# Loki: A distributed database for Logs

Kaviraj Kanagaraj, Loki Maintainer

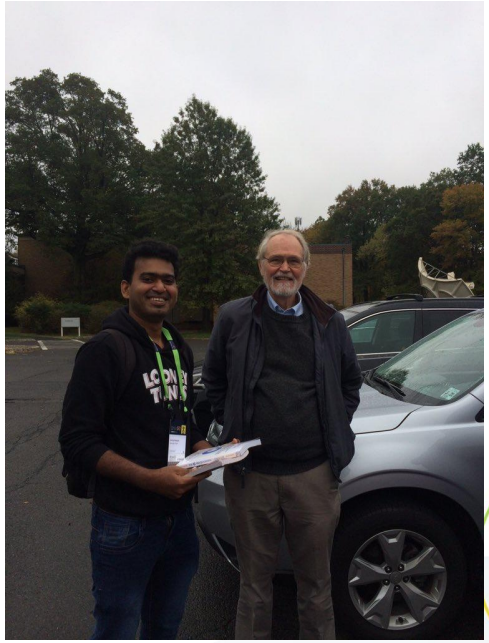
Grafana Labs


LOGS
LOGS EVERYWHERE
makeameme.org

# About Me

- SWE @ Grafana Labs
- One of the Loki Maintainers
- Using Go since ~2016
- Love discussing about distributed systems.
- UNIX fanboy

  (* that's me with Brian Kernighan @ Bell Labs 2019)



# Agenda

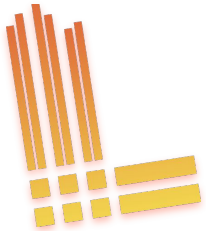
# Agenda

- What is Loki?
- Why Loki?
- Data model
- Architecture
- Operational mode
- Collecting and Querying Logs
- Other Features

# What is Loki?

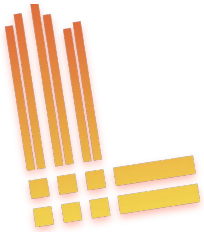
# What is Loki?

Loki is a Horizontally scalable, highly available, multi-tenant Log Aggregation System



# What is Loki?

Loki is a time series database, but for strings



# What is Timeseries database?

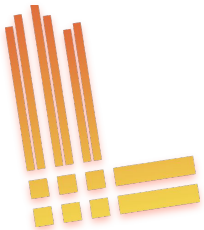
```
identifier -> (t0, v0), (t1, v1), (t2, v2), (t3, v3), ....
```

Prometheus: `{app="nginx", cluster="us-central-0"} -> [(1653994269, 34.5)]`

Loki: `{app="nginx", cluster="us-central-0"} -> [(1653994269, "/ GET")]`

# What is Loki?

Loki is a distributed database for “log-like” data (written in Go)



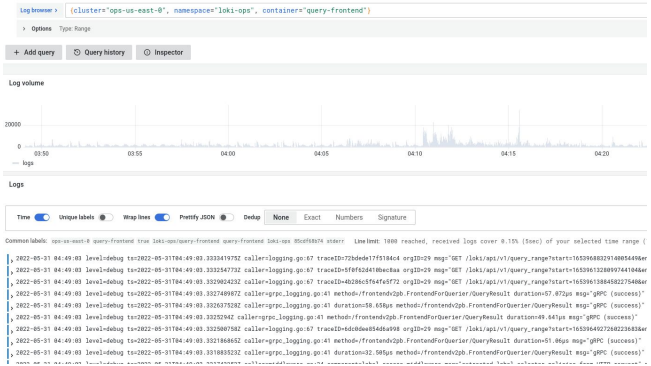
# Why Loki?

# Why Loki?

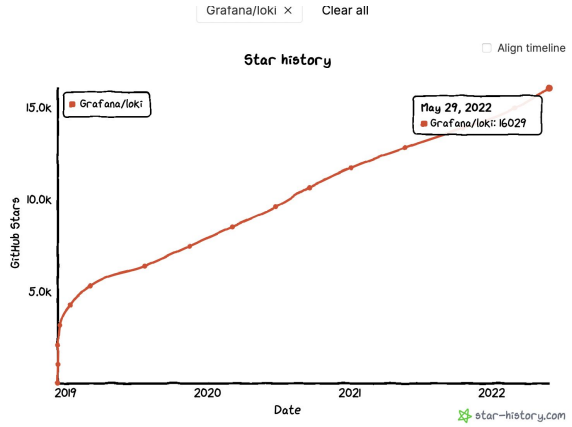

bletchley punk @alicegoldfuss · Apr 5, 2018
just give me log files and grep, I am dying
11
14
88

# Why Loki?

- Cheap & easy to operate
- Simple yet powerful Query Language (LogQL)
- First class support with Grafana
- Open Source with growing community







# Data Model

# Data model

```
2019-12-11T10:01:02.123456789Z  {app="nginx",cluster="us-west1"}  GET /about
```

**Timestamp**
with nanosecond precision

Prometheus-style **Labels**
key-value pairs

**Content**
log line

indexed

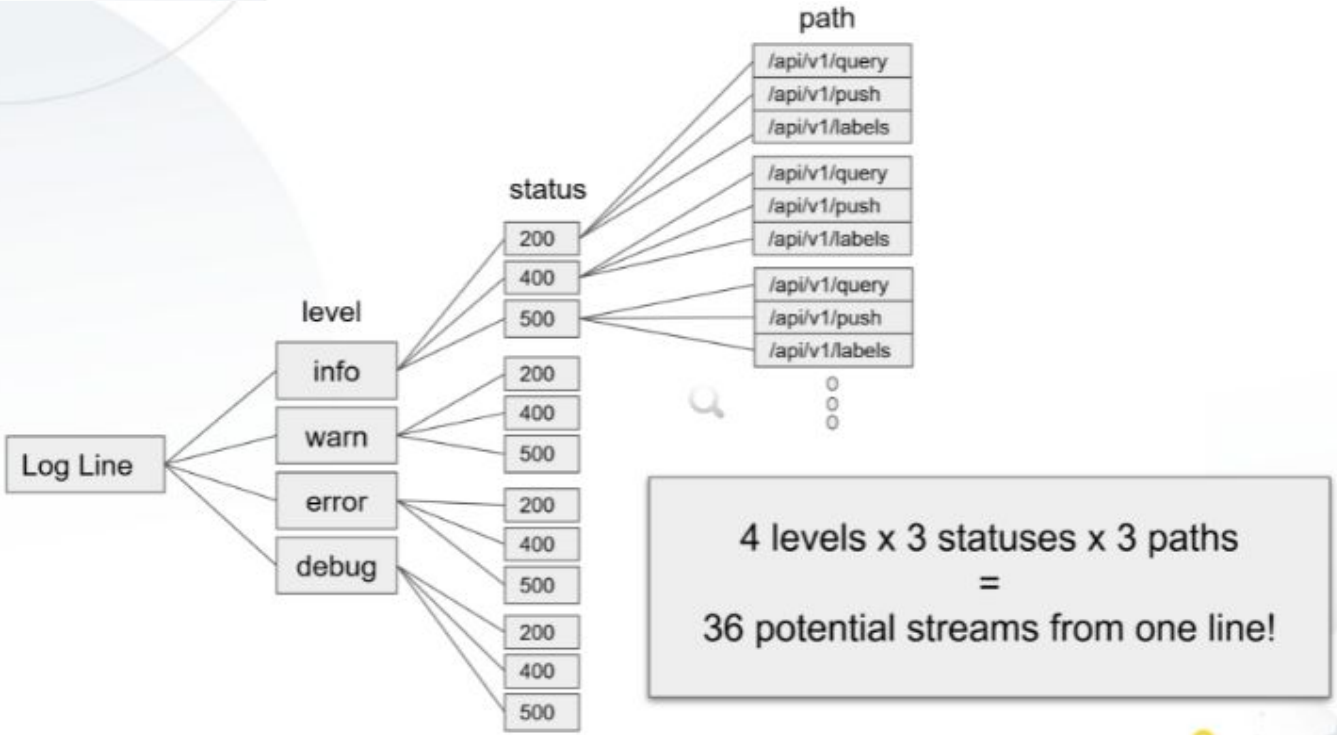
unindexed

## Log doesn't index log line

# Data model - Labels

- Fewest labels as possible
- Avoid extracting content from your logs into labels
- Labels best **describe your environment or topology of your application.**
- Good labels - “cluster”, “namespace”, “host”, “app”
- Bad labels - “UUID”, “TraceID”, “level”, “path”

# Data model - Labels



# Data model

Wait. If Loki doesn’t index original log line, how can search be efficient?

Loki leverages horizontal scaling and query time brute force to find your data

# Data Model - Loki scale

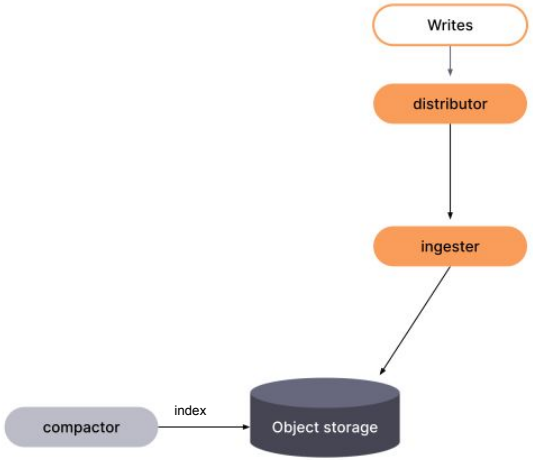
- One of our Loki cluster in Grafana Cloud
  - ~125K users (Loki is multi-tenant)
  - 460Mb/s ingestion rate **~38 TB per day** (160 ingesters)
  - **~80GB/s query throughput** (200 queriers)

- **~200 MB (index) for ~10TB of logs**
  - Can easily fit into memory without any additional hacks

# Architecture

# Architecture

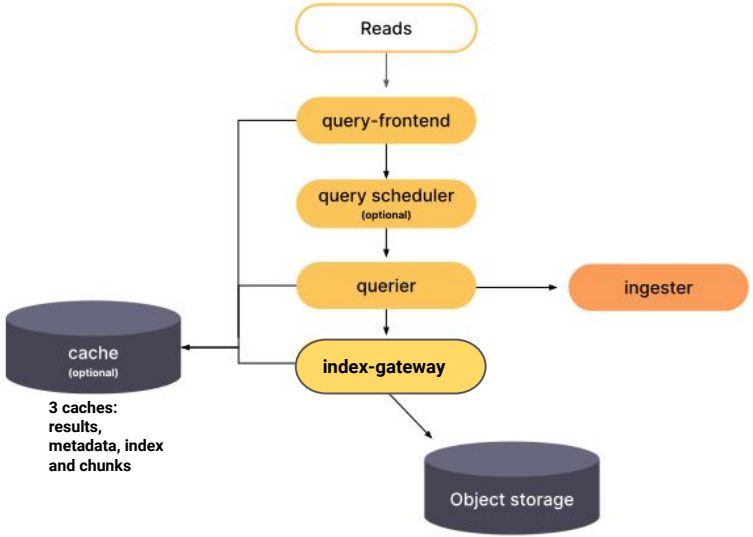
- Ingestion




/loki/api/v1/push

# Architecture

- Query



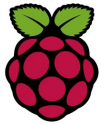

/loki/api/v1/query
/loki/api/v1/query_range
/loki/api/v1/labels
/loki/api/v1/label/<name>/values
/loki/api/v1/tail

# Operational modes

# Operational modes

**Single Binary**

- Testing
- Small installations
- Docker
- Toy projects on PI



**Microservices**

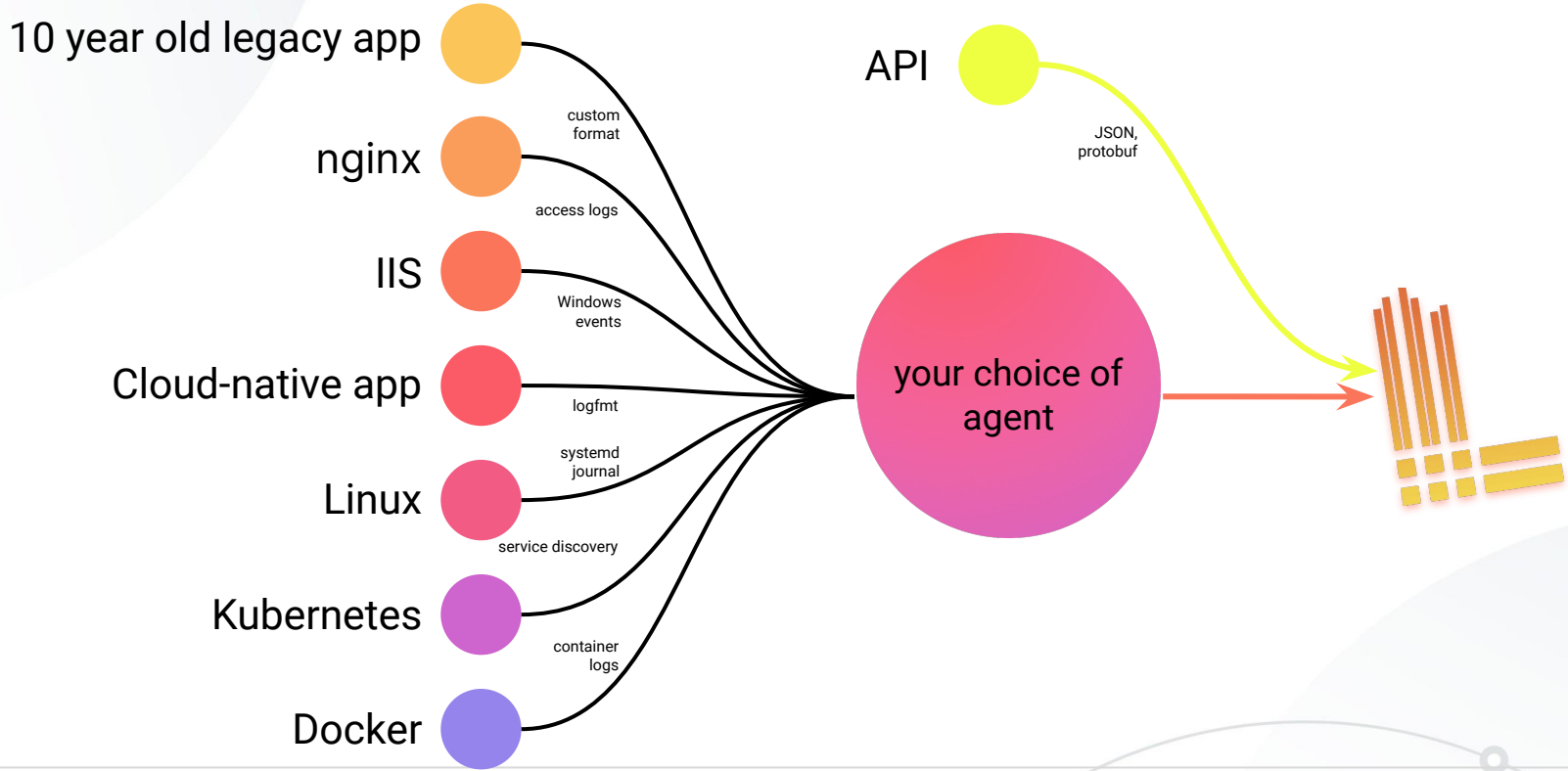
- Horizontal scalability
- Large installations
- Scale components as needed


**Simple Scalable Deployment**

- Horizontal read/write
- Mid-size installations
- Easy of operations

# Collect and Query Logs

# Collecting logs



Promtail


Grafana Agent






# Query Logs

LogQL ≅ PromQL

# LogQL Philosophy: Pipes for logs

```
{app=”nginx”} |= “/users/login” | json | duration > 10
```

***inspired by UNIX pipes**

# LogQL Philosophy: Functions for metrics

```
sum by (path) (
  count_over_time(
    {app=”nginx”} |= “/users/login” | json | duration > 10
    [5m]
  )
)
```

*inspired by Prometheus

# LogQL - Stream Selectors

```
{container="nginx"}

{cluster="us-central-1", container=~"nginx|envoy|caddy|traefik"}

{namespace="prod", app!="agent"}
```

# LogQL - Line Filter

```
{container=”frontend”} |= “error”

{cluster=”us-central-1”} |= “error” != “timeout”

{namespace=”prod” }  |~ `(?i)error`

{container=”nginx”} |= ip("192.168.4.5/16")
```

# LogQL - Extract labels at runtime using Parsers

## Support different parsers

```
{container="frontend"} |= "error" | logfmt

{cluster="us-central-1"} |= "error" != "timeout" | json

{container="nginx"} | pattern `<ip> - - <_> "<method> <uri> <_>"
<status> <size> <_> "<agent>" <_>`

{container="nginx"} | regexp `^(?P<remote_host>\\S+)
(?P<user_identifier>\\S+) (?P<user>\\S+) \\[(?P<ts>[^\\]]+)\\]
\\"(?P<method>\\S+) (?P<uri>\\S+)...
```

# LogQL - Label filter

*msg="request received" latency=20s bytes_consumed=1gb*

```
{container="frontend"}| logfmt | latency > 15s and
bytes_consumed > 20MB
```

*{"user":"1123", "latency": 10}*

```
{cluster="us-central-1"} | json | (user="1123" or user="2234")
and latency > 5
```

# LogQL - Label Format

## Mutate or create new labels

```
{container="query-frontend",namespace=~"cortex-.*"} |= "stats"
|= "query_range" | logfmt | label_format length=`{{sub
.param_end .param_start}}`  | length >432000
```

## Rename Labels

```
{cluster=”us-central-1”} | json | label_format
latency=user_access_latency
```

# LogQL - Line format

{name="querier", namespace="loki"} | logfmt | duration > 5s

**| line_format "{{ .ts}}\t{{.duration}}\tthroughput = {{.throughput}}/s\t{{ .query }} "**

# LogQL - Metric queries

## Range vector aggregation

```
count_over_time({cluster=”us-central-1”} |= “error” != “timeout” [1m])
```

## Unwrap range vector aggregation

```
sum by (org) (

    sum_over_time({cluster=”us-central-1”} |= “processed” | logfmt | unwrap  bytes_processed[1m])

)
```

# LogQL - Metrics queries

```
sum(quantile_over_time(0.99,
  {cluster="ops-tools1",container="ingress-nginx"}
    |  json
    | __error__ = ""
    | unwrap request_time [1m])
) by (path)
```

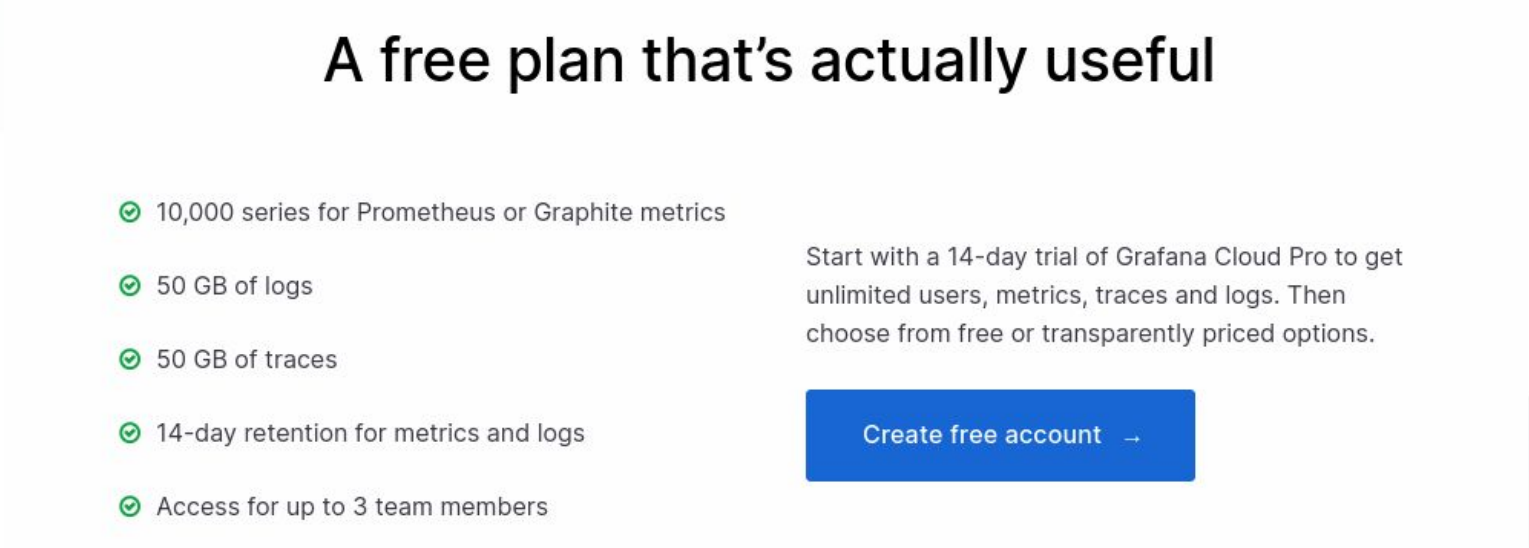
# Other features

# Other features

- Multiline support
- Alerting Rules
- Recording rules
- Custom retention
- Windows Event logs

# Try Loki with Grafana Cloud

- https://grafana.com/products/cloud/



# Thanks for listening!

# Resources

- Grafana Loki docs - https://grafana.com/docs/loki/latest/
- Github - https://github.com/grafana/loki
- Slack channel - #loki in grafana.slack.com

@kvrajk