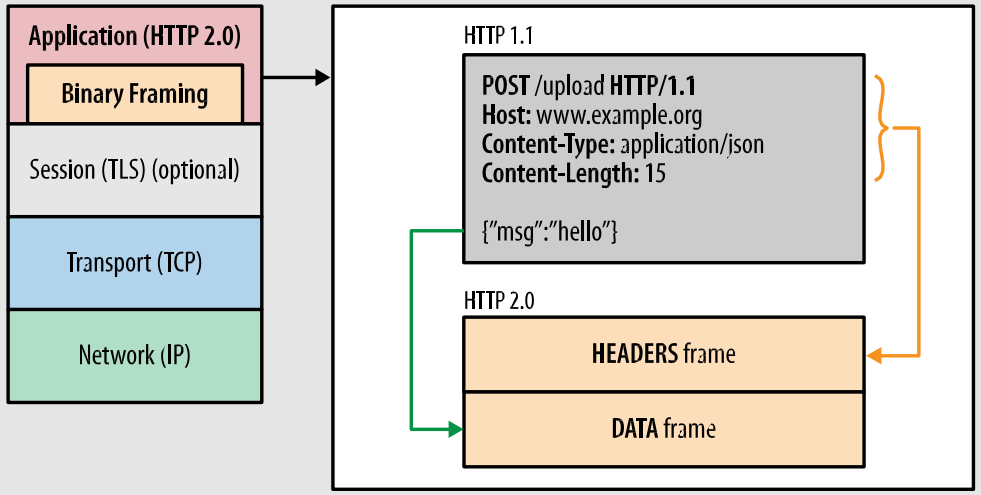
By Ahmed Abouzeid

Agenda

- Brief about HTTP/2
- gRPC Motivation
- What's gRPC ?
- gRPC Testing Tools
- gRPC Method Types
- gRPC Service Versioning
- Authentication and Authorization
- gRPC on .Net Supported Platforms
- gRPC-Web
- When to use gRPC ?
- gRPC JSON transcoding

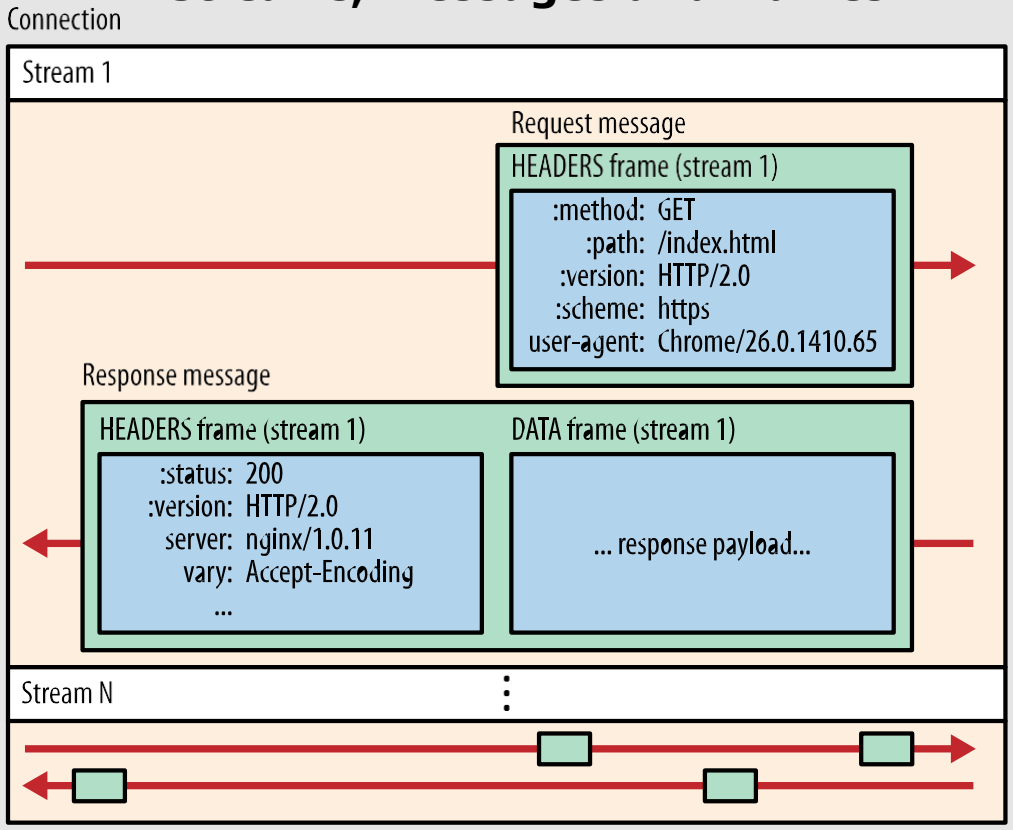
Brief about HTTP/2

- Experimental protocol called ***SPDY*** developed by **Google** & announced in 2009
- ***SPDY*** achieves improvement in page load to **55%** faster
- **HTTP/2** standard approved and published in 2015
- **HTTP/2** 's primary change focus on improving the performance:
- Semantics of **HTTP/1.x** are the same: methods, status code, URI and headers
- **Binary framing layer**



Brief about HTTP/2 (Cont.)

- **Streams, messages and frames**



Stream: A bidirectional flow of bytes within an established connection, which may carry one or more messages

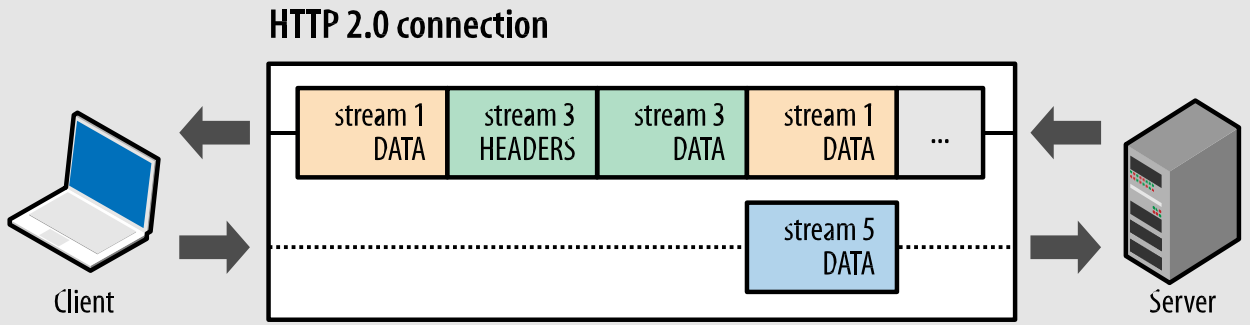
Message: A complete sequence of frames that map to a logical request or response message

Frame: The smallest unit of communication, each containing a frame header identifies the stream to which the frame belongs.

All communication is performed over a **single TCP** connection that can carry **any number of bidirectional streams**.

Brief about HTTP/2 (Cont.)

- **Multiplexing**: multiple streams within same connection to enable parallel requests on single TCP connection
- **Server Push**: is the ability of the server to send multiple responses for a single client request



- **Header Compression** – HPACK Algorithm

gRPC Motivation

- General Purpose RPC infrastructure by **Google** called **Stubby**
- **Stubby** connect large number of microservices within **Google** DCs
- Not based on any standard

gRPC Principles

- Free & Open
- General Purpose & Performant
- Payload Agnostic
- Blocking & Non-Blocking
- Flow Control
- Layered
- Streaming
- Cancellation & Timeout
- Pluggable

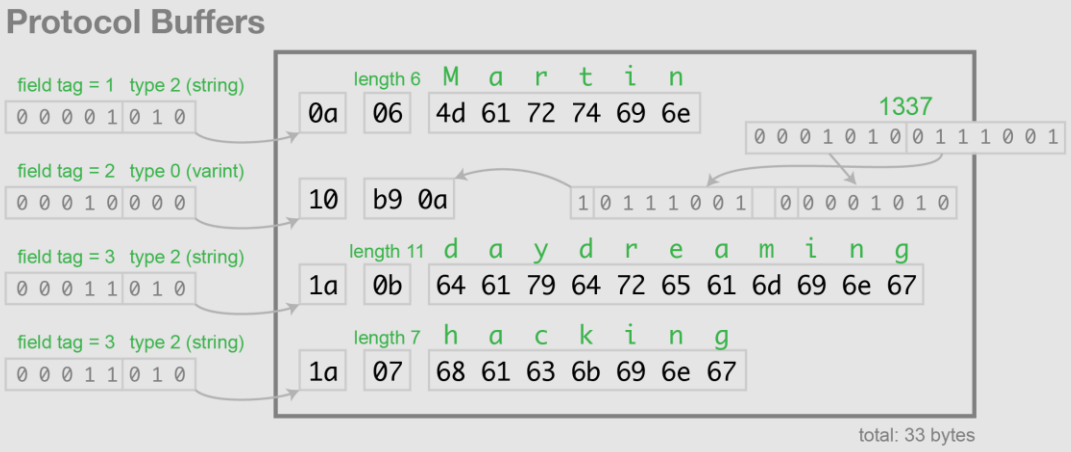
What's gRPC ?

Microsoft | Docs:

> gRPC is a language agnostic, high-performance Remote Procedure Call (RPC) framework.

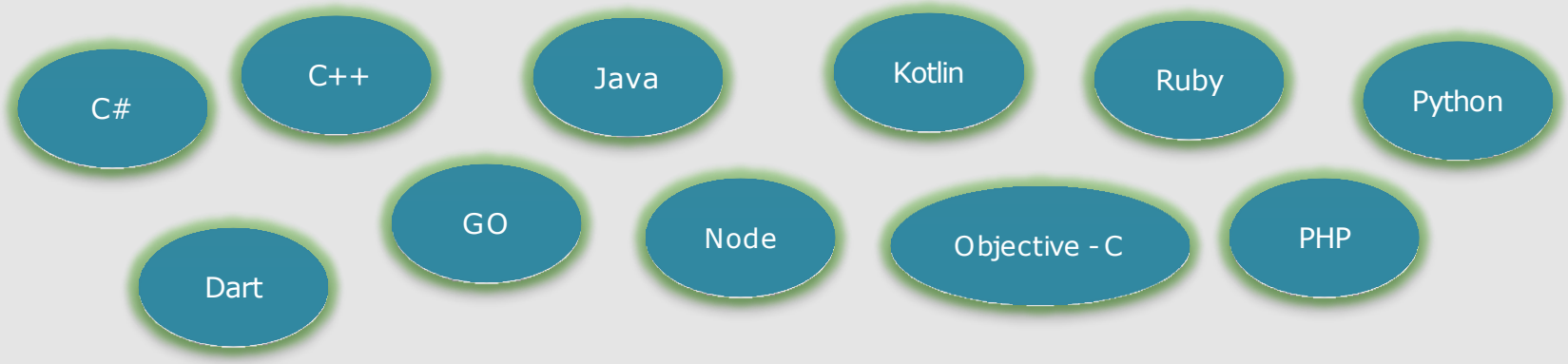
Benefits

- High-performance, lightweight RPC framework
- By default, gRPC uses *Contract-first API* development using **Protocol Buffers** (aka **Protobuf**) as *Interface Definition Language*
- Reduced network usage with **Protobuf** binary serialization
- Supports client, server, and bi-directional streaming calls.



What's gRPC ? (Cont.)

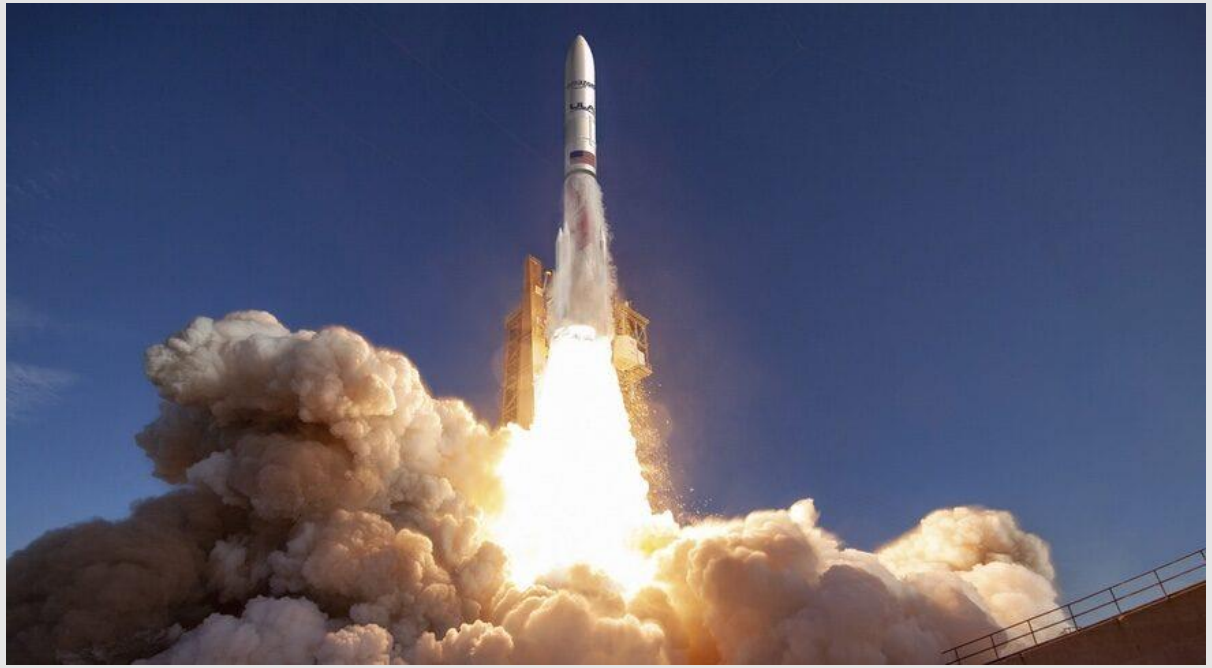
- **gRPC** provides protocol buffer compiler plugins to generate strongly-typed servers and clients



gRPC

Demo

ProtoBuf | My first Server | My first Client



gRPC Testing Tools

- **Postman:** under building blocks section
- **gRPCurl**
  Command-line tool created by the gRPC community
- **gRPCui**
  Interactive web UI for gRPC on top of gRPCurl
- **BloomRPC:** no support for gRPC reflection

gRPC Method Types

- **Unary**

```
rpc SayHello(HelloRequest) returns (HelloResponse);
```

- **Server Streaming**

```
rpc SayHello(HelloRequest) returns (stream HelloResponse);
```

- **Client Streaming**

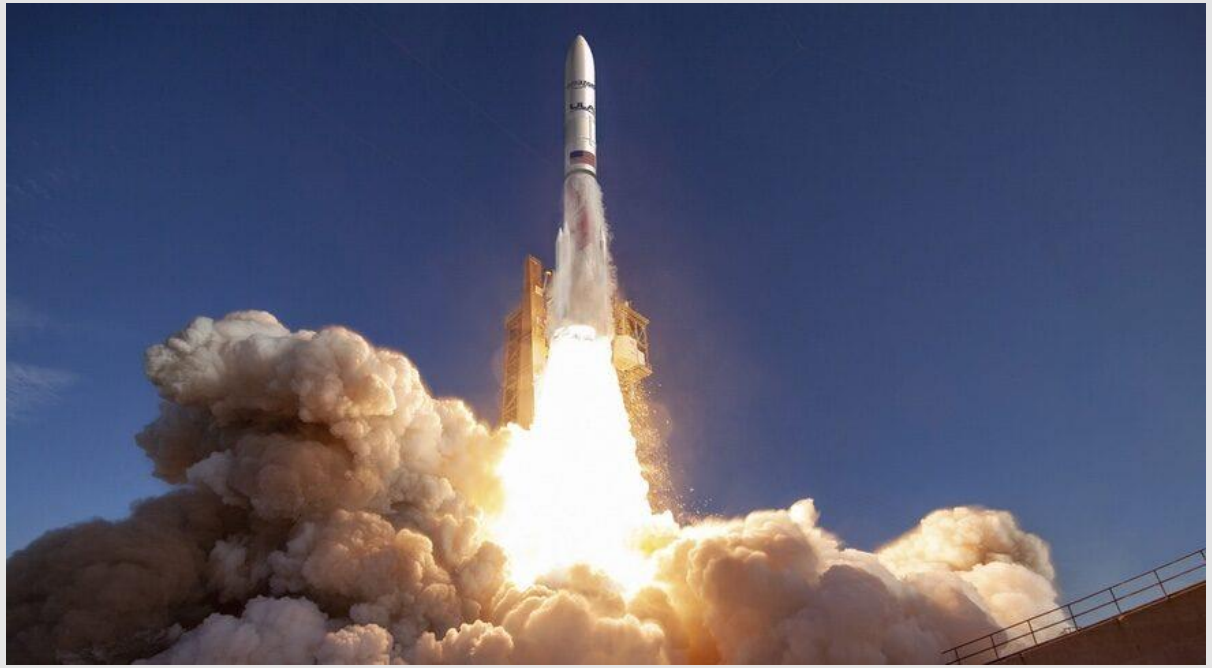
```
rpc SayHello(stream HelloRequest) returns (HelloResponse);
```

- **Bi-directional streaming**

```
rpc SayHello(stream HelloRequest) returns (stream HelloResponse);
```

Demo

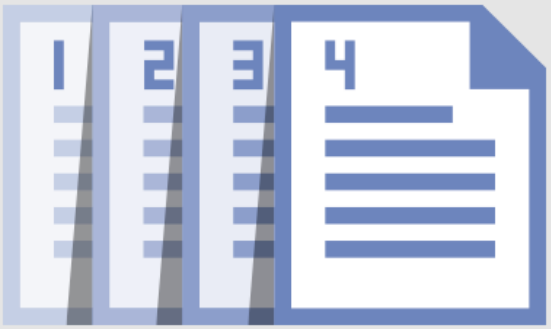
gRPCurl | gRPCui | Build Streams



gRPC Service Versioning

- **Non Breaking Changes Benefits:**
  - Existing clients continue to run
  - Avoid notifying clients of breaking changes, and updating them
  - Only one service version maintained

- **Non-breaking changes**
  - Add New Service
  - Adding a new method to a service
  - Adding a field to a request message – Deserialized with default values on the server
  - Adding a field to a response message – Deserialized into unknown field on the client
  - Adding a value to an enum

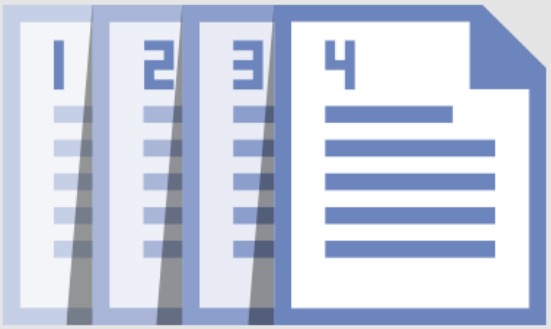


gRPC Service Versioning (Cont.)

- **Binary breaking changes**
  - Removing a field – use *reserved* keyword
  - Renaming a message – unless using *google.protobuf.Any* type, names are sent
  - Nesting or unnesting a message
  - Changing csharp_namespace

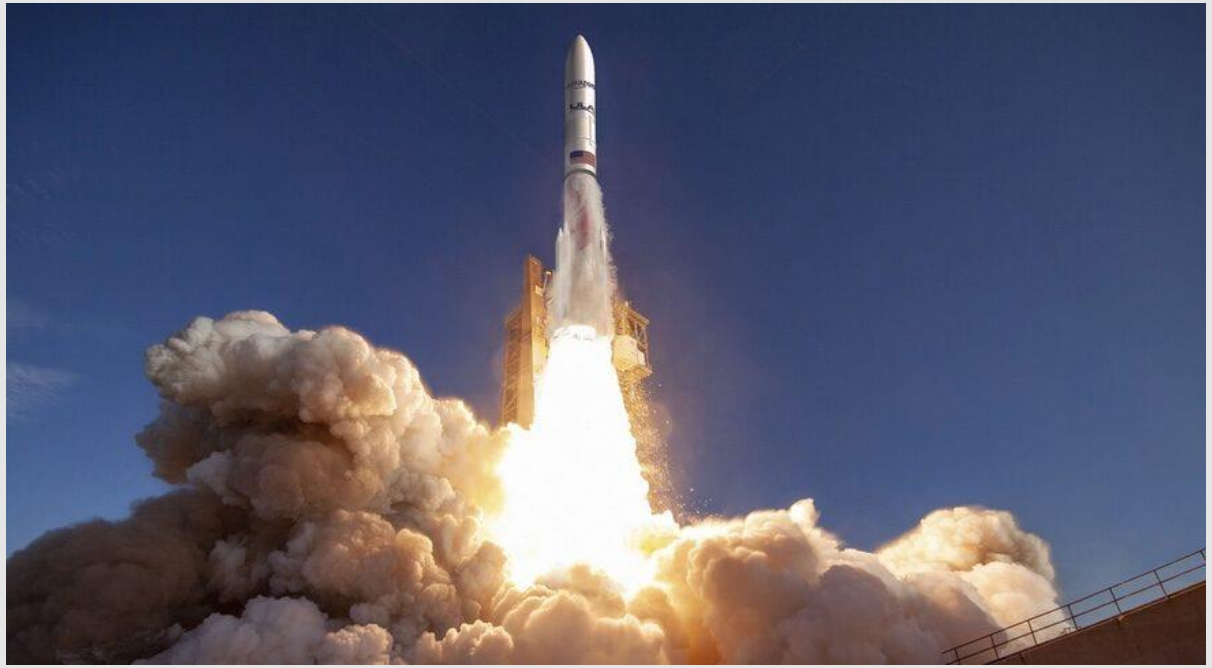
```
import "google/protobuf/any.proto";

message ErrorStatus {
  string message = 1;
  repeated google.protobuf.Any details = 2;
}
```

- **Protocol breaking changes**
  - Changing a field data type – incompatible types will cause deserialization errors
  - Changing a field number
  - Renaming a package, service or method - *UNIMPLEMENTED* status from the server
  - Removing a service or method - *UNIMPLEMENTED* status from the server



Demo

Versioning | RPC Exception



Authentication & Authorization

- **Integrate with ASP.NET Core Authentication**

Demo

gRPC on .Net Supported Platforms

- .NET 5 or Later
- .NET Core 3

- Windows
- Linux
- macOS – No support for ASP.NET Core apps with HTTPS before .NET 8

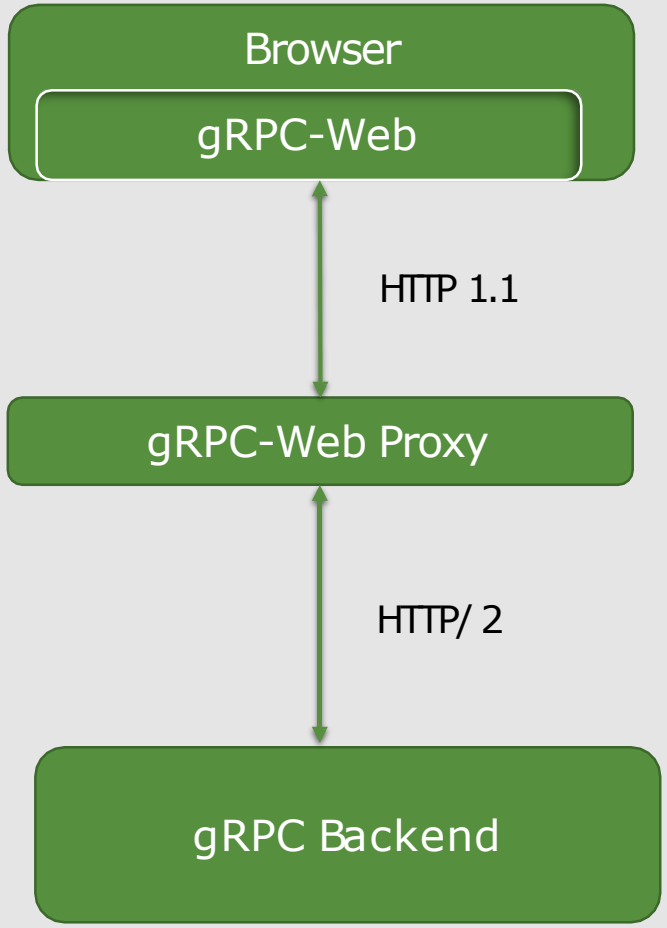
- Kestrel
- TestServer
- IIS *
- HTTP.sys *

* Requires .NET 5 and Windows 11 Build 22000 or Windows Server 2022 Build 20348 or later

- Azure Kubernetes Service (AKS)
- Azure Container Apps
- Azure App Service – new*

gRPC-Web

- gRPC-Web allows browser JavaScript and Blazor apps to call gRPC services
- Transforming gRPC requests by encoding messages in base64
- Two choices for gRPC-Web in .NET:
  - ASP.NET Core Middleware
  - Envoy Proxy
- Unary and server push method types are only supported
- JavaScript clients and .NET clients for Blazor



Demo

gRPC-Web (ts client)

- ✓ Install protoc & install it at sys PATH
  https://github.com/protocolbuffers/protobuf/releases/download/v3.20.1/protoc-3.20.1-win64.zip
- ✓ Install packages
  - ✓ ts-protoc-gen
  - ✓ google-protobuf
  - ✓ @types/google-protobuf
  - ✓ @improbable-eng/grpc-web
- ✓ protoc --plugin=protoc-gen-ts="node_modules\.bin\protoc-gen-ts.cmd" --
  js_out="import_style=commonjs,binary:src/app/generated" --ts_out="service=grpc-web:src/app/generated"
  src/app/protos/*.proto

When to use gRPC ?

- Real-time communication services where you deal with streaming calls
- When efficient communication is a goal
- In multi-language environments
- For internal APIs where you don’t have to force technology choices on clients

gRPC JSON transcoding

- An extension for ASP.NET Core that creates RESTful JSON APIs for gRPC services
- Allows apps to call gRPC services with familiar HTTP concepts: HTTP verbs, HTTP verbs, JSON request / response
- Supported starting from **.NET 7**

```
syntax = "proto3";

import "google/api/annotations.proto";

package greet;

service Greeter {
  rpc SayHello (HelloRequest) returns (HelloReply) {
    option (google.api.http) = {
      get: "/v1/greeter/{name}"
    };
  }
}
```

Resources

- https://web.dev/performance-http2/
- https://grpc.io/blog/principles/
- https://grpc.io/docs/what-is-grpc/core-concepts
- https://docs.microsoft.com/en-us/aspnet/core/grpc/services
- https://docs.microsoft.com/en-us/aspnet/core/grpc/protobuf
- https://docs.microsoft.com/en-us/aspnet/core/grpc/versioning
- https://app.pluralsight.com/library/courses/aspnet-core-grpc/table-of-contents
- https://channel9.msdn.com/Shows/On-NET/gRPC-Web-with-NET
- https://docs.microsoft.com/en-us/aspnet/core/grpc/browser
- https://anthonygiretti.com/2020/03/29/grpc-asp-net-core-3-1-how-to-create-a-grpc-web-client-examples-with-angular-8-and-httpclient/
- https://docs.microsoft.com/en-us/aspnet/core/grpc/httpapi

Thanks

ahmed.abouzeid@outlook.com

https://www.linkedin.com/in/amabouzeid/