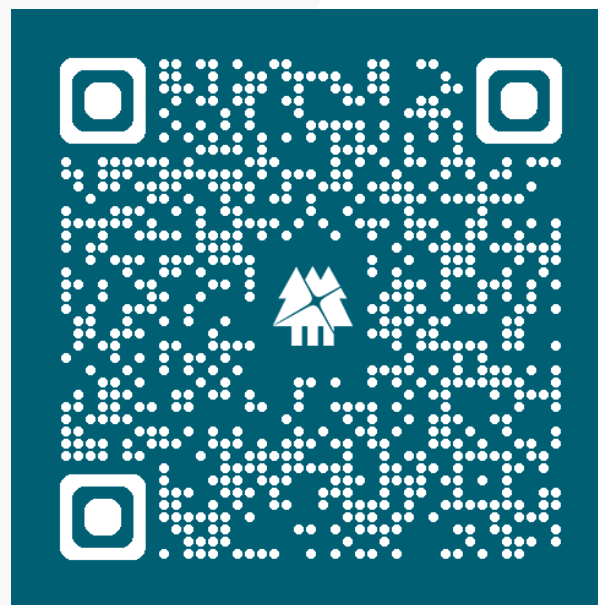


OSS分散ベクトル検索エンジンValdと最新の取り組み

Vald: Cloud Native Distributed Vector Database

LINEヤフー株式会社 Matts966



資料URL

自己紹介

松井誠泰 (GitHub: [Matts966](#))

- LINEヤフー株式会社
 - OSSのベクトルデータベースValdチームに転職して半年
- 趣味
 - 🍺 🍷 🍹 🍲 💻 📖 🚲



お品書き

- Valdのご紹介
- 最新の取り組み
 - Grafana Foundation SDKを用いたGrafana Dashboard as Code
 - E2E V2: YAMLで宣言的にテストシナリオを記述
- ベクトルDB選定のすすめ

Valdのご紹介

- クラウドネイティブな分散近似近傍ベクトルDB
- オープンソース
- CNCF Landscape



vald.vdaas.org

A screenshot of the CNCF Landscape website. The Vald project page is highlighted. The page shows the Vald logo, the name 'Vald' by LY Corporation, and categories 'App Definition and Development' and 'Database'. A description states: 'Vald is a highly scalable, cloud-native distributed vector search engine optimized for machine learning and AI applications. It offers efficient, near real-time search for high-dimensional vector data, ensuring robust performance and flexibility in handling large datasets. Engineered for ease of use and integration, Vald empowers developers with cutting-edge search capabilities.' Below this, the 'Repositories' section shows 'vdaas/vald (primary)' with a link to 'https://github.com/vdaas/vald'. Statistics are displayed: 1.6k Stars, 33 Contributors, Aug '19 First commit, Aug '25 Latest commit, and Jul '25 Latest release. A 'Participation stats' bar chart is at the bottom. The background shows the CNCF Landscape sidebar with various other projects like dapr, operator framework, kubernetes, etc.



最新の取り組み

1. Grafana Foundation SDKを用いたGrafana Dashboard as Code
2. E2E V2: YAMLで宣言的にテストシナリオを記述

Grafanaボード管理の課題

- コンポーネント毎にボード・パネルをたくさん管理
 - コンポーネントがかなり違うものの共通化の余地あり
 - 繰り返し、条件分岐したい
 - パネル毎にアップグレード作業
- JSONでバージョン管理はしていたものの
 - シンプルなパネルでもGUIからexportされたJSONは大きくなってしまい、直接読み書きするのが難しい

Agent
Memory

Cluster
Overview

Discoverer

LB
Gateway

Agent

Benchmark
Operator

Helm
Operator

Index
Correction

Index
Manager

grafana-foundation-sdk の概要

- Grafana公式が提供する言語ごとのSDK
- GrafanaのAPIスキーマをベースに自動生成されている
- Go, TypeScript, Python, Java に対応



選定理由・メリット

- 繰り返しを簡単に表現できる
 - 同じようなダッシュボードをコンポーネントごとにつくっている場合などに、関数等で整理しやすい
- メトリクスを管理しているコードと同じ言語で書くことで、メトリクス名を参照でき、二重管理を避けられる
 - メトリクスの宣言→ダッシュボード作成まで自動化可能

メリット

- メソッドチェーンで書けるので、補完に沿って書ける
- テキストなのでLLMの力を借りやすい
- GUIから出力できるJSONからGoへの自動変換が可能

```
builder.  
    WithPanel(  
        stat.NewPanelBuilder().  
            Title(title).  
            WithTarget(prometheusQuery(  
                addBasicLabel(promql.Vector(config.BenchmarkOperatorInfo)).String(),  
            ).Format("table")).  
            ReduceOptions(common.NewReduceDataOptionsBuilder().Calcs([]string{"lastNotNull"}).Fields  
                (field)).  
            Span(width).Height(heightShort),  
    )
```



メリット

- 簡単にバージョンアップグレード
 - 公式がAPIスキーマから自動生成しているので
 - `go get` でタグを切り替えるだけで簡単に最新に追従できる
 - 網羅性が高い

```
go get github.com/grafana/grafana-foundation-sdk/go@v11.6.x+cog-v0.0.x
```

メリット

- 公式から promql もビルダーが提供されていて、複雑な文字列、括弧の対応の管理を避けられる

```
promql.Sum(promql.Irate(  
    promql.Vector(cpuMetric).  
        Range(intervalVariable),  
)).By([]string{"pod"}).String()
```

注意点

- grafana/grafana-foundation-sdk#673
 - パネル配置にバグがあるため
 - 行や列の位置がズれるなど
 - 自分で整理するコードを書く必要あり
- 現状 [puzzle.go](#) としてValdレポジトリで公開

[Feature]: Support more granular panel ordering #673

Open Listed in #2937

Matts966 opened on Apr 11 · edited by Matts966

Why is this needed?

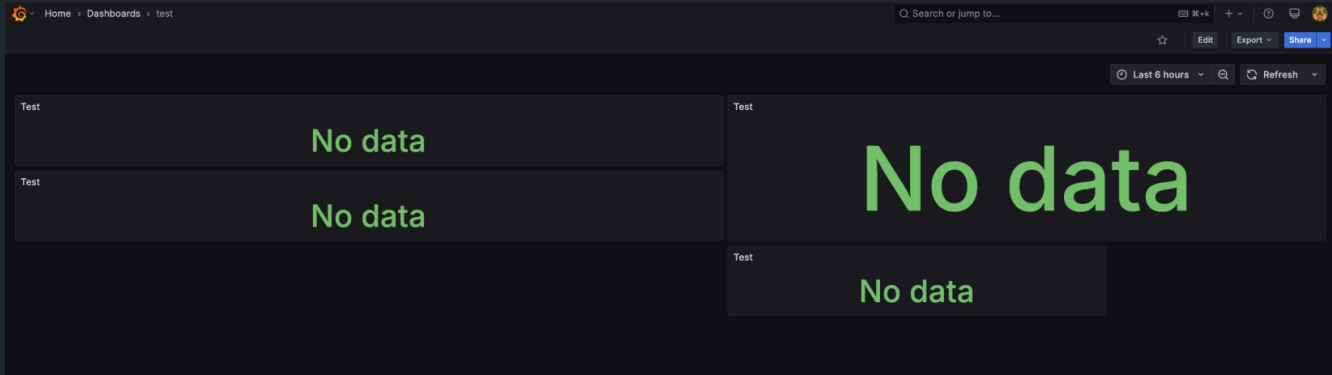
Description

Currently, `GridPos` is calculated by accumulating `currentX` & `currentY`. This sometimes leads to mysterious behaviour in combination with Grafana server.

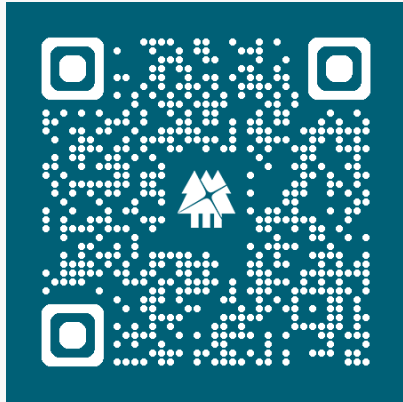
This is because

- [If some right panels are taller than other left panels, `currentY` will be the maximum height](#)
- Grafana automatically pushes panels up when importing boards

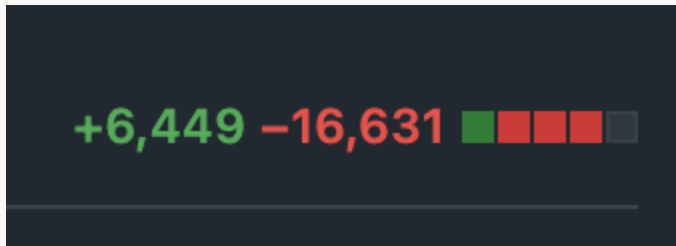
Minimum Example



結果



- コード量を1万行近く削減
- ほぼ同じボードを再現



Board	Before	After
Agent Memory		
Cluster Overview		
Discoverer		
LB Gateway		
Agent		
Benchmark Operator		
Helm Operator		
Index Correction		
Index Manager		

最新の取り組み

1. Grafana Foundation SDKを用いたGrafana Dashboard as Code
2. E2E V2: YAMLで宣言的にテストシナリオを記述

E2Eの悩み

- CRUDの処理がたくさんあるが、コードで書いていると冗長になりがち
- 違うコードベースに同じようなコードが散らばる
- データを取り出してアサートする流れも煩雑になりがち

E2E V2: YAMLで宣言的にテストシナリオを記述

- 得られた成果
 - ジェネリクスを用いた汎用k8s, gRPCクライアント
 - 別環境でもYAMLをもとにk8s JobでE2Eが走る
 - パスで結果を取り出し、アサートできる
 - 並列実行・Loop処理を用いた負荷試験
- Future Work
 - PBT: Property Based Testing

```
- name: Insert
  type: insert
  mode: unary
  parallelism: 10
  num: 60000
  qps: 3000
  wait: 5s
- mode: unary
  name: CreateIndex
  type: create_index
  expect:
    - status_code: ok
- mode: unary
  name: IndexInfo
  type: index_info
  expect:
    - status_code: ok
      path: $.stored
      op: gt
      value: 30000
```


ベクトルDB選定のすすめ

- CNCFにはハイブリッドサーチをサポートするOpenSearchもあり、検索用途で 👍
- 推薦・検出などベクトル検索だけで必要で、パフォーマンス重視の方にはValdは 👍

検索手法	エンジン	90 %ile (ms)	99 %ile (ms)	MRR
全文検索	OpenSearch	10.42	23.79	0.605
ハイブリッド サーチ	OpenSearch	21.56	28.823	0.661
ベクトル検索	OpenSearch	9.60	11.87	0.619
ベクトル検索	Vald	1.93	2.363	0.615

検索エンジン選定ガイド：ベクトル検索・全文検索からハイブリッドサーチまで
LINEヤフー Tech Blog

Contributions are Welcome!



vald.vdaas.org