Designing for Failure

@italolelis

Italo Vietro

From 🇧🇷
Living in Berlin 🇩🇪
Working @Lykon 💻

Worked @HelloFresh and @N26

Designing

systems

for the

unexpected


TC-ATB

39,000,000

flights in 2019

8

deadly crashes

233

fatalities

https://aviation-safety.net/statistics/period/barometer3.php

ELMS

(Electrical Load Management System)

Essentials

“Resilience is a Requirement, Not a Feature.”

– Liang Guo

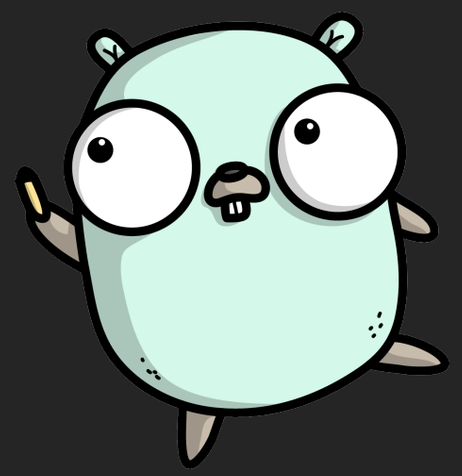
Latency Control

Load Shedding

Isolation

Self Healing

Traffic Control



Latency Control

Load Shedding

Isolation

Self Healing

Traffic Control

Timeouts

Client

Timeouts

Server

Timeouts

```go
func newSimpleTimeout() *http.Client {
    return &http.Client{
        Timeout: 3 * time.Second,
    }
}
```

```go
func newCompleteTimeout() *http.Client {
    return &http.Client{
        Transport: &http.Transport{
            Dial: (&net.Dialer{
                Timeout:   30 * time.Second,
                KeepAlive: 30 * time.Second,
            }).Dial,
            TLSHandshakeTimeout:   10 * time.Second,
            ResponseHeaderTimeout: 10 * time.Second,
            ExpectContinueTimeout: 1 * time.Second,
        },
    }
}
```

```go
func main() {
    ctx, cancel := context.WithTimeout(context.Background(), 5*time.Second)
    defer cancel()

    c := &http.Client{
        Transport: &http.Transport{
            Dial: (&net.Dialer{
                Timeout:   30 * time.Second,
                KeepAlive: 30 * time.Second,
            }).Dial,
        },
    }

    url := "https://httpstat.us/200?sleep=6000"
    req, err := http.NewRequestWithContext(ctx, http.MethodGet, url, nil)
    if err != nil {
        log.Fatalf("failed to create request: %s", err)
    }

    res, err := c.Do(req)
    if err != nil {
        log.Fatal(err)
    }

    fmt.Printf("Code: %d \n", res.StatusCode)
}
```

Client Timeouts

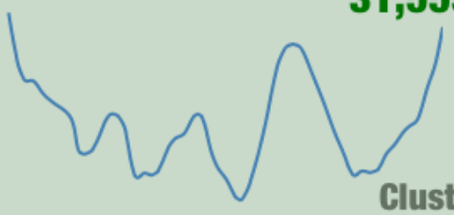
Server Timeouts

```go
func main() {
    h := timeoutHandler{}

    srv := &http.Server{
        ReadTimeout:       5 * time.Second,
        WriteTimeout:      10 * time.Second,
        IdleTimeout:       10 * time.Second,
        ReadHeaderTimeout: 20 * time.Second,
        Handler:           h,
    }

    log.Println(srv.ListenAndServe())
}

type timeoutHandler struct{}

func (h timeoutHandler) ServeHTTP(w http.ResponseWriter, r *http.Request) {
    defer r.Body.Close()
    timer := time.AfterFunc(5*time.Second, func() {
        r.Body.Close()
    })

    bodyBytes := make([]byte, 0)
    for {
        //We reset the timer, for the variable time
        timer.Reset(1 * time.Second)

        _, err := io.CopyN(bytes.NewBuffer(bodyBytes), r.Body, 256)
        if err == io.EOF {
            // This is not an error in the common sense
            // io.EOF tells us, that we did read the complete body
            break
        } else if err != nil {
            //You should do error handling here
            break
        }
    }
}
```

Circuit Breakers

```go
func main() {
    cb := gobreaker.NewCircuitBreaker(gobreaker.Settings{
        Name: "HTTP GET Example",
        ReadyToTrip: func(counts gobreaker.Counts) bool {
            failureRatio := float64(counts.TotalFailures) / float64(counts.Requests)
            return counts.Requests >= 3 && failureRatio >= 0.6
        },
    })

    resp, err := cb.Execute(func() (interface{}, error) {
        resp, err := http.Get(url)
        if err != nil {
            return nil, err
        }

        if resp.StatusCode >= http.StatusBadRequest {
            return resp, errors.New("request failed")
        }

        return resp, nil
    })
}
```

VideoMetadataGetEpisode



31,555,111 | 0 | 0.0 %
0

Host: 5,431.2/s
Cluster: 3,155,511.1/s

Circuit Closed

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

DeviceTypeServiceGetType

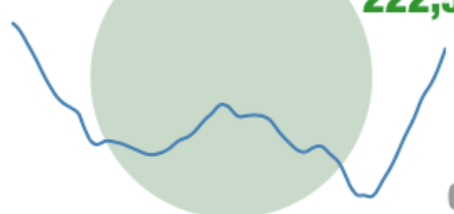
230,574 | 0 | 0.1 %
0 | 194
0

Host: 39.7/s
Cluster: 23,076.8/s

Circuit Closed

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

SubscriberGetAccount



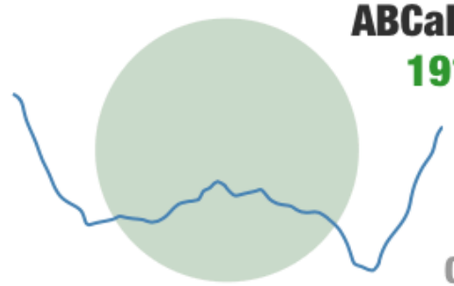
222,584 | 10 | 0.0 %
0 | 104
0

Host: 38.3/s
Cluster: 22,269.8/s

Circuit Closed

Hosts	581	90th	11ms
Median	2ms	99th	51ms
Mean	4ms	99.5th	75ms

ABCallServiceInternal



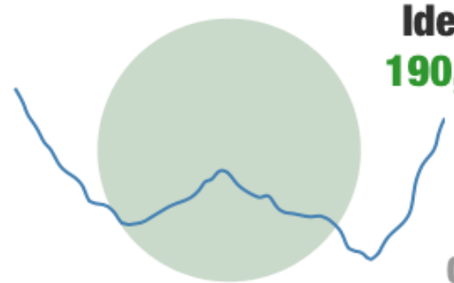
191,390 | 0 | 0.0 %
0 | 5
1

Host: 32.9/s
Cluster: 19,139.6/s

Circuit Closed

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

IdentityCookieAuth



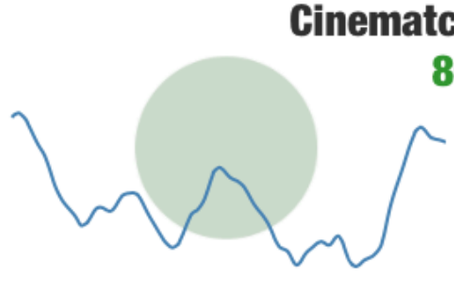
190,804 | 0 | 0.0 %
0 | 50
0

Host: 32.8/s
Cluster: 19,085.4/s

Circuit Closed

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

CinematchGetPredictions



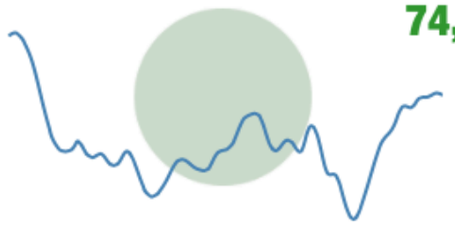
80,105 | 5 | 0.0 %
0 | 0
0

Host: 13.8/s
Cluster: 8,011.0/s

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

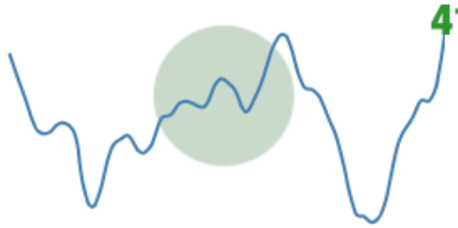
VideoMetadataGetEpisodes

45,663 | 0 | 0.0 %
0

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

Circuit Closed

GPSGetGroup



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

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

Circuit Closed

SimpleDBGetItem

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

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

Circuit Closed

Retries

```go
func main() {
    initalTimeout := 2 * time.Millisecond         // Inital timeout
    maxTimeout := 9 * time.Millisecond            // Max time out
    exponentFactor := 2.0                         // Multiplier
    maximumJitterInterval := 2 * time.Millisecond // Max jitter interval. It must be more than 1*time.Millisecond

    backoff := heimdall.NewExponentialBackoff(initalTimeout, maxTimeout, exponentFactor, maximumJitterInterval)

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

    // Create a new hystrix-wrapped HTTP client with the fallbackFunc as fall-back function
    client := hystrix.NewClient(
        hystrix.WithHTTPTimeout(10*time.Second),
        hystrix.WithCommandName("MyCommand"),
        hystrix.WithHystrixTimeout(10*time.Second),
        hystrix.WithMaxConcurrentRequests(100),
        hystrix.WithErrorPercentThreshold(20),
        hystrix.WithSleepWindow(10),
        hystrix.WithRequestVolumeThreshold(10),
        hystrix.WithRetrier(retrier),
        hystrix.WithRetryCount(3),
    )

    statusCodes := []int{200, 400, 500}
    for i := 0; i <= 50; i++ {
        url := fmt.Sprintf("https://httpstat.us/%d", statusCodes[rand.Intn(len(statusCodes))])
        fmt.Printf("GET %s \n", url)

        if err := get(client, url); err != nil {
            fmt.Printf("failed: %s \n", err)
        }

        fmt.Println("success")
    }
}
```

Latency Control

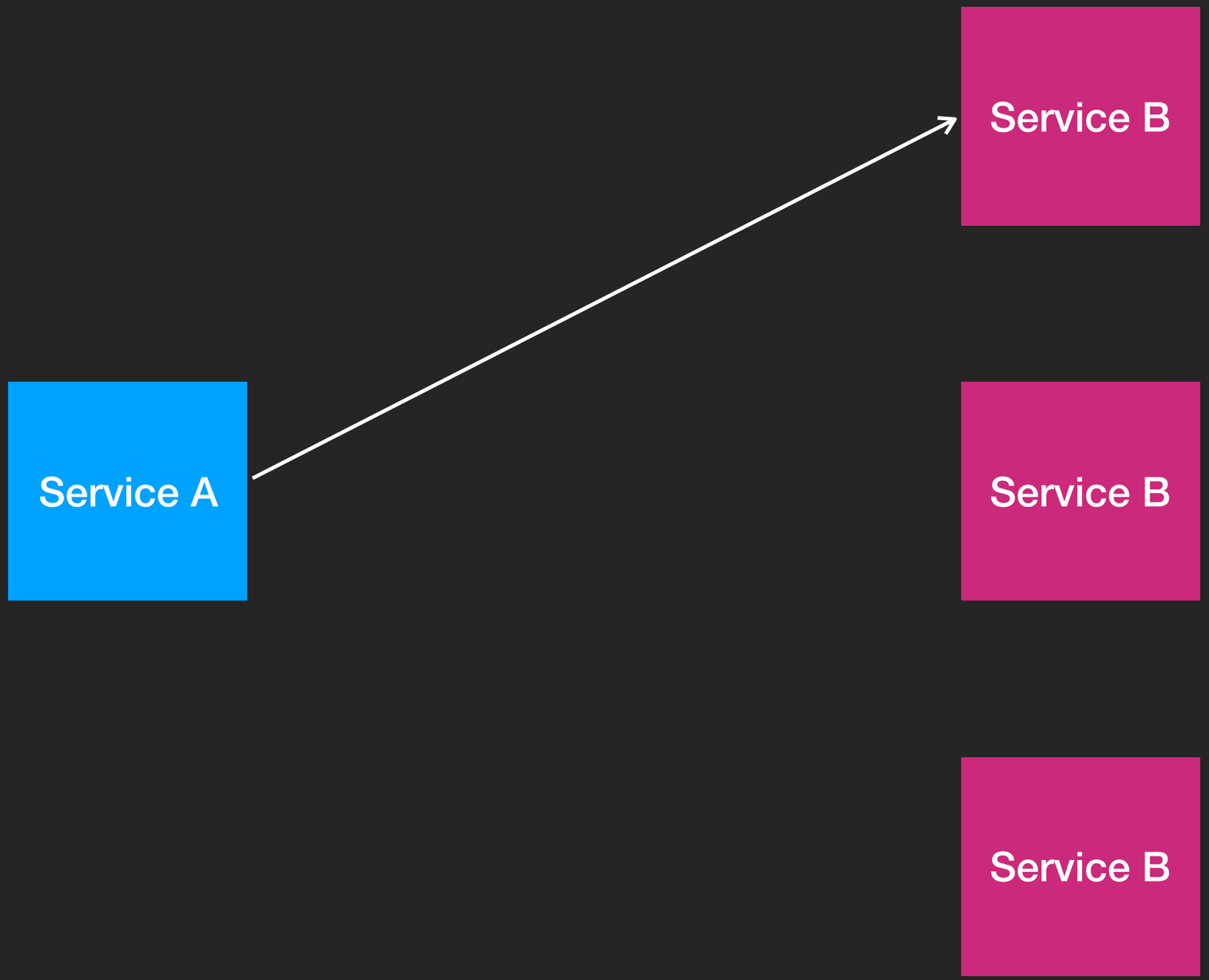
Load Shedding

Isolation

Self Healing

Traffic Control

Load Balancing


Service A
Service B
Service B
Service B

Service B

Service A

Service B

Service B

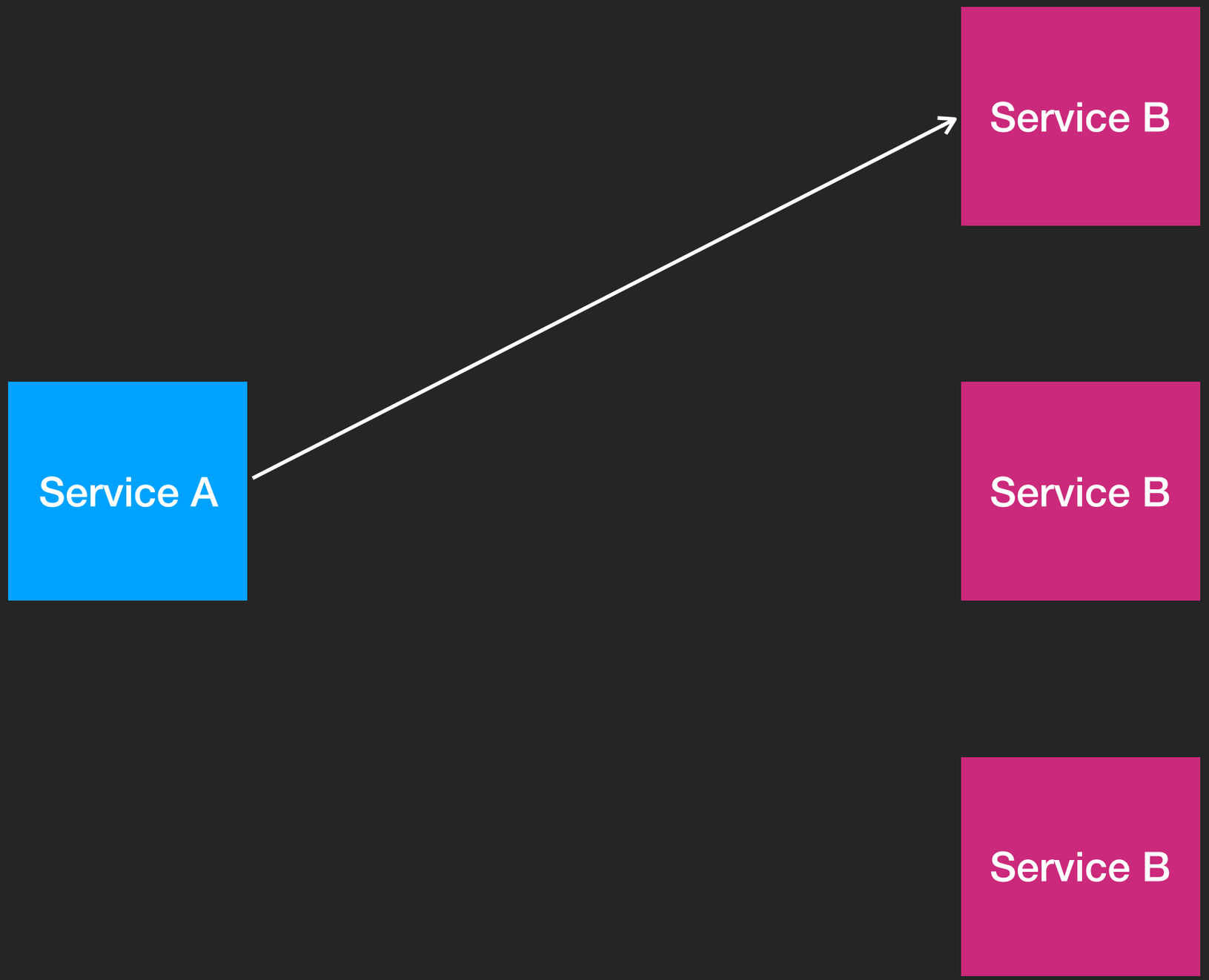
Service B

Service A

Service B

Service B

Outlier Server Host Detection

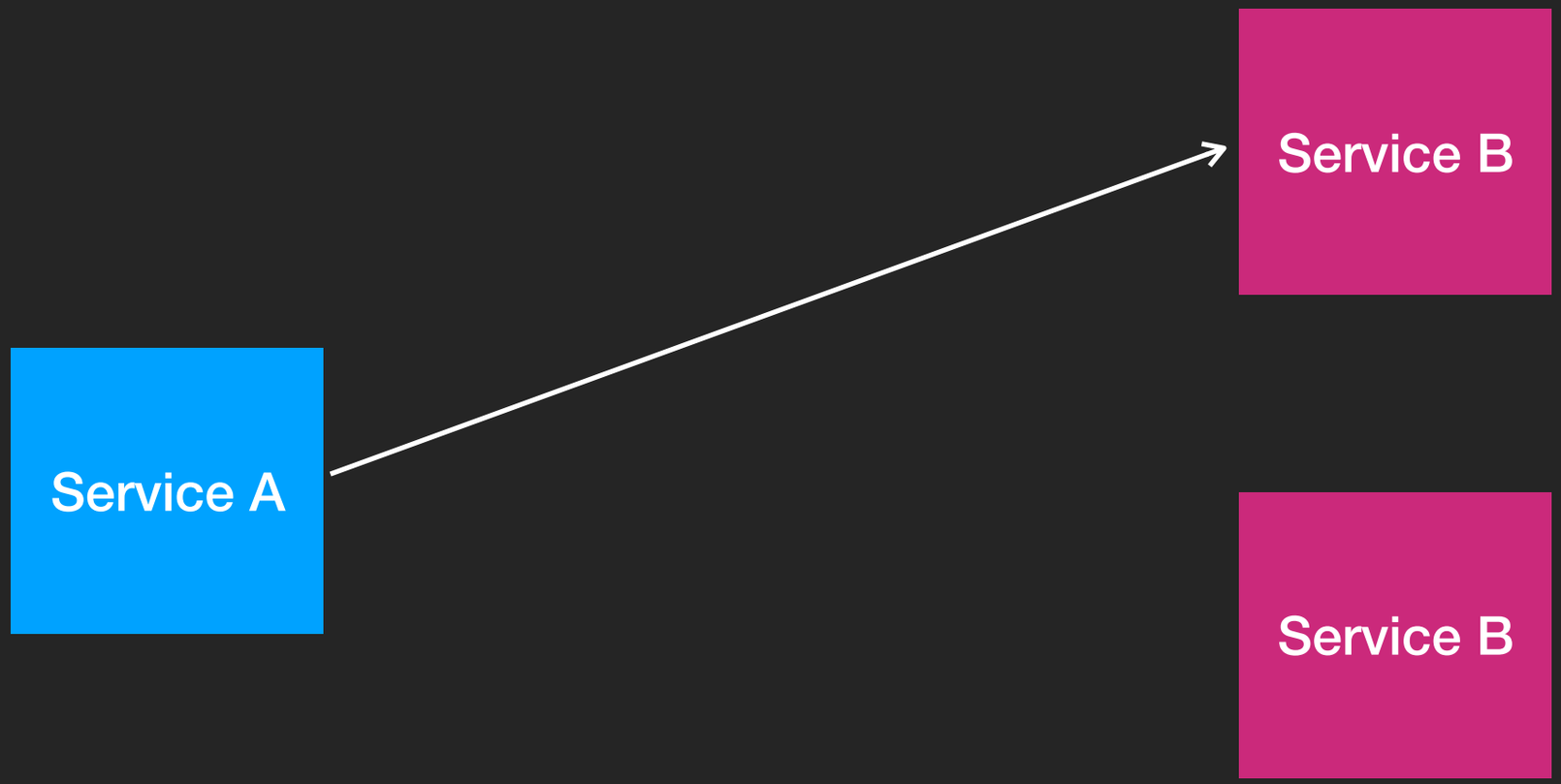

Service A
Service B
Service B
Service B

Service B

Service A

Service B

Service B


Service B
Service A
Service B

Service B

Service A

Service B

Latency Control

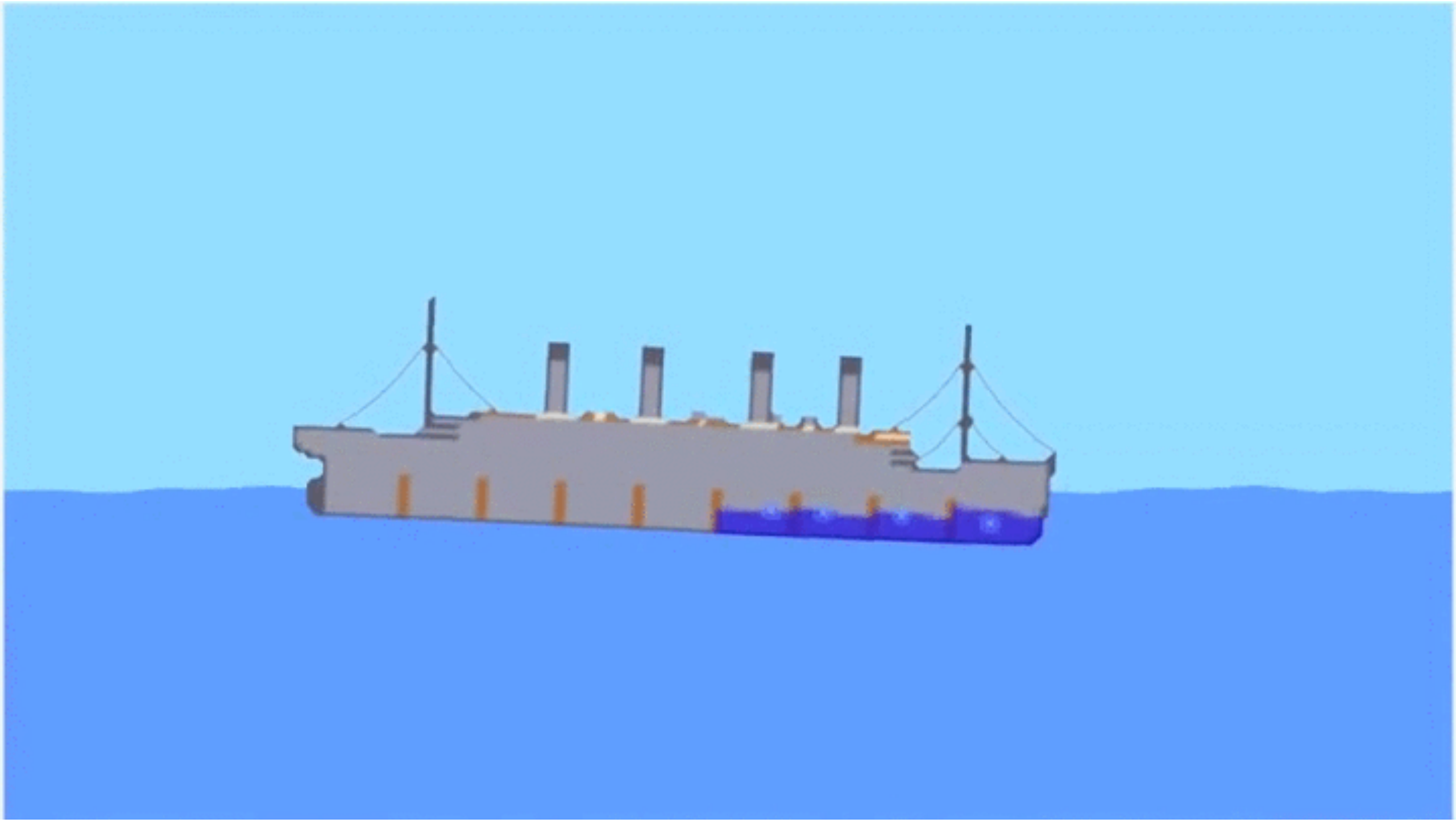
Load Shedding

Isolation

Self Healing

Traffic Control

Bulkhead



Latency Control

Load Shedding

Isolation

Self Healing

Traffic Control

Health Checks

```go
func healthCheckers(cfg *config.Config, db checkers.SQLPinger) (*health.Health, error) {
    h := health.New()

    dbChecker, err := checkers.NewSQL(&checkers.SQLConfig{
        Pinger: db,
    })
    if err != nil {
        return nil, err
    }

    googleChecker, err := createHTTPCheck("https://google.com")
    if err != nil {
        return nil, fmt.Errorf("failed to create http checker: %w", err)
    }

    if err := h.AddChecks([]*health.Config{
        {
            Name:     "db",
            Checker:  dbChecker,
            Interval: time.Duration(3) * time.Second,
            Fatal:    true,
        },
        {
            Name:     "google",
            Checker:  googleChecker,
            Interval: time.Duration(2) * time.Second,
            Fatal:    false,
        },
    }); err != nil {
        return nil, err
    }

    return h, nil
}
```

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"
}
```

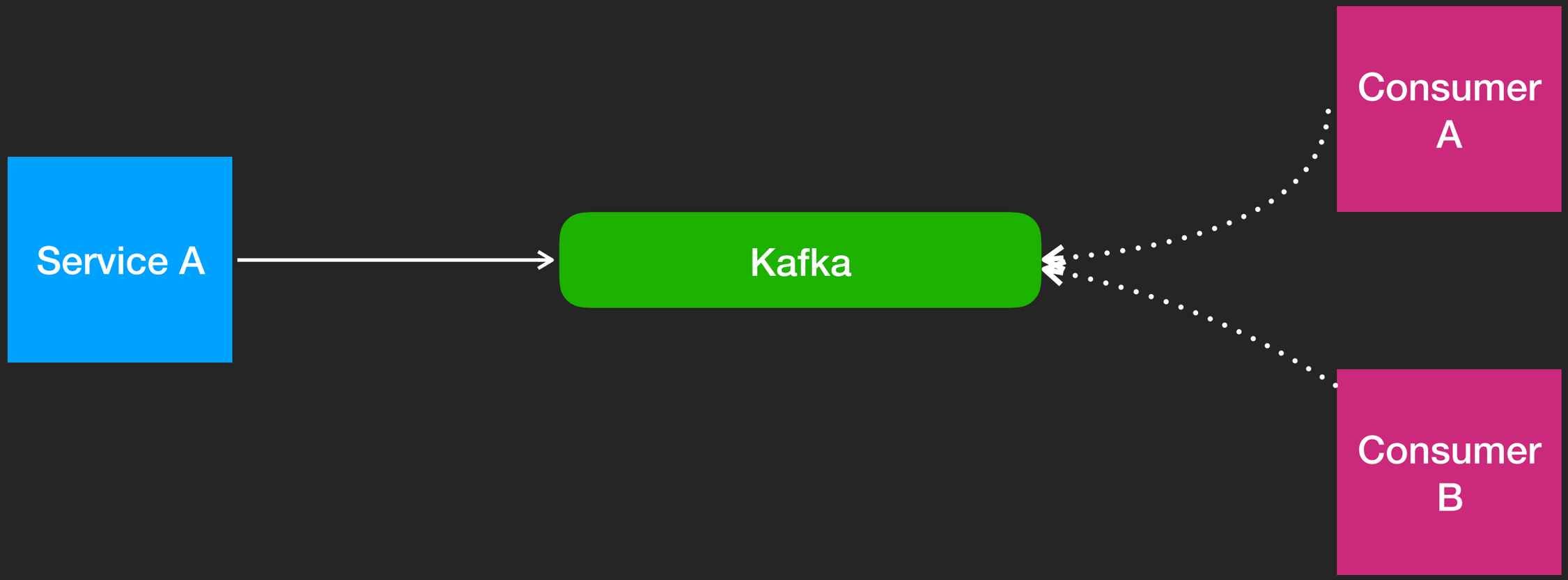
If things are not good but your app still can work...

```
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"
}
```

Outbox Pattern


Service A
Kafka
Consumer A
Consumer B

Service A

Consumer A

Consumer B

Consumer A

Service A

Kafka

Consumer B

Sender

```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)
    }
}
```

https://github.com/italolelis/outboxer

Latency Control

Load Shedding

Isolation

Self Healing

Traffic Control

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)
}
```

Latency Control

Load Shedding

Isolation

Self Healing

Traffic Control

Latency Control

Load Shedding

Isolation

Self Healing

Traffic Control

Latency Control

Load Shedding

Isolation

Self Healing

Traffic Control

Latency Control

Load Shedding

Isolation

Self Healing

Traffic Control

Latency Control

Load Shedding

Isolation

Self Healing

Traffic Control

Latency Control

Load Shedding

Isolation

Self Healing

Traffic Control

Too much...

1000

Services

Service Mesh

Istio

+

kubernetes

Show me some code...

Observability

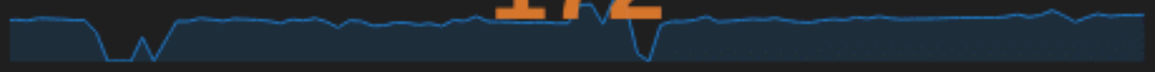
Metrics

```
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)
```

Big Dashboard

Logins

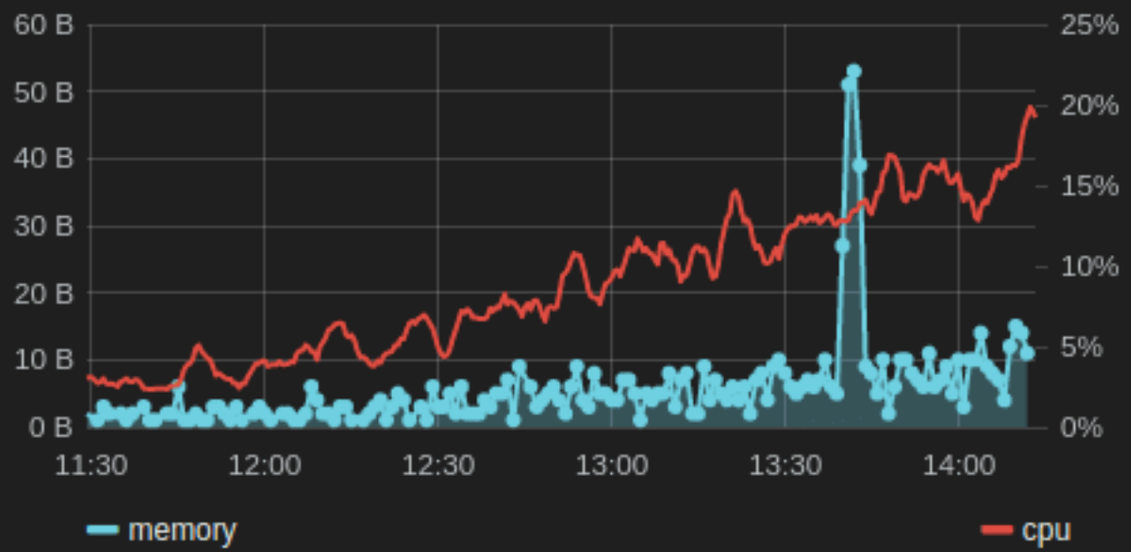
172



Sign ups

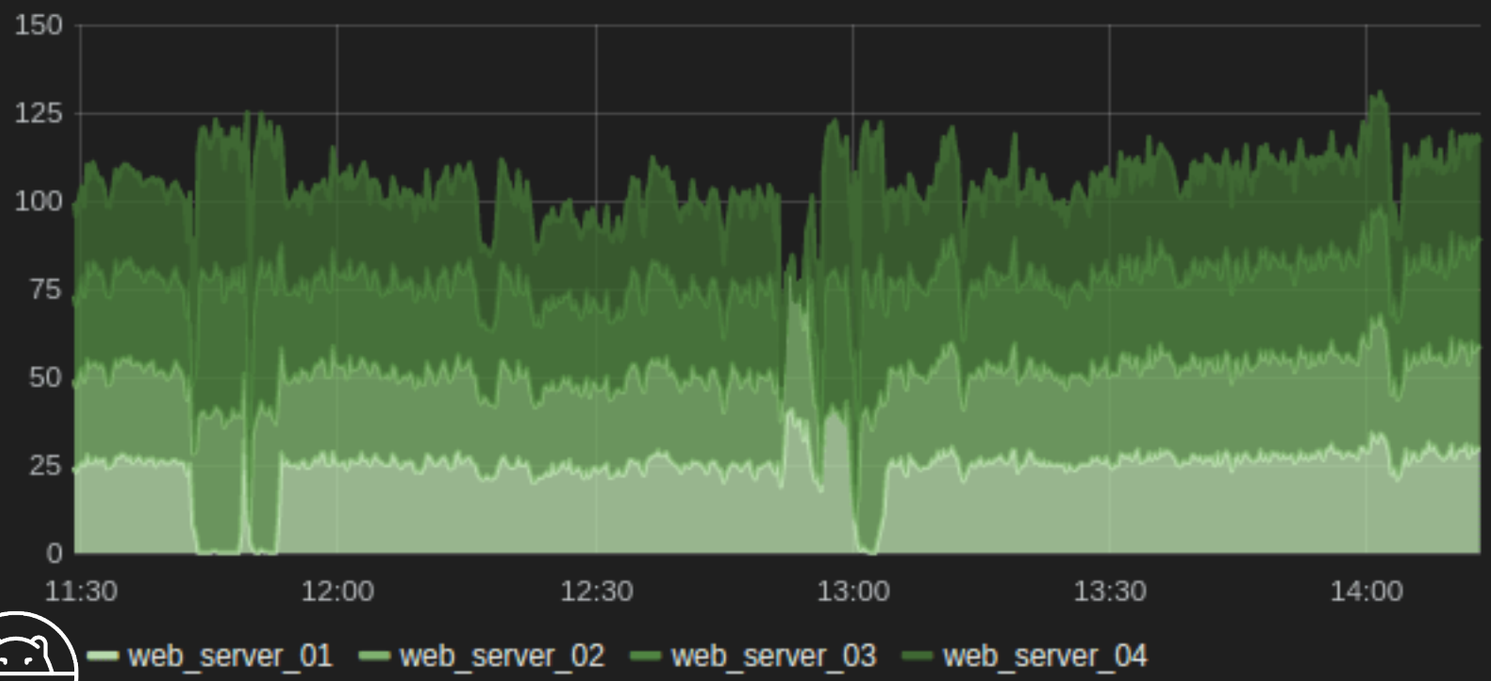
263

Memory / CPU

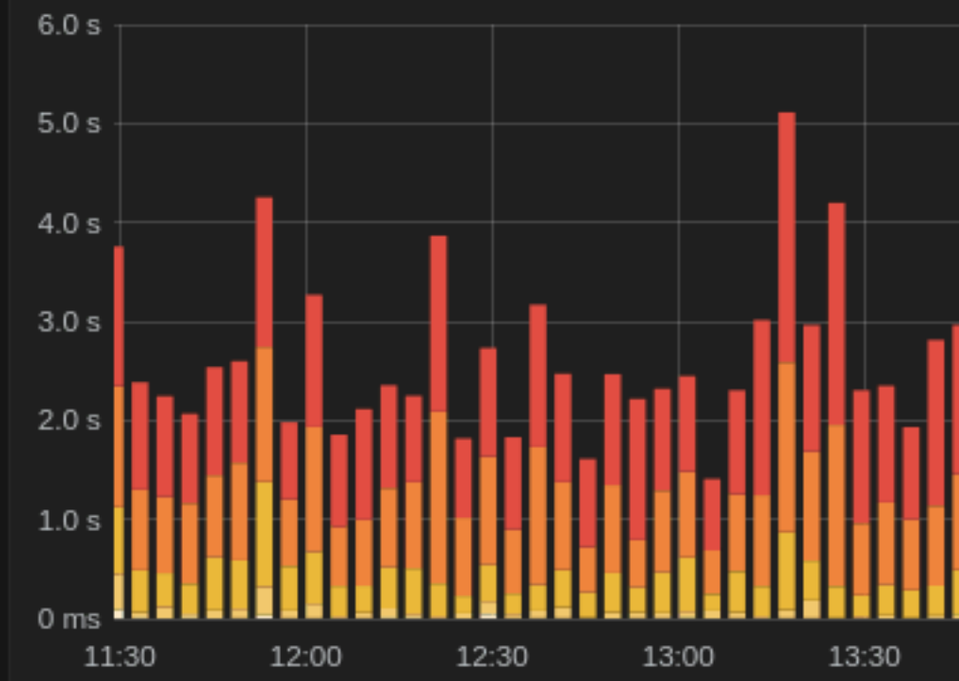


logins

server requests



client side full page loa



Distributed Tracing

```
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()})
```

Open Census

Open Tracing


OpenTelemetry
COMING SOON

https://github.com/italolelis/talks

https://github.com/italolelis/designing-for-failure

https://github.com/italolelis/coffee-shop

Спасибо

@italolelis

Questions?

Study

https://docs.microsoft.com/en-us/previous-versions/msp-n-p/dn589804(v=pandp.10)

Credits

All art work used in this presentation is provided by
Gopher Artwork by Ashley McNamara

https://github.com/ashleymcnamara/gophers