UFC

DEPARTAMENTO DE COMPUTAÇÃO

MDCC

GREat

Protocol Buffers

CK0154 - SISTEMAS DISTRIBUIDOS
TI0151 - SISTEMAS DISTRIBUÍDOS
CKP7500 - SISTEMAS DISTRIBUIDOS E REDES DE COMUNICACAO

Prof. Dr. Paulo Antonio Leal Rego

paulo@dc.ufc.br

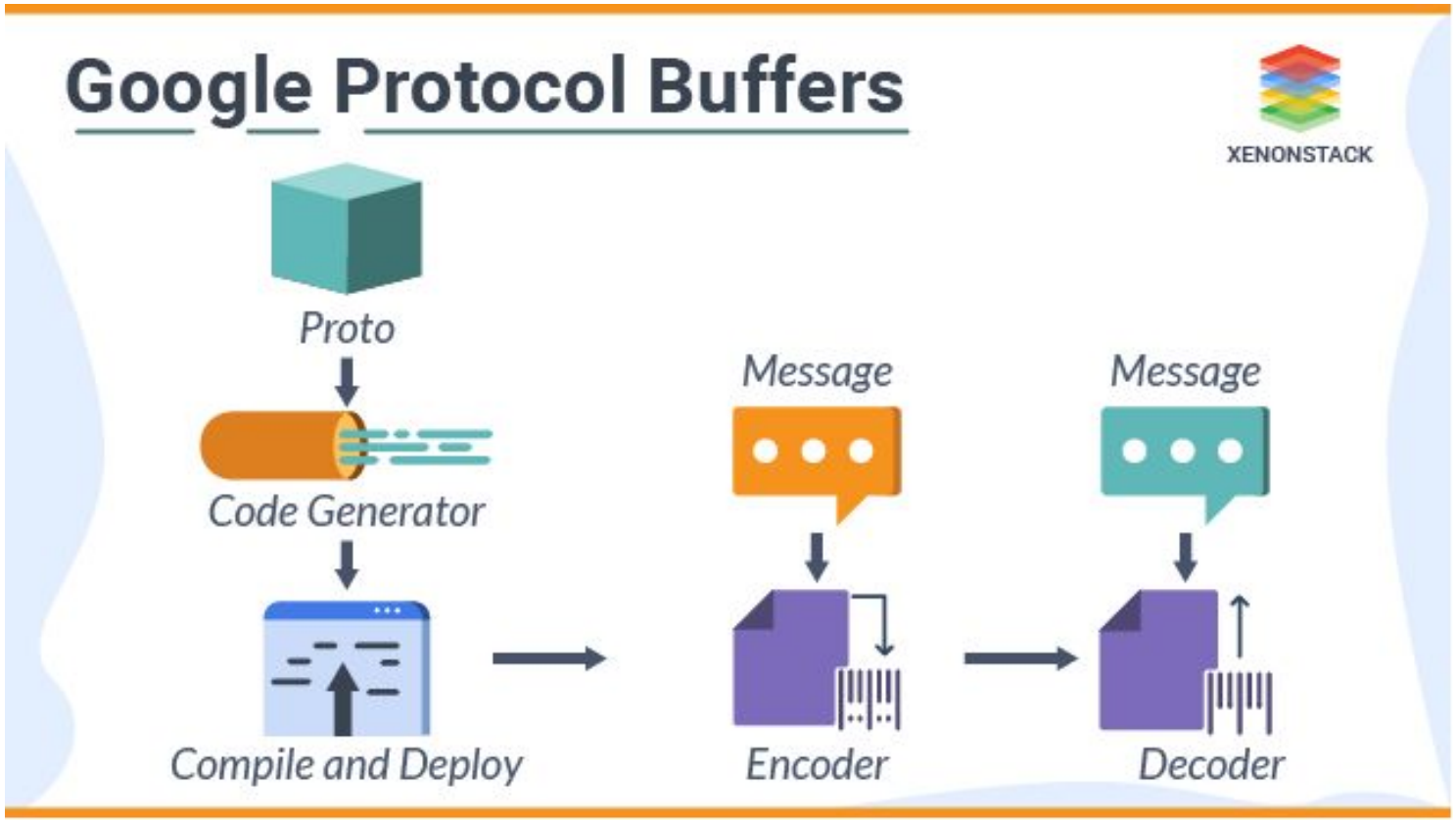
Protocol Buffers

- O Protocol Buffers (também conhecido como protobuf) é um método de serialização de dados estruturados.
- É útil no desenvolvimento de programas que se comunicam uns com os outros ou para armazenar dados.
- O método envolve uma linguagem de descrição de interface que descreve a estrutura de dados e um programa que gera código-fonte a partir dessa descrição para a geração ou a análise de um fluxo de bytes que representa os dados estruturados.

Por que usar protobuf?

- Suporta diversas linguagens de programação
  - Nativamente: C++, Java, Python, Go, Ruby, Objective-C, C# e JavaScript.
  - Implementações de terceiros: Perl, PHP, R, Scala e Swift.
- Independente de plataforma
- Compatibilidade com versões anteriores
  - Atualmente está na versão 3
- Marshalling/Unmarshalling eficiente
- Parsing com bom desempenho
- Fortemente tipado
- Geração automática de código

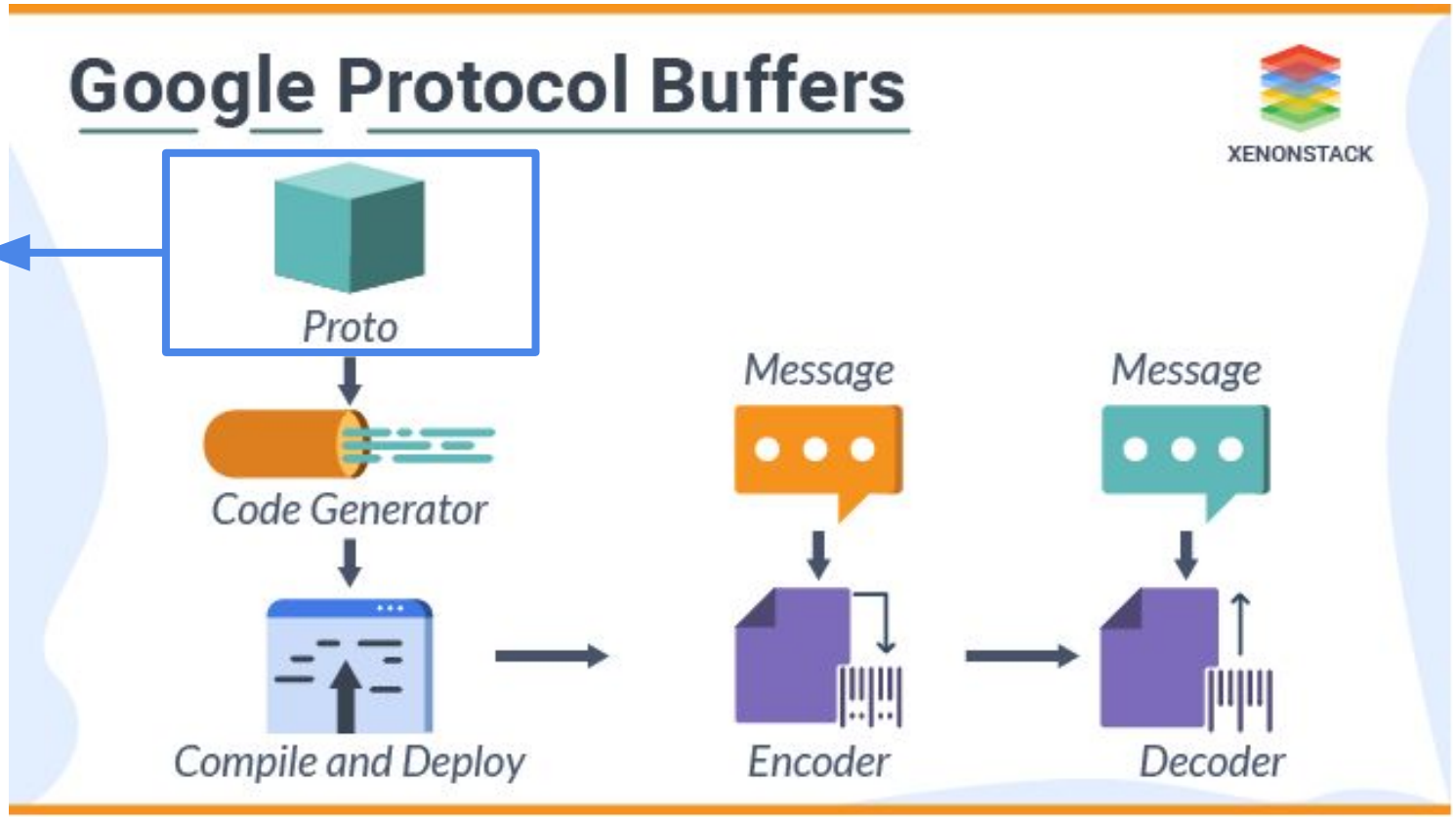
Como funciona?



Fonte: Xenonstack

Como funciona?

```proto
syntax = "proto3";
package teste;
// [END declaration]

// [START java_declaration]
option java_package = "teste";
option java_outer_classname = "PessoaPB";
// [END java_declaration]

// [START messages]
message Pessoa {
  string name = 1;
  int32 id = 2;  // Unique ID number for this person.
  string email = 3;
  string fone = 4;
}
```

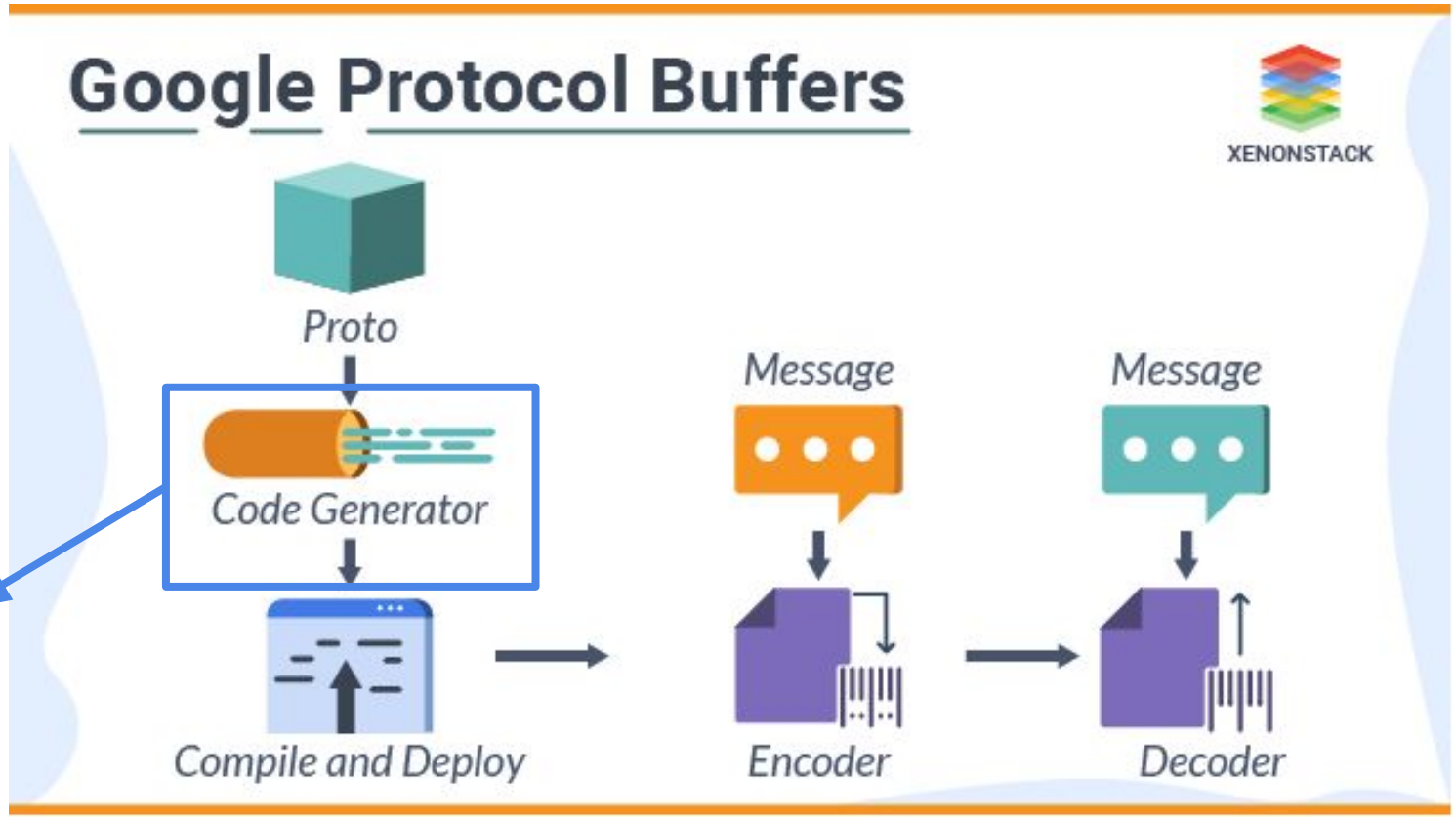


Fonte: Xenonstack

Como funciona?

```proto
syntax = "proto3";
package teste;
// [END declaration]

// [START java_declaration]
option java_package = "teste";
option java_outer_classname = "PessoaPB";
// [END java_declaration]

// [START messages]
message Pessoa {
  string name = 1;
  int32 id = 2;  // Unique ID number for this person.
  string email = 3;
  string fone = 4;
}
```

```
protoc --python_out=. -I. pessoa.proto
protoc --java_out=. -I. pessoa.proto
```



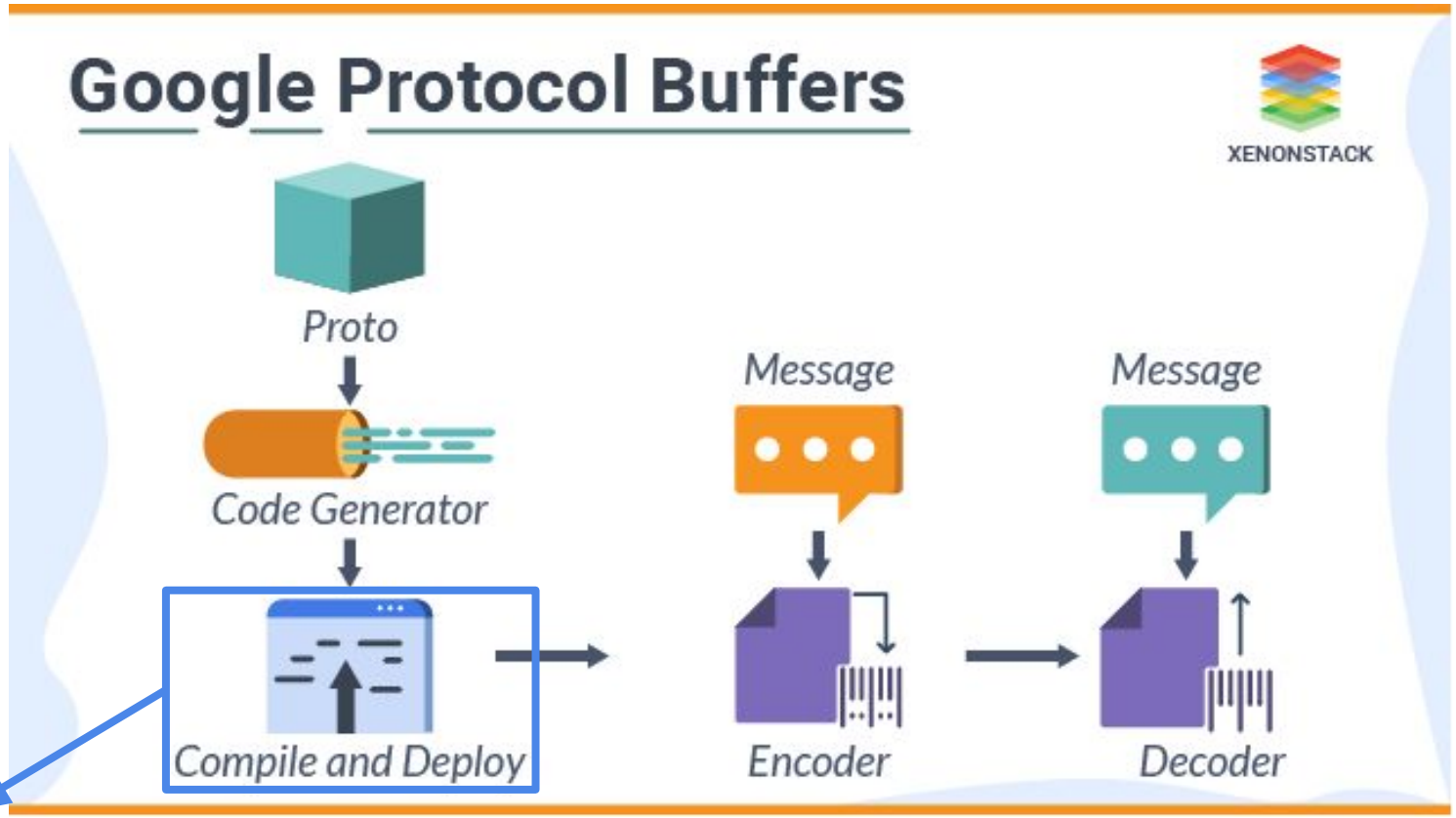
Fonte: Xenonstack

DEPARTAMENTO DE COMPUTAÇÃO

Como funciona?

```proto
syntax = "proto3";
package teste;
// [END declaration]

// [START java_declaration]
option java_package = "teste";
option java_outer_classname = "PessoaPB";
// [END java_declaration]

// [START messages]
message Pessoa {
  string name = 1;
  int32 id = 2;  // Unique ID number for this person.
  string email = 3;
  string fone = 4;
}
```

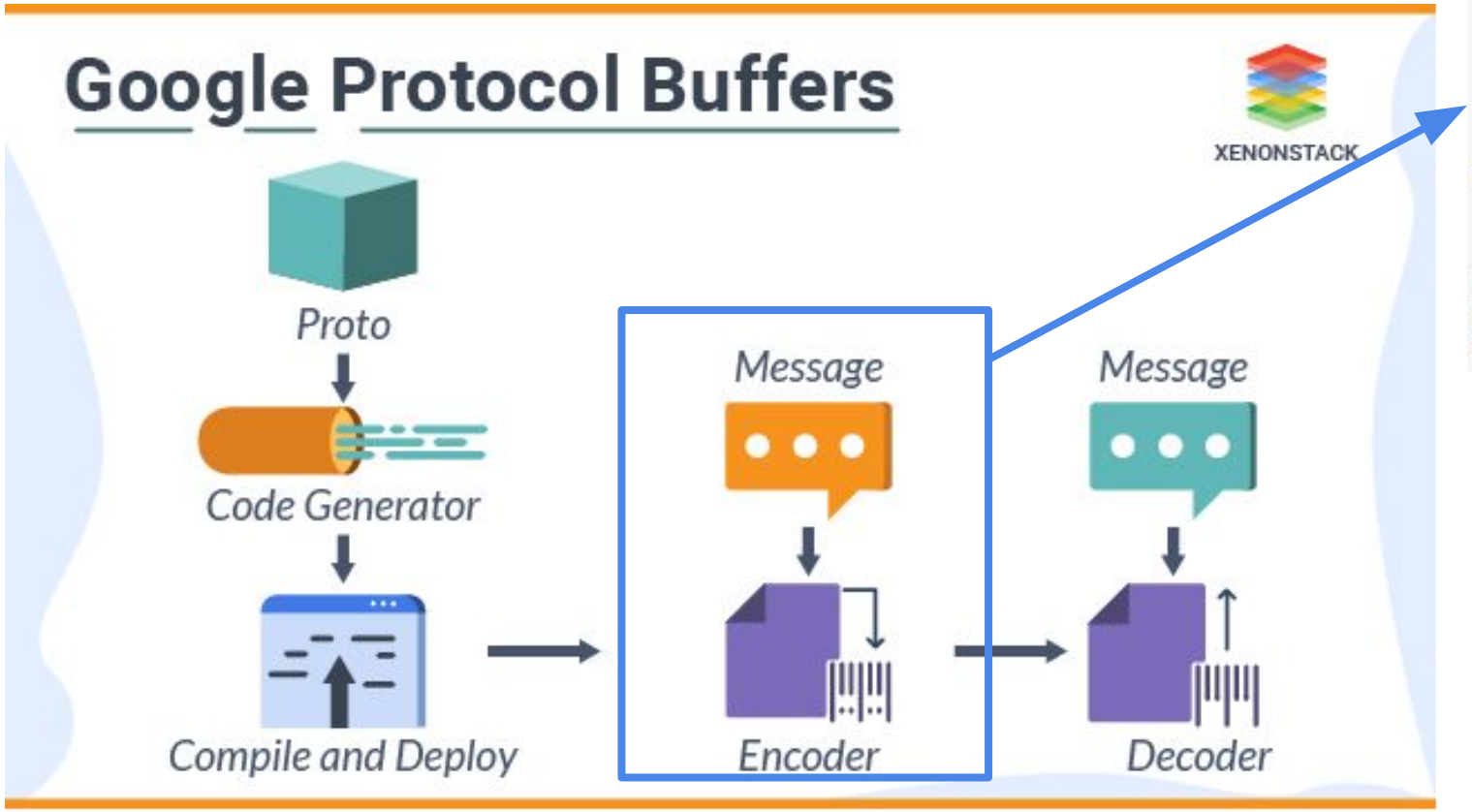
```
protoc --python_out=. -I. pessoa.proto
protoc --java_out=. -I. pessoa.proto
```



Fonte: Xenonstack

DEPARTAMENTO DE COMPUTAÇÃO

Como funciona?

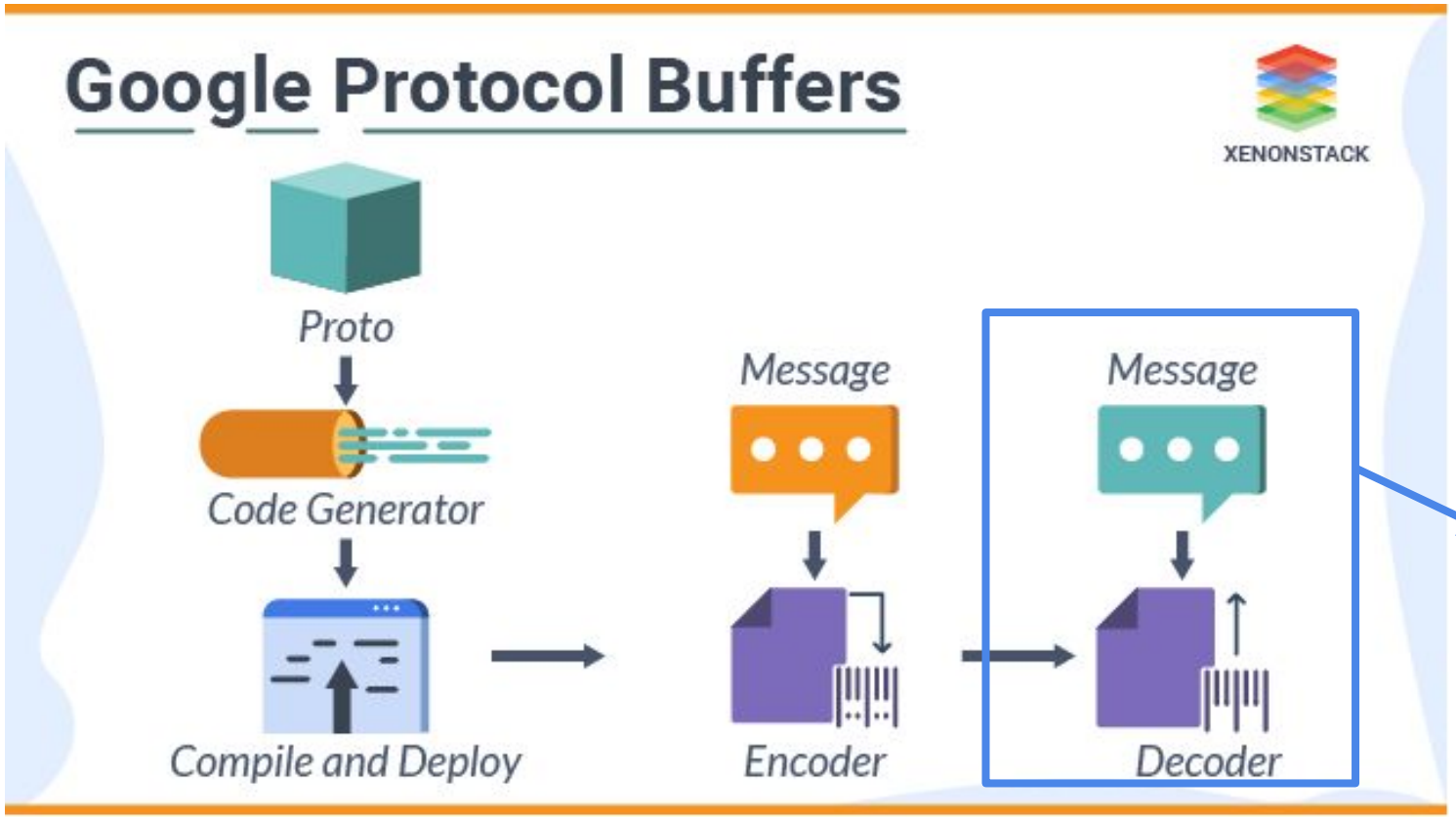



```
clientSocket = new Socket("localhost", 6789);

Pessoa person = Pessoa.newBuilder()
        .setId(1)
        .setName("Paulo")
        .setEmail("pauloalr@ufc.br")
        .setFone("999888777")
        .build();

CodedOutputStream out = CodedOutputStream.
        newInstance(clientSocket.getOutputStream());

person.writeTo(out);
out.flush();
clientSocket.close();
```

Fonte: Xenonstack

DEPARTAMENTO DE COMPUTAÇÃO

Como funciona?



Fonte: Xenonstack

```
clientSocket = new Socket("localhost", 6789);

Pessoa person = Pessoa.newBuilder()
        .setId(1)
        .setName("Paulo")
        .setEmail("pauloalr@ufc.br")
        .setFone("999888777")
        .build();

CodedOutputStream out = CodedOutputStream.
        newInstance(clientSocket.getOutputStream());

person.writeTo(out);
out.flush();
clientSocket.close();
```

```
listenSocket = new ServerSocket(6789);
while (true) {
    Socket cs = listenSocket.accept();
    CodedInputStream in = CodedInputStream.
            newInstance(cs.getInputStream());

    Pessoa p = Pessoa.parseFrom(in);
    System.out.println(p.toString());
```

Definição da mensagem protobuf e tipos suportados

- Sintaxe simples para definição das mensagens
- Campos da mensagem devem ser identificados com um índice numérico
- Cada campo tem um nome, tipo e descritor (e.g., se o campo é opcional ou obrigatório)
- Mensagens podem importar ou herdar outras mensagens
- Tipos suportados: int32, int64, float, string, bool, byte, enum, etc...
- Definição de array/lista com repeated
- Um campo pode ter um valor padrão
- Mais: https://developers.google.com/protocol-buffers/docs/proto3

```
message Person {
  required string name = 1;
  required int32 id = 2;
  optional string email = 3;

  enum PhoneType {
    MOBILE = 0;
    HOME = 1;
    WORK = 2;
  }

  message PhoneNumber {
    required string number = 1;
    optional PhoneType type = 2;
  }
  repeated PhoneNumber phone = 4;
```

Instalação

- https://github.com/protocolbuffers/protobuf/releases