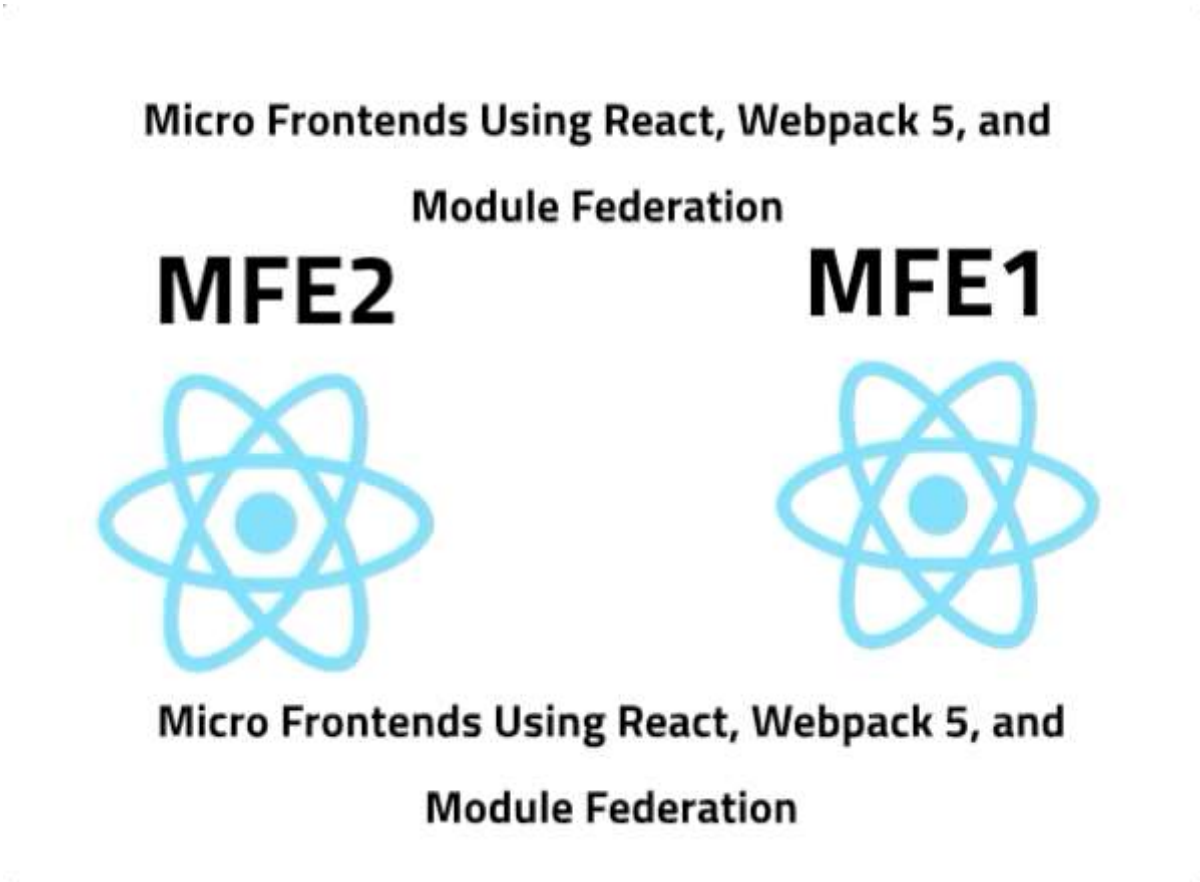
Published in Level Up Coding

Rany ElHousieny Follow

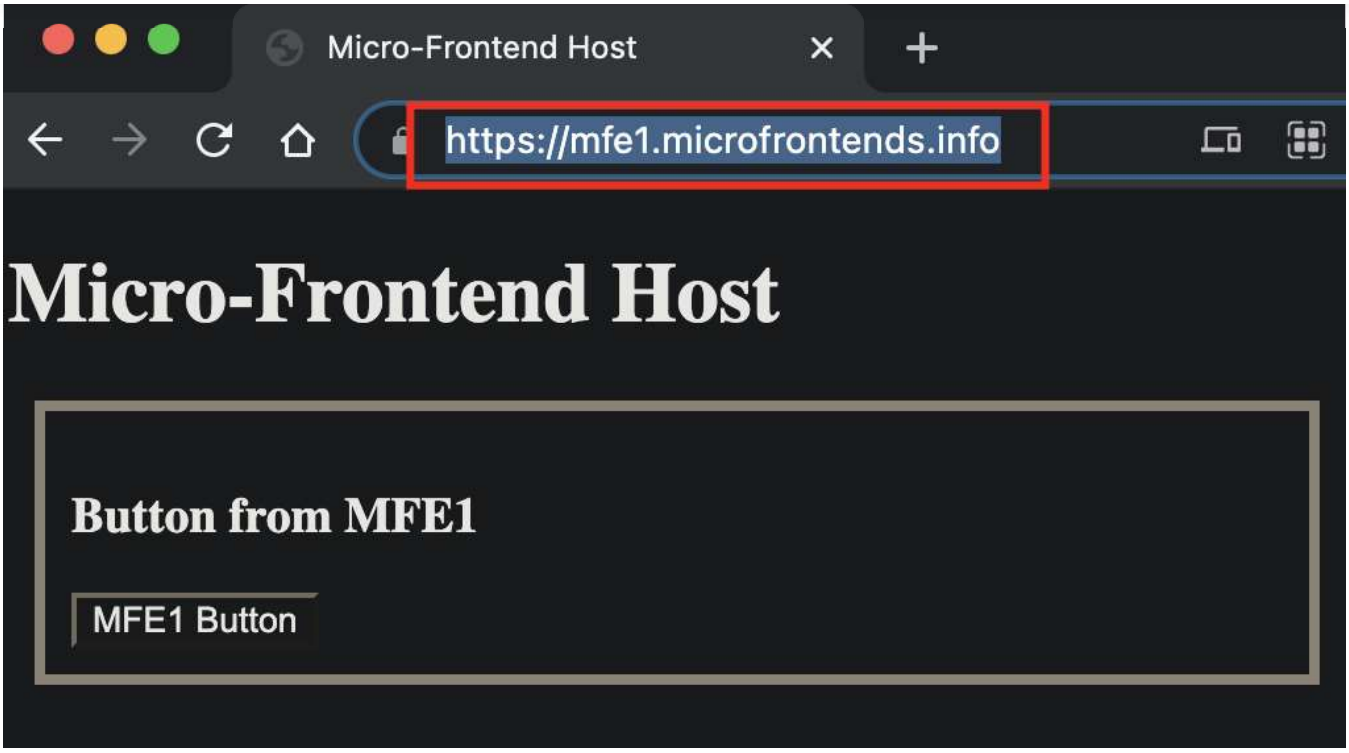
Feb 22, 2021 · 8 min read · Listen

Micro Frontends Step by Step Using React, Webpack 5, and Module Federation With Deployment to AWS



In this article, I will go step by step in creating two Micro Frontend React Components and render a Button component from one into the other.

At the end of this article, you will be able to implement a microfrontend component



If you are new to Microfrontend, you may start with the following article:

Micro-Frontends: What, why, and how

In my previous articles (links at the end of this article and here), I showed hands-on what are Micro Frontends and how...

www.linkedin.com

This hands-on example is a continuation of the following two articles where I explained how Micro Frontends work with Webpack5 and Module Federation. I wanted to give clean steps on React components without all the explanations. Please, refer to those articles for explanations

Micro-Front-Ends: Hands-On Project

Prepare your machine and install nodejs, if needed follow the article/video, below:

medium.com

This article is a continuation of the previous article

medium.com

The final project can be found at https://github.com/ranyelhousieny/react-microfrontends

Micro-Frontends: What, why, and how

In my previous articles (links at the end of this article and here), I showed hands-on what are Micro Frontends and how...

www.linkedin.com

Table of Contents:

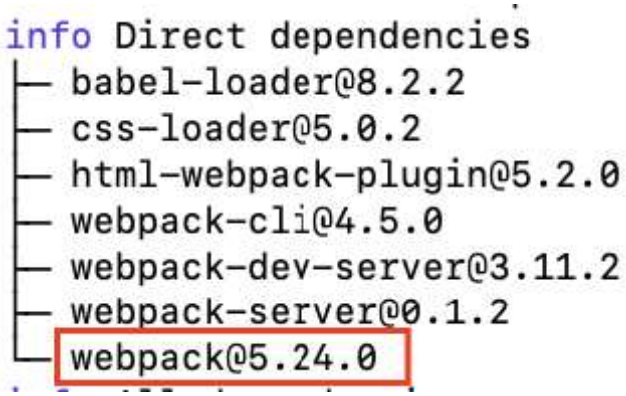
==========

Creating Microfrontends Projects Locally

1. Create two React apps and install dependencies

```
npx create-react-app mfe1

cd mfe1

yarn add webpack webpack-cli webpack-server html-webpack-plugin
babel-loader webpack-dev-server


npx create-react-app mfe2

cd mfe2
```


The nice thing I like about yarn is the report at the end. Check and make sure it installed Webpack 5 or up (you can learn how to install node and yarn here)



I am using yarn with this example because npm has been giving errors with the latest React release.

Note: While I used create-react-app here for an easier walkthrough, I will not depend on it at all and we will create our own webpack.config.js. You may learn how to set up a React project without CRA in this article

Create React App without create-react-app (CRA)

This article will show how to create a React app without using create-react-app (CRA). For more details on why we are...

www.linkedin.com

After this step, you will need to open visual studio code to complete the rest of the steps.

2. In both Apps, rename index.js to bootstrap.js

in *bootstrap.js*, Remove the imports to index.css and reportWebVitals and keep the file as shown below:

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';

ReactDOM.render(
  <App />,
  document.getElementById(
    'root'
  )
);
```

Here are the files on Github

https://github.com/ranyelhousieny/react-microfrontends/blob/main/mfe1/src/bootstrap.js

We do this step to make bootstrap load asynchronously. It is needed for MFE2 to wait for Webpack to fetch components from MFE1.

3. In both Apps, create a new index.js that has one line

```
import('./bootstrap');
```

Here is the file on Github


4. In MFE1, Create. a Button component

Create a file src/Button.js

Copy inside it this code (It is a very simple component)

```
import React from 'react';

const Button = () => (
  <button>MFE1 Button</button>
);

export default Button;
```

Code can be found at

ranyelhousieny/react-microfrontends

Contribute to ranyelhousieny/react-microfrontends development by

4.1. In MFE1, update App.js as follows

Code can be found at https://github.com/ranyelhousieny/react-microfrontends/blob/main/mfe1/src/App.js

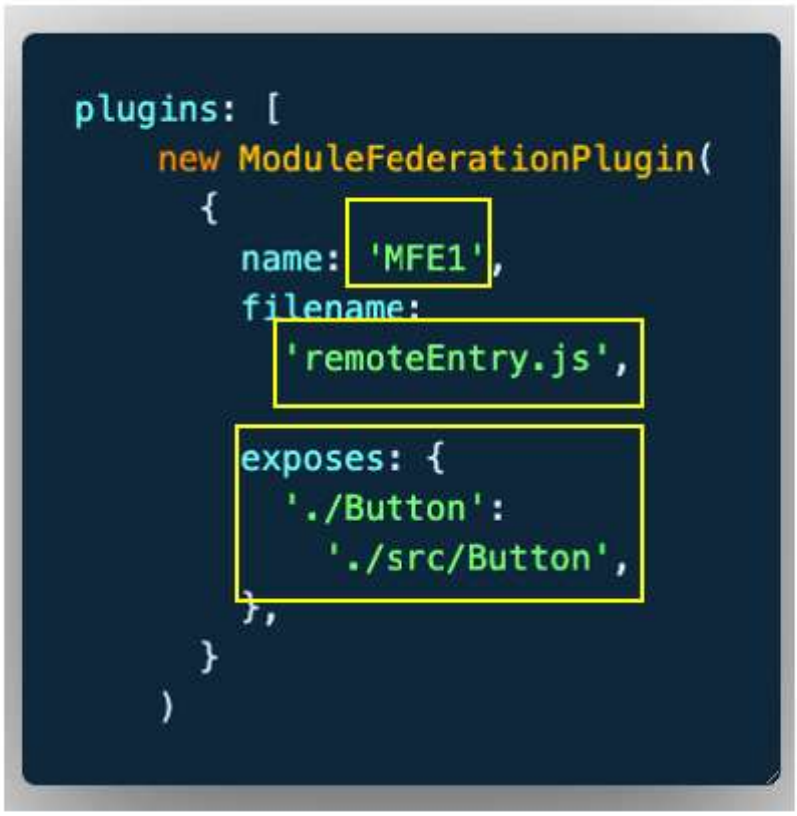
5. In MFE1, Create a Webpack.config.js on the root and put the following inside it

(details about this file and building it step by step can be found here https://www.linkedin.com/pulse/understanding-micro-frontends-webpack5-configurations-rany/ and the video https://youtu.be/AZDDIgJSKU0)

```
const HtmlWebpackPlugin = require('html-webpack-plugin');
const ModuleFederationPlugin =
require('webpack/lib/container/ModuleFederationPlugin');

module.exports = {
  mode: 'development',
  devServer: {
```


```
          /* The following line to ask babel
               to compile any file with extension
               .js */
          test: /\.js?$/,

          /* exclude node_modules directory from babel.
              Babel will not compile any files in this directory*/
          exclude: /node_modules/,

          // To Use babel Loader
          loader:
            'babel-loader',
          options: {
            presets: [
              '@babel/preset-env' /* to transfer any advansed ES to
ES5 */,
              '@babel/preset-react',
            ], // to compile react to ES5
          },
        },
      ],
    },

    plugins: [
      new ModuleFederationPlugin(
        {
          name: 'MFE1',
          filename:
            'remoteEntry.js',

          exposes: {
            './Button':
              './src/Button',
          },
        }
      ),
      new HtmlWebpackPlugin({
        template:
          './public/index.html',
      }),
    ],
  };
```

https://github.com/ranyelhousieny/react-microfrontends/blob/main/mfe1/webpack.config.js

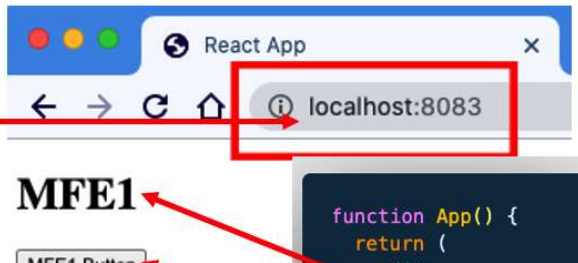



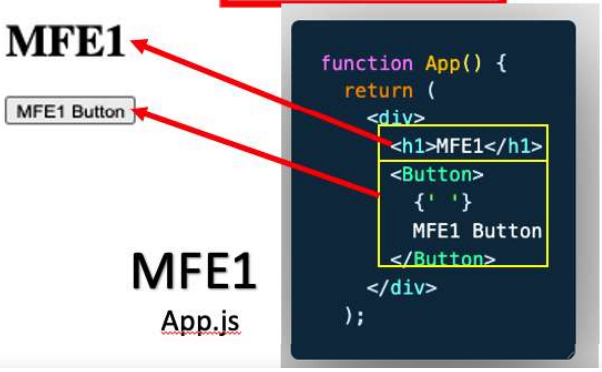
MFE1

webpack.config.js

MFE1

webpack.config.js





MFE1

App.js

In the previous two articles, I demonstrated how to build Micro-Frontends and deploy them to AWS. During the process, I...

www.linkedin.com

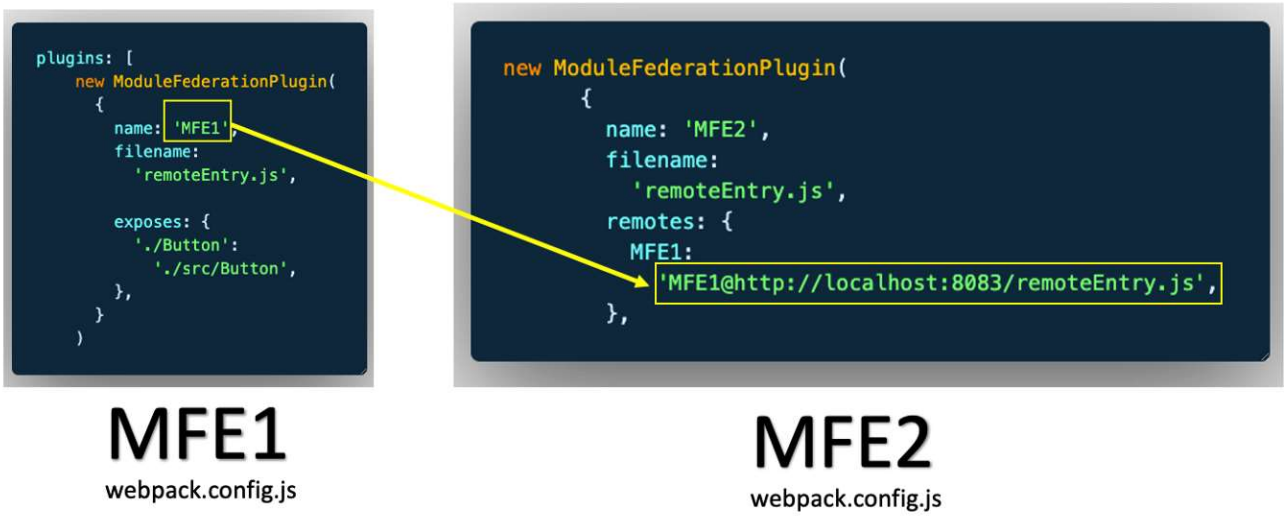
6. In MFE2, Create a Webpack.config.js on the root and put the following inside it

more details about this file here https://www.linkedin.com/pulse/understanding-micro-frontends-webpack5-module-step-rany

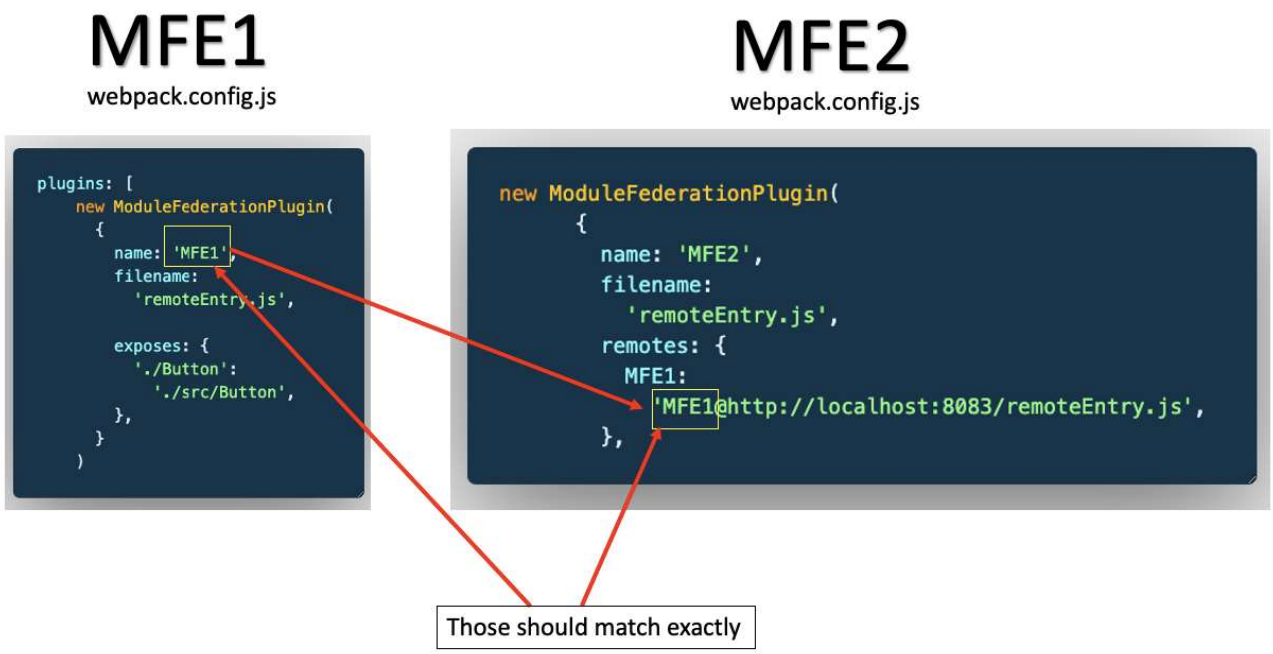
```
const HtmlWebpackPlugin = require('html-webpack-plugin');
const ModuleFederationPlugin =
require('webpack/lib/container/ModuleFederationPlugin');

module.exports = {
  mode: 'development',
  devServer: {
    port: 8082,
  },
  module: {
    rules: [
      {
        /* The following line to ask babel
             to compile any file with extension
             .js */
        test: /\.js?$/,

        /* exclude node_modules directory from babel.
            Babel will not compile any files in this directory*/
        exclude: /node_modules/,

        // To Use babel Loader
        loader:
```


```
            ], // to compile react to ES5
          },
        },
      ],
    },

    plugins: [
      new ModuleFederationPlugin(
        {
          name: 'MFE2',
          filename:
            'remoteEntry.js',
          remotes: {
            MFE1:
              'MFE1@http://localhost:8083/remoteEntry.js',
          },
        }
      ),
      new HtmlWebpackPlugin({
        template:
          './public/index.html',
      }),
    ],
  };
```

https://github.com/ranyelhousieny/react-microfrontends/blob/main/mfe2/webpack.config.js



Here we just said that MFE1 exposed components can be available in MFE2 remotely through http://localhost:8083/remoteEntry.js

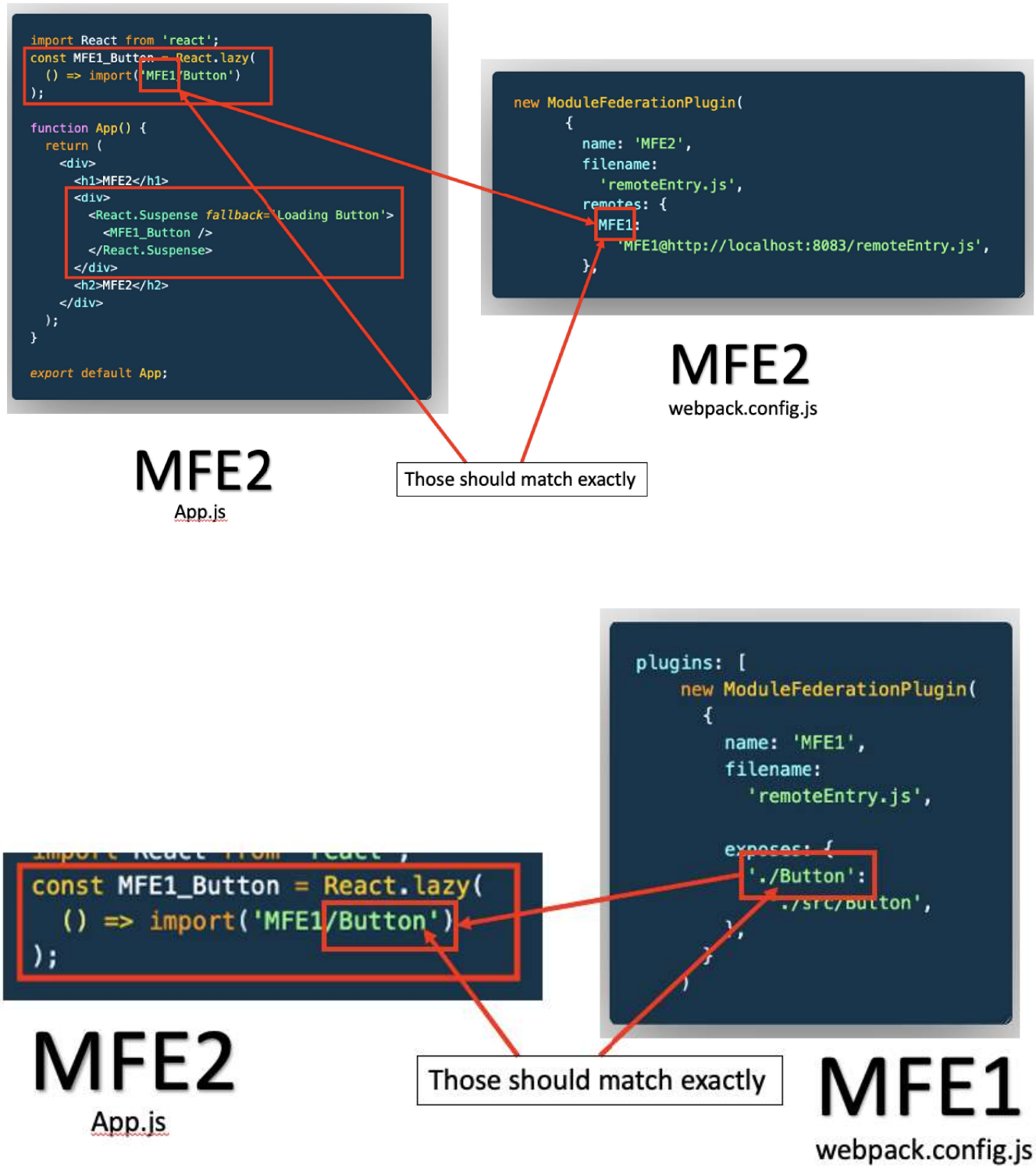


7. In MFE2 src/App.js import Button (Lazy import)

```
const MFE1_Button = React.lazy(
  () => import('MFE1/Button')
);
```

Add it to the App as follows

```
function App() {
  return (
    <div>
      <h1>MFE2</h1>
      <div>
        <React.Suspense fallback='Loading Button'>
          <MFE1_Button />
        </React.Suspense>
      </div>
      <h2>MFE2</h2>
    </div>
  );
}
```



8. Let's run both apps

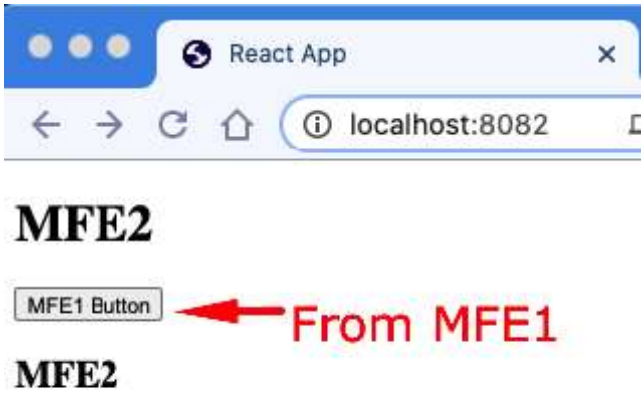
open a terminal inside MFE1 directory and run

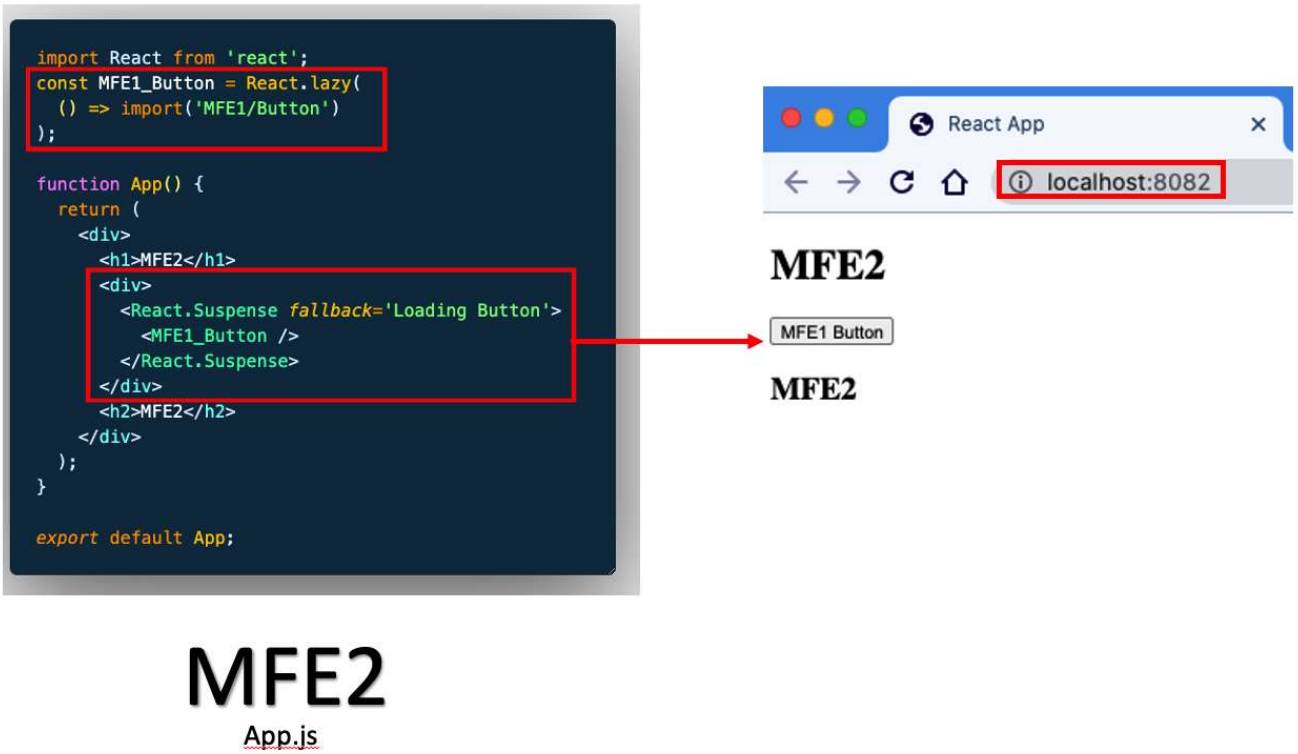
```
yarn webpack serve
```

Now open the browser and navigate to

http://localhost:8082/

You should see the following



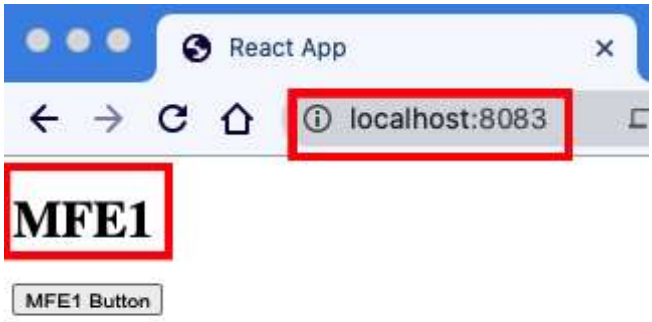


MFE2

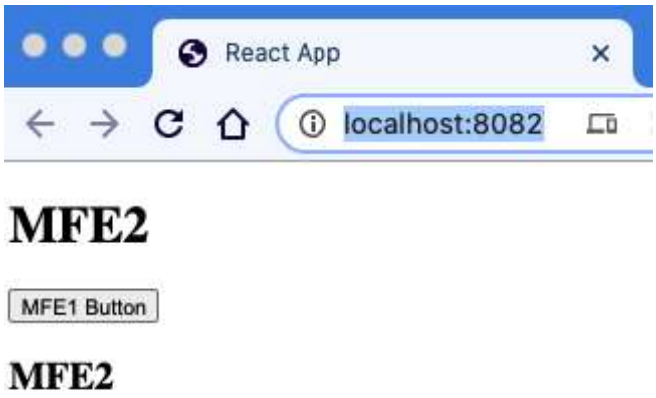
App.js

==========

1. MFE1 on http://localhost:8083/ [We decided this port in the configuration]



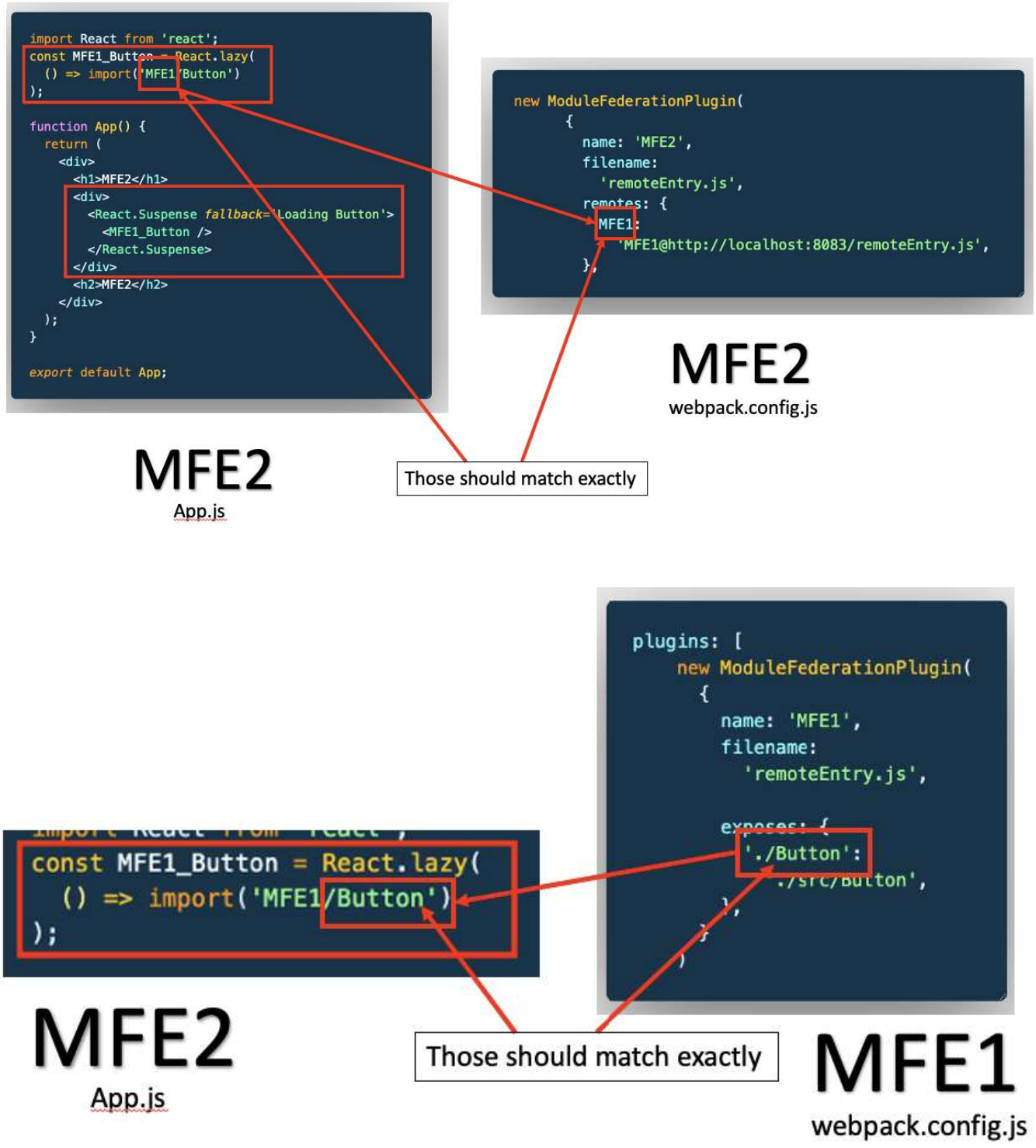
1. MFE2 on http://localhost:8082/



Second: Using Module Federa exposed the Button component from MFE1 site (h 8083/) and imported it into MFE2

1. We added a remote for MFE1 in MFE2

2. We imported the exposed component from MFE1 into App.js in MFE2



3. Finally we Lazy rendered the Button inside MFE1

```
import React from 'react';
const MFE1_Button = React.lazy(
  () => import('MFE1/Button')
);

function App() {
  return (
    <div>
      <h1>MFE2</h1>
      <div>
        <React.Suspense fallback='Loading Button'>
          <MFE1_Button />
        </React.Suspense>
      </div>
      <h2>MFE2</h2>
    </div>
  );
}

export default App;
```

Now, since we have tested and understood how Webpack works locally, let's deploy our apps to AWS in the following article :

Deploying Micro Frontends to AWS Step by Step Using Gitlab, React, Webpack 5, and Module Federation

In my previous article (https://levelup.gitconnected.

www.linkedin.com

==========

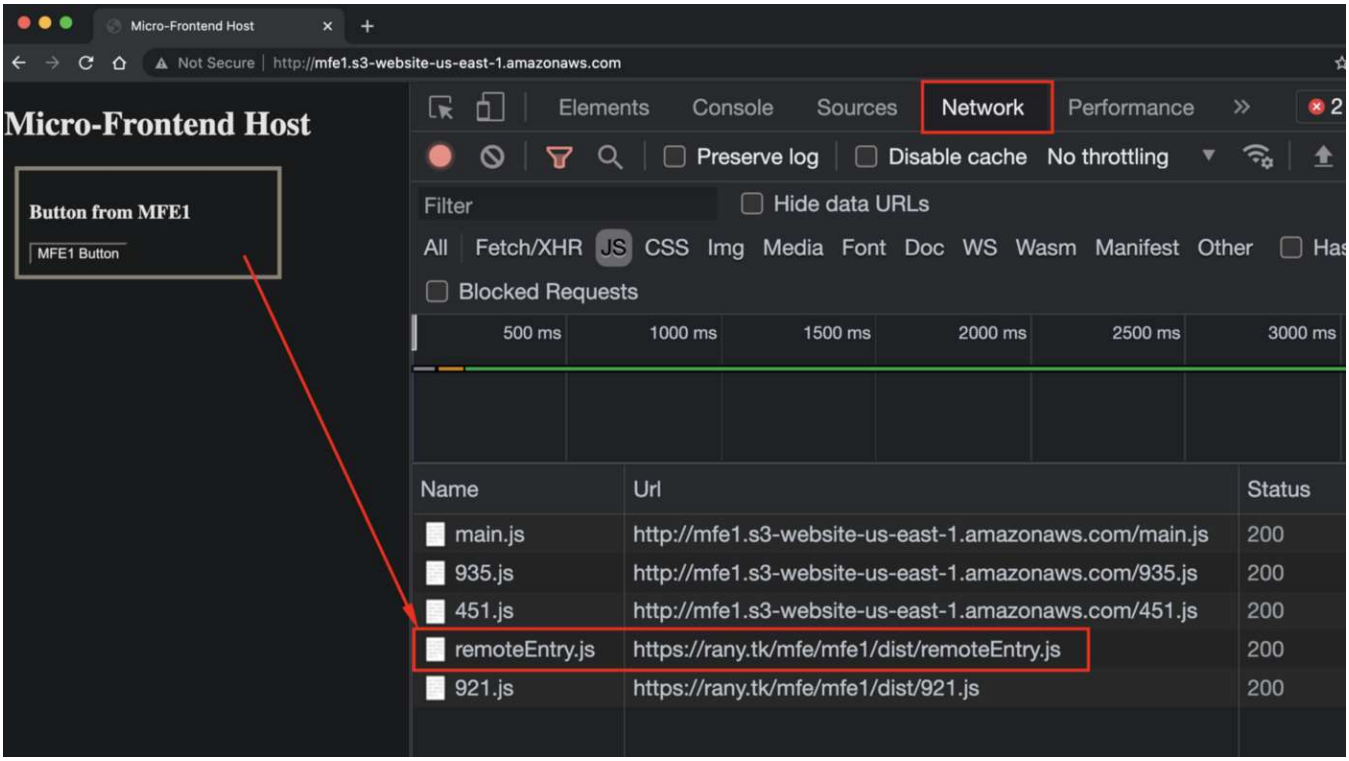
Deploying Microfrontends

After we implemented and tested the Microfrontends locally, let's deploy them to AWS. Follow the steps in the following article to deploy to AWS:

Deploying Micro Frontends to AWS Step by Step Using React, Webpack 5, and Module Federation

In my previous article (https://levelup.gitconnected.

You can browse the deployed version at http://mfe1.s3-website-us-east-1.amazonaws.com/



If you click inspect and then Network as shown above, you can see the call to get https://rany.tk/mfe/mfe1/dist/remoteEntry.js

Now, we created a host and added a microfrontend inside it then we deployed to AWS.

==========

Add AWS CloudFront

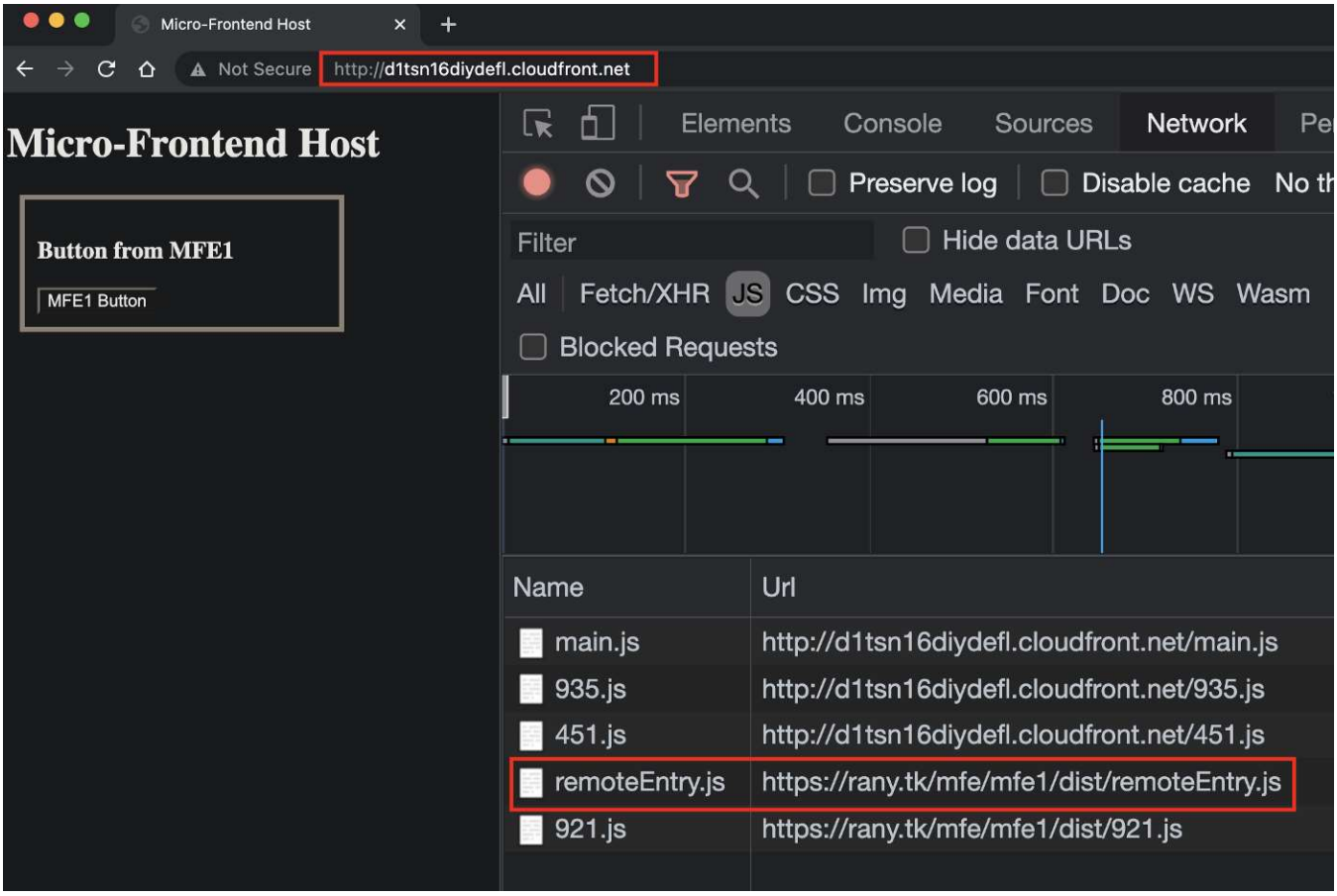
after deploying to S3, let's add CloudFront for caching, security, and to add a domain name. This is a very important step when deploying a website. Follow the following article to add CloudFront

Adding CloudFront to Web-Enabled AWS S3 Bucket

This article builds over the previous article (https://www.linkedin.

www.linkedin.com

Now, you can access MFE2 from the following Link
http://d1tsn16diydefl.cloudfront.net/

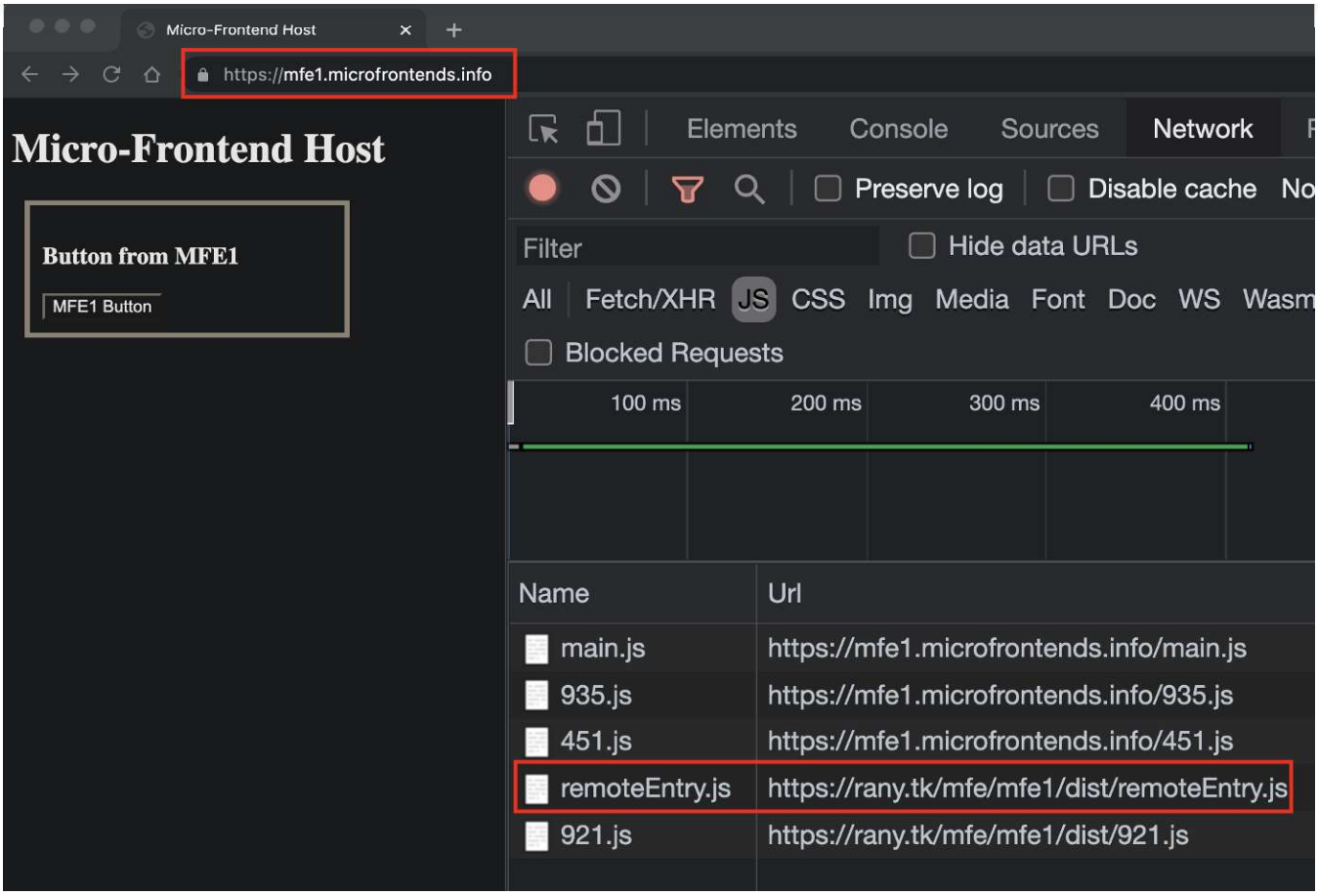


==========

Purchasing and Adding a Domain name:

Now, it is very hard to memorize the previous link http://d1tsn16diydefl.cloudfront.net/. In the following article, I will go step by step to purchase microfrontends.info and render the site to https://mfe1.microfrontends.info/

https://www.linkedin.com/pulse/adding-domain-certificate-website-aws-s3-web-enabled-rany



Other Article By Rany ElHousieny:

Rany ElHousieny , PhDABD - SENIOR MANAGER SOFTWARE ENGINEERING - Zulily | LinkedIn

AWS Solutions Architect Certified®, ***Microservices*** Solutions Architect...

www.linkedin.com

Software Engineering | LinkedIn

Rany ElHousieny , PhDABD | Sharing My Software Engineering Experience in Different Areas, Front End, Back End, Cloud ...

www.linkedin.com

Micro-Front-Ends: Hands-On Project

Prepare your machine and install nodejs, if needed follow the article/video, below:

medium.com

Micro Frontends Hands-On Example Using React, Webpack 5, and Module Federation: Adding a third...

This article is a continuation of the previous article

medium.com

I published originally on LinkedIn

Micro Frontends Step by Step Using React, Webpack 5, and Module Federation

In this article, I will go step by step in creating two Micro Frontend React Components and render a Button component...

www.linkedin.com

Sign up for Top Stories

By Level Up Coding

A monthly summary of the best stories shared in Level Up Coding Take a look.