Event Stream e Data Lake usando Go

[Event Stream e Data Lake] Você possui algum destes problemas?

- Como eu consigo o dado da aplicação X para realizar um relatório?
- Quais dados temos aqui dentro?
- Temos um determinado dado?
- Onde está esse dado?
- Como consigo acesso ao histórico deste dado?

[Event Stream e Data Lake] Como resolver estes problemas?

Evolutionary Database Design - Martin Fowler e Pramod Sadalage (link)

Data Lake - Martin Fowler (link)

Spotify's Event Delivery – The Road to the Cloud (link)

[Event Stream e Data Lake]

- **Schema Manager -** catálogo dos dados;
- **Event Stream -** barramento para comunicação entre todos os serviços da empresa;
- **Data Lake -** repositório único dos dados;

[Event Stream e Data Lake] Arquitetura Event Stream

- Toda troca de mensagem na sua arquitetura é um evento!
- Comunicação de forma assíncrona;
- Benefícios no compartilhamento de dados;
- Normalização do processos de envio e consumo;
- Necessidade de um broker para troca de mensagens:
- Exemplo: **Kafka, PubSub, ...**

[Event Stream e Data Lake] Schema Manager

- Schema é uma descrição da estrutura do seu dado;
- Responsável por catalogar todos os schemas de dados;
- Permite a automatização da validação do seu dados;
- Exemplo: **JsonSchema, AVRO, Parquet, ...**

[Event Stream e Data Lake] Data Lake

- Repositório único de dados de toda a empresa;
- Facilitar o acesso dos times aos dados;
- Possibilidade de acesso aos dados históricos;
- Consolida e organizar todos os dados da empresa;
- Menor custo de armazenamento em blob storage;
- Exemplo: **GCS, S3, ...**

[Event Stream e Data Lake] Ferramentas ou desenvolver algo? 🤔

- Custo de aprendizado/entendimento?
- Monitoramento?
- Possibilidade de evoluir?
- Segurança operacional?
- Lock-in?
- Custooooooo ?

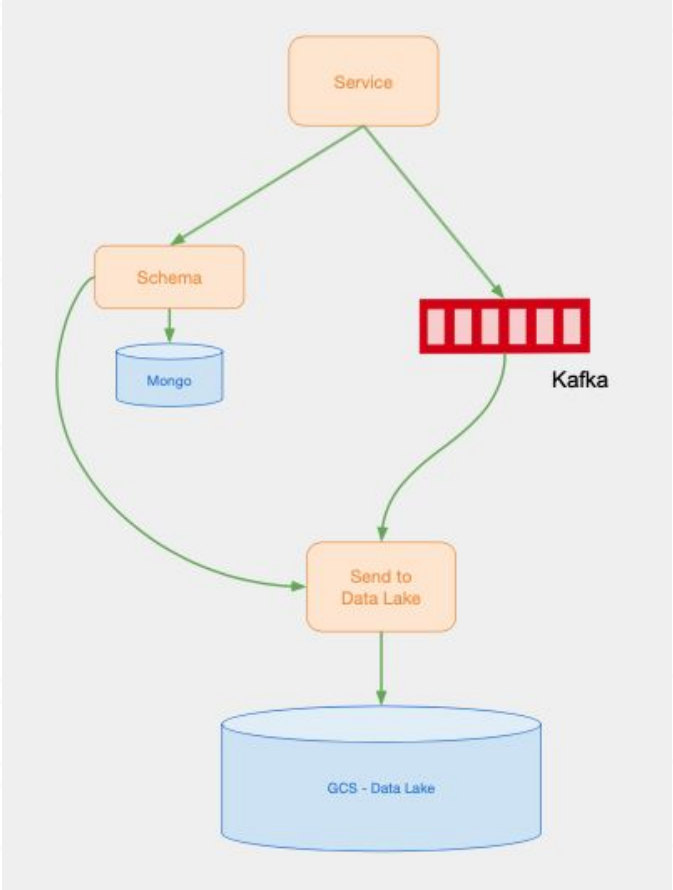
[Event Stream e Data Lake] Bora construir!!!!!

- Gestor de schemas: **Schema**
- Event Stream: **Kafka**
- ETL: **Flow**
- Data Lake: **Google Cloud Storage**

APACHE kafka®

Google Cloud Storage

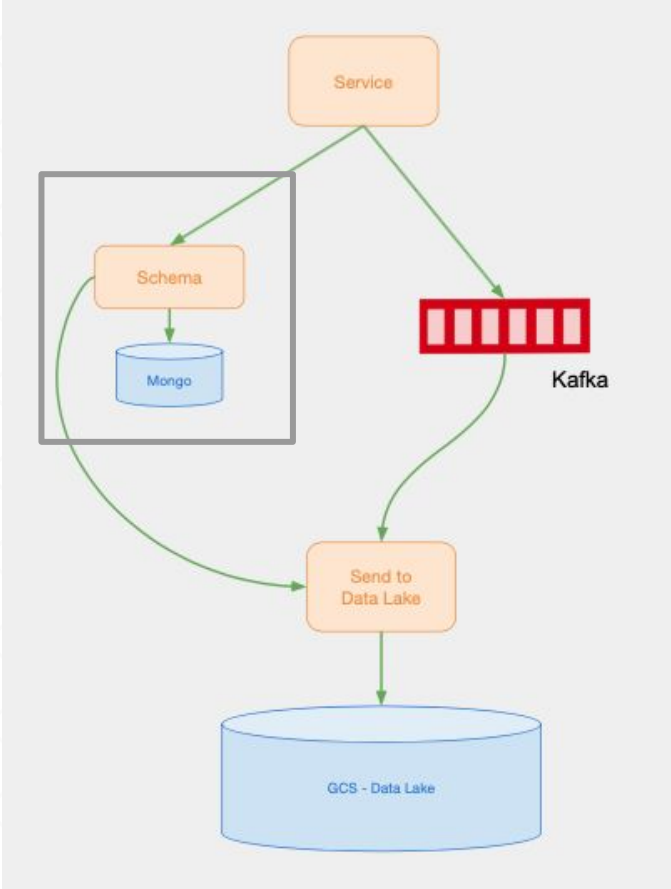
mongoDB




[Schema Manager]

[Event Stream e Data Lake] Schema Manager

- JsonSchema: nosso pipeline já usava JSON
- Imutabilidade
- Backward compatibility



[Event Stream e Data Lake] Lib JsonSchema

README.md

gojsonschema

Description

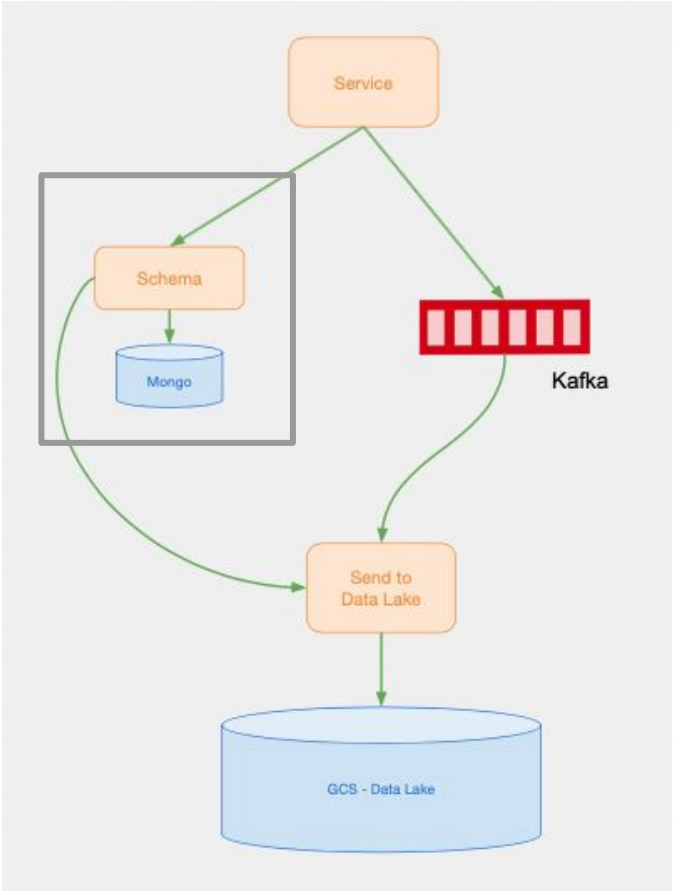
An implementation of JSON Schema for the Go programming language. Supports draft-04, draft-06 and draft-07.

References :

- http://json-schema.org
- http://json-schema.org/latest/json-schema-core.html
- http://json-schema.org/latest/json-schema-validation.html

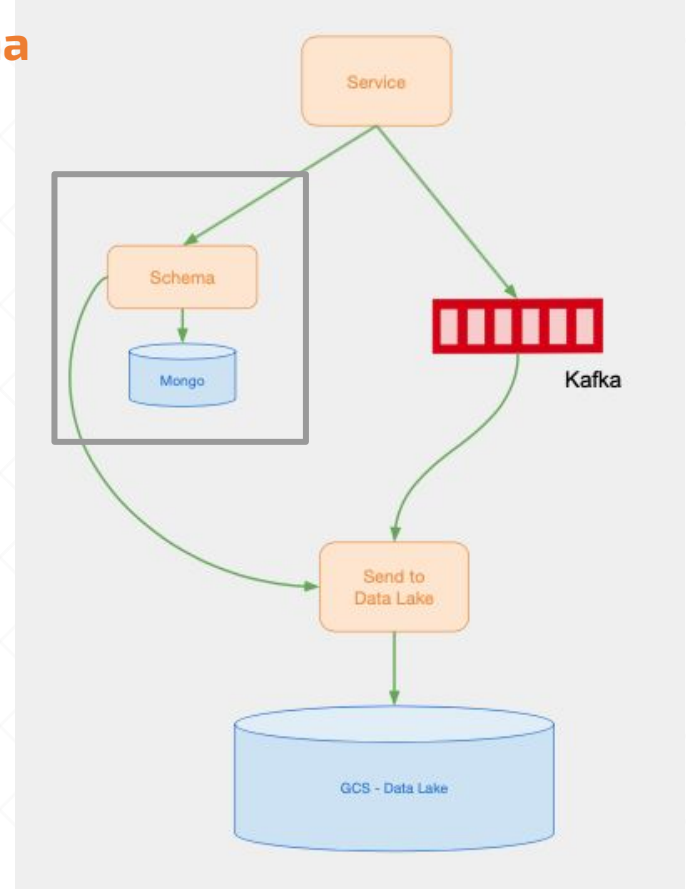
Installation

```
go get github.com/xeipuuv/gojsonschema
```



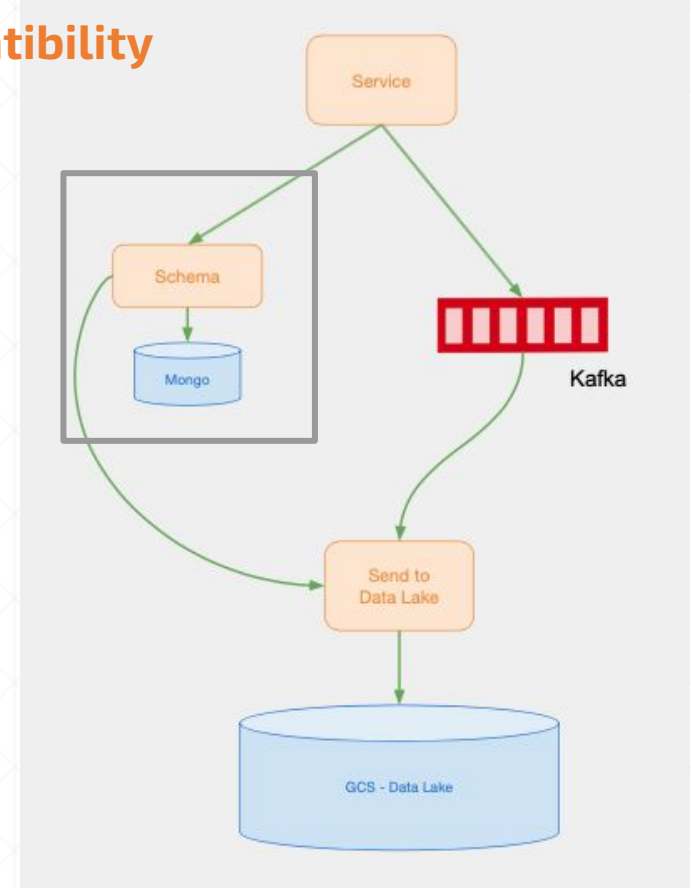
[Event Stream e Data Lake] Validate JsonSchema

```go
import (
    ...
    "github.com/xeipuuv/gojsonschema"
)
...

    schema := `{
        "title": "test",
        "type":  "object",
        "properties": {
            "name": {
                "type": "string",
            },
            "salary": {
                "type": "number",
            },
            "count": {
                "type": "integer",
            }
        },
        "required": ["name"]
    }`

    loader := gojsonschema.NewStringLoader(schema)
    _, err = gojsonschema.NewSchema(loader)

    if err != nil {
        fmt.Println("Deu ruim :S")
        return
    }
```



[Event Stream e Data Lake] Backward Compatibility

- O dado deve evoluir de forma a não quebrar os consumidores;
- **Você não pode!!!!**
  - Remover atributos;
  - Mudar o tipo dos atributos;

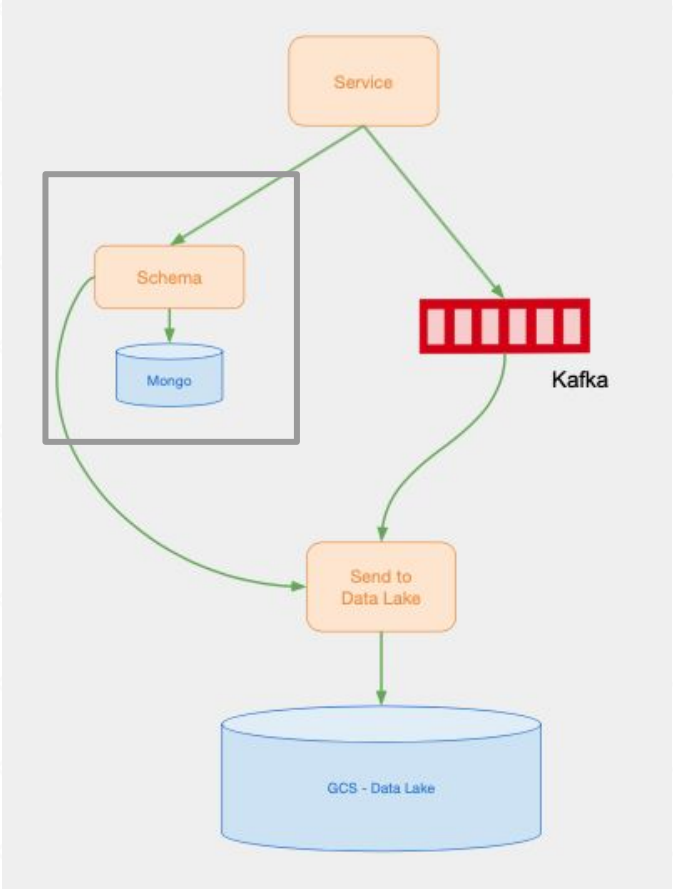


[Event Stream e Data Lake] Imutabilidade

- Todo schema possui um nome do dado:

  Exemplo: team-name => apps-users

- Um schema criado **NÃO** pode ser alterado!!
- Todo schema ganha um ID imutável;
- A cada evolução um novo ID é gerado;

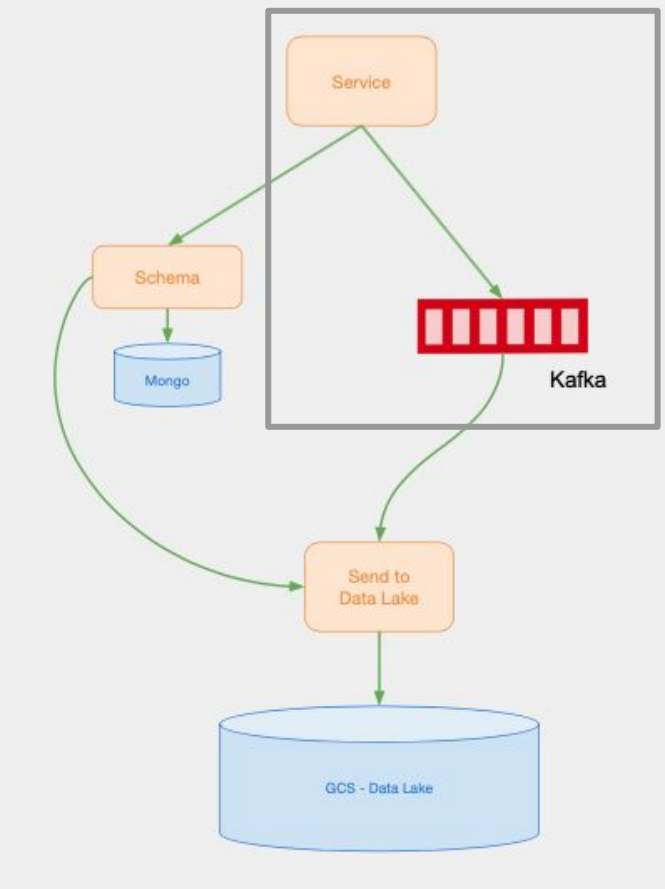



[Event Stream]

[Event Stream e Data Lake] Enviar o dado para o Kafka

- Pegar o **schema ID** registrado;
- Enviar o dado para o **Kafka;**
- O nome do tópico é o nome do schema:

  Exemplo: apps-users



[Event Stream e Data Lake] Lib Kafka

sarama

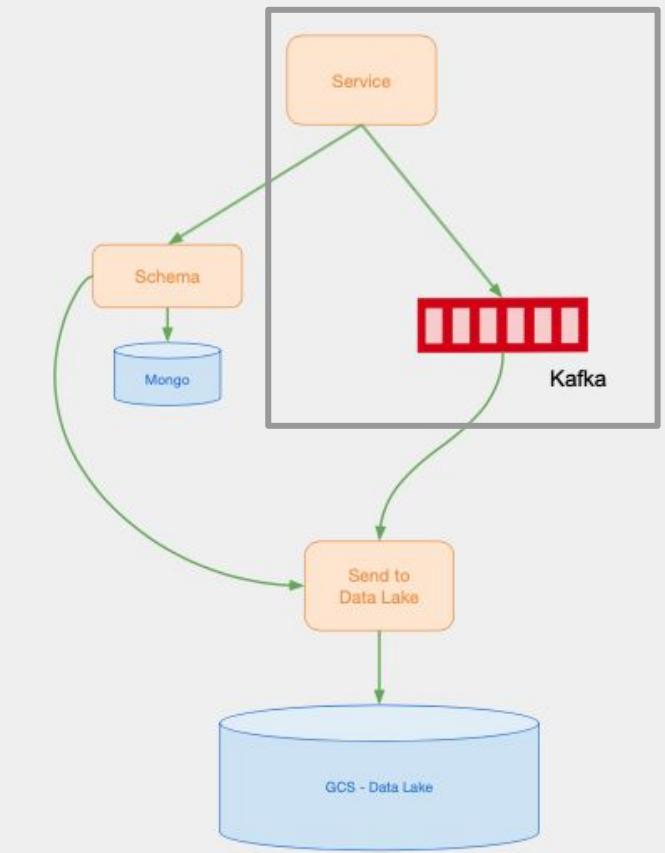
reference | build passing | codecov unknown

Sarama is an MIT-licensed Go client library for Apache Kafka version 0.8 (and later).

Getting started

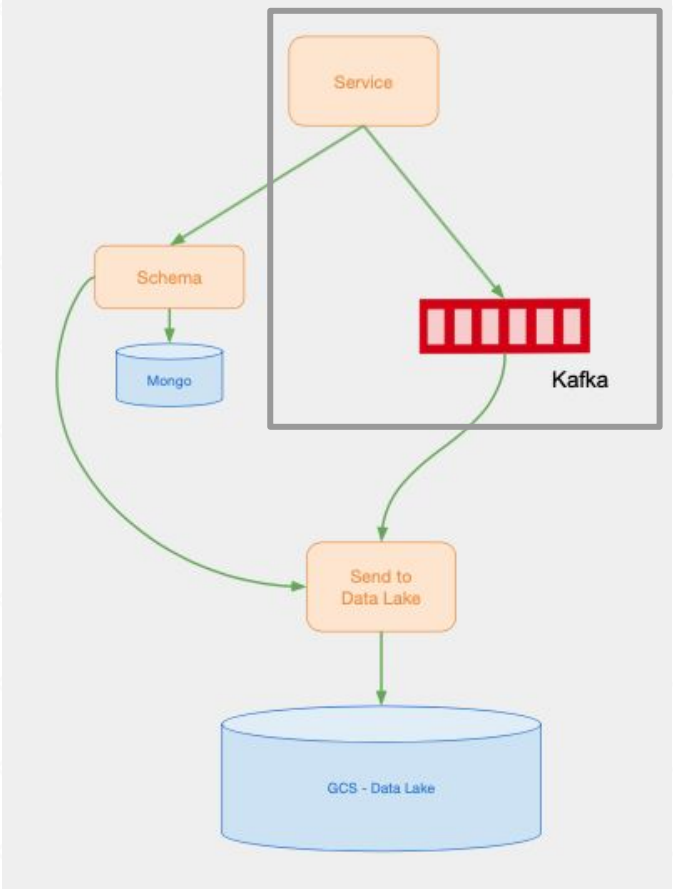
- API documentation and examples are available via pkg.go.dev.
- Mocks for testing are available in the mocks subpackage.
- The examples directory contains more elaborate example applications.
- The tools directory contains command line tools that can be useful for testing, diagnostics, and instrumentation.

You might also want to look at the Frequently Asked Questions.



[Event Stream e Data Lake] Lib Kafka

```go
import (
    ...
    "github.com/Shopify/sarama"
)
...

    conf := sarama.NewConfig()
    conf.Version = sarama.V2_5_0_0
    conf.Producer.RequiredAcks = sarama.WaitForAll
    conf.Producer.Timeout = 60 * time.Second
    conf.Producer.Return.Successes = true
    conf.Producer.Return.Errors = true
    conf.Producer.Retry.Max = 5
    conf.Producer.Flush.MaxMessages = 1
    conf.Producer.Compression = sarama.CompressionSnappy
    conf.Producer.MaxMessageBytes = 16777216

    producer, err := sarama.NewSyncProducer(brokers, conf)
    if err != nil {
        return nil, err
    }

    headers := []sarama.RecordHeader{
        {
            Key:   []byte("schemaID"),
            Value: []byte("1"),
        },
    }

    topic := "apps-users"
    key := []byte(`1`)
    msg := []byte(`{
        "name": "Fulano da Silva",
        "salary": 10000.1,
        "count": 10
    }`)
    message := &sarama.ProducerMessage{
        Topic:     topic,
        Partition: -1,
        Key:       sarama.ByteEncoder(key),
        Value:     sarama.ByteEncoder(msg),
        Headers:   headers,
    }

    _, _, err := producer.SendMessage(message)
    fmt.Println(err)
```




[Data Lake]

[Event Stream e Data Lake] Flow

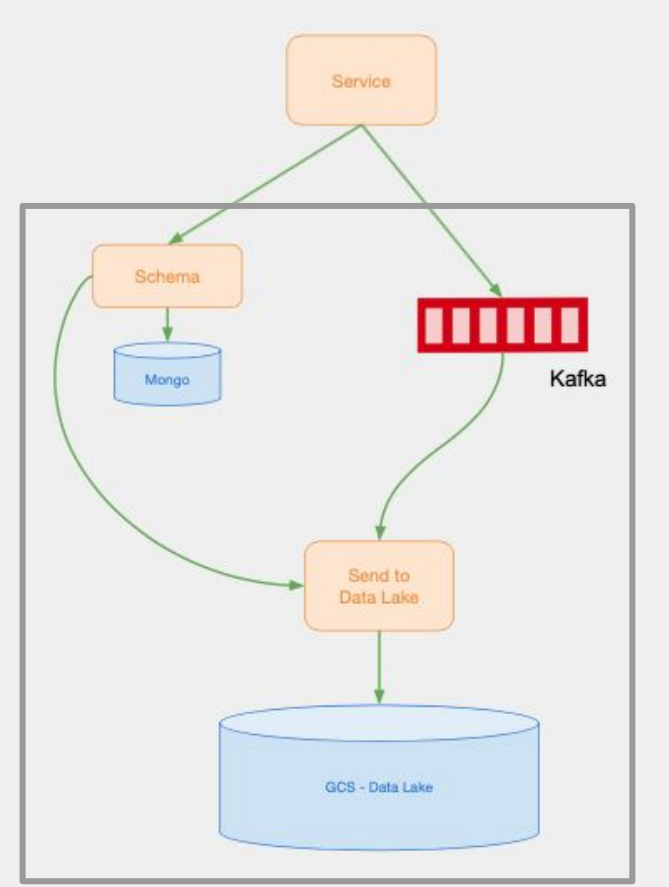
1 - Lê todos os schemas registrados

2 - A partir do nome ele descobre o tópico

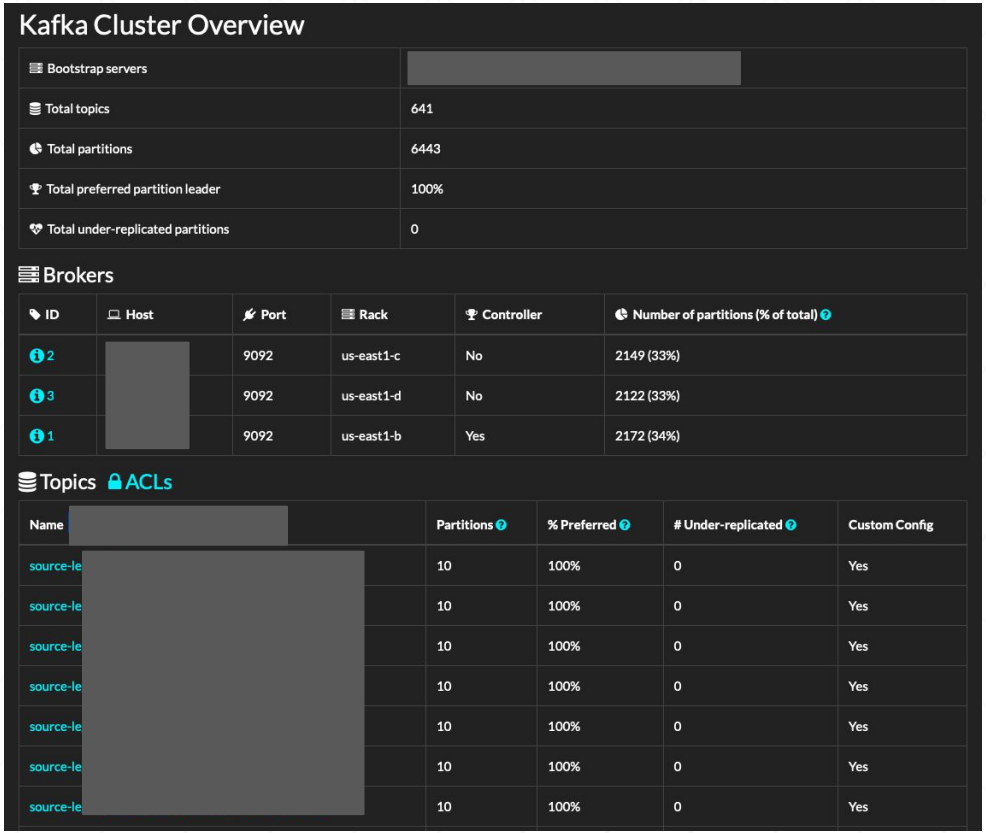
3 - Começa a ler todos os dados de cada tópico

4 - Consolida um arquivo com os dados recebidos

5 - Salva no Data Lake



[Event Stream e Data Lake] Kafka



Kafka Cluster Overview

Bootstrap servers	[redacted]
Total topics	641
Total partitions	6443
Total preferred partition leader	100%
Total under-replicated partitions	0

Brokers

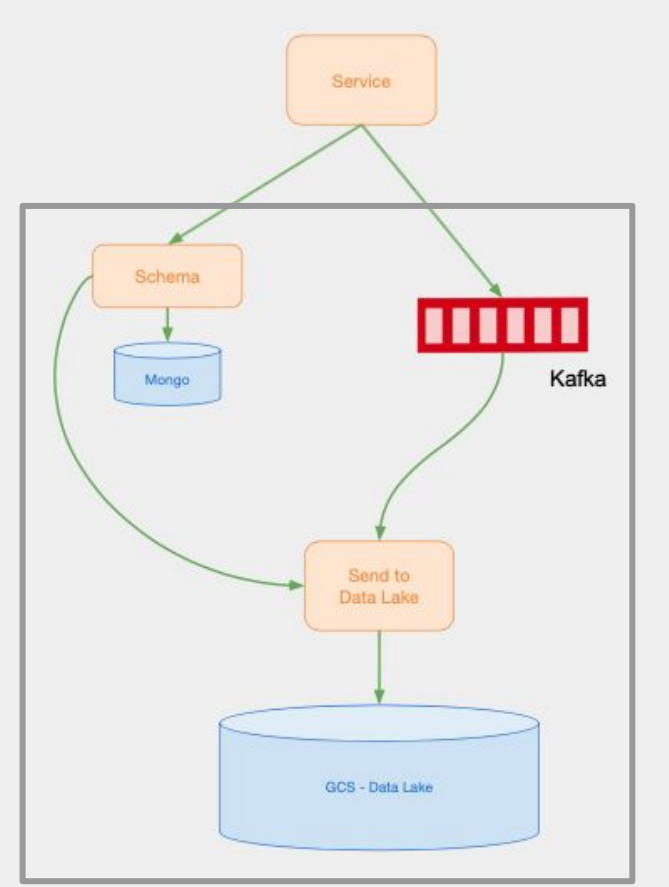
ID	Host	Port	Rack	Controller	Number of partitions (% of total)
2	[redacted]	9092	us-east1-c	No	2149 (33%)
3		9092	us-east1-d	No	2122 (33%)
1		9092	us-east1-b	Yes	2172 (34%)

Topics ACLs

Name [redacted]	Partitions	% Preferred	# Under-replicated	Custom Config
source-le[redacted]	10	100%	0	Yes
source-le	10	100%	0	Yes
source-le	10	100%	0	Yes
source-le	10	100%	0	Yes
source-le	10	100%	0	Yes
source-le	10	100%	0	Yes
source-le	10	100%	0	Yes

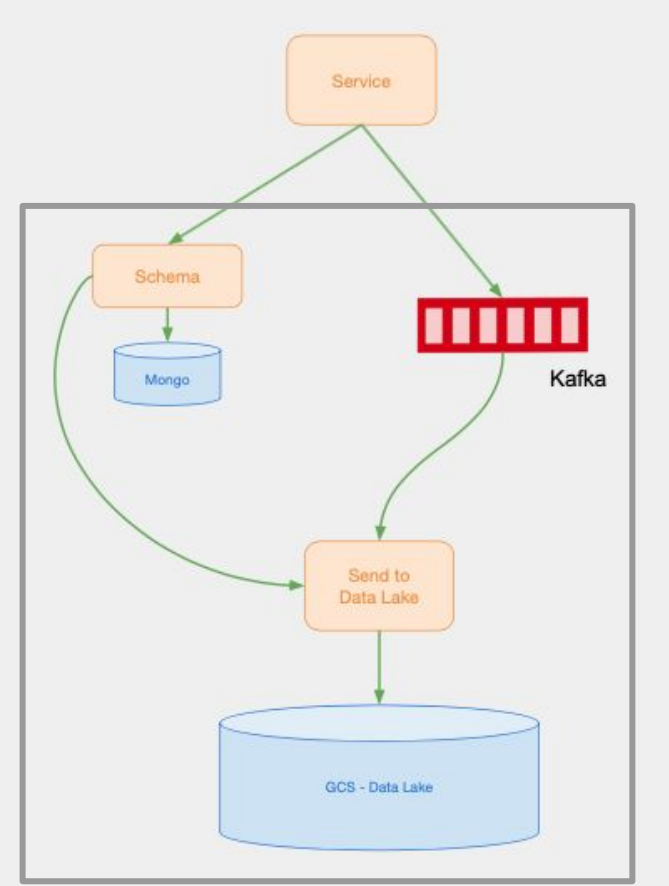
[Event Stream e Data Lake] Lendo dados

```go
import (
    ...
    "github.com/Shopify/sarama"
)

type Message struct {
    session     sarama.ConsumerGroupSession
    consumerMsg *sarama.ConsumerMessage
    Partition   int
    Topic       string
    headers     []*sarama.RecordHeader
    Value       []byte
    Key         []byte
}

type consumerHandler struct {
    messages chan Message
}

func (c *consumerHandler) Setup(session sarama.ConsumerGroupSession) error {
    return nil
}

func (c *consumerHandler) Cleanup(session sarama.ConsumerGroupSession) error {
    return nil
}

func (c *consumerHandler) ConsumeClaim(session sarama.ConsumerGroupSession, claim sarama.ConsumerGroupClaim) error {
    for msg := range claim.Messages() {
        c.messages <- Message{
            session:     session,
            consumerMsg: msg,
            Topic:       msg.Topic,
            Partition:   int(msg.Partition),
            headers:     msg.Headers,
            Value:       msg.Value,
            Key:         msg.Key,
        }
    }
    return nil
}
```



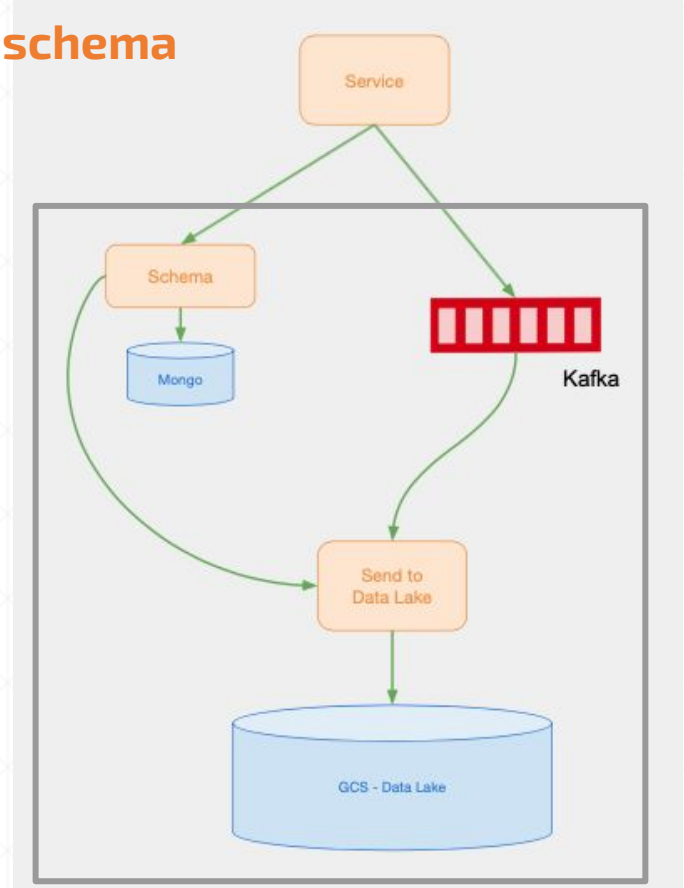
[Event Stream e Data Lake] Lendo dados

```
messages := make(chan Message)
errors := make(chan error)

brokers := "localhost:9092"
topic := "apps-users"
group := "service-name"

conf := sarama.NewConfig()
conf.Version = sarama.V2_5_0_0

conf.Consumer.Return.Errors = true
conf.Consumer.Offsets.Initial = sarama.OffsetOldest
conf.Consumer.Group.Rebalance.Timeout = 1 * time.Second

client, err := sarama.NewConsumerGroup(k.brokers, group, k.conf)
if err != nil {
    errors <- err
    return messages, errors
}

go func() {
    for err := range client.Errors() {
        errors <- err
    }
    client.Close()
}()

handler := &consumerHandler{
    messages: messages,
}

go func() {
    for {
        err := client.Consume(ctx, topics, handler)
        if err != nil {
            errors <- err
        }
    }
}()

msg <- messages
fmt.Println("Message: ", msg)
```



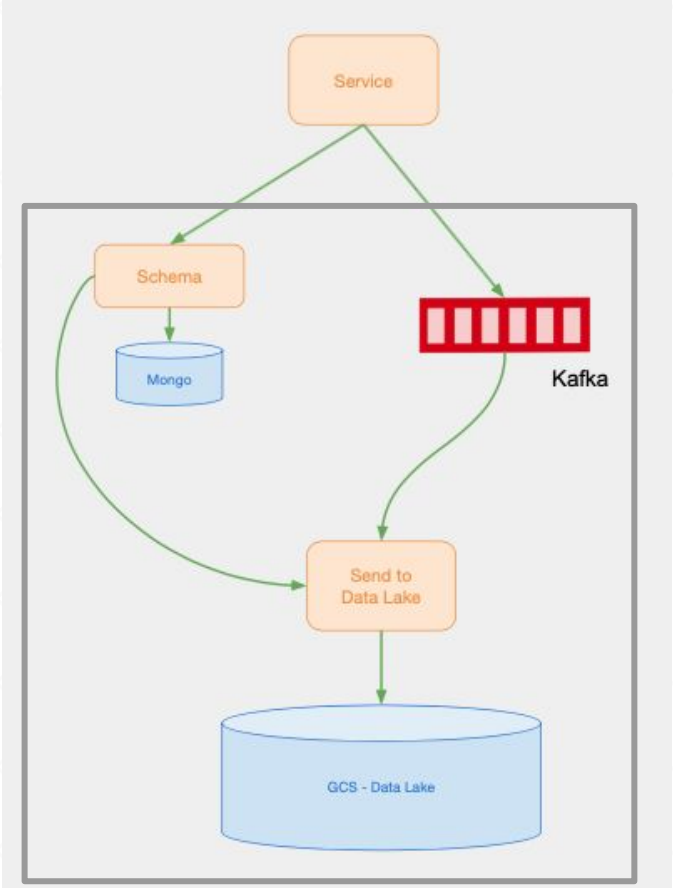
[Event Stream e Data Lake] Validando dado e schema

```go
import (
    ...
    "github.com/xeipuuv/gojsonschema"
)

func main() {
    schema := `{
        "title": "test",
        "type":  "object",
        "properties": {
            "name": {
                "type": "string",
            },
            "salary": {
                "type": "number",
            },
            "count": {
                "type": "integer",
            }
        },
        "required": ["name"]
    }`

    loader := gojsonschema.NewStringLoader(schema)
    schemaLoader, _ = gojsonschema.NewSchema(loader)

    data := `{
        "name": "Fulano da Silva",
        "salary": 10000.1,
        "count": 10
    }`
    dataLoader := gojsonschema.NewGoLoader(data)
    result, err := schemaLoader.Validate(dataLoader)

    if result.Valid() {
        fmt.Println("Segue o jogo!")
    }
}
```



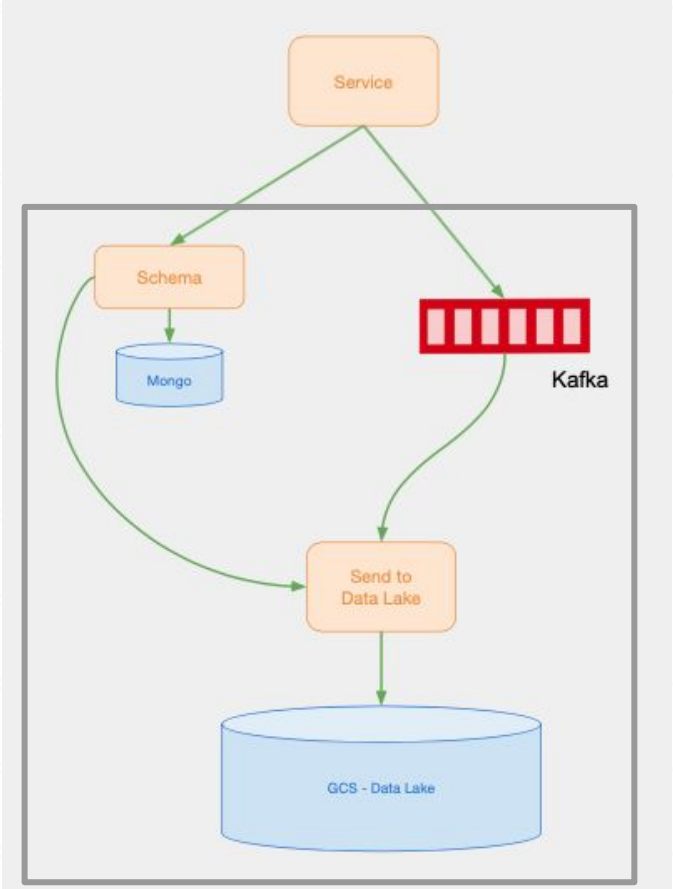
[Event Stream e Data Lake] Enviando dados

```
    go func(configName string, metadataChan chan interface{}) {
        for meta := range metadataChan {
            // commit offset kafka
        }
    }(configName, metadataChan)
...
    for {
        select {
        case <-ticker.C:
            if msgsBuffer.Len() > 0 {
                // save file
                metadataChan <- lastOffsetKafka
                msgsBuffer.Reset()
            }
        case m := <-c.msgChan:
            // append message
            if msgsBuffer.Len() >= bs.appendSize {
                // save file
                metadataChan <- lastOffsetKafka
                msgsBuffer.Reset()
            }
        }
    }
```

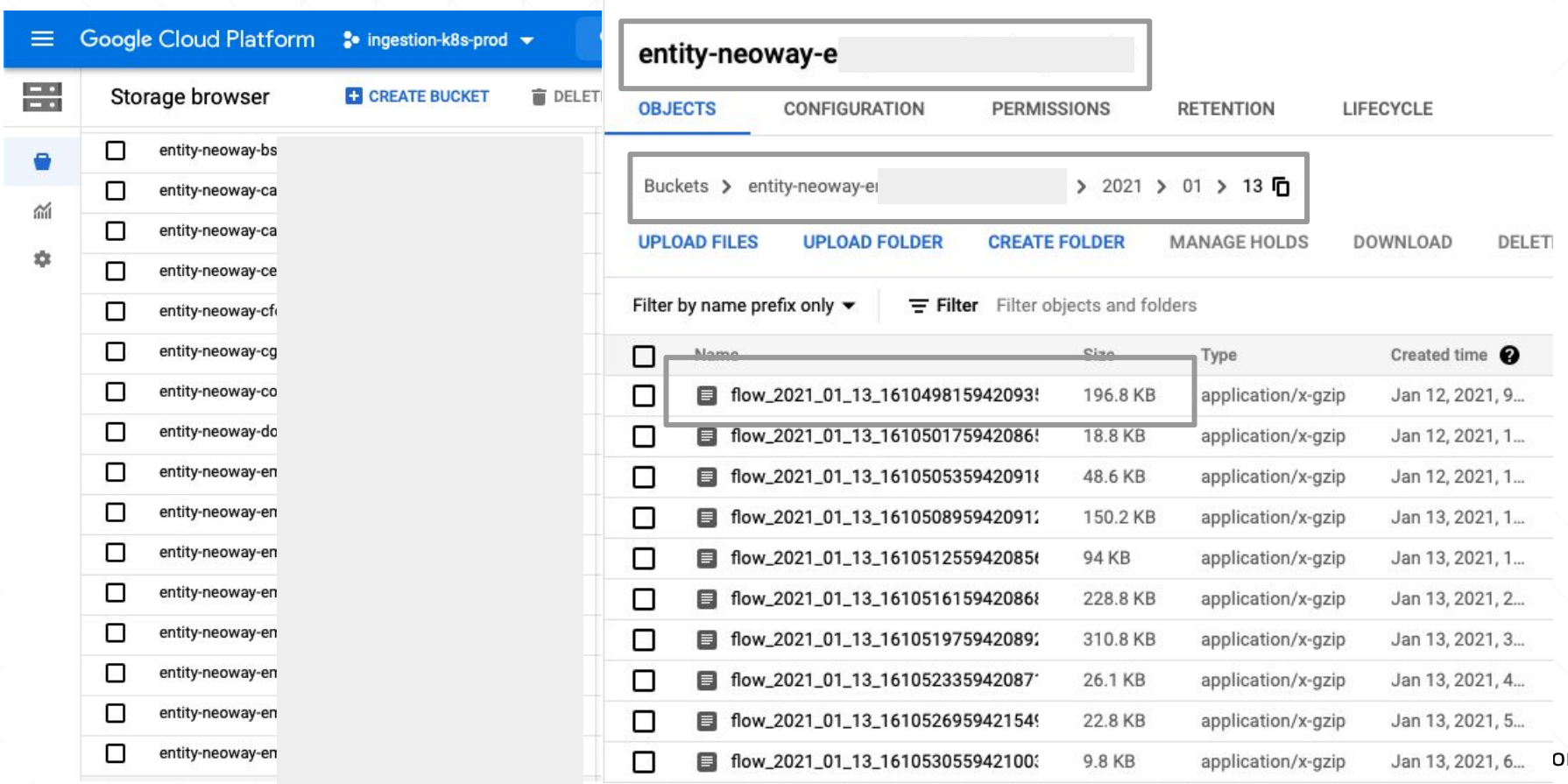


[Event Stream e Data Lake] Enviando dados

```go
import (
    ...
    gcpstorage "cloud.google.com/go/storage"
)

    projectID := "project-gcp"
    bucketName := "apps-users"

    client, _ := gcpstorage.NewClient(ctx)
    bucket := client.Bucket(bucketName).UserProject(projectID)

    bucketAttrs := &gcpstorage.BucketAttrs{
        RequesterPays: true,
        Location: "us-east1",
    }
    err := bucket.Create(ctx, projectID, bucketAttrs)
    ...

    fileName := "some-file"
    contents := strings.NewReader(`
    {"name": "Fulano da Silva","salary": 10000.1,"count": 10}
    {"name": "Ciclano da Silva","salary": 900.1,"count": 30}
    `)
    wc := bucket.Object(fileName).NewWriter(ctx)
    if _, err := io.Copy(wc, contents); err != nil {
        return err
    }
    if err := wc.Close(); err != nil {
        return err
    }
```



[Event Stream e Data Lake] Flow



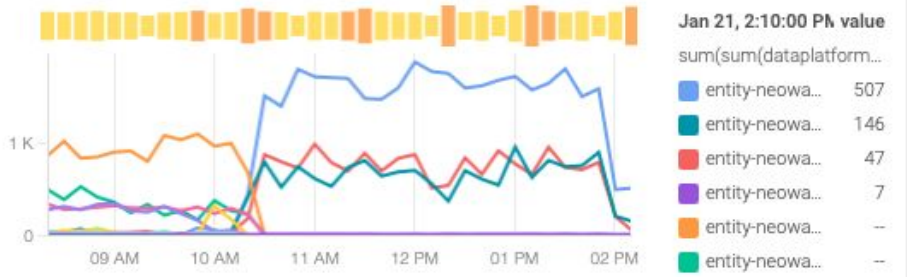
om.br

[Monitoramentos]

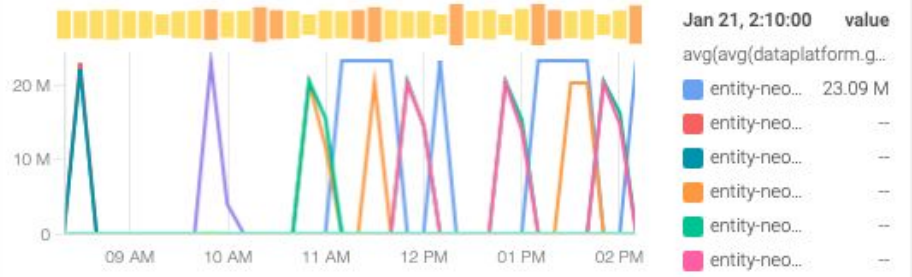
[Event Stream e Data Lake] Monitoramento

Data Lake Gzip - GCS Message Size

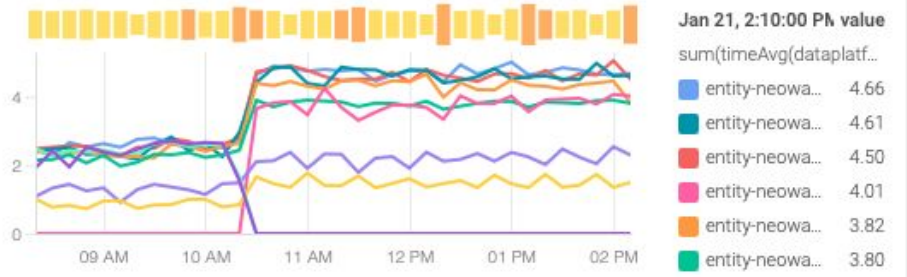
Data Lake Gzip - GCS Message Count



Flow Parquet - GCS Message Size



Flow Parquet - GCS Message Count



StatsDig

Yeah, it is a pretty cheesy name for the idea of mixing up the StatsD protocol and Sysdig Cloud services.

So why another statsd Golang client ? Well, actually there is not a lot of reason, but there is some.

Depending on how you send UDP packets on Go it will work perfectly fine when there is someone listening on the port where the packets are being sent, but when they are not, your metrics won't be collected correctly by the sysdig agent (because Go will not even send them).

Why would I send packets to ports where no one is listening ? Well Sysdig is that magic, it will use kernel introspection to collect the UDP packets independent from where they are being sent. It is a very common use case to just sent to localhost on the default port.

We optimize for this use case and make things as simple as possible. Also we added support to Sysdig extension of tags as a first class citizen on the API.

Also we add here some tools that helped us to debug problems like metrics disappearing (usually our fault), etc.

More on Sysdig StatsD magic:

- Metrics Integration: StatsD
- StatsD Teleportation

Sysdig magic has a know limitation that required the udp to be under IPv4, so the library is hardcoded to always use **udp4** as the protocol.

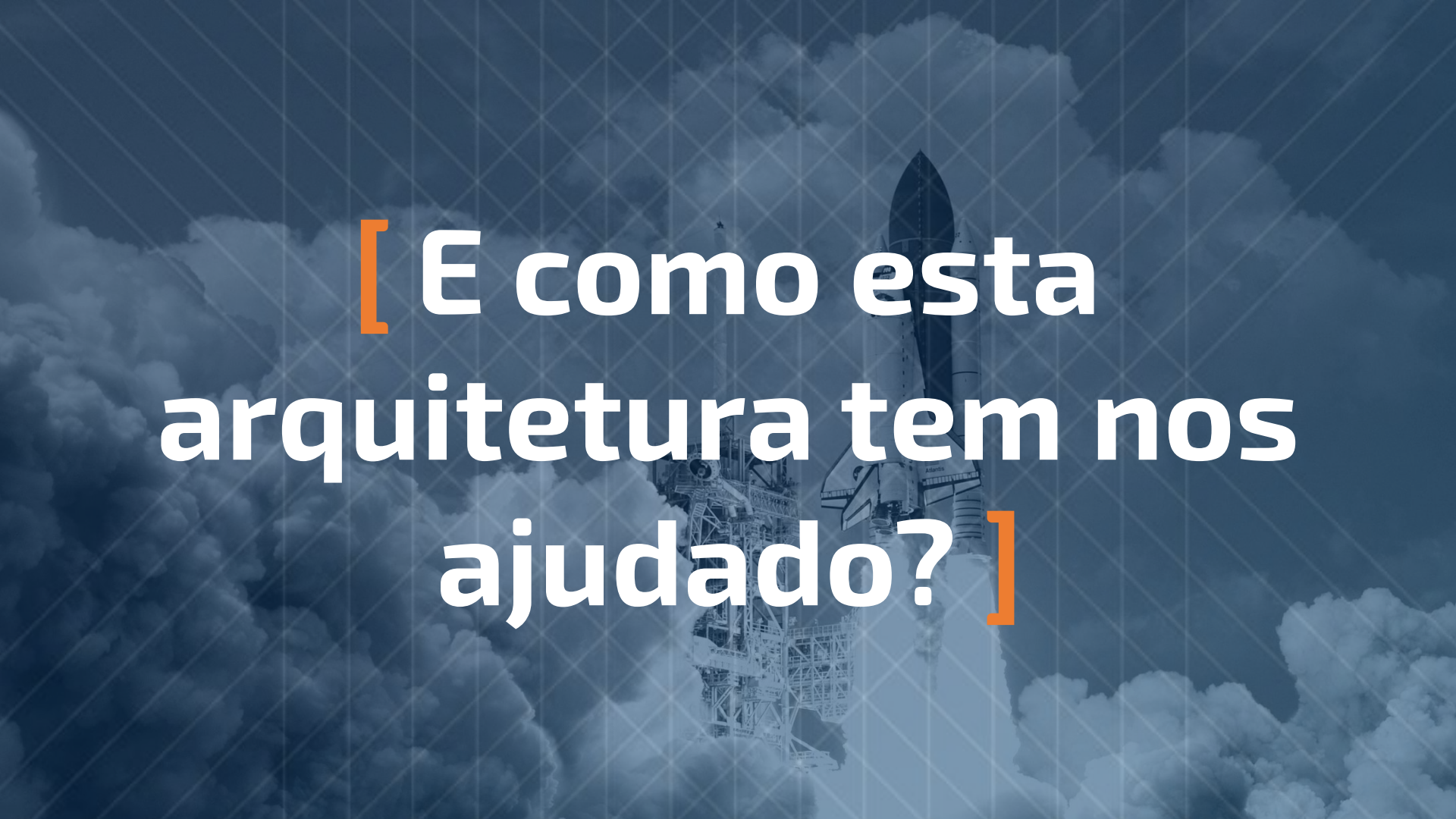
API Reference

https://github.com/NeowayLabs/statsdig

API Reference docs

[Event Stream e Data Lake] Monitoramento

```
import (
    "github.com/NeowayLabs/statsdig"
)
...
    sampler, err := statsdig.NewSysdigSampler()
    if err != nil {
        return
    }

    // increment +1
    sampler.Count("flow.blobstorage.count", statsdig.Tag{
        Name:  "schema",
        Value: "apps-users",
    })

    // increment 100 bytes
    sampler.Gauge("flow.blobstorage.file.size.gauge", 100, statsdig.Tag{
        Name:  "schema",
        Value: "apps-users",
    })
```


[E como esta
arquitetura tem nos
ajudado?]

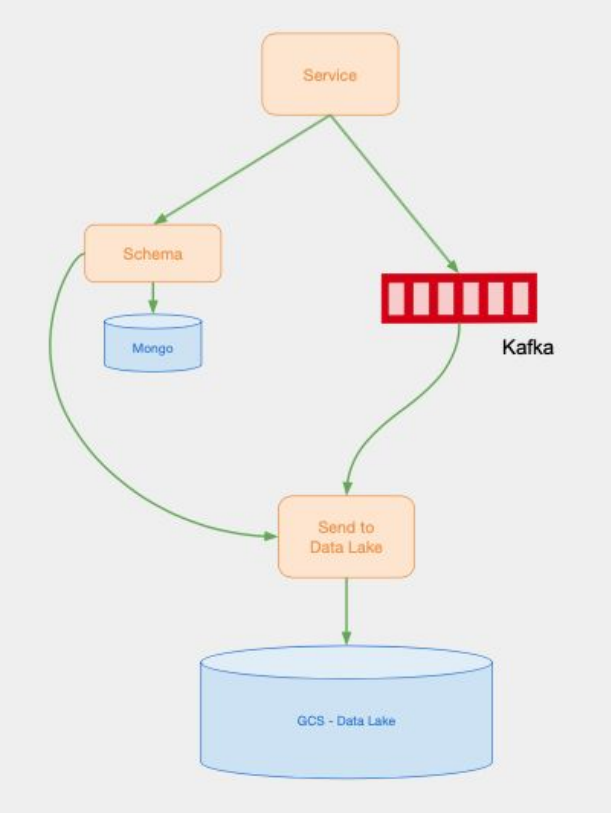
[Event Stream e Data Lake] Como esta arquitetura tem nos ajudado?

1 - Acessibilidade e entendimento aos dados;

- Governança/LGPD;

2 - Facilitar a criação de processamentos em stream/batch;

3 - Criação de novas engines;




Venha ser NEOWAY
Assista ao vídeo
Sobre nos
Vagas
Neoway
https://jobs.kenoby.com/neoway/