

# Grafana Foundation SDK を使った Grafana Dashboard as Code

OSSベクトルデータベースValdチーム Matts966

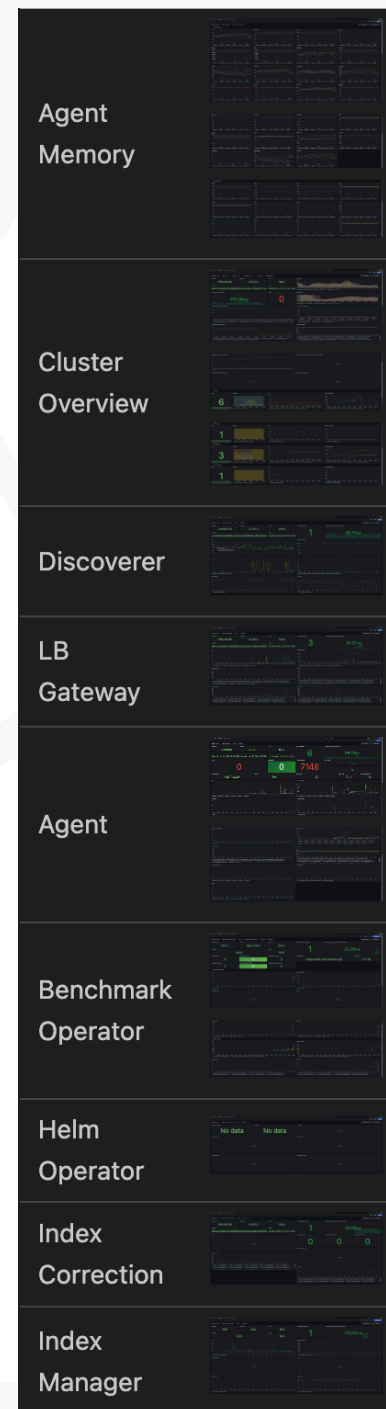
# 自己紹介

松井誠泰

- OSSのベクトルデータベースValdチームに転職して2ヶ月目
- 趣味
  -       
- [github.com/Matts966](https://github.com/Matts966)

# Grafanaボード管理の課題

- 似たパネルをたくさん管理
  - コンポーネント毎に微妙に違う
    - 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
- シンプルなパネルでも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



## Grafana Dashboard as Codeの選択肢 - JSONベース

方法	概要	特徴・注意点
JSON管理（元の手法）	GUIで作成後にJSON出力	単純・最小構成向け、再利用や共通化は弱い
Terraform Provider for Grafana	IaC統合（HCL）	JSON構造の記述が必要、Terraformに統合できる
Git Sync	GUI変更を自動でGit同期（Grafana 12以降）	GUI派に便利、繰り返しや再利用には不向き

# Grafana Dashboard as Codeの選択肢 - コードベース

方法	概要	特徴・注意点
Grizzly	CLIでリソースとして管理可能	CLIが便利・Jsonnet使える
Grafonnet	Jsonnetで生成	繰り返し処理など対応
Grabana	Goで記述、宣言的	唯一JSON逆生成可能、開発は <code>grafana-foundation-sdk</code> に移行傾向
<code>grafana-foundation-sdk</code>	公式SDK (Go等)	★本日のお題★



## 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



# 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
- 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

- 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

```
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

- 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()
```

## デメリット

- grabanaではサポートされていたJSONからのコード生成がない
  - 最初導入する時だけはちょっと大変
- GUIでの操作ができない
  - やるとすると、操作の手順を覚えて関数呼び出しに書き直すイメージ
  - ここが気になる場合、 Grizzly や Git Sync、自前の自動化がおすすめ

# 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?

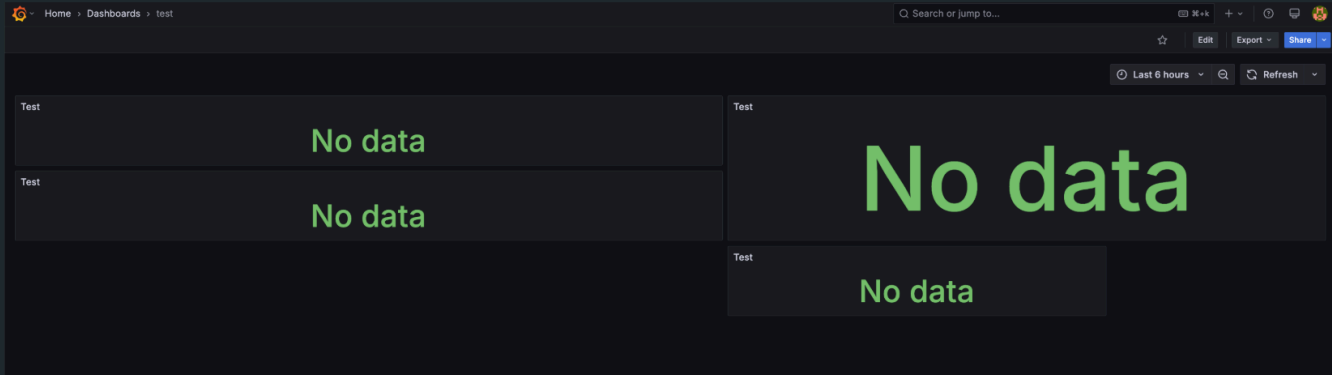
#### Description

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

This is because

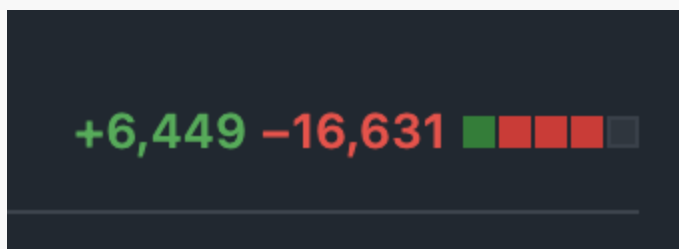
- [If some right panels are taller than other left panels, `currentY` will be the maximum height](#)
- Grafana automatically pushes panels up when importing boards

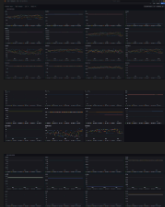
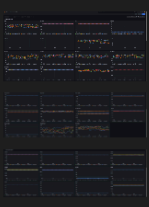
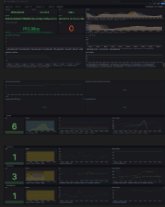
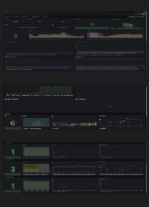




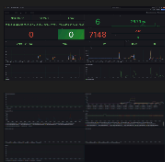



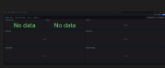





#### Minimum Example



# 結果

- [github.com/vdaas/vald/pull/2937](https://github.com/vdaas/vald/pull/2937)
- コード量を削減
- 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		

## おすすめの選び方

- 繰り返しが少ない → Git Sync, Grizzly や JSON エクスポート
- 再利用性重視 → Grabana / Grafonnet / grafana-foundation-sdk
  - 今後Go/TypeScript/Python/Javaで自動化していくなら grafana-foundation-sdk  
がおすすめ

## 参考リンク

- [Three years of Grafana dashboards as code](#)
  - `grabana` の作者の方で、今は Grafana Labs で `grafana-foundation-sdk` を開発されている方のブログ
- [grafana-foundation-sdk GitHub](#)

# Contributions are Welcome!



[vald.vdaas.org](https://vald.vdaas.org)