
Designing for Failure
@italolelis
Go Days Berlin 2019

Desining systems for the unexpected!

Go Days Berlin 2019

Think about an airplane, a boing 777 to be especific

Go Days Berlin 2019

ELMS

(Electrical Load Management System)

Go Days Berlin 2019

Essentials

Resilience is a Requirement, Not a Feature

Liang Guo

Dependency Isolation and Graceful Degradation

Go Days Berlin 2019

Health-check and Load Balancing

Go Days Berlin 2019

```go
func main() {
        // Create a new health instance
    h := health.New()

    sqlCheck, err := checkers.NewSQL(&checkers.SQLConfig{
        Pinger: db,
    })
    if err != nil {
        log.Fatalf("could not setup sql checker: %s", err)
    }

    if err = h.AddChecks([]*health.Config{
        {
            Name:     "db-reception-check",
            Checker:  sqlCheck,
            Interval: time.Duration(3) * time.Second,
            Fatal:    true,
        },
        {
            Name:     "amqp-reception-check",
            Checker:  stream.NewChecker(stream.WithDSN(amqpDSN)),
            Interval: time.Duration(3) * time.Second,
            Fatal:    true,
        },
    }); err != nil {
        log.Fatalf("could not add checkers: %s", err)
    }

    if err := h.Start(); err != nil {
        log.Fatalf("could not start health check: %s", err)
    }
}
```

If everything is OK you get...

```
HTTP/1.1 200 OK
Content-Length: 401
Content-Type: application/json
Date: Wed, 16 Jan 2019 19:59:49 GMT

{
    "details": {
        "amqp-reception-check": {
            "check_time": "2019-01-16T20:59:48.928127+01:00",
            "fatal": true,
            "first_failure_at": "0001-01-01T00:00:00Z",
            "name": "amqp-reception-check",
            "num_failures": 0,
            "status": "ok"
        },
        "db-reception-check": {
            "check_time": "2019-01-16T20:59:48.894341+01:00",
            "fatal": true,
            "first_failure_at": "0001-01-01T00:00:00Z",
            "name": "db-reception-check",
            "num_failures": 0,
            "status": "ok"
        }
    },
    "status": "ok"
}
```

If things are not good but your app still can work...

```
HTTP/1.1 200 OK
Content-Length: 507
Content-Type: application/json
Date: Wed, 16 Jan 2019 20:03:28 GMT

{
    "details": {
        "amqp-reception-check": {
            "check_time": "2019-01-16T21:03:27.700167+01:00",
            "error":
"rabbitMQ health check failed on dial phase: dial tcp [::1]:5672: connect: connection refused",
            "first_failure_at": "2019-01-16T21:03:24.702708+01:00",
            "name": "amqp-reception-check",
            "num_failures": 2,
            "status": "failed"
        },
        "db-reception-check": {
            "check_time": "2019-01-16T21:03:27.698874+01:00",
            "fatal": true,
            "first_failure_at": "0001-01-01T00:00:00Z",
            "name": "db-reception-check",
            "num_failures": 0,
            "status": "ok"
        }
    },
    "status": "ok"
}
```

Otherwise...

```
HTTP/1.1 500 Internal Server Error
Content-Length: 588
Content-Type: application/json
Date: Wed, 16 Jan 2019 20:06:03 GMT

{
    "details": {
        "amqp-reception-check": {
            "check_time": "2019-01-16T21:06:03.702271+01:00",
            "error":
"rabbitMQ health check failed on dial phase: dial tcp [::1]:5672: connect: connection refused",
            "first_failure_at": "2019-01-16T21:03:24.702708+01:00",
            "name": "amqp-reception-check",
            "num_failures": 54,
            "status": "failed"
        },
        "db-reception-check": {
            "check_time": "2019-01-16T21:06:03.702259+01:00",
            "error": "dial tcp [::1]:5432: connect: connectionrefused",
            "fatal": true,
            "first_failure_at": "2019-01-16T21:05:12.706956+01:00",
            "name": "db-reception-check",
            "num_failures": 18,
            "status": "failed"
        }
    },
    "status": "failed"
}
```

Self-healing

Container orchestrators do this automatically

Load shedding

Circuit Breakers

```go
func main() {
    // Create a new fallback for when a circuit opens
    fallbackFn := func(err error) error {
        _, err := http.Post("post_to_channel_two")
        return err
    }

    // Create a new hystrix-wrapped HTTP client
    client := hystrix.NewClient(
        hystrix.WithHTTPTimeout(200 * time.Millisecond),
        hystrix.WithCommandName("MyCommand"),
        hystrix.WithErrorPercentThreshold(20),
        hystrix.WithSleepWindow(10),
        hystrix.WithRequestVolumeThreshold(10),
        hystrix.WithFallbackFunc(fallbackFn),
    })
    // Create an http.Request instance
    req, _ := http.NewRequest(http.MethodGet, "http://google.com", nil)

    // Call the `Do` method, which has a similar interface to the `http.Do` method
    res, err := client.Do(req)
    if err != nil { panic(err) }
}
```

Circuit Closed

Hosts	581	90th	17ms
Median	6ms	99th	64ms
Mean	10ms	99.5th	114ms

Circuit Closed

Hosts	581	90th	1ms
Median	0ms	99th	34ms
Mean	1ms	99.5th	49ms

Circuit Closed

Hosts	581	90th	22ms
Median	3ms	99th	122ms
Mean	11ms	99.5th	312ms

CryptexDecipher

74,718 | 92 | 0.0 %
0 | 0
58

Host: 12.9/s
Cluster: 7,486.8/s

Circuit Closed

Hosts	581	90th	11ms
Median	3ms	99th	67ms
Mean	10ms	99.5th	405ms

CinematchGetMovieRatings

60,672 | 2 | 0.0 %
0 | 0
0

Host: 10.4/s
Cluster: 6,067.4/s

Circuit Closed

Hosts	581	90th	41ms
Median	14ms	99th	143ms
Mean	21ms	99.5th	215ms

VideoHistoryGetBookmarks

48,281 | 60 | 0.1 %
0 | 0
0

Host: 8.3/s
Cluster: 4,834.1/s

Circuit Closed

Hosts	581	90th	26ms
Median	8ms	99th	104ms
Mean	13ms	99.5th	158ms

VideoMetadataGetEpisodes

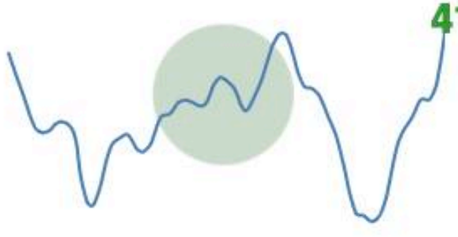
45,663 | 0 | 0.0 %
0

Host: 7.9/s
Cluster: 4,566.3/s

Circuit Closed

Hosts	581	90th	0ms
Median	0ms	99th	0ms
Mean	0ms	99.5th	0ms

GPSGetGroup



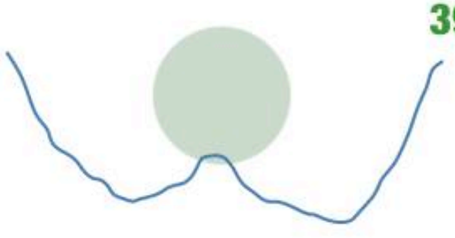
41,142 | 0 | 0.0 %
0 | 0
0

Host: 7.1/s
Cluster: 4,114.2/s

Circuit Closed

Hosts	581	90th	114ms
Median	10ms	99th	318ms
Mean	38ms	99.5th	412ms

SimpleDBGetItem



39,662 | 2 | 0.0 %
0 | 0
0

Host: 6.8/s
Cluster: 3,966.4/s

Circuit Closed

Hosts	581	90th	26ms
Median	14ms	99th	70ms
Mean	17ms	99.5th	123ms

GeoLookupCommand

27,455 | 0 | 0.0 %
0 | 0
9

ABTestGetCellsAsAllocationsList

8,535 | 0 | 0.0 %
0 | 0
0

ABTestGetAllocationsCommand

7,422 | 0 | 0.0 %
0 | 0
0

Retry Logic

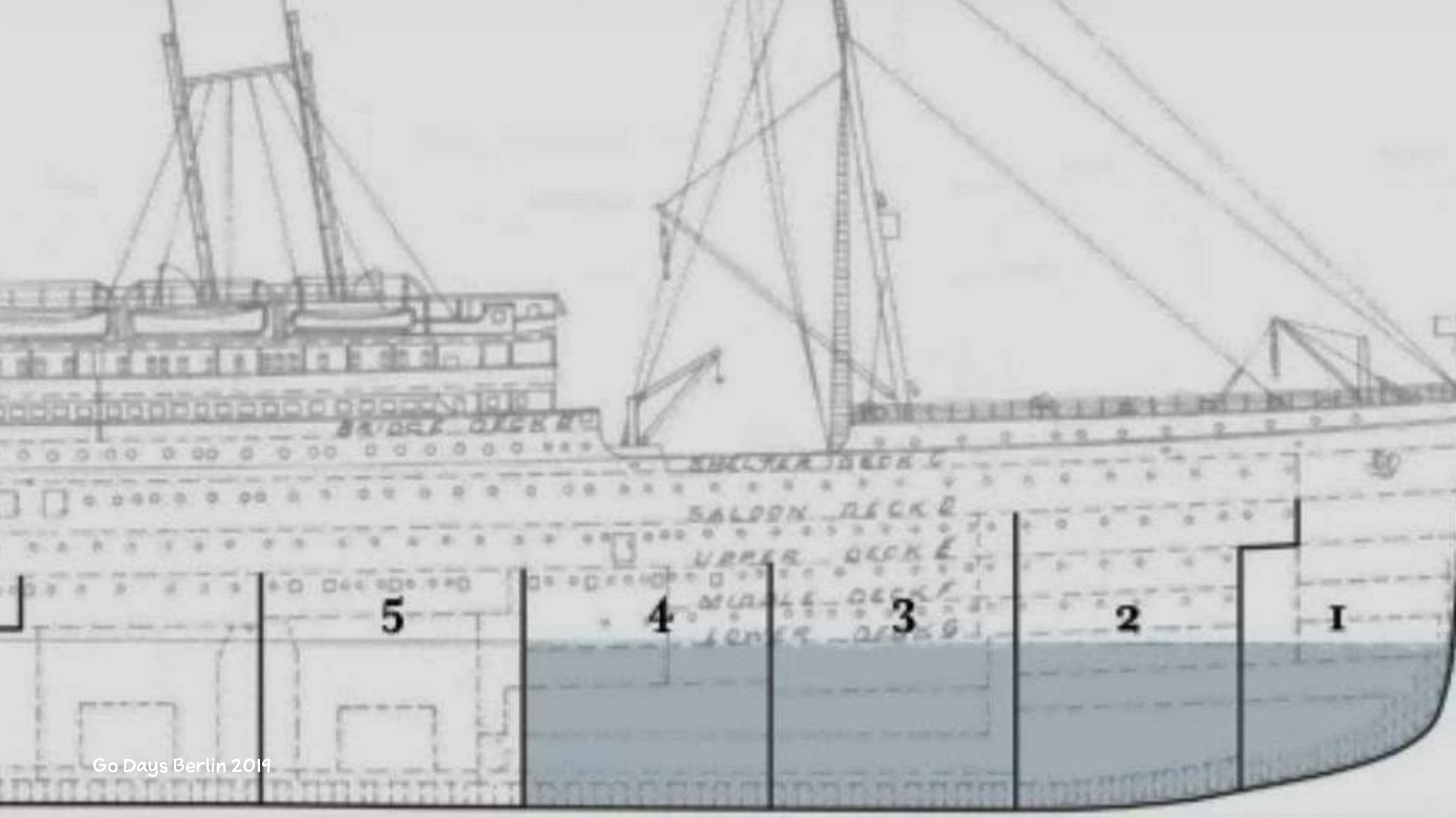
```go
func main() {
    // Exponential Backoff increases the backoff at a exponential rate
    initTimeout := 2*time.Millisecond
    maxTimeout := 10*time.Millisecond
    expFactor := 2
    maxJitterInterval := 2*time.Millisecond

    backoff := heimdall.NewExponentialBackoff(
        initTimeout,
        maxTimeout,
        expFactor,
        maxJitterInterval,
    )

    // Create a new retry mechanism with the backoff
    retrier := heimdall.NewRetrier(backoff)


// Create a new client, sets the retry mechanism, and the number of times you would like to retry
    client := httpclient.NewClient(
        httpclient.WithHTTPTimeout(1000 * time.Millisecond),
        httpclient.WithRetrier(retrier),
        httpclient.WithRetryCount(4),
    )
    // Create an http.Request instance
    req, _ := http.NewRequest(http.MethodGet, "http://google.com", nil)

    // Call the `Do` method, which has a similar interface to the `http.Do` method
    res, err := client.Do(req)
    if err != nil { panic(err) }
}
```

Rate Limiters

```go
func main() {
    rate, err := limiter.NewRateFromFormatted("1000-H")
    if err != nil {
        panic(err)
    }

    store := memory.NewStore()

    // Then, create the limiter instance which takes the store and the rate as arguments.
    // Now, you can give this instance to any supported middleware.
    instance := limiter.New(store, rate)
}
```

Bulkhead


Go Days Berlin 2019


5
4
3
2
1

Outlier Server Host Detection

Go Days Berlin 2019

Outbox Pattern

Go Days Berlin 2019

```go
func main() {
    // ...

    ds, err := postgres.WithInstance(ctx, db.DB)
    if err != nil {
        log.Fatalf("could not create a postgres wrapper", err)
    }

    o, err := outboxer.New(
        outboxer.WithDataStore(ds),
        outboxer.WithEventStream(amqp.NewAMQP(conn)),
        outboxer.WithCheckInterval(1*time.Second),
        outboxer.WithCleanupInterval(5*time.Second),
    )
    defer o.Stop()

    // Start the listeners for sending and cleaning messages
    o.Start()

    // Sends a message
    if err = o.Send(ctx, &outboxer.OutboxMessage{
        Payload: []byte("test payload"),
        Options: map[string]interface{}{
            amqp.ExchangeNameOption: "test",
            amqp.ExchangeTypeOption: "test.send",
        },
    }); err != nil {
        log.Fatalf("could not send message: %s", err)
    }
}
```

Service Mesh

Observability

Defining your SLO's and SLI's

Go Days Berlin 2019

Monitoring

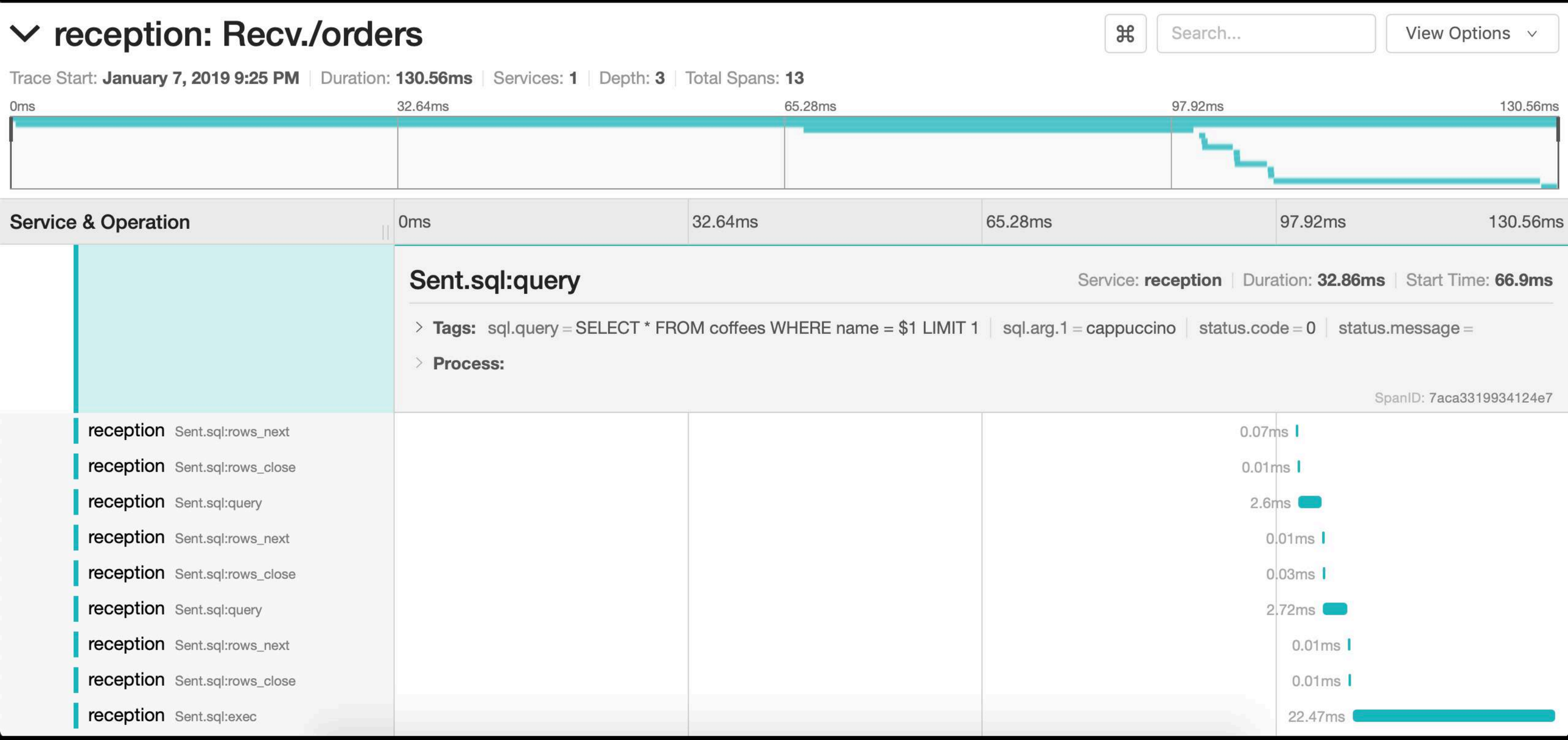
Go Days Berlin 2019

```
if err := view.Register(
        ochttp.ClientSentBytesDistribution,
        ochttp.ClientReceivedBytesDistribution,
        ochttp.ClientRoundtripLatencyDistribution,
    ); err != nil {
        logger.Fatal(err)
    }

    exporter, err := prometheus.NewExporter(prometheus.Options{
        Namespace: cfg.ServiceName,
    })
    if err != nil {
        log.Fatal("failed to create the prometheus stats exporter")
    }
    view.RegisterExporter(exporter)
    view.SetReportingPeriod(cfg.ReportingPeriod)
```

Distributed Tracing

Go Days Berlin 2019

```go
exporter, err := jaeger.NewExporter(jaeger.Options{
    CollectorEndpoint: cfg.CollectorEndpoint,
    Process: jaeger.Process{
        ServiceName: cfg.ServiceName,
    },
})
if err != nil {
    log.Error("could not create the jaeger exporter")
}

trace.RegisterExporter(exporter)
trace.ApplyConfig(trace.Config{DefaultSampler: trace.AlwaysSample()})
```



Go Days Berlin 2019

Open Census

Go Days Berlin 2019

```
import (
    "go.opencensus.io/exporter/prometheus"
    "go.opencensus.io/plugin/ochttp"
    "go.opencensus.io/stats/view"
)
```

Recap

1. Always think about your dependencies
2. Dependency Isolation and Graceful Degradation
3. Load shedding and Request Controlling
4. Observalibility is not optional

Questions and links!

- Example application: https://github.com/italolelis/coffee-shop
- Link to the slides: https://github.com/italolelis/talks


Thank you!
Go Days Berlin 2019