

# Mettez du binaire dans vos API avec gRPC

Breizhcamp 2022



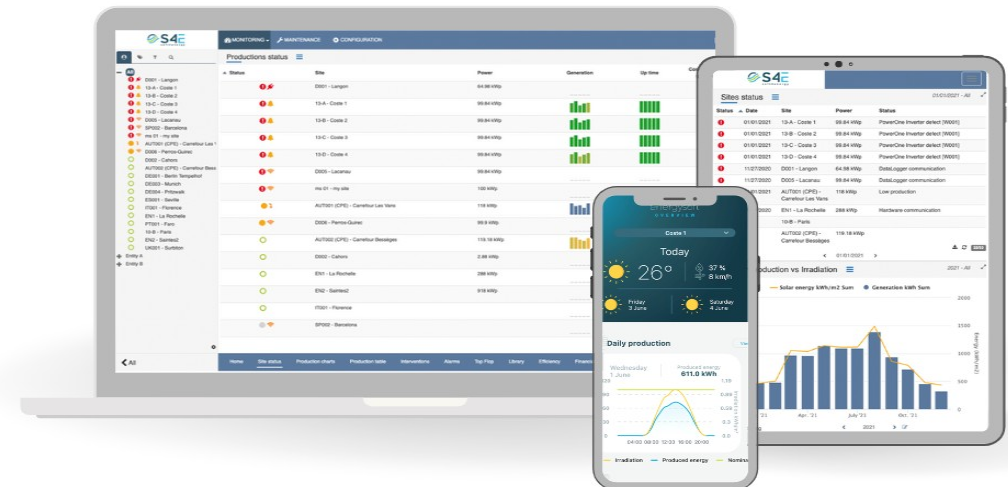
**@jbarotin**

# S4E

Editeur de logiciel Lorientais

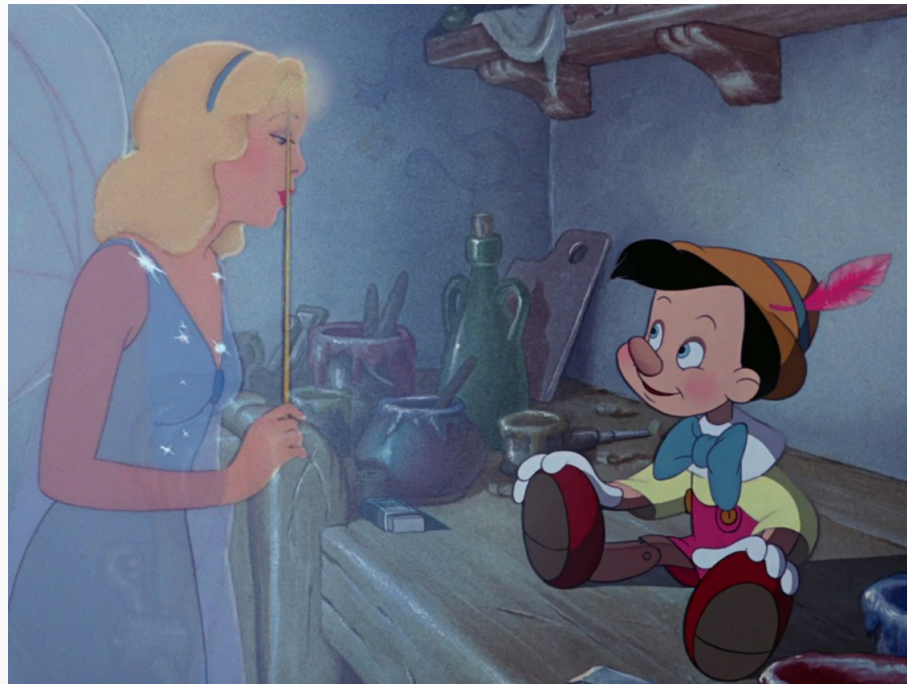
Energysoft :

- une solution SaaS de gestion d'énergie
- spécialisé dans les centrales solaires
- 2GWc supervisés en temps réel



# 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 avec utilisant la librairie Jersey.

# 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 avec utilisant la librairie Jersey.

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  
NOMBRES DE  
NOEUDS SANS RIEN  
CHANGER DANS LE CODE**



**TROUVER  
UNE SOLUTION  
PLUS EFFICACE  
ET CONSOMMER  
MOINS DE RESSOURCES**

# gRPC en bref

Inventé par Google

Sous licence Apache 2.0 depuis 2015

Membre de la 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

# Protocol buffers : un encodage binaire

Défini et créé par Google

<https://developers.google.com/protocol-buffers>

Open source depuis 2008



Technos concurrentes :

- Apache Thrift
- ASN1
- Apache Avro

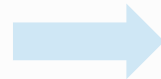


# Protobuff : le fichier proto

Fichier proto : un schéma des données imposées et évolutifs.



```
message MessageBreizhcamp {  
  int32 green = 1;  
  string it = 2;  
  int64 isBeautiful = 3;  
}
```



```
MessageBreizhcamp bzh = MessageBreizhcamp.newBuilder().setGreen(1)  
    .setIt("yes")  
    .setIsBeautiful(2).build();  
byte[] serialize = bzh.toByteArray();
```



08 01	12 03 79 65 73	18 02
-------	----------------	-------

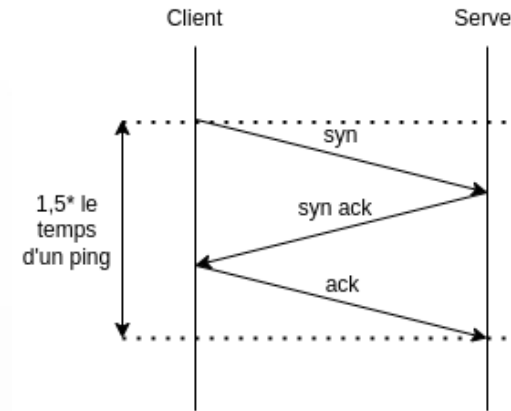
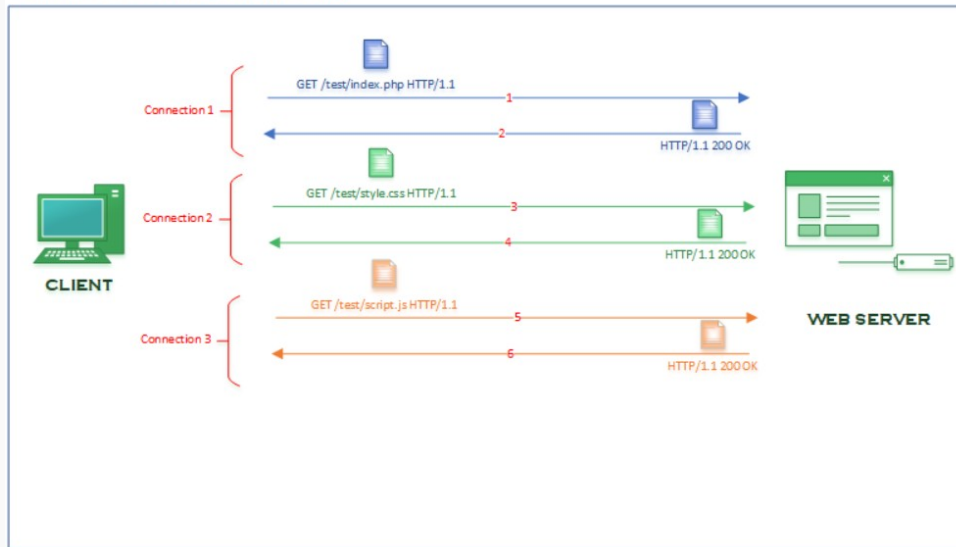
# Protobuff : les types de données

```
enum ThisIsoption {  
    optionA=0;  
    optionB=1;  
    optionC=2;  
};  
  
message SubStructure{  
    bool firstMember = 1;  
    int32 aSecondMember = 2;  
    int64 aThirdMember = 3;  
    bytes aBytesSequence = 4;  
};  
  
message AMoreCompeteExample {  
    string this = 1;  
    repeated SubStructure subCanBeRepeated = 2;  
    optional string anOptionalMember = 3;  
};
```

Taille de paquet limité à 1MB

# HTTP2 : meilleure utilisation de TCP/IP

Sequences of rendering the web page



Config Nginx

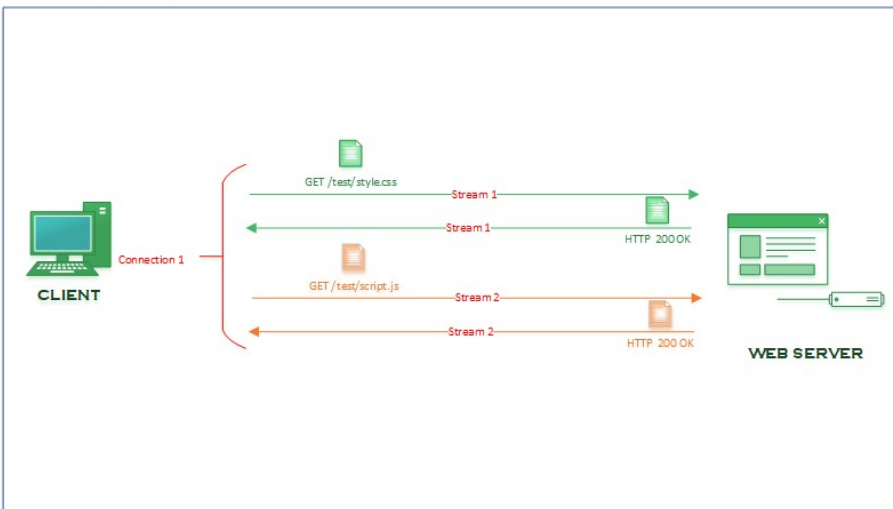
Avant :

```
server {  
    listen 443 ssl;  
    ....  
}
```

Après :

```
server {  
    listen 443 ssl http2;  
    ....  
}
```

HTTP/2 multiplexing(Parallelizing)

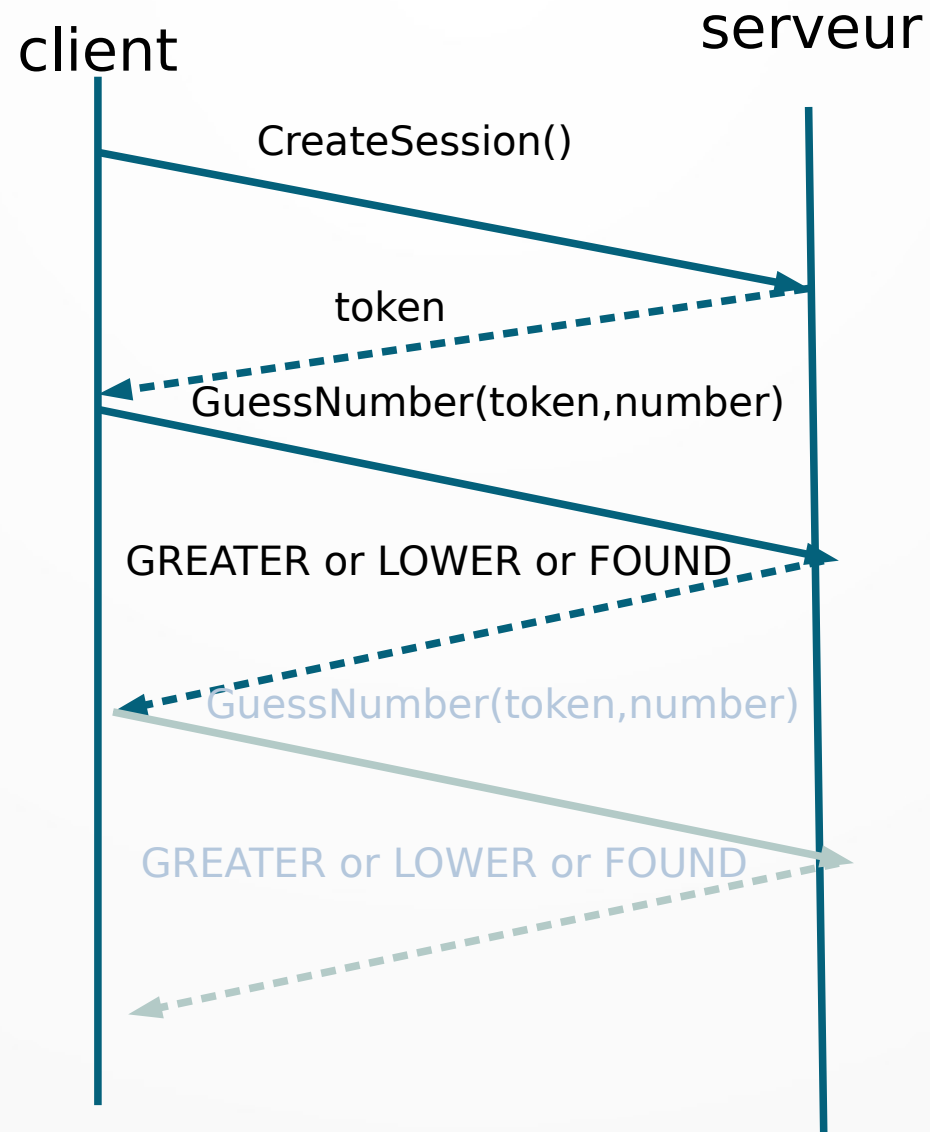


# gRPC : le pack

- API synchrone ou asynchrone
- Stream full duplex ou half duplex
- Serveur Multi-thread
- Gestion round robin intégrée
- Possibilité de faire évoluer le schéma
- Gestion des erreurs
- Mais :
  - Pas de schéma dynamique
  - Pas d'accès « texte »



# Demo : GuessANumber Service





# Les bonnes pratiques

- Privilégier les enums au lieu des strings
- Pour les évolutions de schéma :
  - Mettre les nouveaux champs en optionnel
  - Mettez la version dans le nom du package en cas de « breaking change »
- Mettez en place des streams pour des données supérieures à 1MB

# Conclusion : osez gRPC !

- Pour une communication interservice :
  - Facile à mettre en place
  - Efficace : Encodage binaire & HTTP2 (ou HTTP3)
  - Robuste : techno reconnue (plus de 7 ans d'existence)
  - Structurée : à l'aide du fichier proto
  - Sécurisée (TLS)
  - Load balancable
  - Econome en CPU et en bande passante
- Pour les applications mobiles et l'IoT
- Code et présentation dispo :  
[https://github.com/jbarotin/grpc\\_breizhcamp\\_2022](https://github.com/jbarotin/grpc_breizhcamp_2022)

