Grafana Labs

Instrument your Go backend in minutes

A gentle introduction to the Grafana Observability Stack.

15 September 2022

Presenter

Lukasz Gut
Senior Software Engineer

Overview

Introduction to the three pillars of observability

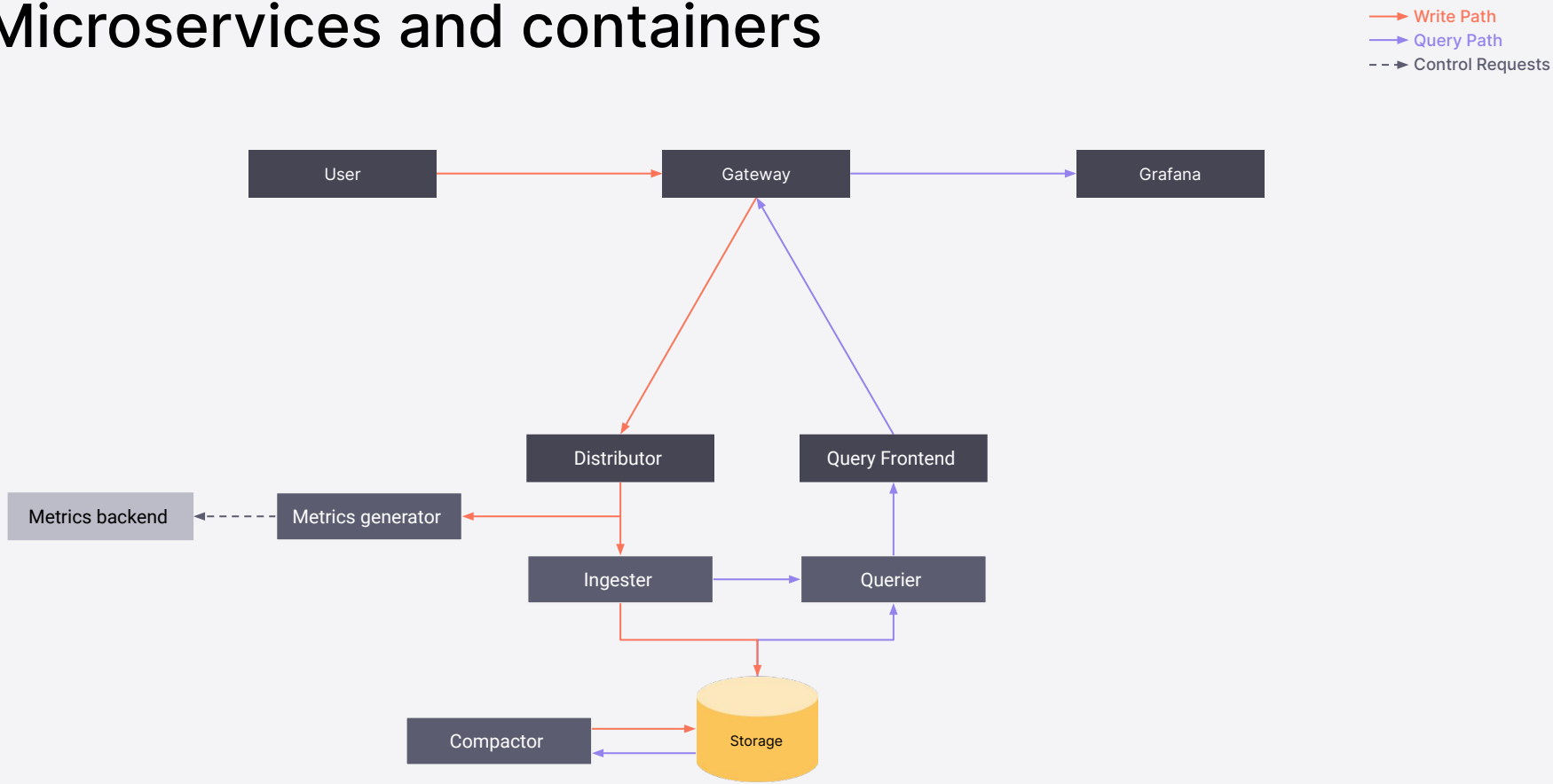
The LGTM stack

Instrumenting your application

Connecting it all together

Demo

Microservices and containers



Logs

Metrics

Traces

Grafana

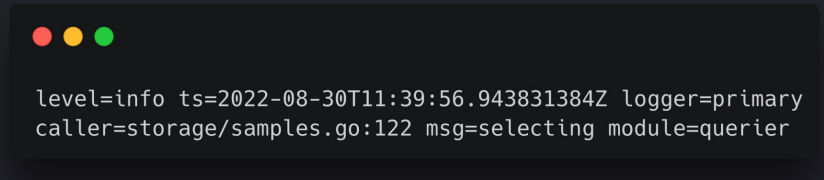
- The open source platform for monitoring and observability
- Lets you visualize data from heterogeneous sources

The holy backends

- Grafana Loki
  - Open source log aggregation system
  - LogQL support
- Grafana Mimir
  - Open source TSDB that provides a scalable long-term storage for Prometheus
  - PromQL support
- Grafana Tempo
  - Open source and high-scale distributed tracing backend
  - TODO: TraceQL support

Logging is simple

- Use correct format
  - logfmt, json
  - Arbitrary format can be used but requires regex parsing



Metrics are even simpler

- Just use Prometheus
- Supports all major languages
- First-class citizen in Grafana

```go
package main

import (
        "net/http"

        "github.com/prometheus/client_golang/prometheus/promhttp"
)

func main() {
        http.Handle("/metrics", promhttp.Handler())
        http.ListenAndServe(":2112", nil)
}
```

Traces... can be tricky

- Many “conflicting” implementations
- OpenTelemetry is the way to Go...

  even though it’s not as simple

```go
package main

import (
    "context"
    "fmt"
    "log"
    "net/http"
    "time"

    "go.opentelemetry.io/contrib/instrumentation/net/http/otelhttp"
    "go.opentelemetry.io/otel"
    "go.opentelemetry.io/otel/attribute"
)

// Package-level tracer.
// This should be configured in your code setup instead of here.
var tracer = otel.Tracer("github.com/full/path/to/mypkg")

// sleepy mocks work that your application does.
func sleepy(ctx context.Context) {
    _, span := tracer.Start(ctx, "sleep")
    defer span.End()

    sleepTime := 1 * time.Second
    time.Sleep(sleepTime)
    span.SetAttributes(attribute.Int("sleep.duration", int(sleepTime)))
}

// httpHandler is an HTTP handler function that is going to be instrumented.
func httpHandler(w http.ResponseWriter, r *http.Request) {
    fmt.Fprintf(w, "Hello, World! I am instrumented automatically!")
    ctx := r.Context()
    sleepy(ctx)
}

func main() {
    // Wrap your httpHandler function.
    handler := http.HandlerFunc(httpHandler)
    wrappedHandler := otelhttp.NewHandler(handler, "hello-instrumented")
    http.Handle("/hello-instrumented", wrappedHandler)

    // And start the HTTP serve.
    log.Fatal(http.ListenAndServe(":3000", nil))
}
```

Connecting it all together

Self-managed

Not all who wander are lost

Requires deploying and operating Grafana Agent.

Requires deploying and operating the Grafana Stack: Grafana, Grafana Loki, Grafana Mimir, and Grafana Tempo.

Free if you don’t value your time.

vs

Grafana Cloud

One click away

Requires deploying and operating Grafana Agent.

Fully-managed Grafana, Grafana Cloud Logs, Grafana Cloud Metrics, Grafana Cloud Traces.

Offers a generous free plan.

Demo

Q&A

Grafana Labs

Thank you