Available online www.jsaer.com

***Journal of Scientific and Engineering Research*, 2021, 8(4):218-227**

Research Article

ISSN: 2394-2630
CODEN(USA): JSERBR

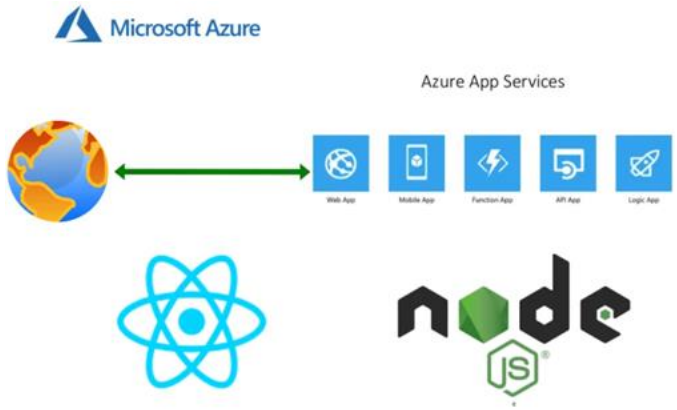
Optimizing Deployment: React with NodeJS Backend on Azure App Services

Bhargav Bachina

Abstract Deploying applications on managed platforms like Azure App Service offers convenience and scalability. Among its offerings, a Function App, Web App, or Logic App can be created under Azure App Services. For applications combining React with a Node.js backend, various deployment methods are available, with Azure App Service being a prominent choice. This paper explores the deployment of a React application with a Node.js backend on the Node.js runtime within Azure App Service. Beginning with prerequisites, it provides an example project and outlines the steps for creating the App Service with Node.js runtime, packaging the project, and deploying it using Local Git on App Service. Additionally, the paper includes demonstrations on monitoring logs and managing repository updates, concluding with a summary of key insights and conclusions drawn from the deployment process.

Keywords React, Nodejs, Programming, Cloud Computing, Azure

1. Introduction



If you're considering deploying your application on a managed platform with customizable runtimes, Azure App Service emerges as a prime option. Within Azure App Services, you have the flexibility to create various types of applications, including function apps, web apps, and logic apps. For React applications paired with Node.js backends, multiple deployment methods are available, with Azure App Service being a notable choice.

In this article, we'll focus on deploying a React application with a Node.js backend on the Node.js runtime within Azure App Service. The process begins with packaging our application and then pushing the local Git repository to Azure App Services.

- Prerequisites
- Example Project

- Creating the App Service with NodeJS Runtime
- Package the Project
- Deploy With Local Git on App Service
- Demo
- How To See the Logs
- How To Clone and update the repo
- Summary
- Conclusion

2. Prerequisites

If you are new to React, please go through the below link on how to develop and build the React project with NodeJS backend.

- How To Develop and Build React App with NodeJS (https://medium.com/bb-tutorials-and-thoughts/how-to-develop-and-build-react-app-with-nodejs-bc06fa1c18f3)

The other prerequisites to this post are that how to package your React with NodeJS Backend for the deployment. Here is the link to go through and understand it better.

- Packaging Your React App With NodeJS Backend For Production (https://medium.com/bb-tutorials-and-thoughts/packaging-your-react-app-with-nodejs-backend-for-production-7ddae2b84f1b)

Microsoft Azure Account

You should have a Microsoft Azure Account. You can get a free account for one year. You should see the below screen after you log in.

- Azure Account (https://azure.microsoft.com/en-us/free/)

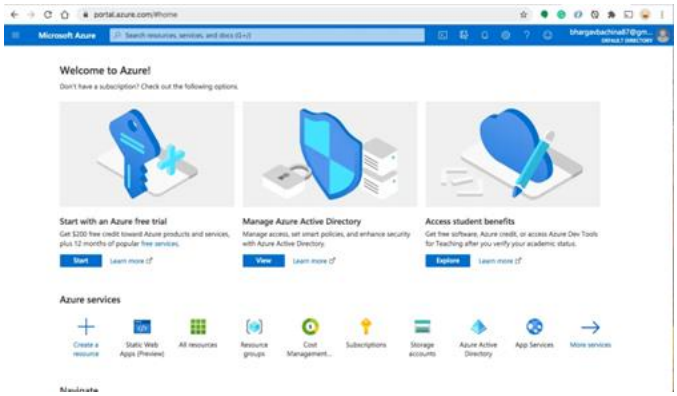


Figure 1: Azure Home Screen

You need to create a subscription for your account. The most common is Pay as You Go subscription.

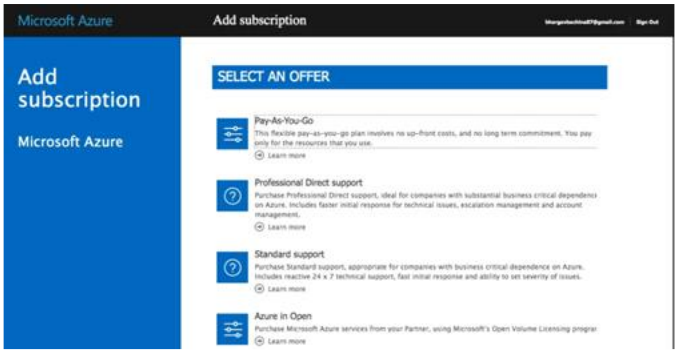


Figure 2: Subscription Offers

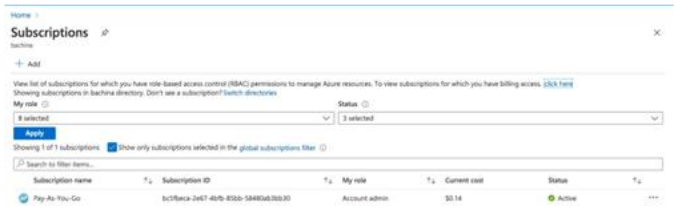


Figure 3: Pay-As-You-Go Subscription

You need a subscription to be associated with your tenant so that all the cost is billed to this subscription.

3. Example Project

This is a simple project which demonstrates developing and running React application with NodeJS. We have a simple app in which we can add users, count, and display them at the side, and retrieve them whenever you want.
https://miro.medium.com/v2/resize:fit:1400/0*6qSTrDIbiqKa888H.gif
As you add users we are making an API call to the NodeJS server to store them and get the same data from the server when we retrieve them. You can see network calls in the following video.
https://miro.medium.com/v2/resize:fit:1400/0*9E35yzyet5ECP7B1.gif
Here is a GitHub link to this project. You can clone it and run it on your machine.
// clone the project git clone https://github.com/bbachi/react-nodejs-app-services.git
// strat the apicd api
npm install
npm run dev
// start the react appcd my-app
npm install
npm start

4. Creating the App Service with NodeJS Runtime

You need to create a web app service before deploying our application with Local git. Let's do it with the portal. You need to go to the App Services and create a web app with NodeJS runtime as below. Make sure you select the right resource group.

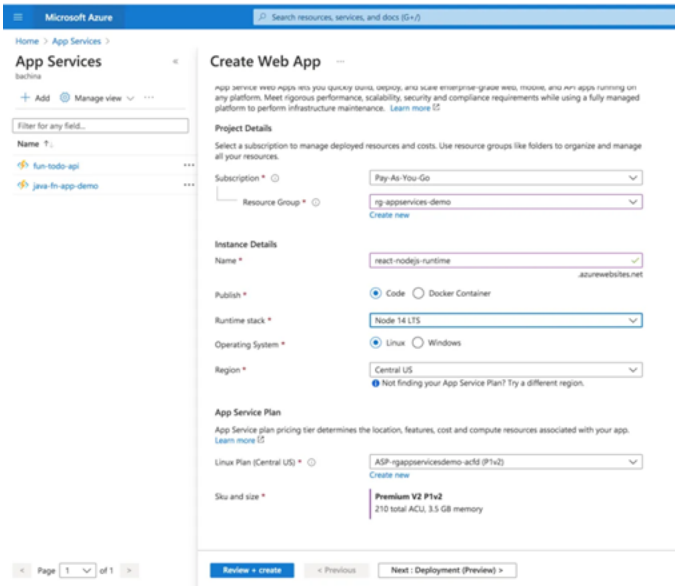


Figure 4: Creating a web app with NodeJS Runtime

On the next screen, you have a choice to enable or disable the Continuous Deployment. If you enable it you have to give your GitHub repository you should disable it for this post.

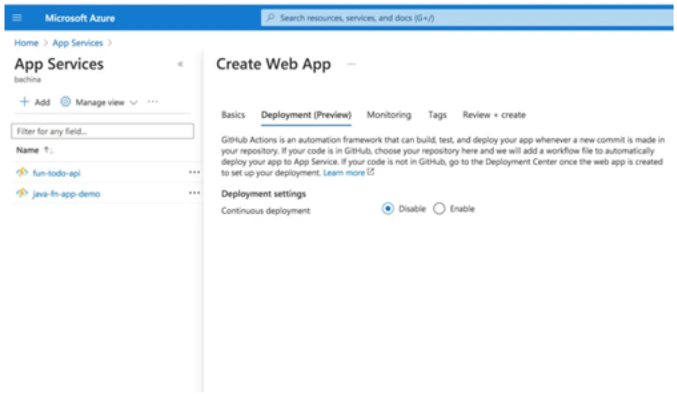


Figure 5: Deployment

Finally, review and create the App Service.

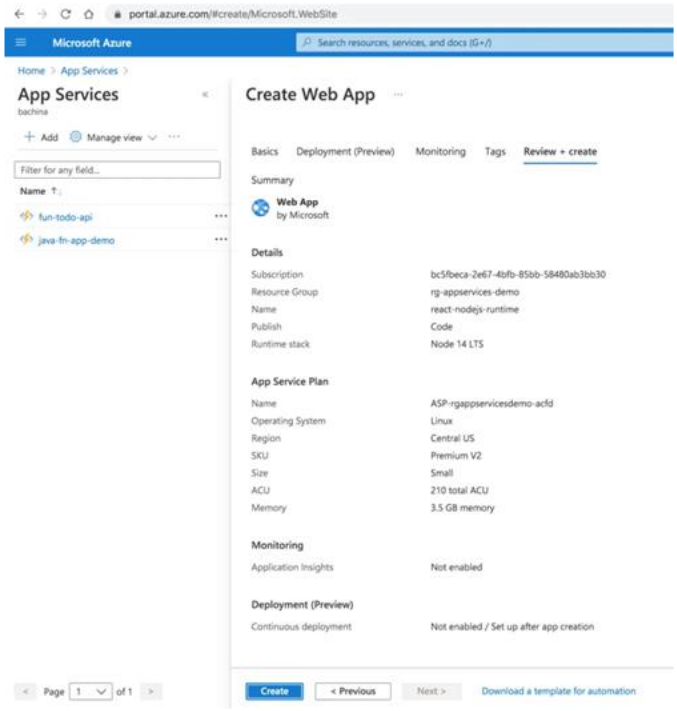


Figure 6: Review and Create

Once you click on the create button you would see the deployment successful screen.

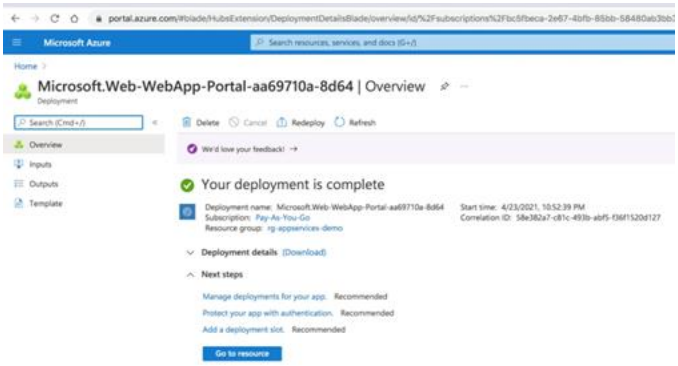


Figure 7: Deployment Successful

Click on the Go to resource button and find the URL below.

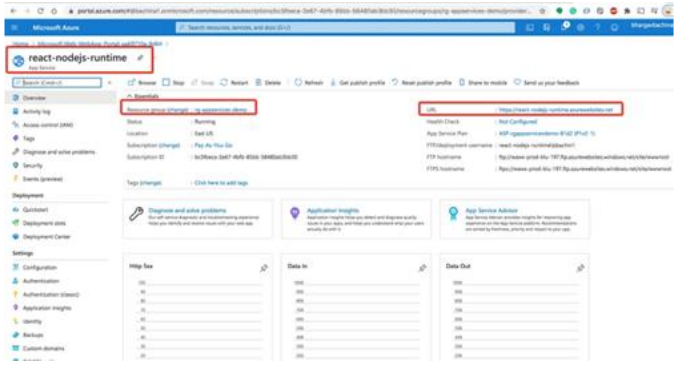


Figure 8: App running on App services.

5. Package the Project

We don't have to package the project for this deployment. If you are maintaining two package.json files for React and NodeJS respectively as below you need to build the Vue project.

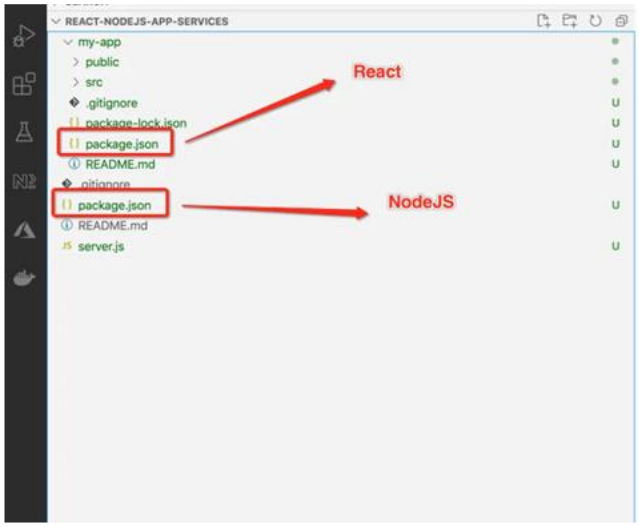


Figure 9: Separate Package.json

A. React

First, run the following command by changing into the my-app directory to build the React project.

npm run build

It compiles the project and puts all the build assets under the folder called ***build***.

```
/Users/bhargavbachina/Projects/React/react-nodejs-app-services/my-app
Bhargavs-MacBook-Pro:my-app bhargavbachina$ npm run build

> my-app@0.1.0 build /Users/bhargavbachina/Projects/React/react-nodejs-app-services/my-app
> react-scripts build

Creating an optimized production build...
Browserslist: caniuse-lite is outdated. Please run:
npx browserslist@latest --update-db
Compiled with warnings.

./src/components/DisplayBoard.js
  Line 5:11:  'headerStyle' is assigned a value but never used  no-unused-vars

Search for the keywords to learn more about each warning.
To ignore, add // eslint-disable-next-line to the line before.

File sizes after gzip:

  42.39 KB  build/static/js/2.67710b93.chunk.js
  22.47 KB  build/static/css/2.af3c1da9.chunk.css
  1.95 KB   build/static/js/main.e203d939.chunk.js
  770 B     build/static/js/runtime-main.7c0ae09a.js
  399 B     build/static/css/main.14ff7981.chunk.css

The project was built assuming it is hosted at /.
You can control this with the homepage field in your package.json.

The build folder is ready to be deployed.
You may serve it with a static server:

  npm install -g serve
  serve -s build

Find out more about deployment here:

  bit.ly/CRA-deploy
```

Figure 10: Building the Project

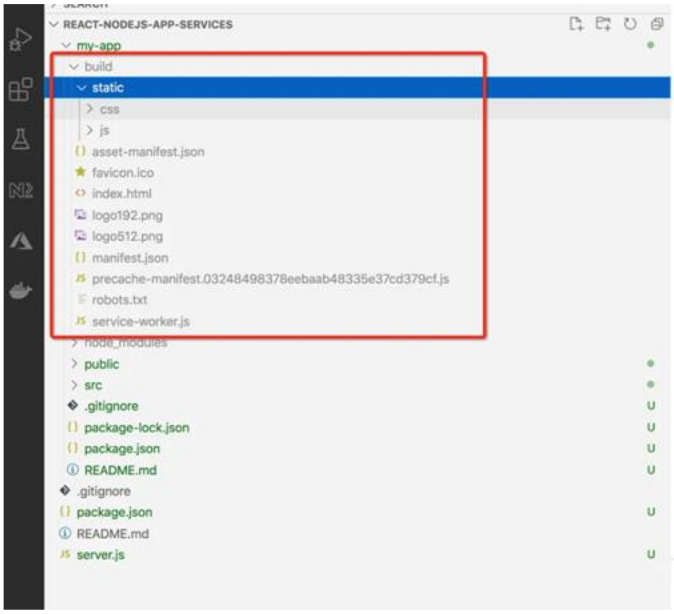


Figure 11: npm run build

B. NodeJS Server

We are deploying the whole project on nodejs runtime and when it comes to the NodeJS server you should run on port ***8080 and you need to serve React assets from the build we generated in the above step.***

https://gist.github.com/bbachi/2066119acd68b98f2876241c525d064a#file-server-js

We need to use express.static to serve React static assets as above. Make sure you have the start command in the package.json as below. App services run the command npm start by default.

https://gist.github.com/bbachi/e32f8c76a7898d2731da3b73cc773f1b#file-package-json

6. Deploy with Local Git on App Service

We have created the app services and went through the structure of the project. It's time to deploy this with the local git server. If you go to the deployment center of the app service that we created above. We have three options, one of them is local git.

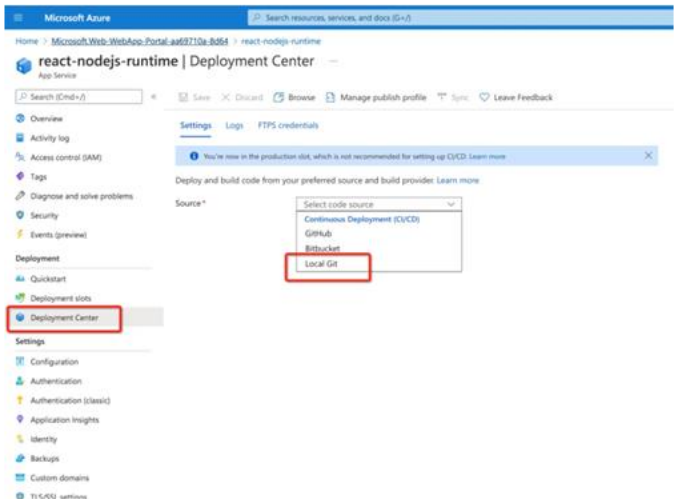


Figure 12: Local Git

First, you need to set the username and password to push the code.

az webapp deployment user set --user-name <username> --password <password>

Since you already created an App Service, and we need to run the following command to add it to the existing app service.

az webapp deployment source config-local-git --name react-nodejs-runtime --resource-group rg-appservices-demo

If you want to create an App Service from the AZ CLI there is another command for that. You can go through the Azure Docs for that.

```
Bhargavs-MacBook-Pro:react-nodejs-app-services bhargavbachina$ az webapp deployment source config-l
esource-group rg-appservices-demo
{
  "url": "https://bb5000@react-nodejs-runtime.scm.azurewebsites.net/react-nodejs-runtime.git"
}
```

Figure 13: az webapp deployment

Once the above command is completed, you need to run this command to add a Git remote using the URL you got from your app.

git remote add azure <url>

Finally, run this command to push the changes and you can see the logs below. Make sure you create a master branch. If you are on a different branch called main, run the following commands.

// create a master branchgit checkout -b master main

// add and commitgit add .git commit -m "initial commit"

// pushgit push azure master

```
remote: added 786 packages from 424 contributors and audited 789 packages in 67.895s
remote: npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.13: wanted {"os":"d
t: {"os":"linux","arch":"x64"})
remote: npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@~2.3.1 (node_modules/nodemon/node_modules/chok
remote: npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.3.2: wanted {"os":"da
: {"os":"linux","arch":"x64"})
remote: npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@~2.3.1 (node_modules/watchpack/node_modules/ch
):
remote: npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.3.2: wanted {"os":"da
: {"os":"linux","arch":"x64"})
remote:
remote:
remote: 29 packages are looking for funding
remote:   run `npm fund` for details
remote:
remote: found 0 vulnerabilities
remote:
remote:
remote: Zipping existing node_modules folder...
remote: Done in 2 sec(s).
remote: Preparing output...
remote:
remote: Copying files to destination directory '/home/site/wwwroot'...
remote: Done in 1 sec(s).
remote:
remote: Removing existing manifest file
remote: Creating a manifest file...
remote: Manifest file created.
remote:
remote: Done in 74 sec(s).
remote: Running post deployment command(s)...
remote: Triggering recycle (preview mode disabled).
remote: Deployment successful.
remote: Deployment Logs : 'https://react-nodejs-runtime.scm.azurewebsites.net/jsonviewer?view_url=/api/deployme
6ca24cf31c4d22df/log'
To https://react-nodejs-runtime.scm.azurewebsites.net/react-nodejs-runtime.git
   0adf44c..3164ae6  master -> master
```

Figure 14: git push azure master

After the above steps, you can see the Local Git configured in the Deployment Center as below.

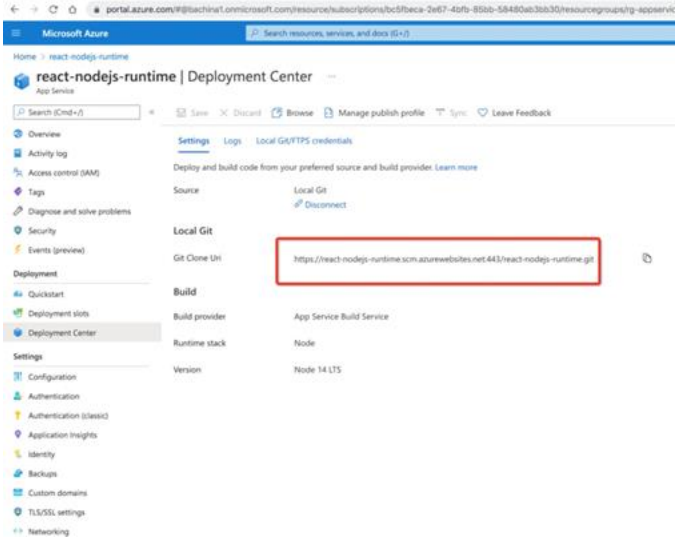


Figure 15: Deployment Center

7. Demo

Once you push the code with the above command `git push azure master` you go to the overview page and get the URL and hit it in the browser.

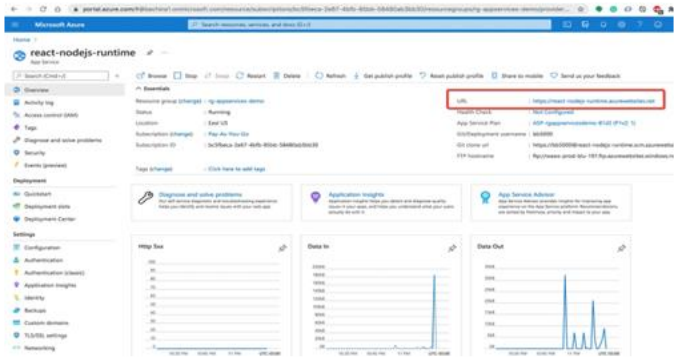


Figure 16: App Services Overview Page

You can hit the URL in the browser, and you can see the application running at https://react-nodejs-runtime.azurewebsites.net

8. How to See the Logs

Sometimes, you might go wrong while deploying in this way. You need to see the logs otherwise there is no way you could find or solve the issues. Azure provides logs and live log stream as well.

Remember, when you push the code with this command `git push azure master` it outputs a lot of logs and you can find the log URL at the end of the logs as below.

Figure 17: Deployment Logs

You can copy the URL and hit it in the browser.

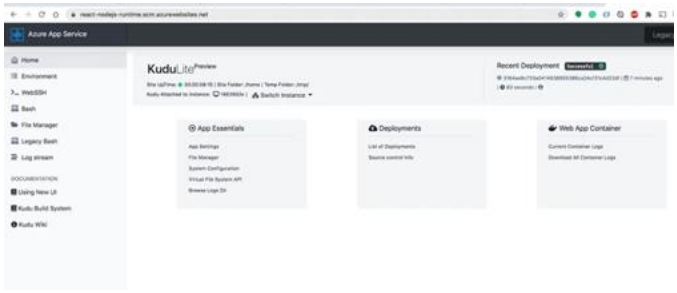


Figure 18: Deployment Logs

Here is the log stream.

Figure 19: log stream

You can check the log files as well if you click on the File Manager on the left as you can see in the below screen.

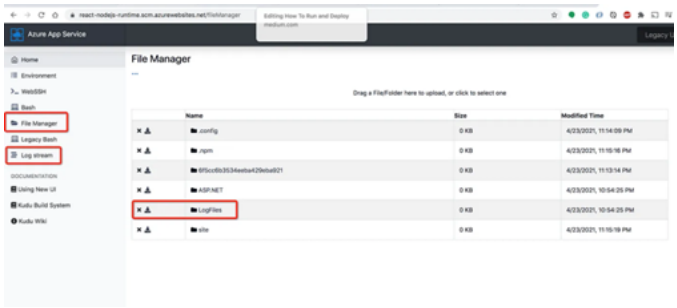


Figure 20: File Manager

9. How to Clone and update the repo

Once you push the code to the App services and you can see the local git repo on the overview page and deployment center page.

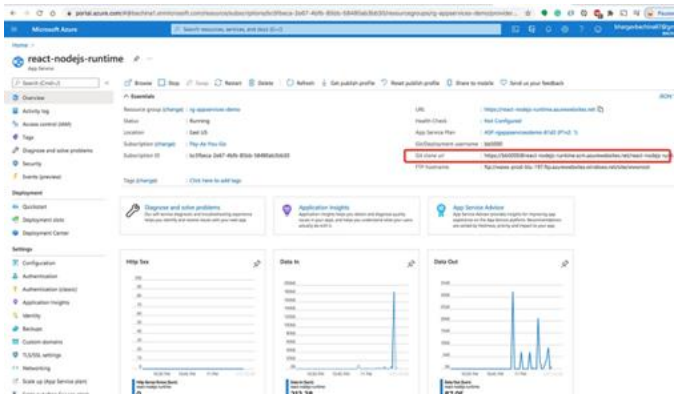


Figure 21: Overview Page

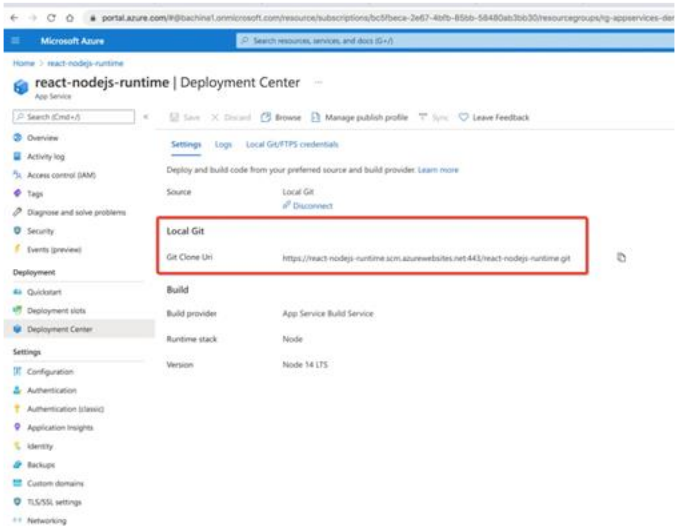


Figure 22: Deployment Center

You can clone it and update and push it as a normal repo.

10. Summary

- If you want to deploy your application on the managed platform by selecting the runtime, An App Service is the right choice.
- When it comes to React with NodeJS Backend you can build and deploy in several ways and the Azure App service is one of them.
- You need to create a web app service before deploying our application with Local git.
- If you are maintaining two package.json files for React and NodeJS respectively as below you need to build the React project.
- When it comes to the NodeJS server you should run on port 8080 and you need to serve React assets from the build we generated in the above step.
- If you go to the deployment center of the app service that we created above. we have three options one of them is local git.
- When you push the code with this command git push azure master it outputs a lot of logs, and you can find the log URL at the end of the logs.

11. Conclusion

In conclusion, Azure App Service offers a streamlined solution for deploying applications on a managed platform, allowing developers to choose their desired runtime environment. Particularly for React applications coupled with Node.js backends, Azure App Service provides a versatile deployment option among several alternatives. To begin deployment, it's essential to first create a web app service before proceeding to deploy the application using Local Git. For projects with separate package.json files for React and Node.js, it's crucial to build the React project prior to deployment. Furthermore, when configuring the Node.js server, ensure it runs on port 8080 and serves React assets from the generated build. Leveraging the deployment center within the app service provides convenient options, including local git, simplifying the deployment process. Finally, pushing the code to Azure via git initiates a detailed logging process, with the log URL conveniently accessible at the conclusion of the logs.

References

[1]. NodeJS documentation https://nodejs.org/en/guides
[2]. React Official Documentation https://react.dev/
[3]. Azure App Service Documentation https://azure.microsoft.com/en-us/products/app-service
[4]. JavaScript Documentation https://developer.mozilla.org/en-US/docs/Web/JavaScript