GraphQL Apollo Federation Gateway With Subscription Websocket?

What i use: NestJS + Graphql Apollo Federation + Websocket + Microservices + React + Proxies Forward

- The Problems
  - Graphql Apollo Federation old version doesn't support websocket
  - Different manipulation of execution context types: http (like express), graphql (graphql apollo), rpc (websocket)
- The Solution
  - Setup another instance of Graphql Module for websocket
- The setup
  - Create another Graphql Websocket Module Endpoint
  - SetupProxies for React Front End Ap

Hôm nay lười quá vì mới solve một số vấn đề WSL2, Docker, Nodejs App không expose được port 3000 để đọc trên local dev máy mình đang vừa học vừa dev. Vì thế mình để tạm một số solution cho bạn nào cần và tìm hiểu thêm nhé.

https://cloud.kyluat.com/s/E2mMa4yJWXBXJ3p

```
GraphQLModule.forRootAsync<ApolloDriverConfig>({
      driver: ApolloDriver,
      useFactory: async (authClient: AuthClientsService) => ({
        autoSchemaFile: false,
        cors: true,
        playground: false,
        path: '/api/graphql-normal',
        typePaths: ['./**/*.graphql'],
        installSubscriptionHandlers: true,
        subscriptions: {
          'graphql-ws': {
            path: '/api/ws/graphql',
            onConnect: async (context: any) => {
              try {
                const request = context.extra?.request;
                const jwt =
                  getJwtFromCookie(request?.headers?.cookie) ||
                  getJwtFromAuthorization(context.connectionParams);
                if (!authClient)
                  throw new UnauthorizedException('Error onConnect');
                const user = await authClient.authenticate(jwt);
                context.user = user;
              } catch (err) {
                new Logger().error(err);
                throw new UnauthorizedException('Error onConnect');
              }
            },
          },
        },
      }),
      imports: [AuthClientsModule],
      inject: [AuthClientsService],
    }),
    GraphQLModule.forRootAsync<ApolloFederationDriverConfig>({
      driver: ApolloFederationDriver,
      useFactory: async (configService: ConfigService) => ({
        path: '/api/graphql',
        playground: true,
        cors: true,
        autoSchemaFile: {
          federation: 2,
        },
        cache: new InMemoryLRUCache({
          // MB
          maxSize:
            Math.pow(2, 20) *
            parseInt(configService.get('APOLLO_MEMORY_CACHE_MB')),
          // 5 minutes (in seconds)
          // convert string to int

          ttl: parseInt(configService.get('APOLLO_MEMORY_CACHE_TTL_SEC')),
        }),
        plugins: [
          ApolloServerPluginCacheControl({
            defaultMaxAge: 60,
          }),
        ],
      }),
      inject: [ConfigService],
    }),
```

```
GraphQLModule.forRootAsync<ApolloDriverConfig>({
      driver: ApolloDriver,
      useFactory: async (authClient: AuthClientsService) => ({
        autoSchemaFile: false,
        cors: true,
        playground: false,
        path: '/api/graphql-normal',
        typePaths: ['./**/*.graphql'],
        installSubscriptionHandlers: true,
        subscriptions: {
          'graphql-ws': {
            path: '/api/ws/graphql',
            onConnect: async (context: any) => {
              try {
                const request = context.extra?.request;
                const jwt =
                  getJwtFromCookie(request?.headers?.cookie) ||
                  getJwtFromAuthorization(context.connectionParams);

                if (!jwt) {
                  throw new UnprocessableEntityException(
                    'onConnect: JWT not found',
                  );
                }

                if (!authClient)
                  throw new RpcException('Error onConnect: AuthClientsService');

                const user = await authClient.authenticate(jwt);
                context.user = user;
              } catch (err) {
                new Logger().error(err);
                throw new UnauthorizedException('Error onConnect');
              }
            },
          },
        },
      }),
      imports: [AuthClientsModule],
      inject: [AuthClientsService],
```

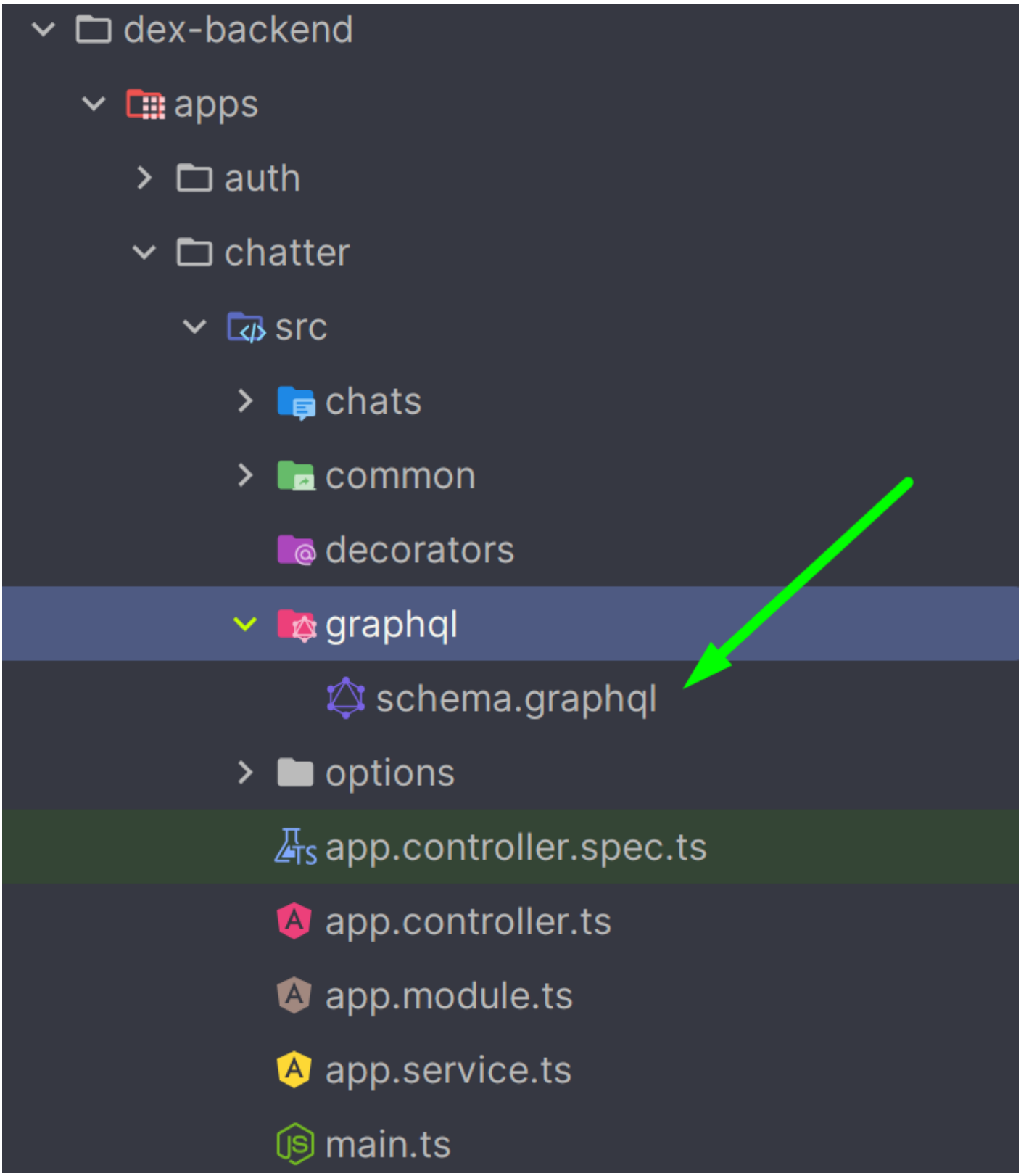
```
    }),
    GraphQLModule.forRootAsync<ApolloFederationDriverConfig>({
      driver: ApolloFederationDriver,
      useFactory: async (configService: ConfigService) => ({
        path: '/api/graphql',
        playground: true,
        cors: true,
        autoSchemaFile: {
          federation: 2,
        },
        cache: new InMemoryLRUCache({
          // MB
          maxSize:
            Math.pow(2, 20) *
            parseInt(configService.get('APOLLO_MEMORY_CACHE_MB')),
          // 5 minutes (in seconds)
          // convert string to int

          ttl: parseInt(configService.get('APOLLO_MEMORY_CACHE_TTL_SEC')),
        }),
        plugins: [
          ApolloServerPluginCacheControl({
            defaultMaxAge: 60,
          }),
        ],
      }),
      inject: [ConfigService],
    }),
```

Vấn đề là do thằng apollo federation nó commercialize vụ websocket với apollo cloud của nó nên nó restrict lại không support cho apollo federation có websocket.

Cách giải quyết là tạo một module graphql khác với driver bình thường nhưng type thì y chang thằng graphql federation.

Mình download schema graphql từ federation rồi save vào project folder dành diêng cho graphql module driver bình thường, đồng thời mình cũng đổi luôn url (path: '/api/graphql-normal',) , lẫn tắt luôn playground, ...



Sau khi setup xong hết thì bên frontend cần subscribe với endpoint websocket như trên ws://URL.COM/api/ws/graphql

Còn dạng REST apollo federation server thì cứ vào end: http://URL.COM/api/graphql



```
const { createProxyMiddleware } = require('http-proxy-middleware');

module.exports = function(app) {
  console.log('Setting up proxy');
  app.use(
    '/api/graphql',
    createProxyMiddleware({
      target: 'http://localhost:3003/graphql',
      changeOrigin: true,
    })
  );

    app.use(
        '/api/ws',
        createProxyMiddleware({
            target: 'http://localhost:3002/api/ws',
            changeOrigin: true,
            ws: true,
        })
    );


};
```

Revision #12
Created 2024-04-28 10:58:19 UTC by Son.KyLuat
Updated 2024-04-29 19:41:43 UTC by Son.KyLuat