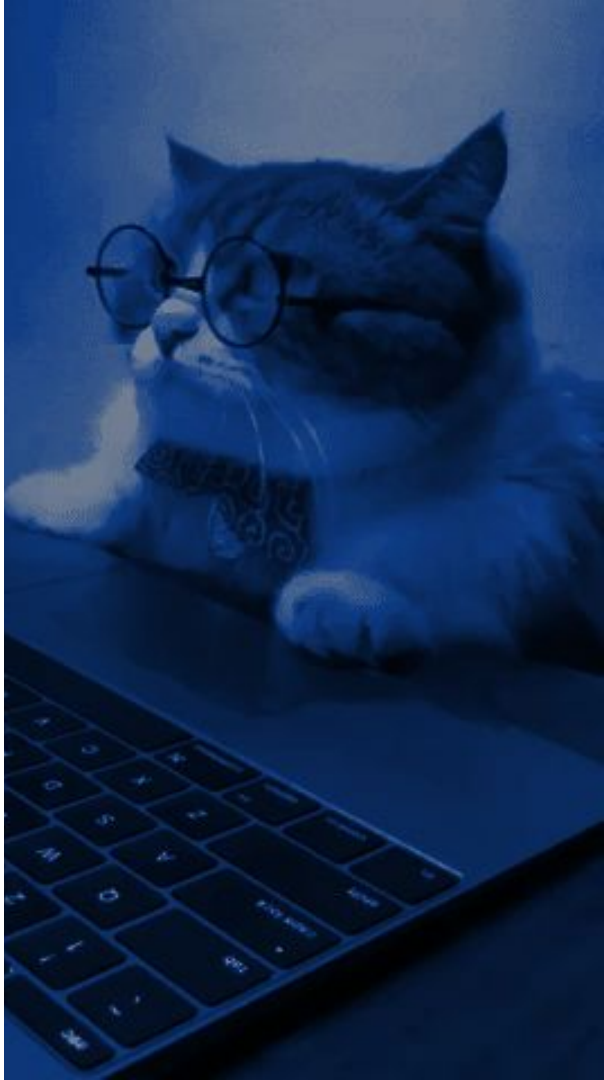
Observabilité : dépoussiérer Prometheus avec

VICTORIA METRICS

devfest

@ju_hnny5



~#whoami

Julien Briault 😎

(ex) SecOps consultant chez **Rudder**

IT/Infrastructure Manager (bénévole) aux Restaurants du Cœur

Network Engineer / SRE chez deezer

Auteur principal sur blog.jbriault.fr

#Networking #FOSS #Dev #Music

@ju_hnny5


Et oui, je ne dors pas beaucoup…😅

Julien Briault
@ju_hnny5

Actuellement aux @restosducoeur nous recherchons des dons de pc portables, d'écran (avec HDMI), raspberry, etc.

Si jamais vous connaissez des entreprises ouvertes aux dons, n'hésitez pas à me contacter. 🙏

Petit historique
de la supervision à l'observabilité 😇


La préhistoire 😵

Années 90/2000 :
Nagios - “la supervision” 🥸

Nagios, créé à la **fin des années 1990**, était l'un des outils de surveillance les plus populaires au début des années 2000.

Il était principalement axé sur la **surveillance de l'état des systèmes et des services**, vérifiant si des services spécifiques étaient en cours d'exécution. Générait des alertes en cas de problèmes.

Nagios®

Années 90/2000 :
Nagios - “la supervision” 🤓

- Une grande variété de “Nagios plugins” (exemple : **check_nrpe**)
  - Majoritairement des scripts (Perl, Bash, Python)
  - Utilise principalement le (trap) SNMP
- Donné “naissance” à un grand nombre de solutions :
  - Centreon
  - Zabbix*
  - Icinga
  - Shinken

ZABBIX

Shinken™

centreon


Les métriques 🙄

[me-trik] : **“La science de la mesure”**

Des **indicateurs exploitables**
vs **indicateurs de vanité**

Volume
Vélocité
Variété

Comment les stocker ?

Les TSDBs (Time Series Database)

Une TSDB (Time Series Database) est une base de données conçue spécifiquement pour stocker et analyser efficacement des données organisées sous forme de séries temporelles. 😍

Autrement dit, ***c'est un type de base de données qui stock des valeurs qui évoluent dans le temps***.

Les TSDBs (Time Series Database)

Plusieurs objectifs :

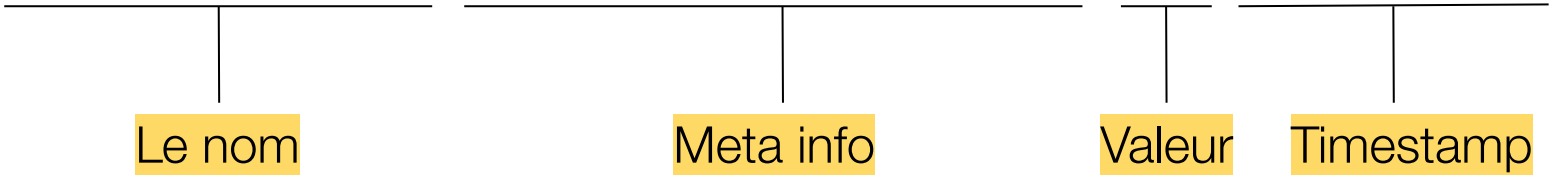
- Observer
- Alerter
- Corréler

Le standard : **OpenMetrics** :

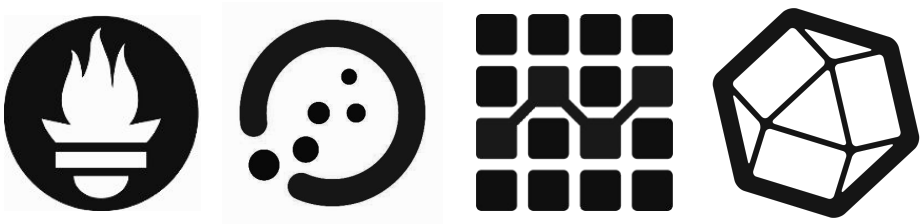
- Fournit une specification : https://github.com/OpenObservability/OpenMetrics/blob/main/specification/OpenMetrics.md
  - Format : `metric{label=valeur_label}`
  - Exemple :
    ```
    pdns_auth_uptime{instance="$host",dc="$dc"}
    ```

La structure des indicateurs (metrics) 🤨

```
requests_total{path="/", code:"200"} 10 1674518400
requests_total{path="/", code:"403"} 1 1674518410
```



Des solutions ? 🤫

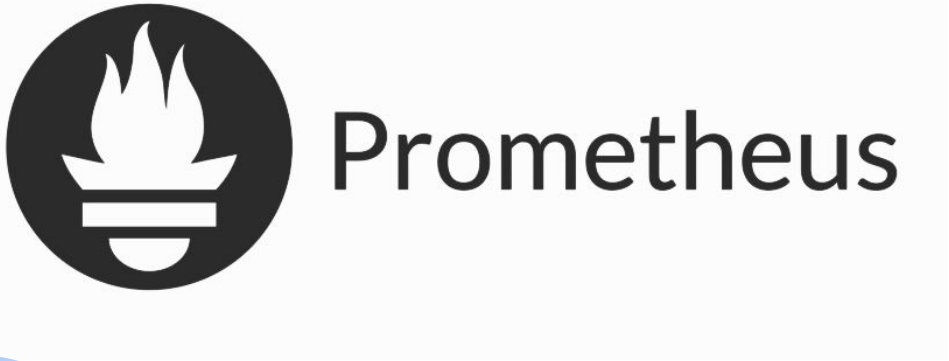


Des solutions ?

Des solutions ?

Des solutions ? 😬

Des solutions ? 😏



Standard de facto : **Prometheus** 😉

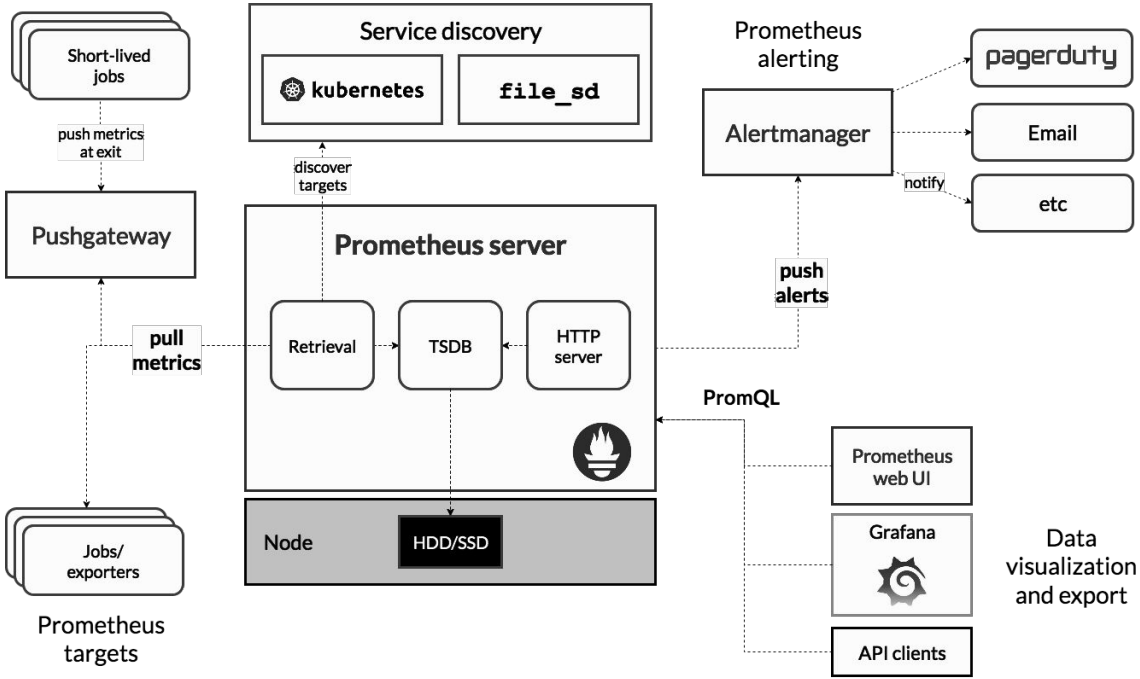
- Création en 2012 par **SoundCloud**
- Rendu Open Source en 2015 sous licence **Apache 2**.
- Intégré à la CNCF en 2016

Fournit plusieurs outils :

- `Alertmanager` pour gérer les alertes
- Prometheus Server (intégrant une Web UI)
- Push gateway

Source : https://developers.soundcloud.com/blog/prometheus-has-come-of-age-a-reflection-on-the-development-of-an-open-source-project

Standard de facto : **Prometheus** 😏



Source : https://prometheus.io/docs/introduction/overview/

Supervision par la collecte de métriques 🤔

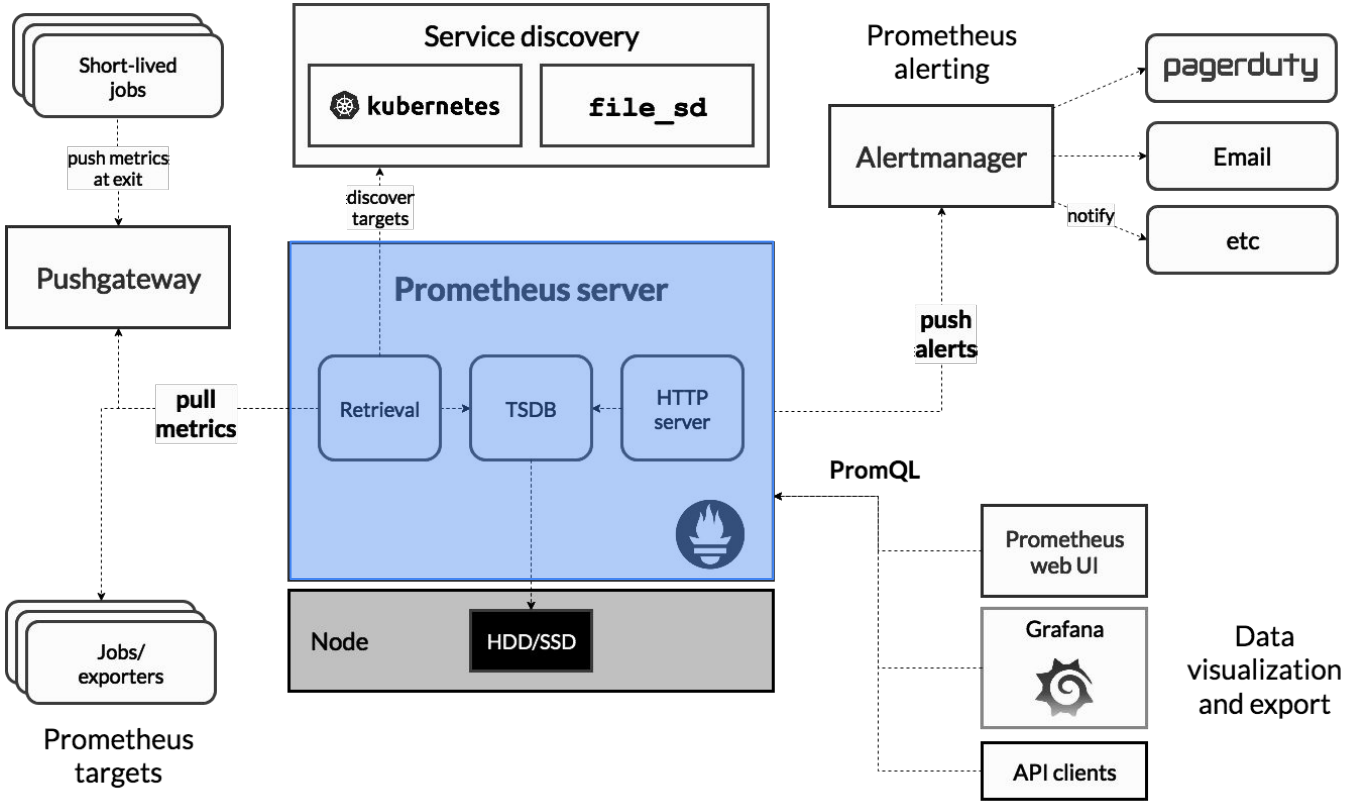
Standard de facto : **Prometheus** 😏

- Agrège ses métriques sur les 15 derniers jours dans sa TSDB (par défaut)
  - Configurable sur une plus longue durée mais ça peut rapidement être un problème :
    - Coûts de stockage
    - Ressources utilisées
  - En cas de dysfonctionnement :
    - Perte des données s'il n'y a pas de backup. 😤

Mais pas sans **défauts** …

- Stockage local
- Évolutivité verticale*
- Rétention de données
- Haute disponibilité = 🤯
- Gestion des alertes
- Mono-tenant
- Complexité opérationnelle**

Mais pas sans **défauts …** 😅



@ju_hnny5


Thanos

Un peu de glue : **Thanos** 🧐

- Infrastructure bien plus complexe*
- Une utilisation des ressources bien plus conséquente
  - Même s'il est possible de partager la charge grâce au sharding.
- Difficile à opérer 🤫


VICTORIA
METRICS

VictoriaMetrics : **Un vent de modernité**

- Solution jeune (2018)
- Solution **Open Source** (Apache 2.0) d'observabilité/supervision et une **TSDB**.
- Support officiellement pleins de protocoles pour l'ingestion de données (push/pull) :
  - Prometheus (exporters) & Prometheus
  - InfluxDB
  - Graphite
  - DataDog Agent
  - OpenTSDB
  - CSV, JSON

  = Modèle de données sans schéma (schemaless)

VICTORIA METRICS

VictoriaMetrics : **Un vent de modernité**

- Fortement inspiré de l'éco-système Prometheus 🤓
- Ecrit en Go(lang)
- Découverte et scrape les cibles Prometheus (et Kubernetes naturellement) 😱
- D'une simplicité enfantine pour l'opérer et le mettre en place (que ça soit en *single-node* ou en *cluster mode*).

VICTORIA METRICS

VictoriaMetrics : **Un vent de modernité**

- Consomme beaucoup moins de ressources
- Des performances bien meilleures qu'avec le combo Thanos/Prometheus 😏

VICTORIA METRICS

VictoriaMetrics : **Un vent de modernité**

- Permet de faire du stockage longue durée mais également de la collecte
  - Rétention minimum de 24h
  - 1 mois par défaut
  - `-retentionPeriod 1200` = 100 ans
- Il peut être utilisé en **backend Prometheus** (cf : démo)
- Possède son propre agent (`vmagent`) pour ingérer les données 😱

Deux versions

Community 🤭

Enterprise 🤫

VictoriaMetrics : **Un vent de modernité**

- A l'inverse de Prometheus qui ne fonctionne par défaut qu'en mode Pull
  - Utilisation de la `pushgateway` pour le mode Push
- VictoriaMetrics supporte nativement les modes **Pull** & **Push**

VictoriaMetrics : **Quelques chiffres**

- 10x moins de RAM qu'InfluxDB
- Jusqu'à 7x moins de RAM que Prom/Thanos|Cortex
- Peut recevoir jusqu'à 70 fois plus de données que TimescaleDB
- Consomme environ 7x moins d'espace de Stockage que Prometheus/Thanos|Cortex

Benchmark réalisé grâce à l'outil de **TimescaleDB**.

Source :
https://valyala.medium.com/measuring-vertical-scalability-for-time-series-databases-in-google-cloud-92550d78d8ae
https://github.com/VictoriaMetrics/VictoriaMetrics#prominent-features

Quelques (gros) utilisateurs

adidas

Les Restaurants du Cœur
Les Relais du Cœur

WiX

CRITEO

ROBLOX

PERCONA

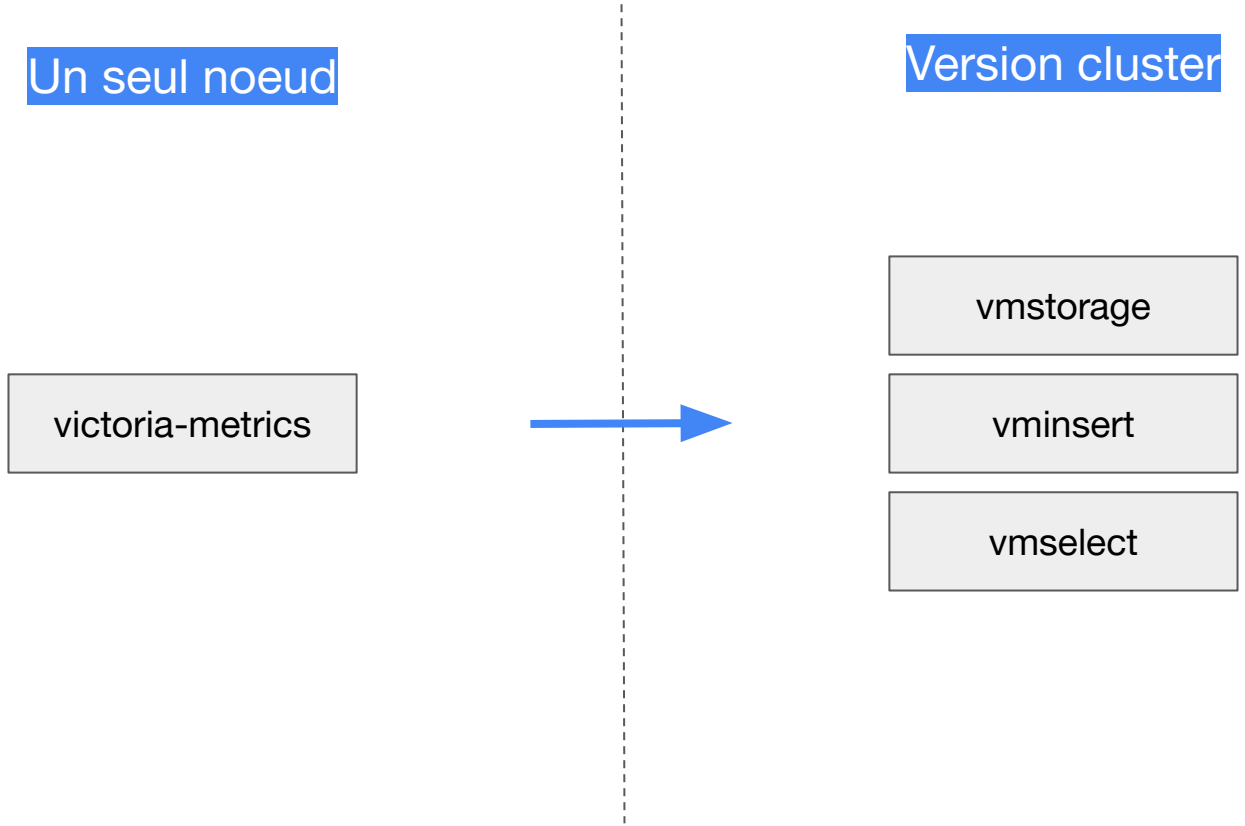
Single-node = **Scales vertically**

Cluster = **Scales horizontally** 🤗

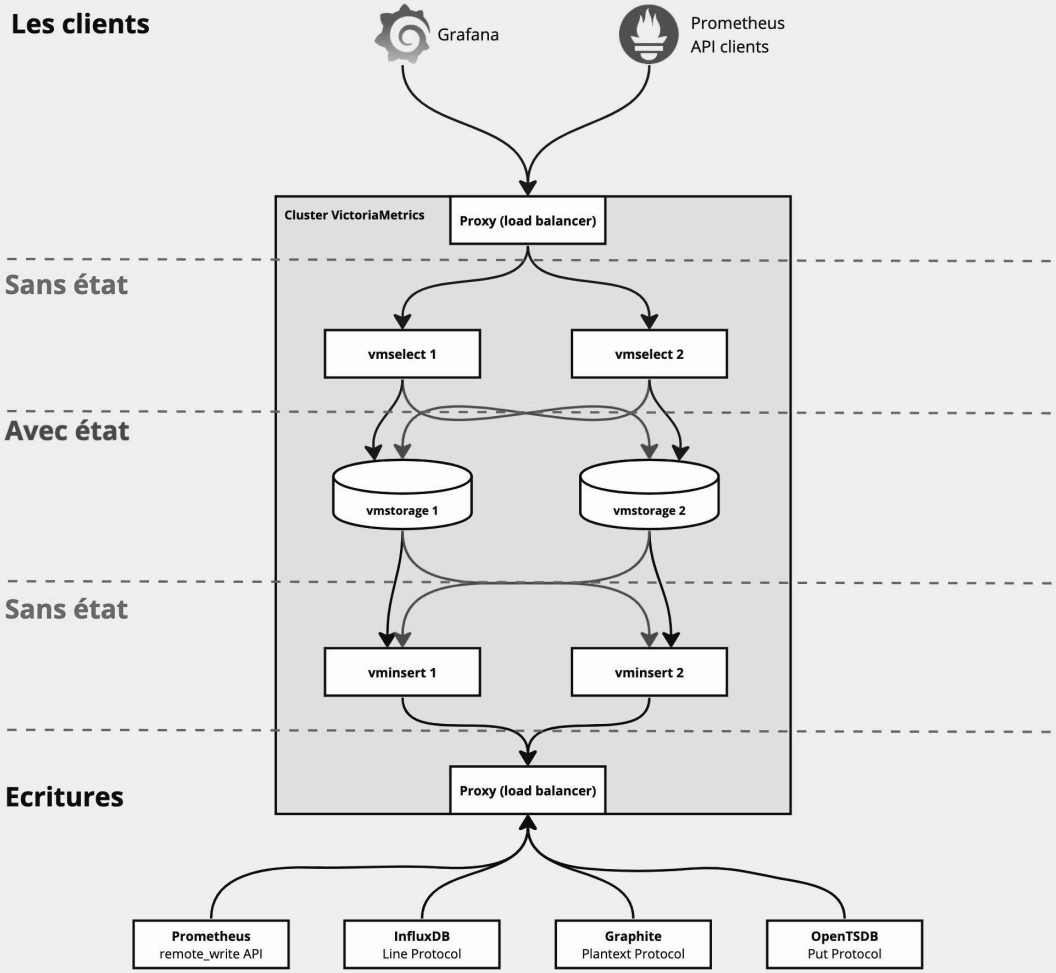
Ces deux modes partagent
le même code.

VictoriaMetrics : **Une architecture simple**

😇

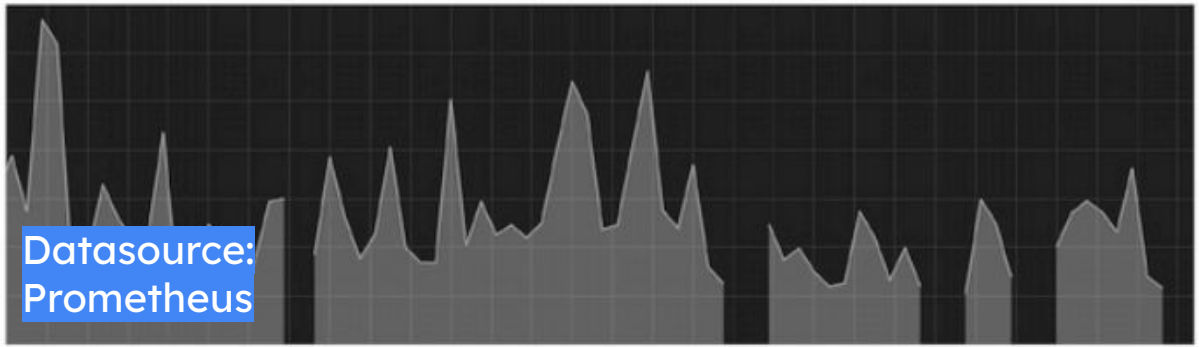


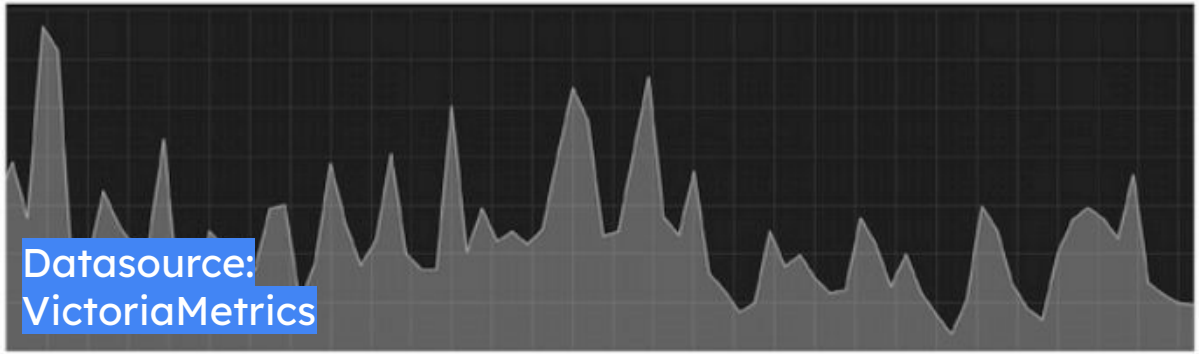
VictoriaMetrics : **Une architecture simple** 😇



@ju_hnny5

VictoriaMetrics (noeud seul) :





Source : https://shorturl.at/nxNX2


Multi-tenancy ? 🙄

VictoriaMetrics : **Namespaces** 😎

- Tenants identifiés par `accountID` ou `accountID:projectID` 😲
- Lister les tenants enregistrés via :
  - `http://<vmselect>:8481/admin/tenants`
- **`vminsert`** peut accepter des données provenants de plusieurs tenants via le point d'entrée `multitenant` :
  - `http://vminsert:8480/insert/multitenant/<suffix>`

Ne prend pas en charge l'interrogation de plusieurs tenants dans une seule requête.

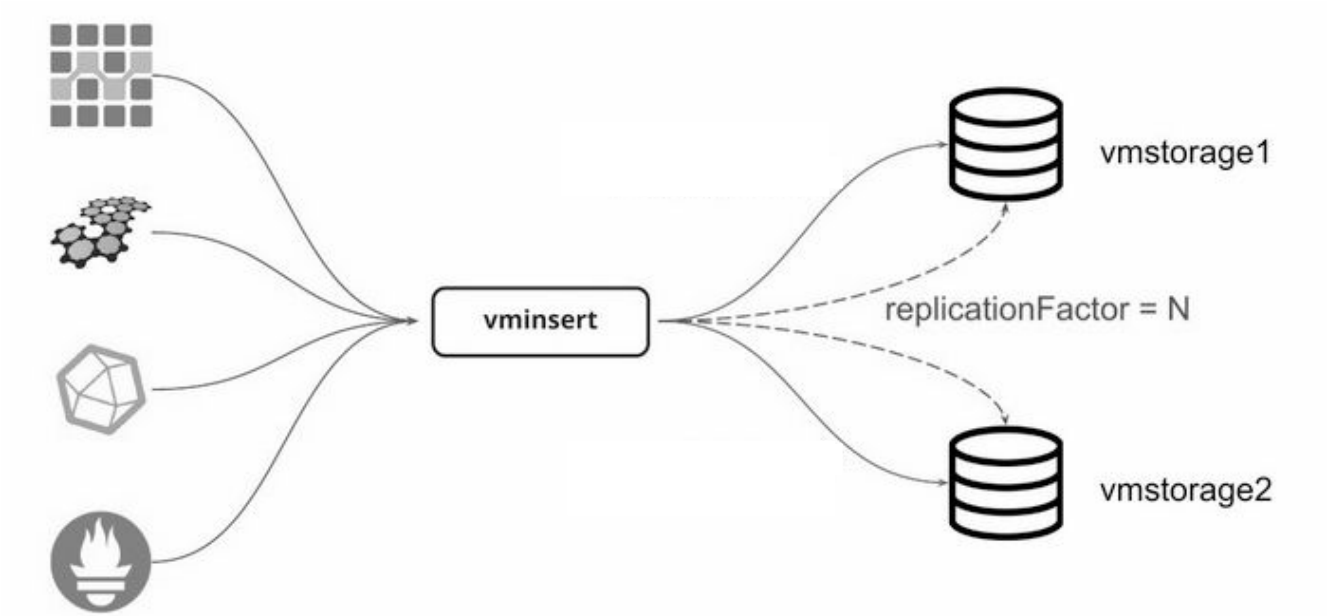
VictoriaMetrics : **MetricsQL** 😨

- Se veut une amélioration de **PromQL** en étant fortement inspiré
- Utilisable dans des dashboards **Grafana**
  - Ayant comme source **VictoriaMetrics** (`datasource_type: Prometheus`)
- Peut être utilisé de manière autonome (via le package `metricsql`) pour analyser MetricsQL dans des applications externes

https://valyala.medium.com/promql-tutorial-for-beginners-9ab455142085


Collecter
les métriques ? 🤩

VictoriaMetrics : **Ingestion de données** 🤩

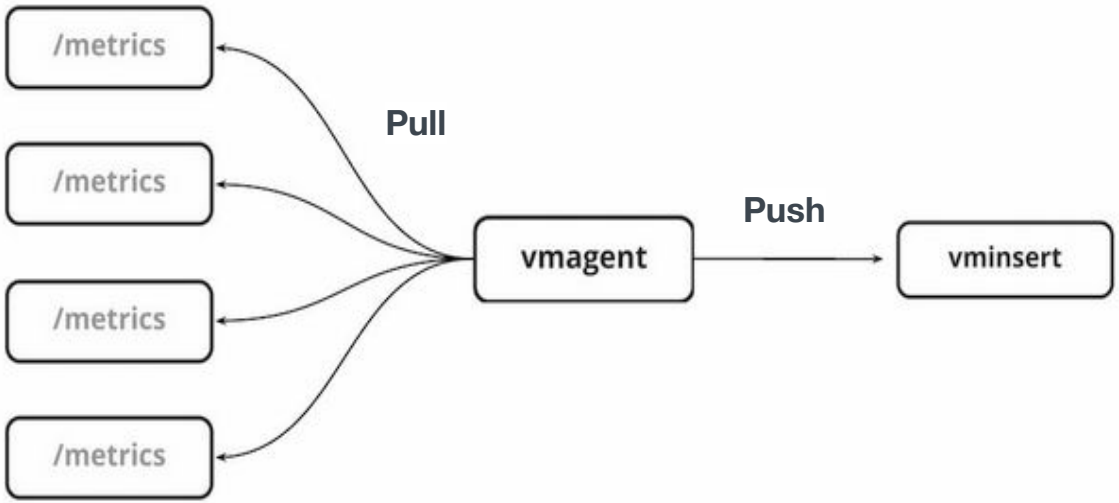


Source : https://shorturl.at/nxNX2

https://github.com/VictoriaMetrics/VictoriaMetrics/tree/master/app/vminsert

VictoriaMetrics : **Utilisation de l'agent** 🥸

Agent pour **collecter les métriques à partir de diverses sources**.

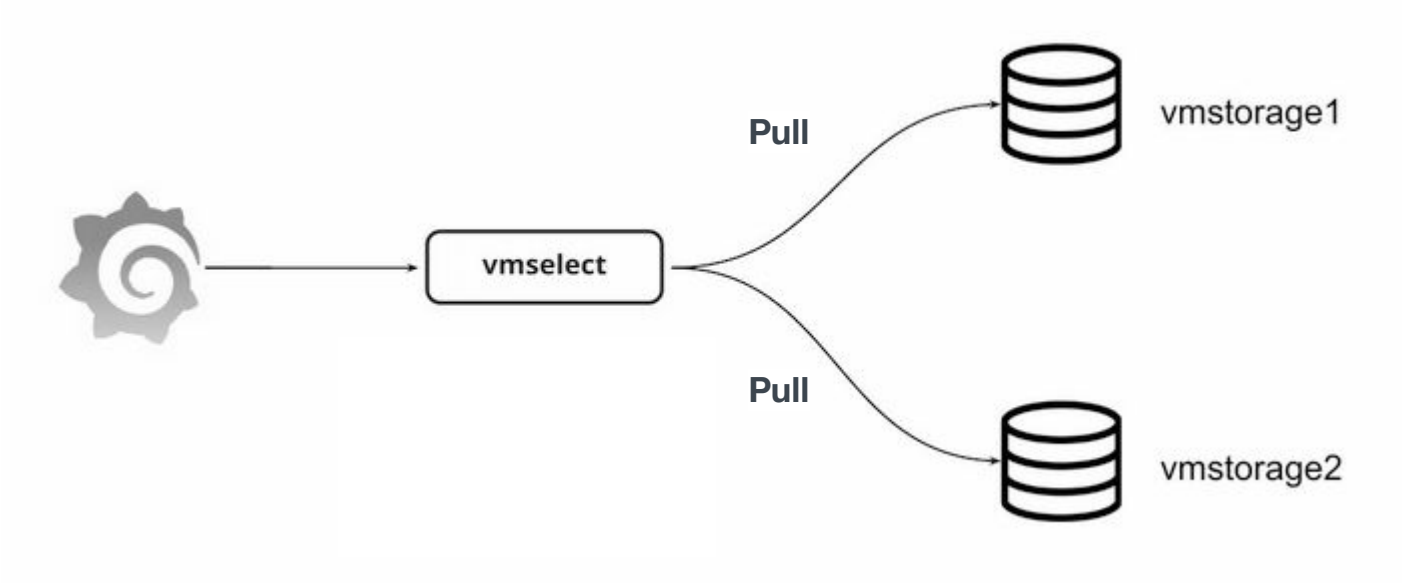


Source : https://shorturl.at/nxNX2
https://github.com/VictoriaMetrics/VictoriaMetrics/tree/master/app/vmagent

VictoriaMetrics : **Utilisation de l'agent** 🥸

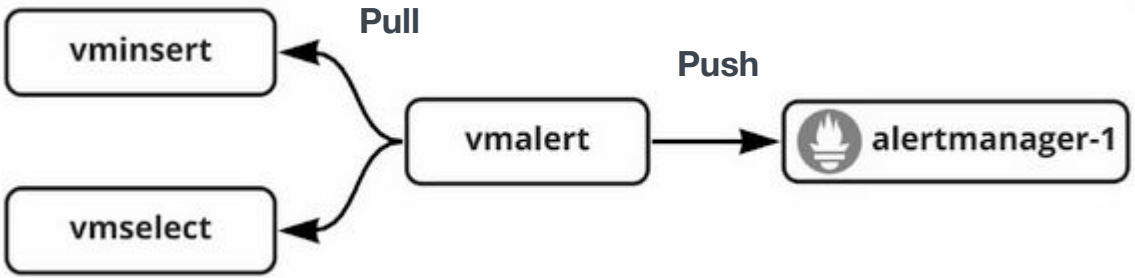
- Naturellement protégé des incidents réseau
- Compatible avec **Prometheus** :
  - Les configs de "scrape" Prometheus
  - Le module de découverte de services (service discovery)
  - Le système de "filtering" et "relabeling"
- Replication
- Clustering
- Ingère des données via les protocoles populaires de "push"
  - Exemple : **`remoteWrite`** de Prometheus

VictoriaMetrics : **Lecture des données** 😲



Source : https://shorturl.at/nxNX2
https://github.com/VictoriaMetrics/VictoriaMetrics/tree/master/app/vmselect

VictoriaMetrics : **Se faire alerter ! □**



Source : https://shorturl.at/nxNX2
https://github.com/VictoriaMetrics/VictoriaMetrics/tree/master/app/vmalert


Migrer les données

VictoriaMetrics : **Migrer ses données**

Une commande : **`vmctl`**.

Permet de migrer les données de :

- Prometheus via l'API de snapshot
- Thanos, Cortex, Mimir
- InfluxDB
- OpenTSDB
- Promscale
- Entre noeuds VictoriaMetrics (`single` ou `cluster`)

Source : https://docs.victoriametrics.com/vmctl.html

VictoriaMetrics : **Migrer ses données**

Exemple :

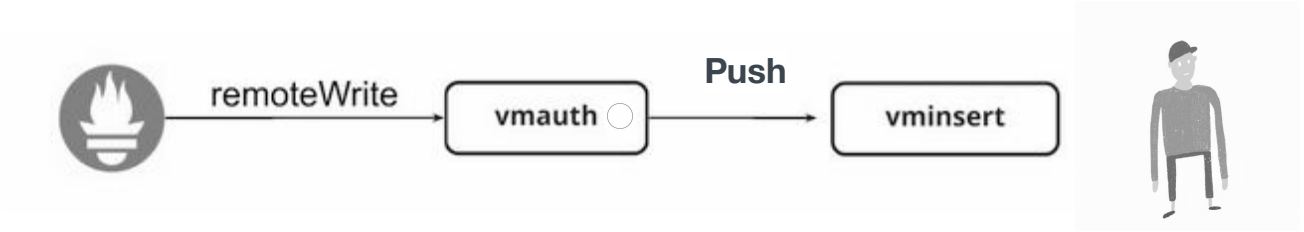
```
vmctl prometheus --prom-snapshot=/path/to/snapshot \
  --prom-filter-time-start=2020-02-07T00:07:01Z \
  --prom-filter-time-end=2020-02-11T00:07:01Z
```

Source : https://docs.victoriametrics.com/vmctl.html


Un peu de
contrôle

VictoriaMetrics : **Ajouter de l'authentification**

- Authentification basique au travers d'HTTP(S).
- Le client envoie une requête à **`vmauth`** en spécifiant le mot de passe, si `vmauth` authetifie correctement cette requête, alors la requête est envoyée à VictoriaMetrics (`vminsert`). 😯

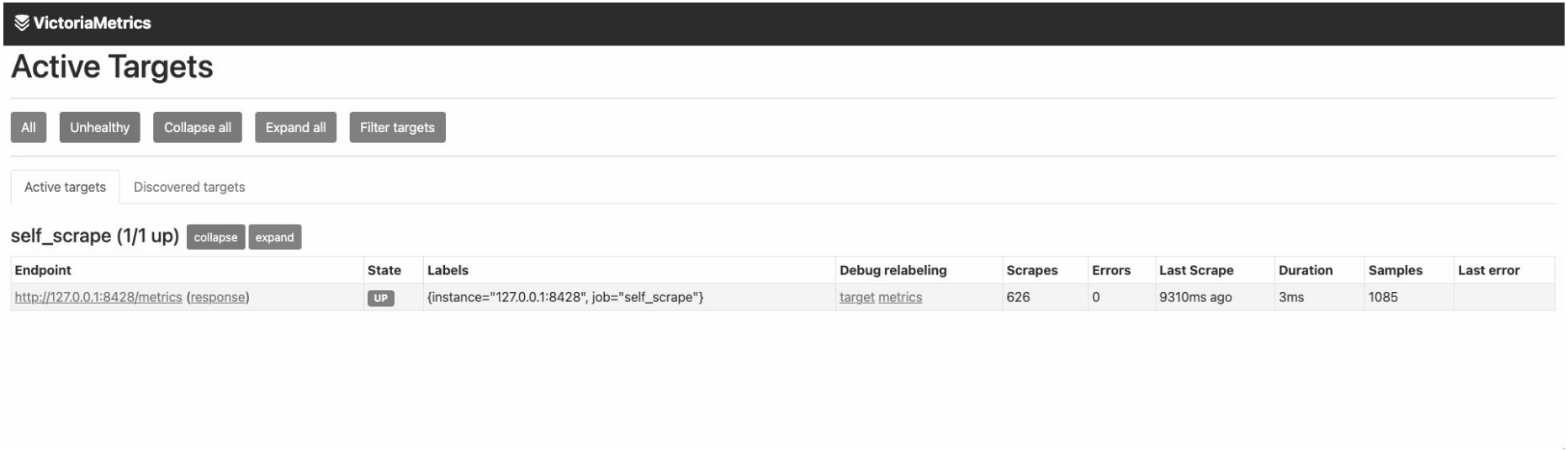


Source : https://shorturl.at/nxNX2
https://github.com/VictoriaMetrics/VictoriaMetrics/tree/master/app/vmauth

VictoriaMetrics : **Une Web UI** (vmui) 🤗

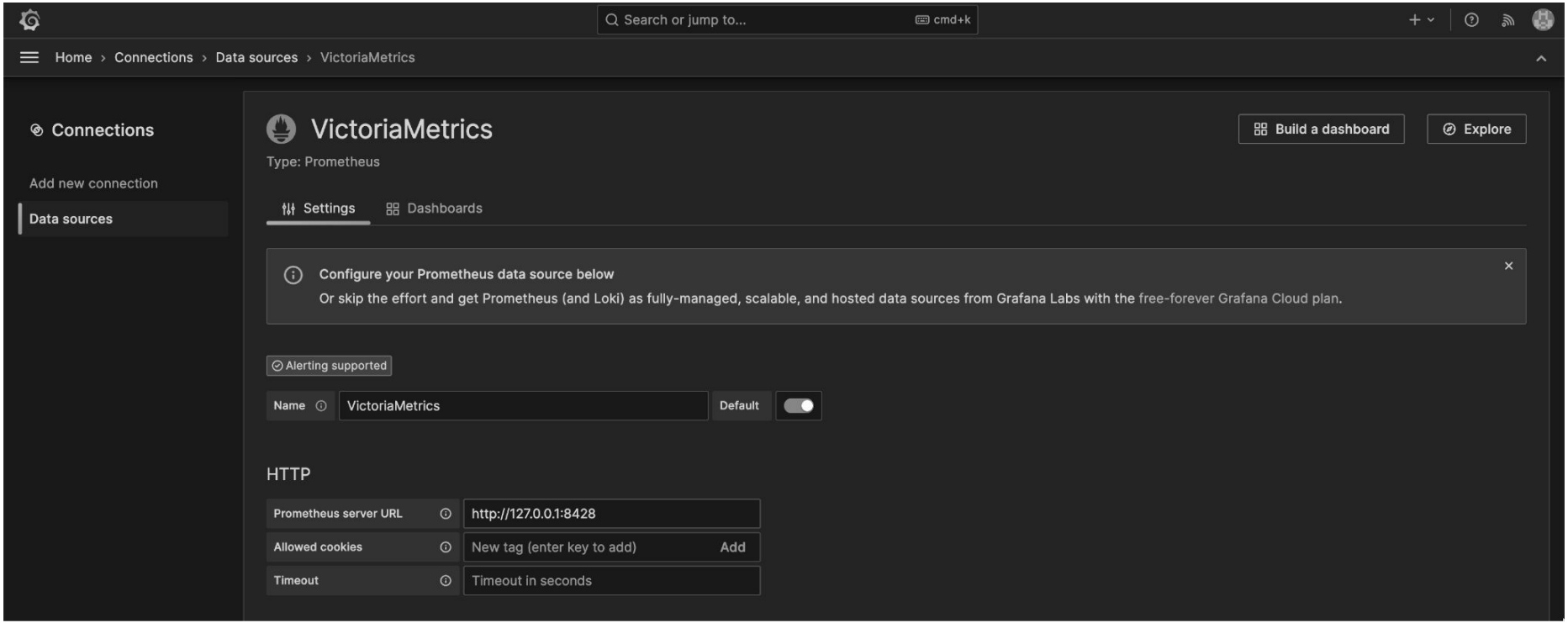
@ju_hnny5 https://github.com/VictoriaMetrics/VictoriaMetrics/tree/master/app/vmui

VictoriaMetrics : **Une Web UI** 🤗



@ju_hnny5

Intégration native : **Grafana**

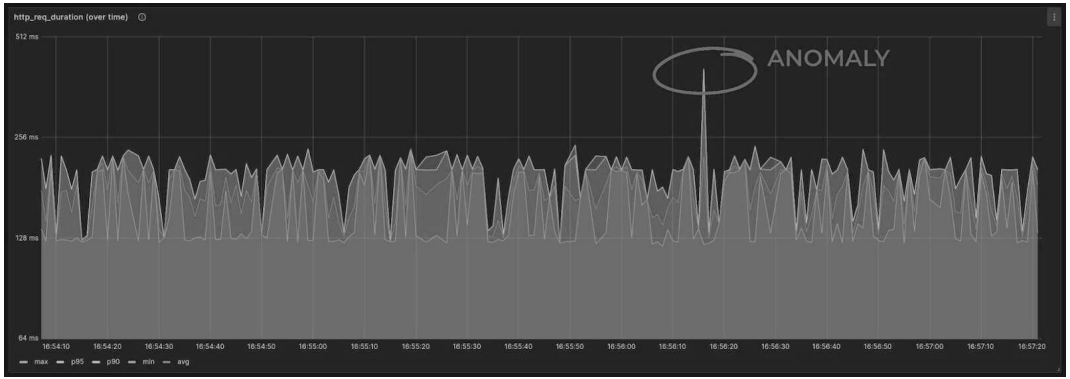


@ju_hnny5


vmgateway, vmbackupmanager
et vmanomaly

VictoriaMetrics : **Détection des anomalies** 😍

- `vmanomaly` : Un service qui analyse en permanence les données dans **VictoriaMetrics** et utilise des mécanismes d'apprentissage automatique pour détecter les changements inattendus qui peuvent être utilisés dans les alertes.



VictoriaMetrics : **Intégration dans Kubernetes** (vmoperator)

Mode cluster dans Kubernetes (via Helm charts)

Simple d'accès :

```
helm install vmoperator vm/victoria-metrics-operator

kubectl --namespace default get pods -l
"app.kubernetes.io/instance=vmoperator"
```

https://github.com/VictoriaMetrics/helm-charts/tree/master/charts/victoria-metrics-operator

L’heure de la démo !* 🤤

*Sous réserve de temps et de l’effet démo

➡ https://play.victoriametrics.com

@ju_hnny5


Voila`

Merci

One link : https://shorturl.at/dgmrE


Merci pour
votre attention !