Contents

1 Installation

1.1 Sample grpc hello-world service

https://hub.docker.com/r/kutbar/grpc-hello-world app runs on port 9000 default. This service doesn't need an authentication. It receives a name as an input, and responds with concat("Hello "+name) as an output. I run it on an external server.

1.2 Kreya App

https://kreya.app/downloads/ for testing grpc services.

1.3 Apisix Docker

The following commands described in https://apisix.apache.org/docs/apisix/getting-started/#step-1-install-apache-apisix are used for installation.

```
# Download the docker-compose file of Apache APISIX
git clone https://github.com/apache/apisix-docker.git
# Switch the current directory to the apisix-docker/example
cd apisix-docker/example
```

Update docker-compose file in apisix_conf directory

docker-compose.yml

Update apisix conf file

config.yaml

```
# Start Apache APISIX with docker-compose
docker-compose -p docker-apisix up -d
```

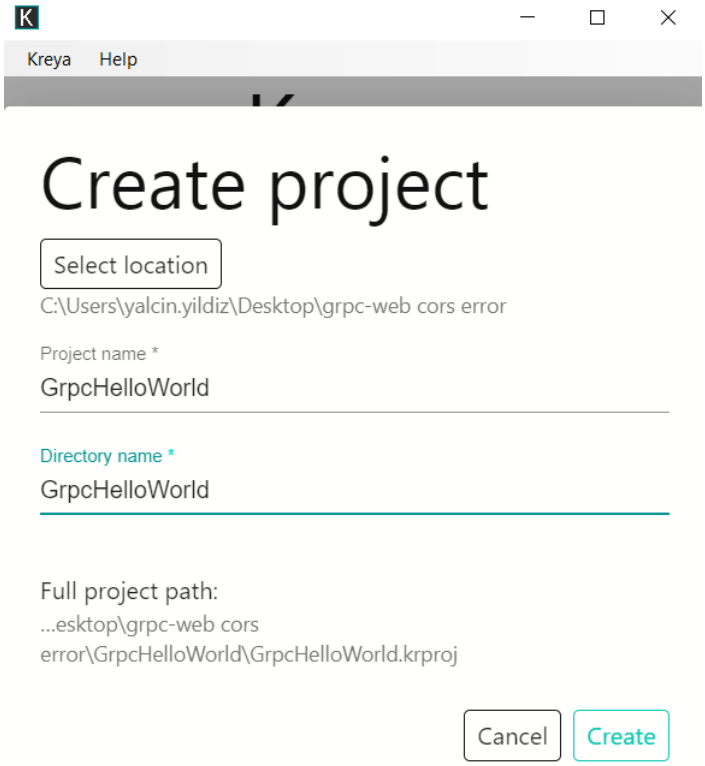
1.4 Sample Angular Project

Unzip first, then run "npm install" and "npm start" commands in the terminal of an appropriate IDE(I'm using Microsoft Visual Studio).

grpc-web-angular.rar

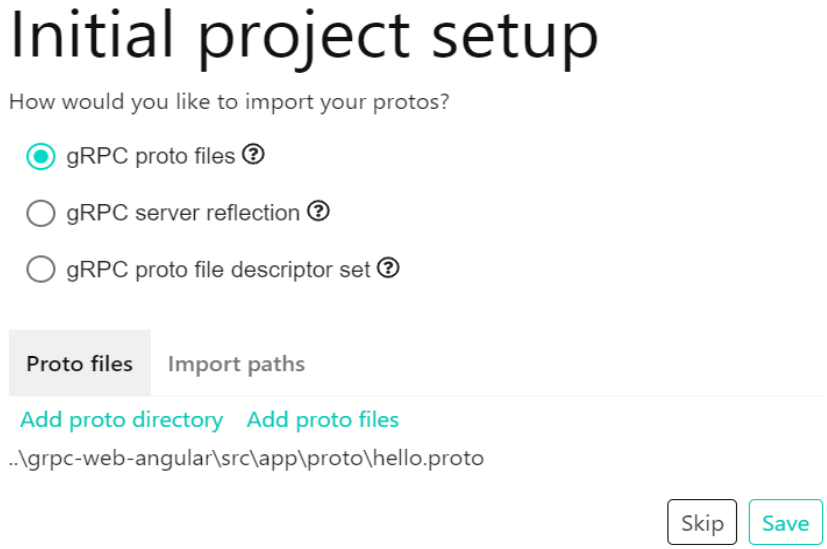
1.5 Call hello-world service via Kreya app

a. First, you need to create a new project with Kreya app. To do it, select “Create Project” on the main screen after starting the app. Then, specify project name and directory name after selecting suitable project location.

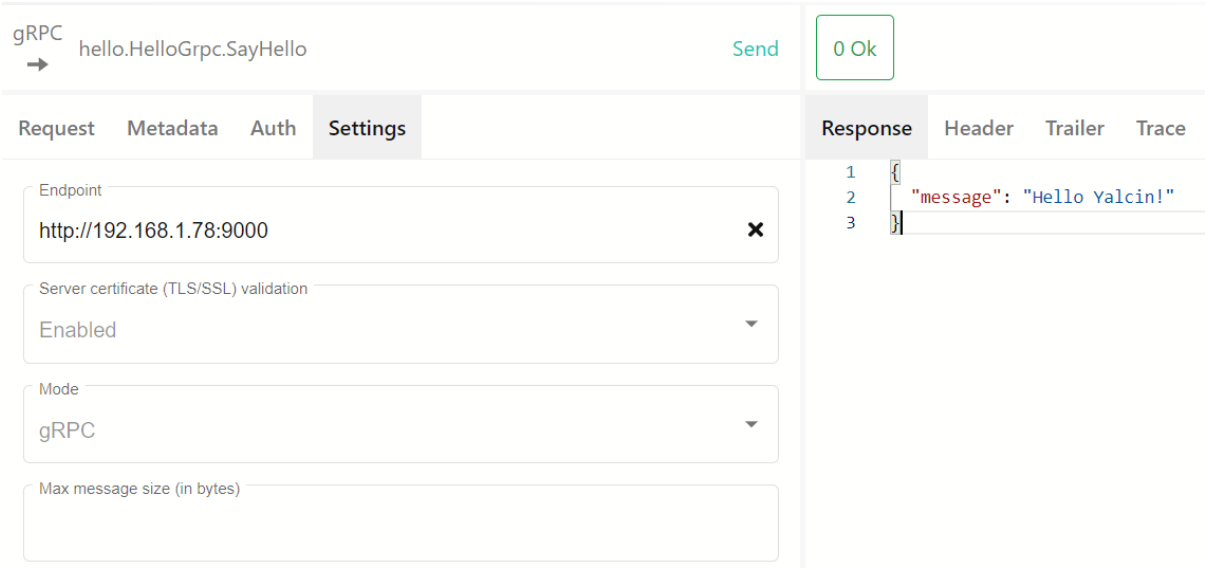


b. Then, you need to select proto file. It is available at grpc-web-angular\src\app\proto\ hello.proto in angular project, also you can use the following file with the same content.


hello.proto



c. We can see that service is up and running.



2 APISIX configurations

2.1 Upstream configuration

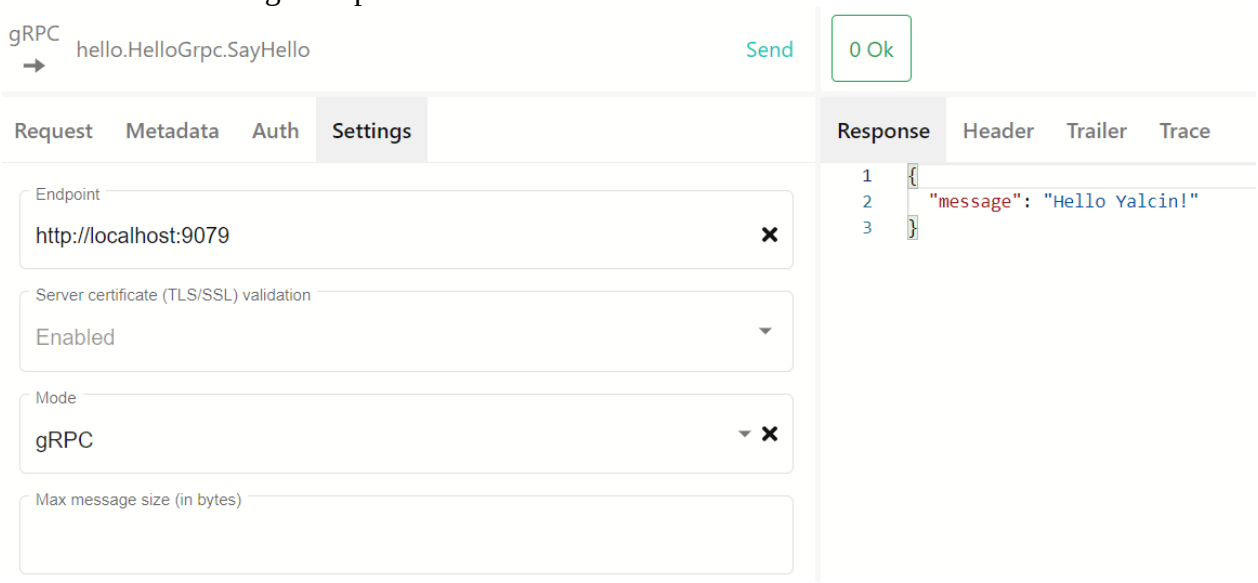
curl "http://localhost:9080/apisix/admin/upstreams/1" -H "X-API-KEY: edd1c9f034335f136f87ad84b625c8f1" -X PUT -d '
{
 "nodes": [
  {
   "host": "192.168.1.78",
   "port": 9000,
   "weight": 1
  }
 ],
 "timeout": {
  "connect": 2,
  "send": 3,
  "read": 4
 },
 "type": "roundrobin",
 "scheme": "grpc",
 "pass_host": "node",
 "name": "Kutbar grpc upstream",
 "desc": "Kutbar grpc upstream",
 "keepalive_pool": {
  "idle_timeout": 60,
  "requests": 1000,
  "size": 320
 }
}'

2.2 Route configuration

```
curl "http://localhost:9080/apisix/admin/routes/1" -H "X-API-KEY:
edd1c9f034335f136f87ad84b625c8f1" -X PUT -d '
{
  "uri": "/*",
  "name": "Kutbar grpc route",
  "methods": [
    "POST",
    "OPTIONS"
  ],
  "upstream_id": "1",
  "status": 1
}'
```

2.3 Test Route

New route is working as expected.



3 Test Steps

3.1 Route with cors plugin

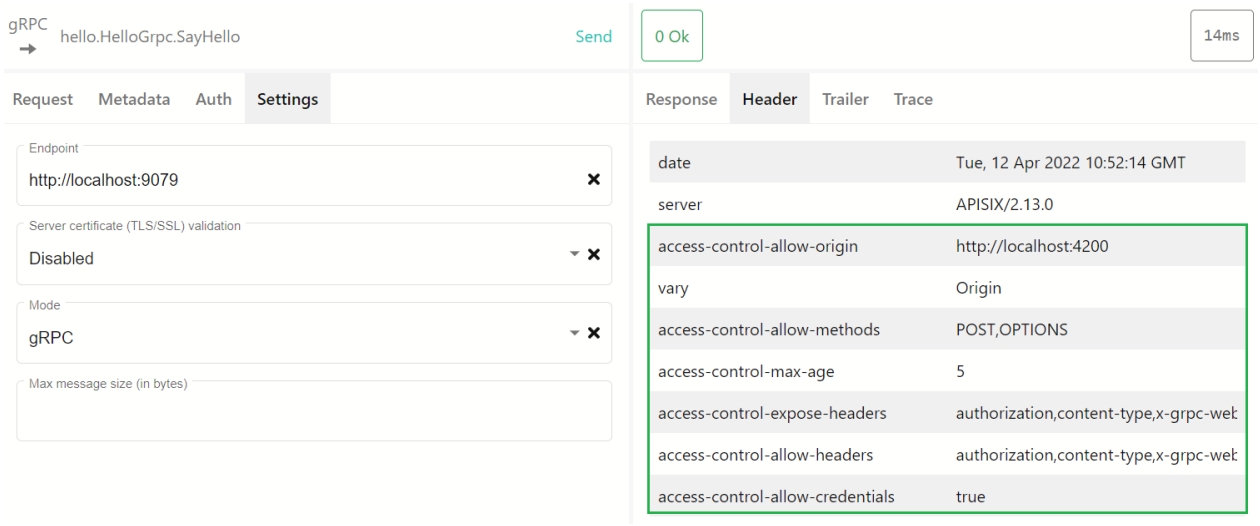
3.1.1 Add cors plugin to route

```
curl "http://localhost:9080/apisix/admin/routes/1" -H "X-API-KEY:
edd1c9f034335f136f87ad84b625c8f1" -X PUT -d '
{
  "uri": "/*",
  "name": "Kutbar grpc route",
  "methods": [
    "POST",
```

```
    "OPTIONS"
  ],
  "plugins": {
    "cors": {
      "allow_credential": true,
      "allow_headers": "authorization,content-type,x-grpc-web,x-user-agent,origin,referer",
      "allow_methods": "POST,OPTIONS",
      "allow_origins": "http://localhost:4200",
      "disable": false,
      "expose_headers": "authorization,content-type,x-grpc-web,x-user-agent,origin,referer",
      "max_age": 5
    }
  },
  "upstream_id": "1",
  "status": 1
}'
```

3.1.2 Test with Kreya App

The cors plugin is working fine.



3.2 Route with grpc-web plugin

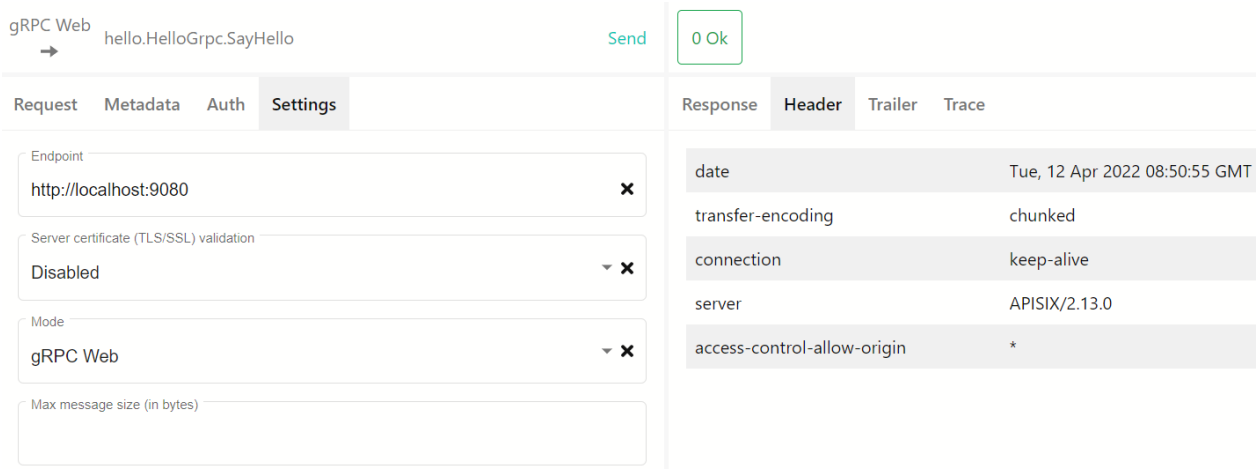
3.2.1 Add grpc-web plugin to route

```
curl "http://localhost:9080/apisix/admin/routes/1" -H "X-API-KEY:
edd1c9f034335f136f87ad84b625c8f1" -X PUT -d '
{
  "uri": "/*",
  "name": "Kutbar grpc route",
  "methods": [
    "POST",
    "OPTIONS"
  ],
```

```
  "plugins": {
    "grpc-web": {}
  },
  "upstream_id": "1",
  "status": 1
}'
```

3.2.2 Test with Kreya App

Since grpc-web works with HTTP1.1, port is updated to 9080. Both “gRPC Web” and “gRPC Web (Text)” modes are working fine and we have allow-origin header with the value ‘*’.



3.3 Add cors & grpc-web plugins to route

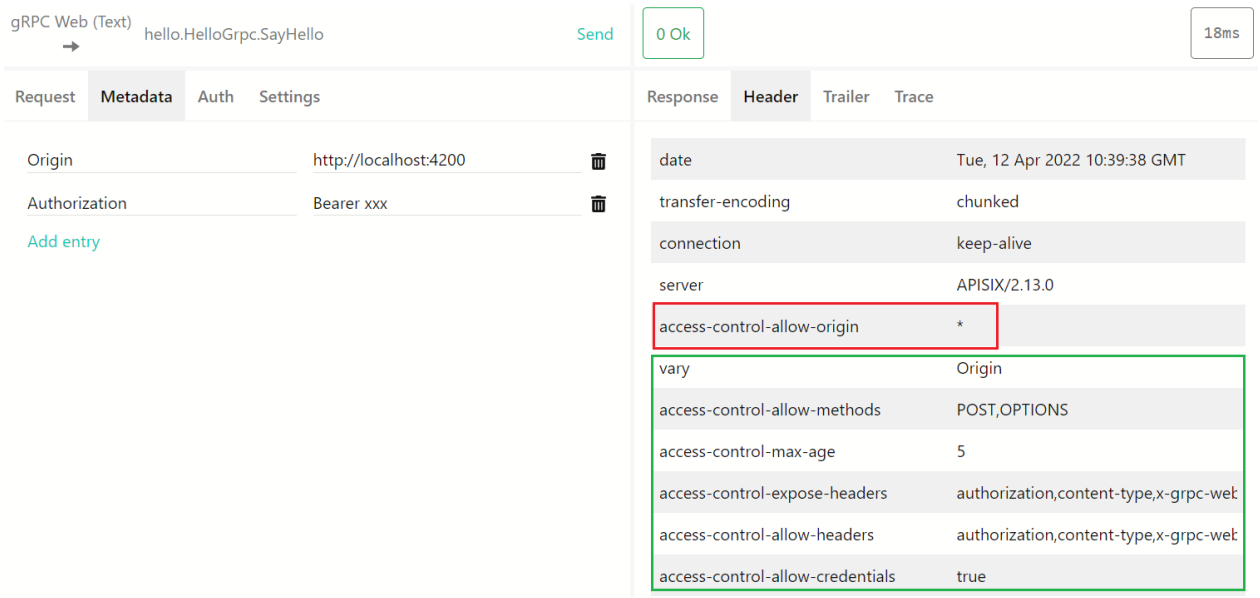
3.3.1 Add cors & grpc-web plugins to route

```
curl "http://localhost:9080/apisix/admin/routes/1" -H "X-API-KEY:
edd1c9f034335f136f87ad84b625c8f1" -X PUT -d '
{
  "uri": "/*",
  "name": "Kutbar grpc route",
  "methods": [
    "POST",
    "OPTIONS"
  ],
  "plugins": {
    "cors": {
      "allow_credential": true,
      "allow_headers": "authorization,content-type,x-grpc-web,x-user-agent,origin,referer",
      "allow_methods": "POST,OPTIONS",
      "allow_origins": "http://localhost:4200",
      "disable": false,
      "expose_headers": "authorization,content-type,x-grpc-web,x-user-agent,origin,referer",
      "max_age": 5
```

```
    },
    "grpc-web": {}
  },
  "upstream_id": "1",
  "status": 1
}'
```

3.3.2 Test with Kreya App

Both “gRPC Web” and “gRPC Web (Text)” modes are working successful. Cors headers other than allow-origin are available as expected. Allow-origin header value should be “http://localhost:4200”



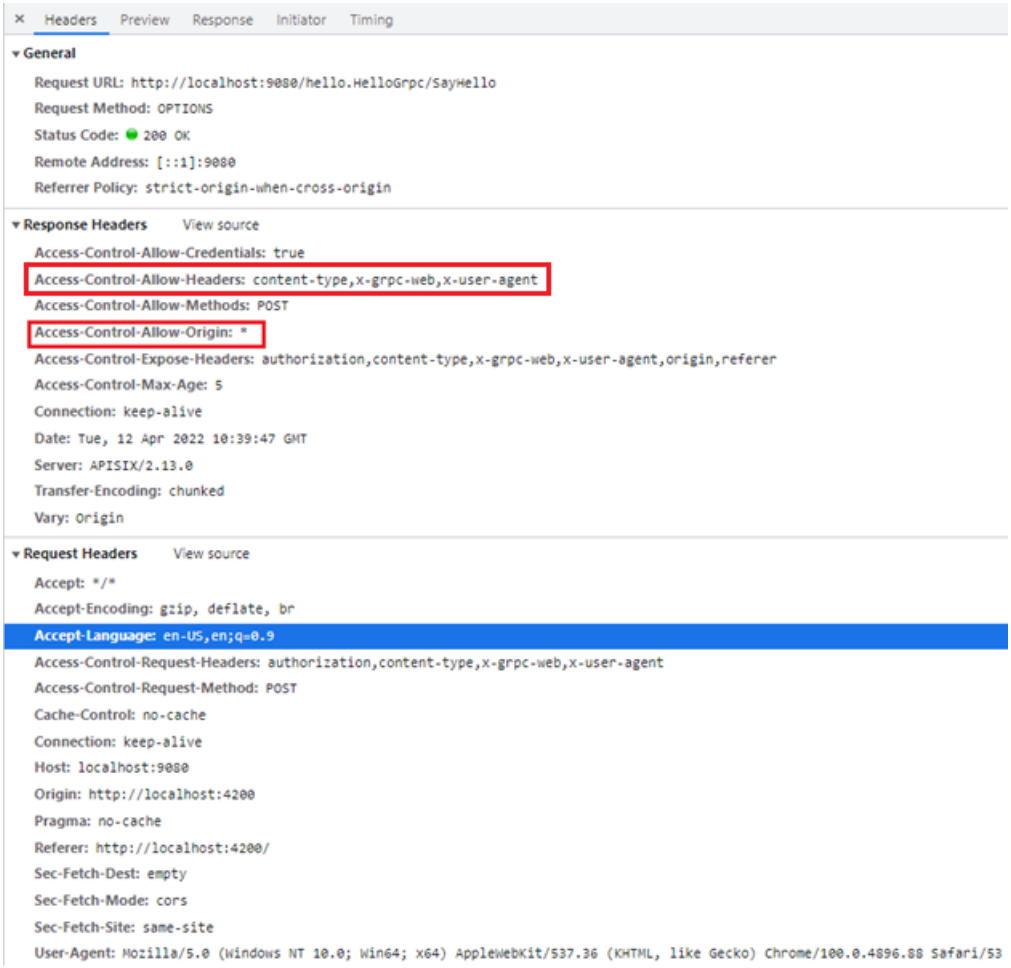
3.3.3 Test with Angular Project

As in previous test, authorization and origin headers are sent by Angular app.

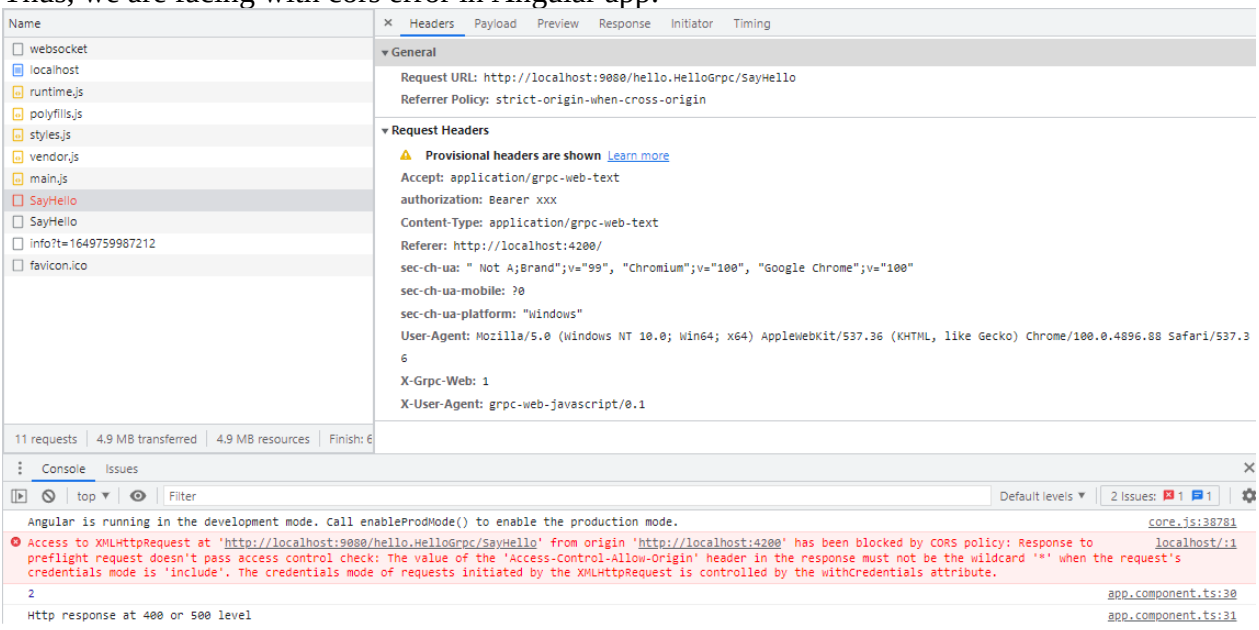
```
const helloService = new HelloGrpcClient('http://localhost:9080',null, {
withCredentials: true });

        const helloReq = new HelloRequest();
        helloReq.setName('Angular');
        const helloHeaders = {
          authorization: 'Bearer xxx'
        };
```

Cors headers other than allow-origin and allow-headers are available as expected. Allow-origin header value should be http://localhost:4200, and allow-headers value should be “authorization,content-type,x-grpc-web,x-user-agent,origin,referer”.



Thus, we are facing with cors error in Angular app.



Access to XMLHttpRequest at 'http://localhost:9080/hello.HelloGrpc/SayHello' from origin 'http://localhost:4200' has been blocked by CORS policy: Response to preflight request doesn't pass access control check: The value of the 'Access-Control-Allow-Origin' header in the response must not be the wildcard '*' when the request's credentials mode is 'include'. The credentials mode of requests initiated by the XMLHttpRequest is controlled by the withCredentials attribute.

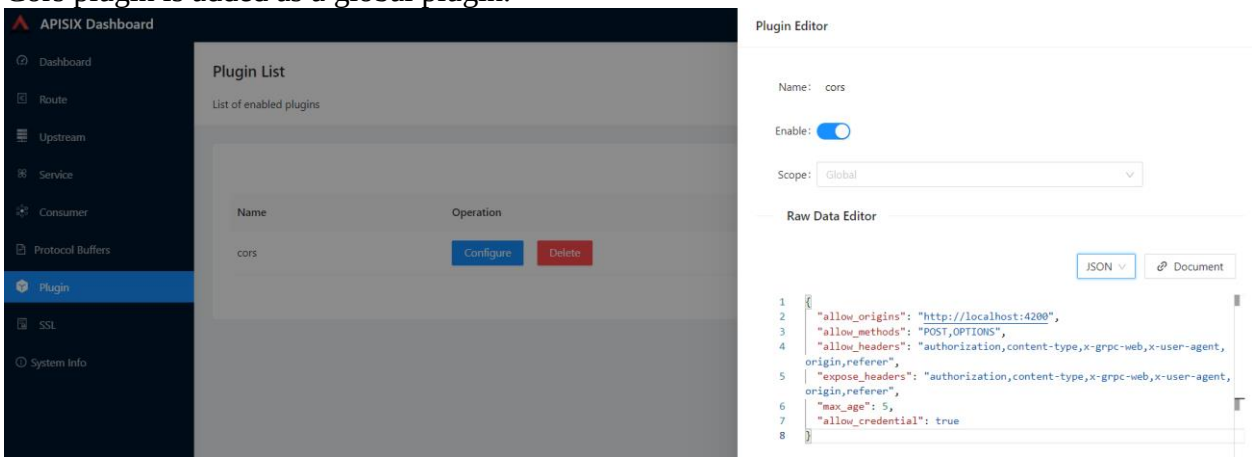
3.4 Add grpc-web plugins to route & global cors plugin

3.4.1 Add grpc-web plugin to route

curl "http://localhost:9080/apisix/admin/routes/1" -H "X-API-KEY: edd1c9f034335f136f87ad84b625c8f1" -X PUT -d '
{
  "uri": "/*",
  "name": "Kutbar grpc route",
  "methods": [
    "POST",
    "OPTIONS"
  ],
  "plugins": {
    "grpc-web": {}
  },
  "upstream_id": "1",
  "status": 1
}'

3.4.2 Add global cors plugin

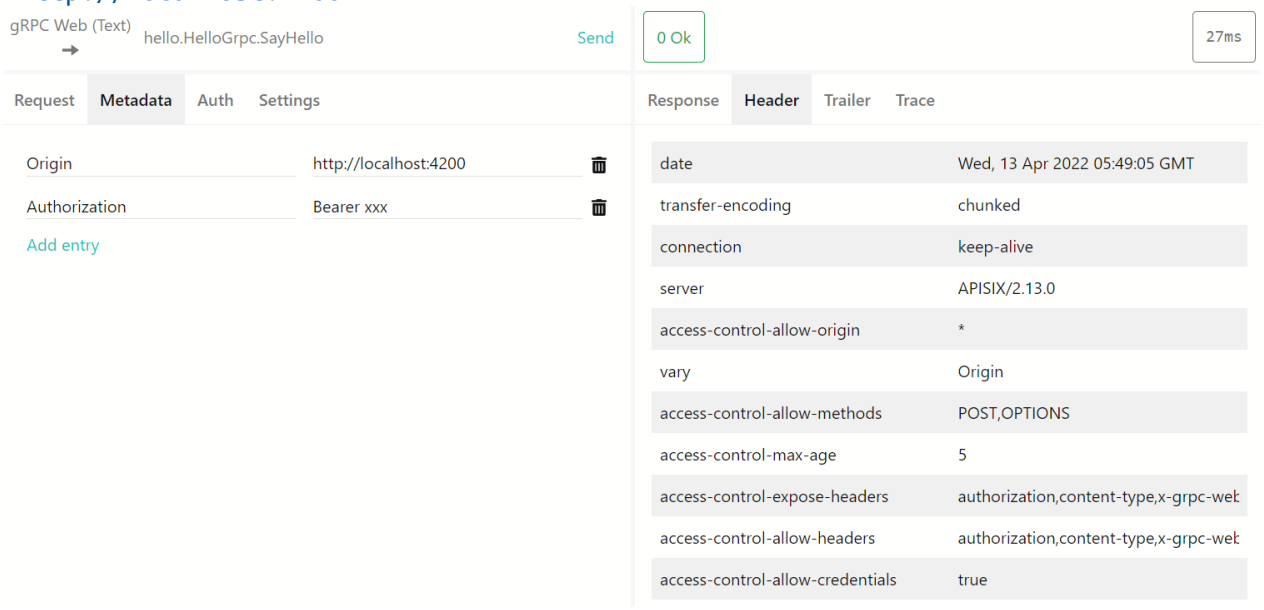
Cors plugin is added as a global plugin.




```
{
  "allow_origins": "http://localhost:4200",
  "allow_methods": "POST,OPTIONS",
  "allow_headers": "authorization,content-type,x-grpc-web,x-user-agent,origin,referer",
  "expose_headers": "authorization,content-type,x-grpc-web,x-user-agent,origin,referer",
  "max_age": 5,
  "allow_credential": true
}
```

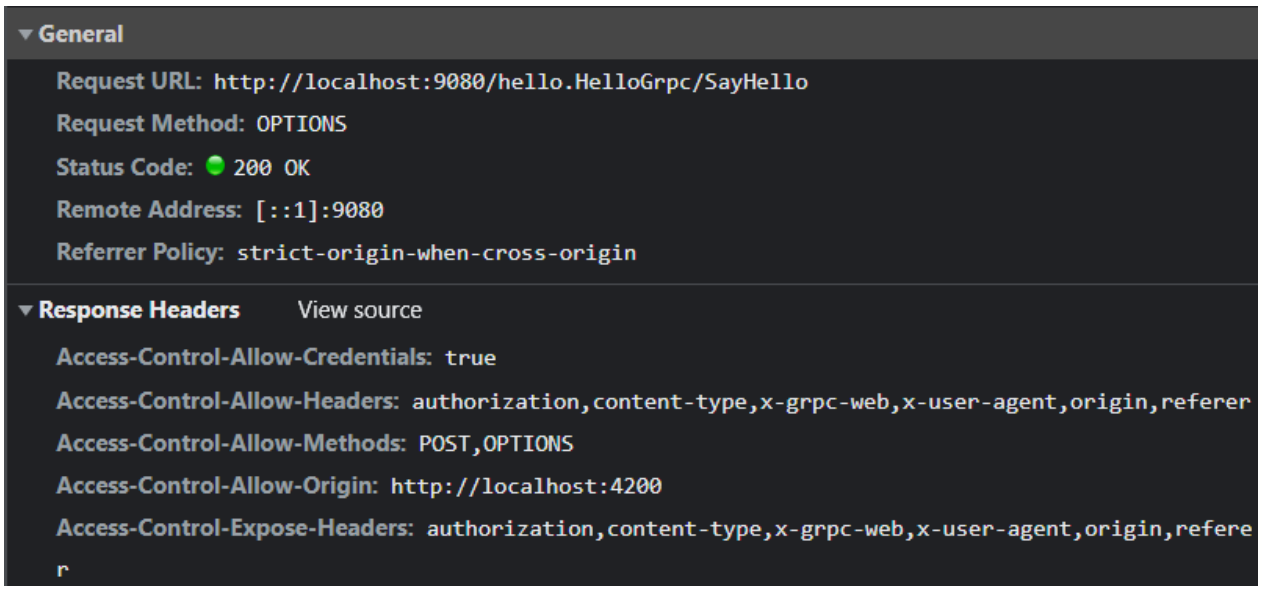
3.4.3 Test with Kreya App

Both “gRPC Web” and “gRPC Web (Text)” modes are working successful. Cors headers other than allow-origin are available as expected. Allow-origin header value should be “`http://localhost:4200`”

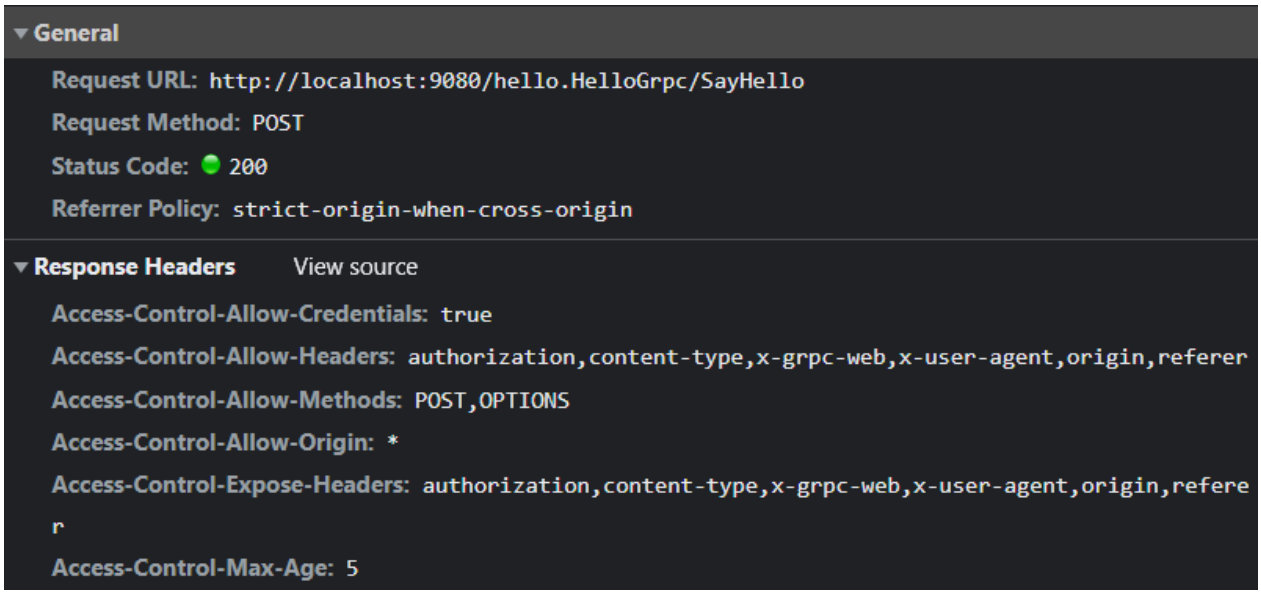


3.4.4 Test with Angular Project

Cors headers other than allow-origin are available as expected. Allow-origin header value should be `http://localhost:4200.`
OPTIONS;



POST;



4 Conclusion

We have gprc services that do and do not require authentication. Grpc services that do not require authentication can be perfectly consumed by web clients after using grpc-web and cors plugins. But we are experiencing cors issues on client side for the grpc services that do require authentication.

Grpc-web plugin manipulates allow-origin and allow-headers cors headers incorrect when it is used with cors plugin in the same route as you can see in section 3.3.

Grpc-web plugin manipulates allow-origin cors header incorrect when it is used with global cors plugin as you can see in section 3.4.