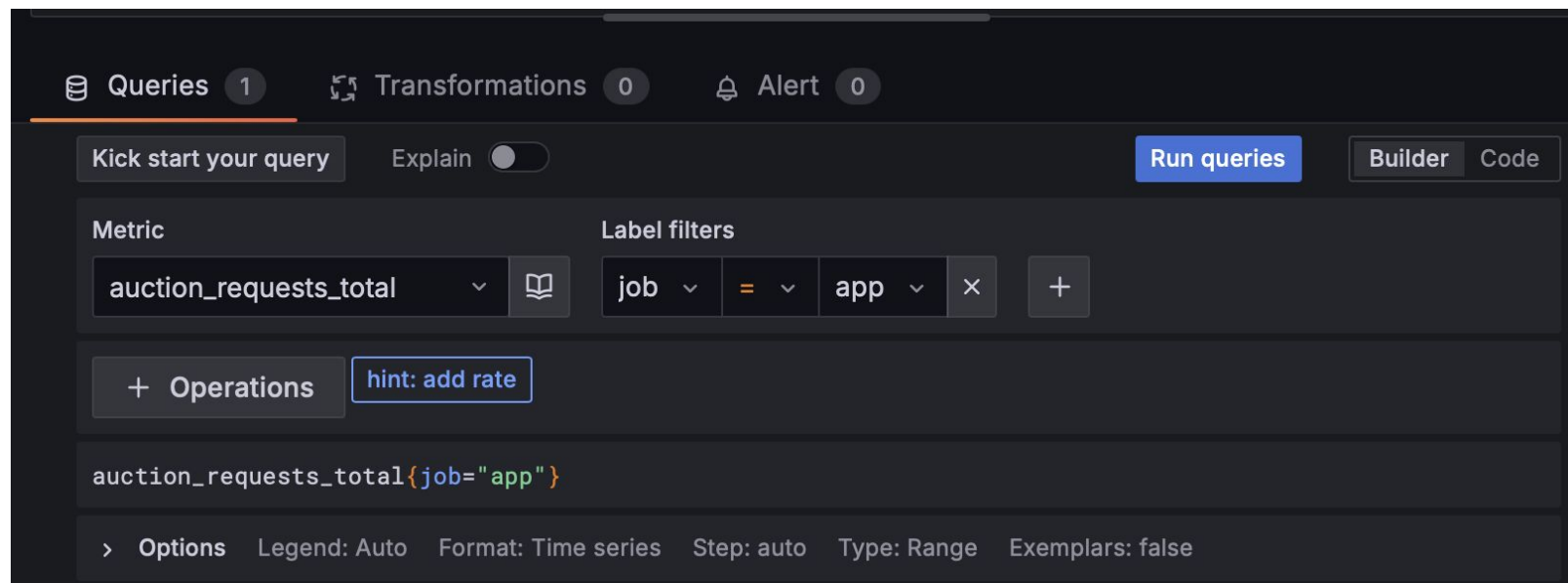
Metricを色々選んでみる

自分でつけたラベル名(例auction_requests_total)

とかで入力すると候補が出てくる



Operations → Range functions → Rate の順に選択



grafanaでダッシュボードを作成

Range 部分を例えば1mにとかにすると1分間当たりの指標が取れる

The screenshot shows the Grafana query editor interface. At the top, the 'Metric' dropdown is set to 'auction_requests_total'. To its right, the 'Label filters' section shows 'job' set to 'app'. Below the metric, the 'Rate' dropdown is selected, and the 'Range' dropdown is set to '\$__rate_interval'. A '+ Operations' button is visible to the right of the 'Rate' dropdown. The query text area displays the resulting query: `rate(auction_requests_total{job="app"}[$__rate_interval])`. At the bottom, the 'Options' tab is active, showing settings: 'Legend: Auto', 'Format: Time series', 'Step: auto', 'Type: Range', and 'Exemplars: false'.

grafanaでダッシュボードを作成

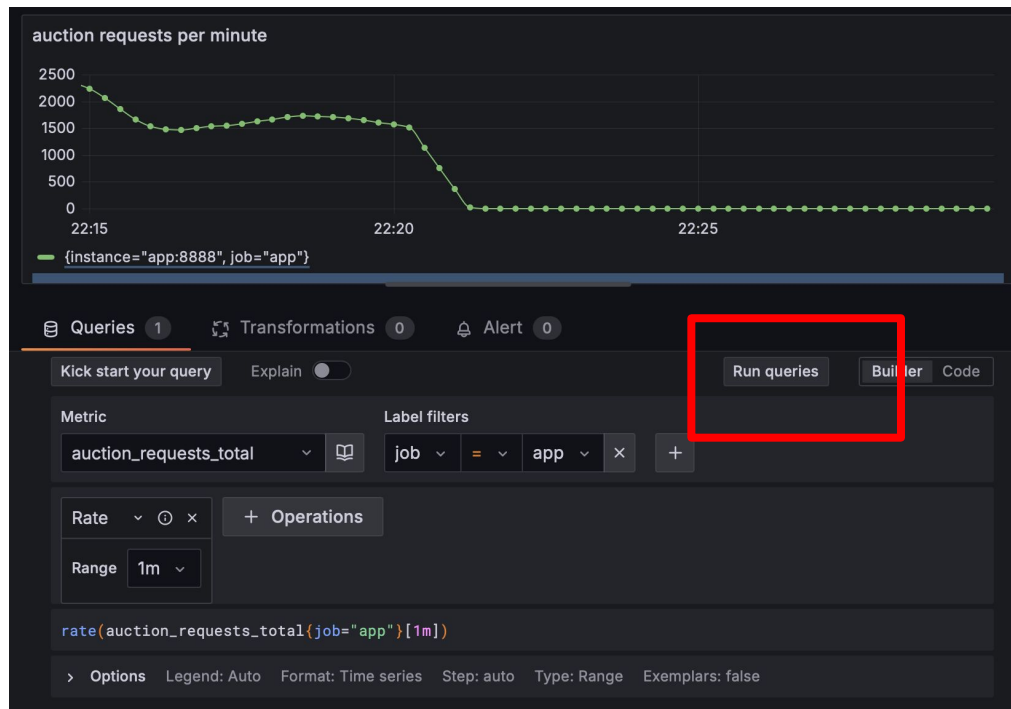
決まったら

Run queries

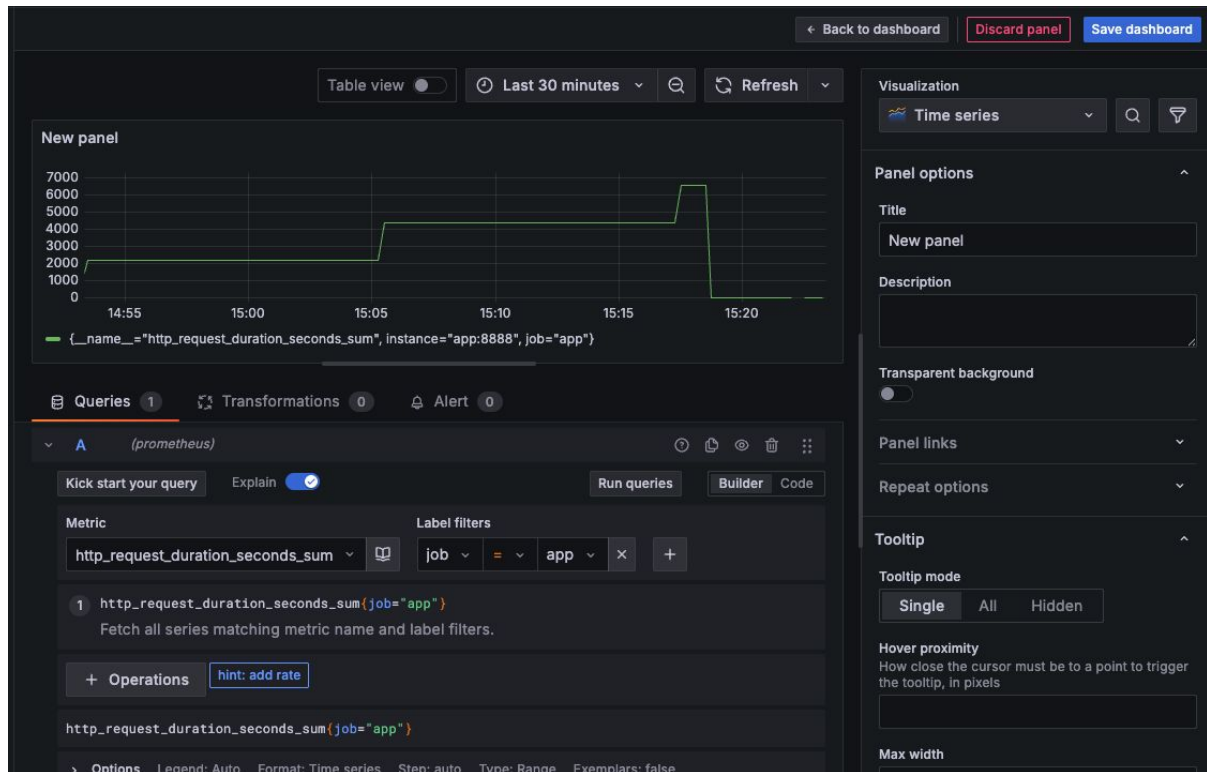
ボタンを推すと

グラフが出ます

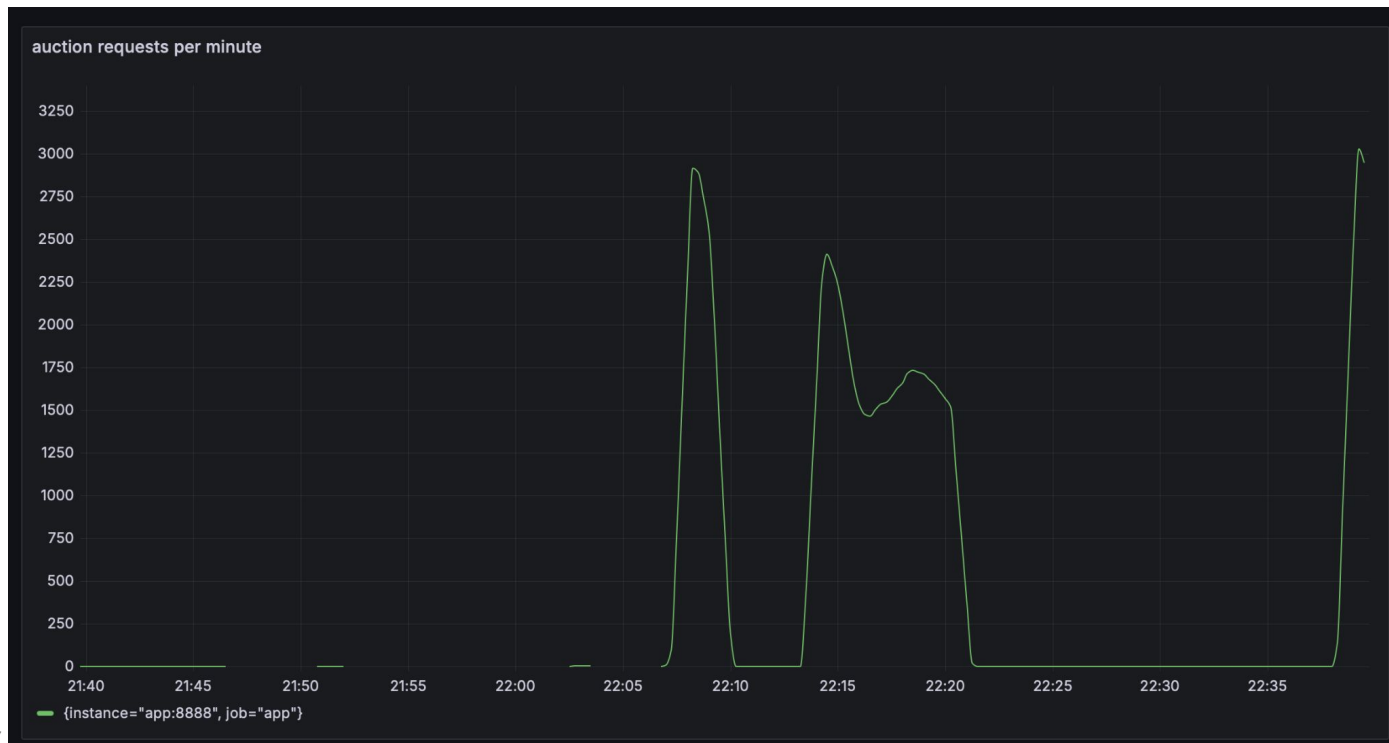
(画像ではすでに出ています)



オプションを通して適当にタイトルなどを変更してsave dashboard



完成例 (波打っているのはk6で負荷テストをしている最中の時間)



終わり