Gerindo bilhões de dados com **uma arquitetura event stream em Go**

Neoway LABS TECH

Matheus Vill

VP of Platform at Neoway

linkedin.com/in/matheusvill

Neoway
Uma Empresa B3

Como a Neoway ajuda os clientes na tomada de decisão?

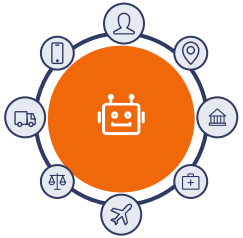
COLETA

API

http://www.

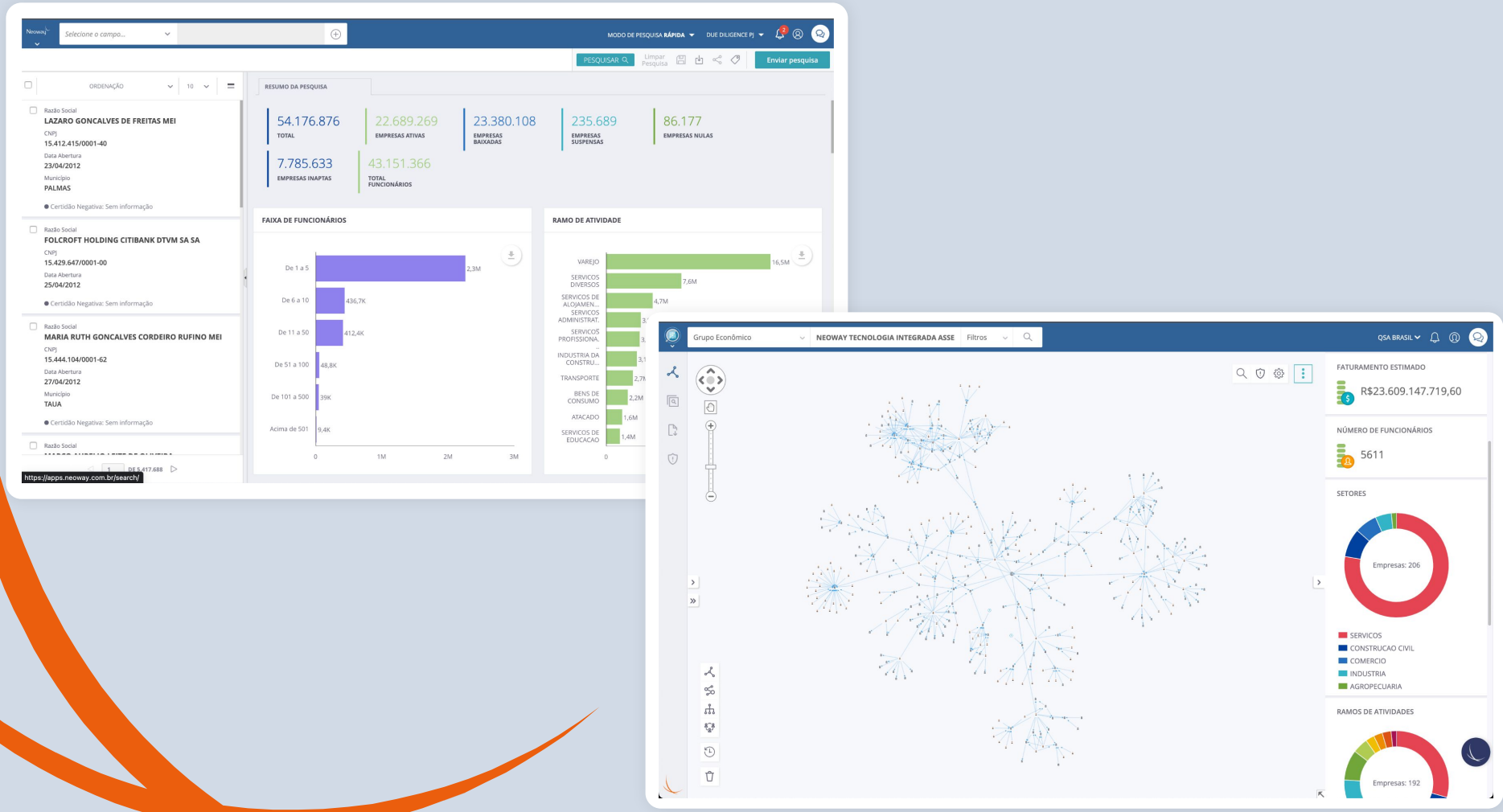
PDF, TXT, CSV, XLS e HTML

FLUXO DE DADOS



ANALYTICS E INSIGHTS

APLICAÇÕES


Selecione o campo...
MODO DE PESQUISA RÁPIDA
DUE DILIGENCE PJ
PESQUISAR
Limpar Pesquisa
Enviar pesquisa
ORDENAÇÃO
RESUMO DA PESQUISA
54.176.876
TOTAL
22.689.269
EMPRESAS ATIVAS
23.380.108
EMPRESAS BAIXADAS
235.689
EMPRESAS SUSPENSAS
86.177
EMPRESAS NULAS
7.785.633
EMPRESAS INAPTAS
43.151.366
TOTAL FUNCIONÁRIOS
Razão Social
LAZARO GONCALVES DE FREITAS MEI
CNPJ
15.412.415/0001-40
Data Abertura
23/04/2012
Município
PALMAS
Certidão Negativa: Sem informação
Razão Social
FOLCROFT HOLDING CITIBANK DTVM SA SA
CNPJ
15.429.647/0001-00
Data Abertura
25/04/2012
Certidão Negativa: Sem informação
Razão Social
MARIA RUTH GONCALVES CORDEIRO RUFINO MEI
CNPJ
15.444.104/0001-62
Data Abertura
27/04/2012
Município
TAUA
Certidão Negativa: Sem informação
Razão Social
FAIXA DE FUNCIONÁRIOS
De 1 a 5
2,3M
De 6 a 10
436,7K
De 11 a 50
412,4K
De 51 a 100
48,8K
De 101 a 500
39K
Acima de 501
9,4K
0
1M
2M
3M
RAMO DE ATIVIDADE
VAREJO
16,5M
SERVICOS DIVERSOS
7,6M
SERVICOS DE ALOJAMEN...
4,7M
SERVICOS ADMINISTRAT.
SERVICOS PROFISSIONA...
INDUSTRIA DA CONSTRU...
TRANSPORTE
2,7M
BENS DE CONSUMO
2,2M
ATACADO
1,6M
SERVICOS DE EDUCACAO
1,4M
https://apps.neoway.com.br/search/
Grupo Econômico
NEOWAY TECNOLOGIA INTEGRADA ASSE
Filtros
QSA BRASIL
FATURAMENTO ESTIMADO
R$23.609.147.719,60
NÚMERO DE FUNCIONÁRIOS
5611
SETORES
Empresas: 206
SERVICOS
CONSTRUCAO CIVIL
COMERCIO
INDUSTRIA
AGROPECUARIA
RAMOS DE ATIVIDADES
Empresas: 192

Neoway

Uma Empresa B3

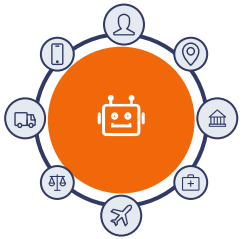
COLETA

API

http://www.

PDF, TXT, CSV, XLS e HTML

FLUXO DE DADOS

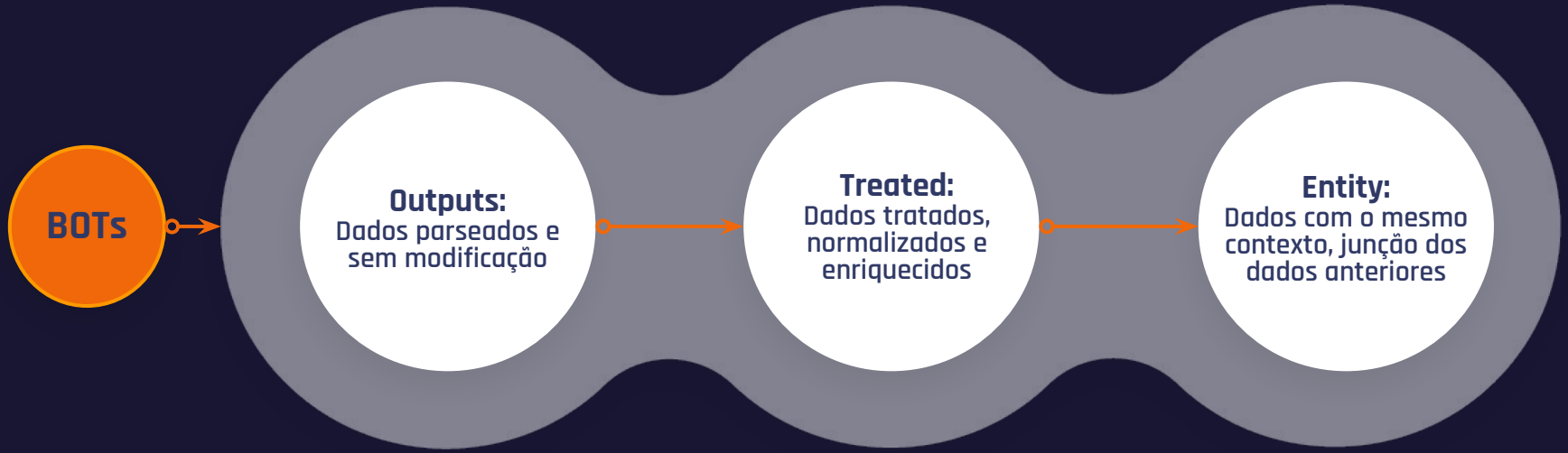


ANALYTICS E INSIGHTS

APLICAÇÕES

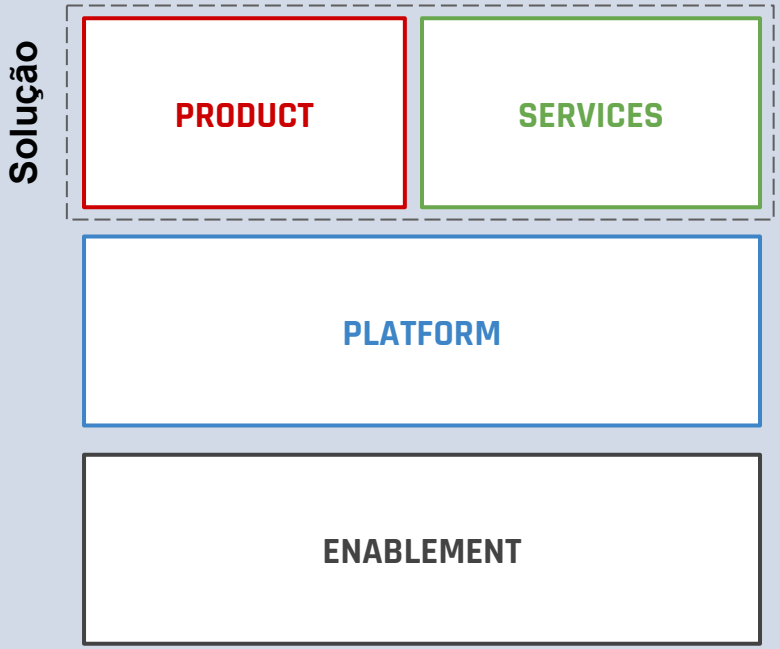
Vamos entender um pouco mais sobre como funciona o **Fluxo de Dados**

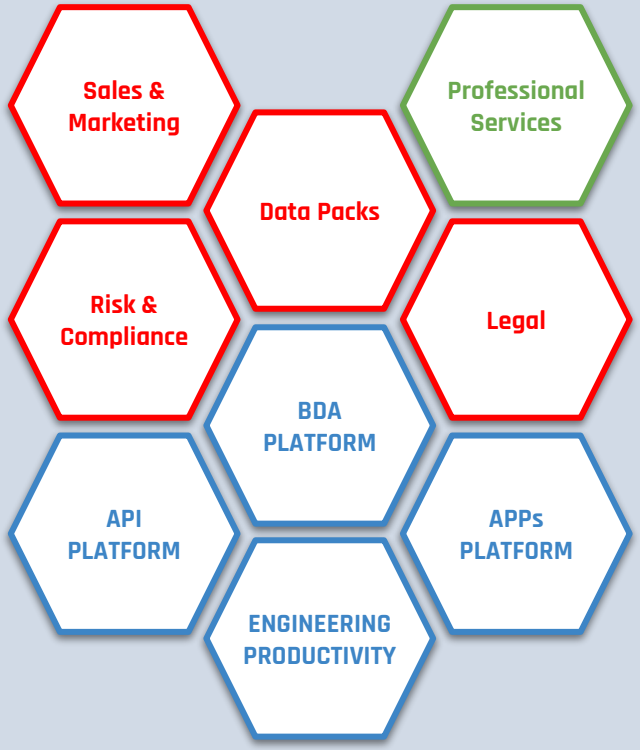
Fluxo de Dados



Fluxo de dados como **PLATAFORMA**

Como organizamos nossos times...

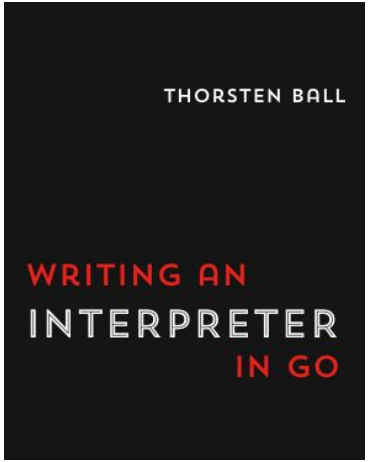




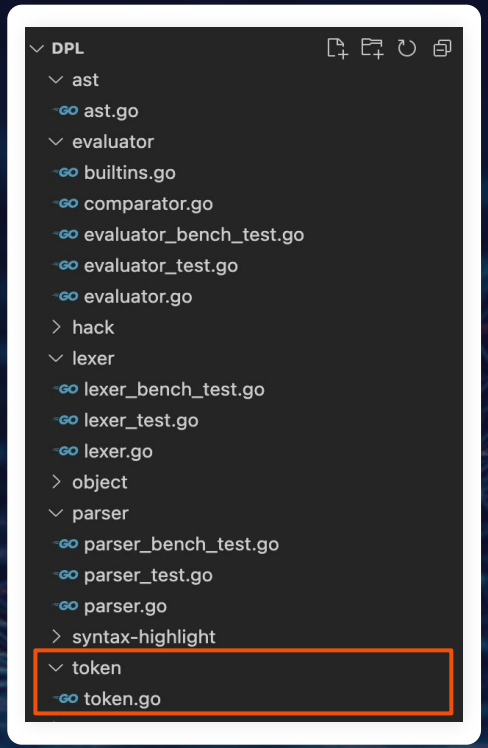
DPL: Data Processing Language

- Linguagem de propósito específico
- Interpretada em Golang
- Isolamento e Segurança
- Aceleração do processo de desenvolvimento

```
entity "fipe" {
    data.marca
    data.nome
    data.combustivel
    data.ano
}

sources "fipe"

mandatory {
    data.codigoFipeTreated
    metadata.input.modelo.codigo
    data.valor
    data.marcaTreated
}

add {
    codigoFipeTreated -> codigoFipe
    combustivelTreated -> combustivel
    marcaTreated -> marca
    ano
    nomeTreated -> nome
    valor
    referencia
}
```

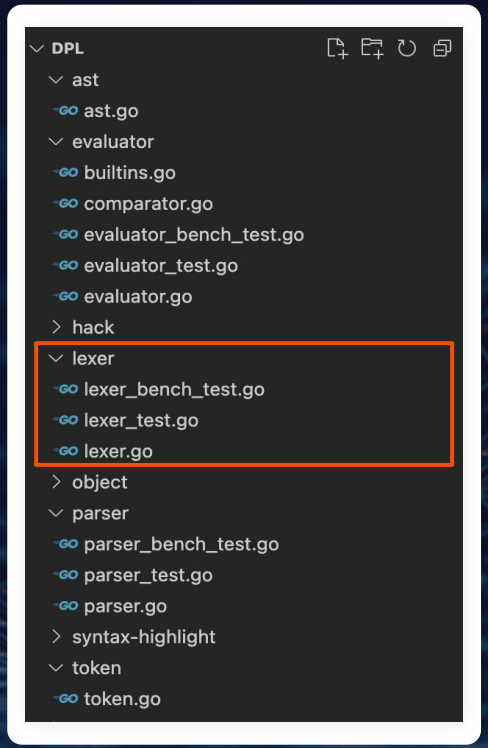


DPL: tokens



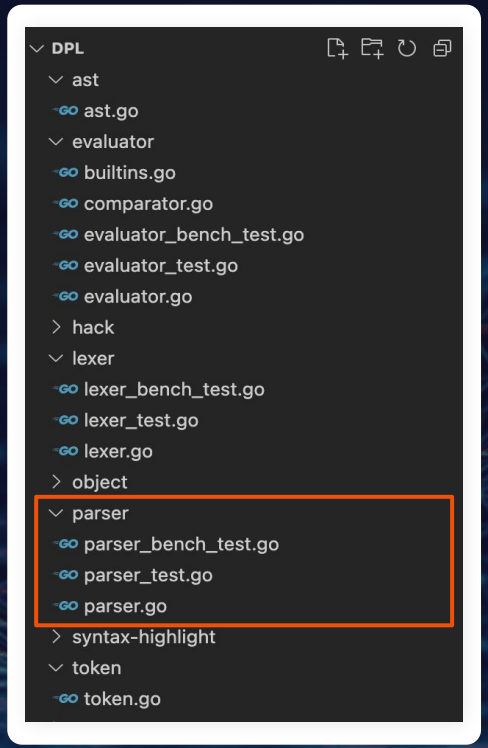

```go
var keywords = map[string]TokenType{
    "entity":          ENTITY,
    "substitutive":    SUBSTITUTIVE,
    "using":           USING,
    "as":              AS,
    "sources":         SOURCES,
    "requirement":     REQUIREMENT,
    "mandatory":       MANDATORY,
    "add":             ADD,
    "all":             ALL,
    "append":          APPEND,
    "delete":          DELETE,
    "true":            TRUE,
    "false":           FALSE,
    "if":              IF,
    "'id'":            ID,
    "'merge'":         MERGE,
    "'substitutive'": SUBSTITUTIVE_ATTRIBUTE,
    "from":            FROM,
    "len":             LEN,
    "sum":             SUM,
    "overwrite":       OVERWRITE,
}
```

DPL: Análise Léxica



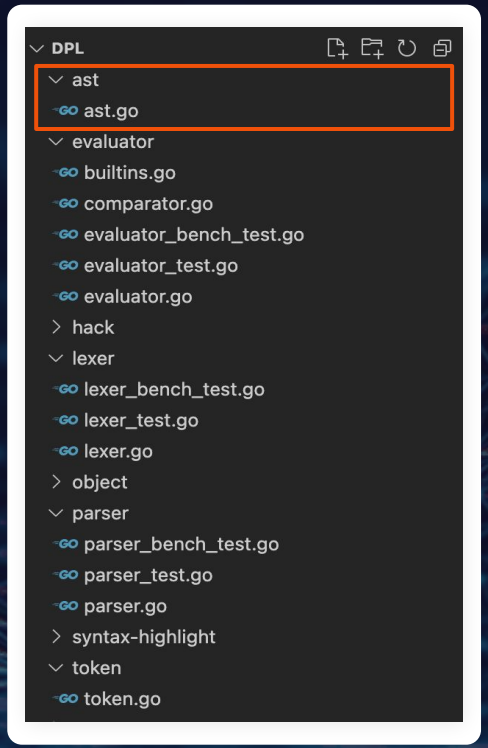

```go
func (l *Lexer) readIdentifier() string {
    position := l.position
    for l.isLetter(l.ch) || l.isDigit(l.ch) {
        l.readChar()
    }
    return l.input[position:l.position]
}

func (l *Lexer) isLetter(ch byte) bool {
    inRules := 'a' <= ch && ch <= 'z' || 'A' <= ch && ch <= 'Z' || ch == '.' || ch == '\'' || ch == '_'
    return inRules
}

func (l *Lexer) readChar() {
    if l.readPosition >= len(l.input) {
        l.ch = 0
    } else {
        l.ch = l.input[l.readPosition]
    }
    l.position = l.readPosition
    l.readPosition += 1
}
```

DPL: Parser



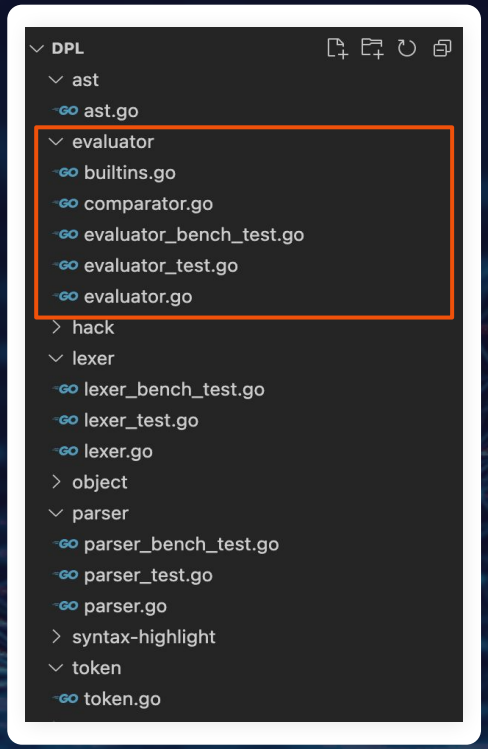

```go
func (p *Parser) parseValue() (ast.Expression, error) {
    if p.peek.Type == token.TO || p.isValueToken(p.peek.Type) {
        return nil, p.getError(Error{expName: "To", tokGot: p.peek, tokExpect: []string{token.IDENT}})
    }

    var err error
    var value interface{}

    switch p.current.Type {
    case token.INT:
        value, err = strconv.Atoi(p.current.Literal)
        if err != nil {
            return nil, p.getError(Error{msg: "expected an int value"})
        }
    case token.FLOAT:
        value, err = strconv.ParseFloat(p.current.Literal, 64)
        if err != nil {
            return nil, p.getError(Error{msg: "expected an float value"})
        }
    case token.TRUE:
        value = true
    case token.FALSE:
        value = false
    default:
        value = p.current.Literal
    }

    return &ast.Value{Token: p.current, Value: value}, nil
}
```

DPL: Abstract Syntax Tree




```go
type Identifier struct {
    Token              token.Token
    Name               string
    FunctionExpression *FunctionExpression
}

func (i *Identifier) expressionNode()      {}
func (i *Identifier) TokenLiteral() string { return i.Token.Literal }
func (i *Identifier) String() string       { return i.Name }

type FunctionExpression struct {
    Token        token.Token
    IfExpression Expression
}

func (fe *FunctionExpression) expressionNode()      {}
func (fe *FunctionExpression) TokenLiteral() string { return fe.Token.Literal }
func (fe *FunctionExpression) String() string {
    var out bytes.Buffer
    out.WriteString(fe.TokenLiteral() + " ")
    out.WriteString(fe.IfExpression.String() + " ")
    return out.String()
}
```

DPL: Evaluate




```go
func (e *Evaluator) evalEntity(s *ast.EntityStatement, source Source) (string, error) {
    var id string

    for _, exp := range s.Keys {
        switch s := exp.(type) {
        case *ast.Identifier:
            path := s.Name
            _, value, err := e.getValue(path, source)
            if err != nil {
                return "", getError(Error{msg: err.Error(), instruction: "entity", fields: []interface{}{path}})
            }
            if value != nil {
                id += fmt.Sprint(value)
            }
        case *ast.InfixExpression:
            _, value, err := e.evalInfixFieldExpression(s, source)
            if err != nil {
                return "", err
            }

            if value != nil {
                id += fmt.Sprint(value)
            }
        case *ast.IfExpression:
            _, value, err := e.evalIfFieldExpression(s, source)

            if err != nil {
                return "", err
            }

            if value != "" {
                id += fmt.Sprint(value)
            }
        }
    }
```

Fluxo de Dados

centenas de bots

muitos fluxos de integração usando a DPL

Bilhões de dados sendo gerados!!!

Fluxo de Dados ✅

Mas e agora como **habilitamos os times a trabalhar** com estes dados?

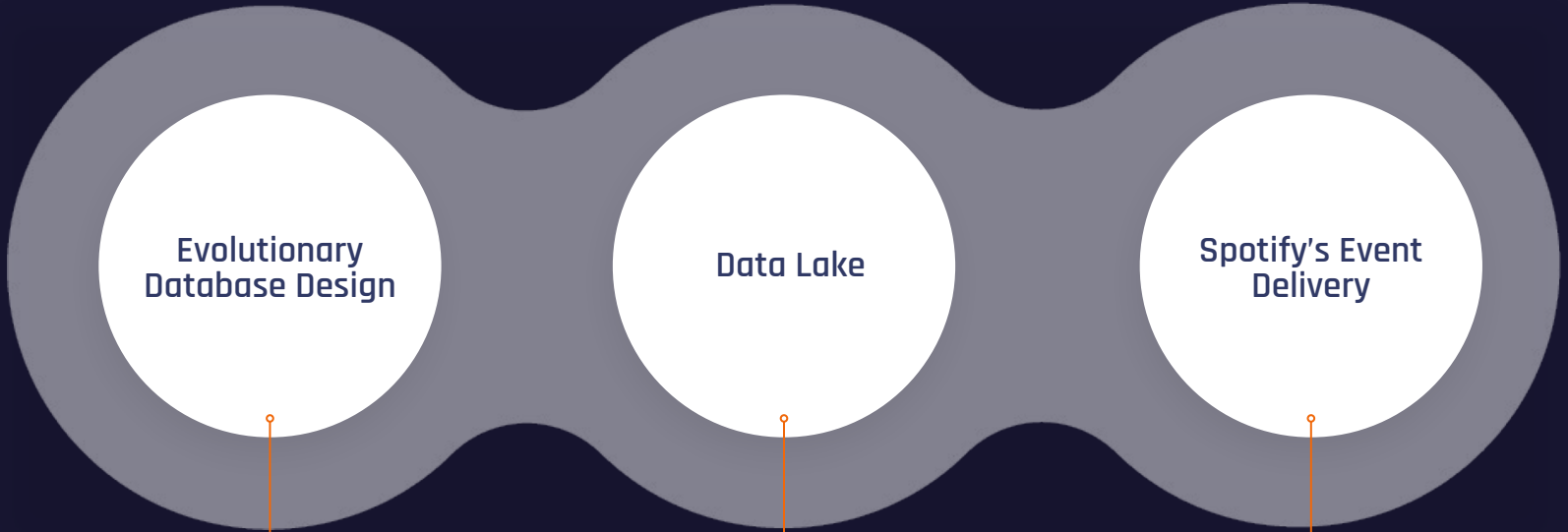
Você possui algum destes problemas?

Event Stream e Data Lake

- 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?

Como resolver esses problemas?

Event Stream e Data Lake



Martin Fowler e Pramod Sadalage (link)

Martin Fowler (link)

The Road to the Cloud (link)

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, ...

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, ...

Data Lake

Repositório único de dados de toda a empresa;

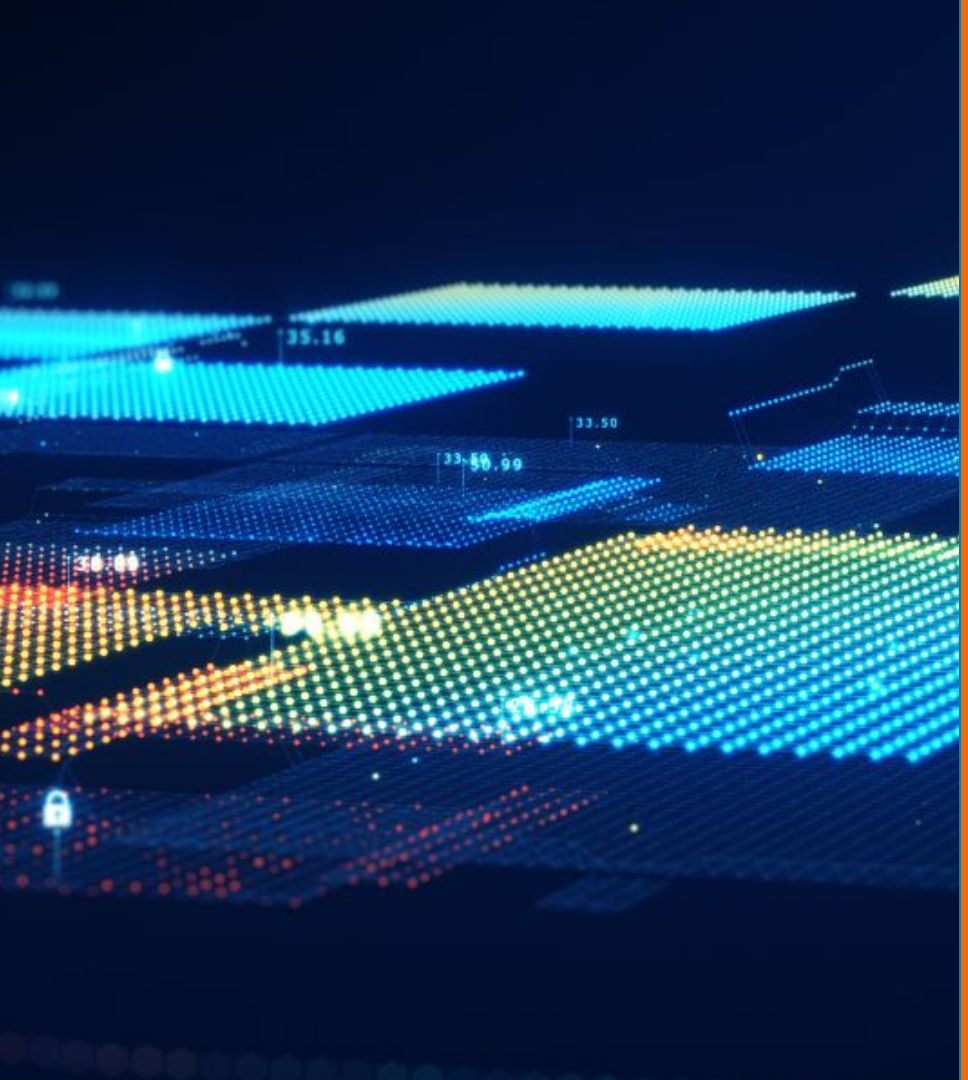
Facilitar o acesso dos times aos dados;

Possibilidade de acesso aos dados históricos;

Consolida e organizar todos os dados da empresa;

Menor custo de armazenamento em blob storage;

Exemplo:
GCS, S3, ...



Desenvolver algo ou usar alguma ferramenta pronta?

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

Bora construir?

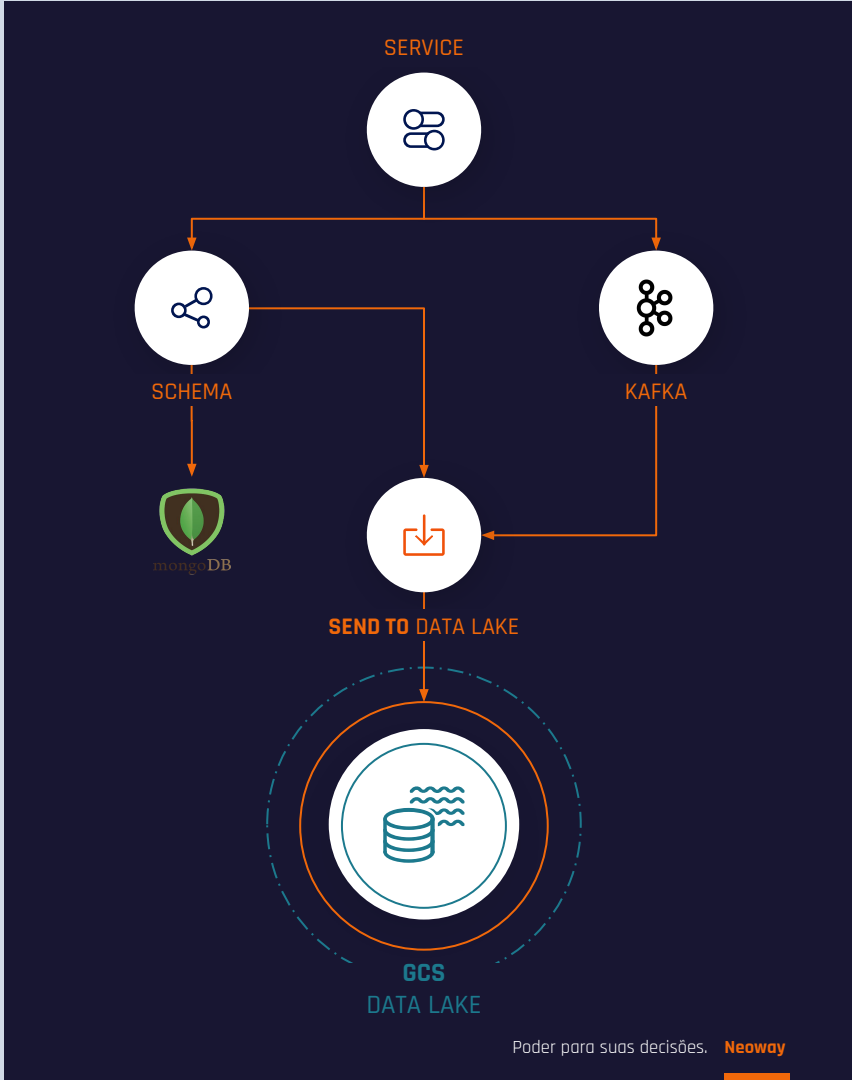
Event Stream e Data Lake

Gestor de schemas: Schema

Event Stream: Kafka

Send to Data Lake: Flow

Data Lake: Google Cloud Storage

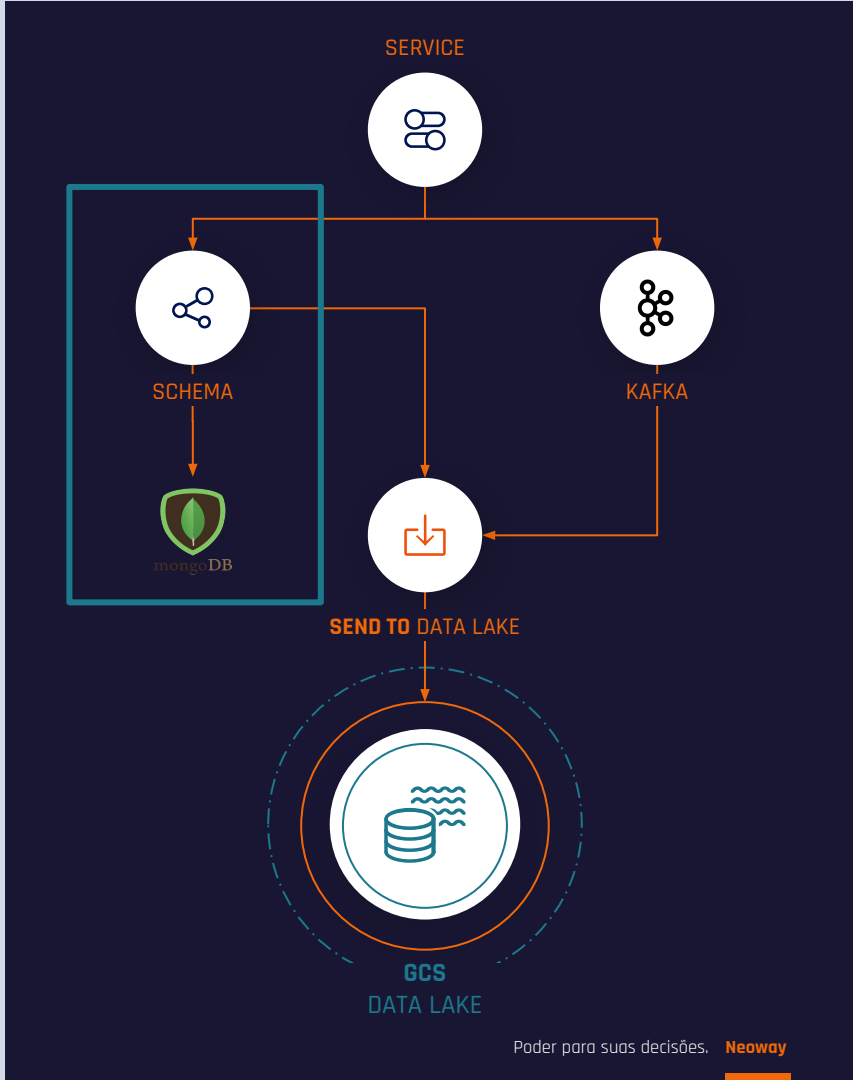


Schema Manager

Schema Manager

Event Stream e Data Lake

- **JSONSchema:** nosso pipeline já usava JSON
- **Imutabilidade**
- **Backward** compatibility: O dado deve evoluir de forma a não quebrar os consumidores;
  - **Você não pode:** Remover atributos; Mudar o tipo dos atributos;



Lib JSONSchema

Event Stream e Data Lake

README.md

godoc reference | build error | go report A+

gojsonschema

Description

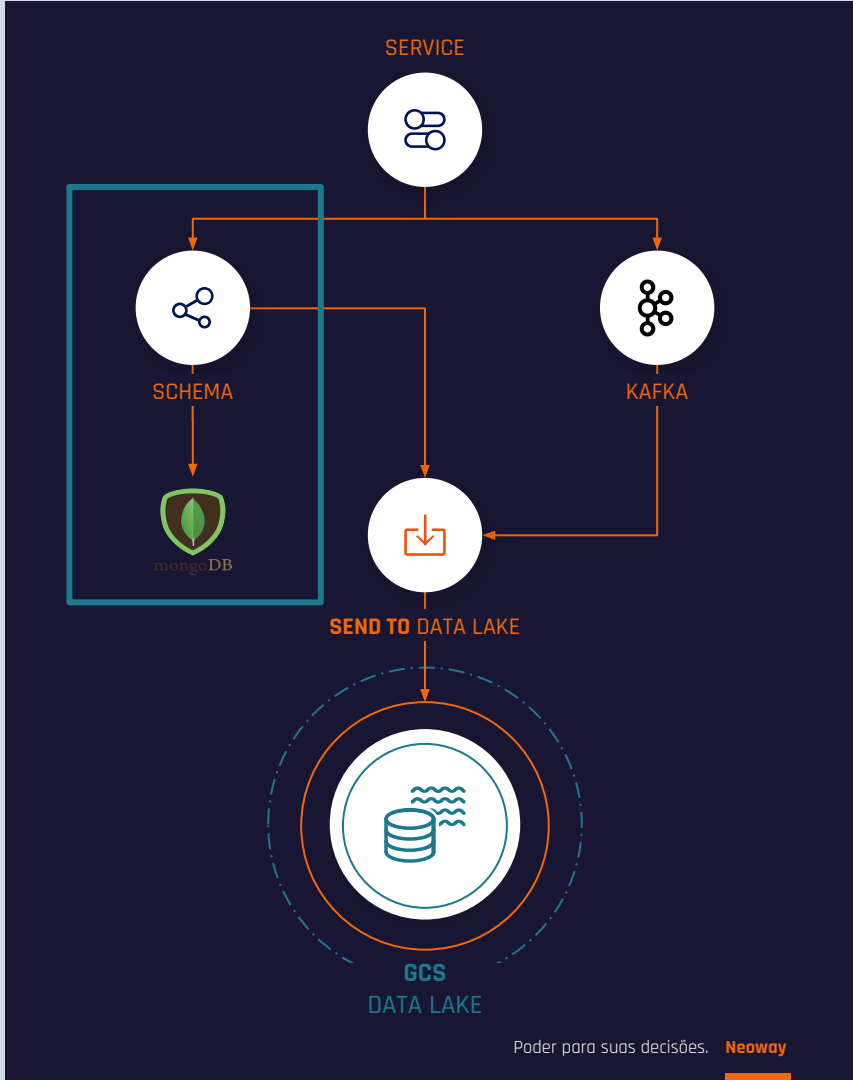
An implementation of JSON Schema for the Go programming language. Supports draft-04, draft-06 and draft-07.

References :

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

Installation

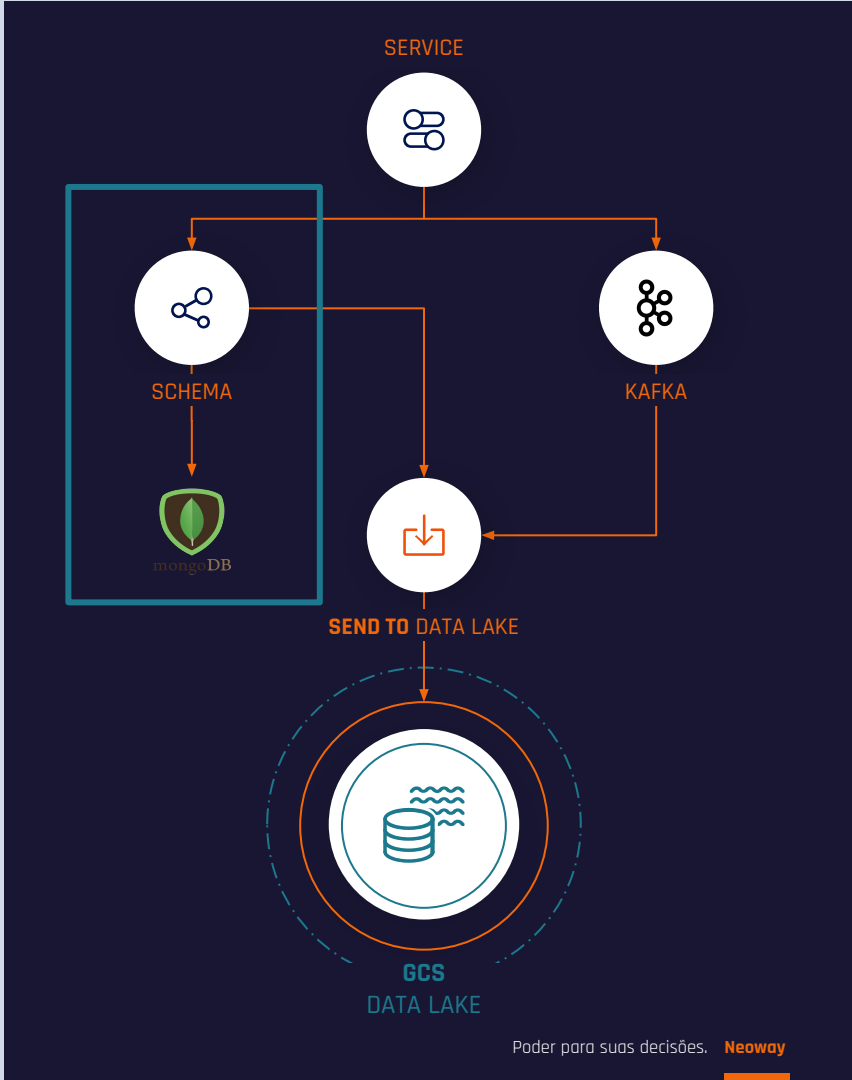
```
go get github.com/xeipuuv/gojsonschema
```



```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
    }
```

Validate JSONSchema

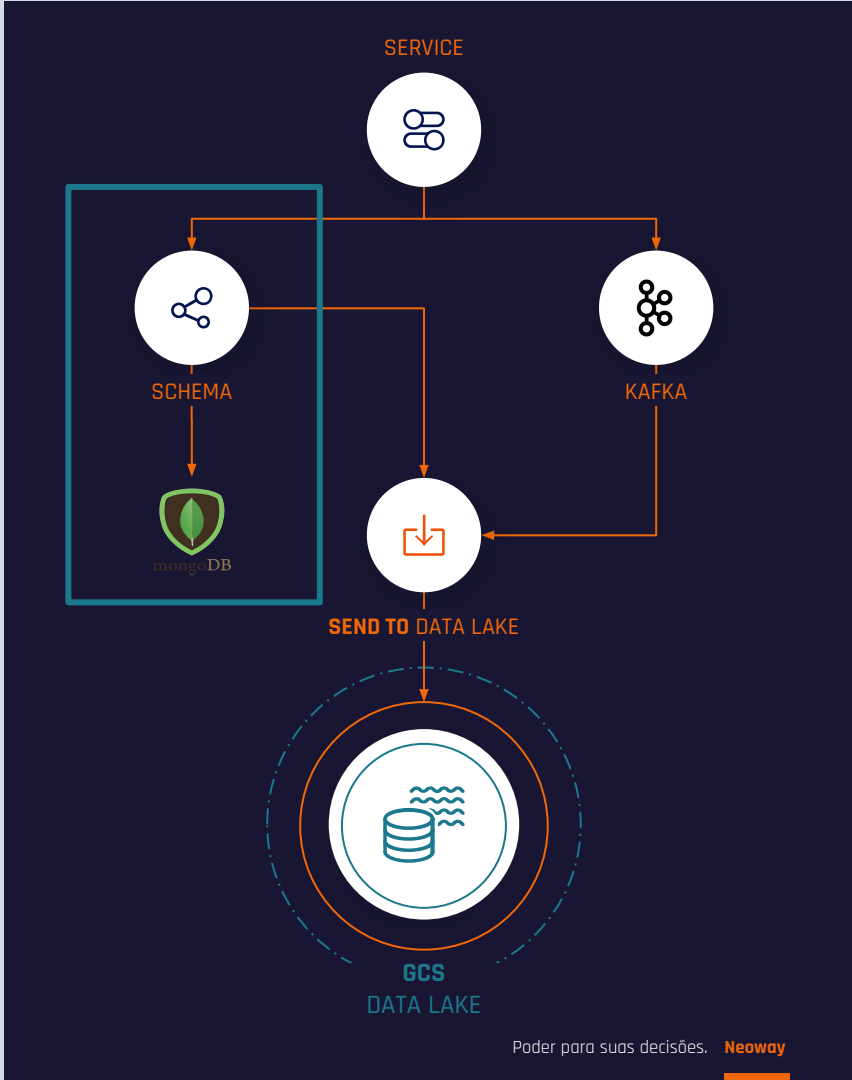
Event Stream e Data Lake



Enviar o schema

Event Stream e Data Lake

- Definir o schema do dado usando JsonSchema
  type: output
  provider: neoway
  name: rf-pj
- Registrar o schema usando a API
- Junto destes schemas adicionamos alguns metadados para governança destes dados



Enviar o schema

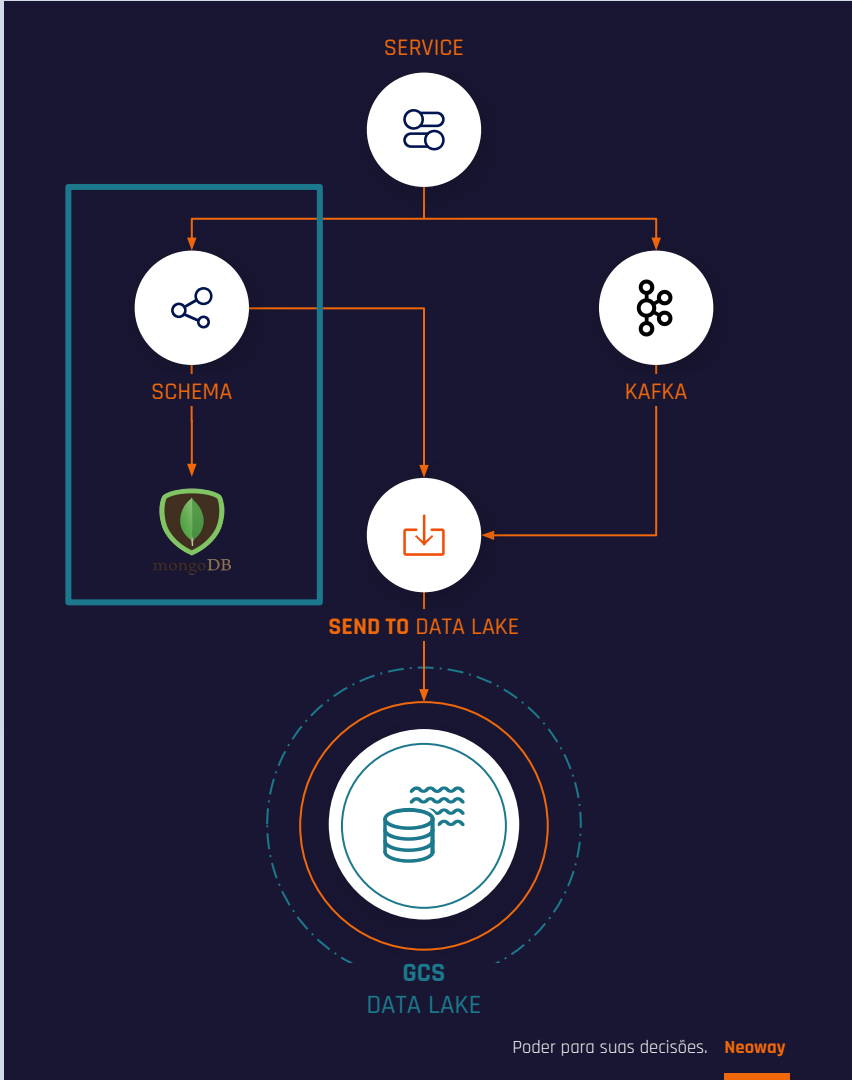
Event Stream e Data Lake

Curl example:

```sh
curl -X PUT \
  https://api.neoway.com.br/providers/partner/schemas/source-test/types/source \
  -H "Authorization: Bearer <YOUR_BEARER_TOKEN>" \
  -H "Content-Type: application/json" \
  -d '{"title": "json-schema-spec","type": "object","required": ["data","metadata"] "properties": {"data": {"type": "string"},"met
```

Full response:

```json
{
  "id":"1"
}
```

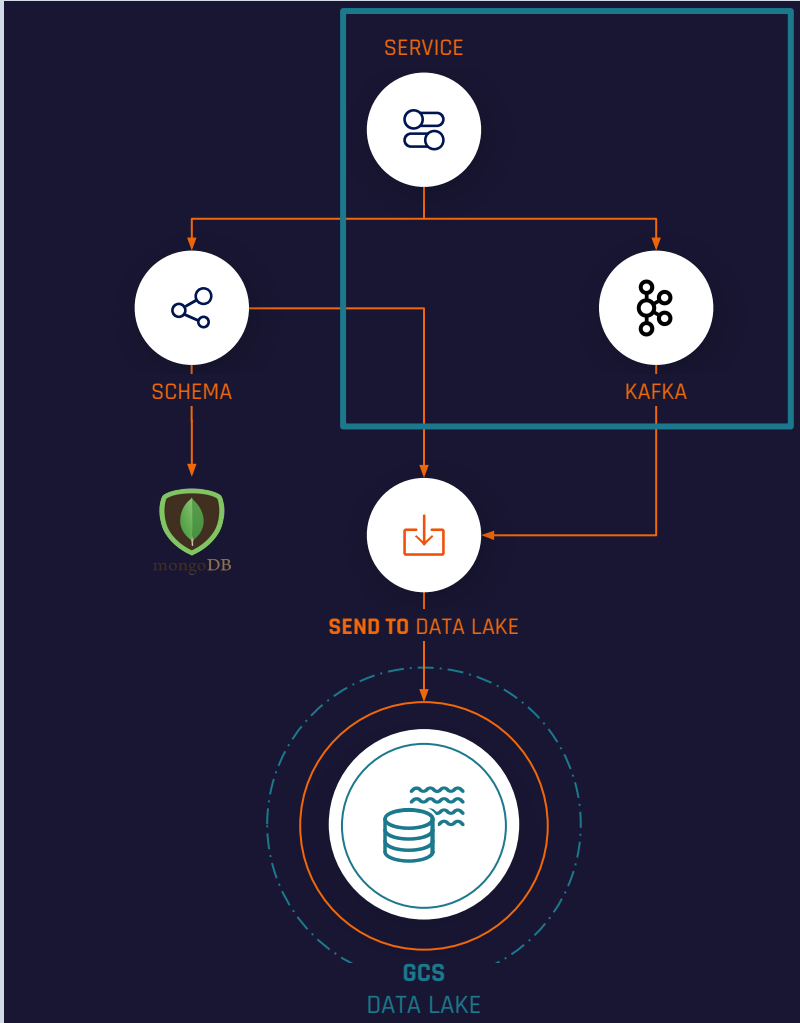



Event **Stream**

Enviar o dado para o Kafka

Event Stream e Data Lake

- Pegar o schema ID registrado;
- Enviar o dado para o Kafka;
- O nome do tópico é o nome do schema:
  Exemplo: **output-neoway-rf-pj**



Lib Kafka

Event Stream e Data Lake

sarama

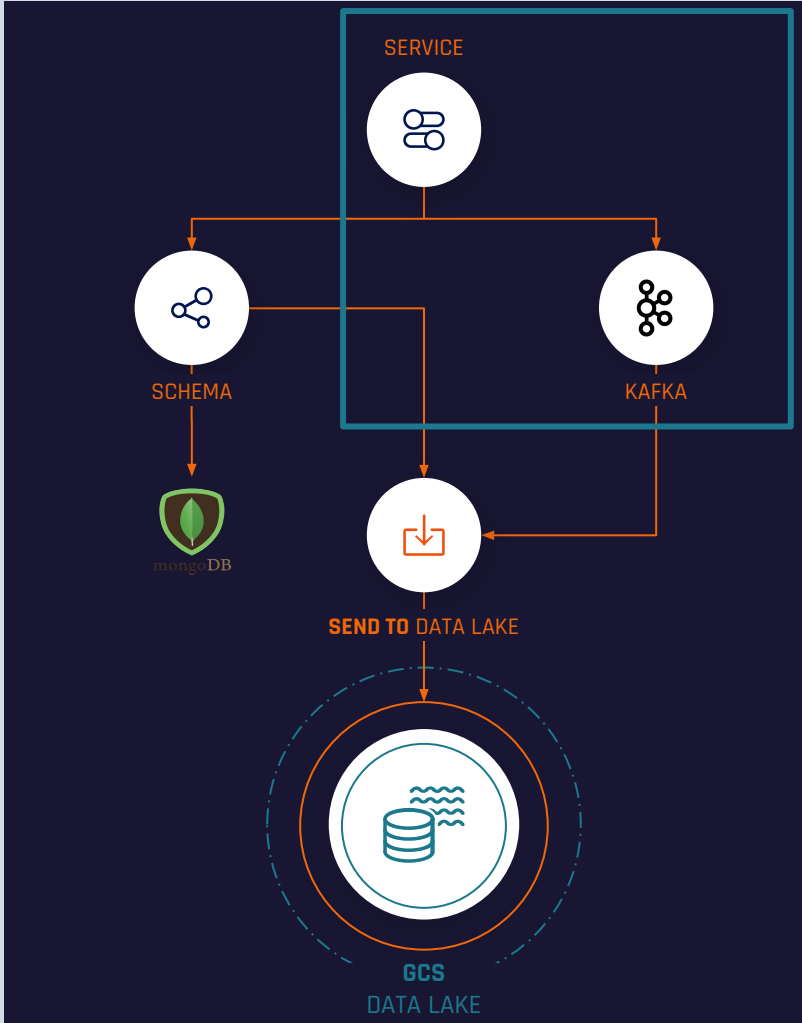
reference | build passing | codecov unknown

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.

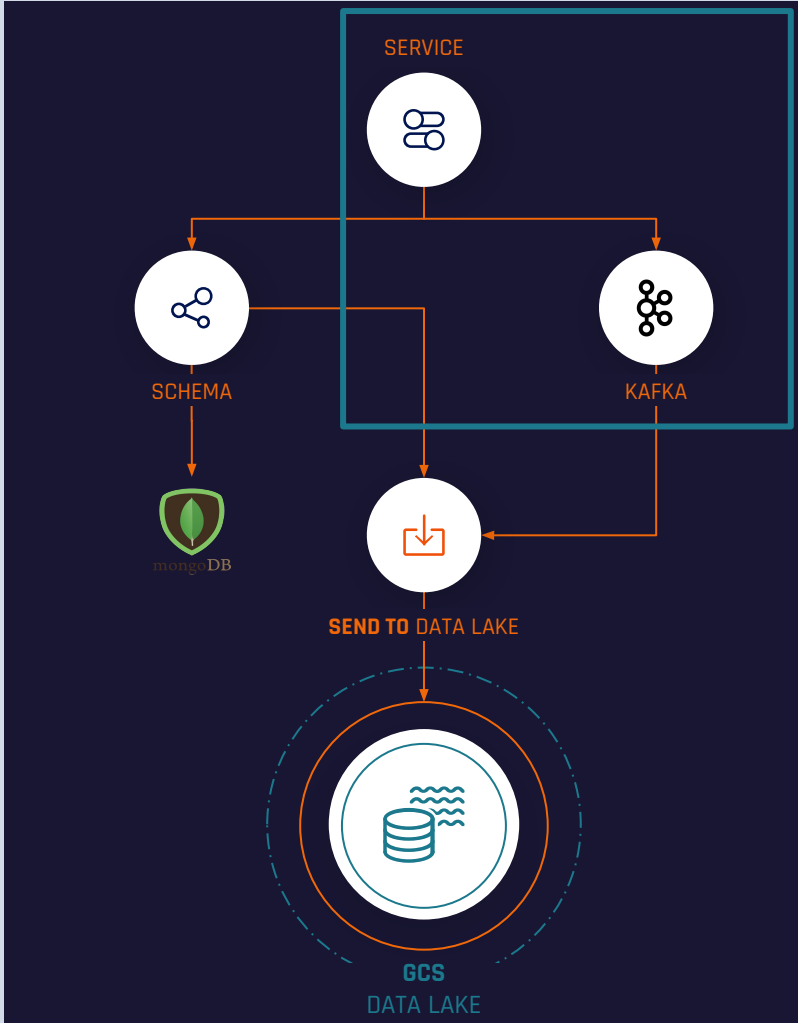
You might also want to look at the Frequently Asked Questions.



```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)
```

Lib Kafka

Event Stream e Data Lake

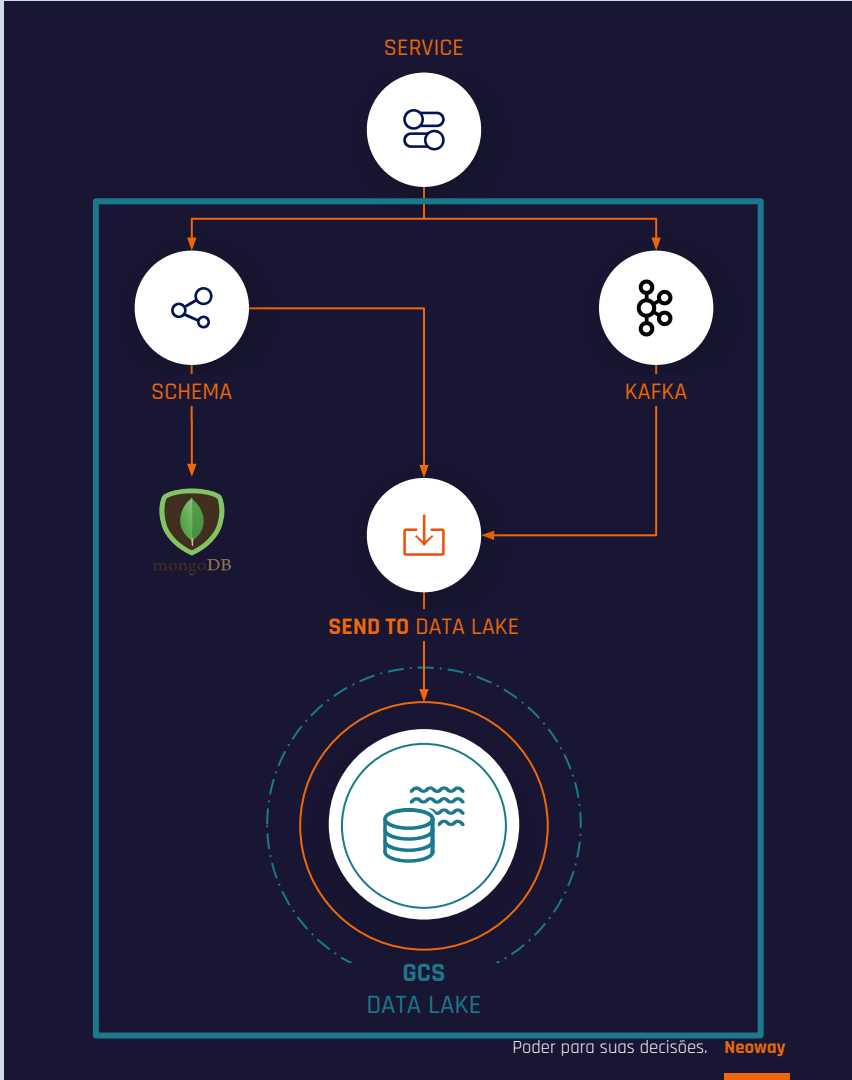


Data Lake

Flow

Event Stream e Data Lake

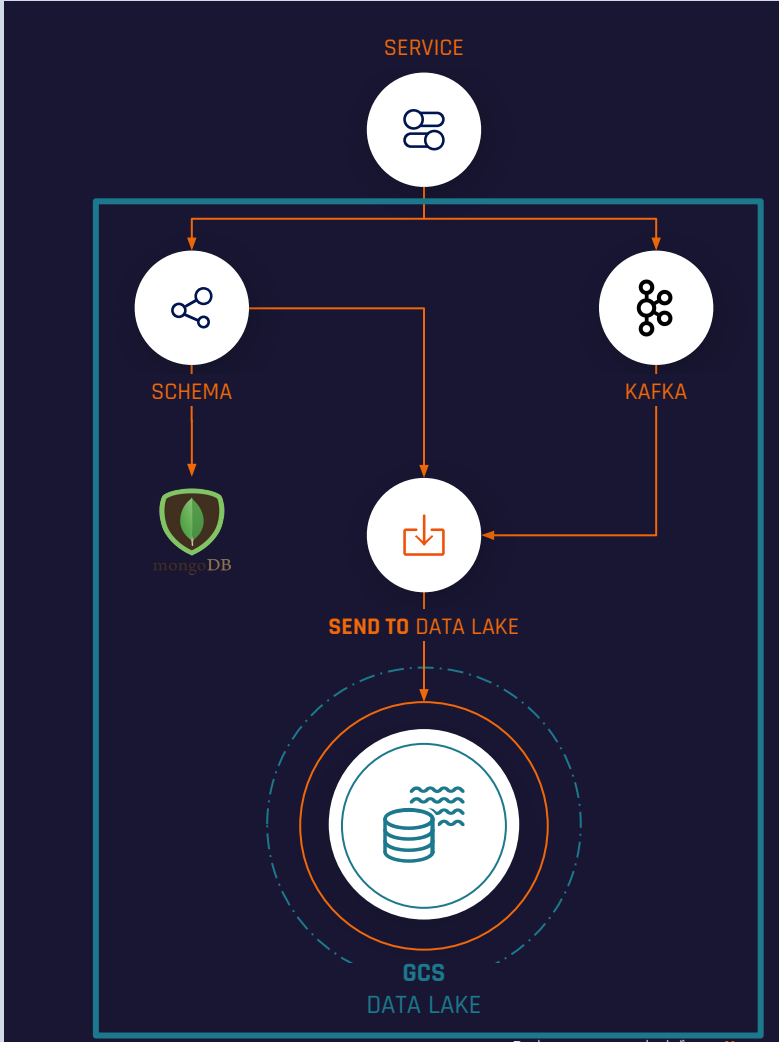
- Lê todos os schemas registrados
- A partir do nome ele descobre o tópico
- Começa a ler todos os dados de cada tópico
- Consolida um arquivo com os dados recebidos
- Salva no Data Lake



Lendo dados

Event Stream e Data Lake

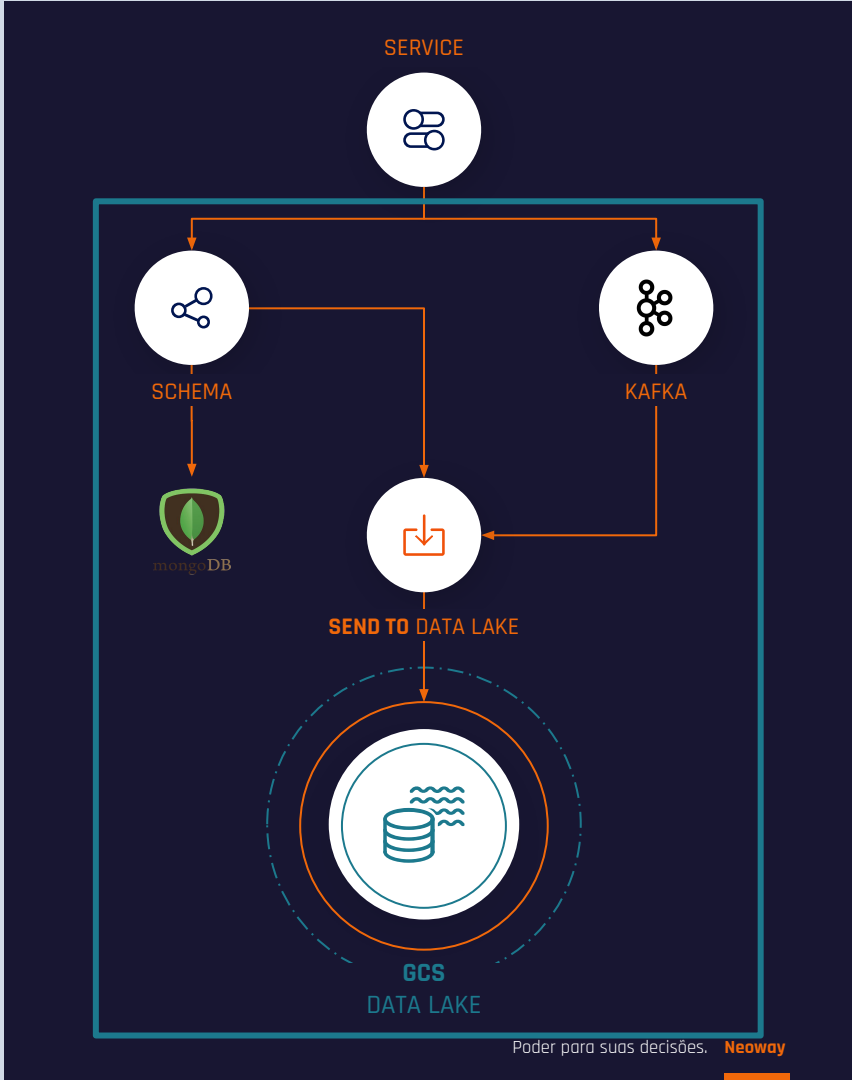
```go
import (
    ...
    "github.com/Shopify/sarama"
)

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
}
```



```go
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)
```

Lendo dados

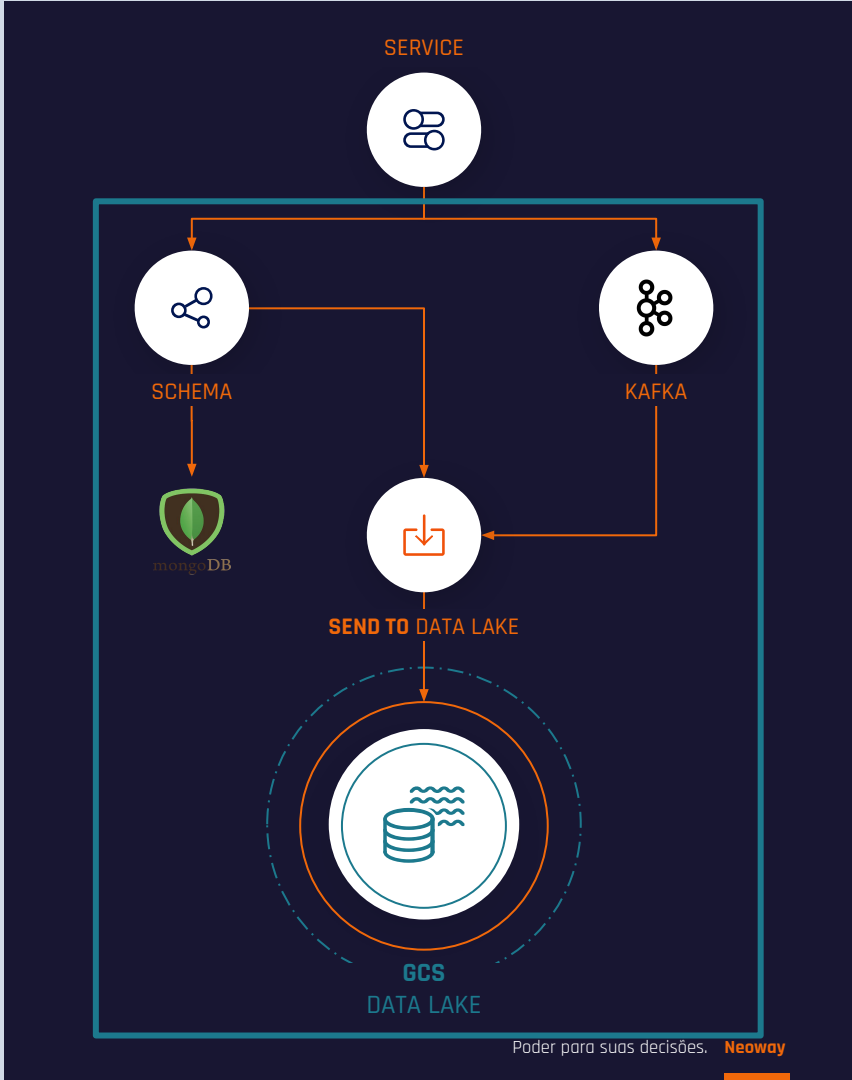
Event Stream e Data Lake



```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!")
    }
}
```

Validando dado e schema

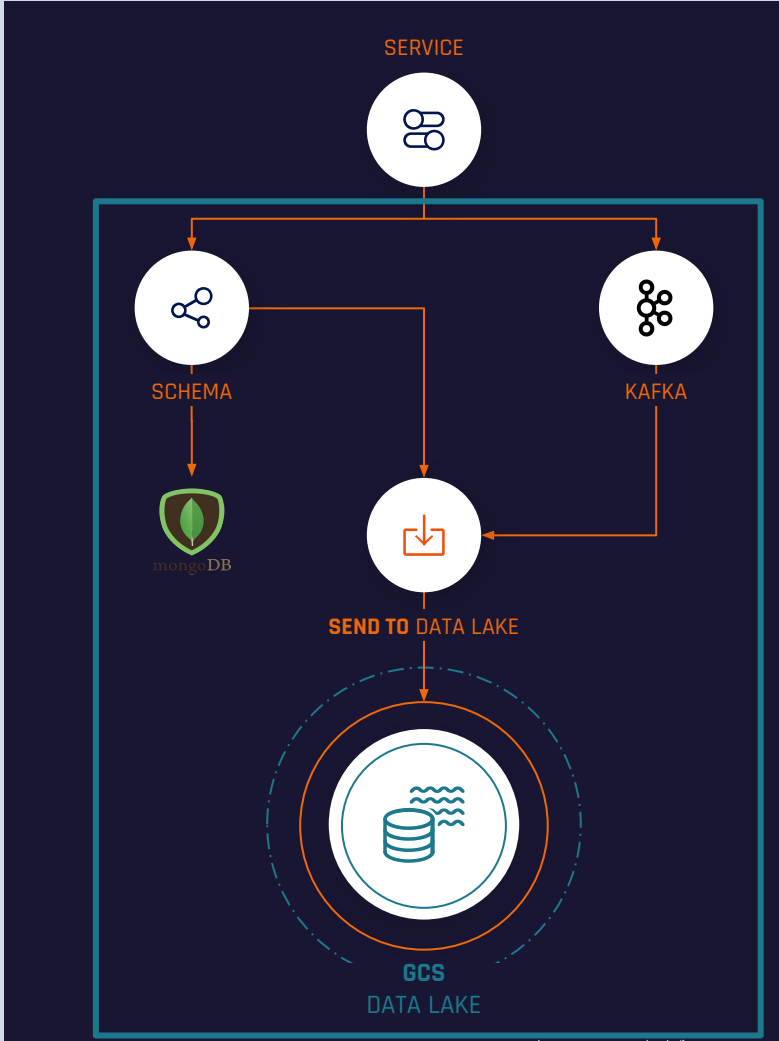
Event Stream e Data Lake



Enviando dados

Event Stream e Data Lake

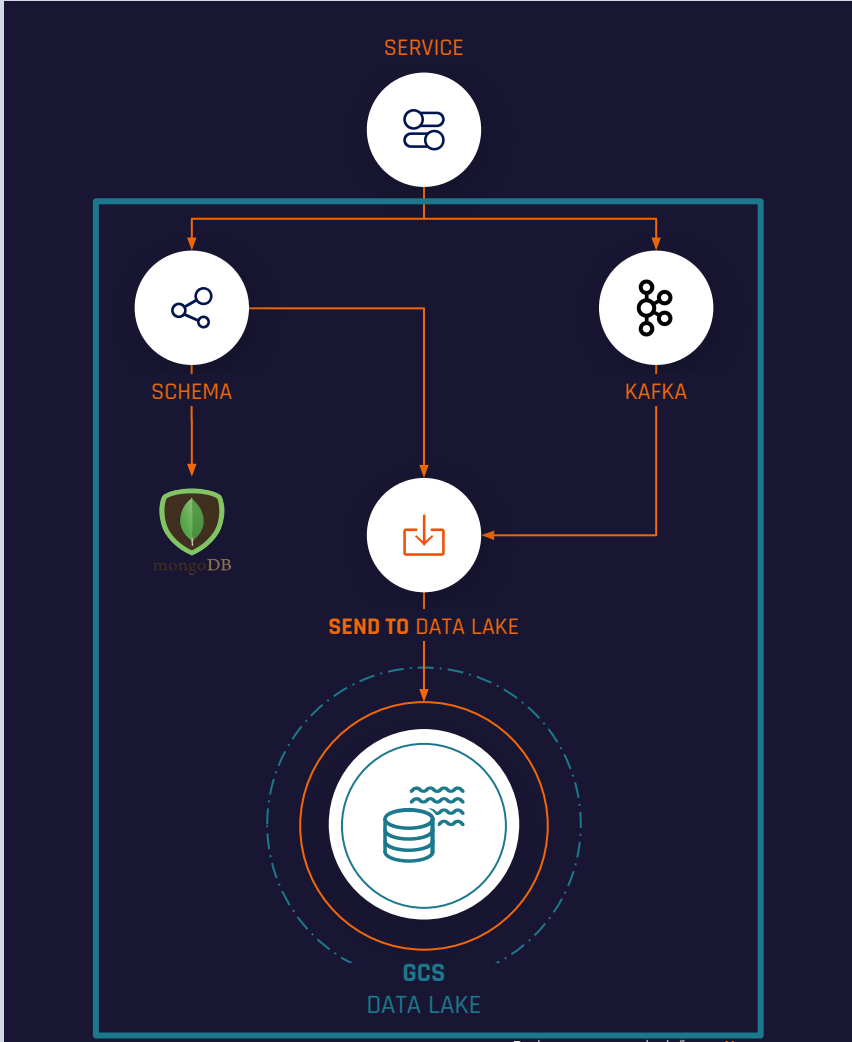
```
    go func(configName string, metadataChan chan interface{}) {
        for meta := range metadataChan {
            // commit offset kafka
        }
    }(configName, metadataChan)
...
    for {
        select {
        case <-ticker.C:
            if msgsBuffer.Len() > 0 {
                // save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



Enviando dados

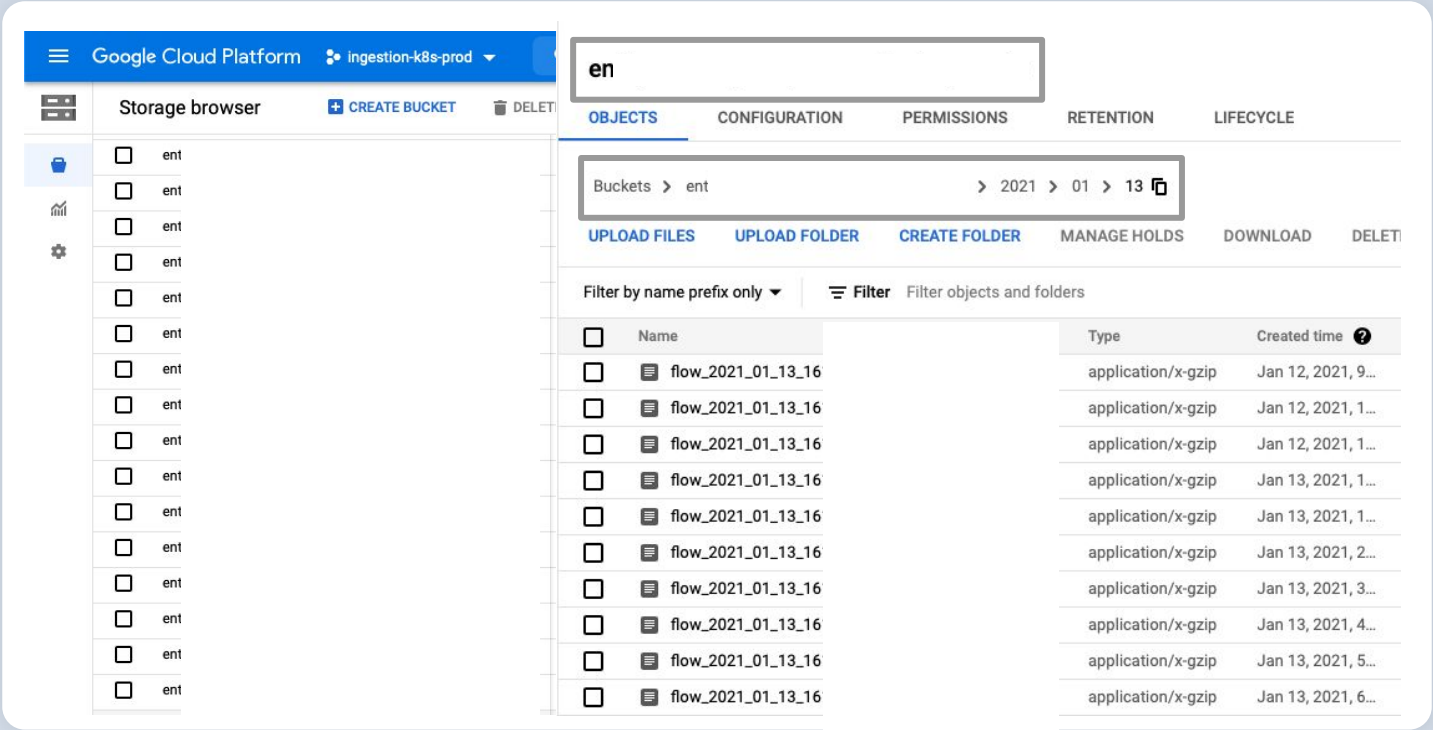
Event Stream e Data Lake

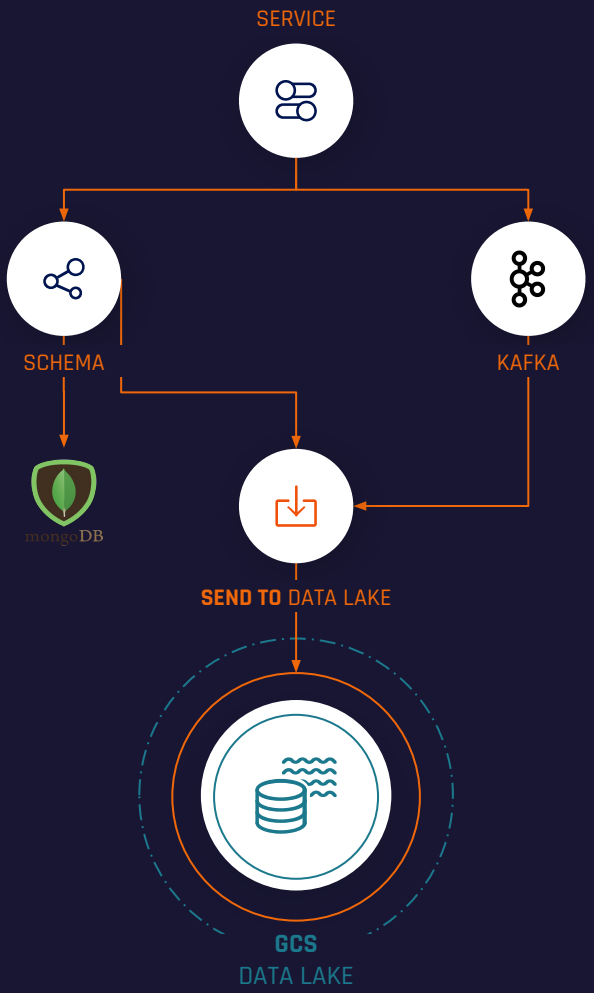
```go
import (
    ...
    gcpstorage "cloud.google.com/go/storage"
)

    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
    }
```



Flow

Event Stream e Data Lake



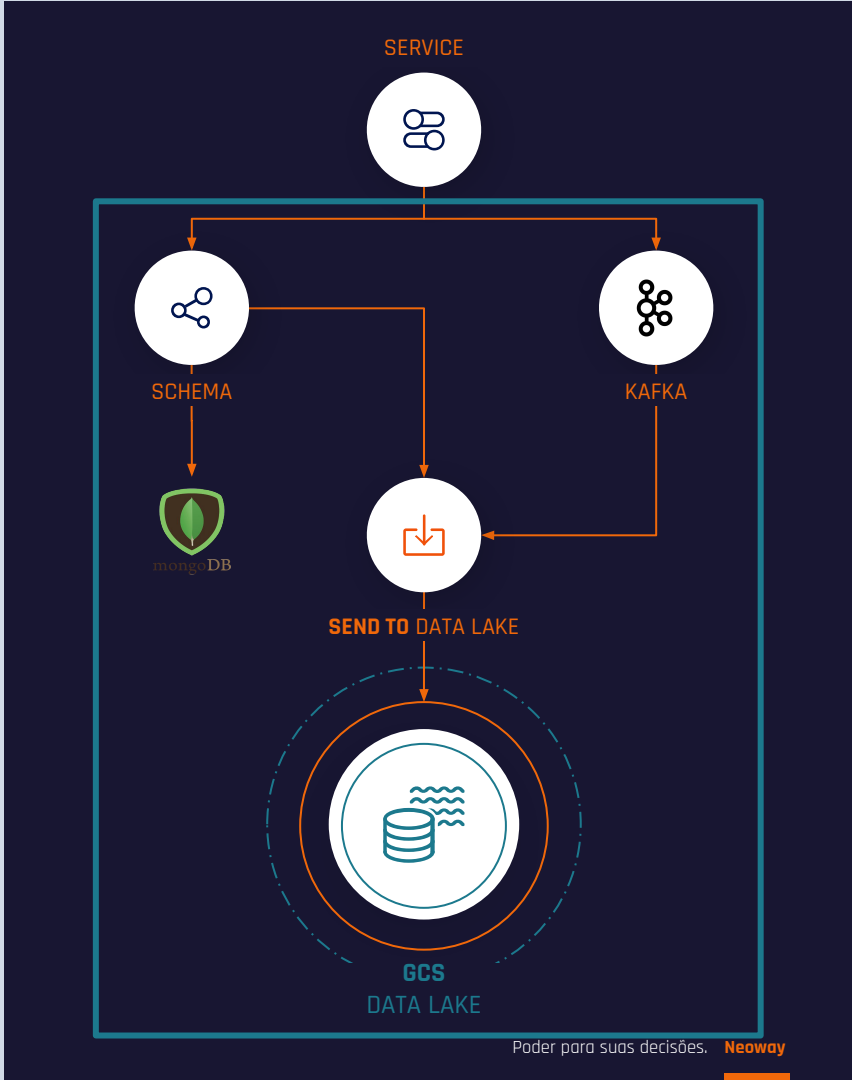

SERVICE
SCHEMA
KAFKA
mongoDB
SEND TO DATA LAKE
GCS
DATA LAKE

Como **esta arquitetura** tem nos ajudado?

Como esta arquitetura tem nos ajudado?

Event Stream e Data Lake

- Acessibilidade e entendimento aos dados; Governança/LGPD;
- Facilitar a criação de processamentos em stream/batch;
- Criação de novas engines;




Para **finalizar**

Conheça o

Neoway LABS

TECH

Inscreva-se

No nosso canal

neowaylabs.github.io

medium.com/neowaylabs

Venha conhecer a Neoway e pegar seu copo para o Happy Hour no nosso stand!!!

Matheus Vill

linkedin.com/in/matheusvill

jobs.kenoby.com/neoway/