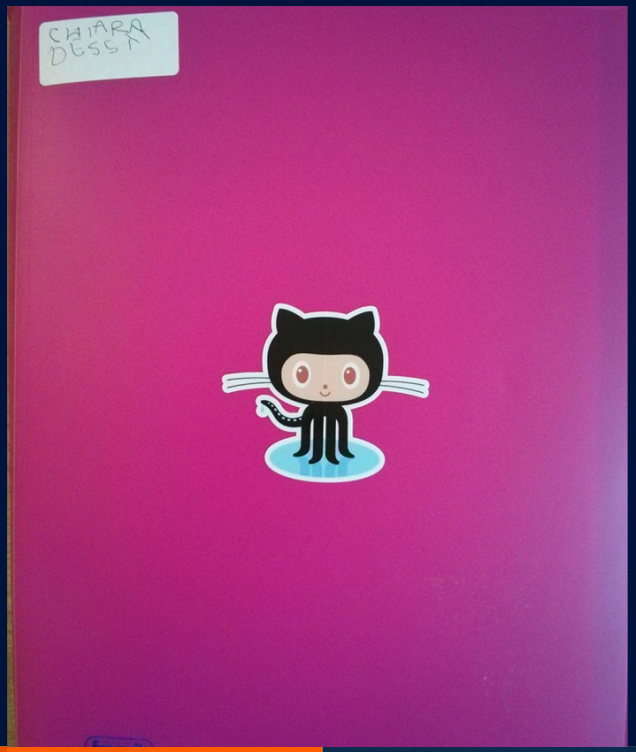
ROME 24-25 MARCH 2017 {codemotion}

{ Microservices in GO

Massimiliano Dessì @desmax74



Speaker

Massimiliano Dessì has more than **17** years of experience in programming, and several years in cloud computing.
Manager of GDG Sardegna,
co-founder of Jug Sardegna,
Founder of SpringFramework IT UG,
Author of Spring 2.5 AOP.
He's a proud father of **three**,
one **girl** and two **boys**.
He currently works for **Red Hat**.

GDG Sardegna

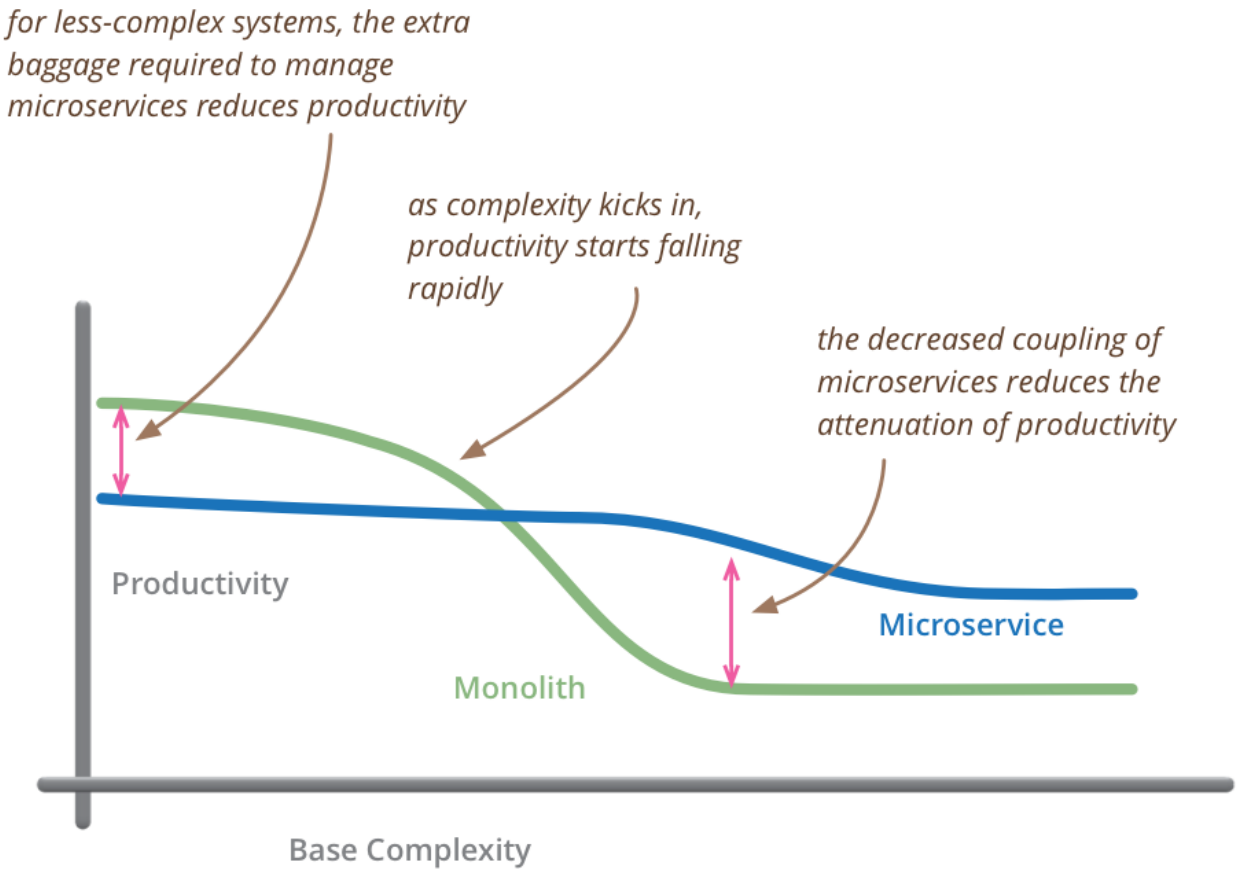
Monolith vs Microservice

It depends...

“.. don't even consider microservices unless you have a system that's too complex to manage as a monolith.”

“...is a microservice architecture a good choice for the system you're working on ?”

https://martinfowler.com/bliki/MicroservicePremium.html



but remember the skill of the team will outweigh any monolith/microservice choice

https://martinfowler.com/bliki/MicroservicePremium.html

NO free launch

After these important premises,

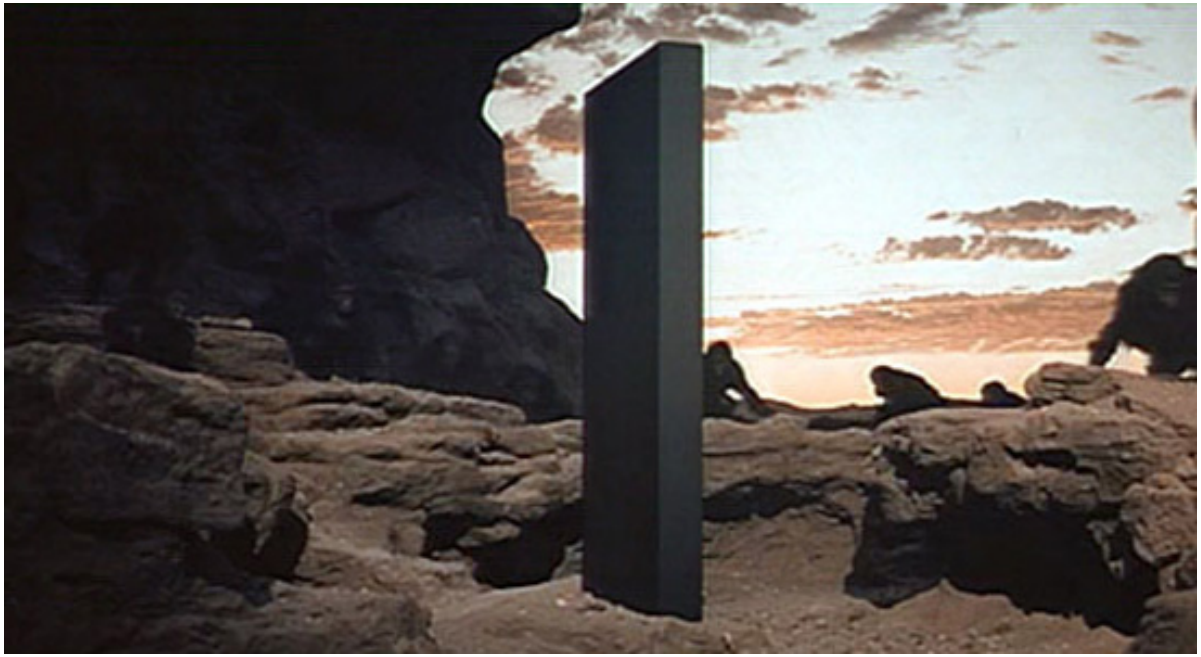
we will see the bad part of monolith and

how microservices

can solve this problems

and how much cost the solution

Monolith in sci-fi



Arthur C.Clarke 2001 : A space Odissey

Software Monolith

A successful application growth
during the years
with new features
and
at some point in time,
it is simply too large for any developer to
fully understand.

Big ball of mud/Lasagna code

Million LOC

Hundreds/thousands jar/libs

Codebase difficult to understand.

Each change makes the codebase incrementally more complex, and more difficult to understand

Large teams
on large monolitic codebase
lead to feature branches
and often
to a painful merges

When your application needs an extra feature after release.

Run the compilation
and
the entire suite test
tooks a significant
amount of time,
long time to diagnose and fix
the cause of a test failures

A monolithic app
force to use
a
single
technology stack,
used since
the start of the
project

WW2

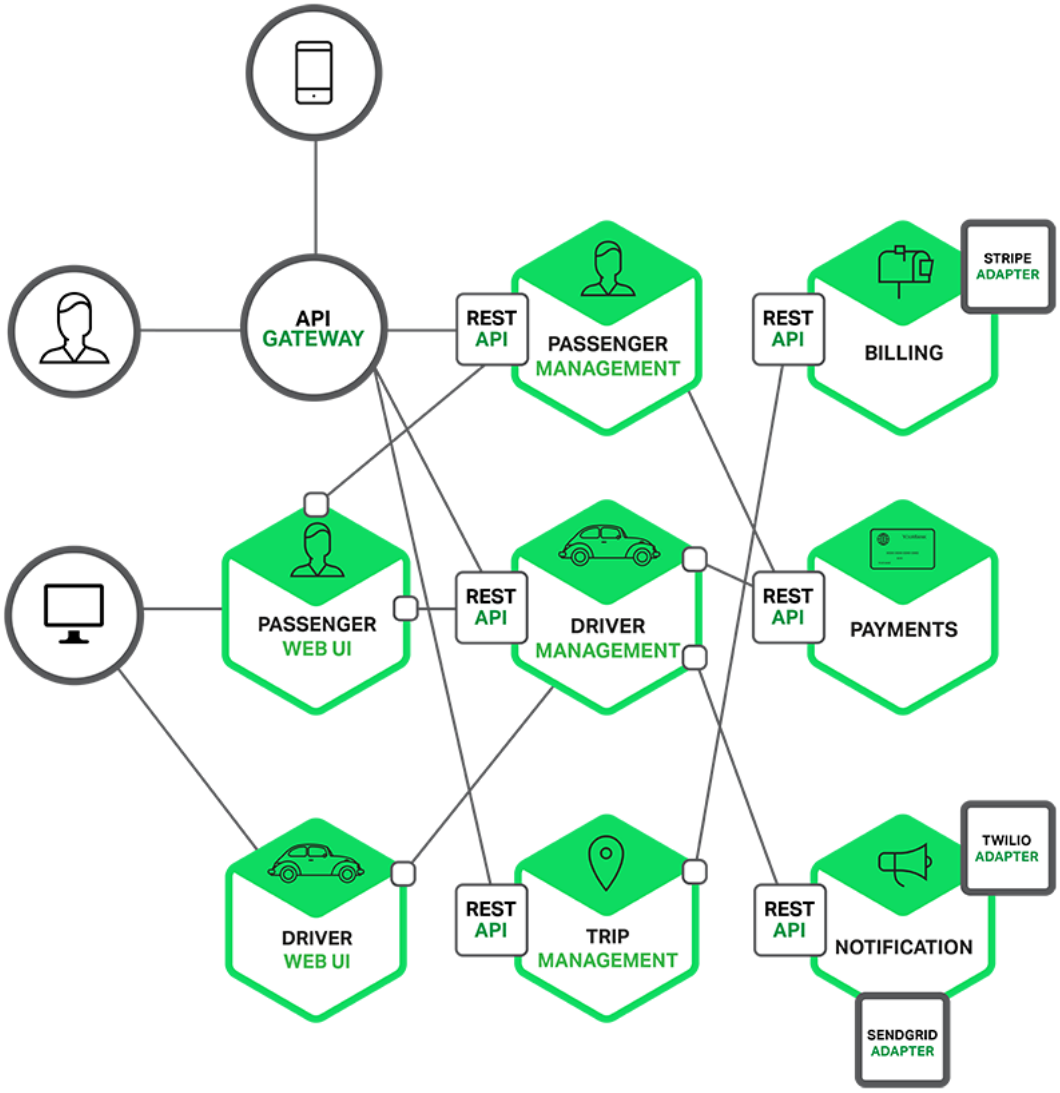
Microservices

http://martinfowler.com/articles/microservices.html

Service-oriented architecture
composed of
loosely coupled elements
that
have bounded contexts.

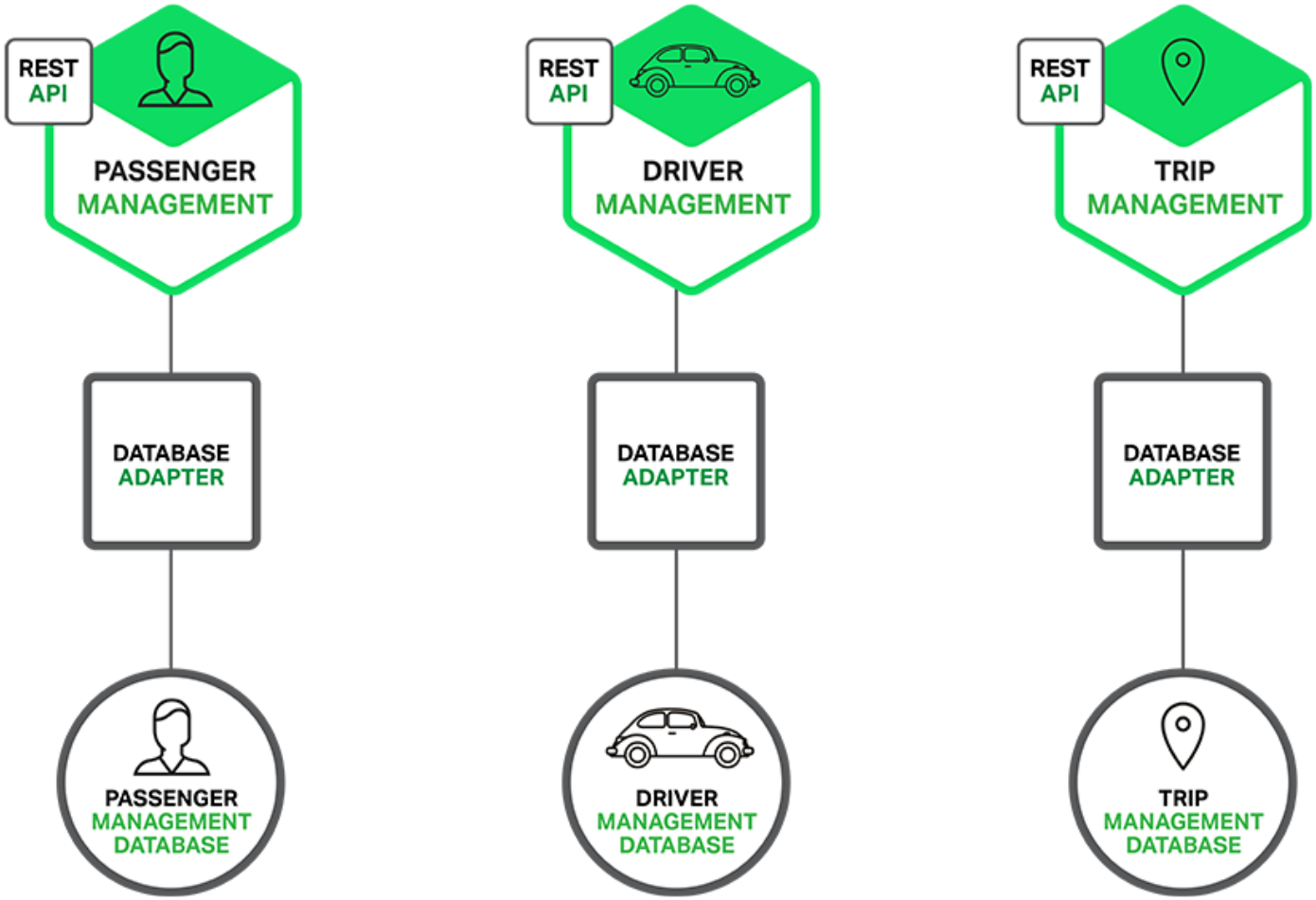
Adrian Cockcroft
Netflix

Functional Decomposition



https://www.nginx.com/blog/introduction-to-microservices/

Bound Context and Data Partition



https://www.nginx.com/blog/introduction-to-microservices/

Benefits of Microservices

A microservice architecture provides benefits like:

Enabling growth at different rate

Deploy multiple instances when needed

Scalability

Benefits of Microservices

Different technology stack for different services

Caren Hartley http://www.hartleycycles.com/

The teams are able to manage every single microservice

Fast testing

Monitoring

Code ownership

Benefits of Microservices

Monitoring/health check

Rewriting with a different stack it’s easy

Circuit breaker (internal/external)

Bulkhead (internal/external)

Costs of Microservices

Coordination/Orchestration

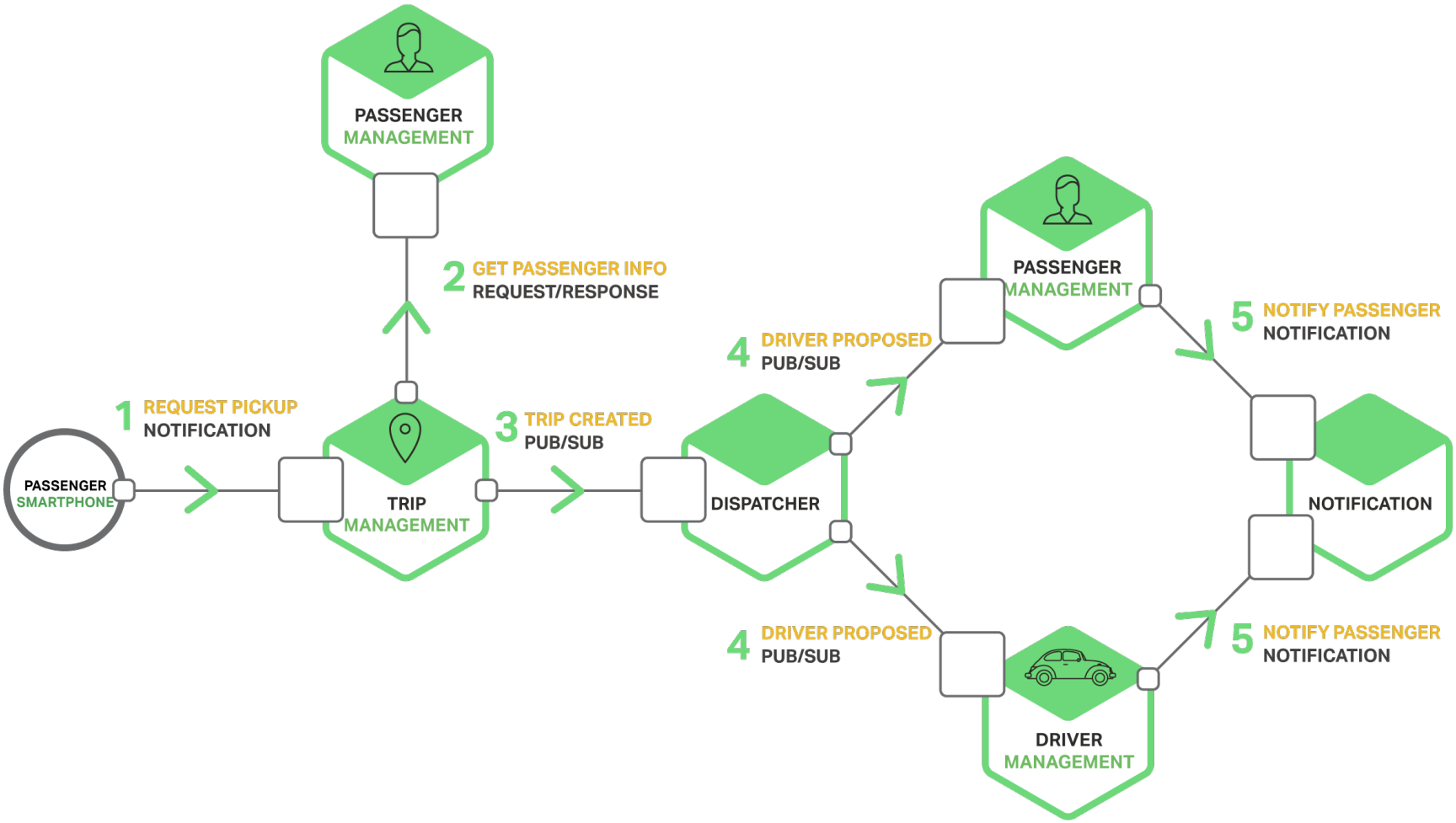
Inter process communication

Discovery

Fallacies of distributed computing

Complexity of the entire system

Costs of Microservices



https://www.nginx.com/blog/building-microservices-inter-process-communication/

Costs of Microservices

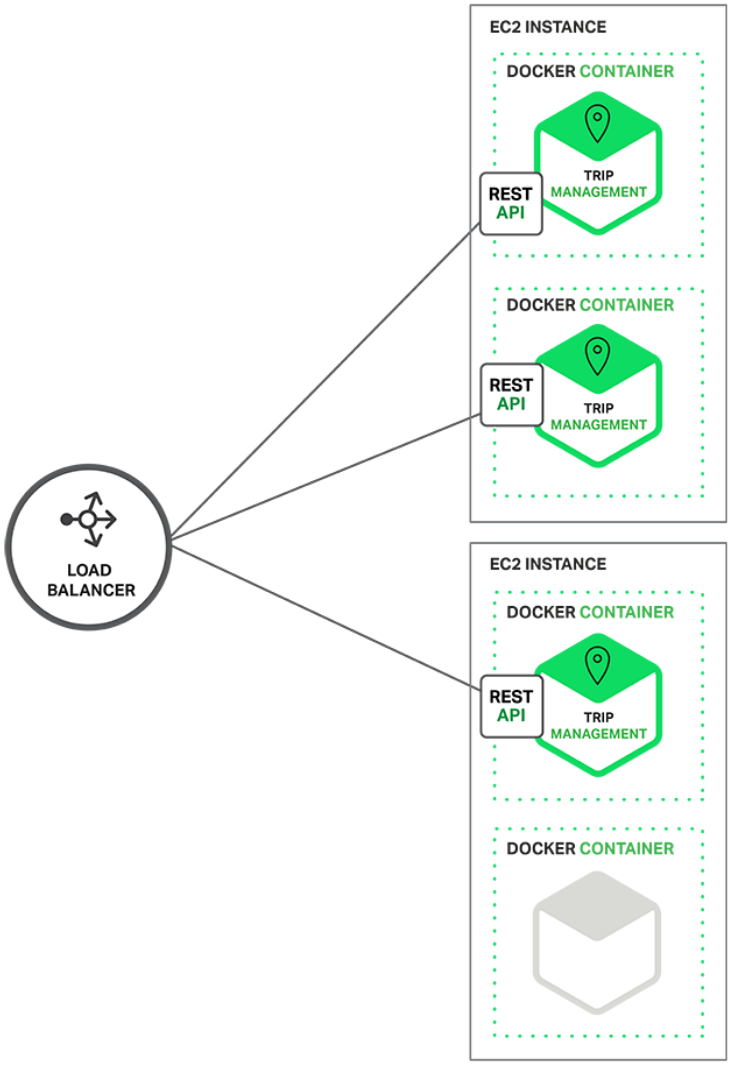
Latency

API Gateway

Load Balancers

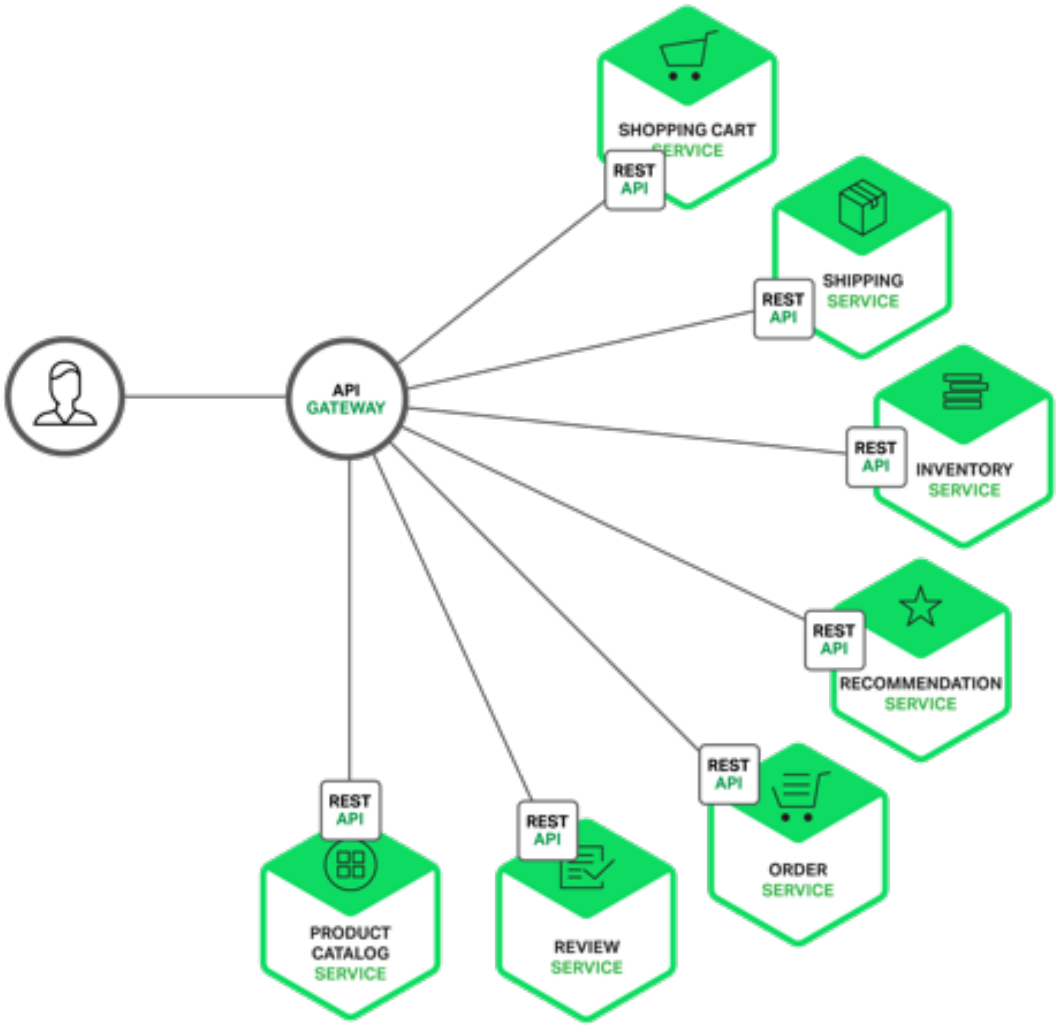
Load Balancer

External
Load balancer



https://www.nginx.com/blog/introduction-to-microservices/

API Gateway



https://www.nginx.com/blog/building-microservices-using-an-api-gateway/

Api Gateway

http://www.apiman.io

https://tyk.io/

https://getkong.org/

Costs of Microservices

Test with other services

Operational complexity

High level of automation

Errors with Microservices

Silver bullet/Golden Hammer
(human fault)

Wrong decomposition of
a monolith/Distributed monolith

Coupled services

Microservices and 12 factor app

Our microservices writed as a cloud
(private/public/hybrid)
native services
must obey the rules of
12 factor app

`https://12factor.net/`

Golang tools

- Rest - fasthttp
- gRPC
- RabbitMQ
- Context
- Env Config- Viper
- Load Test - Hey
- Load test - Vegeta
- Distributed Tracing - Zipkin
- Service discovery - Consul
- Service discovery — Kubernetes
- Load balancer — Kubernetes
- Health check - Kubernetes

REST HTTP

```
import (
     log "github.com/Sirupsen/logrus"
     "github.com/buaazp/fasthttprouter"
     mid "org.desmax/gulch/middlewares"
      …
)
func init(){
…..
}

func main() {
     db := sys.StartDB(Conf)
     defer db.Close()

     /*Routes*/
     router := fasthttprouter.New()
     router.GET("/", mid.ConfigMidw(handlers.LoginHandler, Conf))
     router.POST("/login", mid.DBMidw(mid.ConfigMidw(handlers.LoginHandler, Conf), db))
     router.POST("/auth", mid.DBMidw(mid.ConfigMidw(handlers.AuthHandler, Conf), db))
     router.GET("/healthz",  mid.ConfigMidw(handlers.HealthHandler, Conf)
     router.GET("/items",  mid.ConfigMidw(handlers.ItemHandler, Conf)
       ...

     /*Server*/
     config.HttpServer(router, Conf) //HTTP
     //config.HttpsServer(router, conf) //HTTPS

}
```

REST HTTP

```
func ItemsHandler(ctx *fasthttp.RequestCtx, params fasthttprouter.Params) { {
   ….
  json, err := json.Marshal(MyStruct)
    if err != nil {
        //manage error
        return
    }
    ctx.SetStatusCode(fasthttp.StatusOK)
    ctx.SetBody(json) //[]byte
}
```

gRPC (Google RPC)

Remote Procedure call using protocol buffer

Protocol Buffers, also referred as protobuf, is Google's language-neutral, platform-neutral, extensible mechanism for serializing structured data. Protocol Buffers are smaller, faster, and simpler that provides high performance than other standards such as XML and JSON.

1)
Download Protocol Buffer in your project

```
go get -u github.com/golang/protobuf/{proto,protoc-gen-go}
go get google.golang.org/grpc
```

gRPC (Google RPC)

```
syntax = "proto3";
package user;

service User {
  rpc GetUsers(UserFilter) returns (stream UserRequest) {}
  rpc CreateUser (UserRequest) returns (UserResponse) {}
}

message UserRequest {
  int32 id = 1;
  string name = 2;
  string email = 3;

  message Address {
    string street = 1;
    string city = 2;
    string state = 3;
    string zip = 4;
  }
  repeated Address addresses = 5;
}

message UserResponse {
  int32 id = 1;
  bool success = 2;
}
message UserFilter {
  string keyword = 1;
}
```

2)
Define your proto3 file called `vault.proto` with service, request and response

gRPC (Google RPC)

3) Generate code

```
protoc -I <my_package>/user/user.proto –go_out=plugins=grpc:user

Output user.pb.go
```

4) Create gRPC Server

```
import (
     ...
     "google.golang.org/grpc"
      pb "github.com/desmax74/<prj_name>/grpc/user"
)

const (
     port = ":50051"
)


type server struct {
     savedUsers []*pb.UserRequest
}
```

gRPC (Google RPC)

```
func (s *server) CreateUser(ctx context.Context, in *pb.UserRequest)
(*pb.UserResponse, error) {
    s.savedUsers = append(s.savedUsers, in)
    return &pb.UserResponse{Id: in.Id, Success: true}, nil
}

func (s *server) GetUsers(filter *pb.UserFilter, stream pb.User_GetUserServer) error
{
    for _, user := range users {
        //search the user with the filter
        ….

        //write the user in the stream
        if err := stream.Send(user); err != nil {
            return err
        }
    }
    return nil
}
```

gRPC (Google RPC)

```
func main() {
    lis, err := net.Listen("tcp", port)
    if err != nil {
        log.Fatalf("Error on listen: %v", err)
    }
    s := grpc.NewServer()
    pb.RegisterUserServer(s, &server{})
    s.Serve(lis)
}
```

gRPC (Google RPC)

```
func createUser(client pb.UserClient, user *pb.UserRequest) {
    resp, err := client.CreateUser(context.Background(), user)
    if err != nil {
        log.Printf("Error on create User: %v", err)
    }
    if resp.Success {
        log.Printf("New User has created with id: %d", resp.Id)
    } ...
```

gRPC (Google RPC)

```
func getUsers(client pb.UserClient, filter *pb.UserFilter) {
    stream, err := client.GetUsers(context.Background(), filter)
    if err != nil {
       ...
    for {
        users, err := stream.Recv()
        if err == io.EOF {
            break
        }
        if err != nil {
            log.Fatalf("%v.GetUsers(_) = _, %v", client, err)
        }
        log.Printf("User: %v", user)
    }
}
```

gRPC (Google RPC)

Client code

```
func main() {
    conn, err := grpc.Dial(address, grpc.WithInsecure())
    if err != nil {
        log.Fatalf("did not connect: %v", err)
      }
    defer conn.Close()
    client := pb.NewUserClient(conn)

      user := &pb.UserRequest{
        Id:    17,
        Name:  "Speaker One",
        Email: "spealer1@codemotion.it",
        Phone: "+39-345-1234567",
    ….


      createUser(client, user)
```

RabbitMQ

```
const (
    MQHost = "127.0.0.1"
    MQPort = ":5672"
)

type MyData struct {
    Username string `json:"username"`
    Message string `json:"message"`
}

func MQConnect() (*amqp.Connection, *amqp.Channel, error) {
    url := "amqp://" + MQHost + MQPort
    conn, err := amqp.Dial(url)
    if err != nil {
        return nil, nil, err
    }
    channel, err := conn.Channel()
    if err != nil {
        return nil, nil, err
    }
    if _, err := channel.QueueDeclare("myqueue", false, true, false,
            false,nil)
```


https://github.com/streadway/amqp

RabbitMQ

```
    err != nil {
        return nil, nil, err
    }
    return conn, channel, nil
}

    // Publish
     err = channel.Publish(
           "", // exchange
           q.Name, // routing key
           false, // mandatory
           false, // immediate
           amqp.Publishing{
              ContentType: "text/plain",
              Body: []byte(body),
           }
     )

    // Consume
    msgs, err := ch.Consume(
              q.Name, // queue
              "", // consumer
              true, // auto-ack
              false, // exclusive
              false, // no-local
              false, // no-wait
              nil, // args
     )
```

https://github.com/streadway/amqp

Like Java Thread Local

Variables
type CancelFunc
type Context
func Background() Context
func TODO() Context
func WithCancel(parent Context) (ctx Context, cancel CancelFunc)
func WithDeadline(parent Context, deadline time.Time) (Context, CancelFunc)
func WithTimeout(parent Context, timeout time.Duration) (Context, CancelFunc)
func WithValue(parent Context, key, val interface{}) Context

```
import "context"

func WithValue(parent Context, key interface{}, val interface{}) Context
```

Golang Context

```
Func myBusinessHandler(){
    var timeInMilliseconds time.Duration = 0
    ctx := context.WithValue(context.Background(), "time",
&timeInMilliseconds)

 myBusinessFuncWithTracking(ctx)
 val := ctx.Value("time").(*time.Duration)

}
```

```
func myBusinessFuncWithTracking(ctx context.Context) {
  defer func(s time.Time) {
    val := ctx.Value("time").(*time.Duration)
    *val = *val + time.Since(s)
  }(time.Now())

  //do other
}
```

Configuration from environment

Docker run command:

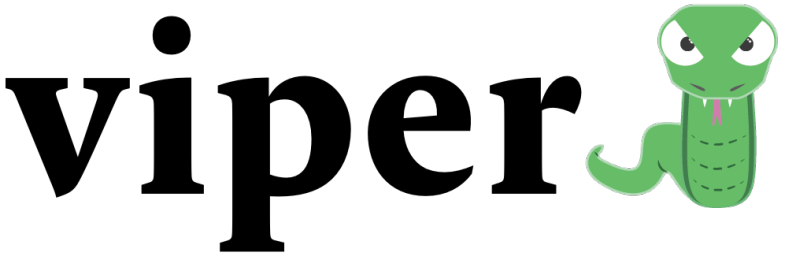
```
docker run -it desmax74/gdg -e PORT='8080' -e APPNAME='gdg'
-p 8080:8080 -e LOGSERVER='127.0.0.1:1902'
```

```
//plain without  external libs

serverPort := os.Getenv(PORT)
if(serverPort == "") {serverPort = "8080"}

appName := os.Getenv(APPNAME)
logServer := os.Getenv(LOGSERVER)
```

Configuration from environment




```
 AutomaticEnv()
 BindEnv(string...) : error
 SetEnvPrefix(string)
 SetEnvReplacer(string...) *strings.Replacer


 //Binding Env to Specific Keys
 viper.BindEnv("port") // bind to ENV "PORT"
 viper.BindEnv("name", USERNAME) // bind to ENV "USERNAME"

 os.Setenv("PORT", "8080") // outside of the app
 os.Setenv("USERNAME", "speaker") // outside app

 port := viper.GetInt("port")) // 13
 name := viper.GetString("name")) // "spf13"


//Automatic Env Binding
viper.SetEnvPrefix("foo") // Becomes "FOO_"
os.Setenv("FOO_PORT", "1313") // outside of the app
viper.AutomaticEnv()
port := viper.GetInt("port"))
```

https://github.com/spf13/viper

Load Test Setup

```
garfield@garfield:~$ docker run desmax74/gogulch
Prontu a tzaccai !
Conf: {LinuxDay2016 8080  true    }

MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMO::MMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMZ..              ... MMMMMMMM
MDZMMMMMMMMMMMMMMMMMMMMMMMMMMZ.      .          ....MMMMMMMM
???~NMMMMMMMMMMMMMMMMMMMM=III=.    .... ....   ....,MMMMMMM
+???NMMMMM?.   . . .. ......IIII.....+..~ZZZZ. .......MMMMMMM
MMIIMM?                  .MMMNO77Z8MMM8IZO:~+  .....MMMMMMM
MM77N,8                MMMMMMMMMMMMMMMDM$,ZO$.......7MMMMMM
MMM    :         .:ZZZ.8MMMMMMMMMMMMMMMZ7.ZZZZ$..... :MMMMMM
MMM   =     Z...ZZZZ.:+MMMNZZZZZ8MMMMM8+MZZZZZZ.......MMMMMM
MMMI7IMN    $..ZZZ..=ZZ.  ZOZZZOZ$7ZZZMD.OZZZZ~..... .NMMMMM
MMMMII7MM    ,.OI ZZZZZZ. ON888888DO8NZZ,ZZZZ.  .... .~MMMMM
MMMMZ7IMM.    .I7.:ZZZZZZ..N?~ 88N.8 888N..ZZZ$. ...... MMMMM
MMMMM7IDMD    .+:?ZZZZZ$. N:M$88NMMO888D .....  .... . MMMMM
MMMMMN7IMM     I~$ZZZZ?.  N8D?????NDO88D   ..........~IMMMMM
MMMMMM7I7MM    ......ZZ. .M8???????OD88D,?IIIIIIIIIIIIIMMMMM
MMMMMMOI7MM+  .   .......N?O7??7Z??.O888,?IIIIIIIIII+,.MMMMM
MMMMMMM7IDMM.    ..  ....7D.D ~$Z.  .Z8.IMMM8    .......ZMMMM
MMMMMMM7IIMMI..7IIIII?D.MM.:.+  .II..I,8MM...Z  ........,MMMM
MMMMMMMD88NMMIIIIIII?I.:MM...~+7.?,...=MM....,~ZZ:......MMMM
MMMMMMM88888MM7I?~..I..MMM....O$=.....MMM.....:.,:......MMMM
MMMMMMM8888M.,M.  .7...MMM,....:....,MMMM......N+OZ:.....~MMM
MMMMMMMMM7II.....~.,..7MMMM........+MMMMM.......DZZ......MMM
MMMMMMMMMMIID.........MMMMMM.....+MMMMMM........=ZI.....IMM
MMMMMMMMMM77I........$MMMMMOI7MMMMMMMMMM=:.......... .  MM
MMMMMMMMMMD7I+.......MMMM=7NMMMMMMM7MMMMM?......,........=M
MMMMMMMMMMMIIO,......MMMM:MMMMMMMMMMMMMMM.......?~........M
MMMMMMMMMMMIIM8..NMZMMMM=NIMMMMMM8MMMMM88888?.+..... ..MMM
MMMMMMMMMMMII7M..???IMMMMMMMMMMMMMMMD?O8888M~IMMMMMMMMMMM
MMMMMMMMMMMO M????????DMMN++MMMMD?MMMMO?IDDN???MMMMMMMMMMM
MMMMMMMMMMMM.$?????????8MDMMMMMMMMMM8????????IMMMMMMMMMM
MMMMMMMMMMMM,8??????????+MZI?MMM??MMMMZ??????????$MMMMMMMM
MMMMMMMMMMMMMM?????????????I?MMMM$MDM.~+7??????????IMMMMMMMM
MMMMMMMMMMMMM?????????????D.....,....:I????????IMMMMMMMMMM
MMMMMMMMMMMMMM?????????????O8888....D88I???????MMMMMMMMMMM
MMMMMMMMMMMMMM8OOMMMII????MMMMMMMMMMO????7MMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM

INFO:2017/03/20 starter.go:53: created bucket: GulchBucket
```

1)Run our docker image/K8s POD

2)Inspect image

3)Read the ip


```
        "Networks": {
            "bridge": {
                "IPAMConfig": null,
                "Links": null,
                "Aliases": null,
                "NetworkID": "f3634ab9f2c823140361efb921ee4c9203850eeba02855
e339dd6d1ff02cc823",
                "EndpointID": "ed8811cd2cc9cd459c5150a1f8c98fa15aceb155f7c87
47b735fcf5120f155dc",
                "Gateway": "172.17.0.1",
                "IPAddress": "172.17.0.2",
                "IPPrefixLen": 16,
                "IPv6Gateway": "",
                "GlobalIPv6Address": "",
                "GlobalIPv6PrefixLen": 0,
                "MacAddress": "02:42:ac:11:00:02"
            }
        }
    }
}
```

Load Test Hey

Hey previously know as boom

```
$ hey -n 100 -c 10 -m GET http://172.17.0.2:8080/
All requests done.

Summary:
  Total:     0.0068 secs
  Slowest:   0.0026 secs
  Fastest:   0.0002 secs
  Average:   0.0005 secs
  Requests/sec:     14772.2227
  Total data:7500 bytes
  Size/request:     75 bytes

Status code distribution:
  [200]      100 responses

Response time histogram:
  0.000 [1]   |∎
  0.000 [65]  |∎∎∎∎∎∎∎∎∎∎∎∎∎∎∎∎∎∎∎∎∎∎∎∎∎∎∎∎∎∎∎∎∎∎∎∎∎∎∎∎
  0.001 [16]  |∎∎∎∎∎∎∎∎∎∎
  0.001 [4]   |∎∎
  0.001 [2]   |∎
  0.001 [3]   |∎∎
  0.002 [5]   |∎∎∎
  0.002 [1]   |∎
  0.002 [0]   |
  0.002 [1]   |∎
  0.003 [2]   |∎

Latency distribution:
  10% in 0.0002 secs
  25% in 0.0003 secs
  50% in 0.0003 secs
  75% in 0.0005 secs
  90% in 0.0013 secs
  95% in 0.0016 secs
  99% in 0.0026 secs
```

https://github.com/rakyll/hey

Load Test Vegeta

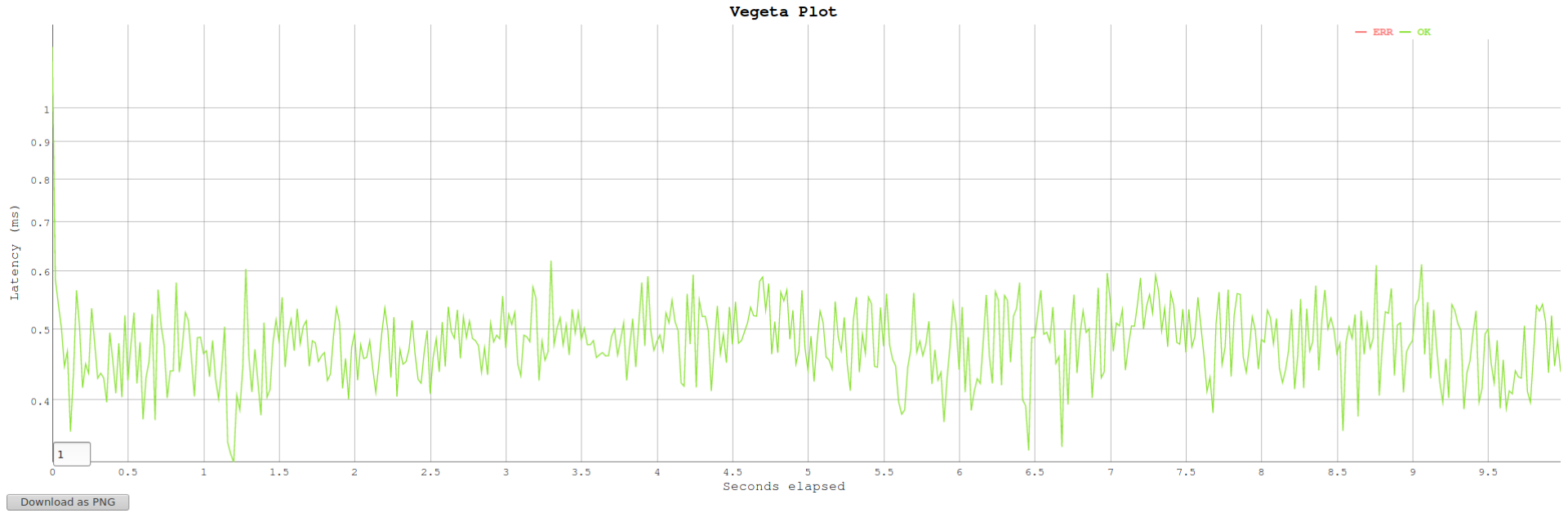
```
echo "GET http://172.17.0.2:8080/" | ./vegeta attack -duration=10s | tee results.bin |
./vegeta report
Requests      [total, rate]            500, 50.10
Duration      [total, attack, wait]    9.98043696s, 9.979999794s, 437.166µs
Latencies     [mean, 50, 95, 99, max]  481.501µs, 479.956µs, 564.821µs, 596.038µs, 1.211673ms
Bytes In      [total, mean]            37500, 75.00
Bytes Out     [total, mean]            0, 0.00
Success       [ratio]                  100.00%
Status Codes  [code:count]             200:500

Error Set:
```

https://github.com/tsenart/vegeta

Load Test Vegeta

cat results.bin | ./vegeta report -reporter=plot > plot.html



Golang Distributed tracing

With microservices we need to correlate the calls between services to track the flow.

We can use OpenZipkin, we add a Correlation ID and propagate it between processes, then the correlated logs can be used by ELK or Splunk

Correlation ID -> unified view of business event

Distributed tracing Lingo

A Span is an individual operation

A Span contains timestamped events and tags.

A Trace is an end-to-end latency graph, composed of spans.

Distributed tracing

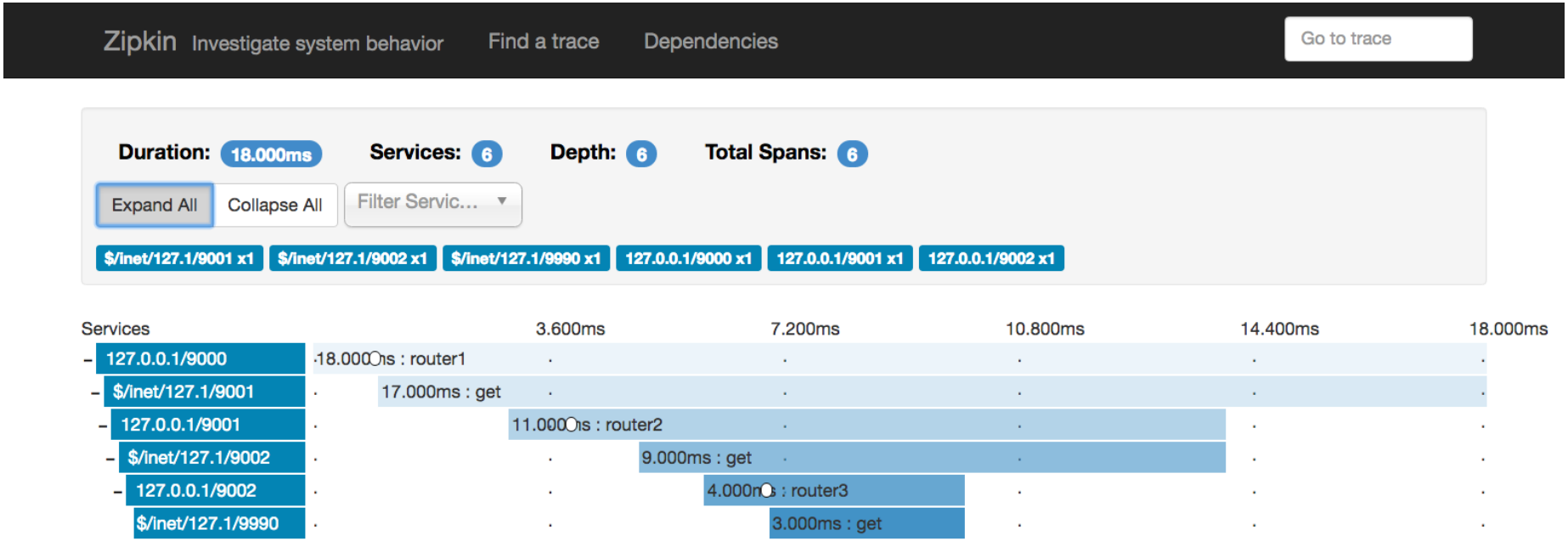
Server Received is an event

POST /data is a span name

325ms is a span duration

http.request-id=xyz1-rf3 is a span tag

Zipkin Docker image



https://github.com/openzipkin/docker-zipkin

Golang Distributed tracing

```
if span := opentracing.SpanFromContext(req.Context());
span != nil {

    // We are going to use this span in a clientrequest
          ext.SpanKindRPCClient.Set(span)

          // Add some standard OpenTracing tags
          ext.HTTPMethod.Set(span, req.Method)
          span.SetTag(zipkincore.HTTP_HOST, req.URL.Host)
          span.SetTag(zipkincore.HTTP_PATH, req.URL.Path)
          ext.HTTPUrl.Set(
              span,
              fmt.Sprintf("%s://%s%s",
                  req.URL.Scheme,
                  req.URL.Host,
                  req.URL.Path),
              )
```

https://github.com/opentracing/opentracing-go
https://github.com/openzipkin/zipkin-go-opentracing/

Golang Distributed tracing

continue

```
// Add information on the peer service we're about to contact.
    if host, portString, err := net.SplitHostPort(req.URL.Host);
        err == nil {
                ext.PeerHostname.Set(span, host)
                if port, err := strconv.Atoi(portString);
                err != nil {
                    ext.PeerPort.Set(span, uint16(port))
                }
    } else {
        ext.PeerHostname.Set(span, req.URL.Host)
    }
```

https://github.com/opentracing/opentracing-go
https://github.com/openzipkin/zipkin-go-opentracing/

Golang Distributed tracing

Continue

```
    // Inject the Span context into the outgoing HTTP Request.
        if err := tracer.Inject(
            span.Context(),
            opentracing.TextMap,
            opentracing.HTTPHeadersCarrier(req.Header),
        );
        err != nil {
            fmt.Printf("error encountered while trying to inject
span: %+v", err)
        }
    }
    return req

}
```

https://github.com/opentracing/opentracing-go
https://github.com/openzipkin/zipkin-go-opentracing/

Service discovery with Consul

```
import consul "github.com/hashicorp/consul/api")

type Client interface {
    Service(string, string) ([]string, error)
    Register(string, int) error
    DeRegister(string) error
}

type client struct {
    consul *consul.Client
}

func (c *client) Register(name string, port int) error {
    reg := &consul.AgentServiceRegistration{
        ID:   name,
        Name: name,
        Port: port,
    }
    return c.consul.Agent().ServiceRegister(reg)
}
```

https://www.consul.io/

Service discovery with Consul

```
func (c *client) DeRegister(id string) error {
    return c.consul.Agent().ServiceDeregister(id)
}

func (c *client) Service(service, tag string) ([]*ServiceEntry,
*QueryMeta, error) {
    passingOnly := true
    addrs, meta, err := c.consul.Health().Service(service, tag,
passingOnly, nil)
    if len(addrs) == 0 && err == nil {
        return nil, fmt.Errorf("service ( %s ) was not found",
service)
    }
    if err != nil {
        return nil, err
    }
    return addrs, meta, nil
}
```

https://github.com/hashicorp/consul

Service discovery inside Kubernetes

```
//with Environment variables
backendHost := os.Getenv("BE_SRV_SERVICE_HOST")
    backendPort := os.Getenv("BE_SRV_SERVICE_PORT")
    backendRsp, backendErr := http.Get(fmt.Sprintf(
        "http://%v:%v/",
        backendHost,
        BackendPort)
    )

    if backendErr == nil {
        defer backendRsp.Body.Close()
    }
```

Service discovery inside Kubernetes

```
//with DNS server
cname, rec, err := net.LookupSRV("be", "tcp",
    "be-srv.default.svc.cluster.local")
    if err != nil {
        http.Error(resp, err.Error(),
        http.StatusInternalServerError)
    }
    fmt.Fprintf(resp, "SRV CNAME: %v\n", cname)
    for i := range rec {
        fmt.Fprintf(resp, "SRV Records: %v \n", rec[i])
        DNSbackendHost = rec[i].Target
        DNSbackendPort = strconv.Itoa(int(rec[i].Port))
    }
```

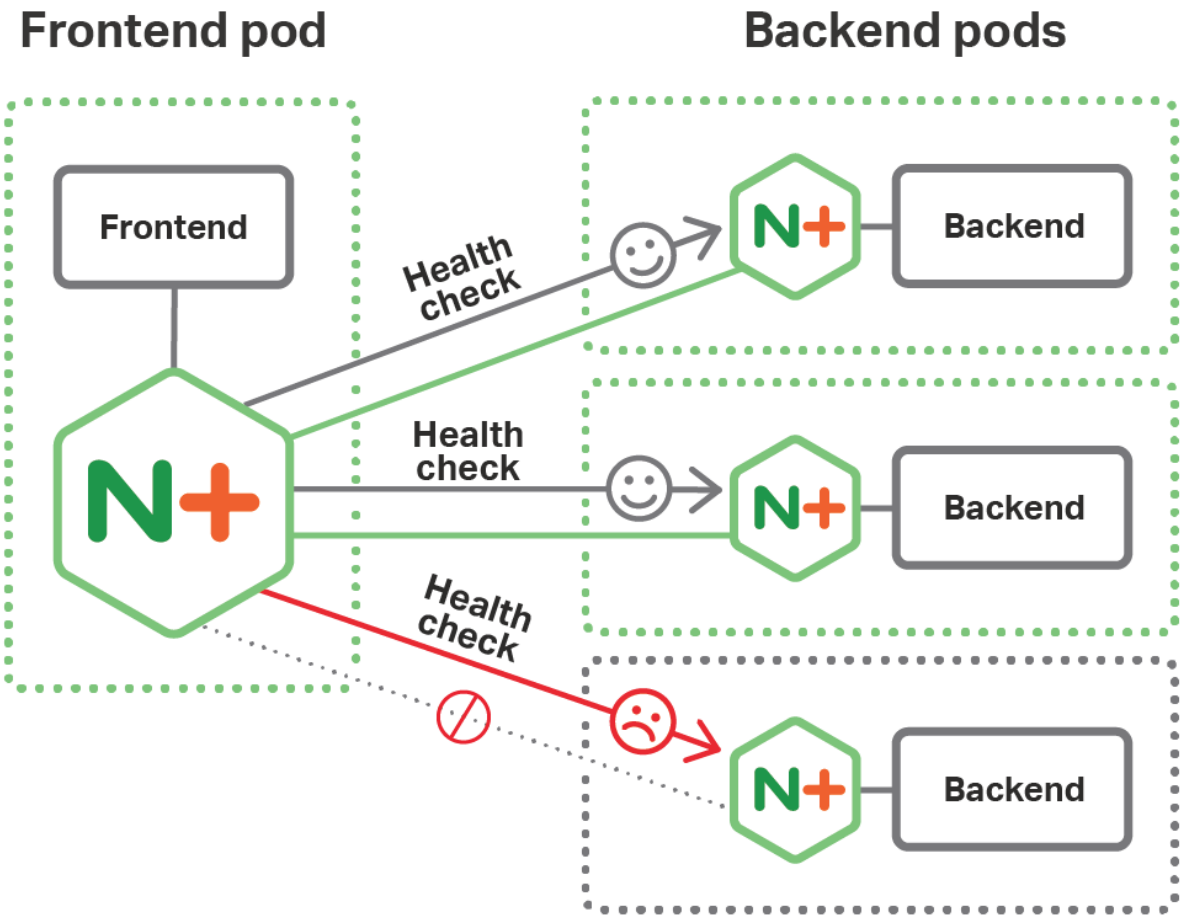
Kubernetes Load Balancer

```
kubectl expose rc desmax --type=LoadBalancer --name desmax-http
```

```
$ kubectl get svc
NAME          CLUSTER_IP     EXTERNAL_IP       PORT(S)   AGE
Kubernetes    10.3.240.1     <none>            443/TCP   35m
Kubia-http    10.3.245.160   XXX.XXX.XXX.XXX   8080/TCP  1m
```

https://www.nginx.com/blog/microservices-openshift-fabric-model-nginx-mra/

Health check



https://www.nginx.com/blog/microservices-openshift-fabric-model-nginx-mra/

Kubernetes Health check

In our microservice code

```
router.GET("/healthz",
          mid.ConfigMidw(handlers.HealthHandler, Conf)
)
```

In kubernetes deployment

```
livenessProbe:
  # an http probe
  httpGet:
    path: /healthz
    port: 8080
  initialDelaySeconds: 15
  timeoutSeconds: 1
```

Resources

http://kubernetes.io/

https://www.openshift.org/

https://martinfowler.com/articles/microservices.html

https://martinfowler.com/bliki/MicroservicePremium.html

https://www.nginx.com/blog/introduction-to-microservices/

http://www.grpc.io/

https://github.com/grpc/grpc-go

https://www.manning.com/books/kubernetes-in-action

https://github.com/micro/go-micro

Q & A

{ Thanks and have fun !

@desmax74
http://slideshare.net/desmax74

All pictures belong
to their respective authors