



CLOUD NATIVE  
COMMUNITY GROUPS

# Améliorer la vitesse des échanges entre vos services avec gRPC

15 février 2024

**CNCF Lorient**  
**Jérôme Barotin**  
**@jbarotin**

# Once upon a time...

C'est l'histoire d'une startup qui a réussi à vendre son concept.



# Once upon a time...

C'est l'histoire d'une startup qui a réussi à vendre son concept.

Au centre de l'application, on y trouve un petit service codé en Java, accessible via une API HTTP REST/JSON.

# Once upon a time...

C'est l'histoire d'une startup qui a réussi à vendre son concept.

Au centre de l'application, on y trouve un petit service codé en Java, accessible via une API HTTP REST/JSON.

Le temps passe, et les clients et les fonctions augmentent. Ce petit service se retrouve de plus en plus sollicité... trop sollicité.

# Comment tenir la charge ?



AUGMENTER LE  
NOMBRE DE NOEUDS  
SANS RIEN  
CHANGER DANS LE CODE



TROUVER UNE  
SOLUTION PLUS  
EFFICACE ET CONSOMER  
MOINS DE RESSOURCES

# gRPC en Bref

- Inventé par Google
- Sous licence Apache 2.0 depuis 2015
- Incubation CNCF
- S'appuie complètement sur Protobuf et HTTP2
- Dispo officiellement pour les langages suivants : Go, C++, Java, Python, C#, dart, Kotlin, Node Js, Objective-C, PHP, Ruby



# Protobuf : un système d'encodage binaire

- Défini et créé par Google en 2008 (<https://protobuf.dev/>)
- Open Source sous Licence BSD
- Technos concurrentes :
  - TLV
  - ASN1
  - Apache Thrift
  - Apache Avro



# Protobuf : fonctionnement

- Ecriture d'un fichier « proto » pour définir la structure des échanges
  - Compilation via « protoc »
  - Génération des Stubs dans le langage de notre choix



## file.proto

**Java, Python, Objective-C,  
and C++** **Kotlin, Dart, Go,  
Ruby, and C#**

```
message MessageCNCF {  
    int32 green = 1;  
    string it = 2;  
    sint64 isBeautiful = 3;  
}
```

```
MessageCNCF cncf = MessageCNCF.newBuilder().setGreen(150)
    .setIt("yes")
    .setIsBeautiful(-2).build();
```

# Protobuf : encodage binaire

```
message MessageCNCF {  
    int32 green = 1;  
    string it = 2;  
    sint64 isBeautiful = 3;  
}
```



## Wire format

0x96 0x01 = 38401 ?

```
10010110 00000001      // Original inputs.  
0010110 0000001        // Drop continuation bits.  
0000001 0010110        // Convert to big-endian.  
00000010010110         // Concatenate.  
128 + 16 + 4 + 2 = 150 // Interpret as an unsigned 64-bit integer.
```

Source : <https://protobuf.dev/programming-guides/encoding/>

```
MessageCNCF cncf = MessageCNCF.newBuilder()  
    .setGreen(150)  
    .setIt("yes")  
    .setIsBeautiful(-2).build();
```



08 96 01 | 12 03 79 65 73 | 18 03

9 octets vs 42 octets pour JSON  
compact :  
« {"green": 1, "it": "yes", "isBeautiful": 2} »

## Zig zag

Signed Original	Encoded As
0	0
-1	1
1	2
-2	3

# Protobuf : les types de données

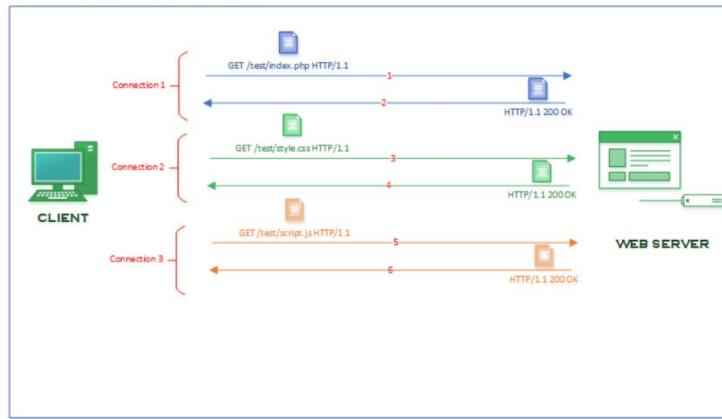
```
// définition d'un enum
enum ThisIsOption {
    optionA = 0;
    optionB = 1;
    optionC = 2;
}

/*
définition d'un message
*/
message SubStructure {
    bool firstMember = 1;
    int32 aSecondMember = 2;
    uint32 sameAsPrevious = 3;
    sint32 optimizedForNegative = 4;
    bytes aBytesSequence = 5;
    AMoreCompleteExample aMoreCompleteExample = 6;
}

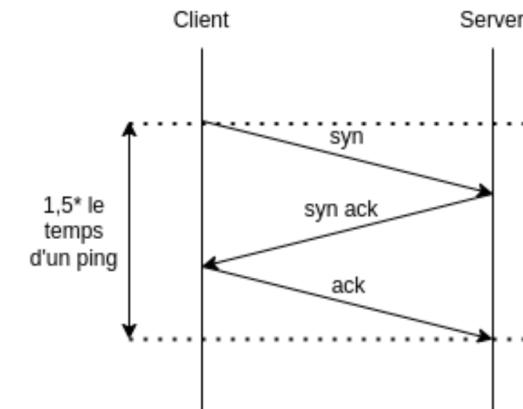
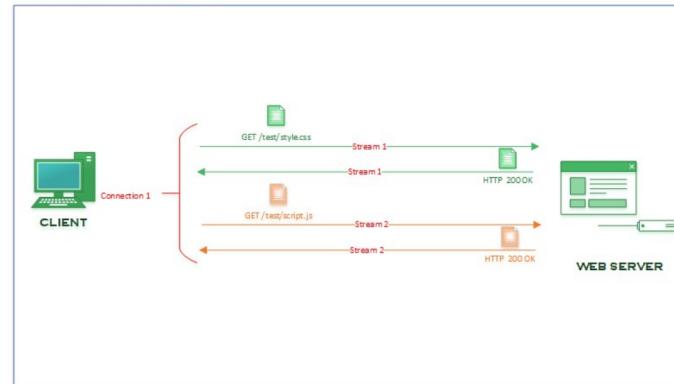
// un exemple avec message imbriqué
message AMoreCompleteExample {
    string stringSameAsBytes = 1;
    repeated string stringList = 2;
    map<int32,string> canBeAMap = 3;
}
```

# HTTP2 : multiplexing dans la même socket TCP

Sequences of rendering the web page



HTTP/2 multiplexing(Parallelizing)

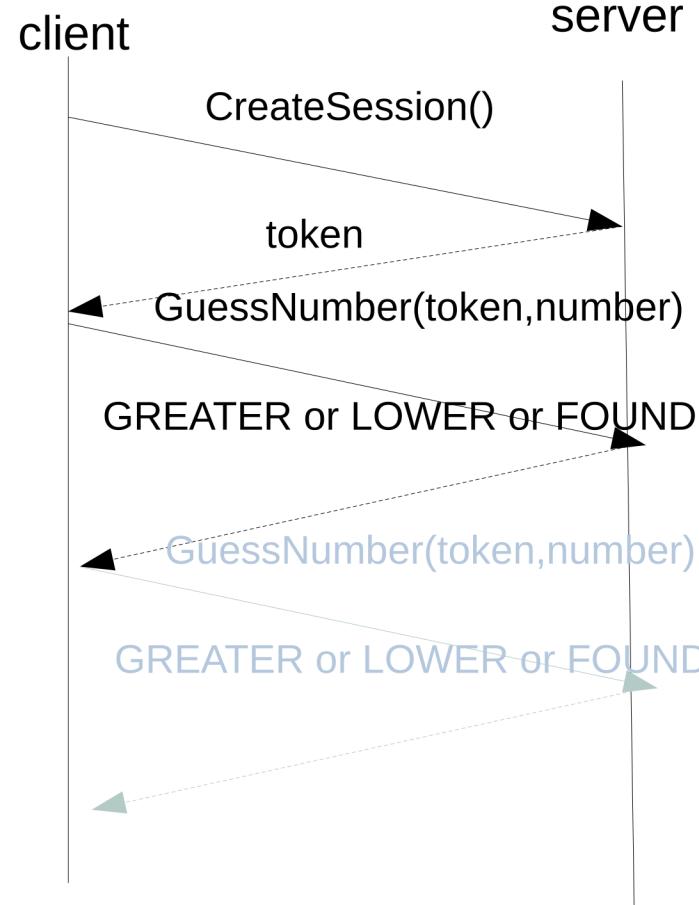


# Le framework gRPC

- Implémente de manière transparente HTTP2 et Protobuf pour l'exécution de fonction distante
- Permet la mise en place d'API synchrone ou asynchrone
- Serveur Multi-thread
- Gestion load balancing / round robin intégrée
- Possibilité de faire évoluer le schéma
- Gestion des erreurs intégrée
- Gestion de flux via le mot clé “stream”



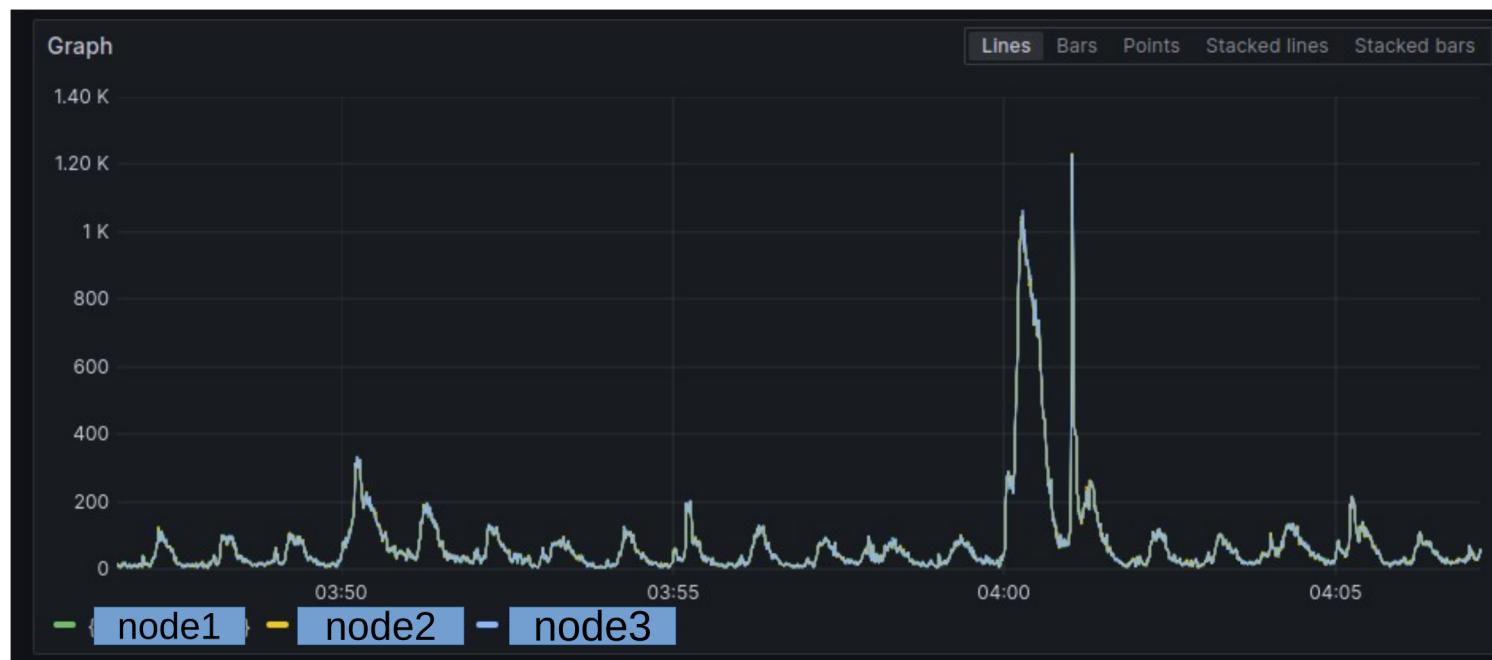
# Demo : GuessANumber Service



# Les bonnes pratiques

- Mettre des messages dédiés pour chaque fonction gRPC
- Privilégier les enums au lieu des strings
- Connaitre les valeurs échangées :
  - sint32/64 gère mieux les valeurs négatives
  - uint/int 32/64 gère mieux des unsigned
- Pour les évolutions de schéma protobuff :
  - Ne pas changer l'ordre des champs
  - Si besoin on peut renommer un champ par exemple en le préfixant par “OBSOLETE\_”
  - Mettez la version dans le nom du package en cas de « breaking change »
- Utiliser des streams pour des données supérieures à 4MB

# Et ça fonctionne !



# Conclusion : osez gRPC !

- Conçu pour une communication interservice moderne :

- Facile à mettre en place
- Efficace : Encodage binaire & HTTP2 (ou HTTP3)
- Robuste : techno reconnue
- Structurée : à l'aide du fichier proto
- Sécurisée : TLS
- Scalable : Load balancing intégré

# Thank you

Body copy here is left aligned and set to 8pt. Keep body copy to a maximum of 2 lines on this page to not detract from the title and important information above