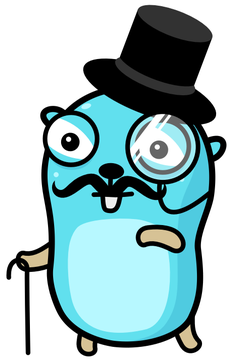
A HEALTHY PRODUCT
IN A HEALTHY BUSINESS



127.0.0.1/me

ȘTEFAN CÎRLIG

@VerySeriousCompany

github.com/**steevehook**

github.com/**go-workshops/ppp**

youtube.com/c/**SteveHook**

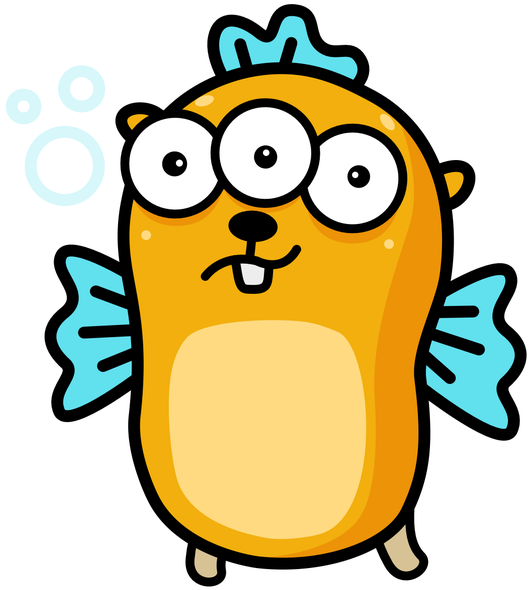
x.com/**steevehook**

t.me/**steevehook**

t.me/**go_workshop_qa**

- HOW MANY  ?
- OBSERVABILITY?
- IN PROD?
- EN / RO?

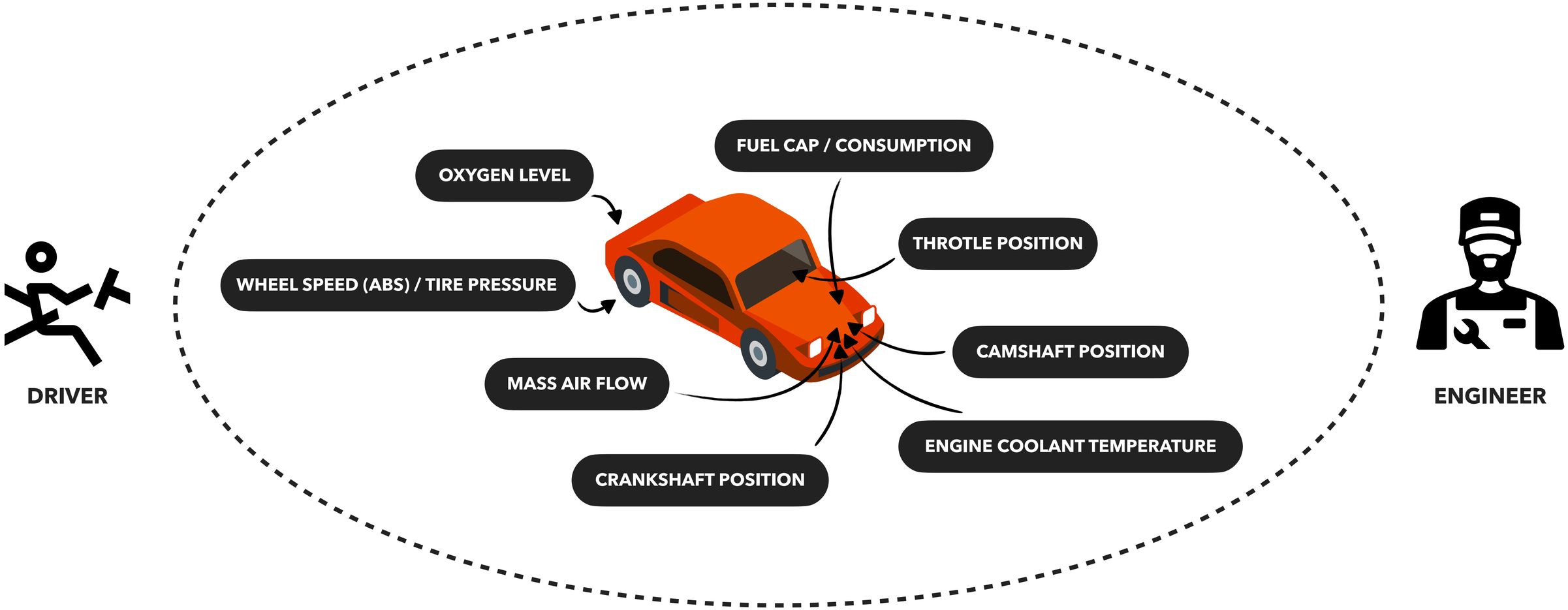
FORMAT



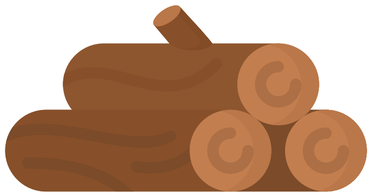
PREVENT

PROFILE

PERFECT



OBSERVABILITY



LOGGING

NO TRAILS, NO TALES

V1

```go
type createTodoRequestV1 struct {
    Title string `json:"title"`
}

func createTodoV1() func(http.ResponseWriter, *http.Request) {
    return func(w http.ResponseWriter, r *http.Request) {
        var req createTodoRequestV1
        err := json.NewDecoder(r.Body).Decode(&req)
        if err != nil {
            w.WriteHeader(http.StatusBadRequest)
            return
        }

        if req.Title == "" {
            w.WriteHeader(http.StatusBadRequest)
            return
        }

        w.WriteHeader(http.StatusOK)
    }
}
```

ERROR

ERROR

INFO

V1

Logs volume

No volume data.

Logs

Time Unique labels Wrap lines Prettify JSON Dedup None Exact Numbers Signature

Display results Newest first Oldest first

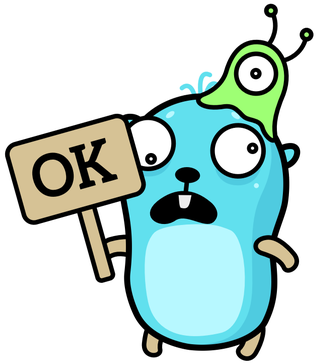
Line limit: 1000 Download

No logs found. Scan for older logs

Older logs

19:09:39 — 19:09:44

Start of range




```
curl -v localhost:8080/v2/todos
```

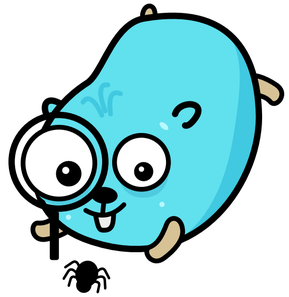
```
curl -v localhost:8080/v2/todos \
-H "Content-Type: application/json" \
-d '{"title": ""}'
```

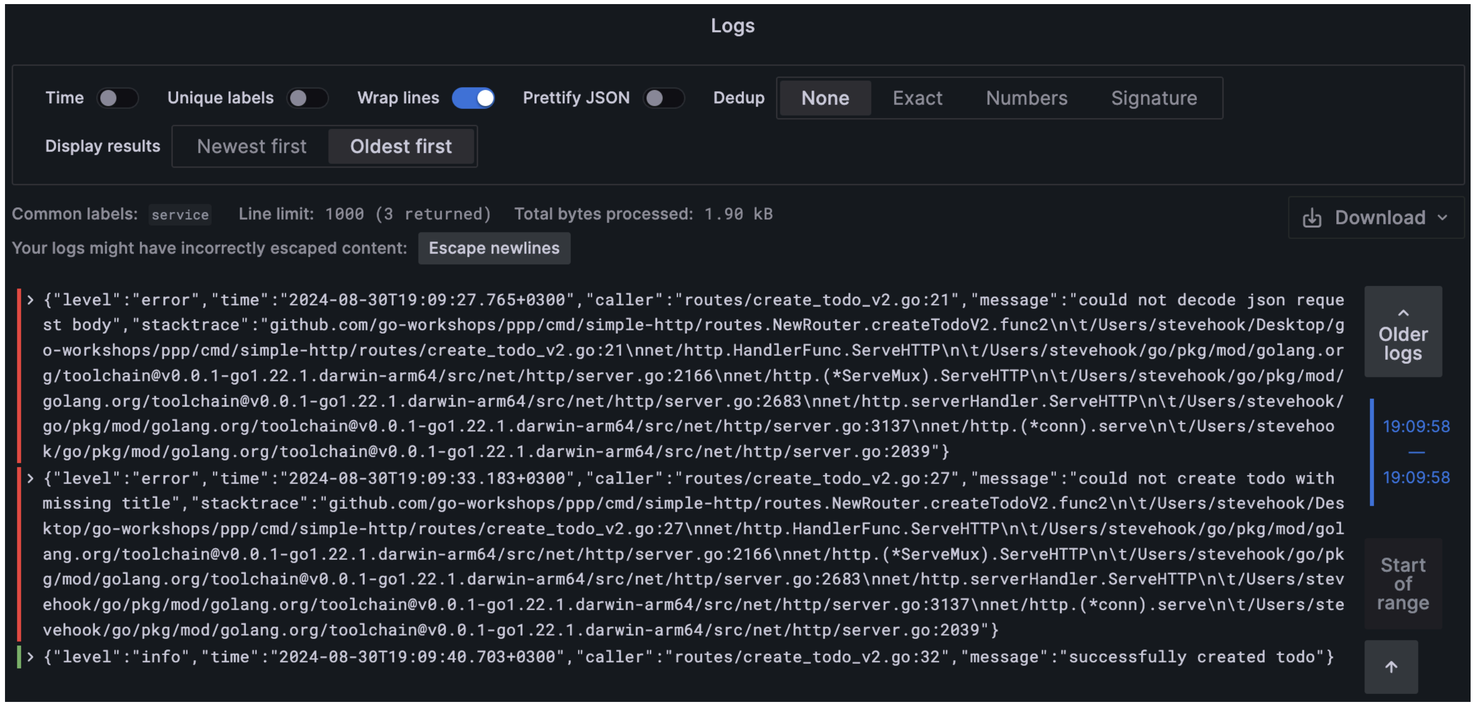
```
curl -v localhost:8080/v2/todos \
-H "Content-Type: application/json" \
-d '{"title": "Todo"}'
```

V2

```go
type createTodoRequestV2 struct {
    Title string `json:"title"`
}

func createTodoV2() func(http.ResponseWriter, *http.Request) {
    return func(w http.ResponseWriter, r *http.Request) {
        logger := sharedContext.Logger(r.Context())

        var req createTodoRequestV2
        err := json.NewDecoder(r.Body).Decode(&req)
        if err != nil {
            logger.Error("could not decode json request body")
            w.WriteHeader(http.StatusBadRequest)
            return
        }

        if req.Title == "" {
            logger.Error("could not create todo with missing title")
            w.WriteHeader(http.StatusBadRequest)
            return
        }

        logger.Info("successfully created todo")
        w.WriteHeader(http.StatusOK)
    }
}
```

V2






```
curl -v localhost:8080/v2/todos
```

```
curl -v localhost:8080/v2/todos \
-H "Content-Type: application/json" \
-d '{"title": ""}'
```

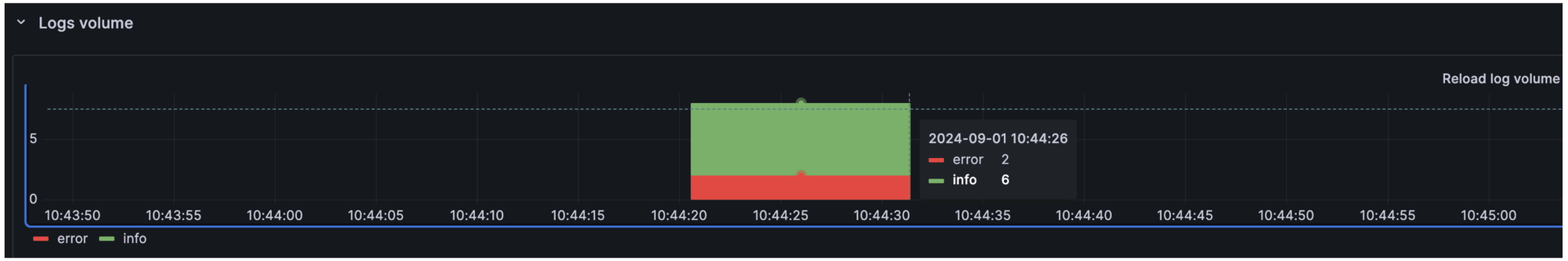
```
curl -v localhost:8080/v2/todos \
-H "Content-Type: application/json" \
-d '{"title": "Todo"}'
```

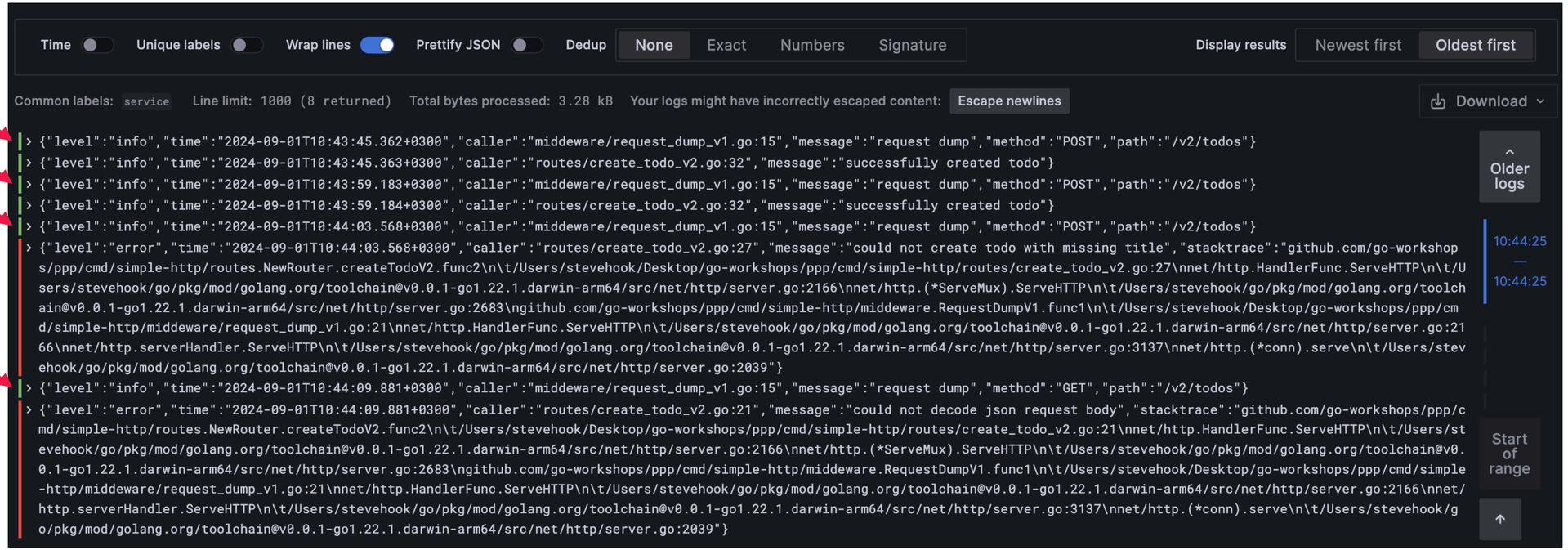
THIS COULD HAVE BEEN A DEBUG

V1

```go
func RequestDumpV1(h http.Handler) http.Handler {
    return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
        logger := sharedContext.Logger(r.Context())

        logger.Info(
            "request dump",
            zap.String("method", r.Method),
            zap.String("path", r.URL.Path),
        )

        h.ServeHTTP(w, r)
    })
}
```

V1

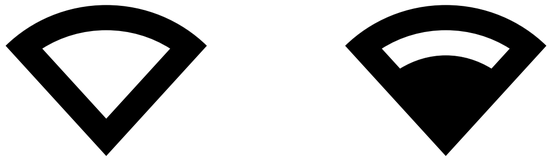

Logs volume
Reload log volume
5
0
2024-09-01 10:44:26
error 2
info 6
10:43:50
10:43:55
10:44:00
10:44:05
10:44:10
10:44:15
10:44:20
10:44:25
10:44:30
10:44:35
10:44:40
10:44:45
10:44:50
10:44:55
10:45:00
error
info


Time
Unique labels
Wrap lines
Prettify JSON
Dedup
None
Exact
Numbers
Signature
Display results
Newest first
Oldest first
Common labels: service
Line limit: 1000 (8 returned)
Total bytes processed: 3.28 kB
Your logs might have incorrectly escaped content:
Escape newlines
Download
Older logs
10:44:25 — 10:44:25
Start of range
{"level":"info","time":"2024-09-01T10:43:45.362+0300","caller":"middeware/request_dump_v1.go:15","message":"request dump","method":"POST","path":"/v2/todos"}
{"level":"info","time":"2024-09-01T10:43:45.363+0300","caller":"routes/create_todo_v2.go:32","message":"successfully created todo"}
{"level":"info","time":"2024-09-01T10:43:59.183+0300","caller":"middeware/request_dump_v1.go:15","message":"request dump","method":"POST","path":"/v2/todos"}
{"level":"info","time":"2024-09-01T10:43:59.184+0300","caller":"routes/create_todo_v2.go:32","message":"successfully created todo"}
{"level":"info","time":"2024-09-01T10:44:03.568+0300","caller":"middeware/request_dump_v1.go:15","message":"request dump","method":"POST","path":"/v2/todos"}
{"level":"error","time":"2024-09-01T10:44:03.568+0300","caller":"routes/create_todo_v2.go:27","message":"could not create todo with missing title","stacktrace":"github.com/go-workshops/ppp/cmd/simple-http/routes.NewRouter.createTodoV2.func2\n\t/Users/stevehook/Desktop/go-workshops/ppp/cmd/simple-http/routes/create_todo_v2.go:27\nnet/http.HandlerFunc.ServeHTTP\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2166\nnet/http.(*ServeMux).ServeHTTP\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2683\ngithub.com/go-workshops/ppp/cmd/simple-http/middeware.RequestDumpV1.func1\n\t/Users/stevehook/Desktop/go-workshops/ppp/cmd/simple-http/middeware/request_dump_v1.go:21\nnet/http.HandlerFunc.ServeHTTP\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2166\nnet/http.serverHandler.ServeHTTP\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:3137\nnet/http.(*conn).serve\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2039"}
{"level":"info","time":"2024-09-01T10:44:09.881+0300","caller":"middeware/request_dump_v1.go:15","message":"request dump","method":"GET","path":"/v2/todos"}
{"level":"error","time":"2024-09-01T10:44:09.881+0300","caller":"routes/create_todo_v2.go:21","message":"could not decode json request body","stacktrace":"github.com/go-workshops/ppp/cmd/simple-http/routes.NewRouter.createTodoV2.func2\n\t/Users/stevehook/Desktop/go-workshops/ppp/cmd/simple-http/routes/create_todo_v2.go:21\nnet/http.HandlerFunc.ServeHTTP\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2166\nnet/http.(*ServeMux).ServeHTTP\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2683\ngithub.com/go-workshops/ppp/cmd/simple-http/middeware.RequestDumpV1.func1\n\t/Users/stevehook/Desktop/go-workshops/ppp/cmd/simple-http/middeware/request_dump_v1.go:21\nnet/http.HandlerFunc.ServeHTTP\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2166\nnet/http.serverHandler.ServeHTTP\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:3137\nnet/http.(*conn).serve\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2039"}

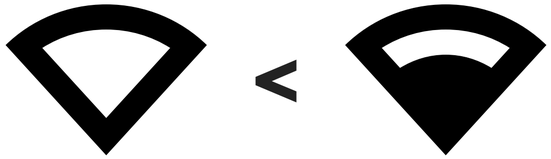
50%

NOISE

NOISE / SIGNAL RATIO



50% 50%



NOISE < SIGNAL

V2

```go
func RequestDumpV2(h http.Handler) http.Handler {
    return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
        logger := sharedContext.Logger(r.Context())

        logger.Debug(
            "request dump",
            zap.String("method", r.Method),
            zap.String("path", r.URL.Path),
        )

        h.ServeHTTP(w, r)
    })
}
```

V2

NOISE

Time | Unique labels | Wrap lines | Prettify JSON | Dedup: None | Exact | Numbers | Signature | Display results: Newest first | Oldest first

Common labels: service | Line limit: 1000 (8 returned) | Total bytes processed: 3.29 kB | Your logs might have incorrectly escaped content: Escape newlines | Download

> {"level":"debug","time":"2024-09-01T11:19:52.327+0300","caller":"middleware/request_dump_v2.go:15","message":"request dump","method":"POST","path":"/v2/todos"}
> {"level":"info","time":"2024-09-01T11:19:52.327+0300","caller":"routes/create_todo_v2.go:32","message":"successfully created todo"}
> {"level":"debug","time":"2024-09-01T11:19:54.999+0300","caller":"middleware/request_dump_v2.go:15","message":"request dump","method":"POST","path":"/v2/todos"}
> {"level":"info","time":"2024-09-01T11:19:54.999+0300","caller":"routes/create_todo_v2.go:32","message":"successfully created todo"}
> {"level":"debug","time":"2024-09-01T11:19:58.648+0300","caller":"middleware/request_dump_v2.go:15","message":"request dump","method":"POST","path":"/v2/todos"}
> {"level":"error","time":"2024-09-01T11:19:58.648+0300","caller":"routes/create_todo_v2.go:27","message":"could not create todo with missing title","stacktrace":"github.com/go-workshops/ppp/cmd/simple-http/routes.NewRouter.createTodoV2.func2\n\t/Users/stevehook/Desktop/go-workshops/ppp/cmd/simple-http/routes/create_todo_v2.go:27\nnet/http.HandlerFunc.ServeHTTP\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2166\nnet/http.(*ServeMux).ServeHTTP\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2683\ngithub.com/go-workshops/ppp/cmd/simple-http/middleware.RequestDumpV2.func1\n\t/Users/stevehook/Desktop/go-workshops/ppp/cmd/simple-http/middleware/request_dump_v2.go:21\nnet/http.HandlerFunc.ServeHTTP\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2166\nnet/http.serverHandler.ServeHTTP\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:3137\nnet/http.(*conn).serve\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2039"}
> {"level":"debug","time":"2024-09-01T11:20:01.548+0300","caller":"middleware/request_dump_v2.go:15","message":"request dump","method":"GET","path":"/v2/todos"}
> {"level":"error","time":"2024-09-01T11:20:01.548+0300","caller":"routes/create_todo_v2.go:21","message":"could not decode json request body","stacktrace":"github.com/go-workshops/ppp/cmd/simple-http/routes.NewRouter.createTodoV2.func2\n\t/Users/stevehook/Desktop/go-workshops/ppp/cmd/simple-http/routes/create_todo_v2.go:21\nnet/http.HandlerFunc.ServeHTTP\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2166\nnet/http.(*ServeMux).ServeHTTP\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2683\ngithub.com/go-workshops/ppp/cmd/simple-http/middleware.RequestDumpV2.func1\n\t/Users/stevehook/Desktop/go-workshops/ppp/cmd/simple-http/middleware/request_dump_v2.go:21\nnet/http.HandlerFunc.ServeHTTP\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2166\nnet/http.serverHandler.ServeHTTP\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:3137\nnet/http.(*conn).serve\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2039"}

Older logs

11:24:20 — 11:24:20

Start of range

V2

INFO

Common labels: service Line limit: 1000 (8 returned) Total bytes processed: 3.29 kB Your logs might have incorrectly escaped content: Escape newlines Download

> {"level":"info","time":"2024-09-01T11:19:52.327+0300","caller":"routes/create_todo_v2.go:32","message":"successfully created todo"}
> {"level":"info","time":"2024-09-01T11:19:54.999+0300","caller":"routes/create_todo_v2.go:32","message":"successfully created todo"}

Older logs

ERROR

Common labels: service Line limit: 1000 (8 returned) Total bytes processed: 3.29 kB Your logs might have incorrectly escaped content: Escape newlines Download

> {"level":"error","time":"2024-09-01T11:19:58.648+0300","caller":"routes/create_todo_v2.go:27","message":"could not create todo with missing title","stacktrace":"github.com/go-workshops/ppp/cmd/simple-http/routes.NewRouter.createTodoV2.func2\n\t/Users/stevehook/Desktop/go-workshops/ppp/cmd/simple-http/routes/create_todo_v2.go:27\nnet/http.HandlerFunc.ServeHTTP\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2166\nnet/http.(*ServeMux).ServeHTTP\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2683\ngithub.com/go-workshops/ppp/cmd/simple-http/middleware.RequestDumpV2.func1\n\t/Users/stevehook/Desktop/go-workshops/ppp/cmd/simple-http/middleware/request_dump_v2.go:21\nnet/http.HandlerFunc.ServeHTTP\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2166\nnet/http.serverHandler.ServeHTTP\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:3137\nnet/http.(*conn).serve\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2039"}
> {"level":"error","time":"2024-09-01T11:20:01.548+0300","caller":"routes/create_todo_v2.go:21","message":"could not decode json request body","stacktrace":"github.com/go-workshops/ppp/cmd/simple-http/routes.NewRouter.createTodoV2.func2\n\t/Users/stevehook/Desktop/go-workshops/ppp/cmd/simple-http/routes/create_todo_v2.go:21\nnet/http.HandlerFunc.ServeHTTP\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2166\nnet/http.(*ServeMux).ServeHTTP\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2683\ngithub.com/go-workshops/ppp/cmd/simple-http/middleware.RequestDumpV2.func1\n\t/Users/stevehook/Desktop/go-workshops/ppp/cmd/simple-http/middleware/request_dump_v2.go:21\nnet/http.HandlerFunc.ServeHTTP\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2166\nnet/http.serverHandler.ServeHTTP\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:3137\nnet/http.(*conn).serve\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2039"}

Older logs

11:24:20 — 11:24:20

Start of range

DEBUG

Common labels: service Line limit: 1000 (8 returned) Total bytes processed: 3.29 kB Your logs might have incorrectly escaped content: Escape newlines Download

> {"level":"debug","time":"2024-09-01T11:19:52.327+0300","caller":"middleware/request_dump_v2.go:15","message":"request dump","method":"POST","path":"/v2/todos"}
> {"level":"debug","time":"2024-09-01T11:19:54.999+0300","caller":"middleware/request_dump_v2.go:15","message":"request dump","method":"POST","path":"/v2/todos"}
> {"level":"debug","time":"2024-09-01T11:19:58.648+0300","caller":"middleware/request_dump_v2.go:15","message":"request dump","method":"POST","path":"/v2/todos"}
> {"level":"debug","time":"2024-09-01T11:20:01.548+0300","caller":"middleware/request_dump_v2.go:15","message":"request dump","method":"GET","path":"/v2/todos"}

Older logs

WHEN EVERYTHING IS AN ERROR

NOTHING IS REALLY AN ~~ERROR~~

V3

```go
func createTodoV3(svc todoCreator) func(http.ResponseWriter, *http.Request) {
    return func(w http.ResponseWriter, r *http.Request) {
        ctx := r.Context()
        logger := sharedContext.Logger(ctx)
        var req createTodoRequestV3
        if err := json.NewDecoder(r.Body).Decode(&req); err != nil {
            logger.Error("could not decode json request body")
            w.WriteHeader(http.StatusBadRequest)
            return
        }
        if req.Title == "" {
            logger.Error("could not create todo with missing title")
            w.WriteHeader(http.StatusBadRequest)
            return
        }
        if req.Description == "" {
            logger.Error("could not create todo with missing description")
            w.WriteHeader(http.StatusBadRequest)
            return
        }
        todo := models.Todo{
            ID:          fmt.Sprintf("%d", time.Now().UnixNano()),
            Title:       req.Title,
            Description: req.Description,
        }
        if err := svc.CreateTodo(ctx, todo); err != nil {
            logger.Error("could not create todo")
            w.WriteHeader(http.StatusInternalServerError)
            return
        }

        logger.Info("successfully created todo")
        w.WriteHeader(http.StatusOK)
    }
}
```

WARN

WARN

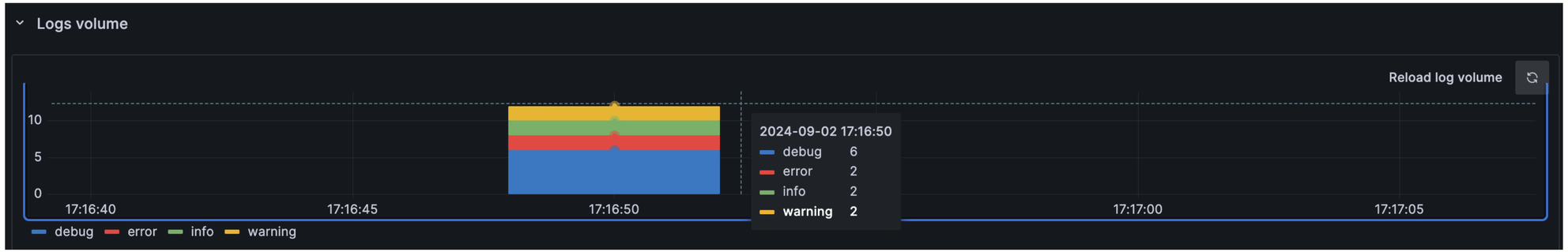
```go
type createTodoRequestV3 struct {
    Title       string `json:"title"`
    Description string `json:"description"`
}

type todoCreator interface {
    CreateTodo(ctx context.Context, todo models.Todo) error
}
```

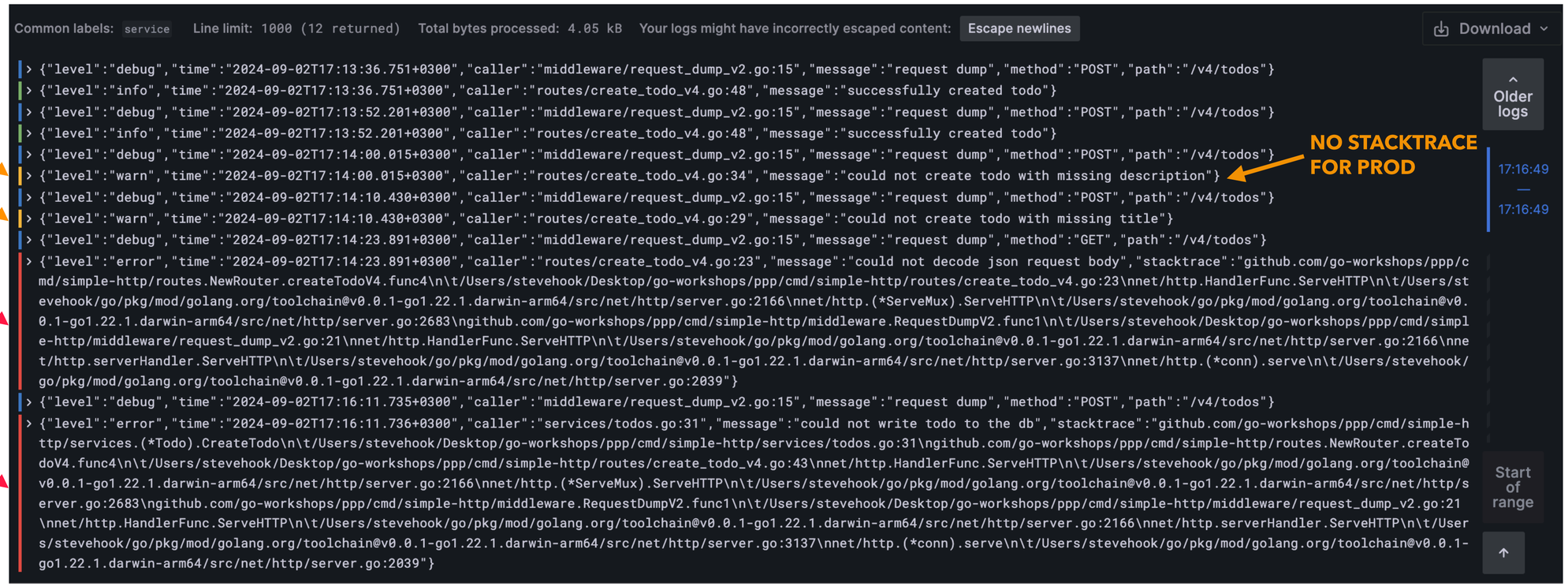
```go
type database interface {
    File(name string) db.FS
}

type Todo struct {
    db database
}

func NewTodo(db database) *Todo {
    return &Todo{
        db: db,
    }
}

func (t *Todo) CreateTodo(ctx context.Context, todo models.Todo) error {
    logger := sharedContext.Logger(ctx)

    err := json.NewEncoder(t.db.File(todo.ID + ".json")).Encode(todo)
    if err != nil {
        logger.Error("could not write todo to the db")
        return err
    }

    return nil
}
```

V4

```go
func createTodoV4(svc todoCreator) func(http.ResponseWriter, *http.Request) {
    return func(w http.ResponseWriter, r *http.Request) {
        ctx := r.Context()
        logger := sharedContext.Logger(ctx)
        var req createTodoRequestV4
        if err := json.NewDecoder(r.Body).Decode(&req); err != nil {
            logger.Error("could not decode json request body")
            w.WriteHeader(http.StatusBadRequest)
            return
        }
        if req.Title == "" {
            logger.Warn("could not create todo with missing title")
            w.WriteHeader(http.StatusBadRequest)
            return
        }
        if req.Description == "" {
            logger.Warn("could not create todo with missing description")
            w.WriteHeader(http.StatusBadRequest)
            return
        }

        todo := models.Todo{
            ID:          fmt.Sprintf("%d", time.Now().UnixNano()),
            Title:       req.Title,
            Description: req.Description,
        }
        if err := svc.CreateTodo(ctx, todo); err != nil {
            w.WriteHeader(http.StatusInternalServerError)
            return
        }

        logger.Info("successfully created todo")
        w.WriteHeader(http.StatusOK)
    }
}
```

V4


Logs volume
Reload log volume
10
5
0
17:16:40
17:16:45
17:16:50
17:17:00
17:17:05
2024-09-02 17:16:50
debug 6
error 2
info 2
warning 2
debug error info warning


Common labels: service
Line limit: 1000 (12 returned)
Total bytes processed: 4.05 kB
Your logs might have incorrectly escaped content:
Escape newlines
Download
Older logs
17:16:49
—
17:16:49
Start of range
NO STACKTRACE FOR PROD
5
2
> {"level":"debug","time":"2024-09-02T17:13:36.751+0300","caller":"middleware/request_dump_v2.go:15","message":"request dump","method":"POST","path":"/v4/todos"}
> {"level":"info","time":"2024-09-02T17:13:36.751+0300","caller":"routes/create_todo_v4.go:48","message":"successfully created todo"}
> {"level":"debug","time":"2024-09-02T17:13:52.201+0300","caller":"middleware/request_dump_v2.go:15","message":"request dump","method":"POST","path":"/v4/todos"}
> {"level":"info","time":"2024-09-02T17:13:52.201+0300","caller":"routes/create_todo_v4.go:48","message":"successfully created todo"}
> {"level":"debug","time":"2024-09-02T17:14:00.015+0300","caller":"middleware/request_dump_v2.go:15","message":"request dump","method":"POST","path":"/v4/todos"}
> {"level":"warn","time":"2024-09-02T17:14:00.015+0300","caller":"routes/create_todo_v4.go:34","message":"could not create todo with missing description"}
> {"level":"debug","time":"2024-09-02T17:14:10.430+0300","caller":"middleware/request_dump_v2.go:15","message":"request dump","method":"POST","path":"/v4/todos"}
> {"level":"warn","time":"2024-09-02T17:14:10.430+0300","caller":"routes/create_todo_v4.go:29","message":"could not create todo with missing title"}
> {"level":"debug","time":"2024-09-02T17:14:23.891+0300","caller":"middleware/request_dump_v2.go:15","message":"request dump","method":"GET","path":"/v4/todos"}
> {"level":"error","time":"2024-09-02T17:14:23.891+0300","caller":"routes/create_todo_v4.go:23","message":"could not decode json request body","stacktrace":"github.com/go-workshops/ppp/cmd/simple-http/routes.NewRouter.createTodoV4.func4\n\t/Users/stevehook/Desktop/go-workshops/ppp/cmd/simple-http/routes/create_todo_v4.go:23\nnet/http.HandlerFunc.ServeHTTP\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2166\nnet/http.(*ServeMux).ServeHTTP\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2683\ngithub.com/go-workshops/ppp/cmd/simple-http/middleware.RequestDumpV2.func1\n\t/Users/stevehook/Desktop/go-workshops/ppp/cmd/simple-http/middleware/request_dump_v2.go:21\nnet/http.HandlerFunc.ServeHTTP\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2166\nnet/http.serverHandler.ServeHTTP\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:3137\nnet/http.(*conn).serve\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2039"}
> {"level":"debug","time":"2024-09-02T17:16:11.735+0300","caller":"middleware/request_dump_v2.go:15","message":"request dump","method":"POST","path":"/v4/todos"}
> {"level":"error","time":"2024-09-02T17:16:11.736+0300","caller":"services/todos.go:31","message":"could not write todo to the db","stacktrace":"github.com/go-workshops/ppp/cmd/simple-http/services.(*Todo).CreateTodo\n\t/Users/stevehook/Desktop/go-workshops/ppp/cmd/simple-http/services/todos.go:31\ngithub.com/go-workshops/ppp/cmd/simple-http/routes.NewRouter.createTodoV4.func4\n\t/Users/stevehook/Desktop/go-workshops/ppp/cmd/simple-http/routes/create_todo_v4.go:43\nnet/http.HandlerFunc.ServeHTTP\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2166\nnet/http.(*ServeMux).ServeHTTP\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2683\ngithub.com/go-workshops/ppp/cmd/simple-http/middleware.RequestDumpV2.func1\n\t/Users/stevehook/Desktop/go-workshops/ppp/cmd/simple-http/middleware/request_dump_v2.go:21\nnet/http.HandlerFunc.ServeHTTP\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2166\nnet/http.serverHandler.ServeHTTP\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:3137\nnet/http.(*conn).serve\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2039"}

THAT ~~INFO~~ COULD HAVE BEEN A //COMMENT

V1

```go
type todoUpdater interface {
    UpdateTodo(ctx context.Context, todo models.Todo) error
}

func updateTodoV1(svc todoUpdater) func(http.ResponseWriter, *http.Request) {
    return func(w http.ResponseWriter, r *http.Request) {
        ctx := r.Context()
        logger := sharedContext.Logger(ctx)

        logger.Info("decoding update todo request")
        var req updateTodoRequestV1
        if err := json.NewDecoder(r.Body).Decode(&req); err != nil {
            logger.Error("could not decode json request body")
            w.WriteHeader(http.StatusBadRequest)
            return
        }
        logger.Info("validating update todo request")
        if !req.Validate(r, w) {
            return
        }

        logger.Info("updating todo model in the database")
        todo := models.Todo{ID: req.ID, Title: req.Title, Description: req.Description}
        if err := svc.UpdateTodo(ctx, todo); err != nil {
            w.WriteHeader(http.StatusInternalServerError)
            return
        }

        logger.Info("successfully updated todo")
        w.WriteHeader(http.StatusOK)
    }
}
```

```go
type updateTodoRequestV1 struct {
    ID          string `json:"id"`
    Title       string `json:"title"`
    Description string `json:"description"`
}

func (req updateTodoRequestV1) Validate(r *http.Request, w http.ResponseWriter) bool {
    logger := sharedContext.Logger(r.Context())
    if req.ID == "" {
        logger.Warn("could not update todo with missing id")
        w.WriteHeader(http.StatusBadRequest)
        return false
    }
    if req.Title == "" && req.Description == "" {
        logger.Warn("could not update todo with missing title and description")
        w.WriteHeader(http.StatusBadRequest)
        return false
    }

    return true
}
```

V2

```go
func updateTodoV2(svc todoUpdater) func(http.ResponseWriter, *http.Request) {
    return func(w http.ResponseWriter, r *http.Request) {
        ctx := r.Context()
        logger := sharedContext.Logger(ctx)

        logger.Debug("decoding update todo request")
        var req updateTodoRequestV2
        if err := json.NewDecoder(r.Body).Decode(&req); err != nil {
            logger.Error("could not decode json request body")
            w.WriteHeader(http.StatusBadRequest)
            return
        }

        logger.Debug("validating update todo request")
        if !req.Validate(r, w) {
            return
        }

        logger.Debug("updating todo model in the database")
        todo := models.Todo{ID: req.ID, Title: req.Title, Description: req.Description}
        if err := svc.UpdateTodo(ctx, todo); err != nil {
            w.WriteHeader(http.StatusInternalServerError)
            return
        }

        logger.Info("successfully updated todo")
        w.WriteHeader(http.StatusOK)
    }
}
```

V3

0

```go
func updateTodoV3(svc todoUpdater) func(http.ResponseWriter, *http.Request) {
    return func(w http.ResponseWriter, r *http.Request) {
        ctx := r.Context()
        logger := sharedContext.Logger(ctx)

        // decoding update todo request
        var req updateTodoRequestV3
        if err := json.NewDecoder(r.Body).Decode(&req); err != nil {
            logger.Error("could not decode json request body")
            w.WriteHeader(http.StatusBadRequest)
            return
        }

        // validating update todo request
        if !req.Validate(r, w) {
            return
        }

        // updating todo model in the database
        todo := models.Todo{ID: req.ID, Title: req.Title, Description: req.Description}
        if err := svc.UpdateTodo(ctx, todo); err != nil {
            w.WriteHeader(http.StatusInternalServerError)
            return
        }

        logger.Info("successfully updated todo")
        w.WriteHeader(http.StatusOK)
    }
}
```

```go
func updateTodoV3(svc todoUpdater) func(http.ResponseWriter, *http.Request) {
    return func(w http.ResponseWriter, r *http.Request) {
        ctx := r.Context()
        logger := sharedContext.Logger(ctx)

        var req updateTodoRequestV3
        if err := json.NewDecoder(r.Body).Decode(&req); err != nil {
            logger.Error("could not decode json request body")
            w.WriteHeader(http.StatusBadRequest)
            return
        }

        if !req.Validate(r, w) {
            return
        }

        todo := models.Todo{ID: req.ID, Title: req.Title, Description: req.Description}
        if err := svc.UpdateTodo(ctx, todo); err != nil {
            w.WriteHeader(http.StatusInternalServerError)
            return
        }

        logger.Info("successfully updated todo")
        w.WriteHeader(http.StatusOK)
    }
}
```

NO CONTEXT, NO ~~CONTENT~~

V3

ERROR CONTEXT

TODO CONTEXT

```go
func updateTodoV3(svc todoUpdater) func(http.ResponseWriter, *http.Request) {
    return func(w http.ResponseWriter, r *http.Request) {
        ctx := r.Context()
        logger := sharedContext.Logger(ctx)

        var req updateTodoRequestV3
        if err := json.NewDecoder(r.Body).Decode(&req); err != nil {
            logger.Error("could not decode json request body")
            w.WriteHeader(http.StatusBadRequest)
            return
        }

        if !req.Validate(r, w) {
            return
        }

        todo := models.Todo{ID: req.ID, Title: req.Title, Description: req.Description}
        if err := svc.UpdateTodo(ctx, todo); err != nil {
            w.WriteHeader(http.StatusInternalServerError)
            return
        }

        logger.Info("successfully updated todo")
        w.WriteHeader(http.StatusOK)
    }
}
```

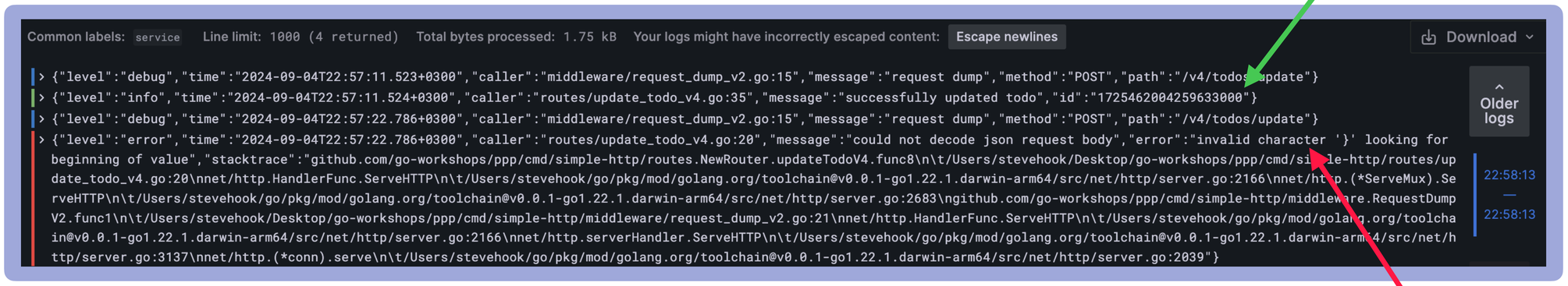
V4

```go
func updateTodoV4(svc todoUpdater) func(http.ResponseWriter, *http.Request) {
    return func(w http.ResponseWriter, r *http.Request) {
        ctx := r.Context()
        logger := sharedContext.Logger(ctx)

        var req updateTodoRequestV4
        if err := json.NewDecoder(r.Body).Decode(&req); err != nil {
            logger.Error("could not decode json request body", zap.Error(err))
            w.WriteHeader(http.StatusBadRequest)
            return
        }

        if !req.Validate(r, w) {
            return
        }

        todo := models.Todo{ID: req.ID, Title: req.Title, Description: req.Description}
        if err := svc.UpdateTodo(ctx, todo); err != nil {
            w.WriteHeader(http.StatusInternalServerError)
            return
        }

        logger.Info("successfully updated todo", zap.String("id", todo.ID))
        w.WriteHeader(http.StatusOK)
    }
}
```

V3

V4

TODO CONTEXT

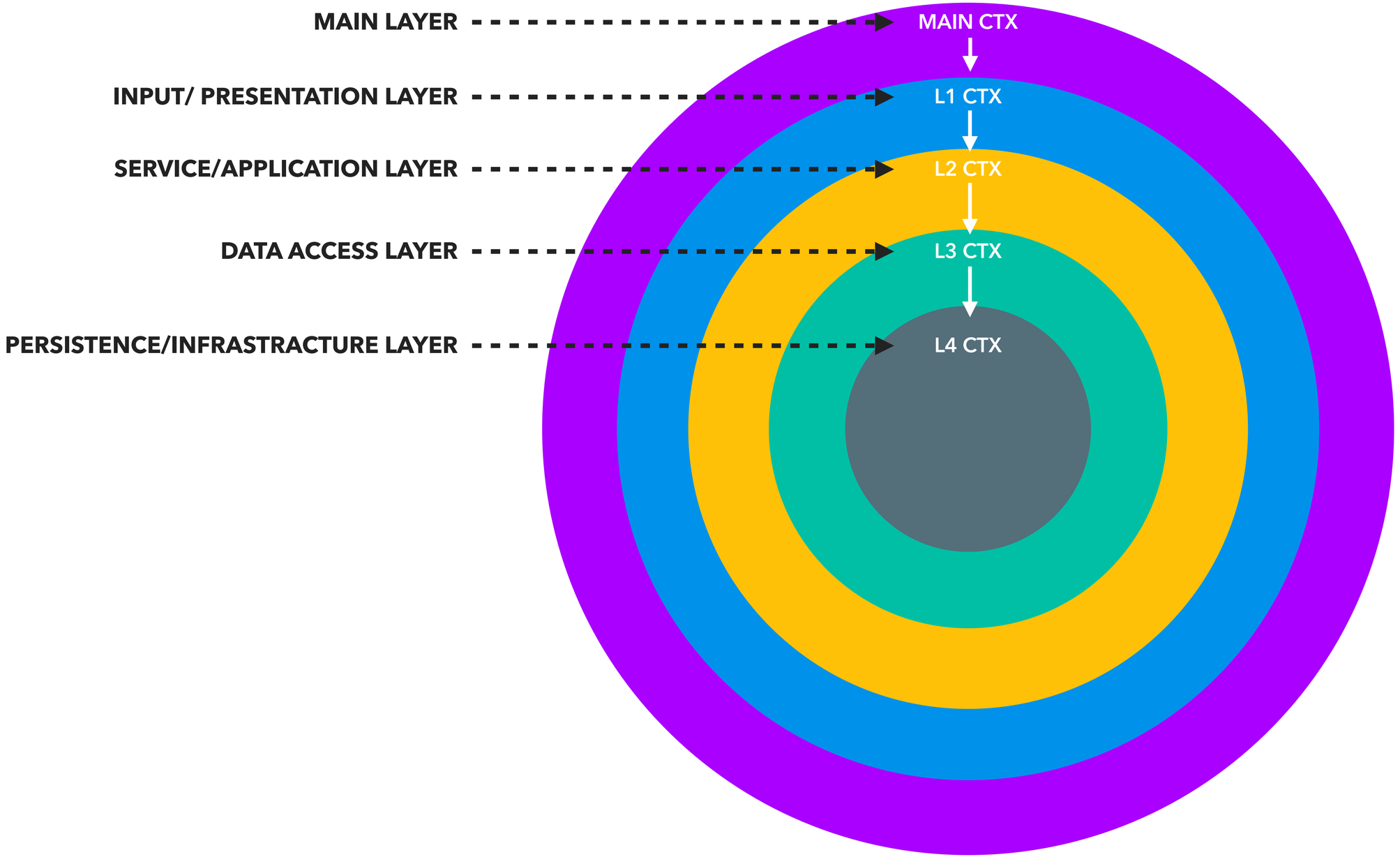


ERROR CONTEXT

COMPOSE (CAREFULLY)

LAYERS


"OGRES ARE LIKE ONIONS.
OGRES HAVE LAYERS, ONIONS HAVE LAYERS..."
"YOU GET IT? WE BOTH HAVE LAYERS."


MAIN LAYER
INPUT/ PRESENTATION LAYER
SERVICE/APPLICATION LAYER
DATA ACCESS LAYER
PERSISTENCE/INFRASTRACTURE LAYER
MAIN CTX
L1 CTX
L2 CTX
L3 CTX
L4 CTX

CTX IS KEY

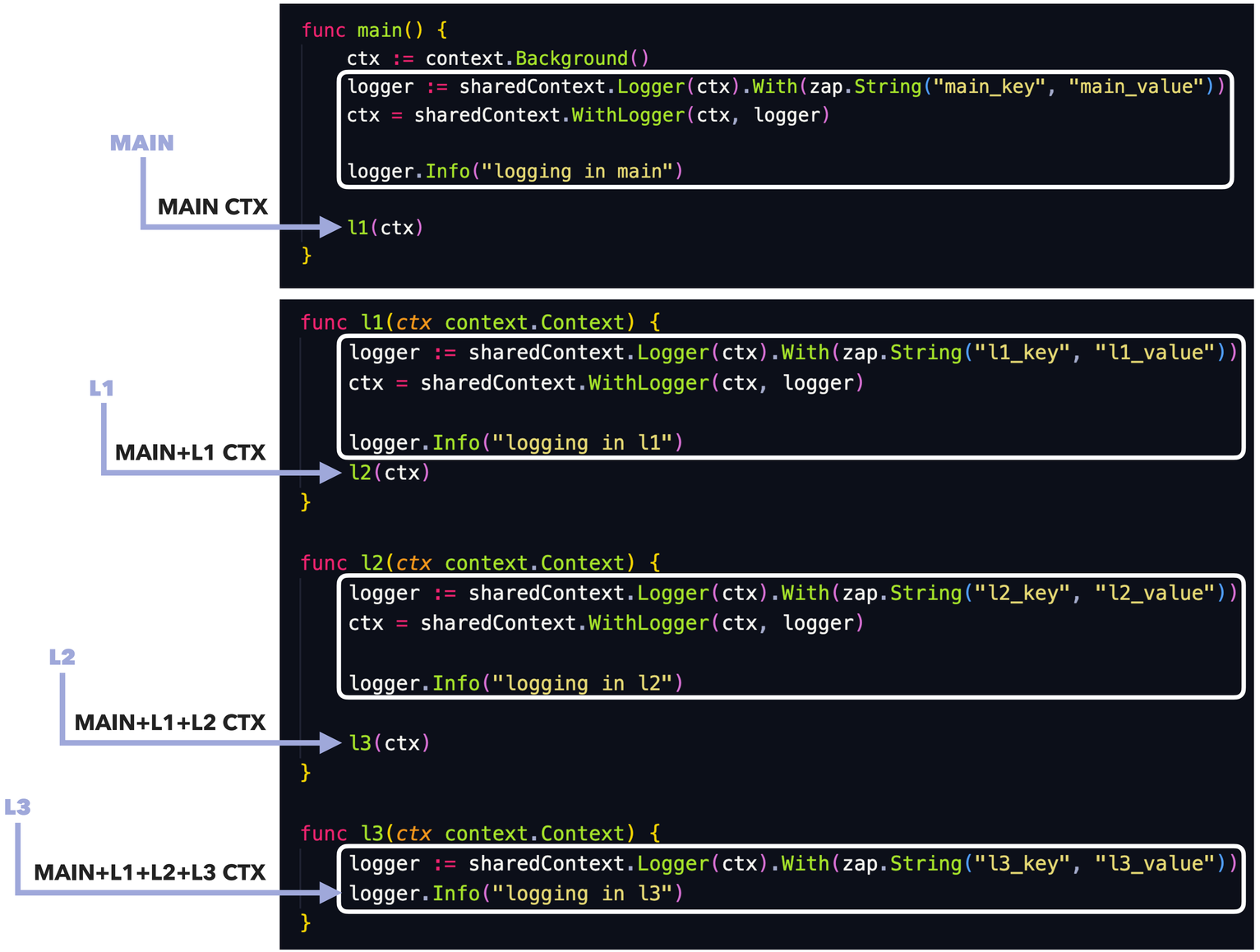
SET LOGGER TO **CTX**

GET LOGGER FROM **CTX**

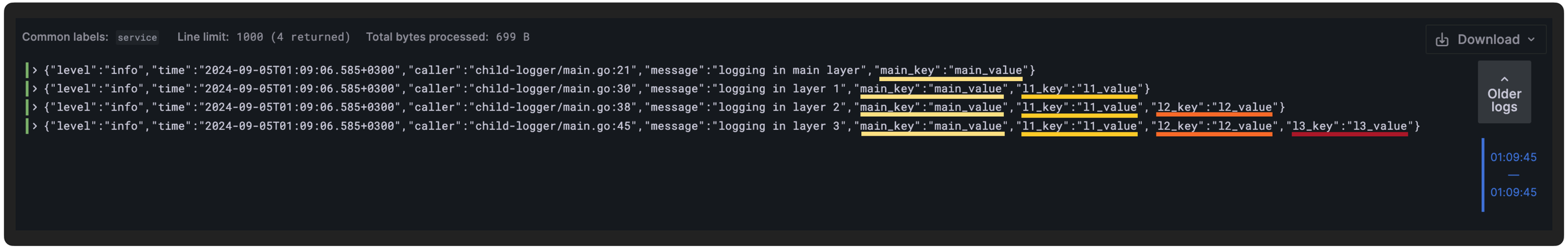
SHARED CONTEXT

```go
type key int

const (
    loggerCtxKey key = iota
)

func WithLogger(ctx context.Context, logger *zap.Logger) context.Context {
    return context.WithValue(ctx, loggerCtxKey, logger)
}

func Logger(ctx context.Context) *zap.Logger {
    logger, ok := ctx.Value(loggerCtxKey).(*zap.Logger)
    if ok {
        return logger
    }

    return logging.GetLogger()
}
```

LAYERS AGAIN



LAYERS AGAIN



V4

```go
func updateTodoV4(svc todoUpdater) func(http.ResponseWriter, *http.Request) {
    return func(w http.ResponseWriter, r *http.Request) {
        ctx := r.Context()
        logger := sharedContext.Logger(ctx)

        var req updateTodoRequestV4
        if err := json.NewDecoder(r.Body).Decode(&req); err != nil {
            logger.Error("could not decode json request body", zap.Error(err))
            w.WriteHeader(http.StatusBadRequest)
            return
        }

        if !req.Validate(r, w) {
            return
        }

        todo := models.Todo{ID: req.ID, Title: req.Title, Description: req.Description}
        if err := svc.UpdateTodo(ctx, todo); err != nil {
            w.WriteHeader(http.StatusInternalServerError)
            return
        }

        logger.Info("successfully updated todo", zap.String("id", todo.ID))
        w.WriteHeader(http.StatusOK)
    }
}
```

MISSING TODO CONTEXT

V5

```go
func updateTodoV5(svc todoUpdater) func(http.ResponseWriter, *http.Request) {
    return func(w http.ResponseWriter, r *http.Request) {
        ctx := r.Context()
        logger := sharedContext.Logger(ctx)

        var req updateTodoRequestV5
        if err := json.NewDecoder(r.Body).Decode(&req); err != nil {
            logger.Error("could not decode json request body", zap.Error(err))
            w.WriteHeader(http.StatusBadRequest)
            return
        }

        if !req.Validate(r, w) {
            return
        }

        logger = logger.With(zap.String("id", req.ID))
        ctx = sharedContext.WithLogger(ctx, logger.With(zap.String("id", req.ID)))
        todo := models.Todo{ID: req.ID, Title: req.Title, Description: req.Description}
        if err := svc.UpdateTodo(ctx, todo); err != nil {
            w.WriteHeader(http.StatusInternalServerError)
            return
        }

        logger.Info("successfully updated todo", zap.String("id", todo.ID))
        w.WriteHeader(http.StatusOK)
    }
}
```

V4

Common labels: service Line limit: 1000 (2 returned) Total bytes processed: 1.58 kB Your logs might have incorrectly escaped content: Escape newlines Download

> {"level":"debug","time":"2024-09-05T15:20:08.572+0300","caller":"middleware/request_dump_v2.go:15","message":"request dump","method":"POST","path":"/v4/todos/update"}

> {"level":"error","time":"2024-09-05T15:20:08.572+0300","caller":"services/todos.go:46","message":"could not read todo from the db","error":"open .db/1725462004259633001.json: no such file or directory","stacktrace":"github.com/go-workshops/ppp/cmd/simple-http/services.(*Todo).UpdateTodo\n\t/Users/stevehook/Desktop/go-workshops/ppp/cmd/simple-http/services/todos.go:46\ngithub.com/go-workshops/ppp/cmd/simple-http/routes.NewRouter.updateTodoV4.func8\n\t/Users/stevehook/Desktop/go-workshops/ppp/cmd/simple-http/routes/update_todo_v4.go:30\nnet/http.HandlerFunc.ServeHTTP\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2166\nnet/http.(*ServeMux).ServeHTTP\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2683\ngithub.com/go-workshops/ppp/cmd/simple-http/middleware.RequestDumpV2.func1\n\t/Users/stevehook/Desktop/go-workshops/ppp/cmd/simple-http/middleware/request_dump_v2.go:21\nnet/http.HandlerFunc.ServeHTTP\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2166\nnet/http.serverHandler.ServeHTTP\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:3137\nnet/http.(*conn).serve\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2039"}

Older logs

15:23:55 — 15:23:55

V5

TODO CONTEXT

Common labels: service Line limit: 1000 (2 returned) Total bytes processed: 1.60 kB Your logs might have incorrectly escaped content: Escape newlines Download

> {"level":"debug","time":"2024-09-05T15:19:32.081+0300","caller":"middleware/request_dump_v2.go:15","message":"request dump","method":"POST","path":"/v5/todos/update"}

> {"level":"error","time":"2024-09-05T15:19:32.082+0300","caller":"services/todos.go:46","message":"could not read todo from the db","id":"1725462004259633001","error":"open .db/1725462004259633001.json: no such file or directory","stacktrace":"github.com/go-workshops/ppp/cmd/simple-http/services.(*Todo).UpdateTodo\n\t/Users/stevehook/Desktop/go-workshops/ppp/cmd/simple-http/services/todos.go:46\ngithub.com/go-workshops/ppp/cmd/simple-http/routes.NewRouter.updateTodoV5.func9\n\t/Users/stevehook/Desktop/go-workshops/ppp/cmd/simple-http/routes/update_todo_v5.go:32\nnet/http.HandlerFunc.ServeHTTP\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2166\nnet/http.(*ServeMux).ServeHTTP\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2683\ngithub.com/go-workshops/ppp/cmd/simple-http/middleware.RequestDumpV2.func1\n\t/Users/stevehook/Desktop/go-workshops/ppp/cmd/simple-http/middleware/request_dump_v2.go:21\nnet/http.HandlerFunc.ServeHTTP\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2166\nnet/http.serverHandler.ServeHTTP\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:3137\nnet/http.(*conn).serve\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2039"}

Older logs

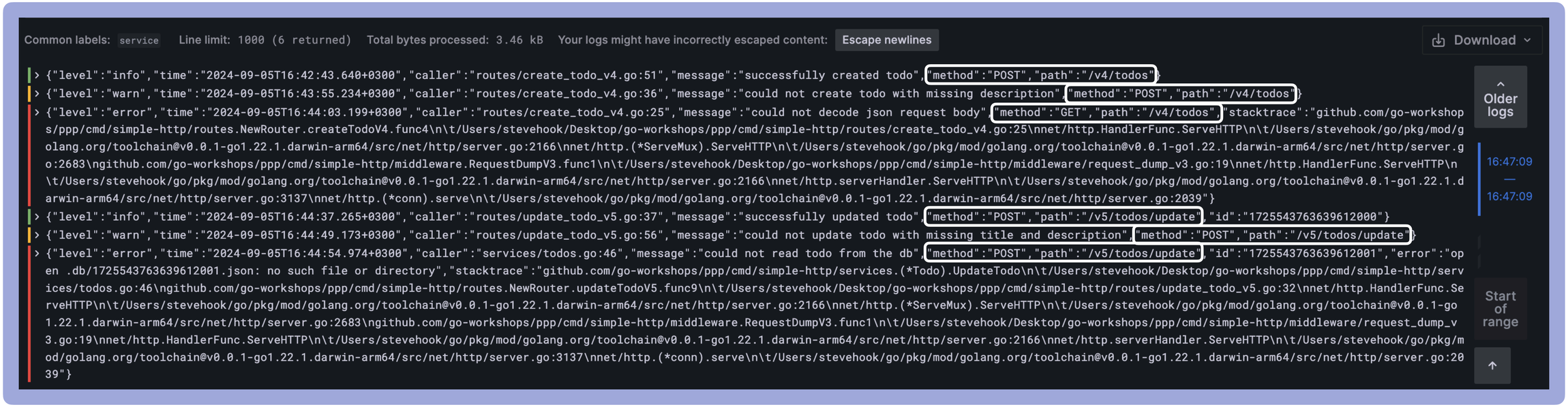
15:30:30 — 15:30:30

REDUCE MORE NOISE BY USING CONTEXT?

V3

```go
func RequestDumpV3(h http.Handler) http.Handler {
    return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
        ctx := r.Context()
        logger := sharedContext.Logger(ctx).With(
            zap.String("method", r.Method),
            zap.String("path", r.URL.Path),
        )

        h.ServeHTTP(w, r.WithContext(sharedContext.WithLogger(ctx, logger)))
    })
}
```

V3


Common labels: service
Line limit: 1000 (6 returned)
Total bytes processed: 3.46 kB
Your logs might have incorrectly escaped content:
Escape newlines
Download
{"level":"info","time":"2024-09-05T16:42:43.640+0300","caller":"routes/create_todo_v4.go:51","message":"successfully created todo","method":"POST","path":"/v4/todos"}
{"level":"warn","time":"2024-09-05T16:43:55.234+0300","caller":"routes/create_todo_v4.go:36","message":"could not create todo with missing description","method":"POST","path":"/v4/todos"}
{"level":"error","time":"2024-09-05T16:44:03.199+0300","caller":"routes/create_todo_v4.go:25","message":"could not decode json request body","method":"GET","path":"/v4/todos","stacktrace":"github.com/go-workshops/ppp/cmd/simple-http/routes.NewRouter.createTodoV4.func4\n\t/Users/stevehook/Desktop/go-workshops/ppp/cmd/simple-http/routes/create_todo_v4.go:25\nnet/http.HandlerFunc.ServeHTTP\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2166\nnet/http.(*ServeMux).ServeHTTP\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2683\ngithub.com/go-workshops/ppp/cmd/simple-http/middleware.RequestDumpV3.func1\n\t/Users/stevehook/Desktop/go-workshops/ppp/cmd/simple-http/middleware/request_dump_v3.go:19\nnet/http.HandlerFunc.ServeHTTP\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2166\nnet/http.serverHandler.ServeHTTP\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:3137\nnet/http.(*conn).serve\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2039"}
{"level":"info","time":"2024-09-05T16:44:37.265+0300","caller":"routes/update_todo_v5.go:37","message":"successfully updated todo","method":"POST","path":"/v5/todos/update","id":"1725543763639612000"}
{"level":"warn","time":"2024-09-05T16:44:49.173+0300","caller":"routes/update_todo_v5.go:56","message":"could not update todo with missing title and description","method":"POST","path":"/v5/todos/update"}
{"level":"error","time":"2024-09-05T16:44:54.974+0300","caller":"services/todos.go:46","message":"could not read todo from the db","method":"POST","path":"/v5/todos/update","id":"1725543763639612001","error":"open .db/1725543763639612001.json: no such file or directory","stacktrace":"github.com/go-workshops/ppp/cmd/simple-http/services.(*Todo).UpdateTodo\n\t/Users/stevehook/Desktop/go-workshops/ppp/cmd/simple-http/services/todos.go:46\ngithub.com/go-workshops/ppp/cmd/simple-http/routes.NewRouter.updateTodoV5.func9\n\t/Users/stevehook/Desktop/go-workshops/ppp/cmd/simple-http/routes/update_todo_v5.go:32\nnet/http.HandlerFunc.ServeHTTP\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2166\nnet/http.(*ServeMux).ServeHTTP\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2683\ngithub.com/go-workshops/ppp/cmd/simple-http/middleware.RequestDumpV3.func1\n\t/Users/stevehook/Desktop/go-workshops/ppp/cmd/simple-http/middleware/request_dump_v3.go:19\nnet/http.HandlerFunc.ServeHTTP\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2166\nnet/http.serverHandler.ServeHTTP\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:3137\nnet/http.(*conn).serve\n\t/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2039"}
Older logs
16:47:09
—
16:47:09
Start of range

-50% NOISE

ENV CONTEXT

`debug.ReadBuildInfo`

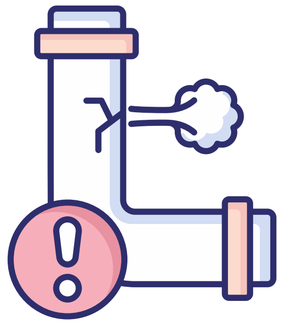
```go
var (
    revision  string
    buildTime string
)

func main() {
    logger := logging.GetLogger().With(
        zap.String("revision", revision),
        zap.String("build_time", buildTime),
    )
    ctx := sharedContext.WithLogger(context.Background(), logger)

    router := http.NewServeMux()
    router.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {
        logger.Info("hello world")
    })

    srv := &http.Server{
        Addr:    ":8080",
        Handler: router,
        BaseContext: func(net.Listener) context.Context {
            return ctx
        },
    }

    log.Fatalln(srv.ListenAndServe())
}
```

```
go build -ldflags " \
-X 'main.revision=$(git rev-parse --short HEAD)' \
-X 'main.buildTime=$(date +%s000000000)'" \
-o bin/env-context playground/env-context/main.go
```

ENV CONTEXT

Common labels: `service` Line limit: 1000 (4 returned) Total bytes processed: 3.33 kB Your logs might have incorrectly escaped content: Remove escaping

> {"level":"info","time":"2024-09-05T18:24:41.357+0300","caller":"routes/create_todo_v4.go:51","message":"successfully created todo","revision":"c3e6114","build_time":"2024-09-05T18:24:31.000+0300","method":"POST","path":"/v4/todos"}
> {"level":"error","time":"2024-09-05T18:24:47.170+0300","caller":"routes/create_todo_v4.go:25","message":"could not decode json request body","revision":"c3e6114","build_time":"2024-09-05T18:24:31.000+0300","method":"GET","path":"/v4/todos","stacktrace":"github.com/go-workshops/ppp/cmd/simple-http/routes.NewRouter.createTodoV4.func4
/Users/stevehook/Desktop/go-workshops/ppp/cmd/simple-http/routes/create_todo_v4.go:25
net/http.HandlerFunc.ServeHTTP
/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2166
net/http.(*ServeMux).ServeHTTP
/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2683
github.com/go-workshops/ppp/cmd/simple-http/middleware.RequestDumpV3.func1
/Users/stevehook/Desktop/go-workshops/ppp/cmd/simple-http/middleware/request_dump_v3.go:19
net/http.HandlerFunc.ServeHTTP
/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2166
net/http.serverHandler.ServeHTTP
/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:3137
net/http.(*conn).serve
/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2039"}
> {"level":"info","time":"2024-09-05T18:24:51.554+0300","caller":"routes/update_todo_v5.go:37","message":"successfully updated todo","revision":"c3e6114","build_time":"2024-09-05T18:24:31.000+0300","method":"POST","path":"/v5/todos/update","id":"1725543763639612000"}
> {"level":"error","time":"2024-09-05T18:24:57.382+0300","caller":"services/todos.go:46","message":"could not read todo from the db","revision":"c3e6114","build_time":"2024-09-05T18:24:31.000+0300","method":"POST","path":"/v5/todos/update","id":"1725543763639612001","error":"open .db/1725543763639612001.json: no such file or directory","stacktrace":"github.com/go-workshops/ppp/cmd/simple-http/services.(*Todo).UpdateTodo
/Users/stevehook/Desktop/go-workshops/ppp/cmd/simple-http/services/todos.go:46
github.com/go-workshops/ppp/cmd/simple-http/routes.NewRouter.updateTodoV5.func9
/Users/stevehook/Desktop/go-workshops/ppp/cmd/simple-http/routes/update_todo_v5.go:32
net/http.HandlerFunc.ServeHTTP
/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2166
net/http.(*ServeMux).ServeHTTP
/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2683
github.com/go-workshops/ppp/cmd/simple-http/middleware.RequestDumpV3.func1
/Users/stevehook/Desktop/go-workshops/ppp/cmd/simple-http/middleware/request_dump_v3.go:19
net/http.HandlerFunc.ServeHTTP
/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2166
net/http.serverHandler.ServeHTTP
/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:3137
net/http.(*conn).serve
/Users/stevehook/go/pkg/mod/golang.org/toolchain@v0.0.1-go1.22.1.darwin-arm64/src/net/http/server.go:2039"}



LEAKS

COMPOSITION

```go
func main() {
    ctx := context.Background()
    logger := sharedContext.Logger(ctx)

    // simulate a streaming service
    for {
        id := time.Now().UnixNano()
        logger = logger.With(zap.Int64("id", id))
        worker(sharedContext.WithLogger(ctx, logger))
    }
}

func worker(ctx context.Context) {
    logger := sharedContext.Logger(ctx)
    logger.Info("worker started")

    // simulate some work
    time.Sleep(1 * time.Second)

    logger.Info("worker finished")
}
```

CONSISTENT {FORMAT}

CONSOLE

DEVELOPMENT

JSON

PRODUCTION

A RECOVER A DAY
KEEPS THE PANIC AWAY

```go
func createPanic() func(http.ResponseWriter, *http.Request) {
    return func(w http.ResponseWriter, r *http.Request) {
        panic("this will panic")
    }
}
```

UNRECOVERED

HTTP ERROR LOGGER

```go
srv := &http.Server{
    Addr:    ":8080",
    Handler: router,
    BaseContext: func(net.Listener) context.Context {
        return ctx
    },
    ErrorLog: logging.HTTPErrorLogger(),
}
```

```go
func HTTPErrorLogger() *log.Logger {
    return log.New(&httpErrorLogger{logger: GetLogger()}, "", 0)
}

type httpErrorLogger struct {
    logger *zap.Logger
}

func (l *httpErrorLogger) Write(p []byte) (n int, err error) {
    l.logger.Error(string(p))
    return len(p), nil
}
```

RECOVERED

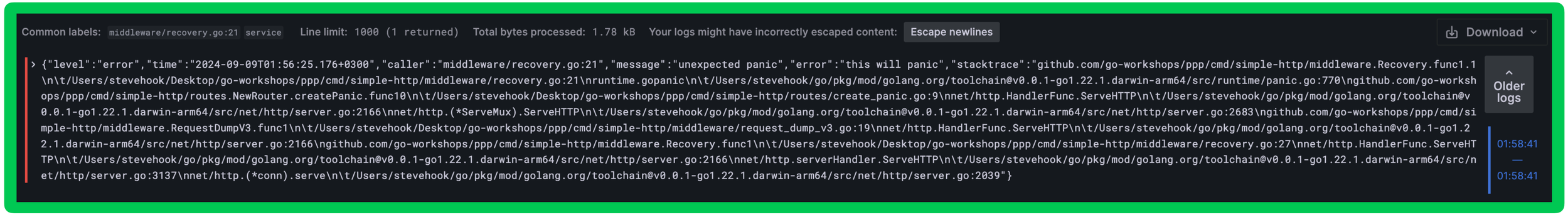
ERROR MESSAGE + STACK TRACE

RECOVERY MIDDLEWARE

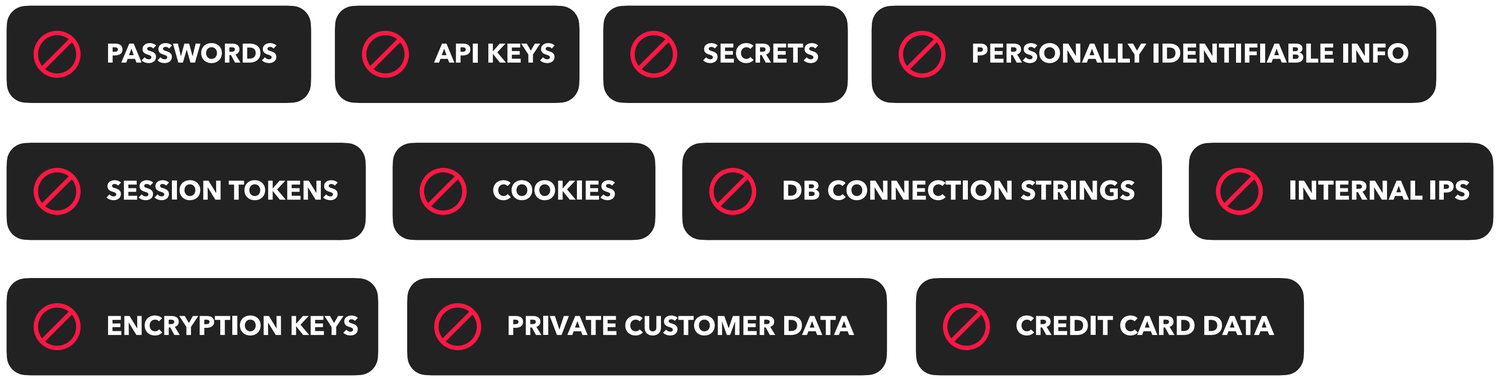
```go
func Recovery(h http.Handler) http.Handler {
    return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
        ctx := r.Context()
        logger := sharedContext.Logger(ctx)
        defer func() {
            err := panicToError(recover())
            if err != nil {
                logger.Error("unexpected panic", zap.Error(err))
                w.WriteHeader(http.StatusInternalServerError)
                return
            }
        }()

        h.ServeHTTP(w, r)
    })
}
```

```go
func panicToError(value any) error {
    var panicErr error
    switch e := value.(type) {
    case nil:
        return nil
    case string:
        if e == "" {
            return nil
        }
        panicErr = errors.New(e)
    case error:
        panicErr = e
    default:
        panicErr = fmt.Errorf("unknown panic: %v", e)
    }
    return panicErr
}
```

RECOVERY MIDDLEWARE

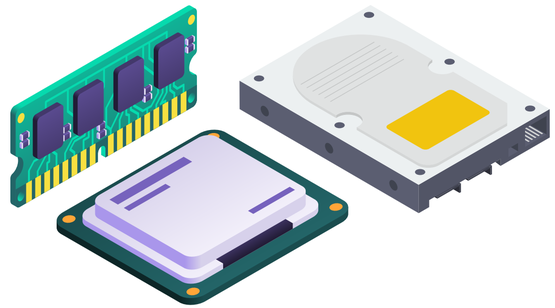


BE SENSITIVE

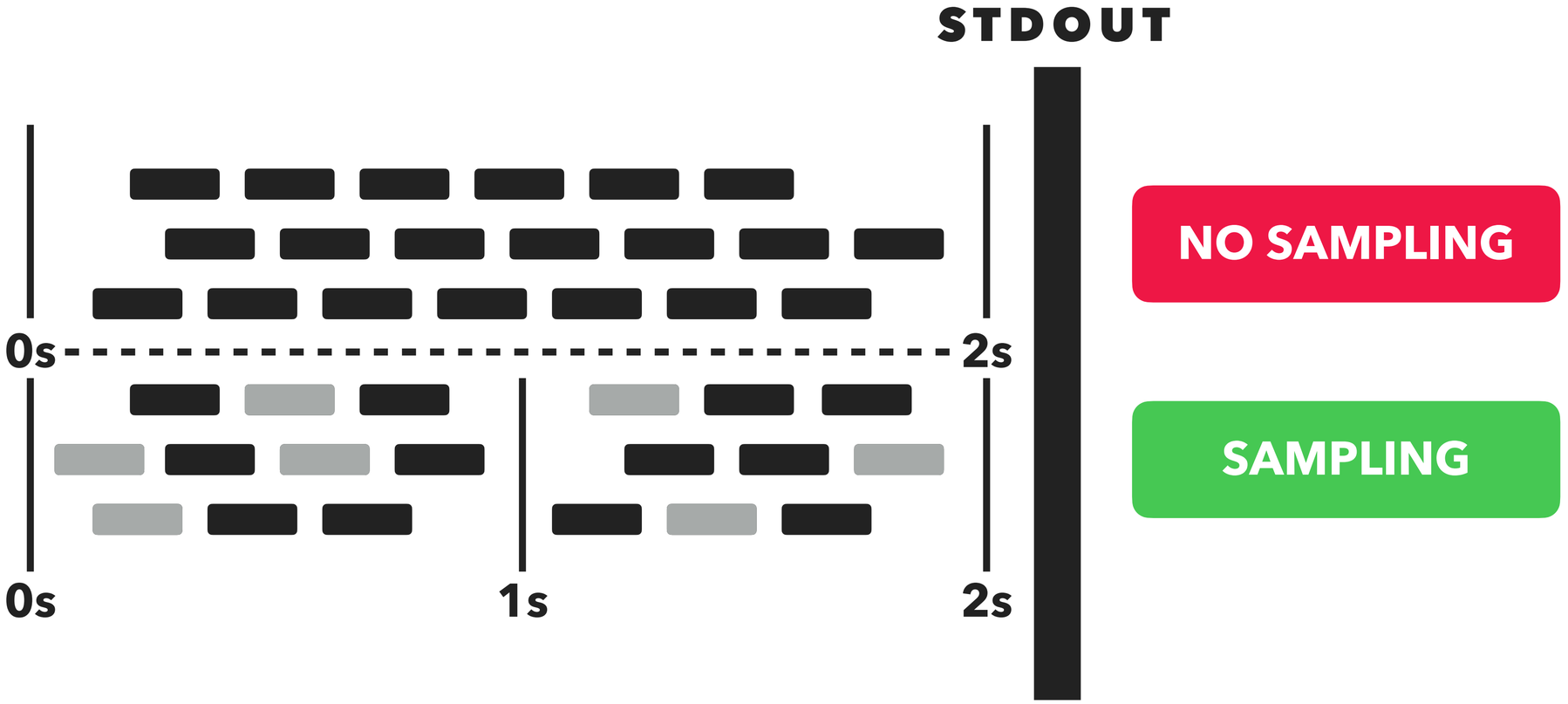


BE SENSITIVE, NOT ~~EXCLUSIVE~~

- ERROR SANITISATION & FILTERING
- NON SENSITIVE IDS
- REQUEST/RESPONSE METADATA
- DB QUERY EXECUTION TIMES
- ADMIN ACTIONS AUDIT TRAILS
- 3RD PARTY INTERACTIONS



CAN'T BREAK THE BANK


STDOUT
0s
2s
0s
1s
2s
NO SAMPLING
SAMPLING

SAMPLING VS ~~SAMPLING~~

~~STANDARD LOG PACKAGE~~

10M ROUTINES

1K LIMITER

JSON FORMAT

1 INFO / ROUTINE

M1 MAX

WITHOUT SAMPLING

```go
func main() {
    file, err := os.Create("std.log")
    if err != nil {
        log.Fatalf("could not create log file: %v", err)
    }
    defer func() { _ = file.Close() }()
    log.SetOutput(file)
    log.SetFlags(0)

    var wg sync.WaitGroup
    limiter := make(chan struct{}, 1000)
    numberOfRequests, start := 10_000_000, time.Now()
    for i := 1; i <= numberOfRequests; i++ {
        limiter <- struct{}{}
        wg.Add(1)
        go req(&wg, i, limiter)
    }
    wg.Wait()
    fmt.Printf("done in: %v\n", time.Since(start))
}
```

```go
func line(level, msg string, fields map[string]interface{}) string {
    ts := time.Now().Format(time.RFC3339)
    logEntry := map[string]interface{}{
        "level": level,
        "msg":   msg,
        "ts":    ts,
    }
    for k, v := range fields {
        logEntry[k] = v
    }
    bs, _ := json.Marshal(logEntry)
    return string(bs)
}

func req(wg *sync.WaitGroup, id int, limiter chan struct{}) {
    defer wg.Done()
    defer func() { <-limiter }()
    log.Println(line("info", "request", map[string]any{"req_id": id}))
}
```

WITH SAMPLING

```go
func main() {
    l, err := logger()
    if err != nil {
        panic(err)
    }
    defer func() { _ = l.Sync() }()

    var wg sync.WaitGroup
    limiter := make(chan struct{}, 1000)
    numberOfRequests, start := 10_000_000, time.Now()
    for i := 1; i <= numberOfRequests; i++ {
        limiter <- struct{}{}
        wg.Add(1)
        go req(&wg, i, l, limiter)
    }

    wg.Wait()
    fmt.Printf("done in: %v\n", time.Since(start))
}
```

```go
func logger() (*zap.Logger, error) {
    cfg := zap.NewProductionConfig()
    cfg.EncoderConfig.EncodeTime = zapcore.ISO8601TimeEncoder
    cfg.OutputPaths = []string{"zap.log"}

    return cfg.Build()
}

func req(wg *sync.WaitGroup, id int, logger *zap.Logger, limiter chan struct{}) {
    defer wg.Done()
    defer func() { <-limiter }()
    logger.Info("request", zap.Int("req_id", id))
}
```

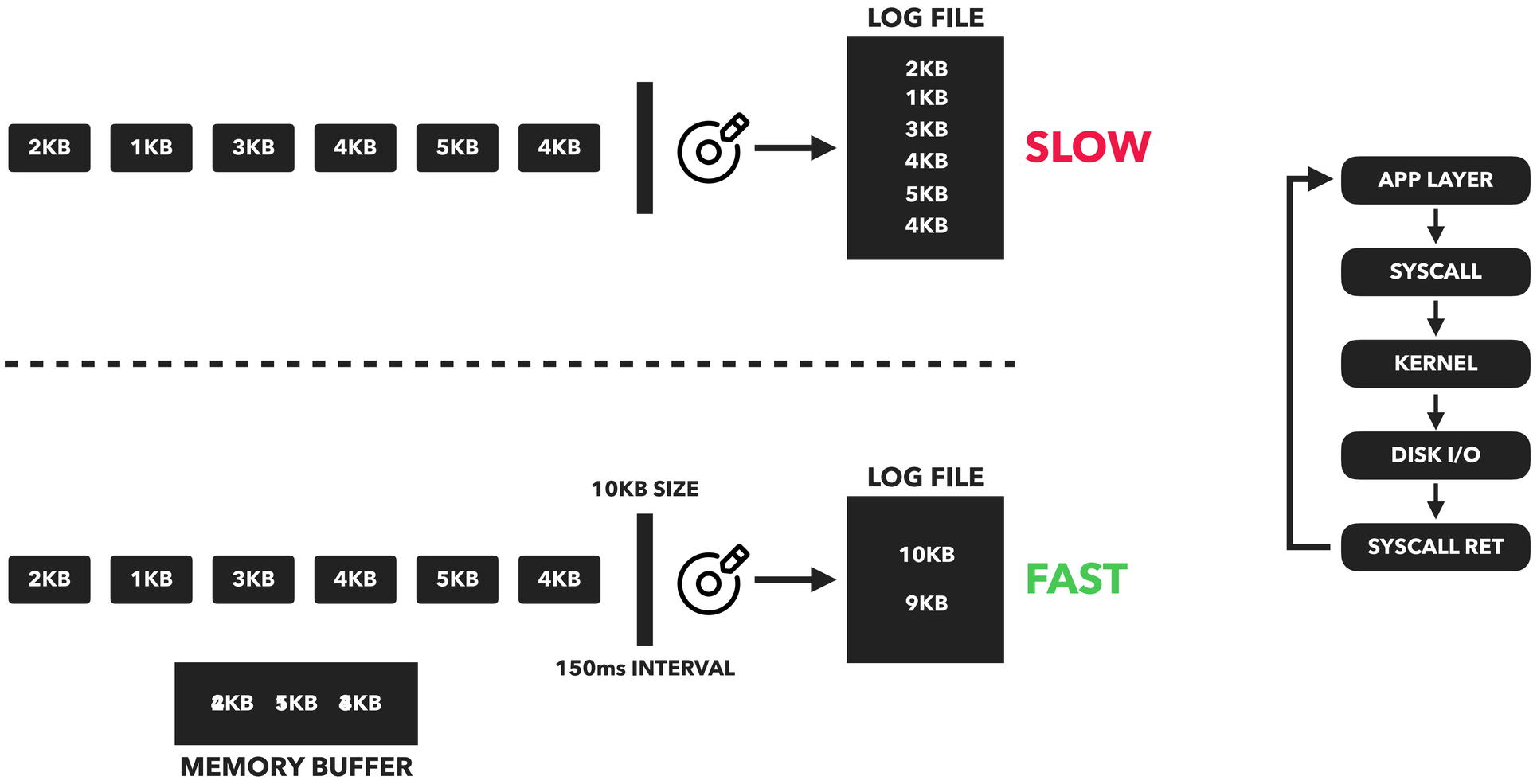
SAMPLING BY DEFAULT

ZAP SAMPLING DEFAULT

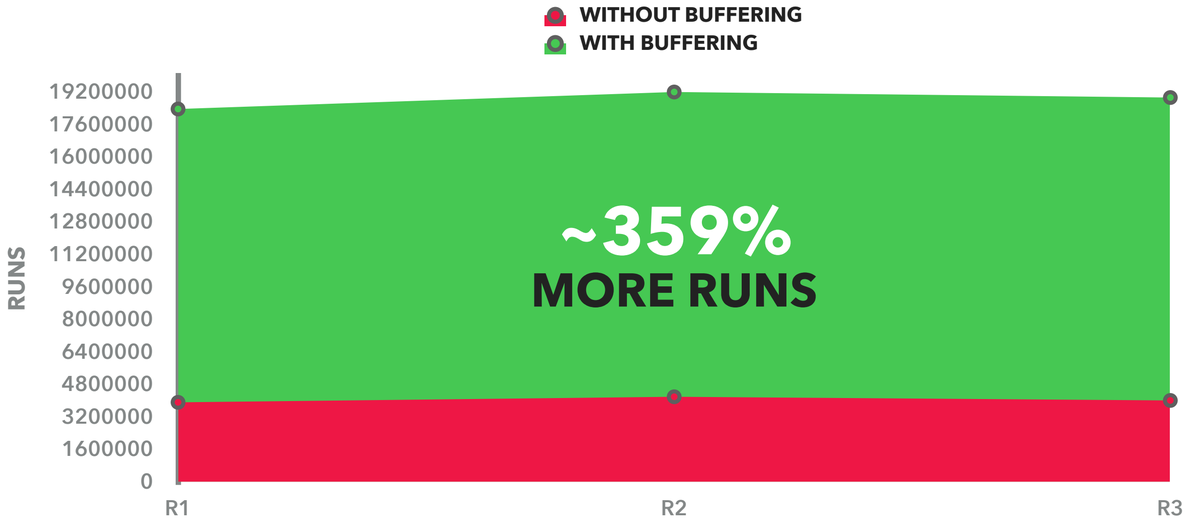
```go
if scfg := cfg.Sampling; scfg != nil {
    opts = append(opts, WrapCore(func(core zapcore.Core) zapcore.Core {
        var samplerOpts []zapcore.SamplerOption
        if scfg.Hook != nil {
            samplerOpts = append(samplerOpts, zapcore.SamplerHook(scfg.Hook))
        }
        return zapcore.NewSamplerWithOptions(
            core,
            time.Second,
            cfg.Sampling.Initial,
            cfg.Sampling.Thereafter,
            samplerOpts...,
        )
    }))
}
```

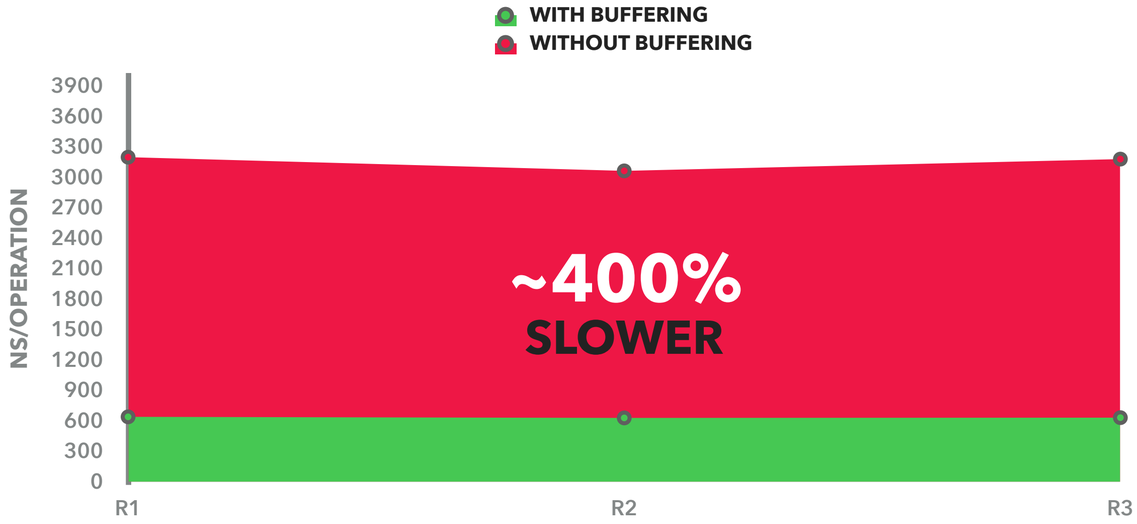
```go
func NewProductionConfig() Config {
    return Config{
        Level:       NewAtomicLevelAt(InfoLevel),
        Development: false,
        Sampling: &SamplingConfig{
            Initial:    100,
            Thereafter: 100,
        },
        Encoding:         "json",
        EncoderConfig:    NewProductionEncoderConfig(),
        OutputPaths:      []string{"stderr"},
        ErrorOutputPaths: []string{"stderr"},
    }
}
```

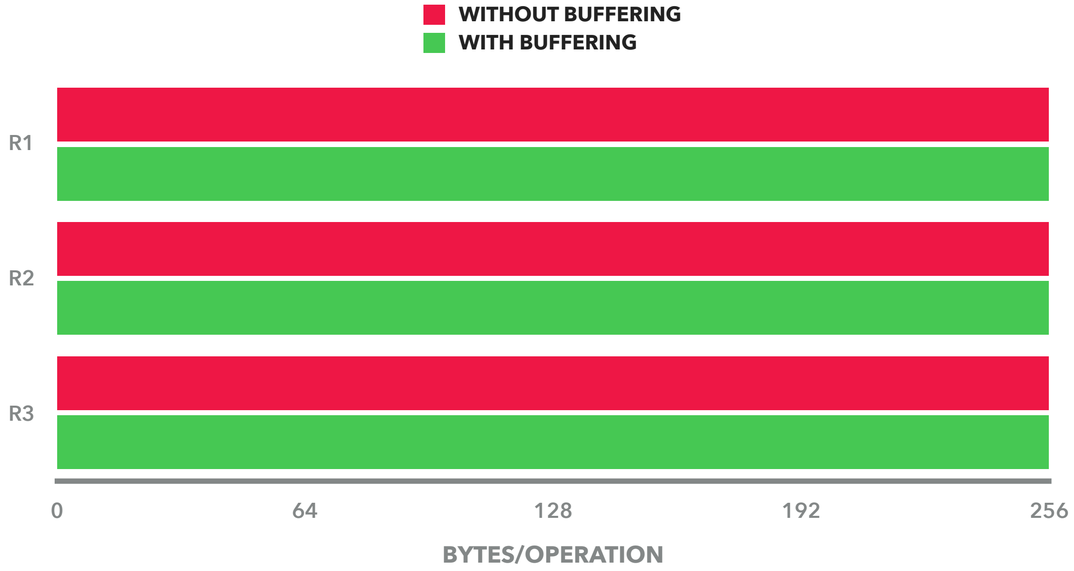
BUFFERING

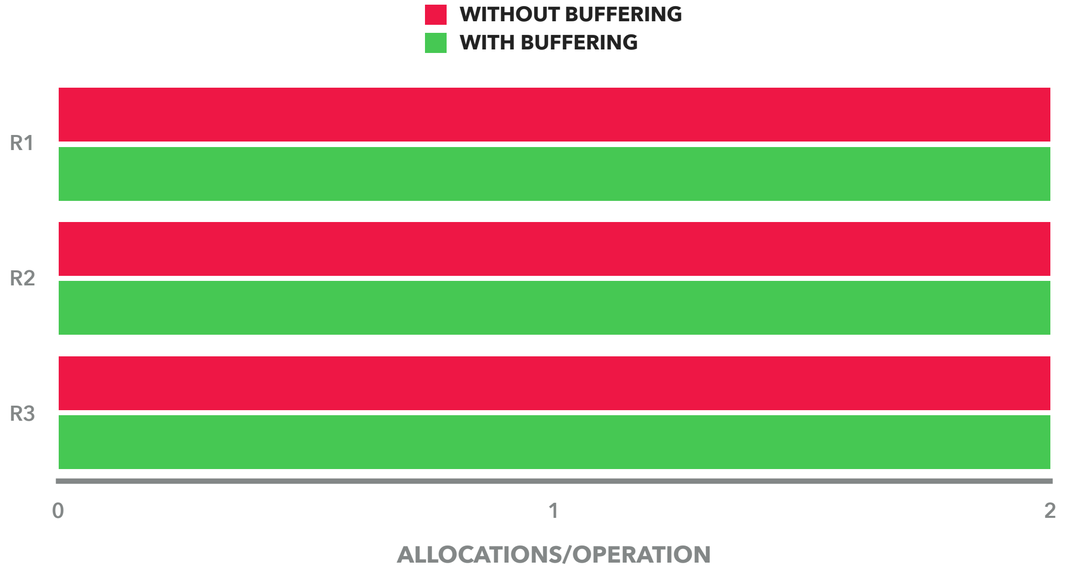

LOG FILE
2KB
1KB
3KB
4KB
5KB
4KB
SLOW
2KB
1KB
3KB
4KB
5KB
4KB
10KB SIZE
150ms INTERVAL
LOG FILE
10KB
9KB
FAST
MEMORY BUFFER
APP LAYER
SYSCALL
KERNEL
DISK I/O
SYSCALL RET

BUFFERING VS ~~BUFFERING~~









NO BUFFERING

```
run benchmark | debug benchmark
func BenchmarkWithoutBuffering(b *testing.B) {
    cfg := zap.NewProductionConfig()
    cfg.EncoderConfig.EncodeTime = zapcore.ISO8601TimeEncoder
    cfg.OutputPaths = []string{"zap.log"}
    cfg.Sampling = nil
    logger, _ := cfg.Build()

    b.ReportAllocs()
    b.ResetTimer()
    for i := 0; i < b.N; i++ {
        logger.Info("some message")
    }
}
```

BUFFERING

```
run benchmark | debug benchmark
func BenchmarkWithBuffering(b *testing.B) {
    cfg := zap.NewProductionConfig()
    cfg.EncoderConfig.EncodeTime = zapcore.EpochNanosTimeEncoder
    cfg.OutputPaths = []string{"zap.log"}
    cfg.Sampling = nil
    logger, _ := cfg.Build()
    ws, _, _ := zap.Open(cfg.OutputPaths...)
    bufferedWriteSyncer := &zapcore.BufferedWriteSyncer{
        WS:            ws,
        Size:          256 * 1024,
        FlushInterval: 2 * time.Second,
    }
    logger = logger.WithOptions(zap.WrapCore(func(core zapcore.Core) zapcore.Core {
        return zapcore.NewCore(
            zapcore.NewJSONEncoder(cfg.EncoderConfig),
            bufferedWriteSyncer,
            cfg.Level,
        )
    }))

    b.ReportAllocs()
    b.ResetTimer()
    for i := 0; i < b.N; i++ {
        logger.Info("some message")
    }
}
```

WHICH ONE?

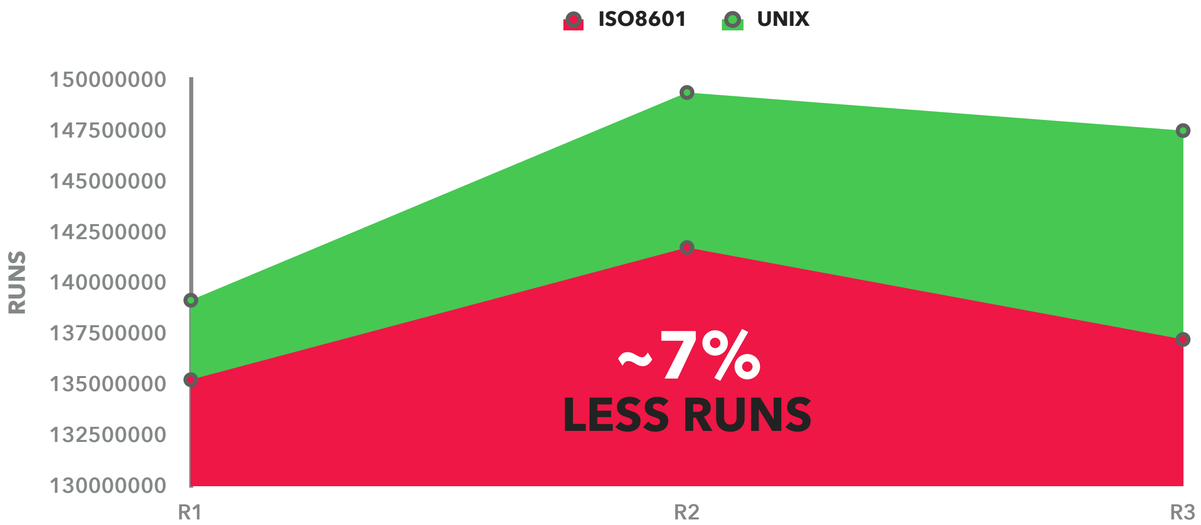
UNIX EPOCH NANO

```
1697041234567890123
```

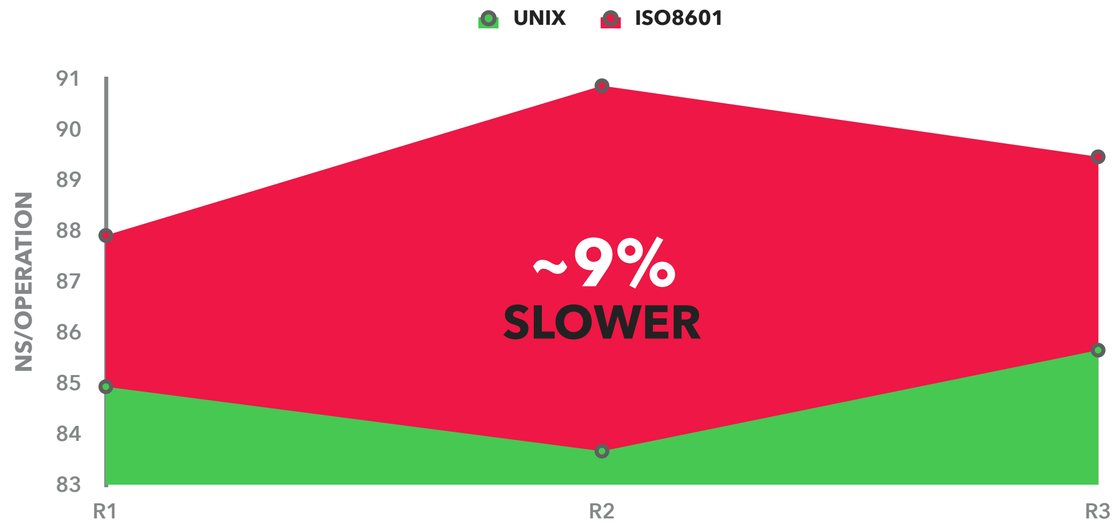
ISO 8601

```
2023-10-11T14:20:34Z
```

UNIX VS ISO8601



2BYTES/OPERATION



0 ALLOCATIONS/OPERATION

UNIX VS ISO8601

```
run benchmark | debug benchmark
func BenchmarkUnixTime(b *testing.B) {
    cfg := zap.NewProductionConfig()
    cfg.EncoderConfig.EncodeTime = zapcore.EpochNanosTimeEncoder
    cfg.OutputPaths = []string{"zap.log"}
    logger, _ := cfg.Build()

    b.ReportAllocs()
    b.ResetTimer()
    for i := 0; i < b.N; i++ {
        logger.Info("some message")
    }
}
```

```
run benchmark | debug benchmark
func BenchmarkISO8601Time(b *testing.B) {
    cfg := zap.NewProductionConfig()
    cfg.EncoderConfig.EncodeTime = zapcore.ISO8601TimeEncoder
    cfg.OutputPaths = []string{"zap.log"}
    logger, _ := cfg.Build()

    b.ReportAllocs()
    b.ResetTimer()
    for i := 0; i < b.N; i++ {
        logger.Info("some message")
    }
}
```

HOT PATHS

TX1

```go
func tx1(logger *zap.Logger, session db.Session, wg *sync.WaitGroup, limiter chan struct{}, r req) {
    defer wg.Done()
    defer func() { <-limiter }()
    _ = session.Tx(func(sess db.Session) error {
        authorID := uuid.New().String()
        a := author{ID: authorID, Name: r.Author}
        if _, err := sess.SQL().InsertInto("authors1").Values(a).Exec(); err != nil {
            return err
        }
        logger.Info("inserted author", zap.String("author_id", authorID))
        bookID := uuid.New().String()
        b := book{ID: bookID, Title: r.Book, AuthorID: authorID}
        if _, err := sess.SQL().InsertInto("books1").Values(b).Exec(); err != nil {
            return err
        }
        bookCopyID := uuid.New().String()
        logger.Info("inserted book", zap.String("book_id", bookID))
        c := bookCopy{ID: bookCopyID, Status: "AVAILABLE", BookID: bookID}
        if _, err := sess.SQL().InsertInto("book_copies1").Values(c).Exec(); err != nil {
            return err
        }
        logger.Info("inserted book", zap.String("book_copy_id", bookCopyID))
        return nil
    })
}
```

```sql
CREATE TABLE authors1
(
    id   UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    name VARCHAR(100) NOT NULL
);

CREATE TABLE books1
(
    id   UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    author_id UUID         NOT NULL,
    title     VARCHAR(100) NOT NULL,
    FOREIGN KEY (author_id) REFERENCES authors1 (id)
);

CREATE TABLE book_copies1
(
    id   UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    book_id UUID        NOT NULL,
    status  VARCHAR(20) NOT NULL,
    FOREIGN KEY (book_id) REFERENCES books1 (id)
);
```

TX2

```go
func tx2(logger *zap.Logger, session db.Session, wg *sync.WaitGroup, limiter chan struct{}, r req) {
    defer wg.Done()
    defer func() { <-limiter }()
    _ = session.Tx(func(sess db.Session) error {
        authorID := uuid.New().String()
        a := author{ID: authorID, Name: r.Author}
        logger.Info("inserting author", zap.Any("author", a))
        if _, err := sess.SQL().InsertInto("authors2").Values(a).Exec(); err != nil {
            return err
        }
        logger.Info("inserted author", zap.Any("author", a))
        bookID := uuid.New().String()
        b := book{ID: bookID, Title: r.Book, AuthorID: authorID}
        logger.Info("inserting book", zap.Any("book", b))
        if _, err := sess.SQL().InsertInto("books2").Values(b).Exec(); err != nil {
            return err
        }
        logger.Info("inserted book", zap.Any("book", b))
        bookCopyID := uuid.New().String()
        c := bookCopy{ID: bookCopyID, Status: "AVAILABLE", BookID: bookID}
        logger.Info("inserting book copy", zap.Any("book_copy", c))
        if _, err := sess.SQL().InsertInto("book_copies2").Values(c).Exec(); err != nil {
            return err
        }
        logger.Info("inserted book copy", zap.Any("book_copy", c))
        return nil
    })
}
```

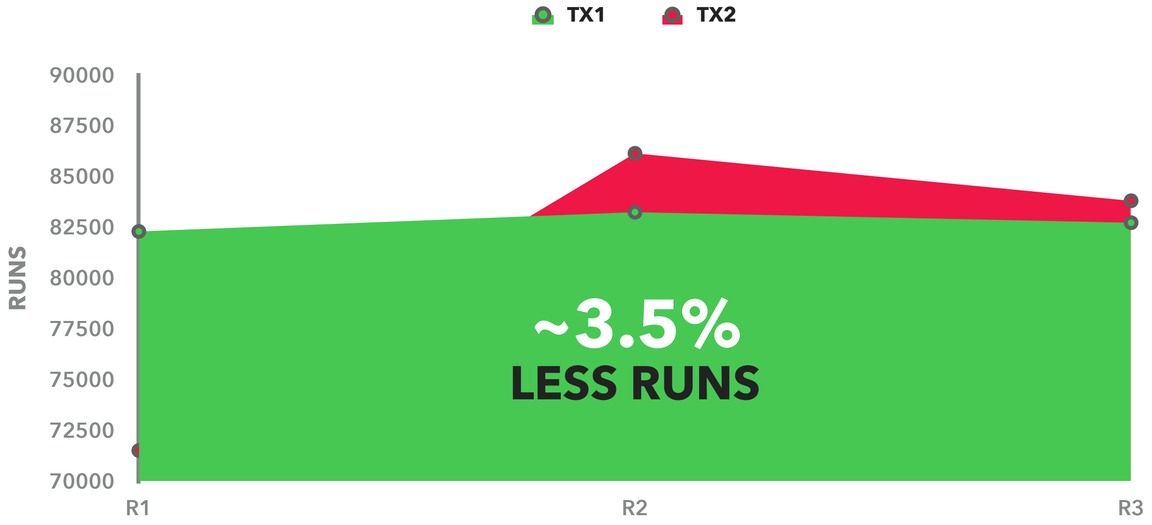
REFLECTION

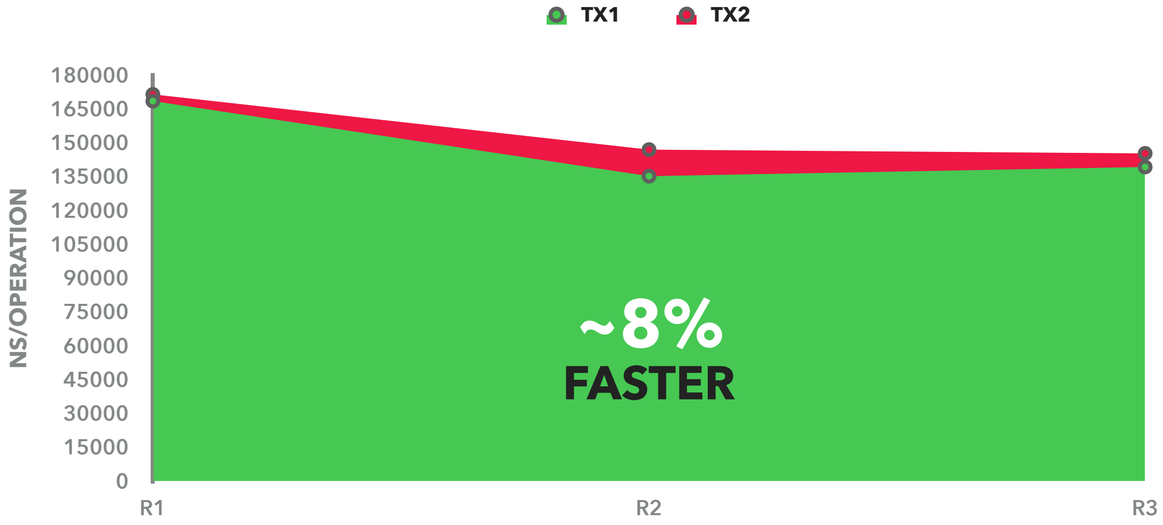
```sql
CREATE TABLE authors2
(
    id   UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    name VARCHAR(100) NOT NULL
);

CREATE TABLE books2
(
    id   UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    author_id UUID         NOT NULL,
    title     VARCHAR(100) NOT NULL,
    FOREIGN KEY (author_id) REFERENCES authors2 (id)
);

CREATE TABLE book_copies2
(
    id   UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    book_id UUID        NOT NULL,
    status  VARCHAR(20) NOT NULL,
    FOREIGN KEY (book_id) REFERENCES books2 (id)
);
```

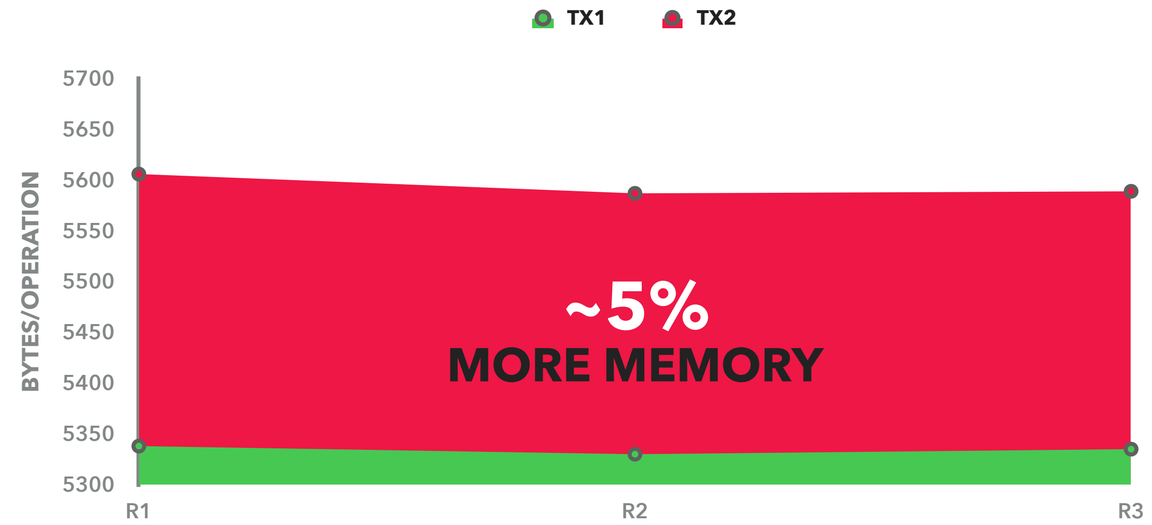
TX1 VS TX2

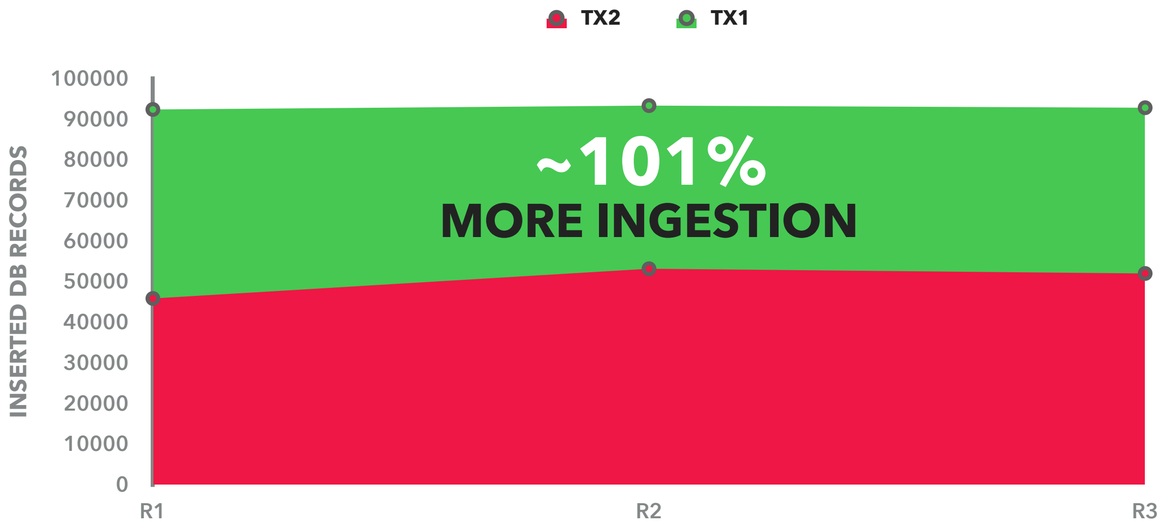
```
run benchmark | debug benchmark
func BenchmarkTx1(b *testing.B) {
    var wg sync.WaitGroup
    cfg := zap.NewProductionConfig()
    cfg.OutputPaths = []string{"zap1.log"}
    logger, _ := cfg.Build()
    session := postgres()
    limiter := make(chan struct{}, 100)

    b.ReportAllocs()
    b.ResetTimer()
    for i := 0; i < b.N; i++ {
        wg.Add(1)
        limiter <- struct{}{}
        go func(i int) {
            defer wg.Done()
            defer func() { <-limiter }()
            tx1(logger, session, &wg, limiter, req{
                Author: fmt.Sprintf("Author %d", i),
                Book:   fmt.Sprintf("Book %d", i),
            })
        }(i)
    }
    wg.Wait()
}
```

```
run benchmark | debug benchmark
func BenchmarkTx2(b *testing.B) {
    var wg sync.WaitGroup
    cfg := zap.NewProductionConfig()
    cfg.OutputPaths = []string{"zap2.log"}
    logger, _ := cfg.Build()
    session := postgres()
    limiter := make(chan struct{}, 100)

    b.ReportAllocs()
    b.ResetTimer()
    for i := 0; i < b.N; i++ {
        wg.Add(1)
        limiter <- struct{}{}
        go func(i int) {
            defer wg.Done()
            defer func() { <-limiter }()
            tx2(logger, session, &wg, limiter, req{
                Author: fmt.Sprintf("Author %d", i),
                Book:   fmt.Sprintf("Book %d", i),
            })
        }(i)
    }
    wg.Wait()
}
```


TX1
TX2
RUNS
90000
87500
85000
82500
80000
77500
75000
72500
70000
R1
R2
R3
~3.5%
LESS RUNS


TX1
TX2
NS/OPERATION
180000
165000
150000
135000
120000
105000
90000
75000
60000
45000
30000
15000
0
R1
R2
R3
~8%
FASTER


TX1
TX2
BYTES/OPERATION
5700
5650
5600
5550
5500
5450
5400
5350
5300
R1
R2
R3
~5%
MORE MEMORY


TX2
TX1
INSERTED DB RECORDS
100000
90000
80000
70000
60000
50000
40000
30000
20000
10000
0
R1
R2
R3
~101%
MORE INGESTION

STORAGE AIN'T FREE

CLOUD LOGGING

FREE
FIRST 50 GiB/PROJECT/MONTH

PAID
$0.50/GiB INDEXING + < 30D STORAGE
$0.01/GiB/MONTH BUCKET STORAGE

CLOUDWATCH

FREE
5 GiB/MONTH STORAGE & RETENTION

PAID
$0.50/GiB INDEXING + < 30D STORAGE
$0.03/GiB/MONTH BUCKET STORAGE
$0.005/GiB SCANNED LOG QUERYING

MONITOR

FREE
5 GiB/MONTH STORAGE

PAID
$0.10/GiB AUXILIARY LOGS
$0.50/GiB BASIC LOGS
$2.30/GiB ANALYTICS LOGS
$0.10/GiB INTERACTIVE RETENTION
$0.02/GiB LONG TERM RETENTION
$0.005/GiB OF DATA SCANNED

CLOUD STORAGE

FREE
5 GiB/MONTH US REGIONS ONLY

PAID
STANDARD **$0.02/GiB/MONTH**
NEARLINE **$0.01/GiB/MONTH**
COLDLINE **$0.004/GiB/MONTH**
ARCHIVE **$0.0012/GiB/MONTH**

S3

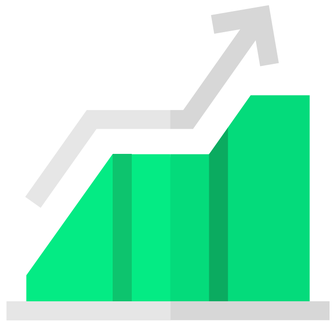
FREE
5 GiB STANDARD STORAGE

PAID
S3 STANDARD **$0.023/GiB/MONTH**
S3 IA **$0.0125/GiB/MONTH**
S3 GLACIER **$0.004/GiB/MONTH**
S3 GLACIER DEEP **$0.00099/GiB/MONTH**

BLOG STORAGE

FREE
5 GiB AZURE BLOB STORAGE

PAID
PREMIUM **$0.15/GiB/MONTH**
HOT **$0.018/GiB/MONTH**
COOL **$0.01/GiB/MONTH**
COLD **$0.0036/GiB/MONTH**
ARCHIVE **$0.00099/GiB/MONTH**



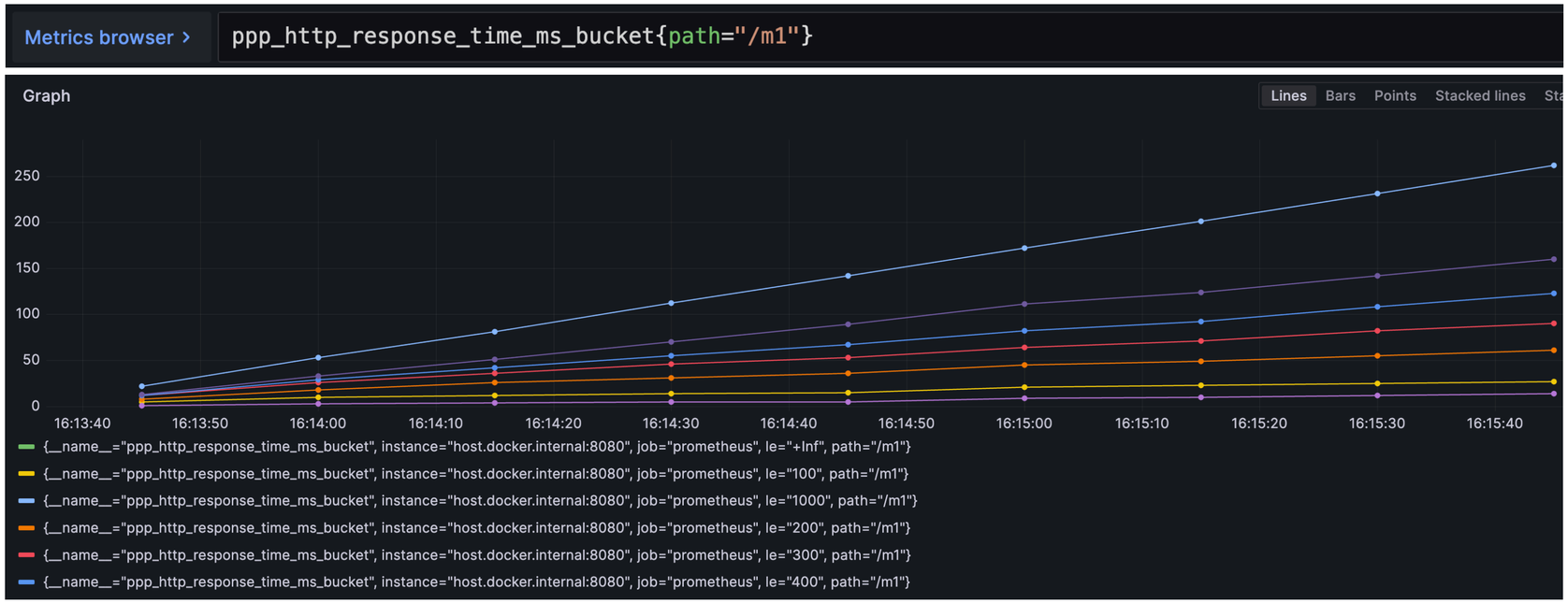
METRICS

RESPONSE TIME MIDDLEWARE

```go
var responseTimeHistogramMetric = metrics.HistogramVecWithBuckets(
    "http_response_time_ms",
    []float64{50, 100, 200, 300, 400, 500, 1000},
    "Response time of the HTTP requests",
    "path",
)

func ResponseTime(h http.Handler) http.Handler {
    return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
        start := time.Now()
        h.ServeHTTP(w, r)
        duration := float64(time.Since(start).Milliseconds())
        responseTimeHistogramMetric.With(map[string]string{
            "path": r.URL.Path,
        }).Observe(duration)
    })
}
```

```go
func NewRouter() http.Handler {
    mux := http.NewServeMux()
    rand.New(rand.NewSource(time.Now().UnixNano()))

    mux.Handle("/metrics", metrics.PrometheusHandler())
    mux.HandleFunc("/m1", func(w http.ResponseWriter, r *http.Request) {
        time.Sleep(time.Duration(rand.Int63n(900)) * time.Millisecond)
        w.WriteHeader(http.StatusOK)
    })
    mux.HandleFunc("/m2", func(w http.ResponseWriter, r *http.Request) {
        time.Sleep(time.Duration(rand.Int63n(900)) * time.Millisecond)
        w.WriteHeader(http.StatusOK)
    })

    return middleware.New(
        mux,
        middleware.ResponseTime,
    )
}
```

HISTOGRAM BUCKETS



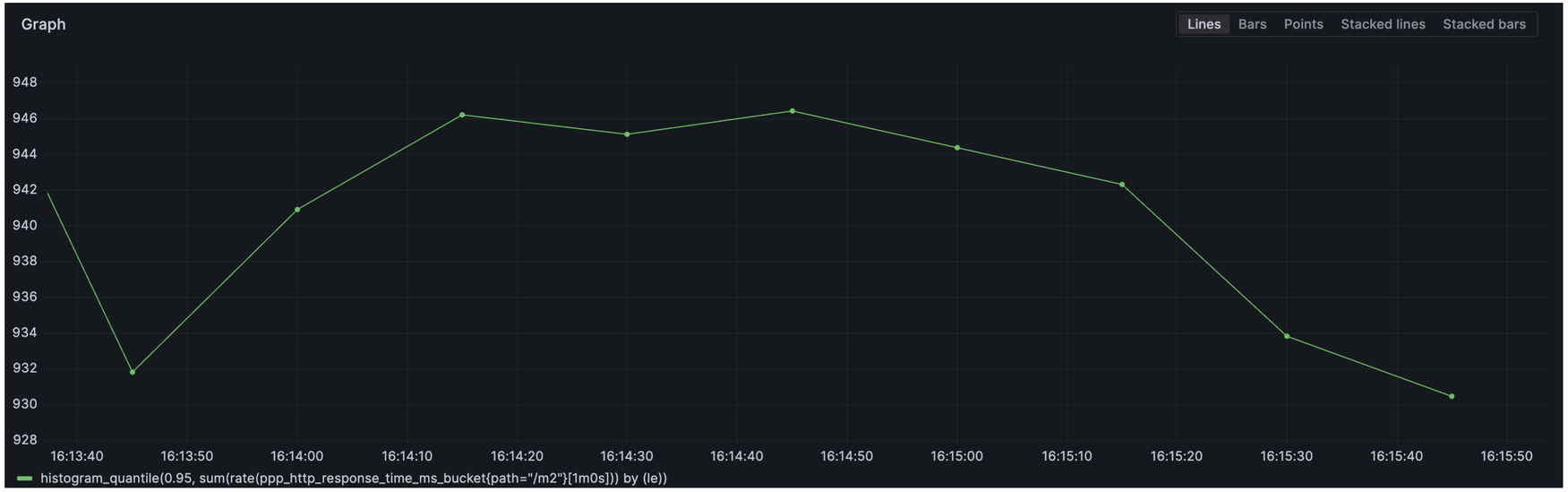
HISTOGRAM 95TH PERCENTILE

Metrics browser ›

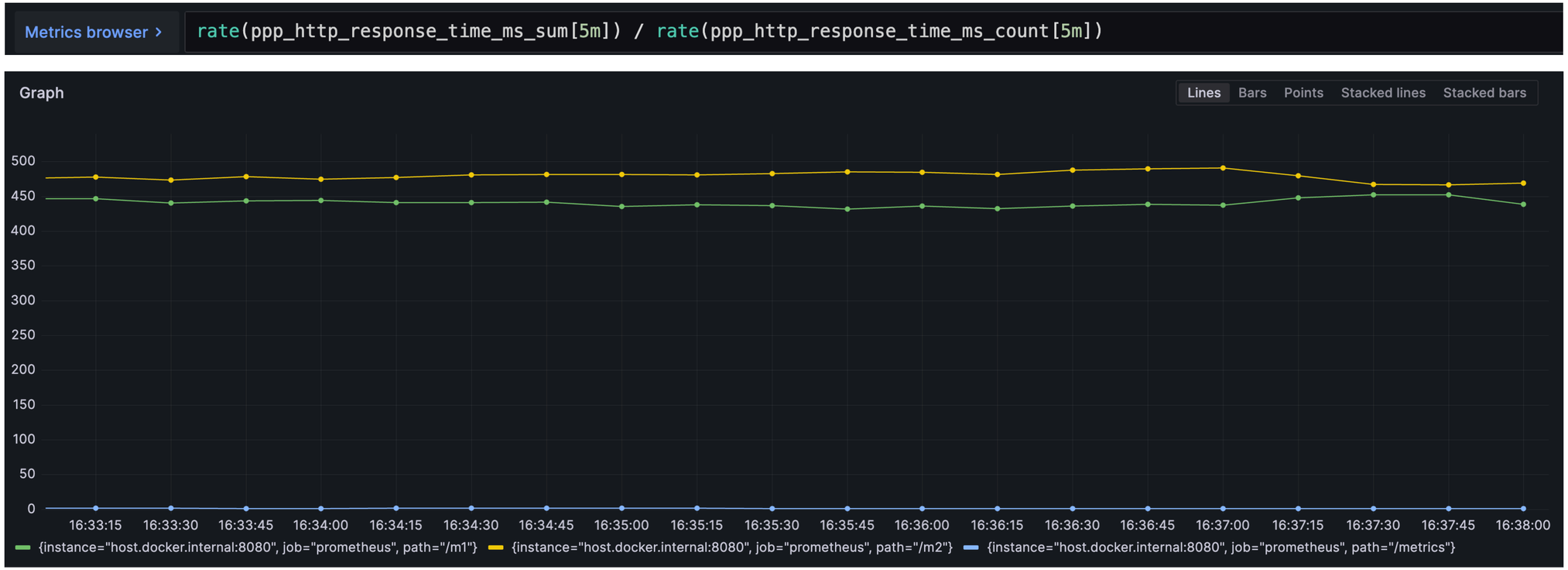
```
histogram_quantile(0.95, sum(rate(ppp_http_response_time_ms_bucket{path="/m1"}[$__rate_interval])) by (le))
```

Metrics browser ›

```
histogram_quantile(0.95, sum(rate(ppp_http_response_time_ms_bucket{path="/m2"}[$__rate_interval])) by (le))
```



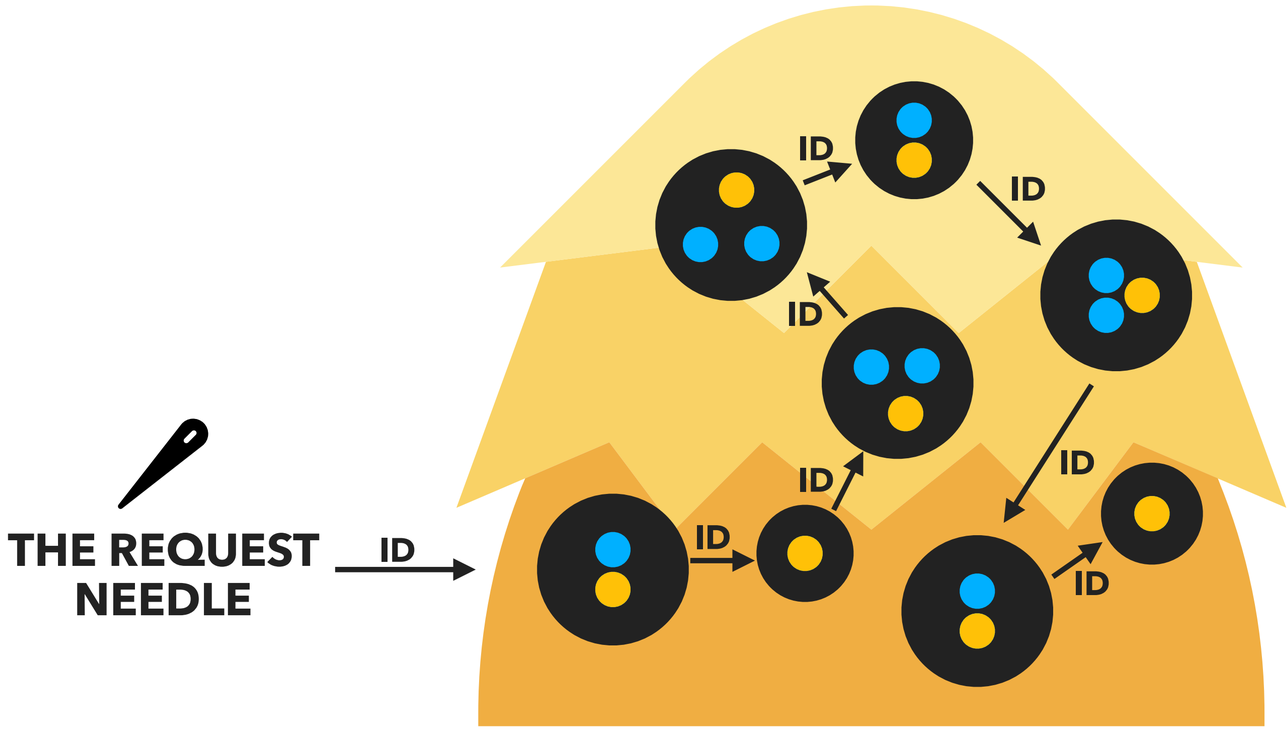
HISTOGRAM MEAN/AVERAGE



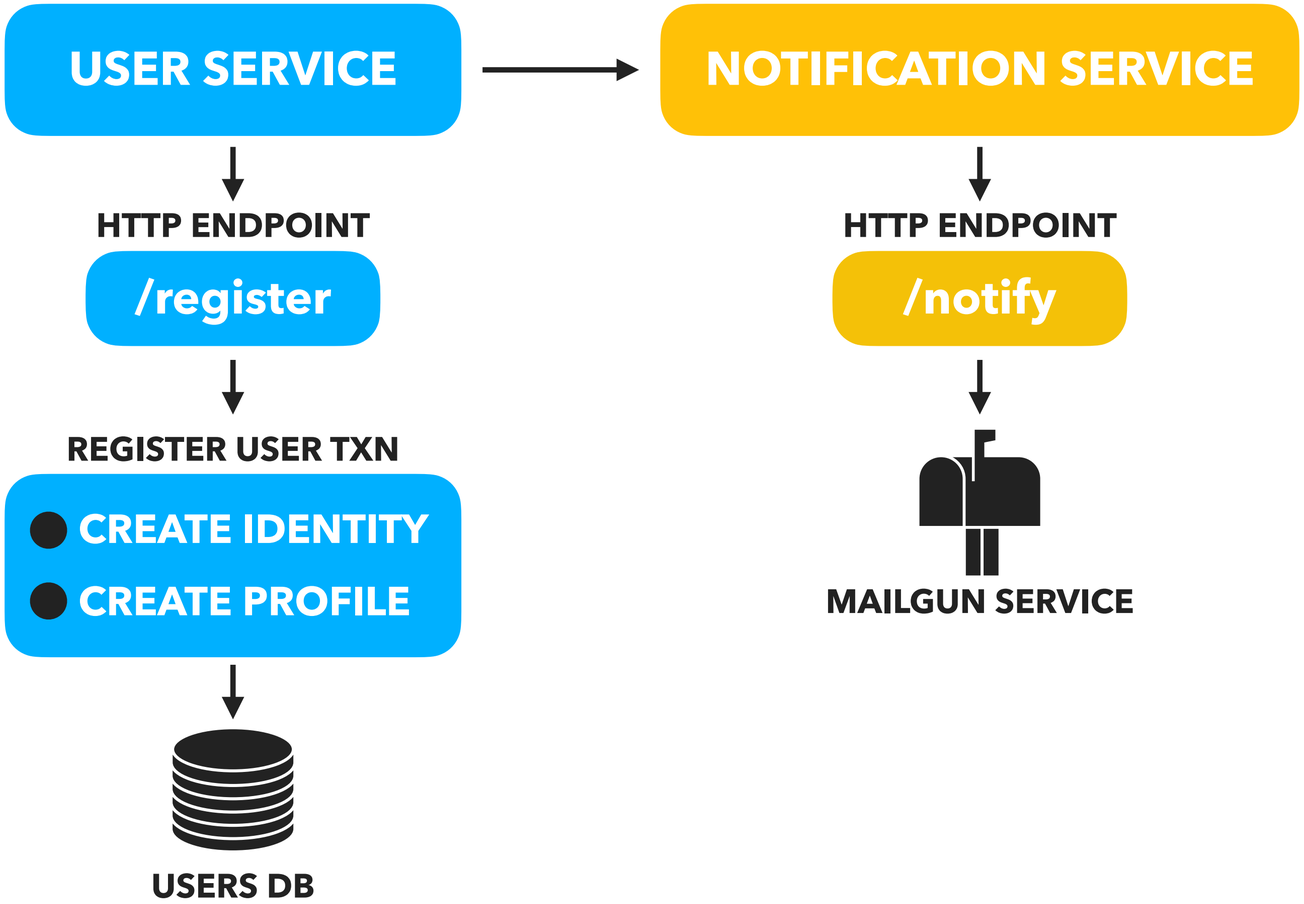
ADVICE

- START ADDING METRICS
- AVOID HIGH CARDINALITY LABELS
- AVOID OVERUSING COUNTERS OVER GAUGES
- AVOID INSUFFICIENTLY DETAILED HISTOGRAM BUCKETS
- DEFINE PROPER METRIC AGGREGATIONS
- DEFINE A GLOBAL METRIC PREFIX
- DEFINE USEFUL ALERTS FOR EXISTING METRICS

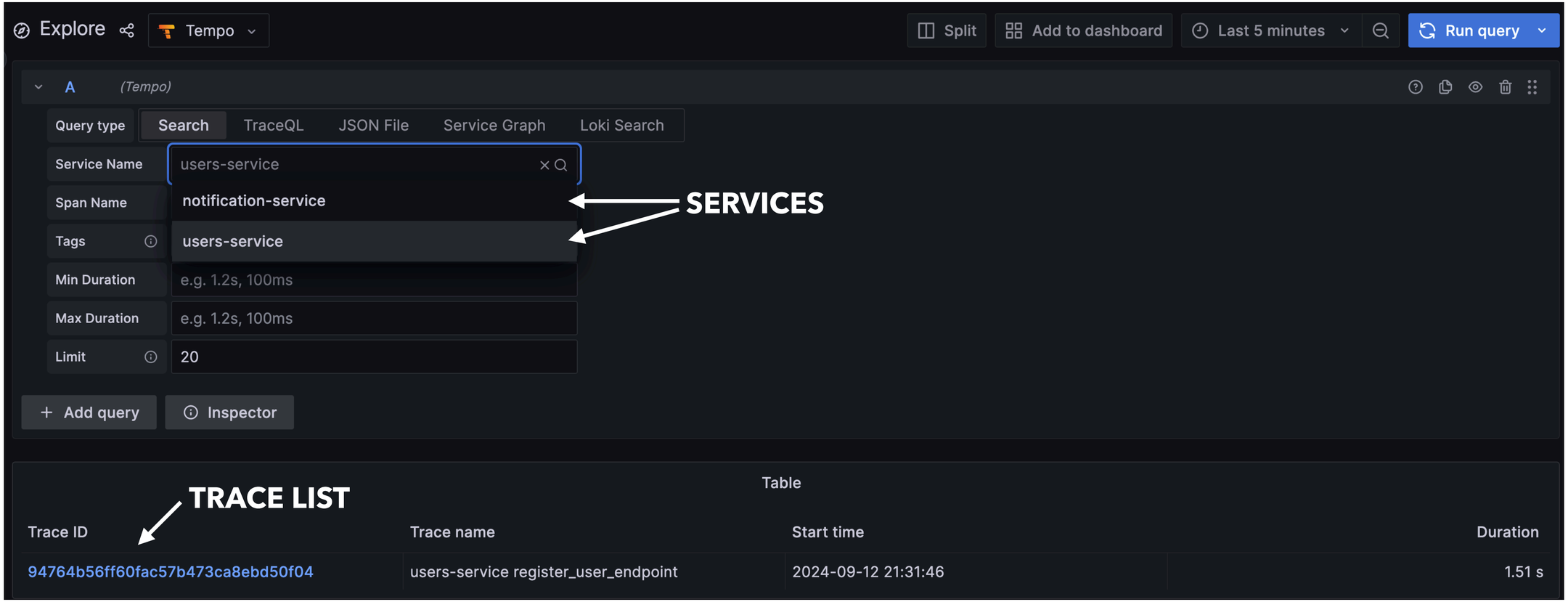
DISTRIBUTED TRACING



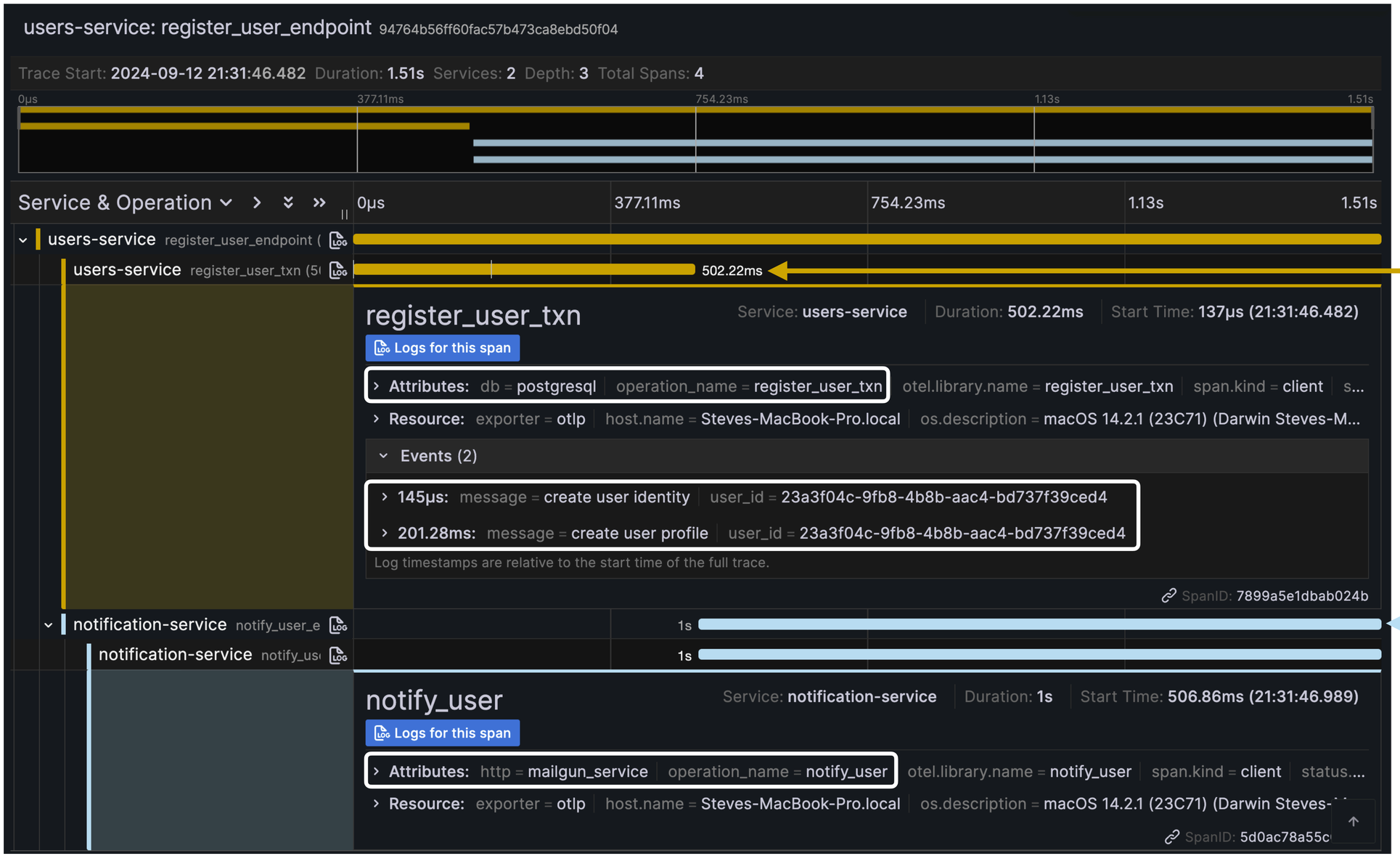
THE MICRO SERVICE HAYSTACK


USER SERVICE
NOTIFICATION SERVICE
HTTP ENDPOINT
/register
REGISTER USER TXN
CREATE IDENTITY
CREATE PROFILE
USERS DB
HTTP ENDPOINT
/notify
MAILGUN SERVICE

TEMPO EXPLORE



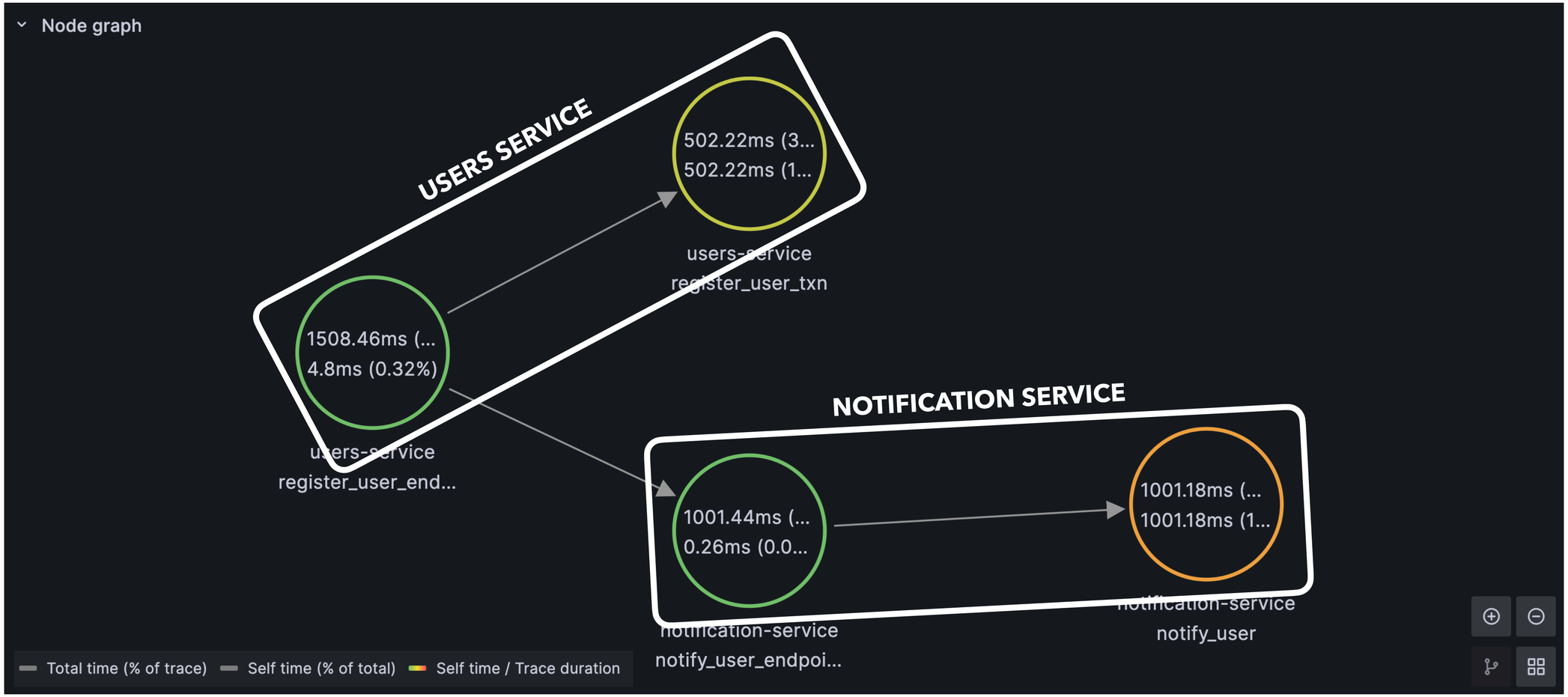
TRACE VIEW



USER SVC TIMELINE

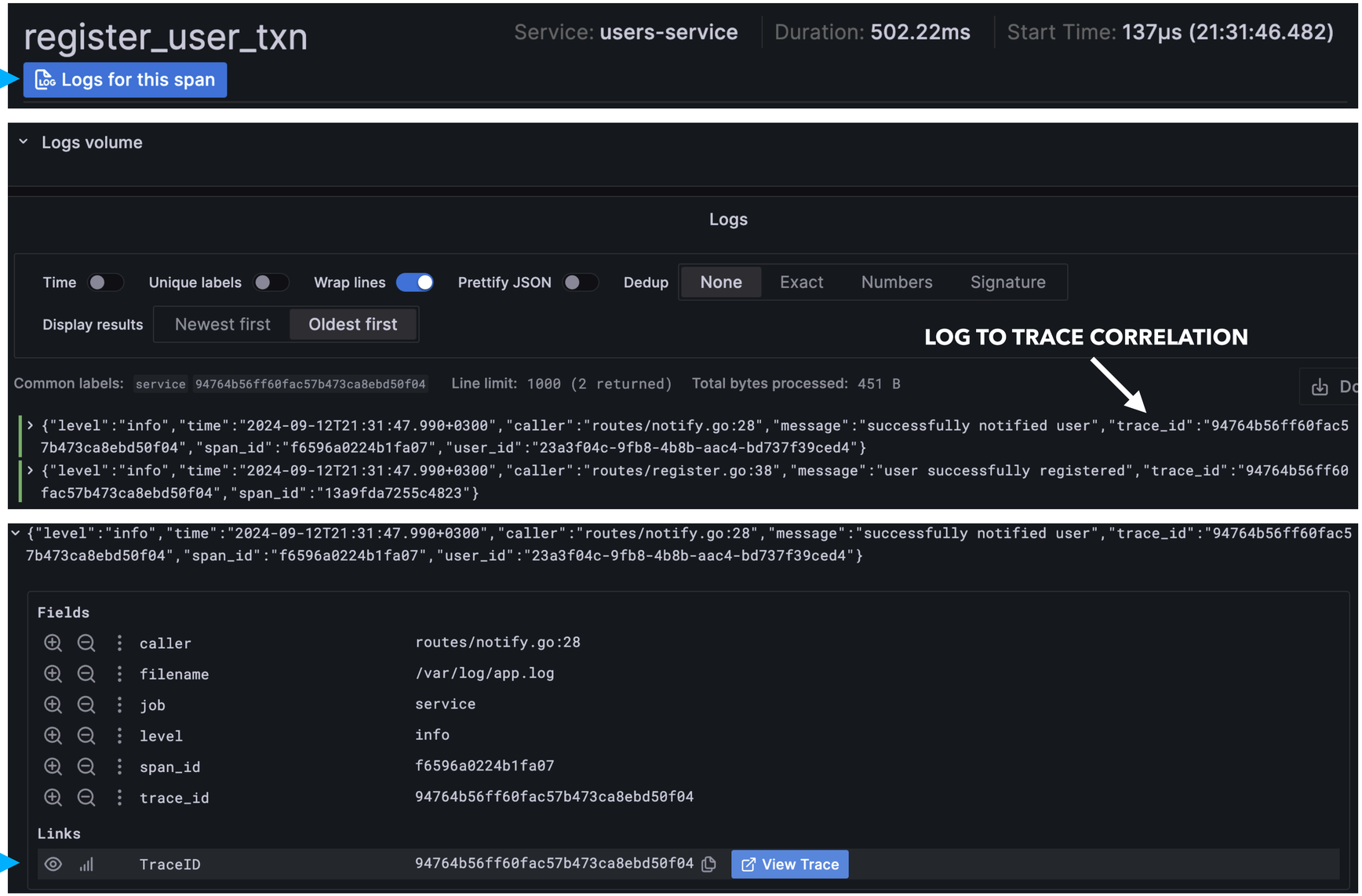
NOTIFICATION SVC TIMELINE

TRACE NODE GRAPH



TRACE TO LOGS CORRELATION

LOGS CORRELATION



TRACE CORRELATION

OTLP + GRAFANA SETUP

ADJUST THE CONTEXT LOGGER

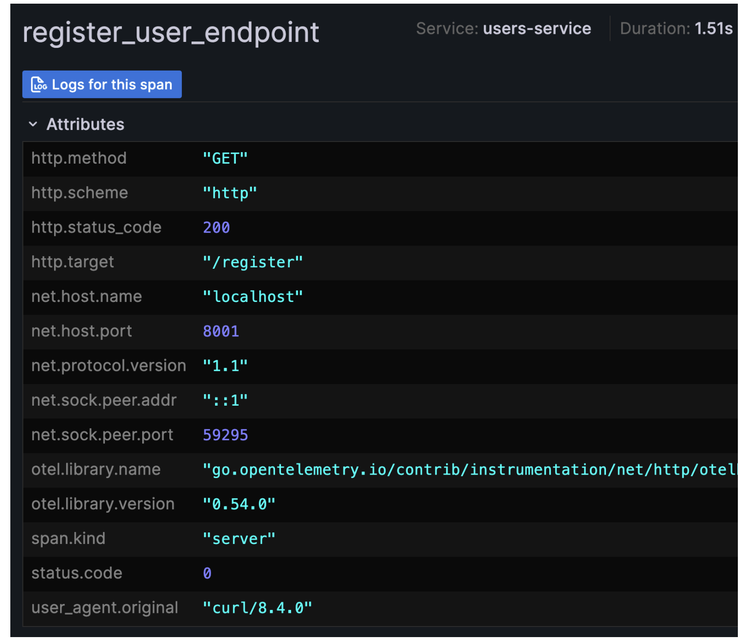
```go
func Logger(ctx context.Context) *zap.Logger {
    fields := make([]zap.Field, 0)
    spanCtx := trace.SpanContextFromContext(ctx)
    if spanCtx.TraceID().IsValid() && spanCtx.SpanID().IsValid() {
        fields = append(fields, zap.String(logging.DefaultTraceIDKey, spanCtx.TraceID().String()))
        fields = append(fields, zap.String(logging.DefaultSpanIDKey, spanCtx.SpanID().String()))
    }

    logger, ok := ctx.Value(loggerCtxKey).(*zap.Logger)
    if ok {
        return logger.With(fields...)
    }

    return logging.GetLogger().With(fields...)
}
```

HTTP UTILS

```go
func InstrumentHTTP(h http.Handler, operation string) http.Handler {
    return otelhttp.NewHandler(h, fmt.Sprintf("%s_endpoint", operation))
}

func HTTPMiddleware(h http.Handler) http.Handler {
    return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
        ctx := r.Context()
        carrier := propagation.HeaderCarrier(w.Header())
        otel.GetTextMapPropagator().Inject(ctx, carrier)
        h.ServeHTTP(w, r)
    })
}

type HTTPTransport struct {
    Transport http.RoundTripper
}

func (t *HTTPTransport) RoundTrip(req *http.Request) (*http.Response, error) {
    otel.GetTextMapPropagator().Inject(req.Context(), propagation.HeaderCarrier(req.Header))
    return t.Transport.RoundTrip(req)
}
```

INJECT TRACE HEADERS IN ALL HTTP RESPONSES

INJECT TRACE HEADERS IN ALL HTTP REQUESTS




```go
func instrument(h http.HandlerFunc, operation string) http.Handler {
    return tracing.InstrumentHTTP(tracing.HTTPMiddleware(h), operation)
}
```

SPAN UTILS

```go
func StartHTTP(ctx context.Context, httpServiceName, traceName string) (context.Context, trace.Span) {
    tr := otel.Tracer(traceName)
    attributes := []attribute.KeyValue{
        operationNameAttribute(traceName),
        attribute.String(HTTPAttributeKey, httpServiceName),
    }

    ctx, span := tr.Start(ctx, traceName, trace.WithSpanKind(trace.SpanKindClient))
    span.SetAttributes(attributes...)

    return ctx, span
}
```

```go
func StartPostgres(ctx context.Context, traceName string) (context.Context, trace.Span) {
    tr := otel.Tracer(traceName)
    attributes := []attribute.KeyValue{
        operationNameAttribute(traceName),
        attribute.String(DBAttributeKey, PostgresAttributeValue),
    }

    ctx, span := tr.Start(ctx, traceName, trace.WithSpanKind(trace.SpanKindClient))
    span.SetAttributes(attributes...)

    return ctx, span
}
```

```go
func RecordError(ctx context.Context, err error, description string) {
    span := trace.SpanFromContext(ctx)
    span.SetStatus(codes.Error, description)
    if err != nil {
        attr := trace.WithAttributes(semconv.ExceptionStacktraceKey.String(string(debug.Stack())))
        span.RecordError(err, attr)
    }
}
```

NOTIFICATION SVC

```go
func register(
    registerer userRegisterer,
    notifier userNotifier,
) func(w http.ResponseWriter, r *http.Request) {
    return func(w http.ResponseWriter, r *http.Request) {
        ctx := r.Context()
        logger := sharedContext.Logger(ctx)

        userID, err := registerer.Register(ctx)
        if err != nil {
            logger.Error("could not register user", zap.Error(err))
            w.WriteHeader(http.StatusInternalServerError)
            return
        }

        if err = notifier.Notify(ctx, userID); err != nil {
            logger.Error("could not notify user", zap.Error(err))
            w.WriteHeader(http.StatusInternalServerError)
            return
        }

        logger.Info("user successfully registered")
        w.WriteHeader(http.StatusOK)
    }
}
```

```go
type userRegisterer interface {
    Register(context.Context) (string, error)
}

type userNotifier interface {
    Notify(ctx context.Context, userID string) error
}
```

SERVICE LOGIC LAYER

NOTIFICATION HTTP CLIENT

```go
func (s *Users) Register(ctx context.Context) (string, error) {
    // omitting the data layer in here for brevity
    _, span := tracing.StartPostgres(ctx, "register_user_txn")
    defer span.End()

    userID := uuid.New().String()
    attributes := trace.WithAttributes(attribute.String("user_id", userID))
    span.AddEvent("create user identity", attributes)
    time.Sleep(200 * time.Millisecond)
    // code for creating the user identity

    span.AddEvent("create user profile", attributes)
    time.Sleep(300 * time.Millisecond)
    // code for creating a user profile

    return userID, nil
}
```

NOTIFICATION SVC

```go
func notify() func(w http.ResponseWriter, r *http.Request) {
    return func(w http.ResponseWriter, r *http.Request) {
        // omitting the service layer in here for brevity
        ctx := r.Context()
        userID := r.URL.Query().Get("user_id")
        logger := sharedContext.Logger(ctx).With(zap.String("user_id", userID))

        _, span := tracing.StartHTTP(ctx, "mailgun_service", "notify_user")
        defer span.End()
        span.SetAttributes(attribute.String("user_id", userID))

        // simulate sending the notification
        time.Sleep(time.Second)

        logger.Info("successfully notified user")
        w.WriteHeader(http.StatusOK)
    }
}
```

ADVICE

- START RECORDING TRACES
- ONLY TRACE WHAT MAKES SENSE
- MINIMISE ATTRIBUTES NOISE
- MINIMISE EVENTS NOISE
- DON'T BREAK THE BANK, SF IS BETTER

PROTOCOL

PROPRIETARY

CUSTOM FORMAT

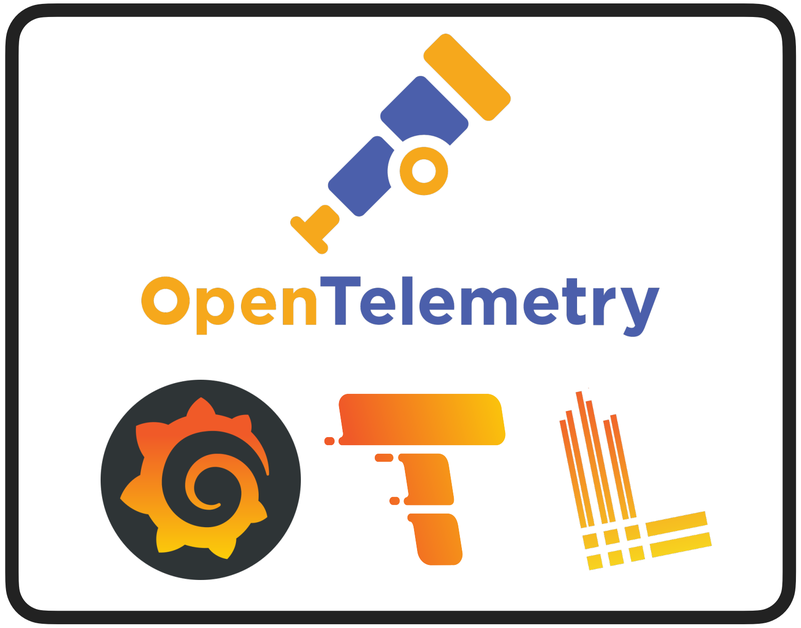
EXTRA CLOUD COSTS

DATADOG

OPEN SOURCE

STANDARD FORMAT

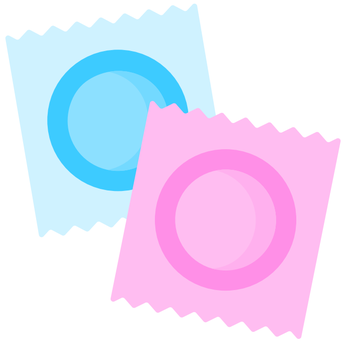
CLOUD/SF SOLUTION



CONTINUOUS PROFILING

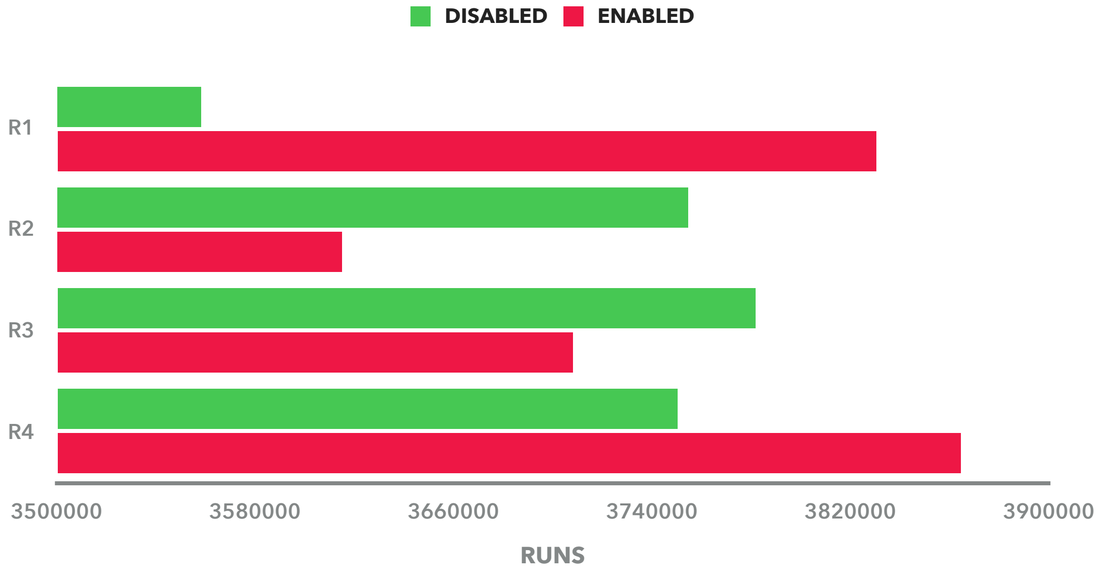
PROFILING IN PROD

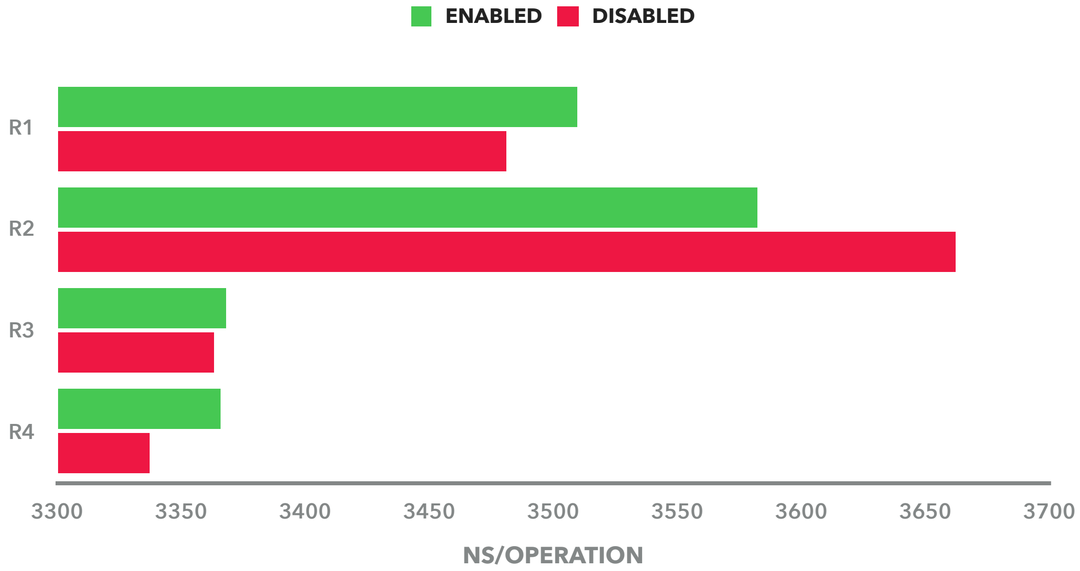
```go
func router(withProfiling bool) http.Handler {
    mux := http.NewServeMux()

    // publicly exposed routes available on the ingress controller
    mux.HandleFunc("/v1/test", func(w http.ResponseWriter, r *http.Request) {
        req := r.URL.Query().Get("req")
        log.Printf("test %s\n", req)
    })

    if !withProfiling {
        return mux
    }

    // internal routes only available via port forwarding
    mux.HandleFunc("pprof/", pprof.Index)
    mux.HandleFunc("pprof/cmdline", pprof.Cmdline)
    mux.HandleFunc("pprof/profile", pprof.Profile)
    mux.HandleFunc("pprof/symbol", pprof.Symbol)
    mux.HandleFunc("pprof/trace", pprof.Trace)
    mux.Handle("pprof/goroutine", pprof.Handler("goroutine"))
    mux.Handle("pprof/heap", pprof.Handler("heap"))
    mux.Handle("pprof/allocs", pprof.Handler("allocs"))
    mux.Handle("pprof/mutex", pprof.Handler("mutex"))
    mux.Handle("pprof/threadcreate", pprof.Handler("threadcreate"))
    mux.Handle("pprof/block", pprof.Handler("block"))

    return mux
}
```



SAFE TO RUN IT IN PROD

ENABLED VS DISABLED





~1-5%

SLOWER

ENABLED VS DISABLED

```
run benchmark | debug benchmark
func BenchmarkRouterProfilingDisabled(b *testing.B) {
    handler := router(false)
    b.ReportAllocs()
    b.ResetTimer()
    for i := 0; i < b.N; i++ {
        w := httptest.NewRecorder()
        req := httptest.NewRequest("GET", fmt.Sprintf("/test?req=%d", i), nil)
        handler.ServeHTTP(w, req)
    }
}

run benchmark | debug benchmark
func BenchmarkRouterProfilingEnabled(b *testing.B) {
    handler := router(true)
    b.ReportAllocs()
    b.ResetTimer()
    for i := 0; i < b.N; i++ {
        w := httptest.NewRecorder()
        req := httptest.NewRequest("GET", fmt.Sprintf("/test?req=%d", i), nil)
        handler.ServeHTTP(w, req)
    }
}
```

PPROF CPU

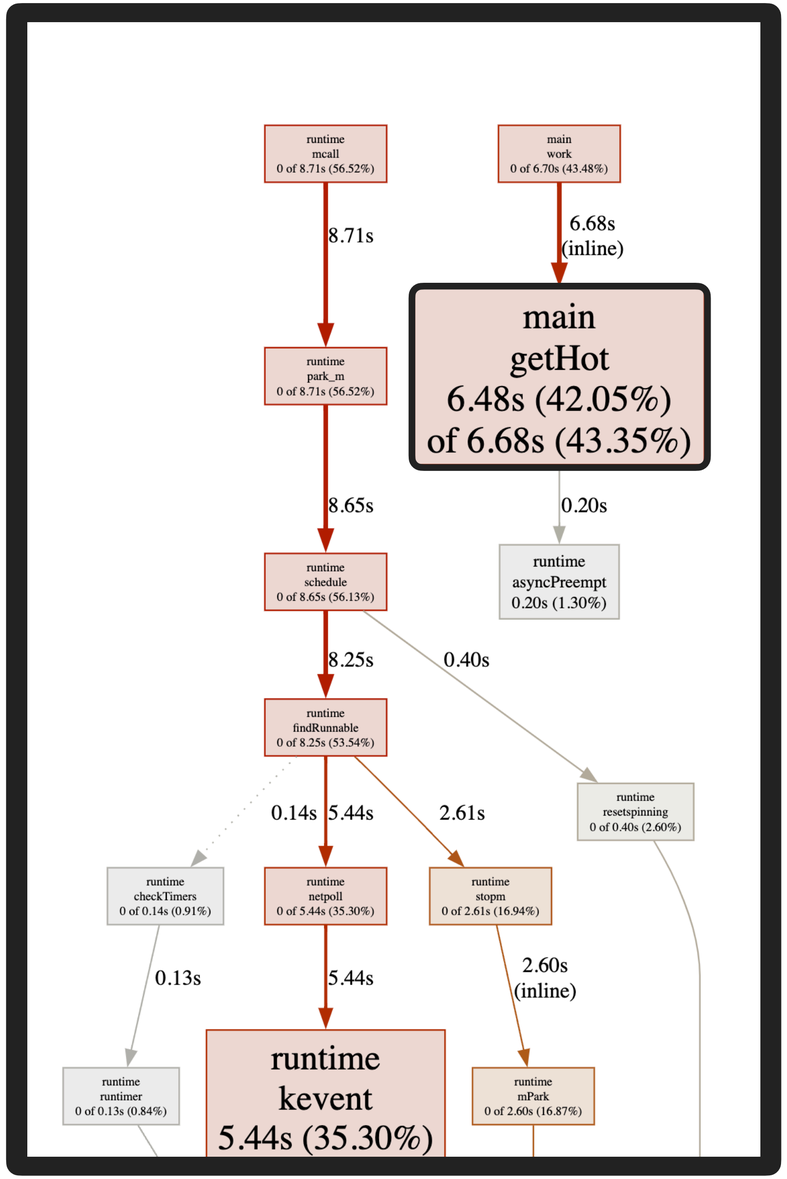
```go
func main() {
    go func() {
        log.Fatalln(http.ListenAndServe(":8080", pprofRouter()))
    }()

    for i := 0; i < 100; i++ {
        go work()
        time.Sleep(time.Second)
    }

    time.Sleep(10 * time.Minute)
}
```

```go
func work() {
    for {
        getHot()
        time.Sleep(100 * time.Millisecond)
    }
}

func getHot() {
    sum := 0
    for i := 0; i < 1e6; i++ {
        sum += i
    }
}
```

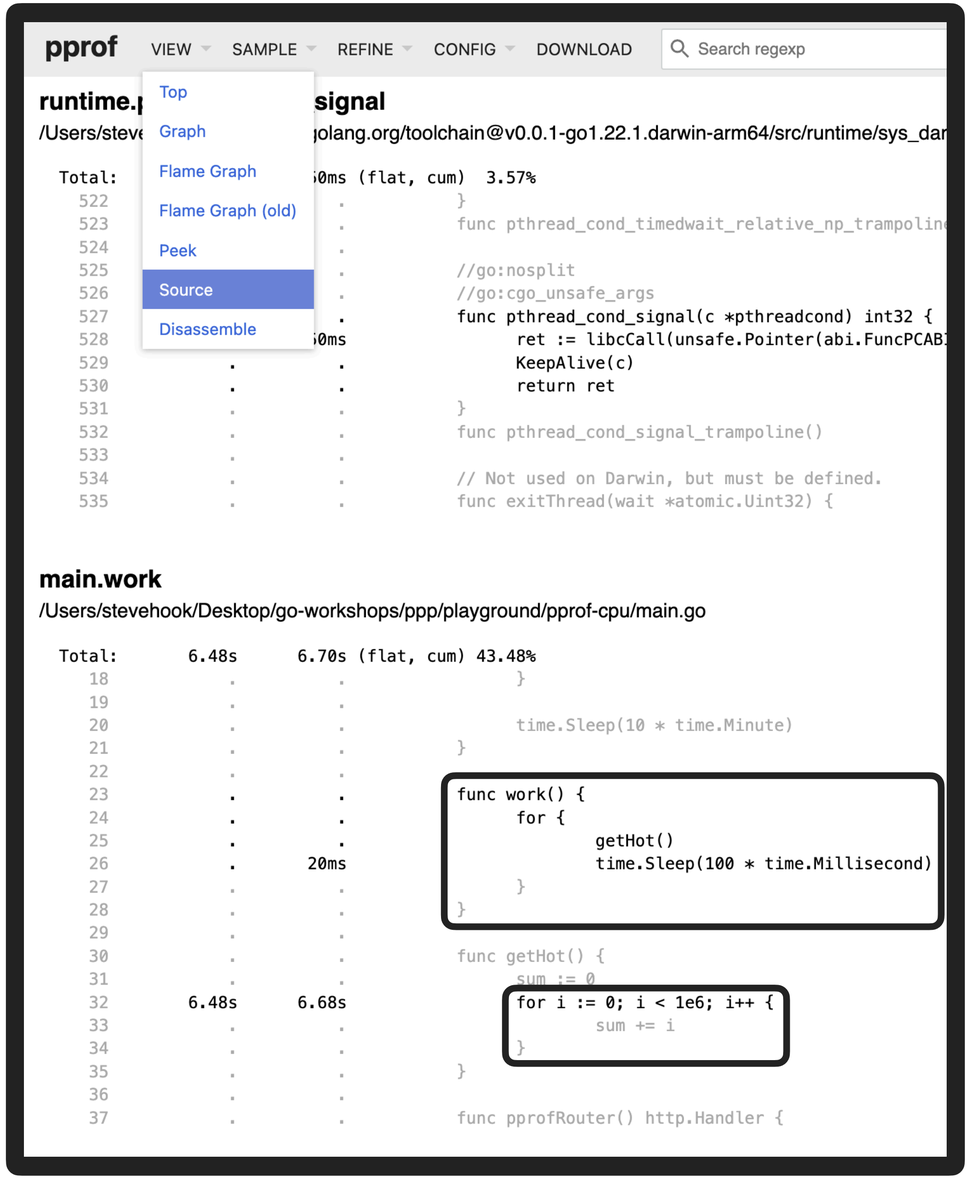
```
go tool pprof -http=:9000 "localhost:8080/pprof/profile?seconds=30"
```

PPROF CPU

GRAPH



SOURCE



PPROF MEMORY

LEAK

```go
package leak
```

```go
type Resource struct {
    id   int
    data []byte
}
```

```go
func Process() {
    for i := 0; i < 1000; i++ {
        r := &Resource{
            id:   i,
            data: make([]byte, 10*1024*1024), // 10MiB
        }
        go func(res *Resource) {
            for {
                time.Sleep(1 * time.Second)
                fmt.Println("Working on:", res.id)
            }
        }(r)

        time.Sleep(time.Second)
    }
}
```

NEVER EXITS

```
go tool pprof -http=:9000 "localhost:8080/pprof/heap"
```

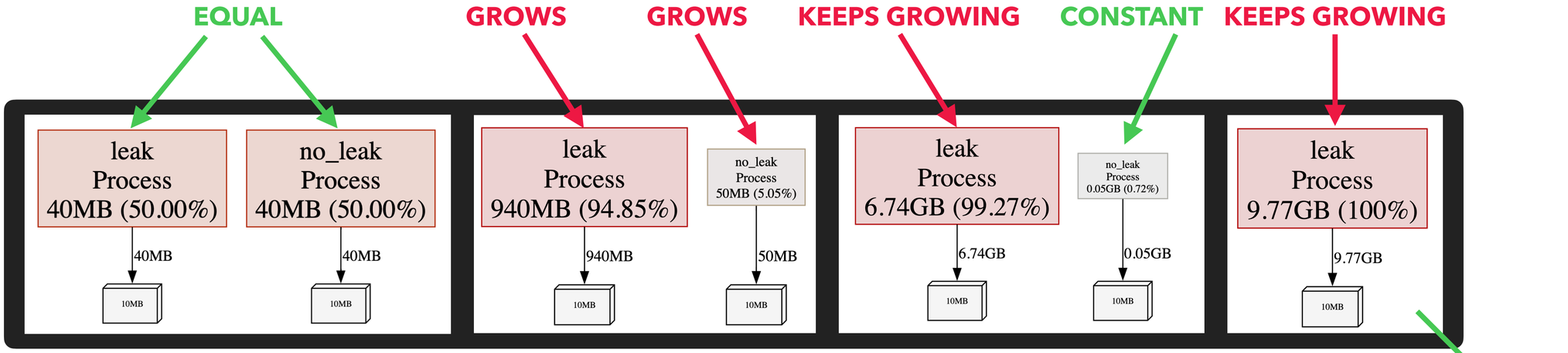
NO LEAK

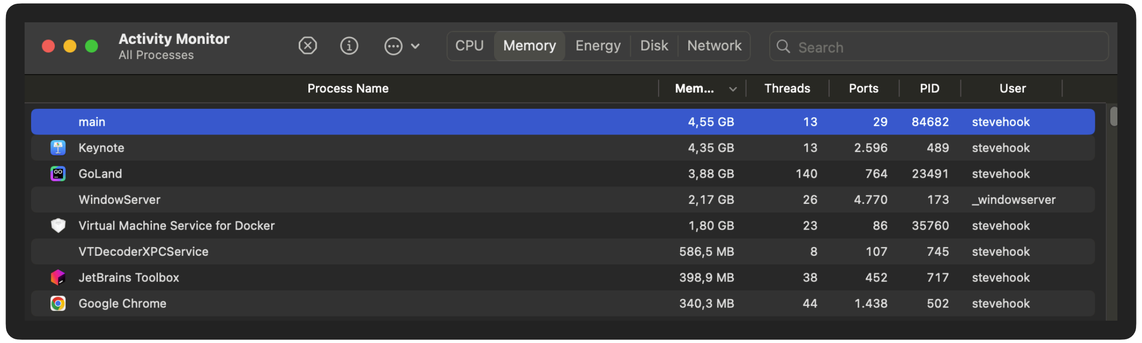
```go
package no_leak
```

```go
func Process() {
    for i := 0; i < 1000; i++ {
        ctx, cancel := context.WithTimeout(
            context.Background(),
            5*time.Second,
        )
        r := &Resource{
            id:   i,
            data: make([]byte, 10*1024*1024), // 10MiB
        }
        go func(res *Resource, ctx context.Context) {
            defer cancel()
            for {
                select {
                case <-ctx.Done():
                    fmt.Println("Done with:", res.id)
                    return
                default:
                    time.Sleep(1 * time.Second)
                    fmt.Println("Working on:", res.id)
                }
            }
        }(r, ctx)
        time.Sleep(time.Second)
    }
}
```

EXITS AFTER TIMEOUT

PPROF MEMORY





~30m

~10GiB

PPROF MEMORY

```
package workers
```

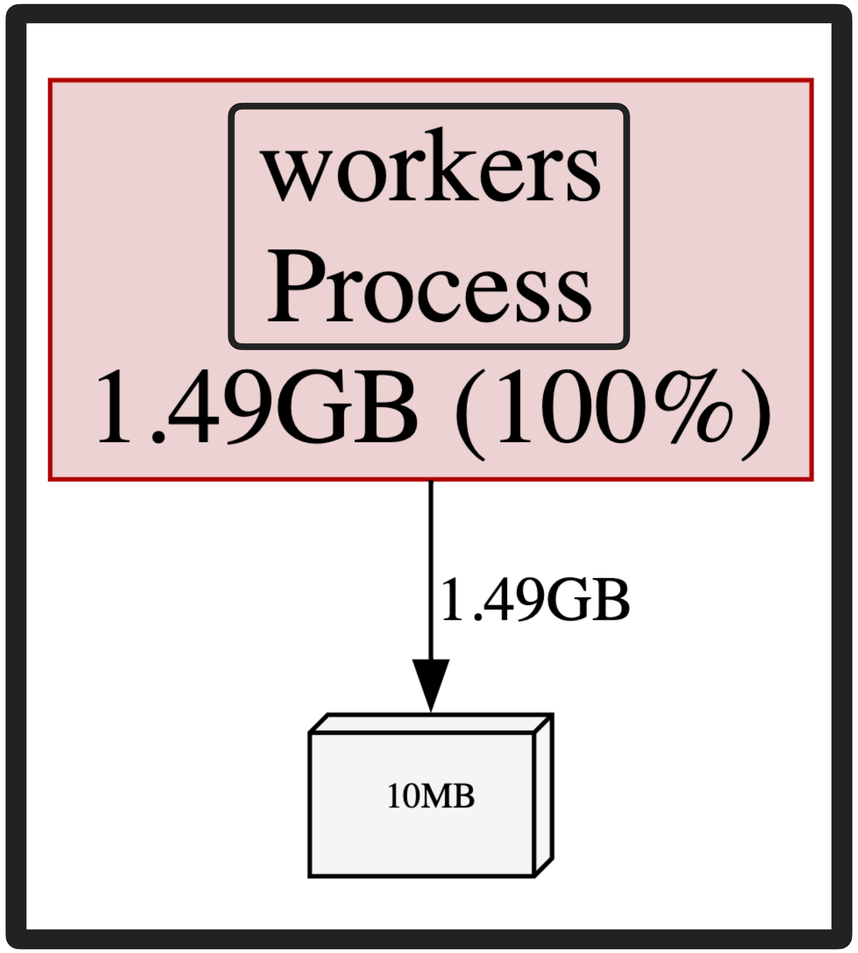
```
type Resource struct {
    id   int
    data []byte
}

func Process(worker func(context.Context, *Resource)) {
    for i := 0; i < 1000; i++ {
        ctx, cancel := context.WithTimeout(
            context.Background(),
            5*time.Second,
        )
        r := &Resource{
            id:   i,
            data: make([]byte, 10*1024*1024), // 10MiB
        }

        go func() {
            defer cancel()
            worker(ctx, r)
        }()
        time.Sleep(time.Second)
    }
}
```

```
func Leak(_ context.Context, res *Resource) {
    for {
        time.Sleep(time.Second)
        fmt.Println("Working on:", res.id)
    }
}
```

```
func NoLeak(ctx context.Context, res *Resource) {
    for {
        select {
        case <-ctx.Done():
            fmt.Println("Done with:", res.id)
            return
        default:
            time.Sleep(1 * time.Second)
            fmt.Println("Working on:", res.id)
        }
    }
}
```

PPROF MEMORY

CONFUSED!?



pprof VIEW SAMPLE REFINE CONFIG DOWNLOAD Search regexp

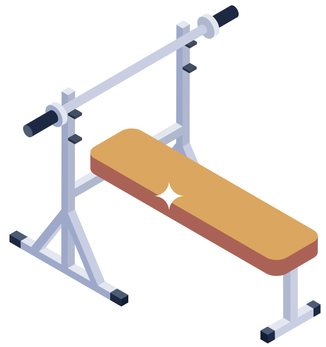
github.com/go-workshops/ppp/playground/pprof-mem-leak-subtle/workers.Process

/Users/stevehook/Desktop/go-workshops/ppp/playground/pprof-mem-leak-subtle/workers/workers.go

```
Total:      1.49GB     1.49GB (flat, cum)   100%
     8            .          .           type Resource struct {
     9            .          .                   id   int
    10            .          .                   data []byte
    11            .          .           }
    12            .          .
    13            .          .           func Process(worker func(context.Context, *Resource)) {
    14            .          .                   for i := 0; i < 1000; i++ {
    15            .          .                           ctx, cancel := context.WithTimeout(
    16            .          .                                   context.Background(),
    17            .          .                                   5*time.Second,
    18            .          .                           )
    19            .          .                           r := &Resource{
    20            .          .                                   id:   i,
    21      1.49GB     1.49GB                                   data: make([]byte, 10*1024*1024), // 10MiB
    22            .          .                           }
    23            .          .
    24            .          .                           go func(ctx context.Context, r *Resource) {
    25            .          .                                   defer cancel()
    26            .          .                                   worker(ctx, r)
    27            .          .                           }(ctx, r)
    28            .          .                           time.Sleep(time.Second)
    29            .          .                   }
    30            .          .           }
```

ADVICE

- GOOD PACKAGE SEPARATION
- ELIMINATE NOISY LOGS
- ISOLATE THE PROFILED CONTEXT



BENCHING

BENCHMARK BEFORE & AFTER REFACTORING CODE

MINIMISE ENV NOISE

- CLOSE CPU/MEM INTENSIVE PROGRAMS
- AVOID RUNNING ON LAPTOP BATTERY
- DISABLE THE NETWORK IF POSSIBLE
- RUN ON A DEDICATED MACHINE IF POSSIBLE
- MAKE SMALL PAUSES IN BETWEEN RUNS

MINIMISE CTX NOISE

- EXTRACT THE PROFILED CODE
- REDUCE UNRELATED LOGIC OR I/O
- REDUCE NOISY LOGS

USE STABLE / CONSISTENT INPUT

- USE FIXED DATASETS TO MINIMISE VARIATIONS
- REPLACE NON DETERMINISTIC SOURCES (TIME, NETWORK) WITH FIXED / MOCKED SOURCES
- CONTROL INPUT DATA SIZE & DATA CONSISTENCY ACROSS RUNS

MINIMISE WORK INSIDE BENCH CODE

- LITTLE TO NO SETUP IN THE BENCH CODE
- MOVE THE SETUP FROM BENCHED CODE IN THE BENCHMARK CODE
- ALWAYS RESET THE BENCHMARK TIMER
- REPORT ALLOCATIONS IN THE BENCHMARK

```
b.ReportAllocs()
b.ResetTimer()
```

MORE REPETITIONS & LESS VARIATIONS

```
go test -bench -count=5
```

MORE CONSISTENT RESULTS

```
go test -bench -benchtime=10s
```

MORE WARMUP TIME
AND LESS VARIATIONS

REDUCE GC

```
run benchmark | debug benchmark
func BenchmarkMyFunction(b *testing.B) {
    debug.SetGCPercent(-1)
    defer debug.SetGCPercent(100)

    for i := 0; i < b.N; i++ {
        // MyFunction()
    }
}
```

PPROF BENCH RESULTS

MEMORY

```
go test -bench -memprofile=mem.pprof
```

```
go tool pprof -http=:9000 mem.pprof
```

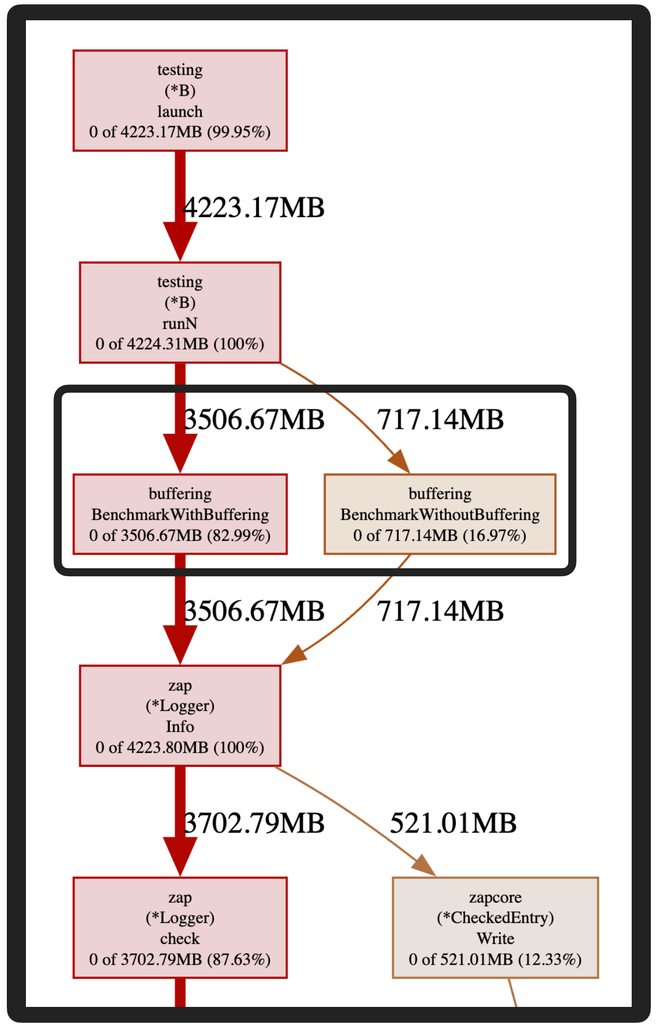
CPU

```
go test -bench -cpuprofile=cpu.pprof
```

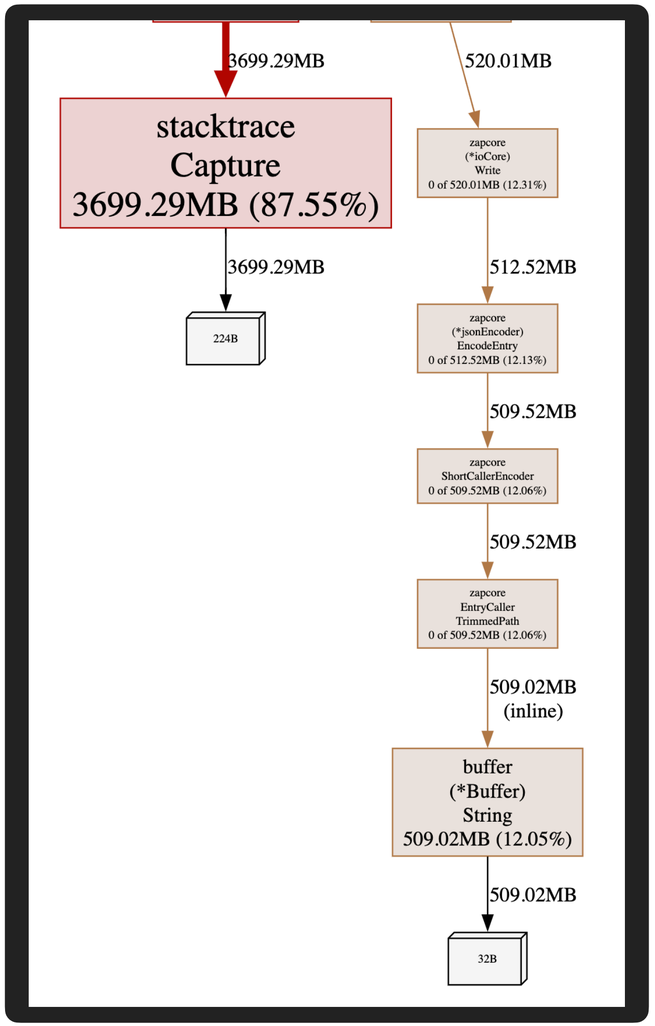
```
go tool pprof -http=:9000 cpu.pprof
```

BUFFERED VS UNBUFFERED BENCHMARK

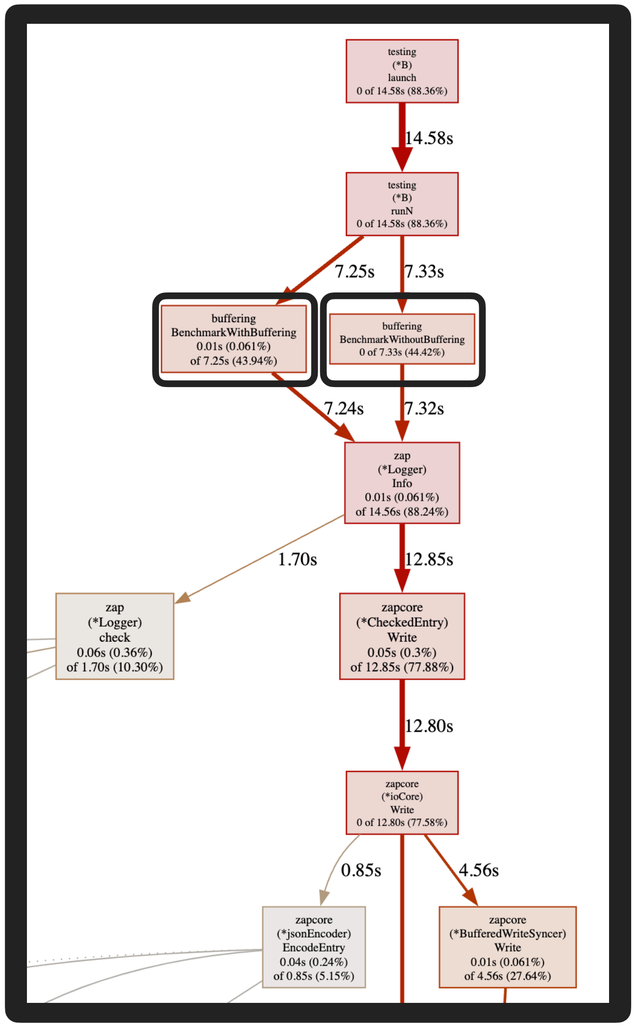
MEMORY

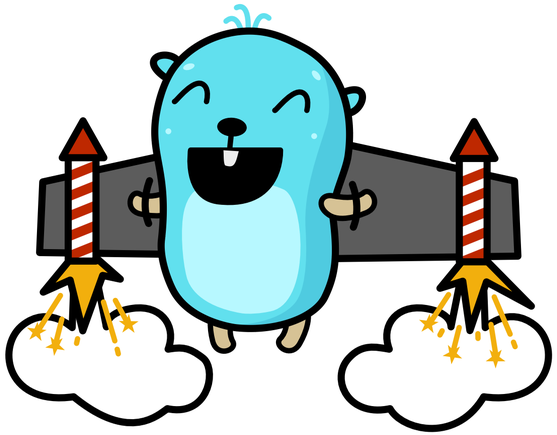


MEMORY



CPU





METRICS

ALERTS

TRACES

LOGS

DEBUG


Q
A