Invisible Insight: Strategies for Auto-Instrumenting Go Applications Without Code Changes

Invisible

Au ations

Changes

WHAT IS

AUTO-INSTRUMENTATION

About me

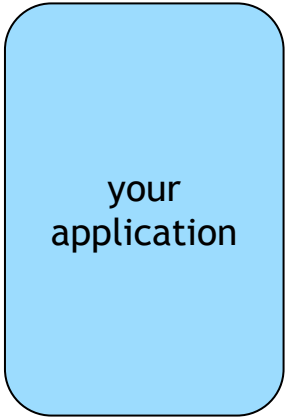
- Hannah Kim
- Graduated undergrad in 2024
- SWE I at Datadog
- Approx 1.008 years of experience

About me

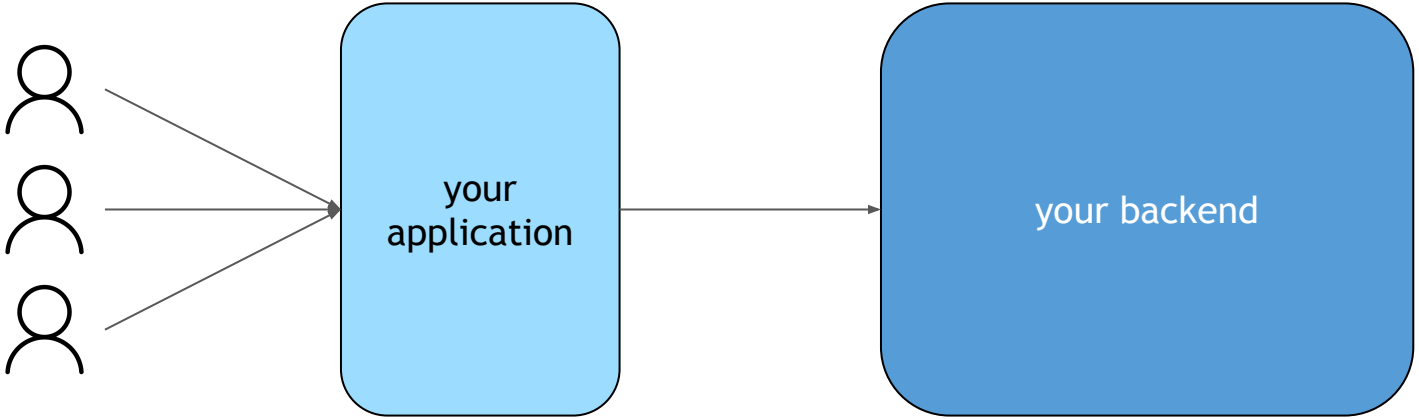
- Hannah Kim
- Graduated undergrad in 2024
- SWE I at Datadog
- Approx 1.008 years of experience

What is instrumentation?

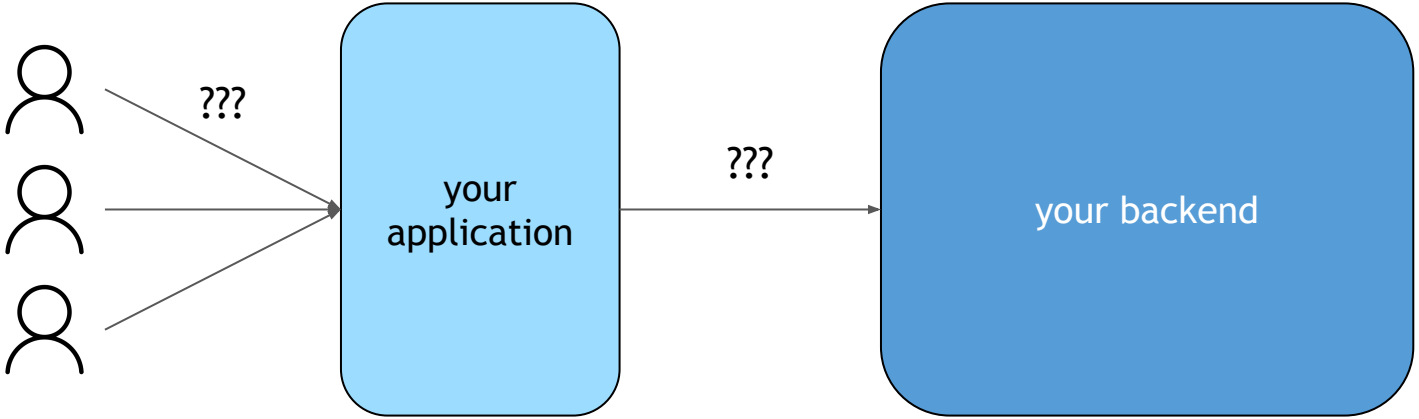
What is instrumentation?



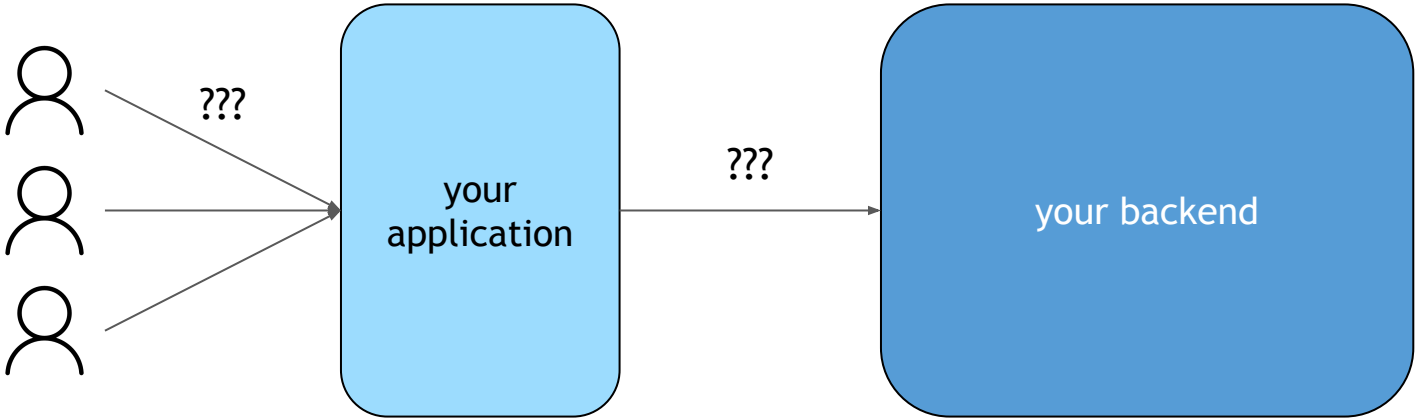
What is instrumentation?



What is instrumentation?

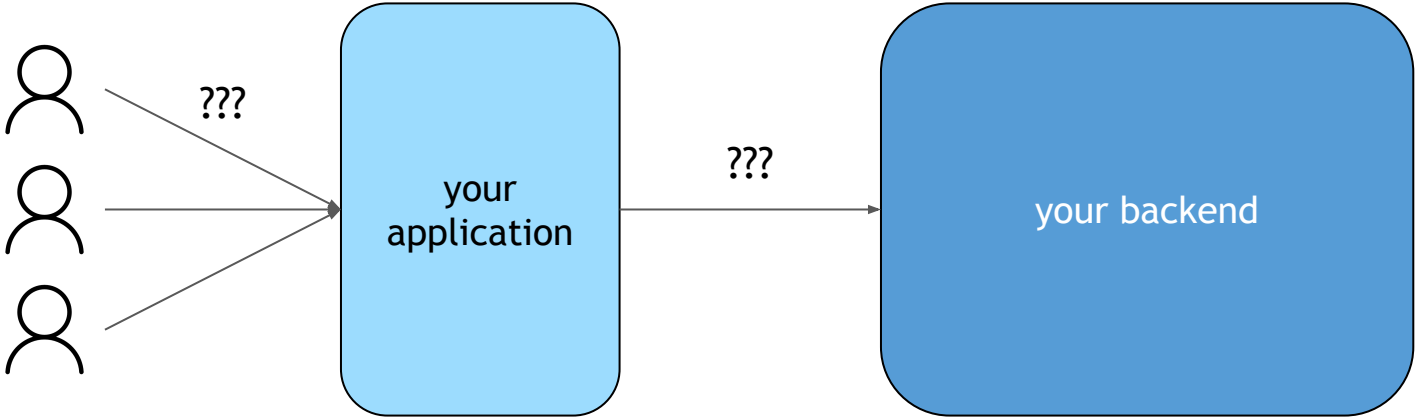


What is instrumentation?



LOGS
(what happened)

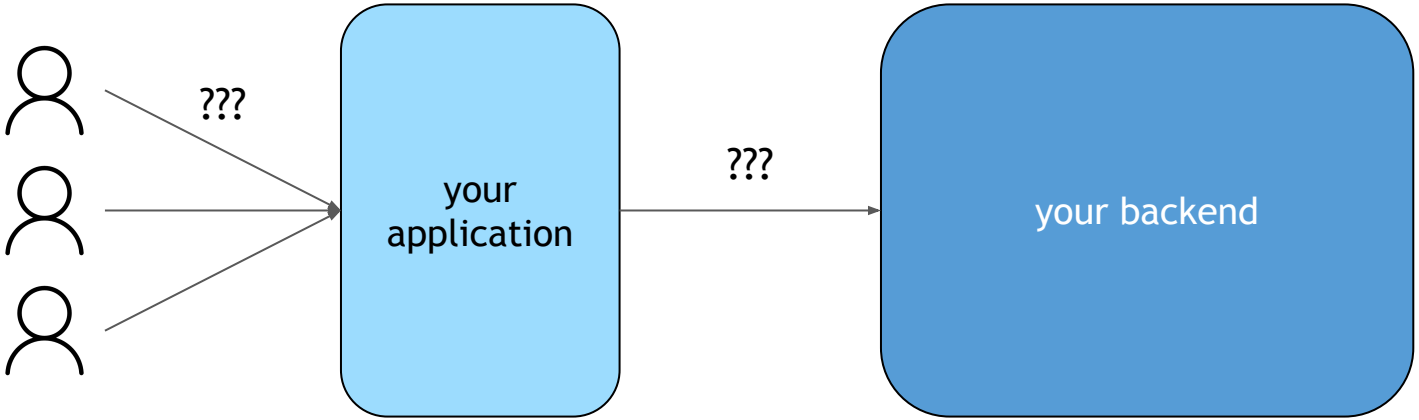
What is instrumentation?



LOGS
(what happened)

METRICS
(how much/fast things happened)

What is instrumentation?



LOGS
(what happened)

METRICS
(how much/fast things happened)

TRACES
(*how* things happened)

What is *auto*-instrumentation?

What is *auto*-instrumentation?

1. I want to know more about my code

What is *auto*-instrumentation?

1. I want to know more about my code
2. I need to instrument it, but I’m too lazy to do it myself

What is *auto*-instrumentation?

1. I want to know more about my code
2. I need to instrument it, but I'm too lazy to do it myself
3. ???

What is *auto*-instrumentation?

1. I want to know more about my code
2. I need to instrument it, but I'm too lazy to do it myself
3. ???
4. Profit 💸💸💸

What is *auto*-instrumentation?

What is *auto*-instrumentation?

⭐ auto-instrumentation: instrumenting your code (getting traces + data!) without manual code changes

What is *auto*-instrumentation?

⭐ auto-instrumentation: instrumenting your code (getting traces + data!) without manual code changes

RUN TIME

- Happens at runtime
- Sometimes causes ***source*** code changes
- Meh with compiler languages like Go

What is *auto*-instrumentation?

⭐ auto-instrumentation: instrumenting your code (getting traces + data!) without manual code changes

RUN TIME

- Happens at runtime
- Sometimes causes ***source*** code changes
- Meh with compiler languages like Go

COMPILE TIME

- Happens at… compile time
- (Before run time)
- Works great with compiler languages like Go

What is *auto*-instrumentation?

What is *auto*-instrumentation?

RUN TIME

- iovisor/gobpf
- cilium/eBPF
- OpenTelemetry Auto-Instrumentation
- Hooking
  - Shared library injection
  - Binary trampolining

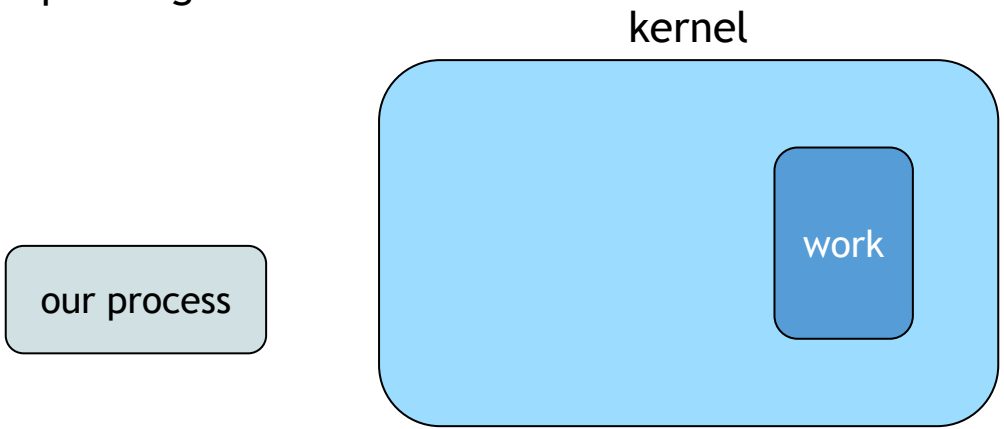
⭐ eBPF: extended Berkeley packet filter

What is *auto*-instrumentation?

RUN TIME

- iovisor/gobpf
- cilium/eBPF
- OpenTelemetry Auto-Instrumentation
- Hooking
  - Shared library injection
  - Binary trampolining

⭐ eBPF: extended Berkeley packet filter

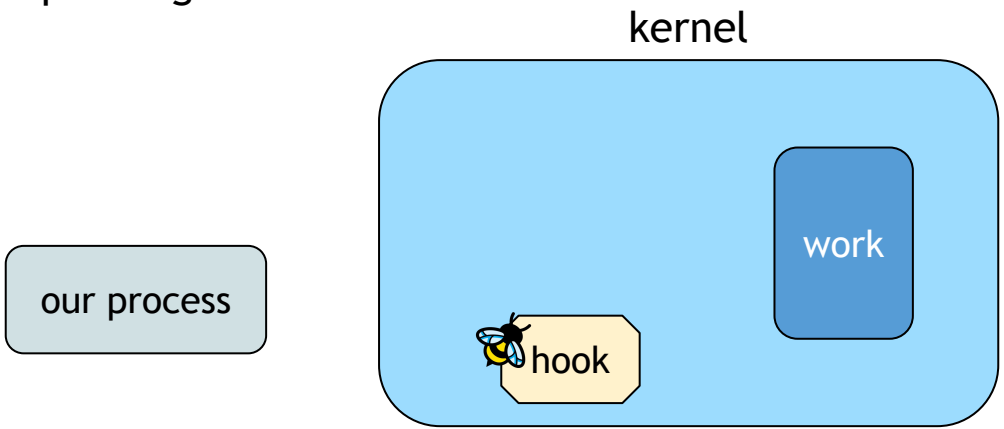


What is *auto*-instrumentation?

RUN TIME

- iovisor/gobpf
- cilium/eBPF
- OpenTelemetry Auto-Instrumentation
- Hooking
  - Shared library injection
  - Binary trampolining

⭐ eBPF: extended Berkeley packet filter

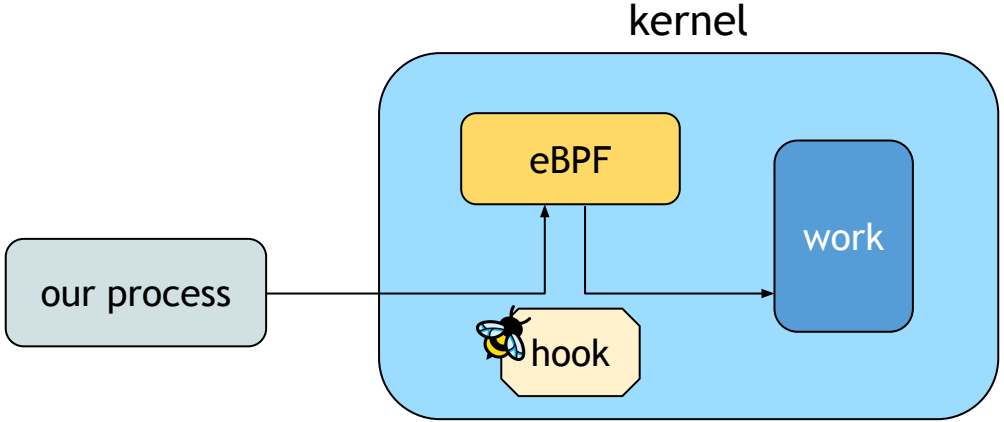


What is *auto*-instrumentation?

RUN TIME

- iovisor/gobpf
- cilium/eBPF
- OpenTelemetry Auto-Instrumentation
- Hooking
  - Shared library injection
  - Binary trampolining

⭐ eBPF: extended Berkeley packet filter



What is *auto*-instrumentation?

RUN TIME

- iovisor/gobpf
- cilium/eBPF
- OpenTelemetry Auto-Instrumentation
- Hooking
  - Shared library injection
  - Binary trampolining

COMPILE TIME

- Datadog Orchestrion
- OpenTelemetry Compile Time Instrumentation SIG

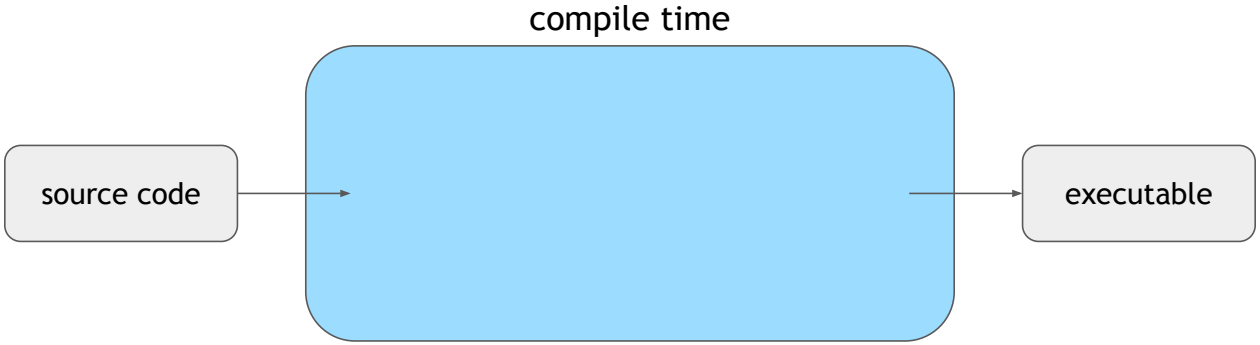
What is *auto*-instrumentation?

RUN TIME

- iovisor/gobpf
- cilium/eBPF
- OpenTelemetry Auto-Instrumentation
- Hooking
  - Shared library injection
  - Binary trampolining

COMPILE TIME

- Datadog Orchestrion
- OpenTelemetry Compile Time Instrumentation SIG



What is *auto*-instrumentation?

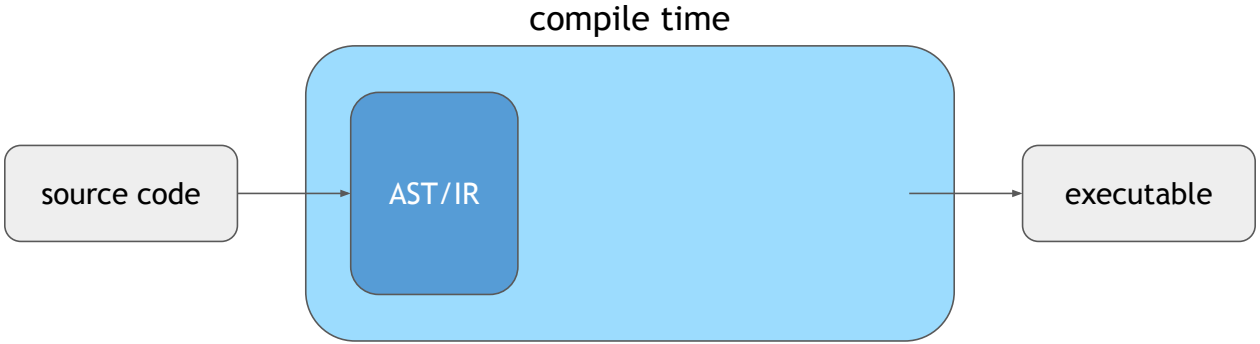
RUN TIME

- iovisor/gobpf
- cilium/eBPF
- OpenTelemetry Auto-Instrumentation
- Hooking
  - Shared library injection
  - Binary trampolining

COMPILE TIME

- Datadog Orchestrion
- OpenTelemetry Compile Time Instrumentation SIG

⭐ AST: abstract syntax tree
⭐ IR: intermediate representation



What is *auto*-instrumentation?

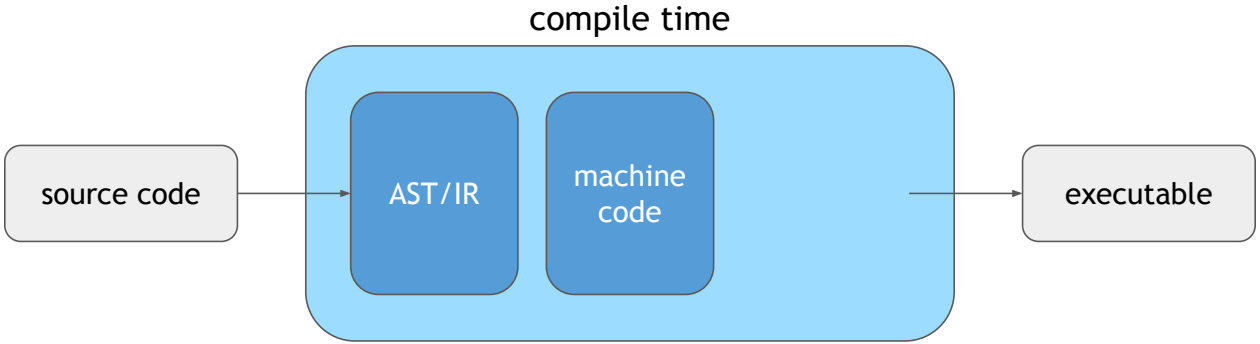
RUN TIME

- iovisor/gobpf
- cilium/eBPF
- OpenTelemetry Auto-Instrumentation
- Hooking
  - Shared library injection
  - Binary trampolining

COMPILE TIME

- Datadog Orchestrion
- OpenTelemetry Compile Time Instrumentation SIG

⭐ AST: abstract syntax tree
⭐ IR: intermediate representation



What is *auto*-instrumentation?

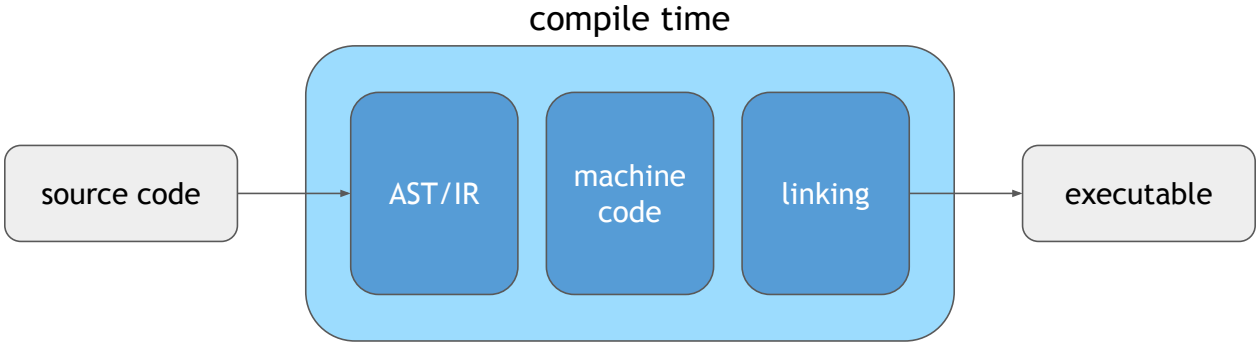
RUN TIME

- iovisor/gobpf
- cilium/eBPF
- OpenTelemetry Auto-Instrumentation
- Hooking
  - Shared library injection
  - Binary trampolining

COMPILE TIME

- Datadog Orchestrion
- OpenTelemetry Compile Time Instrumentation SIG

⭐ AST: abstract syntax tree
⭐ IR: intermediate representation



What is *auto*-instrumentation?

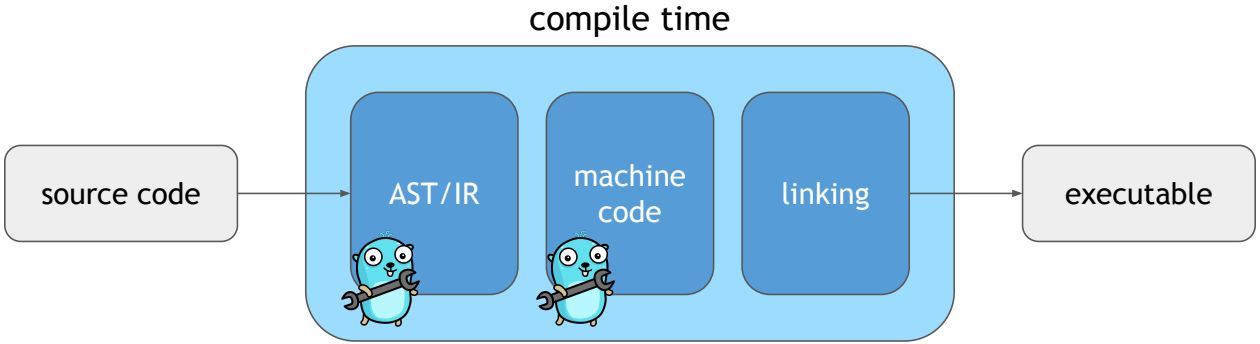
RUN TIME

- iovisor/gobpf
- cilium/eBPF
- OpenTelemetry Auto-Instrumentation
- Hooking
  - Shared library injection
  - Binary trampolining

COMPILE TIME

- Datadog Orchestrion
- OpenTelemetry Compile Time Instrumentation SIG

⭐ AST: abstract syntax tree
⭐ IR: intermediate representation




```
go run -toolexec 'orchestrion toolexec' .
```

What is *auto*-instrumentation?

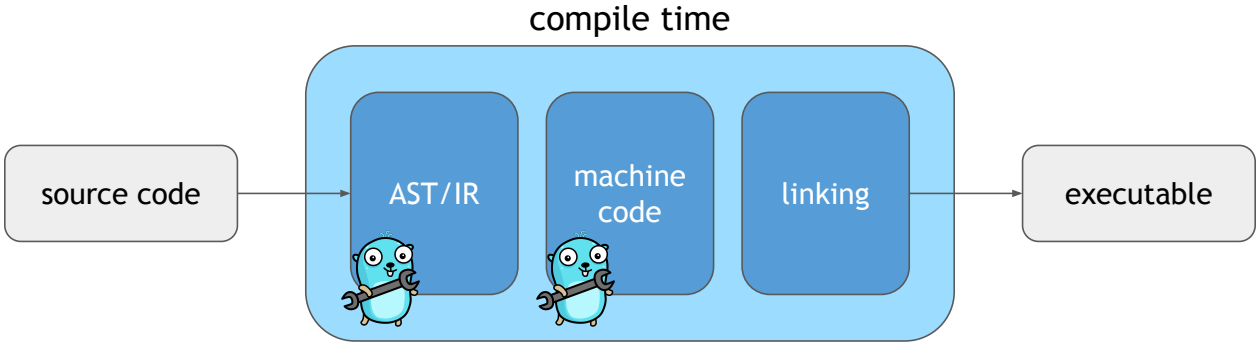
RUN TIME

- iovisor/gobpf
- cilium/eBPF
- OpenTelemetry Auto-Instrumentation
- Hooking
  - Shared library injection
  - Binary trampolining

COMPILE TIME

- Datadog Orchestrion
- OpenTelemetry Compile Time Instrumentation SIG

⭐ AST: abstract syntax tree
⭐ IR: intermediate representation




```
go run -toolexec 'orchestrion toolexec' .
```

How do they perform?

Check out the code

```
mux := http.NewServeMux()
mux.HandleFunc("/hello", func(w http.ResponseWriter, r *http.Request) {
    handlers.HelloHandler(w, r)
})

func POST(db *sql.DB, instrumentationType string, hasError bool) error {
   query := `INSERT INTO instrumentation_logs (instrumentation,
error_status) VALUES ($1, $2)`
   _, err := db.Exec(query, instrumentationType, hasError)
   return err
}
```

How do they perform?

Check out the code

```
stages: [
    // avg load-testing
    { duration: '15s', target: 100 }, // traffic ramp-up
    { duration: '30s', target: 100 }, // hold steady
    { duration: '15s', target: 0 }, // ramp-down to 0 users

    // spike-testing
    { duration: '2s', target: 1000 }, // sudden jump to 1000 users
    { duration: '2s', target: 0 }, // drop down to 0 users
]
```

How do they perform?

Check out the code

github.com/mackerelio/go-osstat

```
type CPUStats struct {
   User   uint64 `json:"user"`
   System uint64 `json:"system"`
   Idle   uint64 `json:"idle"`
   Total  uint64 `json:"total"`
}
type MemoryStats struct {
   Total uint64 `json:"total"`
   Used  uint64 `json:"used"`
}
type UptimeStats struct {
   Milliseconds uint64 `json:"milliseconds"`
}
```

How do they perform?

Check out the code

No Instrumentation

Manual Instrumentation
(OpenTelemetry SDK)

Auto Instrumentation
(OpenTelemetry eBPF)

Auto Instrumentation
(Orchestrion using OpenTelemetry)

Who wins?

Approach	Performance	Stability	Security	Portability
Auto (eBPF)				
Auto (toolchain)				

Who wins?

Approach	Performance	Stability	Security	Portability
Auto (eBPF)	⚠️			
Auto (toolchain)	⚠️			

Who wins?

Approach	Performance	Stability	Security	Portability
Auto (eBPF)	⚠️	⚠️		
Auto (toolchain)	⚠️	✅		

Who wins?

Approach	Performance	Stability	Security	Portability
Auto (eBPF)	⚠️	⚠️	⚠️	
Auto (toolchain)	⚠️	✅	✅	

Who wins?

Approach	Performance	Stability	Security	Portability
Auto (eBPF)	⚠️	⚠️	⚠️	⚠️
Auto (toolchain)	⚠️	✅	✅	✅

Who wins?

Approach	Performance	Stability	Security	Portability
Auto (eBPF)	⚠️	⚠️	⚠️	⚠️
Auto (toolchain)	⚠️	✅	✅	✅

The future

The future

- We asked, the Go team answered…
  - golang/go#63185 – Flight recording (released in Go 1.25!)

The future

- We asked, the Go team answered…
  - golang/go#63185 – Flight recording (released in Go 1.25!)
- Go Compile Time Instrumentation SIG
  - Tuesdays 12:30-1:30PM EST

The future

- We asked, the Go team answered…
  - golang/go#63185 – Flight recording (released in Go 1.25!)
- Go Compile Time Instrumentation SIG
  - Tuesdays 12:30-1:30PM EST
- ???

Final thoughts

Final thoughts

1. Instrumentation is helpful and *important*

Final thoughts

1. Instrumentation is helpful and *important*
2. Auto-instrumentation is EASY

Final thoughts

1. Instrumentation is helpful and *important*
2. Auto-instrumentation is EASY
3. What are you (👉) going to do next?

Thanks :)

@hannahkm

hannahkm.github.io

linkedin.com/in/hannah-kim24/

hannahs.kim@datadoghq.com

Gopher Icons: github.com/MariaLetta/free-gophers-pack