HKOSCon 2024

Metrics, Logs, Traces and Profiles with Grafana LGTM

Wong Hoi Sing, Edison

2024-07-06

Introduction

What is Grafana LGTM Stack?

Demo with AlviStack Vagrant Box

Create a Grafana Dashboard

Connect with Data Sources

Q&A

Introduction

Get the Code

- GitHub Repo: Source code
- GitHub Page: Online reveal.js, converted by pandoc
- index.pdf: Offline PDF, converted by pandoc
- Vagrantfile: For running demo

About Me

- Wong Hoi Sing, Edison (hswong3i)
- 2005: Drupal, Developer & Contributor
- 2008: HKDUG, Founder
- 2010: PantaRei Design, Founder
- 2020: HKOSCON 2020, Speaker
- 2021: AlviStack, Founder
- 2022: HKOSCON 2022, Speaker
- 2024: Most Active GitHub user in Hong Kong


3700
cervelo

DMC-FZ18 F2.8 1/30s ISO100


Microsoft
Red Hat

AlviStack Vagrant Box Packaging for Kubernetes

PantaRei Design Limited

Platform | Certified Kubernetes - Installer

This Vagrant Box provides Libvirt and VirtualBox image for running Kubernetes in single node all-in-one mode.

Repositories

alvistack/vagrant-kubernetes (primary)

https://github.com/alvistack/vagrant-kubernetes PRIMARY | Apache License 2.0 | good first issues 0 open

24	1	Oct '20	Jul '24	Jul '21
Stars	Contributors	First commit	Latest commit	Latest release

Participation stats

Languages

AlviStack – Hong Kong Based Kubernetes Distribution

Time: 10:15 – 11:15
Language: English

Search

SEARCH

Schedule

Speakers

Mr. Edison Wong Hoi Sing

Mr. Eason Lai

Ms. Amanda Lam

Mr. Cyrus Wong

Mr. Davide Benvegnu

Mr. Ivan Ma

github-actions[bot] Update Hong Kong ✓ 36fb4d9 · 3 weeks ago History

Preview | Code | Blame | 11368 lines (11354 loc) · 320 KB | Code 55% faster with GitHub Copilot | Raw

Top GitHub Users By Total Contributions in Hong Kong

Top GitHub Users passing | Views 827278

The `public contributions` and `private contributions` by users in Hong Kong on `2024/6/11 12:56 AM UTC`. This list contains users from `Hong Kong` and cities `Hong-kong` `Kowloon`.

There are `138 countries` and `674 cities` can be found here.

There are `930 users` in Hong Kong. You need at least `30 followers` to be on this list.

Don't forget to star ⭐ this repository

Top Users By Public Contributions | **Top Users By Total Contributions** | Top Users By Followers

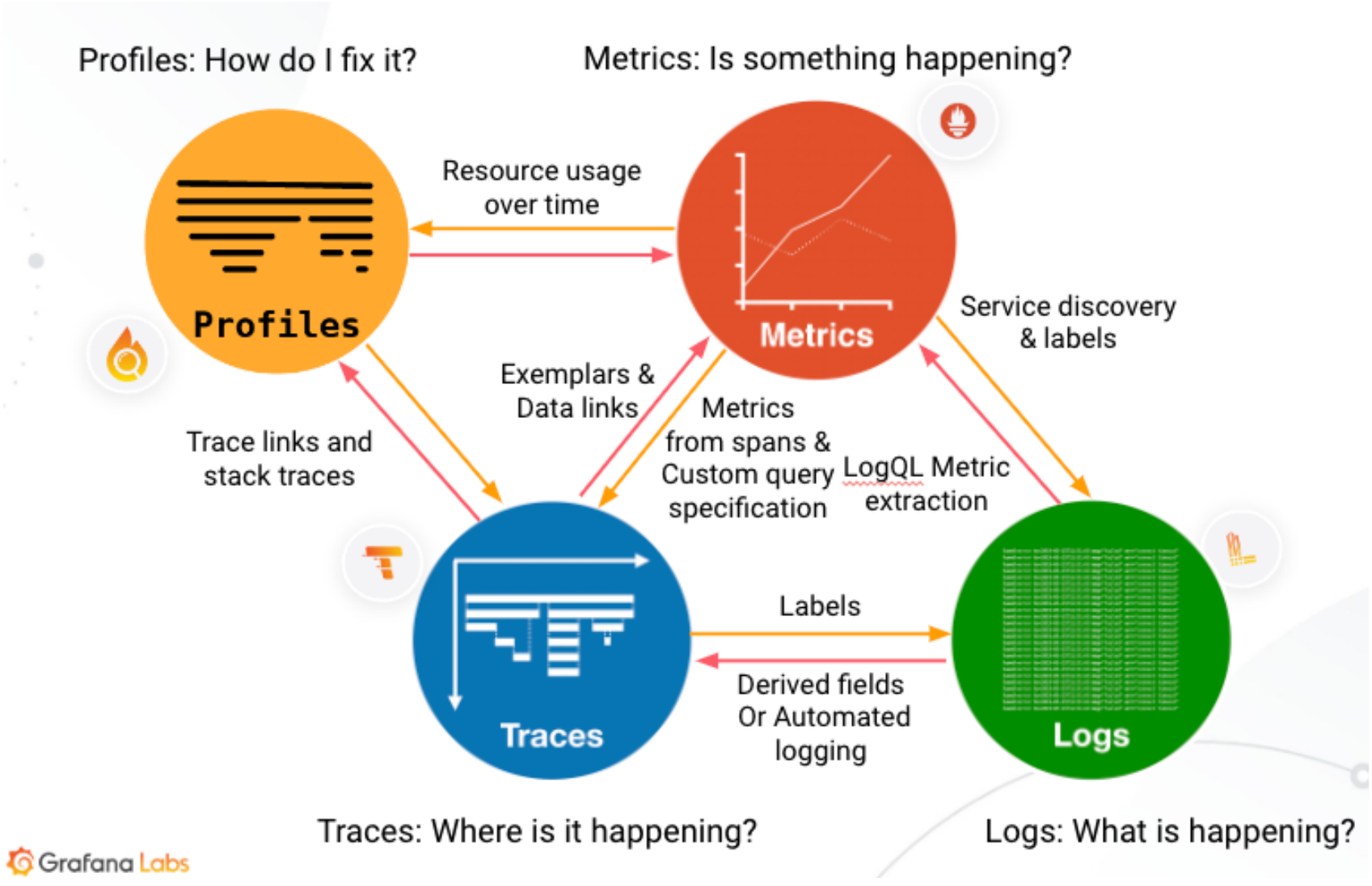
🚀 Share on

#	Name	Company	Twitter Username	Location	Public Contributions	Total Contributions
1	hswong3i Wong Hoi Sing Edison	Http://pantarei-desi	hswong3i	Hong Kong	42611	42611
2	BattlefieldDuck TatLead	No Company	BattlefieldDuck	Hong Kong	26207	30200
3	dirkarnez Dirk Arnez	Freelance	No Twitter Username	Hong Kong	7944	15359
	HTGAzureX1212	@teamhartex |	No Twitter			

What is Grafana LGTM Stack?

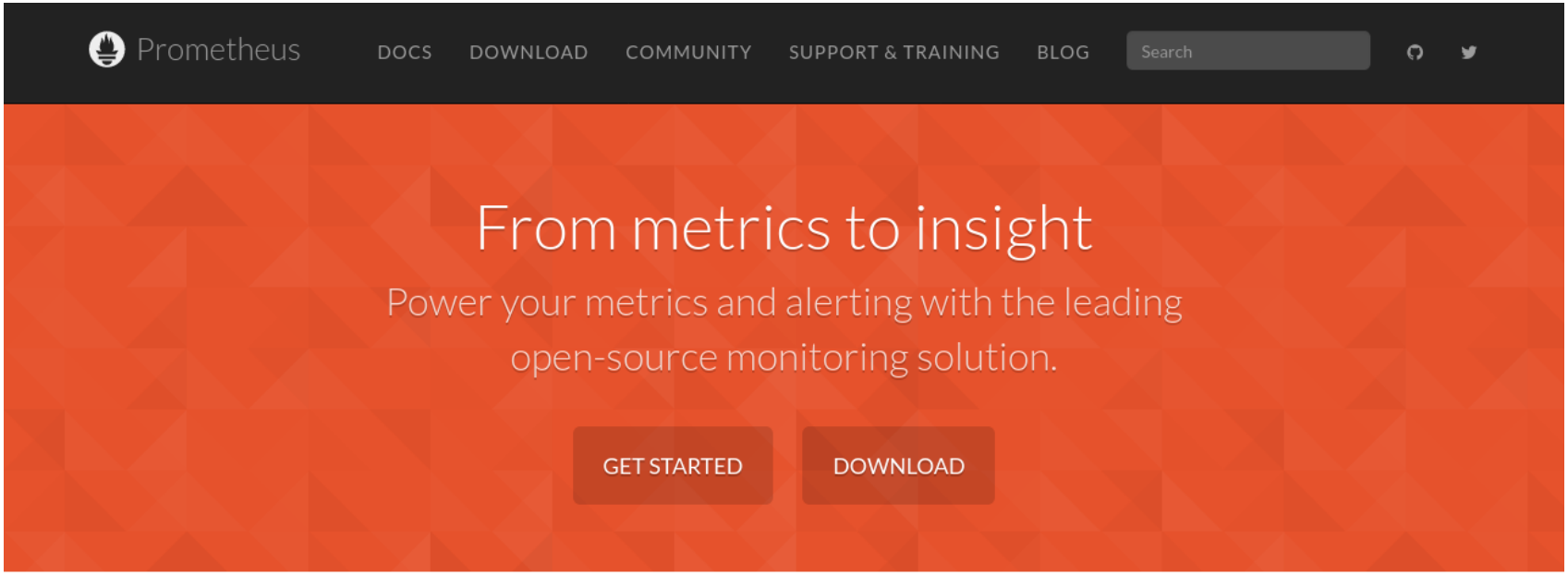
Prometheus + LGTM Stack + Pyroscope

- Monitoring: Collects and stores metrics as time series data
- Logs: What is happening?
- Metrics: Is something happening?
- Traces: Where is it happening?
- Profiles: How do I fix it?
- Dashboard: Data visualization and monitoring


Profiles: How do I fix it?
Metrics: Is something happening?
Profiles
Metrics
Resource usage over time
Service discovery & labels
Exemplars & Data links
Metrics from spans & Custom query specification
LogQL Metric extraction
Trace links and stack traces
Traces
Logs
Labels
Derived fields Or Automated logging
Traces: Where is it happening?
Logs: What is happening?
Grafana Labs

Prometheus

- Prometheus is a systems monitoring and alerting toolkit
- Multi-dimensional data model
- PromQL, a flexible query language to leverage this dimensionality
- 2nd CNCF graduated project after Kubernetes
- Apache License 2.0



Dimensional data

Prometheus implements a highly dimensional data model. Time series are identified by a metric name and a set of key-value pairs.

Powerful queries

PromQL allows slicing and dicing of collected time series data in order to generate ad-hoc graphs, tables, and alerts.

Great visualization

Prometheus has multiple modes for visualizing data: a built-in expression browser, Grafana integration, and a console template language.

Efficient storage

Prometheus stores time series in memory and on local disk in an efficient custom format. Scaling is achieved by functional sharding and federation.

Simple operation

Each server is independent for reliability, relying only on local storage. Written in Go, all binaries are statically linked and easy to deploy.

Precise alerting

Alerts are defined based on Prometheus's flexible PromQL and maintain dimensional information. An alertmanager handles notifications and silencing.

Many client libraries

Client libraries allow easy instrumentation of services. Over ten languages are supported already and custom libraries are easy to implement.

Many integrations

Existing exporters allow bridging of third-party data into Prometheus. Examples: system statistics, as well as Docker, HAProxy, StatsD, and JMX metrics.

- Prometheus Operator deploy with Kubernetes CRDs
- kube-prometheus provides example configurations for a complete cluster monitoring stack
- Here we are going to demo with kube-prometheus-stack with Helm chart style

kube-prometheus-stack

Installs the kube-prometheus stack, a collection of Kubernetes manifests, Grafana dashboards, and Prometheus rules combined with documentation and scripts to provide easy to operate end-to-end Kubernetes cluster monitoring with Prometheus using the Prometheus Operator.

See the kube-prometheus README for details about components, dashboards, and alerts.

Note: This chart was formerly named `prometheus-operator` chart, now renamed to more clearly reflect that it installs the `kube-prometheus` project stack, within which Prometheus Operator is only one component.

Prerequisites

- Kubernetes 1.19+
- Helm 3+

Get Helm Repository Info

```
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
helm repo update
```

See `helm repo` for command documentation.

Install Helm Chart

```
helm install [RELEASE_NAME] prometheus-community/kube-prometheus-stack
```

See configuration below.

See helm install for command documentation.

Dependencies

Grafana

- Grafana is a multi-platform open source analytics and interactive visualization web application
- It can produce charts, graphs, and alerts for the web when connected to supported data sources
- AGPLv3

Grafana Labs Products Open source Solutions Learn Docs Company Downloads Contact us Sign in

Star us on GitHub | ★ 61,456

Grafana OSS

Visualize your data, optimize your performance

Easily collect, correlate, and visualize data with beautiful dashboards using Grafana — the open source data visualization and monitoring solution that drives informed decisions, enhances system performance, and streamlines troubleshooting.

The easiest way to get started with the latest version of Grafana is with Grafana Cloud, our fully managed observability stack.

Download Docs GitHub

Transform data into real-time insights

Grafana's growing suite of visualizations, ranging from time series graphs to heatmaps to cutting-edge 3D charts, help you decode complex datasets.

Seamlessly build a single pane of glass

With 150+ Grafana plugins, you can unify all your data sources into a single dashboard to streamline data monitoring and troubleshooting.

Resolve issues faster with Grafana Alerting

Alert on data from a wide variety of data sources to identify problems and fix them quickly.

The evolution of Grafana

Inspired by the open source community and our growing number of users, Grafana introduces new innovations in every release to improve how you visualize, correlate, and share your data.

- Support multiple installation style, e.g. monolithic, microservice, or even manually
- Here we are going to demo with distributed Helm chart style

Grafana Labs documentation

Search docs

Product

Grafana

Viewing: v11.1 (latest)

Find another version

Grafana documentation

- What's new
- Breaking changes
- Upgrade Grafana
- About Grafana
- Introduction
- Get started with Grafana Open Source
- Set up
  - Install Grafana
    - Debian or Ubuntu
    - RHEL or Fedora
    - SUSE or openSUSE
    - Grafana Docker image
    - Grafana on Kubernetes
    - Grafana on Helm Charts
    - macOS
    - Windows
  - Configure Grafana
  - Start Grafana
  - Sign in to Grafana

Documentation > Grafana documentation > Set up > Install Grafana > Grafana on Helm Charts

Open source

Deploy Grafana using Helm Charts

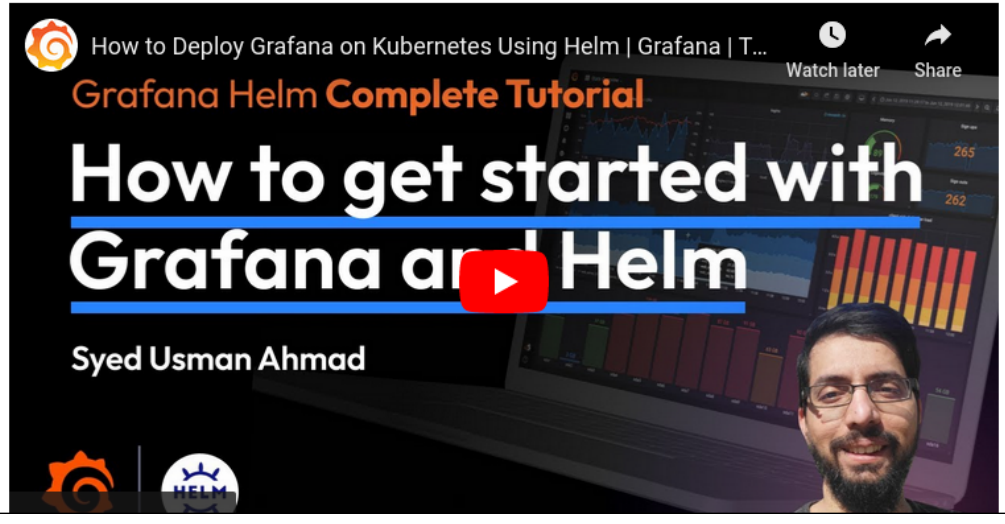
This topic includes instructions for installing and running Grafana on Kubernetes using Helm Charts.

Helm is an open-source command line tool used for managing Kubernetes applications. It is a graduate project in the CNCF Landscape.

NOTE

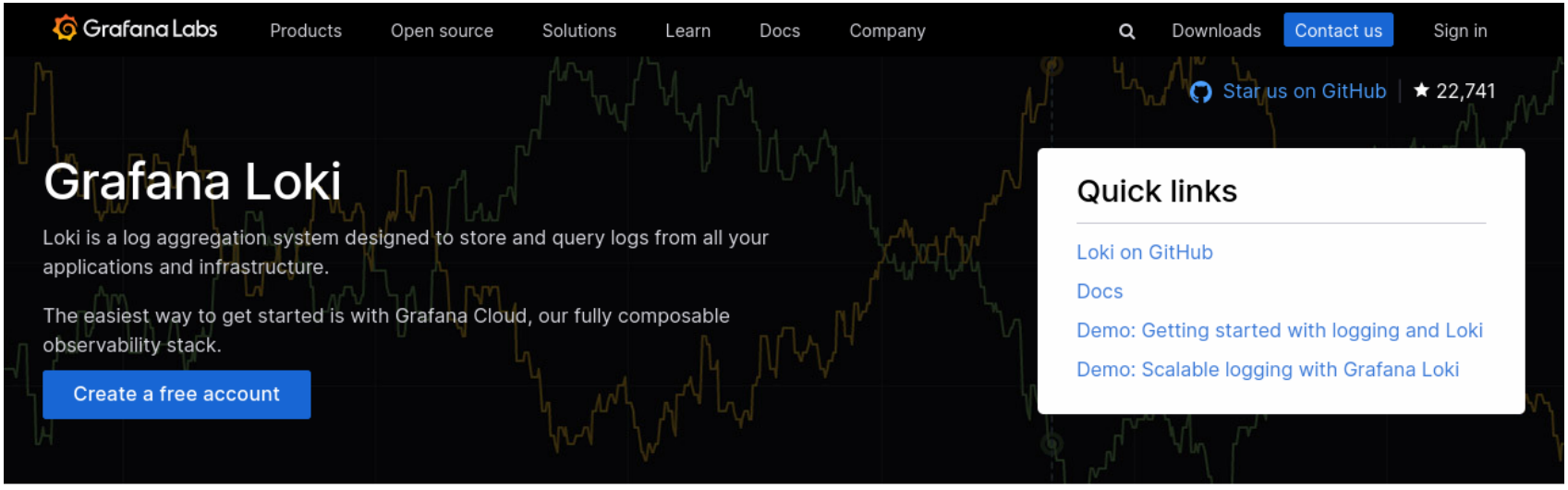
The Grafana open-source community offers Helm Charts for running it on Kubernetes. Please be aware that the code is provided without any warranties. If you encounter any problems, you can report them to the Official GitHub repository.

Watch this video to learn more about installing Grafana using Helm Charts:



Loki

- Loki is a backend store for long-term log retention
- Designed to be very cost effective and highly scalable
- AGPLv3



Grafana Loki

Loki is a horizontally scalable, highly available, multi-tenant log aggregation system inspired by Prometheus. It is designed to be very cost effective and easy to operate. It does not index the contents of the logs, but rather a set of labels for each log stream.

The Loki project was started at Grafana Labs in 2018, and announced at KubeCon Seattle. Loki is released under the AGPLv3 license.

Grafana Labs is proud to lead the development of the Loki project, building first-class support for Loki into Grafana, and ensuring Grafana Labs customers receive Loki support and features they need.

Why use Grafana Loki?

- Suport multiple installation style, e.g. monolithic, microservice, or even manually
- Here we are going to demo with distributed Helm chart style

Grafana Labs documentation

Search docs

Product

Grafana Loki

Viewing: v3.1.x (latest)

Find another version

Grafana Loki

- Release notes
- Get started
- Set up
  - Install
    - Install using Helm
      - Helm chart components
      - Install monolithic Loki
      - Install microservice Loki
      - Install scalable Loki
      - Configure storage
      - Helm chart values
      - Monitoring
    - Install using Tanka
    - Install using Docker
    - Install locally
    - Install on Istio
    - Install from source
  - Migrate
  - Upgrade

Documentation > Grafana Loki > Set up > Install > Install using Helm > Install microservice Loki

Open source

RSS

Install the microservice Helm chart

This Helm Chart deploys Grafana Loki on Kubernetes.

This chart configures Loki to run Loki in microservice / distributed mode. The microservices deployment mode runs components of Loki as distinct processes.

The default Helm chart deploys the following components:

- **Compactor component** (1 replica): Compacts and processes stored data.
- **Distributor component** (3 replicas, maxUnavailable: 2): Distributes incoming requests. Up to 2 replicas can be unavailable during updates.
- **IndexGateway component** (2 replicas, maxUnavailable: 1): Handles indexing. Up to 1 replica can be unavailable during updates.
- **Ingester component** (3 replicas): Handles ingestion of data.
- **Querier component** (3 replicas, maxUnavailable: 2): Processes queries. Up to 2 replicas can be unavailable during updates.
- **QueryFrontend component** (2 replicas, maxUnavailable: 1): Manages frontend queries. Up to 1 replica can be unavailable during updates.
- **QueryScheduler component** (2 replicas): Schedules queries.

It is not recommended to run scalable mode with `filesystem` storage. For the purpose of this guide, we will use MinIO as the object storage to provide a complete example.

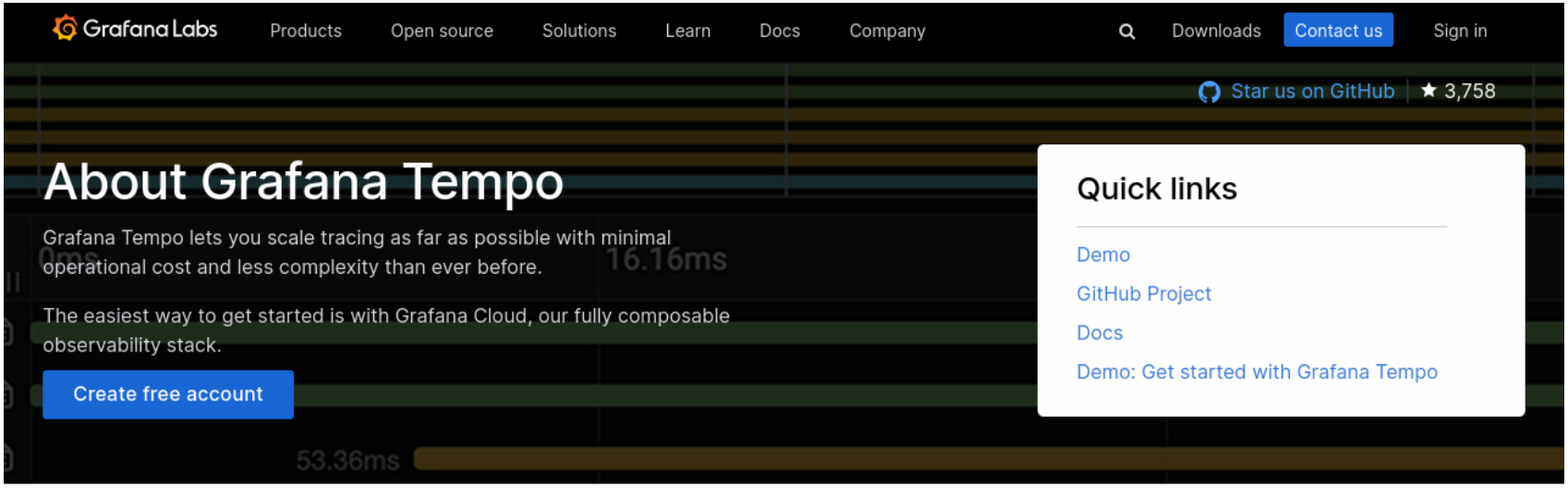
Prerequisites

- Helm 3 or above. See Installing Helm.
- A running Kubernetes cluster.
- (Optional) A Memcached deployment for better query performance. For information on configuring Memcached, refer to the caching section.

To deploy Loki in microservice mode (with MinIO):

Tempo

- Tempo is a distributed tracing backend
- Designed to be cost-efficient, requiring only object storage to operate
- Deeply integrated with Grafana, Prometheus, and Loki
- AGPLv3



Introduction to Grafana Tempo

Grafana Tempo is an open source, easy-to-use, and high-scale distributed tracing backend. Tempo is cost-efficient, requiring only object storage to operate, and is deeply integrated with Grafana, Prometheus, and Loki. Tempo can ingest common open source tracing protocols, including Jaeger, Zipkin, and OpenTelemetry.

The Tempo project was started at Grafana Labs and announced at Grafana ObservabilityCON in October 2020. It became generally available with the 1.0 release in June 2021. Tempo is released under the AGPLv3 license.

Grafana Labs is proud to lead the development of the Tempo project, building first-class support for Tempo into Grafana, and ensuring Grafana Labs customers receive Tempo support and features they need.

The Goal 🚀 Sample 100% of our read path

- Suport multiple installation style, e.g. monolithic, microservice, or even manually
- Here we are going to demo with distributed Helm chart style

Grafana Labs documentation

Search docs

Product

Grafana Tempo

Viewing: v2.5.x (latest)

Find another version

Grafana Tempo

- Release notes
- Introduction
- Get started
- Set up
  - Plan your deployment
  - Upgrade
  - Deploy with Helm
  - Deploy with operator
  - Deploy on Linux
  - Deploy on Kubernetes with Tanka
  - Set up a test application for a Tempo cluster
- Configure
- Manage
- Metrics-generator
- Query with TraceQL
- Troubleshoot
- API
- Community

Open source

Deploy Tempo with Helm

The Helm charts for Grafana Tempo and Grafana Enterprise Traces allows you to configure, install, and upgrade Grafana Tempo or Grafana Enterprise Traces within a Kubernetes cluster.

To deploy Tempo using the `tempo-distributed` Helm chart, read the Get started with Grafana Tempo using Helm.

Example Helm chart

The Tempo repository has an example Helm chart that shows a complete microservice-based deployment.

Helm charts for deployment

Tempo has two primary charts used for deployment:

- `tempo-distributed` Helm chart deploys Tempo in microservices mode (read the documentation)
- `tempo` Helm chart deploys Tempo in monolithic (single binary) mode

Mimir

- Mimir is a TSDB for long-term storage for Prometheus
- Designed to be horizontally scalable, highly available, multi-tenant
- 100% Prometheus compatible
- AGPLv3

Star us on GitHub | ★ 3,858

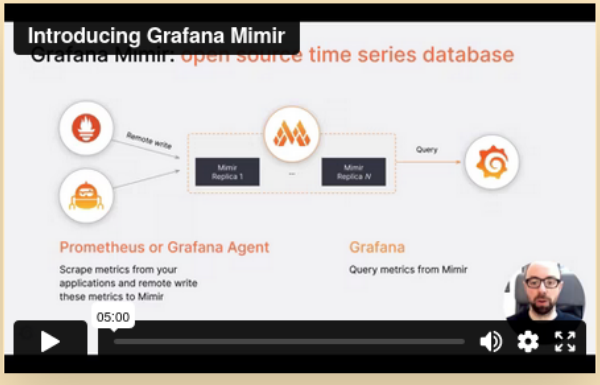
What is Grafana Mimir?

Mimir is an open source, horizontally scalable, highly available, multi-tenant TSDB for long-term storage for Prometheus.

Read the announcement blog post →

Get Mimir up and running in 10 minutes:

Tutorial | Intro to metrics webinar



Mimir overview

Mimir lets you scale metrics to 1 billion active series and beyond, with high availability, multi-tenancy, durable storage, and blazing fast query performance over long periods of time.

Mimir was started at Grafana Labs and announced in 2022. The mission for the project is to make it the most scalable, most performant open source time series database for metrics, by incorporating what Grafana Labs engineers have learned running Grafana Enterprise Metrics and Grafana Cloud Metrics at massive scale. Mimir is released under the AGPLv3 license.

Grafana Labs is proud to lead the development of the Mimir project, building first-class support for Mimir into Grafana, and ensuring Grafana Labs customers receive Mimir support and features they need.

Why use Mimir?

- Suport multiple installation style, e.g. monolithic, microservice, or even manually
- Here we are going to demo with distributed Helm chart style

Grafana Labs documentation

Search docs

Product

Grafana Mimir

Viewing: v2.12.x (latest)

Find another version

Grafana Mimir

- Release notes
- Get started
- Set up
  - Deploy with Helm
  - Deploy with Puppet
  - Deploy with Jsonnet and Tanka
  - Migrate
- Configure
- Send
- Manage
- Query
- Visualize
- References

Copyright notice

Open source

Deploy Mimir with Helm

The `mimir-distributed` Helm chart for Grafana Mimir and Grafana Enterprise Metrics allows you to configure, install, and upgrade Grafana Mimir or Grafana Enterprise Metrics within a Kubernetes cluster.

To deploy Mimir using the `mimir-distributed` Helm chart, see Get started with Grafana Mimir using the Helm chart.

Was this page helpful?

👍 Yes 👎 No

Suggest an edit Contribute to docs Report a problem

Community Support

Related documentation

Docs

Deploying the Grafana Mimir monitoring mixin

Docs

Grafana mimir-distributed Helm chart documentation

Docs

Monitor Grafana Mimir

Pyroscope

- Pyroscope is a continuous profiling database
- Designed to be fast, scalable, highly available, and efficient storage and querying
- AGPLv3

Grafana Labs Products Open source Solutions Learn Docs Company Downloads Contact us Sign in

ANNOUNCEMENT Grafana Cloud Profiles, powered by Pyroscope, is now GA Learn more →

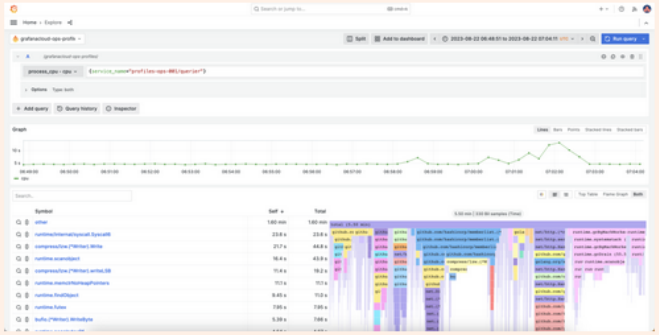
Star us on GitHub | ★ 9,643

Continuous profiling

What is Grafana Pyroscope?

Grafana Pyroscope is an open source continuous profiling database that provides fast, scalable, highly available, and efficient storage and querying. This helps you get a better understanding of resource usage in your applications down to the line number.

GitHub project Blog announcement Documentation



Grafana Pyroscope overview

Grafana Pyroscope allows you to aggregate continuous profiling data with high availability, multi-tenancy, and durable storage.

Grafana Pyroscope was formed by the merger of two open source continuous profiling projects: Phlare, which was launched by Grafana Labs in 2022, and Pyroscope, which was founded by Ryan Perry and Dmitry Filimonov in 2021. The mission for the Grafana Pyroscope project is to enable continuous profiling at scale for the open source community, giving developers a better understanding of the resource usage of their code. By doing so, Pyroscope allows users to understand their application performance and optimize their infrastructure spend.

Grafana Labs is proud to lead the development of the Grafana Pyroscope project. The project

- Suport multiple installation style, e.g. monolithic, microservice, or even manually
- Here we are going to demo with distributed Helm chart style

Grafana Labs documentation

Search docs

Product

Grafana Pyroscope

Viewing: v1.6.x (latest)

Find another version

Grafana Pyroscope

- Release notes
- Introduction
- Get started
- Configure the client to send profiles
- Upgrade
- Deploy on Kubernetes
  - Deploy with Helm
  - Deploy with Jsonnet and Tanka
- Configure the server
- View and analyze profile data
- Reference: Pyroscope Architecture
- Copyright notice

Documentation > Grafana Pyroscope > Deploy on Kubernetes > Deploy with Helm

Open source

Deploy Pyroscope using the Helm chart

The Helm chart allows you to configure, install, and upgrade Pyroscope within a Kubernetes cluster.

Before you begin

These instructions are common across any flavor of Kubernetes and assume that you know how to install, configure, and operate a Kubernetes cluster as well as use `kubectl`.

Hardware requirements:

- A single Kubernetes node with a minimum of 4 cores and 16GiB RAM

Software requirements:

- Kubernetes 1.20 or higher
- The `kubectl` command for your version of Kubernetes
- Helm 3 or higher

Verify that you have:

- Access to the Kubernetes cluster
- Persistent storage is enabled in the Kubernetes cluster, which has a default storage class set up. You can change the default StorageClass.
- DNS service works in the Kubernetes cluster

Install the Helm chart in a custom namespace

Use a custom namespace so that you don't have to overwrite the default namespace later in the procedure.

Demo with AlviStack Vagrant Box

System Requirements

- Windows 10+
- 4 Core CPU (for running a 2 Core CPU VM)
- 16GB Memory (for running a 8GB VM)
- 50GB SSD

Install VirtualBox

- Download and install VirtualBox
- Download and install Extension Pack
- Verify result
  - `vboxmanage --version`
- https://www.virtualbox.org/wiki/Downloads

About
Screenshots
Downloads
Documentation
End-user docs
Technical docs
Contribute
Community

Download VirtualBox

Here you will find links to VirtualBox binaries and its source code.

VirtualBox binaries

By downloading, you agree to the terms and conditions of the respective license.

VirtualBox 7.0.18 platform packages

- ⇨Windows hosts
- ⇨macOS / Intel hosts
- Linux distributions
- ⇨Solaris hosts
- ⇨Solaris 11 IPS hosts

The binaries are released under the terms of the GPL version 3.

See the changelog for what has changed.

You might want to compare the checksums to verify the integrity of downloaded packages. *The SHA256 checksums should be favored as the MD5 algorithm must be treated as insecure!*

- SHA256 checksums, MD5 checksums

Note: After upgrading VirtualBox it is recommended to upgrade the guest additions as well.

VirtualBox 7.0.18 Oracle VM VirtualBox Extension Pack

- ⇨All supported platforms

Support VirtualBox RDP, disk encryption, NVMe and PXE boot for Intel cards. See this chapter from the User Manual for an introduction to this Extension Pack. The Extension Pack binaries are released under the VirtualBox Personal Use and Evaluation License (PUEL). *Please install the same version extension pack as your installed version of VirtualBox.*

VirtualBox 7.0.18 Software Developer Kit (SDK)

- ⇨All platforms

User Manual

The VirtualBox User Manual is included in the VirtualBox packages above. If, however, you would like to take a look at it without having to install the whole thing, you also access it here:

Install Vagrant

- Download and install Vagrant
- Verify Result
  - `vagrant --version`
- https://developer.hashicorp.com/vagrant/install

Vagrant Home
Install Vagrant

Operating Systems
macOS
Windows
Linux

Release information

Resources
Tutorial Library
Community Forum
Support
GitHub

Install Vagrant

2.4.1 (latest)

macOS

Package manager

```
brew tap hashicorp/tap
brew install hashicorp/tap/hashicorp-vagrant
```

Binary download

AMD64
Version: 2.4.1
Download

ARM64
Version: 2.4.1
Download

Windows

Binary download

AMD64
Version: 2.4.1
Download

I686
Version: 2.4.1
Download

Linux

About Vagrant

Vagrant is the command line utility for managing the lifecycle of virtual machines.

Featured docs

Install
Command-Line Interface
Boxes
Networking
Synced Folders
Providers

Prepare Vagrantfile

- Describe the type of machine
- How to configure these machines
- How to provision these machines
- https://developer.hashicorp.com/vagrant/docs/vagrantfile

Vagrant Home
Filter sidebar
Documentation
Installation
Commands (CLI)
Vagrant Share
Vagrantfile
Overview
Configuration Version
Minimum Vagrant Version
Tips & Tricks
config.vm
config.ssh
config.winrm
config.winssh
config.vagrant
Boxes
Provisioning
Networking
Synced Folders
Cloud-Init
Disks

v2.4.1 (latest)

Vagrantfile

The primary function of the Vagrantfile is to describe the type of machine required for a project, and how to configure and provision these machines. Vagrantfiles are called Vagrantfiles because the actual literal filename for the file is `Vagrantfile` (casing does not matter unless your file system is running in a strict case sensitive mode).

Vagrant is meant to run with one Vagrantfile per project, and the Vagrantfile is supposed to be committed to version control. This allows other developers involved in the project to check out the code, run `vagrant up`, and be on their way. Vagrantfiles are portable across every platform Vagrant supports.

The syntax of Vagrantfiles is Ruby, but knowledge of the Ruby programming language is not necessary to make modifications to the Vagrantfile, since it is mostly simple variable assignment. In fact, Ruby is not even the most popular community Vagrant is used within, which should help show you that despite not having Ruby knowledge, people are very successful with Vagrant.

Lookup Path

When you run any `vagrant` command, Vagrant climbs up the directory tree looking for the first Vagrantfile it can find, starting first in the current directory. So if you run `vagrant` in `/home/mitchellh/projects/foo`, it will search the following paths in order for a Vagrantfile, until it finds one:

`vagrant up`

- `vagrant up --provider virtualbox`
- Running VM with VirtualBox
- Configure CPU/RAM/Disk
- Configure network port mapping
- Mount `$PWD` into VM as `/vagrant`
- Provision after VM is up and running
- https://developer.hashicorp.com/vagrant/docs/cli/up

Vagrant Home
Filter sidebar
Documentation
Installation
Commands (CLI)
Overview
box
cloud
connect
destroy
global-status
halt
init
login
package
plugin
port
powershell
provision
rdp
reload
resume

v2.4.1 (latest)

Up

Command: `vagrant up [name|id]`

This command creates and configures guest machines according to your Vagrantfile.

This is the single most important command in Vagrant, since it is how any Vagrant machine is created.

Options

- `name` - Name of machine defined in Vagrantfile. Using `name` to specify the Vagrant machine to act on must be done from within a Vagrant project (directory where the Vagrantfile exists).
- `id` - Machine id found with `vagrant global-status`. Using `id` allows you to call `vagrant up id` from any directory.
- `--[no-]destroy-on-error` - Destroy the newly created machine if a fatal, unexpected error occurs. This will only happen on the first `vagrant up`. By default this is set.
- `--[no-]install-provider` - If the requested provider is not installed, Vagrant will attempt to automatically install it if it can. By default this is enabled.
- `--[no-]parallel` - Bring multiple machines up in parallel if the provider supports it. Please consult the provider documentation to see if

`vagrant provision`

- `vagrant provision`
- Sometime you hope to R&D and debug the provision session
- After VM up and running you could re-run the provision steps
- https://developer.hashicorp.com/vagrant/docs/cli/provision

Vagrant Home
Filter sidebar
Documentation
Installation
Commands (CLI)
Overview
box
cloud
connect
destroy
global-status
halt
init
login
package
plugin
port
powershell
provision
rdp
reload
resume

v2.4.1 (latest)

Provision

Command: `vagrant provision [vm-name]`

Runs any configured provisioners against the running Vagrant managed machine.

This command is a great way to quickly test any provisioners, and is especially useful for incremental development of shell scripts, Chef cookbooks, or Puppet modules. You can just make simple modifications to the provisioning scripts on your machine, run a `vagrant provision`, and check for the desired results. Rinse and repeat.

Options

- `--provision-with x,y,z` - This will only run the given provisioners. For example, if you have a `:shell` and `:chef_solo` provisioner and run `vagrant provision --provision-with shell`, only the shell provisioner will be run.

Edit this page on GitHub

`vagrant ssh`

- `vagrant ssh`
- SSH into VM with user `vagrant`
- Could switch as `root` with `sudo su -`
- Host `$PWD` already mount into VM as `/vagrant`
- https://developer.hashicorp.com/vagrant/docs/cli/ssh

Vagrant Home
Filter sidebar
Documentation
Installation
Commands (CLI)
Overview
box
cloud
connect
destroy
global-status
halt
init
login
package
plugin
port
powershell
provision
rdp
reload
resume

v2.4.1 (latest)

SSH

Command: `vagrant ssh [name|id] [-- extra_ssh_args]`

This will SSH into a running Vagrant machine and give you access to a shell.

On a simple vagrant project, the instance created will be named default.

Vagrant will ssh into this instance without the instance name:

```
$ vagrant ssh

Welcome to your Vagrant-built virtual machine.
Last login: Fri Sep 14 06:23:18 2012 from 10.0.2.2
$ logout
Connection to 127.0.0.1 closed.
```

Or you could use the name:

```
$ vagrant ssh default


Welcome to your Vagrant-built virtual machine.
Last login: Fri Jul 20 15:09:52 2018 from 10.0.2.2
$ logout
Connection to 127.0.0.1 closed.
$
```

On multi-machine setups, you can login to each VM using the name as displayed on `vagrant status`

kubectl get node

```
root@kubernetes-1:~# kubectl get node
NAME           STATUS   ROLES           AGE     VERSION
kubernetes-1   Ready    control-plane   9m18s   v1.29.2
```

kubectl get pod

```
root@kubernetes-1:~# kubectl get pod --all-namespaces
NAMESPACE      NAME                                    READY
csi-hostpath   csi-hostpath-socat-0                    1/1
csi-hostpath   csi-hostpathplugin-0                    8/8
kube-system    coredns-76f75df574-2tdbp                1/1
kube-system    coredns-76f75df574-g687d                1/1
kube-system    kube-addon-manager-kubernetes-1         1/1
kube-system    kube-apiserver-kubernetes-1             1/1
kube-system    kube-controller-manager-kubernetes-1    1/1
kube-system    kube-flannel-ds-kmq79                   1/1
kube-system    kube-proxy-vd4qj                        1/1
kube-system    kube-scheduler-kubernetes-1             1/1
kube-system    snapshot-controller-7b6f9cf9b4-rj5v5    1/1
```

Verify with Browser

- Check the result (from host, with VirtualBox port mapping enabled in Vagrantfile):
  - http://localhost:8080

Create a Grafana Dashboard

Connect with Data Sources

Q&A

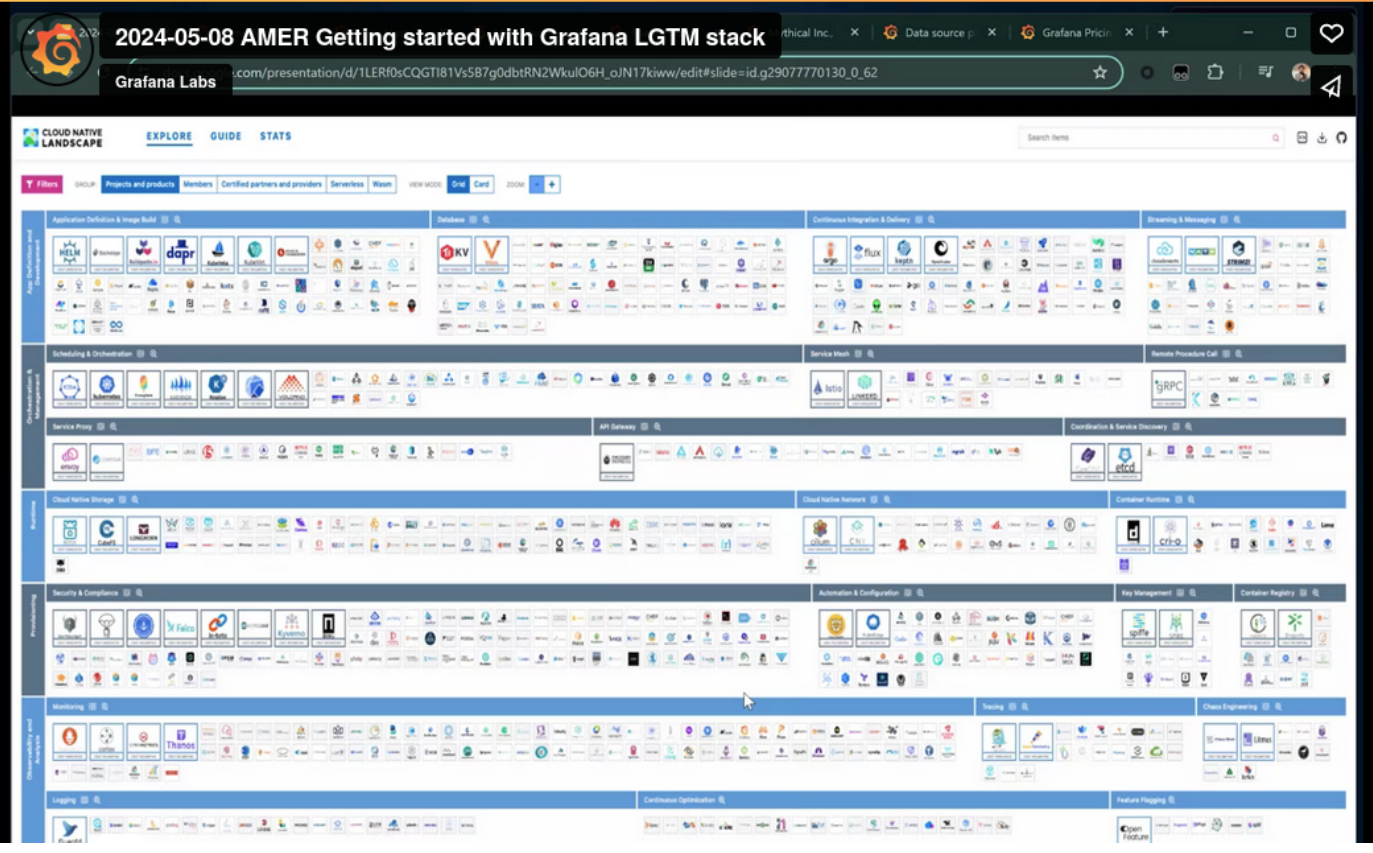
References

- Getting started with managing your metrics, logs, and traces using Grafana
- Official webinar from Grafana Labs
- For beginner with simple show case
- Around 60 mintues

English

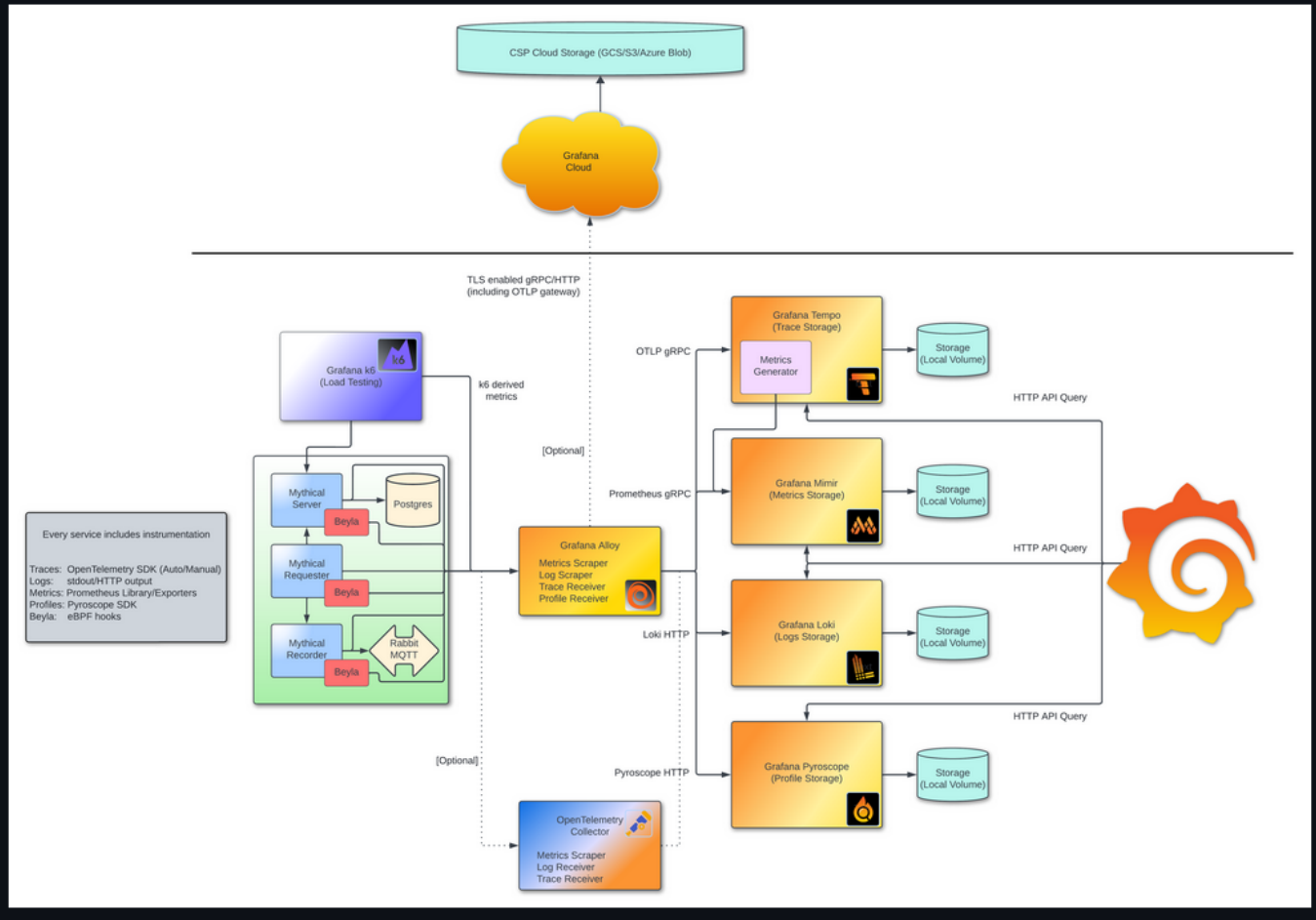
Getting started with managing your metrics, logs, and traces using Grafana

Share this:



- Introduction to Metrics, Logs, Traces and Profiles in Grafana
- Official complete demo for Grafana LGTM stack
- For advance user with Docker + Docker Compose experience
- Around 2 hours

Introduction to Metrics, Logs, Traces and Profiles in Grafana



This readme has the following sections:

- 可觀測性宇宙的第一天 - Grafana LGTM 全家桶的起點
- Detail explanation for Grafana LGTM + Kubernetes
- For advance user with daily Kubernetes experience
- AT LEAST 6 HOURS (reading doc only…)

可觀測性宇宙的第一天 - Grafana LGTM 全家桶的起點

15th鐵人賽 kubernetes grafana observability 可觀測性

mikehsu0618
團隊 所以隊名要叫什麼
2023-09-16 00:36:13
6984 瀏覽

分享至

前言

這是一個從異世界歸來後，一個不小心踏入 Kubernetes 可觀測性宇宙的故事，一切都要從一個非本科小白在愚昧山丘的頂端遇見 Grafana 開始說起。

- AlviStack Vagrant Box Packaging for Kubernetes
- CNCF Certified Kubernetes on Virtual Box or Libvrt
- Simply `vagrant up` and ready for use
- May customize for Dev/CI/CD
- Around 10 minutes

alvistack / kubernetes-1.30 Vagrant box

How to use this box with Vagrant:

Vagrantfile New

```
Vagrant.configure("2") do |config|
  config.vm.box = "alvistack/kubernetes-1.30"
end
```

v20240629.1.1 currently released version

This version was created 6 days ago.

There isn't a description.

2 providers for this version.

libvirt		
unknown *	Hosted by Vagrant Cloud (1.75 GB)	

virtualbox		
unknown *	Hosted by Vagrant Cloud (1.68 GB)	

Contact Me

- Address: Unit 326, 3/F, Building 16W, No.16 Science Park West Avenue, Hong Kong Science Park, Shatin, N.T.
- Phone: +852 3576 3812
- Fax: +852 3753 3663
- Email: sales@pantarei-design.com
- Web: http://pantarei-design.com