



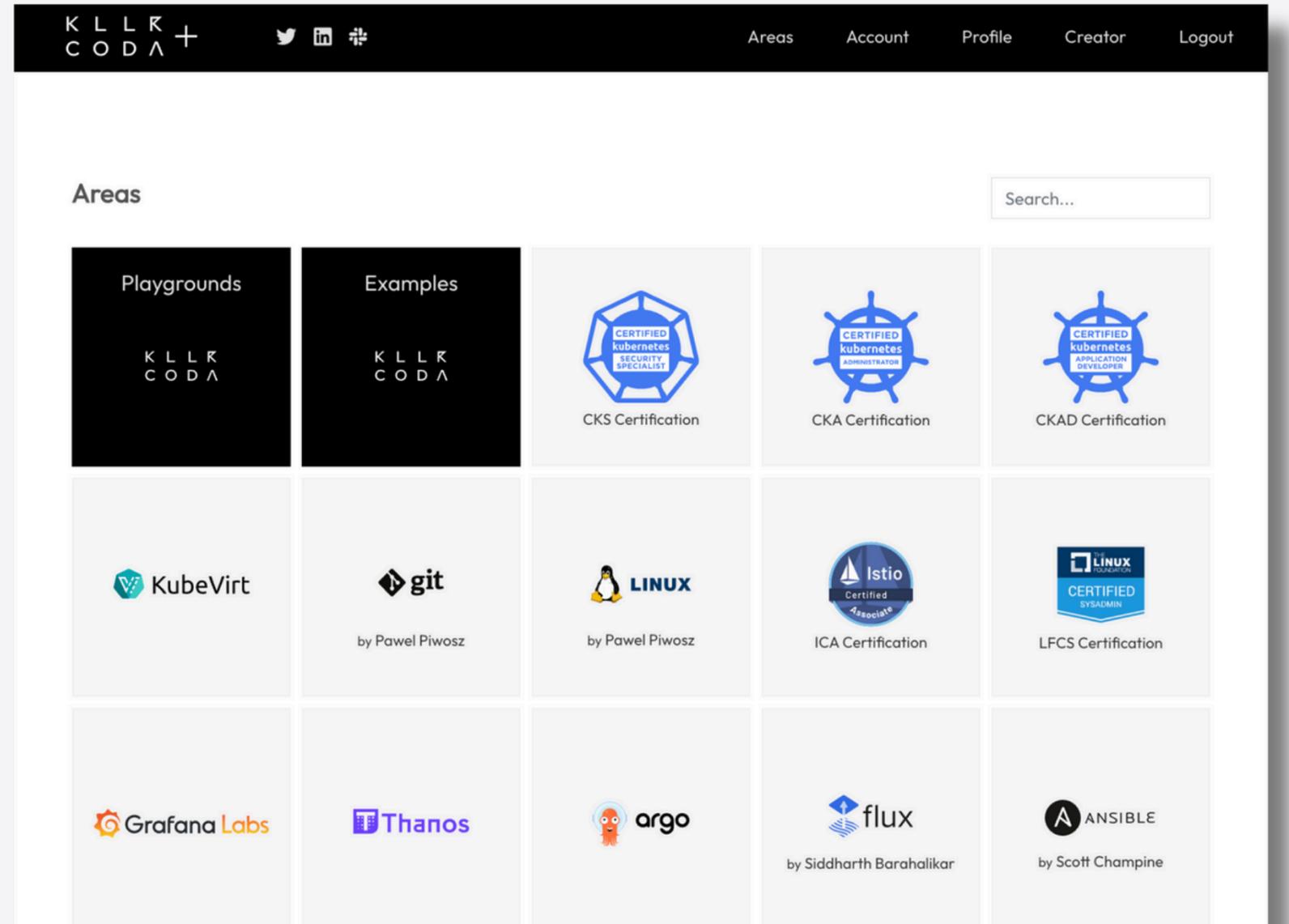
9,800 Sandboxes and Counting...

Transforming Documentation with Interactive Learning Environments



Jay Clifford

Sr. Developer Advocate | Grafana Labs



A bit about me...



Work at Grafana:

-  Create education content for Grafana Loki
-  Create documentation
-  Run our Interactive Sandbox initiative

Life:

-  Big gamer
-  Lego enthusiast
-  IoT enthusiast



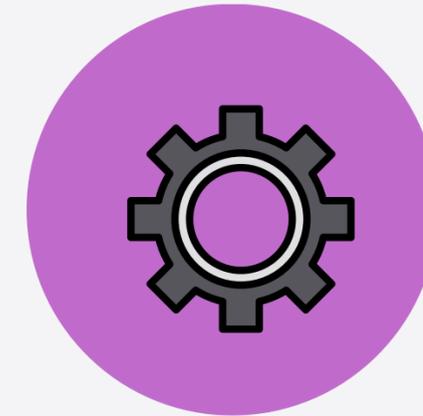
What are we going to cover



The goal of Dev Docs



What are sandboxes?



How we scaled to 9600

Goal



Make your own



The Goal of Dev docs?

Key Source of Truth

for product information

Enablement

grow from beginners to experts



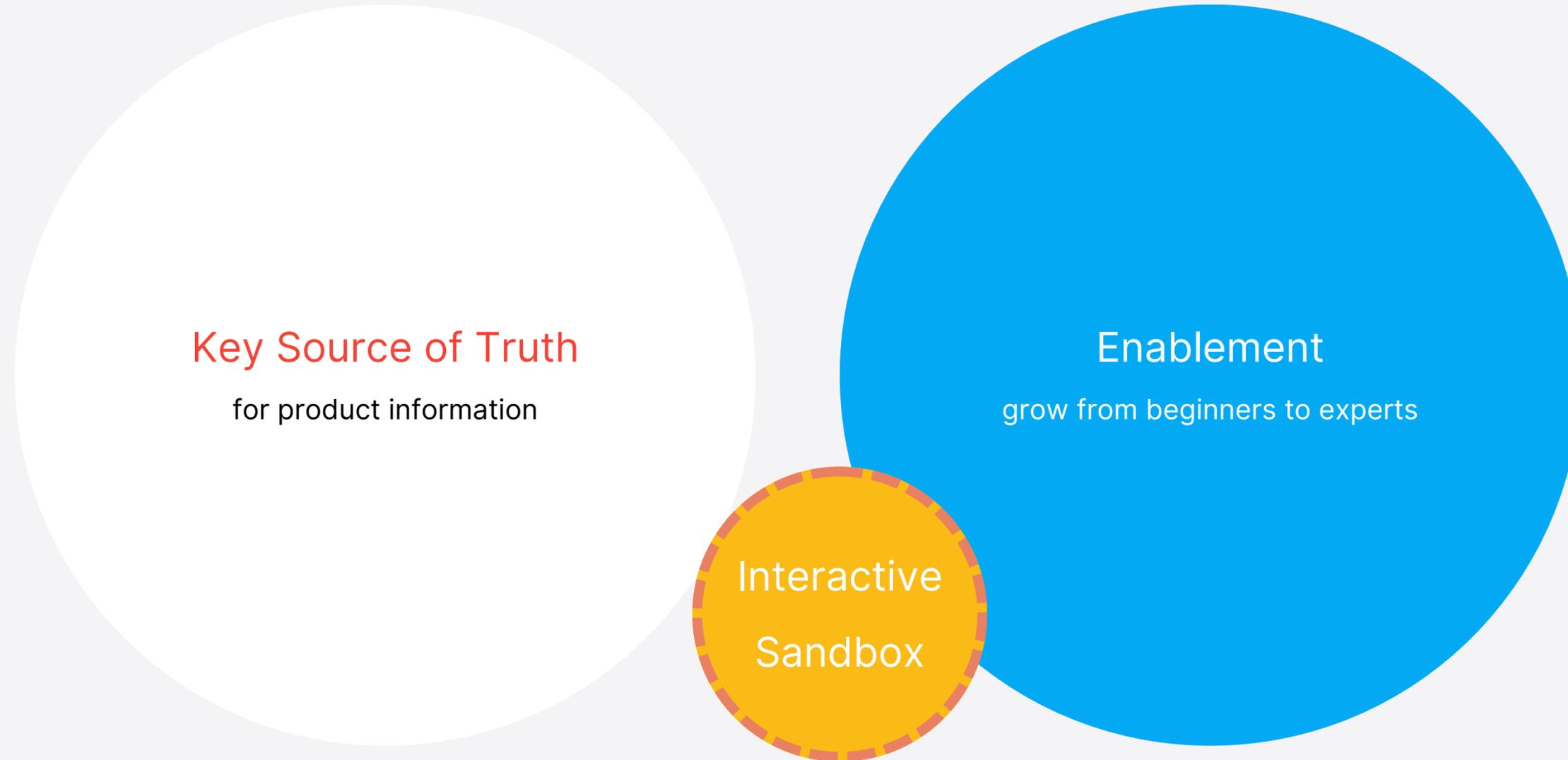
The Goal of Dev docs?

Key Source of Truth
for product information

Enablement
grow from beginners to experts



The Goal of Dev docs?



What is a sandbox?



A disposable VM hosted online



Focused on building practical tutorials



Allows the installation and execution of code and applications



Why sandboxes?



Engage with the documentation interactively, allowing for a more practical and immersive learning experience.



Experiment and practice in a risk-free environment, enhancing their understanding and confidence in using our products.



Accelerate their learning curve, transitioning from beginners to proficient users more effectively.



KLLRCODA +

Kubernetes Monitoring with Loki

Step 2: Add the Grafana Helm repository

All three helm charts (Loki, Grafana, and the Kubernetes Monitoring Helm) are available in the Grafana Helm repository. Add the Grafana Helm repository by running the following command:

```
helm repo add grafana https://grafana.github.io/helm-charts
```

It's recommended to also run `helm repo update` to ensure you have the latest version of the charts.

Step 3: Clone the tutorial repository

Clone the tutorial repository by running the following command:

```
git clone https://github.com/grafana/alloy-scenarios.git &&
```

As well as cloning the repository, we have also changed directories to `alloy-scenarios/k8s-logs`. The rest of this tutorial assumes you are in this directory.

controlplane \$ kubectl create namespace meta && kubectl create namespace prod
namespace/meta created
namespace/prod created
controlplane \$ helm repo add grafana https://grafana.github.io/helm-charts && helm repo update
"grafana" has been added to your repositories
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "grafana" chart repository
Update Complete. *Happy Helming!*
controlplane \$ git clone https://github.com/grafana/alloy-scenarios.git && cd alloy-scenarios/k8s-logs
Cloning into 'alloy-scenarios'...
remote: Enumerating objects: 105, done.
remote: Counting objects: 100% (105/105), done.
remote: Compressing objects: 100% (74/74), done.
remote: Total 105 (delta 51, reused 76 (delta 26), pack-reused 0 (from 0))
Receiving objects: 100% (105/105), 3.01 MiB | 14.49 MiB/s, done.
Resolving deltas: 100% (51/51), done.
controlplane \$

BACK NEXT



Free to the user

Creation and participation only require signing up

Build with markdown

Courses are built with markdown and JSON

Flexible environments

Ubuntu 20.04 with Docker and Podman

Kubeadm cluster with one controlplane, taint removed, ready to schedule workload

Engagement Tracking

Check course utilisation over time with inbuilt trackers



How do we use it?

The screenshot shows the Grafana Labs documentation website. The navigation bar includes links for Products, Open Source, Solutions, Learn, Docs, Pricing, Downloads, Contact us, and My Account. The left sidebar shows a product catalog with 'Loki' selected, and a list of sub-articles under 'Send data' > 'Grafana Alloy', with 'Sending OpenTelemetry logs to Loki using Alloy' highlighted. The main content area features the article title 'Sending OpenTelemetry logs to Loki using Alloy' and a breadcrumb trail: Documentation > Grafana Loki > Send data > Grafana Alloy > Sending OpenTelemetry logs to Loki using Alloy. The article text explains that Alloy natively supports receiving logs in the OpenTelemetry format and lists three components: OpenTelemetry Receiver, OpenTelemetry Processor, and OpenTelemetry Exporter. A 'Dependencies' section lists Docker and Docker Compose. A 'TIP' box suggests an interactive learning environment. A 'Grafana Cloud' banner offers a free tier with 10k metrics, 50GB logs, 50GB traces, 3 active users, and 14-day retention. A 'Feedback' section asks if the page is helpful with 'Yes' and 'No' buttons. A 'On this page' section lists 'Dependencies' and 'Scenario' with sub-steps for environment setup and configuration.

Documentation > Grafana Loki > Send data > Grafana Alloy > Sending OpenTelemetry logs to Loki using Alloy

Open source

Sending OpenTelemetry logs to Loki using Alloy

Alloy natively supports receiving logs in the OpenTelemetry format. This allows you to send logs from applications instrumented with OpenTelemetry to Alloy, which can then be sent to Loki for storage and visualization in Grafana. In this example, we will make use of 3 Alloy components to achieve this:

- **OpenTelemetry Receiver:** This component will receive logs in the OpenTelemetry format via HTTP and gRPC.
- **OpenTelemetry Processor:** This component will accept telemetry data from other `otelcol.*` components and place them into batches. Batching improves the compression of data and reduces the number of outgoing network requests required to transmit data.
- **OpenTelemetry Exporter:** This component will accept telemetry data from other `otelcol.*` components and write them over the network using the OTLP HTTP protocol. We will use this exporter to send the logs to the Loki native OTLP endpoint.

Dependencies

Before you begin, ensure you have the following to run the demo:

- Docker
- Docker Compose

TIP
Alternatively, you can try out this example in our interactive learning environment: [Sending OpenTelemetry logs to Loki using Alloy](#).

It's a fully configured environment with all the dependencies already installed.

LOKI TUTORIALS
Grafana Labs

Is this page helpful?
Yes No

On this page

Dependencies

Scenario

Step 1: Environment setup

Step 2: Configure Alloy to ingest OpenTelemetry logs

Open your code editor and locate the `config.alloy` file

Receive OpenTelemetry logs via gRPC and HTTP

Create batches of logs using a

Grafana Cloud

The fastest way to get started is with the Grafana Cloud free tier which includes:

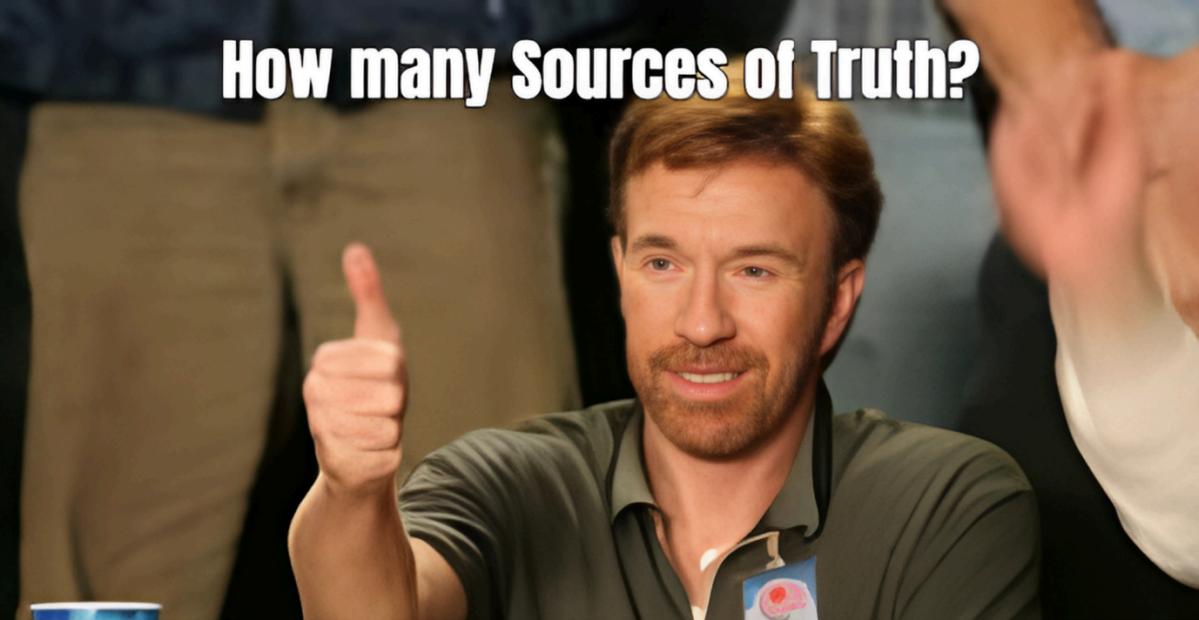
- 10k metrics
- 50GB logs
- 50GB traces
- 3 active users
- 14-day retention

Create free account

- Linked directly in our docs
- Used for step by step tutorials



How many Sources of Truth?



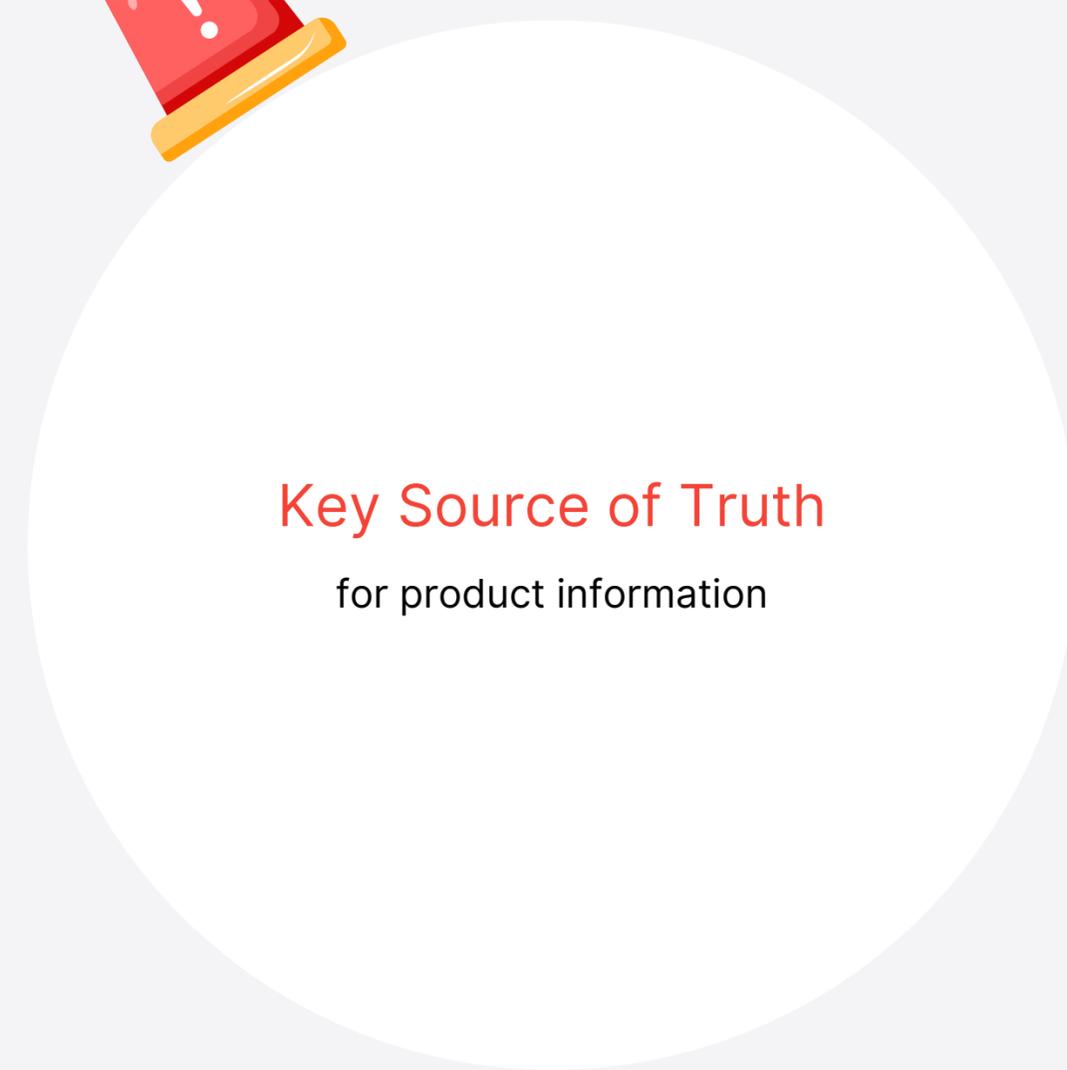
**I WANT THE SINGLE
SOURCE OF TRUTH!**



YOU CAN'T HANDLE THE TRUTH!



Key Source of Truth
for product information





I am going to create
a **source of truth** that is so **single**



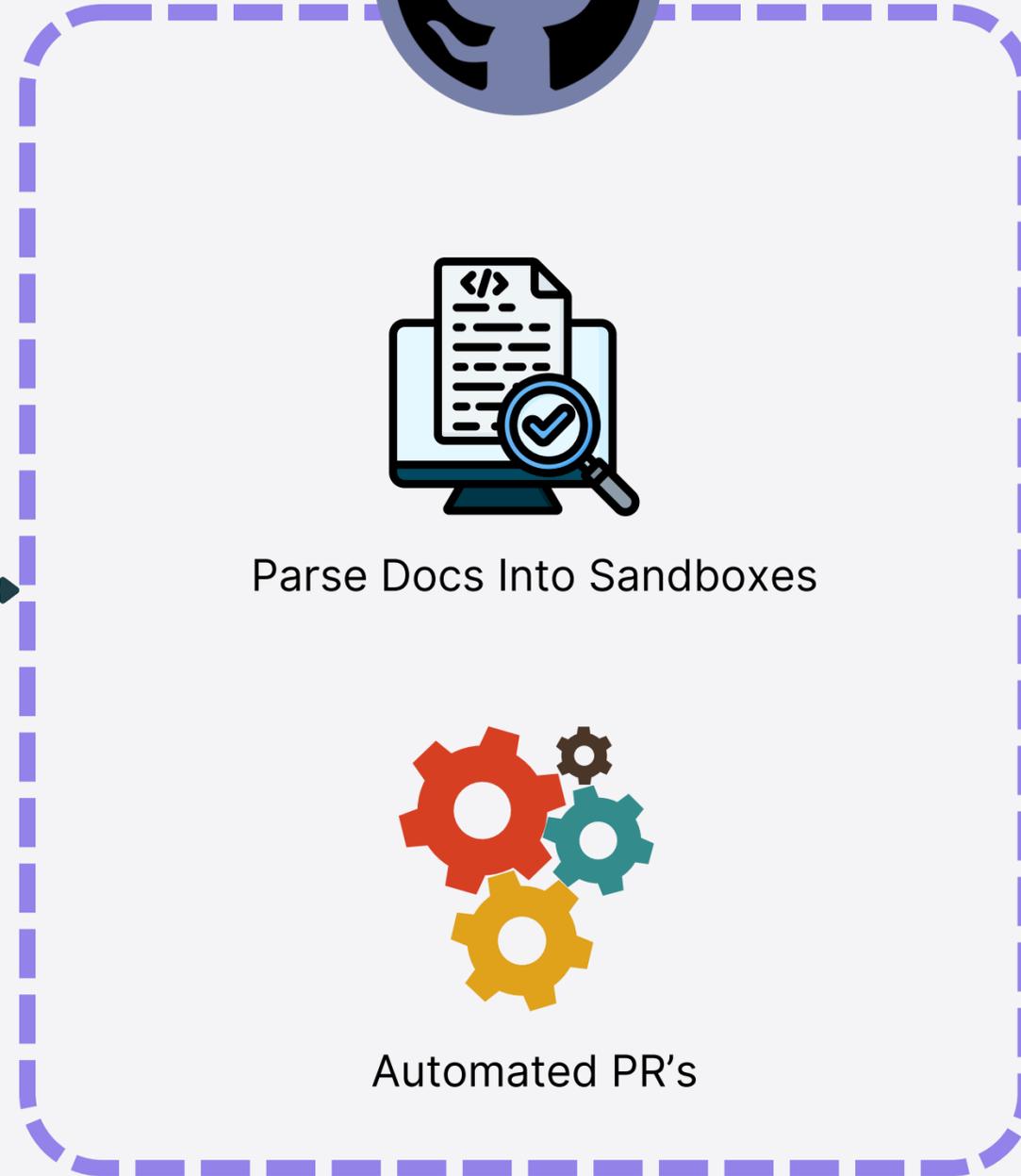
The Transformer



Parse Docs Into Sandboxes



Automated PR's



Sandbox Transformer



A look under the covers

1



Packages

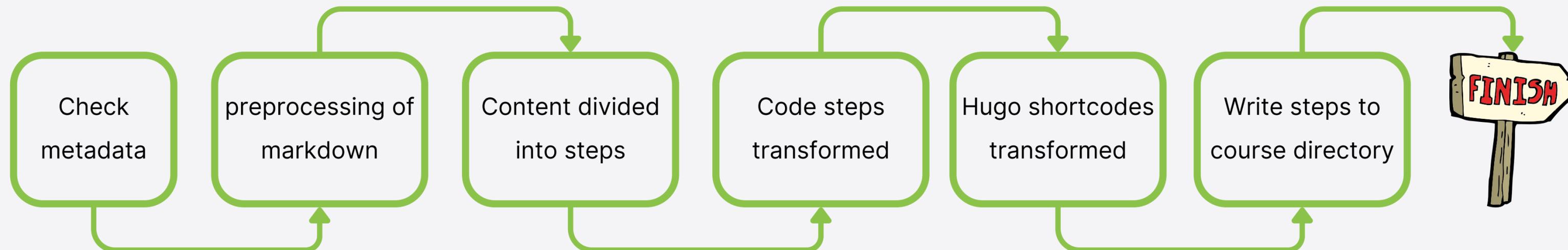
- Goldmark by Yuin
- go-diff by Sergi
- go-spew by dajohi



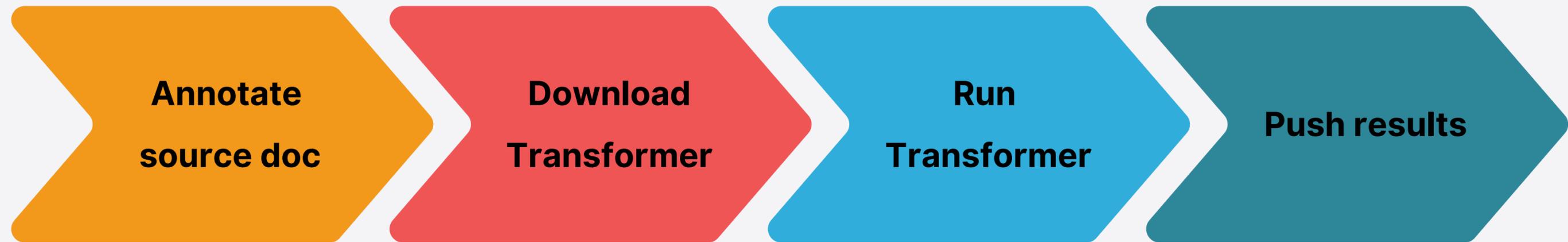
2

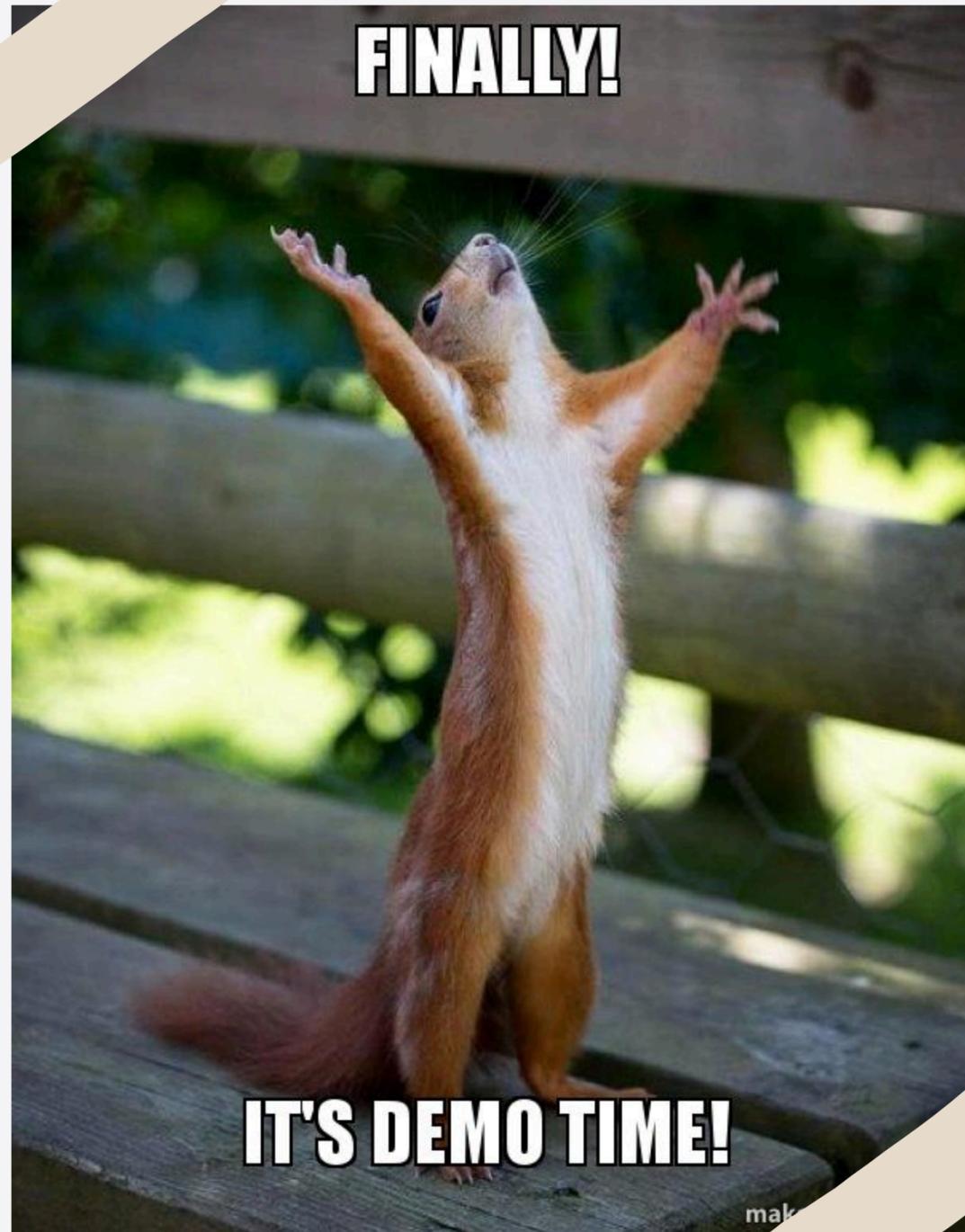
- Designed to work with Hugo annotated markdown docs
- But also works with standard markdown with the exception of requiring metadata

3



Flow to creating a new sandbox





Grafana Labs

The Open Observability Stack

K L L R
C O D A

Grafana Labs

At Grafana Labs, we believe in Open Source and Open Standards. Most of our code is open and free for you to use, and you can find it here. From homelabs to the largest organizations on Earth, everyone is using our...

 killercoda.com



Time is limited



Lets get you up and running!



Transformer Repo



<https://github.com/grafana/killercoda>

grafana/killercoda



This repository holds tutorials designed and hosted on Killercoda and interactive education platform.

7 Contributors

9 Issues

15 Stars

9 Forks



grafana/killercoda: This repository holds tutorials designed and hosted on Killercoda and interactive...

This repository holds tutorials designed and hosted on Killercoda and interactive education platform. - grafana/killercoda

GitHub



How to run the transformer



K L L K C O D A +

Learn how to use the Sandbox Transformer

Prerequisites

In this section we will cover the prerequisites you need to have in place in order to download and run the Sandbox Transformer.

Download the transformer

The Sandbox Transformer is written in Go and is distributed as a binary. You may also build the transformer from source if you prefer.

1. Download the Transformer binary from the [releases page](#):

```
wget https://github.com/grafana/killercoda/releases/download/v0.1.5/transformer-linux-amd64 -O transformer
```
2. Make the binary executable:

```
chmod +x transformer
```

Clone the repository

You will also need to clone the repository to your local machine. You can do this by running the following command:

```
git clone https://github.com/grafana/killercoda.git && cd killercoda
```

Its best practise to create a new branch for each new course you create. You can do this by running the following command:

```
git checkout -b my-new-course
```

Editor Tab 1 + 120 min

```
ubuntu $ wget https://github.com/grafana/killercoda/releases/download/v0.1.5/transformer-linux-amd64 -O transformer
--2025-01-22 11:55:13-- https://github.com/grafana/killercoda/releases/download/v0.1.5/transformer-linux-amd64
Resolving github.com (github.com)... 20.26.156.215
Connecting to github.com (github.com)[20.26.156.215]:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/768676934/62e08011-5d57-4818-9be7-39a27a0ee55b?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=releaseassetproduction%2F20250122%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20250122T115513Z&X-Amz-Expires=300&X-Amz-Signature=bd27d6cf340d4d07c62c83201161b6ee50de75ecb5a4429b15b5d6d6156371d9&X-Amz-SignedHeaders=host&response-content-disposition=attachment%3B%20filename%3Dtransformer-linux-amd64&response-content-type=application%2Foctet-stream [following]
--2025-01-22 11:55:14-- https://objects.githubusercontent.com/github-production-release-asset-2e65be/768676934/62e08011-5d57-4818-9be7-39a27a0ee55b?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=releaseassetproduction%2F20250122%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20250122T115513Z&X-Amz-Expires=300&X-Amz-Signature=bd27d6cf340d4d07c62c83201161b6ee50de75ecb5a4429b15b5d6d6156371d9&X-Amz-SignedHeaders=host&response-content-disposition=attachment%3B%20filename%3Dtransformer-linux-amd64&response-content-type=application%2Foctet-stream
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.108.133, 185.199.111.133, 185.199.110.133, ...
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)[185.199.108.133]:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 6321335 (6.0M) [application/octet-stream]
Saving to: 'transformer'

transformer 100%[=====] 6.03M 30.1MB/s in 0.2s

2025-01-22 11:55:14 (30.1 MB/s) - 'transformer' saved [6321335/6321335]

ubuntu $ chmod +x transformer
ubuntu $ git clone https://github.com/grafana/killercoda.git && cd killercoda
Cloning into 'killercoda'...
remote: Enumerating objects: 3585, done.
remote: Counting objects: 100% (296/296), done.
remote: Compressing objects: 100% (183/183), done.
remote: Total 3585 (delta 143), reused 113 (delta 113), pack-reused 3289 (from 2)
Receiving objects: 100% (3585/3585), 3.44 MiB | 15.93 MiB/s, done.
Resolving deltas: 100% (2149/2149), done.
ubuntu $
```

<https://killercoda.com/grafana-labs/course/sandbox-developer/sandbox-transformer-walk-through>



Example in our docs



<https://grafana.com/docs/loki/latest/send-data/alloy/examples/alloy-otel-logs/>



Sending OpenTelemetry logs to Loki using Alloy

Alloy natively supports receiving logs in the OpenTelemetry format. This allows you to send logs from applications instrumented with OpenTelemetry to Alloy, which can then be sent to Loki for storage and visualization in Grafana. In this example, we will make use of 3 Alloy components to achieve this:

- **OpenTelemetry Receiver:** This component will receive logs in the OpenTelemetry format via HTTP and gRPC.
- **OpenTelemetry Processor:** This component will accept telemetry data from other `otelcol.*` components and place them into batches. Batching improves the compression of data and reduces the number of outgoing network requests required to transmit data.
- **OpenTelemetry Exporter:** This component will accept telemetry data from other `otelcol.*` components and write them over the network using the OTLP HTTP protocol. We will use this exporter to send the logs to the Loki native OTLP endpoint.

Dependencies

Before you begin, ensure you have the following to run the demo:

- Docker
- Docker Compose

TIP

Alternatively, you can try out this example in our interactive learning environment: [Sending OpenTelemetry logs to Loki using Alloy](#).

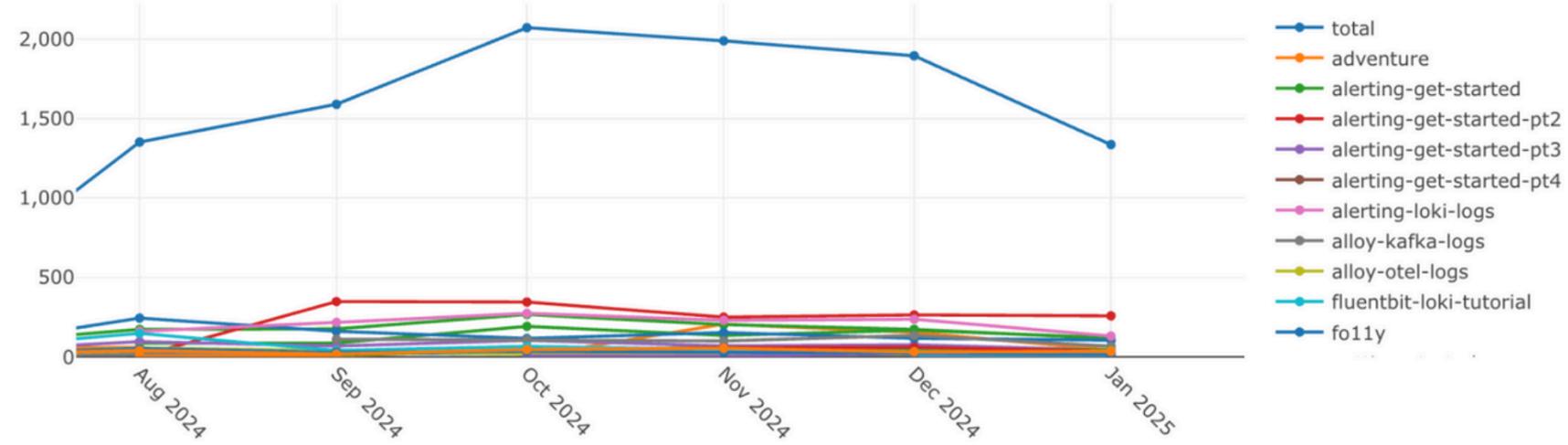
It's a fully configured environment with all the dependencies already installed.



Provide feedback, report bugs, and raise issues in the [Grafana Killercoda repository](#).

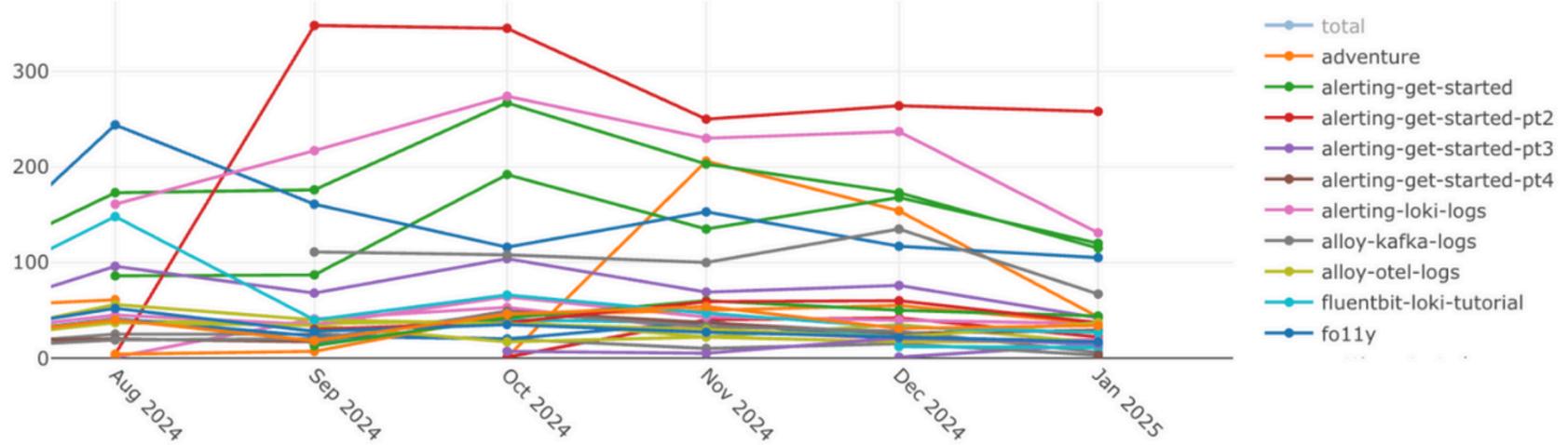
How its going?

Monthly Scenario Starts | Past 6 months



- Numbers in graphs are excluding your own account
- Times in graphs are in UTC timezone
- Records older than 6 months aren't available
- Single click on legend to exclude entries
- Double click on legend for focus mode
- Use mouse to zoom into areas

Monthly Scenario Starts | Past 6 months



Thank you for attending

Any questions?

