Member-only story

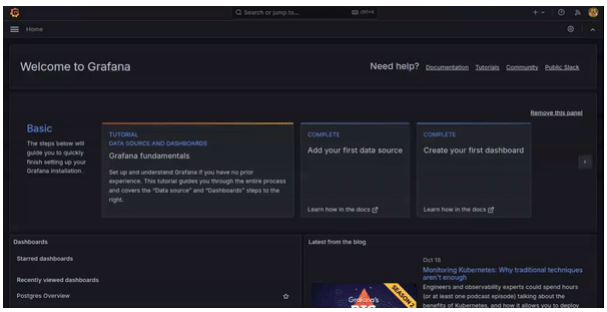
Rails: Monitor Your App Logs and Metrics Using Grafana, Loki, and Promtail

Rails to Rescue · Follow
Published in Level Up Coding
7 min read · 1 day ago

Why Do You Need Monitoring?

First off, why should you care about monitoring? As your app scales, it's not just about writing good code anymore. You need visibility into how your app is performing in real-time. Whether it's tracking response times, monitoring error rates, or keeping an eye on server load, having proper monitoring tools in place ensures:

- **Early Detection of Issues**: Find and fix problems before they affect your users.
- **Insightful Performance Metrics**: Know how your app is performing and whether there are any bottlenecks.
- **Improved Debugging**: Logs provide valuable context when things go wrong, helping you get to the root of the problem faster.



Grafana: The Visualisation Layer

Grafana is the front-end tool in this stack. It allows you to visualise your app's data through beautiful, customisation dashboards. Think of it as the control room where you monitor everything happening in your app.

Why Grafana?

- **Real-Time Monitoring**: You can visualise key performance metrics such as request times, server load, and error rates as they happen.
- **Custom Dashboards**: Create tailored dashboards that show you exactly the metrics you care about. Whether you want to monitor database performance or API response times, Grafana can pull data from various sources and display it in one place.
- **Alerting**: With Grafana, you can set up alerts for specific thresholds — like when response times get too high or error rates spike — so you can take action immediately.

Loki: Lightweight Log Aggregation

Next, we have Loki, which is a log aggregation system. Logs are invaluable for tracking down issues, but as your app scales, managing logs can become expensive and complicated. That's where Loki shines.

Why use Loki?

- **Cost-effective Logging**: Traditional logging systems like Elasticsearch require indexing, which can be expensive and resource-heavy. Loki, on the other hand, only indexes labels (metadata) rather than the full log contents, making it far more affordable.
- **Seamless Integration with Grafana**: Loki integrates directly into Grafana, meaning you can view and search logs right alongside your metrics.
- **Label-based Queries**: Instead of searching through massive logs, you can use labels to filter out logs based on things like environment (production or development), service names, or error types.

Promtail: Collecting Logs

Promtail is the agent that collects logs from your Rails application and sends them to Loki. Think of it as the bridge between your application and Loki, ensuring that all the relevant logs are collected and labeled properly.

Why Promtail?

- **Automated Log Discovery**: Promtail automatically detects log files from your Rails app (or containers) and monitors them in real-time.
- **Log Enrichment**: Promtail doesn't just collect logs — it enriches them with metadata like application name or environment, which makes it easier to filter and analyse them later.
- **Works with Rails**: Promtail can be easily configured to work with your Rails app's log formats, ensuring seamless collection and forwarding of logs to Loki.

All Together

So why use **Grafana**, **Loki**, and **Promtail** together? Because they create a holistic monitoring solution. With Grafana, you can visualise both performance metrics and logs in one place. Loki ensures that your log management is cost-effective, while Promtail takes care of log collection and forwarding.

Let's setup Log monitoring

In any production-ready Ruby on Rails application, keeping track of logs, system performance, and app metrics is crucial. This helps you detect performance bottlenecks, track errors, and make sure everything is running smoothly. One powerful stack for monitoring and observability is Grafana with **Loki** for log aggregation and **Promptail** for log streaming.

Steps to Follow To setup Log Monitoring

1) Setup Grafana

2) Setup Loki

3) Setup Promtail

1. Setting Up Grafana for Monitoring

Grafana is a open-source visualisation and monitoring platform. To get started, let's manually install and configure Grafana on your system.

Installing Grafana

On Ubuntu-based systems, you can install Grafana via the APT package manager. Here's

Step 1: Add the Grafana GPG Key

First, you need to add the Grafana GPG key to ensure the packages are trusted:

```
sudo apt-get install -y software-properties-common
sudo wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add -
```

Step 2: Add the Grafana APT Repository

Next, add the Grafana APT repository to your system:

```
sudo add-apt-repository "deb https://packages.grafana.com/oss/release/deb stable main"
```

Step 3: Update Package Lists

After adding the repository, update your package lists:

```
sudo apt-get update
```

Step 4: Install Grafana

Now you should be able to install Grafana:

```
sudo apt-get install grafana
```

Step 5: Start and Enable Grafana

Once installed, you can start and enable the Grafana service:

```
sudo systemctl start grafana-server
sudo systemctl enable grafana-server
```

Once installed, you can access Grafana at `http://localhost:3000`. The default login is **admin/admin** (you'll be prompted to change this on your first login).

User name: admin
Password: admin

2. Installing Loki for Log Aggregation

Loki is a log aggregation system built by Grafana Labs. Unlike traditional logging systems, Loki does not index the content of the logs. Instead, it only indexes metadata, which makes it efficient and cost-effective.

Step 2: Installing Loki

Download and install Loki directly on your machine:

```
# Download Loki binary
wget https://github.com/grafana/loki/releases/download/v2.7.1/loki-linux-amd64.zip
# Unzip the Loki binary
unzip loki-linux-amd64.zip
# Make it executable
sudo chmod a+x loki-linux-amd64
```

Step 3: Configuring Loki

Now, create a configuration file for Loki (`loki-config.yaml`):

```
#loki-config.yaml

auth_enabled: false

server:
  http_listen_port: 3100

ingester:
  lifecycler:
    ring:
      kvstore:
        store: inmemory
      replication_factor: 1
    final_sleep: 0s
  chunk_idle_period: 5m
  chunk_retain_period: 30s
  max_transfer_retries: 0

schema_config:
  configs:
    - from: 2020-10-24
      store: boltdb-shipper
      object_store: filesystem
      schema: v11
      index:
        prefix: index_
        period: 168h

storage_config:
  boltdb_shipper:
    active_index_directory: /tmp/loki/boltdb-shipper-active
    cache_location: /tmp/loki/boltdb-shipper-cache
    shared_store: filesystem
  filesystem:
    directory: /tmp/loki/chunks

limits_config:
  enforce_metric_name: false
  max_streams_per_user: 0
  max_entries_limit_per_query: 0

chunk_store_config:
  max_look_back_period: 0s

table_manager:
  retention_deletes_enabled: false
  retention_period: 0s
```

Start Loki with:

```
./loki-linux-amd64 -config.file=loki-local-config.yaml
```

Try accessing the correct Loki metrics endpoint using:

```
http://loki:3100/metrics
```

This endpoint should return Prometheus-style metrics from Loki, including information on the health and performance of the Loki service. If you still encounter issues, here are a few troubleshooting steps:

Loki will now be running on port `3100`.

Add Loki to Grafana:

- In Grafana, go to **Data Sources** and click **Add Data Source**.
- Choose **Loki**.
- Enter `http://localhost:3100` as the URL and save it.

3. Setting Up Promtail for Log Streaming

Promtail is a log shipper that streams logs from your application to Loki. In this setup, we will stream logs from our Rails app to Loki using Promtail.

Step 4: Installing Promtail

You can download Promtail from its official repository:

```
# Download Promtail binary
wget https://github.com/grafana/loki/releases/download/v2.8.2/promtail-linux-amd64.zip

# Unzip the binary
unzip promtail-linux-amd64.zip

# Make it executable
sudo chmod a+x promtail-linux-amd64
```

Step 5: Configuring Promtail

Create a configuration file for Promtail to stream logs from your Rails application:

```
- job_name: devlog
  static_configs:
  - targets:
      - localhost
    labels:
      job: devlog
      __path__: /home/developer/Desktop/myapp/log/developement.log #Path of log file
      stream: stdoutIn this config:
```

- The `path` should point to the log file of your Rails app.
- **`localhost`** means that Promtail is configured to scrape logs from the local system where it's running, instead of an external machine.

Start Promtail with:

```
promtail -config /path/to/your/promtail-config.yaml
```

If Promtail does not have read access, change the permissions:

```
sudo chmod +r /home/developer/Desktop/myapp/log/development.log
```

4. Integrating Grafana with Loki

Now that Loki is set up and receiving logs via Promtail, we need to configure Grafana to visualise the logs.

Step 6: Adding Loki as a Data Source in Grafana

1. In Grafana, go to **Configuration** > **Data Sources**.
2. Click **Add data source** and select **Loki**.

3. In the **URL** field, enter `http://localhost:3100`.
4. Click **Save & Test** to ensure Grafana can connect to Loki.

Step 7: Creating Dashboards for Log Visualisation

Once Loki is added as a data source, you can create a dashboard to visualise your Rails logs.

1. Go to **Create** > **Dashboard**.
2. Click **Add new panel**.
3. In the **Query** field, select **Loki** as the data source and use a simple query like this:

- `{job="rails_logs"}`

1. Customise the panel to display log lines and their corresponding metadata, like timestamps and log levels.
2. Save the dashboard and enjoy real-time log visualisation for your Rails app.

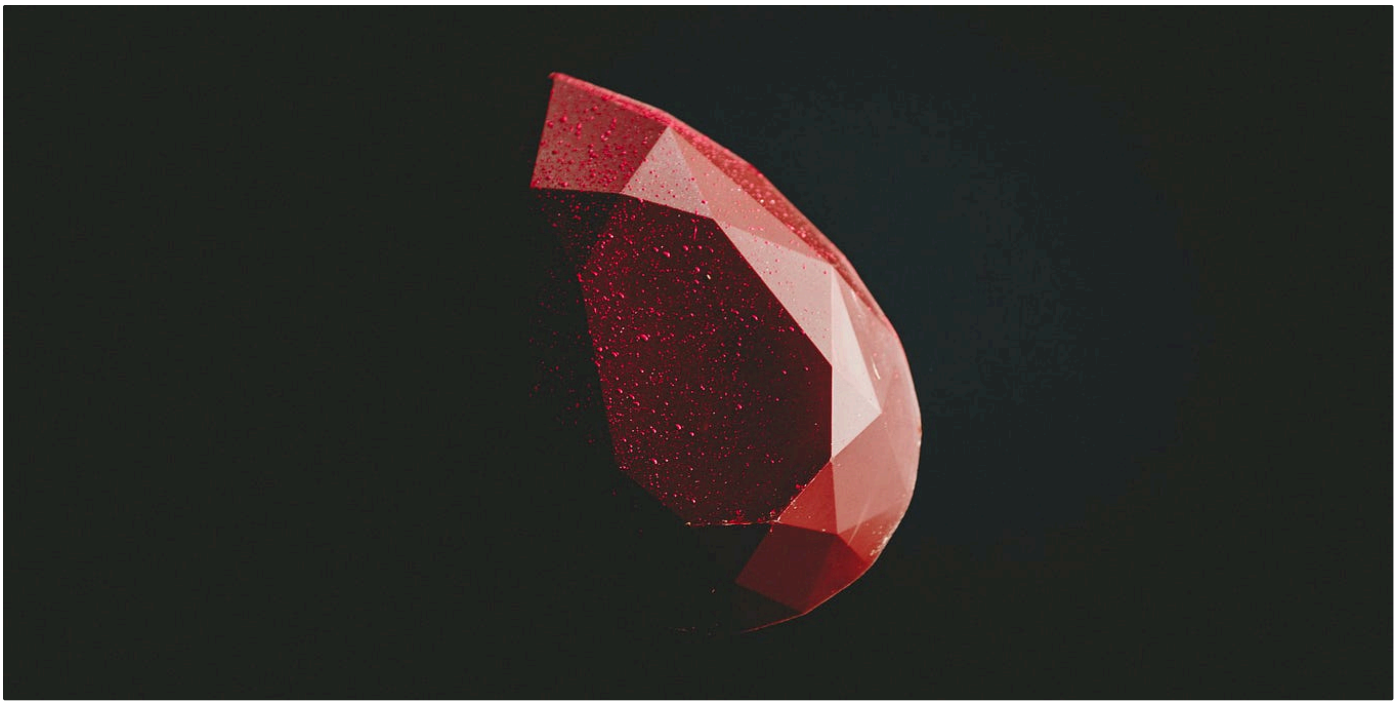
5. Conclusion

In this blog, we've successfully set up Grafana, Loki, and Promtail to monitor and visualise logs from a Ruby on Rails application. Grafana helps you keep an eye on important metrics, while Loki and Promtail make log management efficient and scalable. By using this stack, you'll have a real-time overview of your application's performance and logs, which can drastically improve troubleshooting and debugging.

You can read more about what Loki and Promtail ***here....***

**Follow us and lets explore Ruby on Rails together **

Follow us on Medium



Rails to Rescue

Ruby on Rails 8: Big Changes, Big Benefits!

Photo by Joshua Fuller on Unsplash

Sep 28 97 1

Bryson Meiling in Level Up Coding

Stop making your python projects like it was 15 years ago...

I have a few things I've seen across companies and projects that I've seen working with Python that are annoying, hard to maintain, and are...

Sep 28 3.9K 44

Amazon.com Seattle, WA

Software Development Engineer Mar. 2020 – May 2021

- Developed Amazon checkout and payment services to handle traffic of 10 Million daily global transactions
- Integrated Iframes for credit cards and bank accounts to secure 80% of all consumer traffic and prevent CSRF, cross-site scripting, and cookie-jacking
- Led Your Transactions implementation for JavaScript front-end framework to showcase consumer transactions and reduce call center costs by $25 Million
- Recovered Saudi Arabia checkout failure impacting 4000+ customers due to incorrect GET form redirection

Projects

NinjaPrep.io (React)

- Platform to offer coding problem practice with built in code editor and written + video solutions in React
- Utilized Nginx to reverse proxy IP address on Digital Ocean hosts
- Developed using Styled-Components for 95% CSS styling to ensure proper CSS scoping
- Implemented Docker with Seccomp to safely run user submitted code with < 2.2s runtime

HeatMap (JavaScript)

- Visualized Google Takeout location data of location history using Google Maps API and Google Maps heatmap code with React
- Included local file system storage to reliably handle 5mb of location history data
- Implemented Express to include routing between pages and jQuery to parse Google Map and implement heatmap overlay

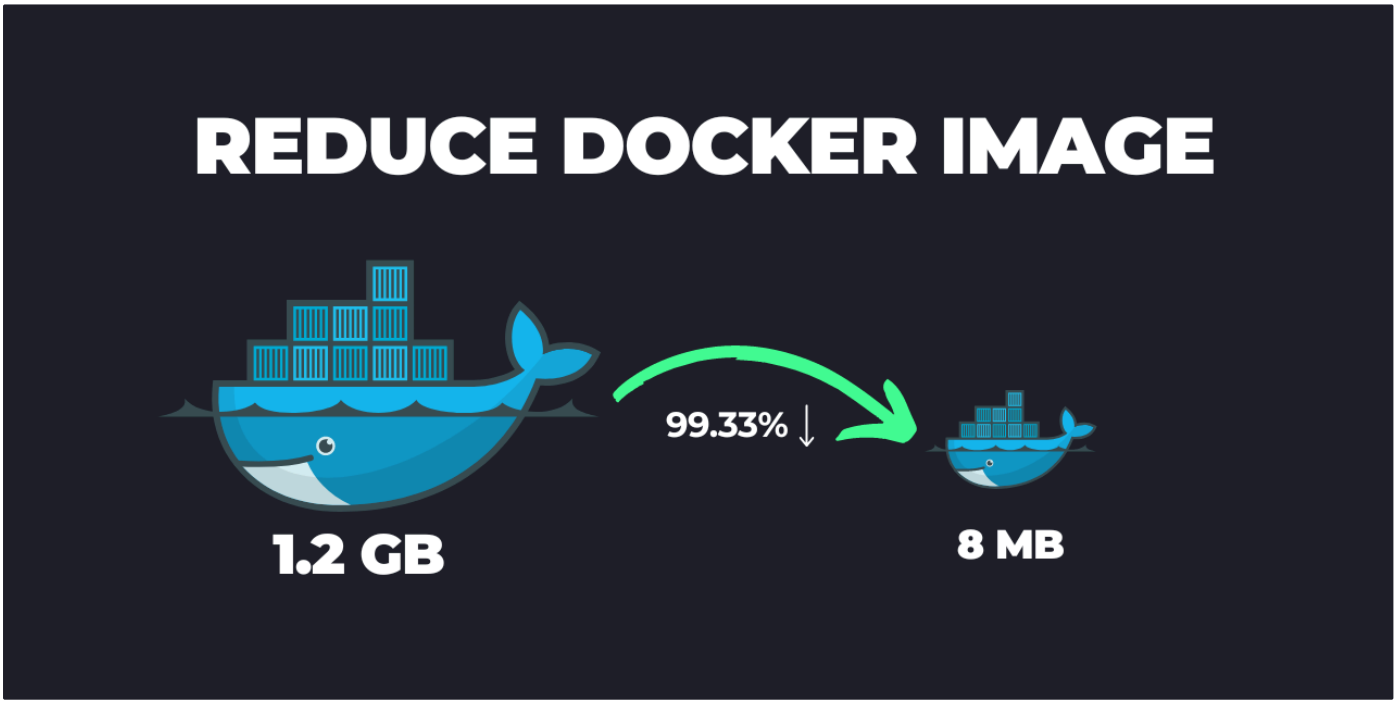
Recommended from Medium

Hayk Simonyan in Level Up Coding

STOP using Docker Desktop: Faster Alternative Nobody Uses

Ditch Docker Desktop and try this faster, lighter tool that will make your life easier!

Oct 8 2.1K 37



Dipanshu in AWS in Plain English

Docker pros are shrinking images by 99%: The hidden techniques you can't afford to miss

Unlock the secrets to lightning-fast deployments and slashed costs—before your competitors do

Sep 18 1.5K 8

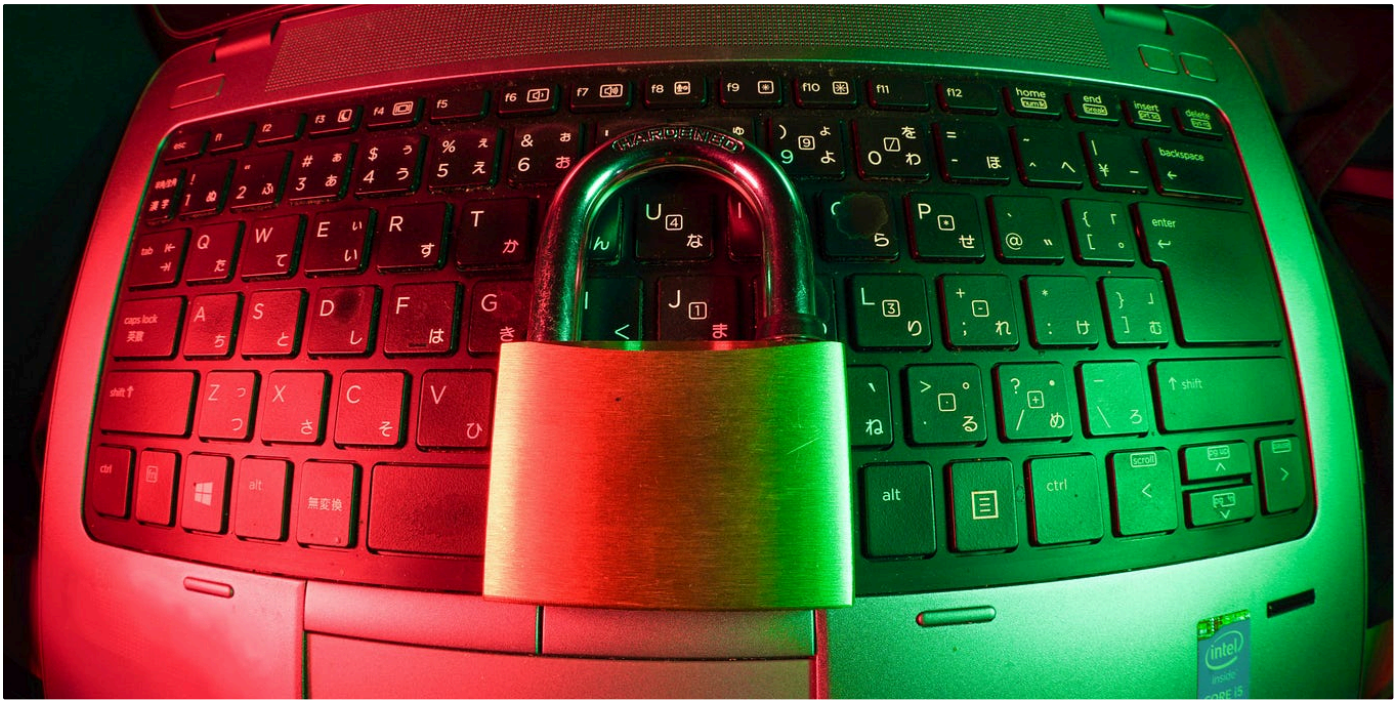
Lists

General Coding Knowledge
20 stories · 1672 saves

Coding & Development
11 stories · 863 saves

Stories to Help You Grow as a Software Developer
19 stories · 1439 saves

Leadership
61 stories · 468 saves

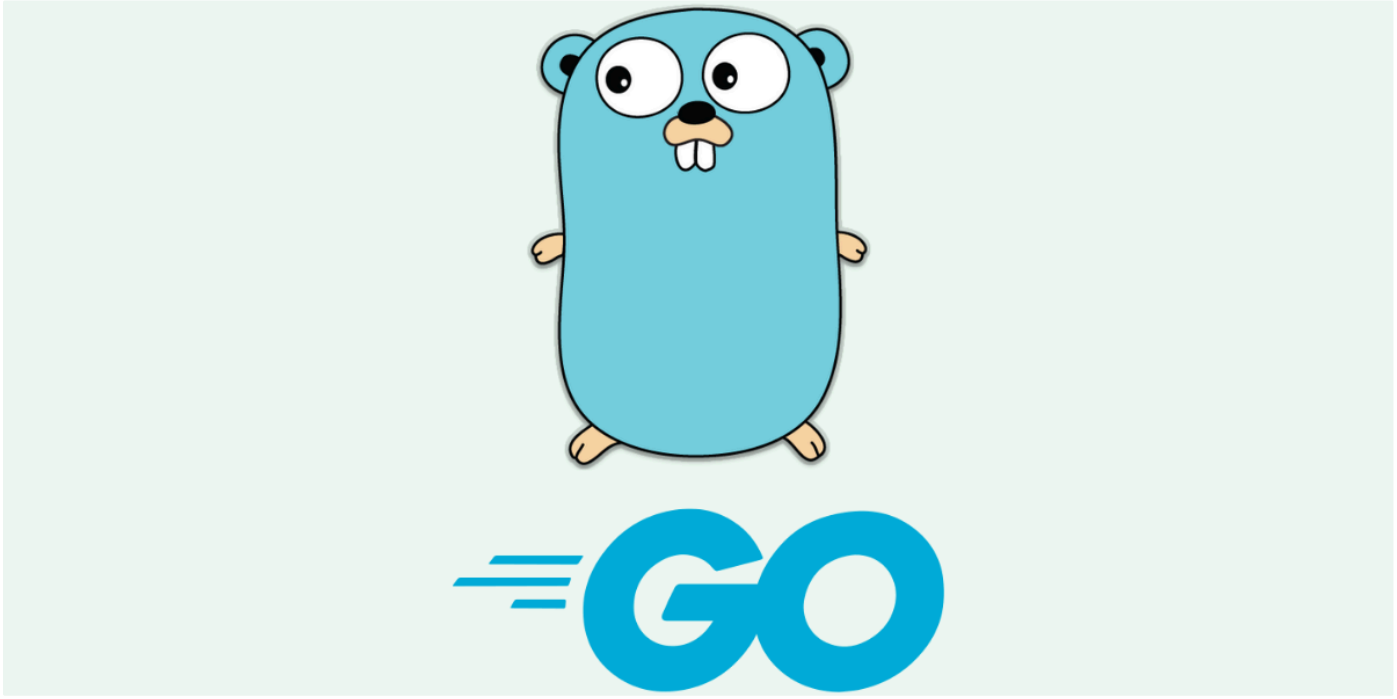


Seno Wijayanto

Securing Your Go Backend: Encryption, Vulnerability Prevention, and More!

If you're building a backend in Go (Golang), security should be a top priority. The good news? It's totally achievable to implement solid...

Sep 27 102 3



Yaswanth

Why I'm Switching from Go to Python

Hey everyone! So, after spending years writing Go code (and loving it, honestly), I've decided it's time for a change. Yup, I'm switching...

Sep 3 275 6