Grafana Labs

Grafana Beyla

Instrumenta tu aplicación sin añadir una sola línea de código

Mario Macías Lloret

Grafana and Friends BCN Meetup
Septiembre 2023

Acerca de mí

- Programando desde 2003
  - 2006-2016 Investigador Ph.D.
  - 2009-2019 Profesor a tiempo parcial

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

- Desde 2016, trabajando en grupos relacionados con la

midokura new relic. Red Hat

Grafana

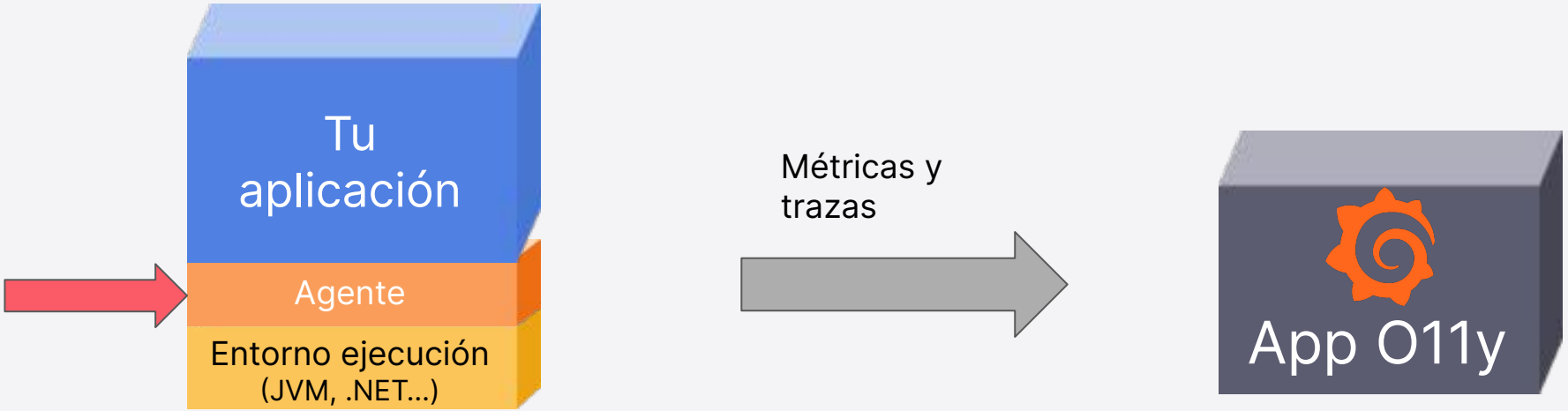
- Libros
  - Programación en Go. Marcombo Editorial
  - Introducción a Apache Spark. Editorial UOC
  - Del bit a la Nube. Amazon Kindle

Grafana Labs

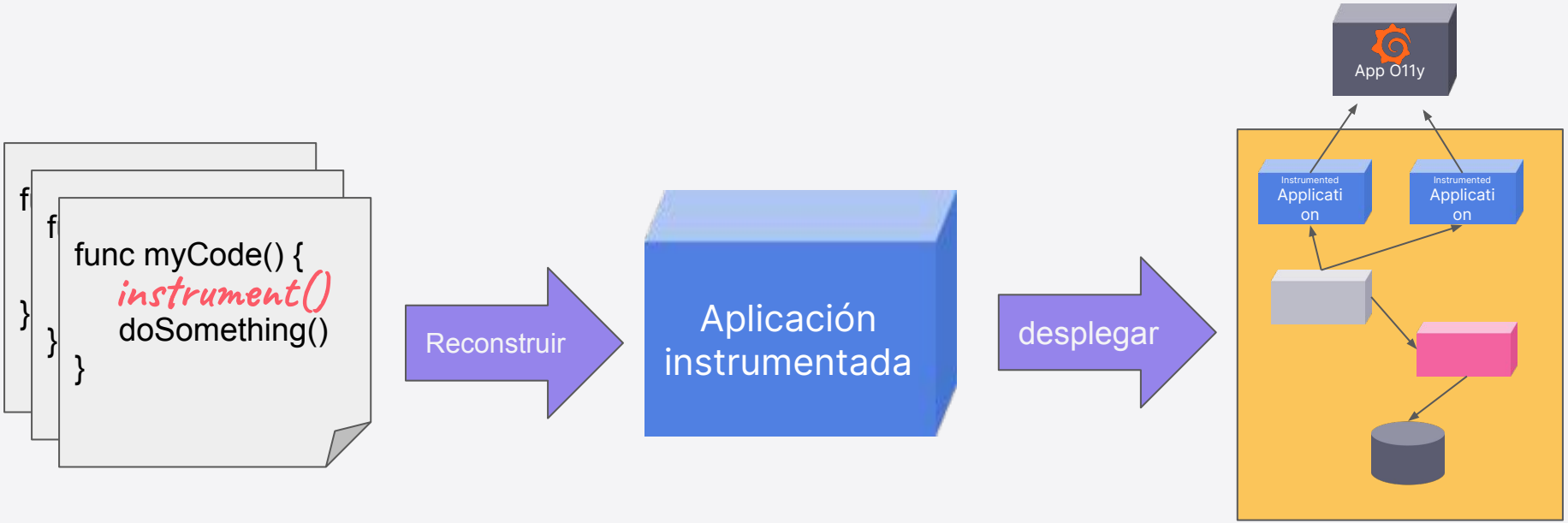
Introducción

Instrumentando una aplicación existente

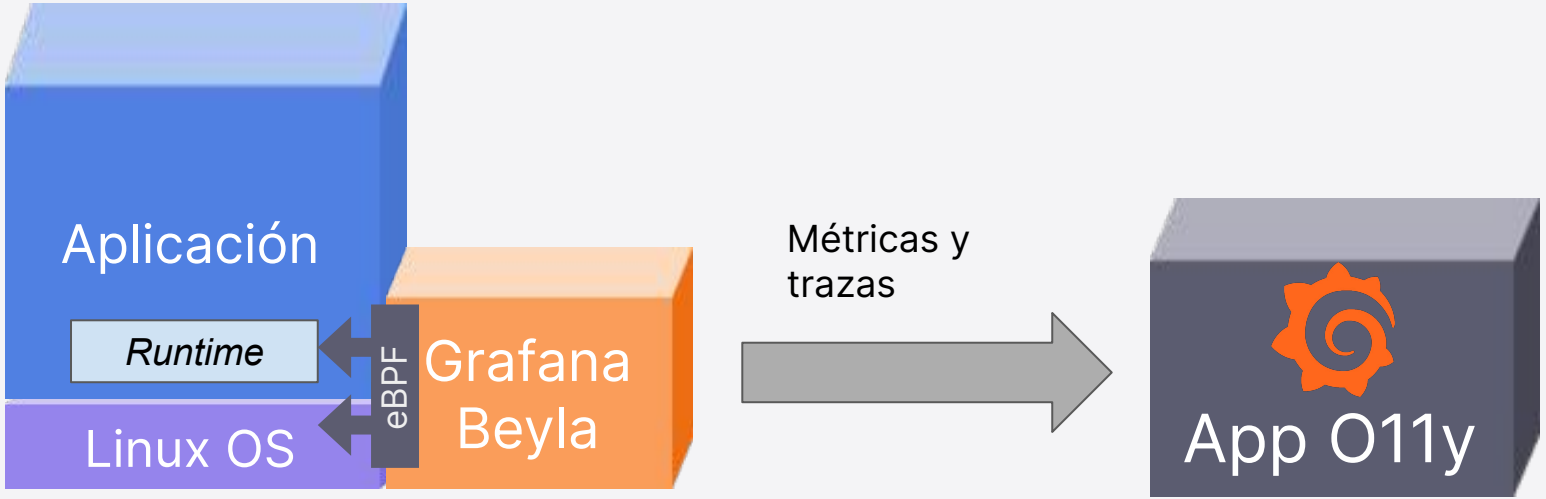
Instrumentación mediante agente



Instrumentación manual



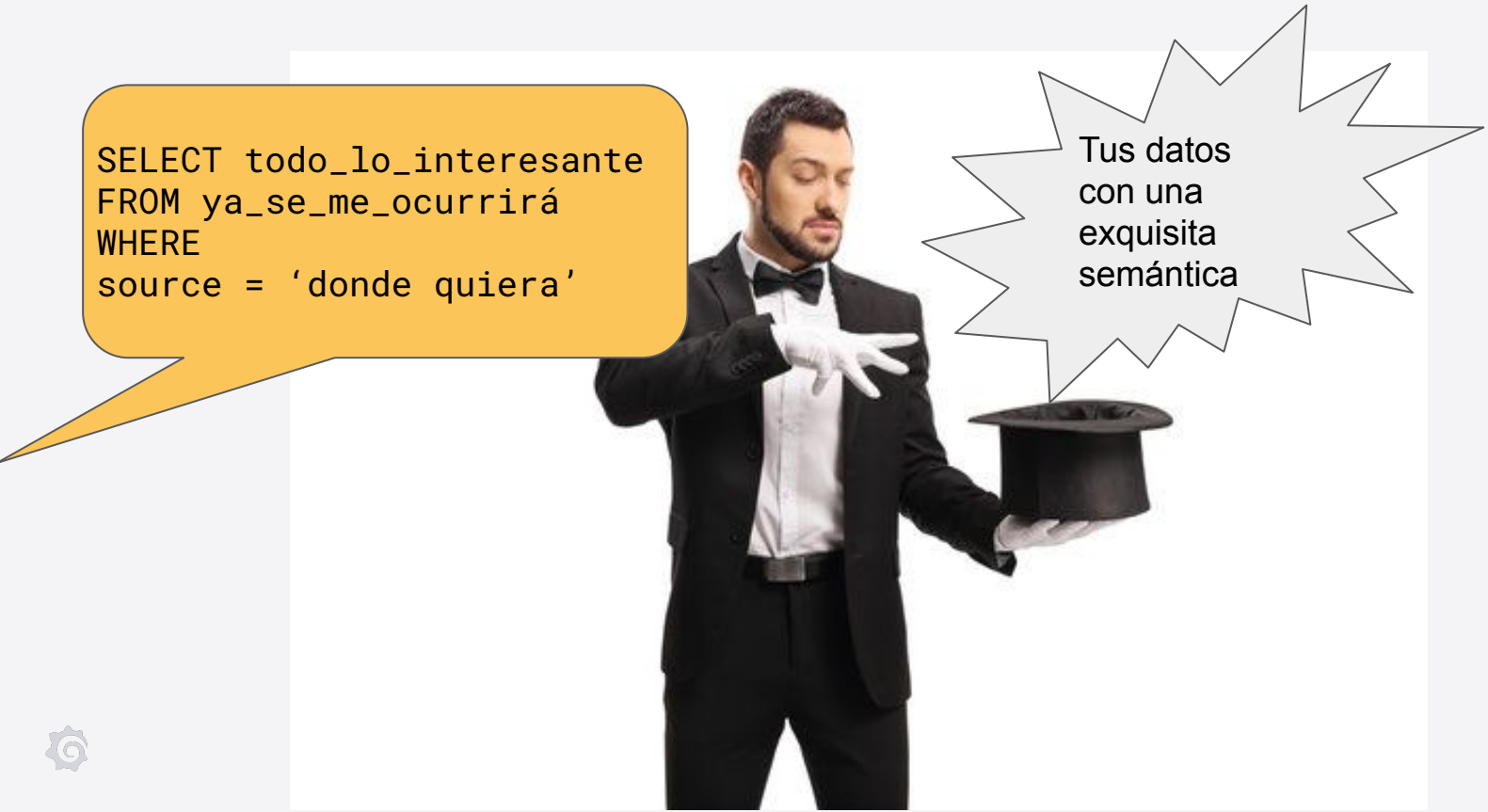
Auto-instrumentación nativa con Beyla y eBPF



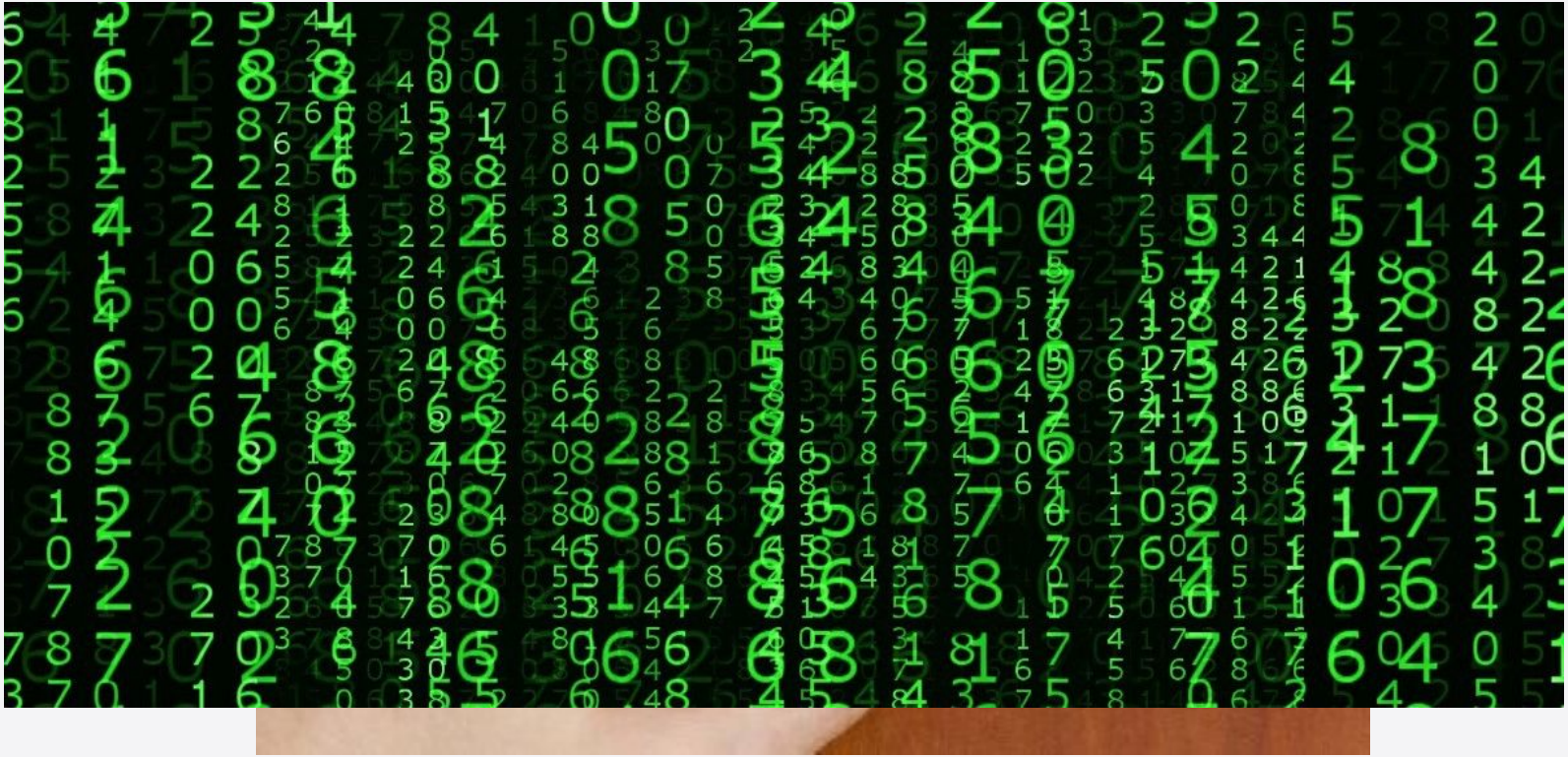
Grafana Labs

¿E... B... P... qué?

Qué no es eBPF



¿A qué se parece eBPF?

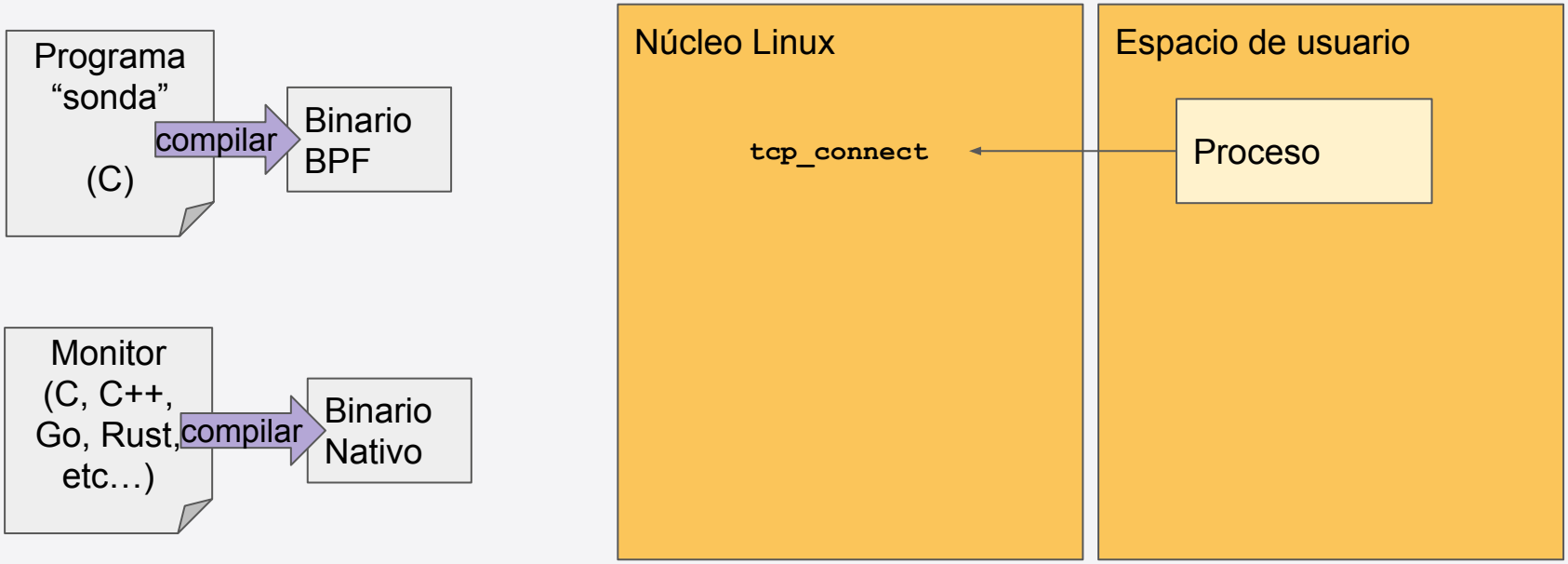


eBPF

- Extended Berkeley Packet Filter
- Máquina Virtual JIT situada en el Kernel de Linux
- Permite “enganchar” tus propios programas a diferentes eventos del Kernel o el código de usuario
- Desde esos programas puedes ver (incluso modificar) la memoria de tu sistema en ese momento.
- Es necesario conocer cómo está organizada la memoria a nivel binario
    - Argumentos
    - Variables locales
    - Valores de retorno
    - Estructuras de datos

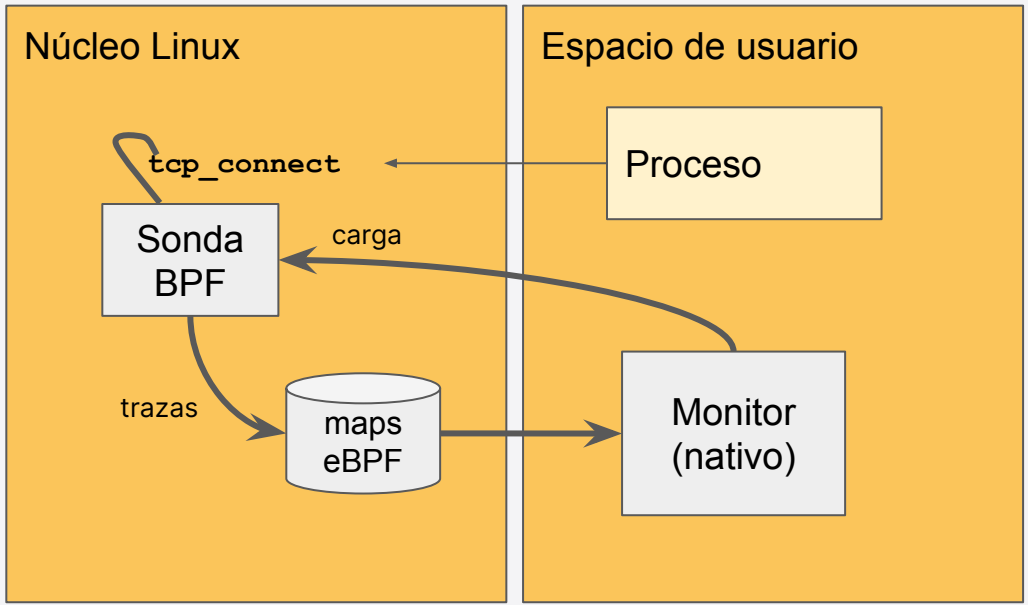
Ejemplo: trazar cada nueva conexión TCP

```
int tcp_connect(struct sock *sk);
```



Ejemplo: trazar cada nueva conexión TCP

```
int tcp_connect(struct sock *sk);
```



Ventajas de eBPF

- Rápido: compilado para máquina JIT
- Estable
  - Los programas se pre-verifican antes de cargarse
  - Imposibilidad de colgar el Kernel
- Limpio
  - Si el proceso “monitor” se para, sus recursos eBPF se liberan

Desventajas de eBPF

- Difícil de depurar
- Dependiente de los detalles de implementación
  - Argumentos en pila vs. registros
  - convenciones según la arquitectura o el lenguaje
  - bigendian vs little endian
  - etc...
- Cambios en las APIs inspeccionadas pueden romper tu código
- El programa del espacio de usuario requiere de permisos elevados

Grafana Labs

Grafana Beyla

Grafana Beyla

- Instrumentación automática de servicios y clientes web
  - Go: HTTP, HTTPS y GRPC
  - Otros lenguajes: HTTP y HTTPS
- Dos tipos de “sonda”
  - User probes: añade sondas a los símbolos de un programa en GO que indiquen un servicio o cliente HTTP, HTTPS o GRPC.
  - Kernel probes: añade sondas a los símbolos del Kernel que puedan implementar servicios HTTP y conexiones TCP

Grafana Beyla

- Exportación como
  - Métricas RED (Request, Errors, Duration)
    - Prometheus
    - OpenTelemetry
  - Trazas OpenTelemetry

Beyla vs instrumentación manual

- Desventajas
  - Menos detalle/granularidad que con instrumentación manual
- Ventajas
  - No necesitas cambiar tu código
    - ¡Puedes evaluar Grafana al instante!
  - Tienes acceso a detalles del entorno de ejecución que la instrumentación manual no te permite

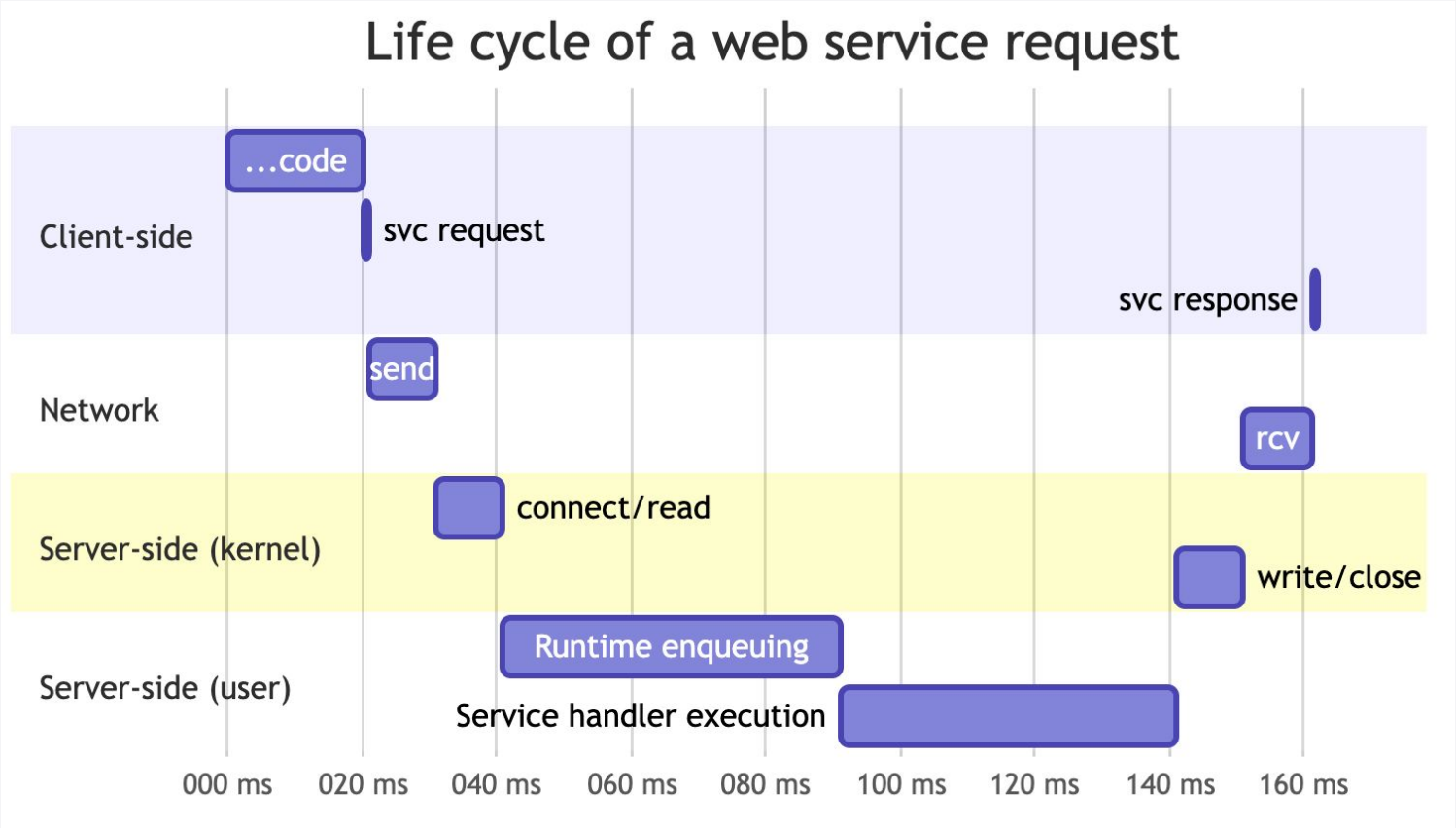
Caso práctico:
tiempo de ejecución vs tiempo de respuesta

```
func HTTPHandler(rw http.ResponseWriter, req *http.Request) {
    _, trace := tracer.Start(req.Context(), “hola”)
    rw.Write([]byte("Hola!"))
    trace.End()
}
```

tiempo de ejecución

Tiempo de ejecución (medido) < Tiempo de respuesta (percibido)

Tiempo de ejecución vs tiempo de respuesta



Grafana Labs

¡Demo!

Grafana Labs

Conclusiones

Grafana Labs

Más información:

https://grafana.com/docs/grafana-cloud/monitor-applications/beyla/