PostCodes Deployment Steps

Step 1: Create Deployable of PostCodes.WebAPI

Source Code: https://github.com/shashankuttekar/postcodeswebapi

Instruction to create Deployable

- Open PostCodes.WebAPI.sln file in Visual Studio.
- From project PostCodes.WebAPI right click and build the project. During the build process NuGet will download all dependencies packages. After successful build you can see deployable files in ~\PostCodes.WebAPI\bin\Debug\netcoreapp3.1\ Folder.
- Create a Zip of all files. E.g. PostCodes.WebAPI.Artifact.zip
- This zip should contain direct files not folders.

Step 2: Create Deployable of PostCodes.Frontend

Source Code: https://github.com/shashankuttekar/postcodesfrontend

Instruction to install npm packages, run applications and build deployment.
We will create a production deployment package once REST API service is live in step 3 because we need API URL to set in *src/AppConfig.json*

- Open postcodesfrontend folder in VSCode
- Open terminal/command prompt and run Command ***npm install***
- Use ***npm run start*** to run application

Follow below steps after ***step 3***

- Open src/AppConfig.json and replace postcodes_base_url with your API URL {https://Your URL}/Prod
- Run ***npm run build*** to create a production deployable package.
- It will create deployable files in the ***build*** folder in project root directly.

Step 3: AWS Deployment Steps

- Login to AWS Console
- Create new bucket for RestAPI Service e.g <postcodes-webapi-services>
- Upload zip file from Step 1
- Upload PostCodesAWSFormation.yml in bucket
- Get URL (**Object URL**) of PostCodeAWSFormation.yml file from properties.
- Open cloudshell

Change ParameterValues in parameters.json

Key	Value
ReactArtifactBucketName	Set bucket name to be created for react application deployment
CoreArtifactBucketName	Set bucket name created for RestAPIService in above step
ZipFileNameForCoreAPIArtifact	Set zip file name which is created in step 1

Run the following command (keep parameters.json in same folder or give complete path for these files)
Change template-url of PostCodesAWSFormation.yml

```
aws cloudformation create-stack --stack-name PostCodeStack --template-url
"http://amazopn.path/PostCodesAWSFormation.yml" --parameters "file://parameters.json" -
-capabilities CAPABILITY_IAM
```

- Once above command executed then open PostCodesWebAPIStack Stack in AWS cloud formation and check progress till it completed
- Once stack created Successfully click on Stack Name and go to output tab
- You will get API URL
- Browse that URL . You will get Message in browser *Welcome to running ASP.NET Core on AWS Lambda*
- To deploy the react application, please follow the remaining steps of Step 2.
- Go to the S3Buckets, Refresh AWS Window.
- You can see a new bucket is created with publicly accessible rights. (You have provided name for bucket in Json file => ReactArtifactBucketName)
- Open That bucket and Upload artifacts (all files and folder) of React App. (To create artifacts, follow the remaining steps of Step 2.)

- Once uploaded, go to the properties of that Bucket and scroll down to the “Static website hosting” section. You will get a bucket website endpoint, browse that URL.
- To disable public access to S3, create cloud front distribution to frontend S3 bucket. This is a one time activity. Note, cloud front changes took some time hours to reflect new changes.
- Application URI : https://d2eenj31ejnt8z.cloudfront.net/