**Deployment Using Docker**

Step-by-step example of two micro frontend apps with Webpack Module Federation, each deployed using Docker. The setup includes:

* **One Host app that loads a component from a Remote app.**
* **Both apps are React-based with minimal shared code.**
* **Docker setup for containerization.**

Host app remotely imports the exposed Widget component from the Remote app using Module Federation.

* The address **host.docker.internal:3001** is used inside the Docker container to access the host machine’s localhost on Windows/macOS. Adjust if on Linux or different network setup.
* Both apps use React 18 and latest Webpack 5 features.
* This is a minimal, working example to build upon for larger micro frontend architectures.

1. Remote App

Folder structure

remote-app/

│

├── src/

│ ├── Widget.js

│ └── index.js

│

├── package.json

├── webpack.config.js

├── Dockerfile

└── .babelrc

**Source files**

**Package.json**

{

"name": "remote-app",

"version": "1.0.0",

"private": true,

"scripts": {

"start": "webpack serve --config webpack.config.js --mode development",

"build": "webpack --config webpack.config.js --mode production"

},

"dependencies": {

"react": "^18.2.0",

"react-dom": "^18.2.0"

},

"devDependencies": {

"@babel/core": "^7.21.0",

"@babel/preset-react": "^7.21.0",

"babel-loader": "^9.1.2",

"webpack": "^5.88.0",

"webpack-cli": "^5.1.2",

"webpack-dev-server": "^4.15.0"

}

}

**.babelrc**

{

"presets": ["@babel/preset-react"]

**}**

**webpack.config.js**

const path = require('path');

const { ModuleFederationPlugin } = require('webpack').container;

module.exports = {

  entry: './src/index',

  mode: 'development',

  devServer: {

    port: 3001,

    static: path.resolve(\_\_dirname, 'dist'),

    headers: { 'Access-Control-Allow-Origin': '\*' },

    allowedHosts: 'all',  // allows all hosts

  },

  output: {

    publicPath: 'auto',

    clean: true,

    path: path.resolve(\_\_dirname, 'dist'),

  },

  module: {

    rules: [

      {

        test: /\.jsx?$/,

        loader: 'babel-loader',

        exclude: /node\_modules/

      }

    ]

  },

  plugins: [

    new ModuleFederationPlugin({

      name: 'remoteApp',

      filename: 'remoteEntry.js',

      exposes: {

        './Widget': './src/Widget'

      },

      shared: {

        react: {

          singleton: true,

          requiredVersion: '^18.2.0',

          eager: false,

        },

        'react-dom': {

          singleton: true,

          requiredVersion: '^18.2.0',

          eager: false,

        },

      }

    })

  ],

  resolve: {

    extensions: ['.js', '.jsx']

  }

};

**src/Widget.js**

import React from 'react';

const Widget = () => {

return <div style={{ padding: 20, backgroundColor: '#def' }}>Hello from Remote Widget!</div>;

};

export default Widget;

**src/index.js**

**// Entry point can be empty or minimal for remote app**

**Dockerfile**

FROM node:18

WORKDIR /app

COPY package\*.json ./

RUN npm ci

COPY . .

RUN npm run build

EXPOSE 3001

CMD ["npx", "webpack", "serve", "--config", "webpack.config.js", "--mode", "development", "--host", "0.0.0.0"]

**2. Host App**

**Folder structure**

**host-app/**

**│**

**├── src/**

**│ └── index.js**

**│**

**├── public/**

**│ └── index.html**

**├── package.json**

**├── webpack.config.js**

**├── Dockerfile**

**└── .babelrc**

package.json

**{**

**"name": "host-app",**

**"version": "1.0.0",**

**"private": true,**

**"scripts": {**

**"start": "webpack serve --config webpack.config.js --mode development",**

**"build": "webpack --config webpack.config.js --mode production"**

**},**

**"dependencies": {**

**"react": "18.2.0",**

**"react-dom": "18.2.0"**

**},**

**"devDependencies": {**

**"@babel/core": "^7.21.0",**

**"@babel/preset-react": "^7.21.0",**

**"babel-loader": "^9.1.2",**

**"html-webpack-plugin": "^5.5.0",**

**"webpack": "^5.88.0",**

**"webpack-cli": "^5.1.2",**

**"webpack-dev-server": "^4.15.0"**

**}**

**}**

.babelrc

{

"presets": ["@babel/preset-react"]

}

webpack.config.js

const path = require('path');

const HtmlWebpackPlugin = require('html-webpack-plugin');

const { ModuleFederationPlugin } = require('webpack').container;

module.exports = {

  entry: './src/index',

  mode: 'development',

  devServer: {

    port: 3000,

    static: path.resolve(\_\_dirname, 'dist'),

    headers: { 'Access-Control-Allow-Origin': '\*' },

    allowedHosts: 'all',  // allows all hosts

  },

  output: {

    publicPath: 'auto',

    clean: true,

    path: path.resolve(\_\_dirname, 'dist'),

  },

  module: {

    rules: [

      {

        test: /\.jsx?$/,

        loader: 'babel-loader',

        exclude: /node\_modules/

      }

    ]

  },

  plugins: [

    new ModuleFederationPlugin({

      name: 'hostApp',

      remotes: {

        remoteApp: 'remoteApp@http://host.docker.internal:3001/remoteEntry.js'

      },

      shared: {

        react: {

          singleton: true,

          requiredVersion: '^18.2.0',

          eager: true,

        },

        'react-dom': {

          singleton: true,

          requiredVersion: '^18.2.0',

          eager: true,

        },

      }

    }),

    new HtmlWebpackPlugin({

      template: './public/index.html'

    })

  ],

  resolve: {

    extensions: ['.js', '.jsx']

  }

};

src/index.js

import React, { Suspense, lazy } from 'react';

import ReactDOM from 'react-dom/client';

const RemoteWidget = lazy(() => import('remoteApp/Widget'));

const App = () => (

  <div>

    <h1>Host App</h1>

    <Suspense fallback={<div>Loading Remote Widget...</div>}>

      <RemoteWidget />

    </Suspense>

  </div>

);

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(<App />);

public/index.html

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8" />

  <meta name="viewport" content="width=device-width,initial-scale=1" />

  <title>Host App</title>

</head>

<body>

  <div id="root"></div>

</body>

</html>

Dockerfile

FROM node:18

WORKDIR /app

COPY package\*.json ./

RUN npm ci

COPY . .

RUN npm run build

EXPOSE 3000

CMD ["npx", "webpack", "serve", "--config", "webpack.config.js", "--mode", "development", "--host", "0.0.0.0"]

**3. Run Locally With Docker**

* Build Remote App image:

cd remote-app

npm i

docker build -t mfe-remote .

* Run Remote App container:

bash

docker run -d -p 3001:3001 mfe-remote

* Build Host App image:

bash

cd host-app

npm i

docker build -t mfe-host .

* Run Host App container:

bash

docker run -d -p 3000:3000 mfe-host

**Summary:**

* **Run npm install on your local setup to generate package-lock.json.**
* **Remember to include this file when building Docker images.**
* **This will allow npm ci to run successfully in your Docker build process.**

**Open http://localhost:3000 in your browser. The Host app loads and renders the remote Widget from the Remote app.**

**Notes**

* Host app remotely imports the exposed Widget component from the Remote app using Module Federation.
* The address host.docker.internal:3001 is used inside the Docker container to access the host machine’s localhost on Windows/macOS. Adjust if on Linux or different network setup.
* Both apps use React 18 and latest Webpack 5 features.
* This is a minimal, working example to build upon for larger microfrontend architectures.