

★ Get unlimited access to the best of Medium for less than \$1/week. [Become a member](#)



How to deploy a Node Express API on Azure Web App with GitHub

Step-by-Step Guide: Deploying a Node.js REST API on Azure Cloud



Prasad Lakshan · [Follow](#)

Published in Enlear Academy · 6 min read · Sep 15, 2023



123



1

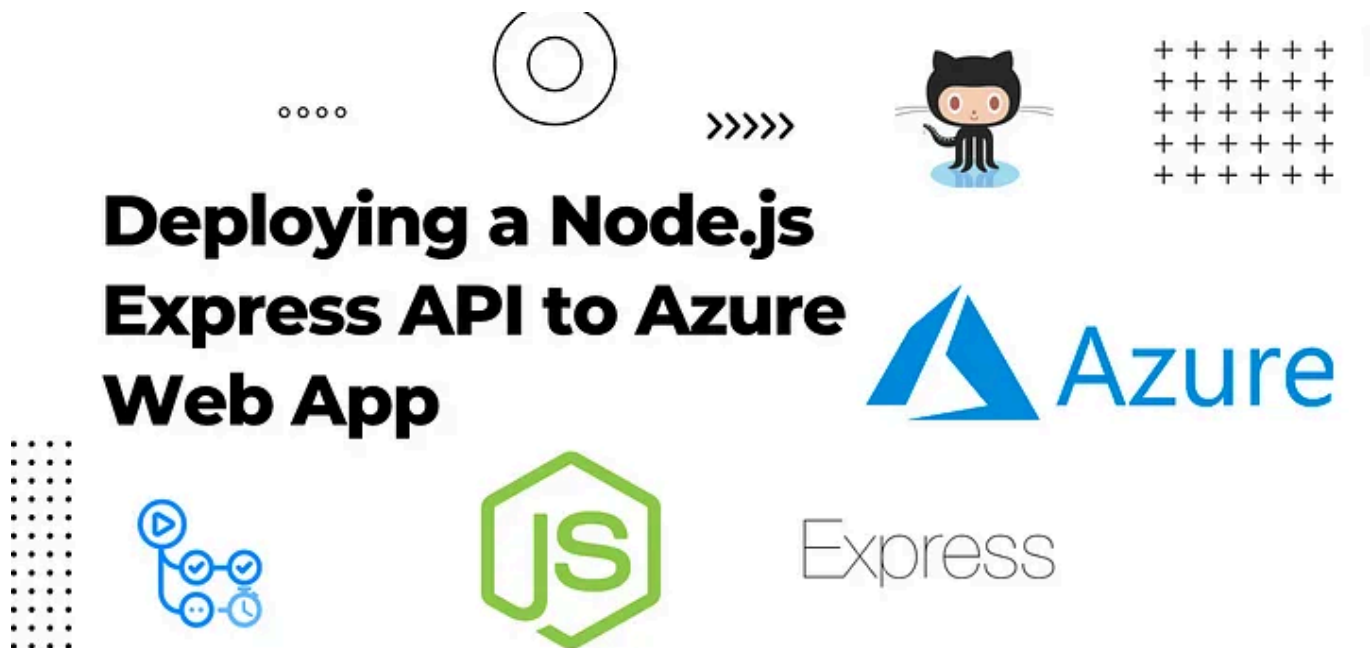


Figure 1.1 Deploying a Node.js Express API to Azure Web App with GitHub

Are you looking to deploy your Node.js API in Azure? Azure app service is a powerful platform for hosting your web application in Microsoft Azure cloud. You may speed up the process of delivering your Node.js Express application to Microsoft Azure by combining it with GitHub for version control and continuous deployment. We will walk you through the steps to achieve this seamless deployment procedure in this article.

Prerequisites

Before you begin, make sure you have the following in place:

1. **Node.js Installed:** Ensure that you have Node.js and npm(node package manager) installed on your local development machine. You can download and install Node.js from official website: [Node.js Download](#)
2. **GitHub Account:** You need a GitHub account to host your application source code.
3. **Azure Account:** Sign up for an Azure account if you don't already have one. You can get started with a free account, which includes a limited amount of resources.

Step 1: Create a Simple Node.js Express API

Let's start by creating a simple Node.js Express API. If you already have an API, you can skip this step. Otherwise, follow these instructions:

1. Create a new directory for your API project.
2. Open your terminal in the project directory and run the following commands to initialize your project and install Express:

```
mkdir my-express-api
cd my-express-api
npm init -y
npm install express --save
```

3. Create a JavaScript file (e.g., `app.js`) and set up a basic Express server. Here's a simple example in JavaScript:

```
const express = require('express');
const app = express();
const port = process.env.PORT || 3000;

app.get('/', (req, res) => {
  res.send('Hello, World!');
});

app.listen(port, () => {
  console.log(`Server is running on port ${port}`);
});
```

3. Update the `package.json` with the following script.

```
"scripts": {
  "start": "node app.js"
},
```

4. Test your API locally by running `npm start`. You should see "Server is running on port 3000" in the console.

5. Also test your API locally, you can use the `curl` command-line tool. Open a new terminal window and run the following `curl` command to make a GET request to your API:

```
curl http://localhost:3000
```

This command sends a GET request to your locally running Express API. You should receive the response “Hello, World!” from your API.

Testing your API locally with `curl` allows you to ensure that it's functioning correctly before deploying it to the Azure app service. With these additional instructions, you can verify that your locally hosted Express API responds as expected using the `curl` command-line tool.

Step 2: Create a GitHub Repository

If you haven't already, create a GitHub repository to host your Node.js Express API's source code. Follow these steps:

1. Sign in to GitHub.
2. Click the “+” sign in the upper right corner.
3. Select “New repository.”
4. Follow the instructions to create a new repository.

Step 3: Push Your Code to GitHub

With your GitHub repository set up, it's time to push your Node.js Express API's code to it. Open your terminal and navigate to your project's root directory. Use these commands to initialize your Git repository, add your files, and push to GitHub:

```
git init
git add .
git commit -m "Initial commit"
git branch -M main
git remote add origin <your-github-repo-url>
git push -u origin main
```

Replace `<your-github-repo-url>` with the URL of your GitHub repository.

Step 4: Set Up an Azure App Services

Now, let's create an Azure App service to host your Node.js Express API:

1. Sign in to the Azure portal (<https://portal.azure.com>).
2. Search App services on the Azure portal click on Create and select Web App from the dropdown as follows.

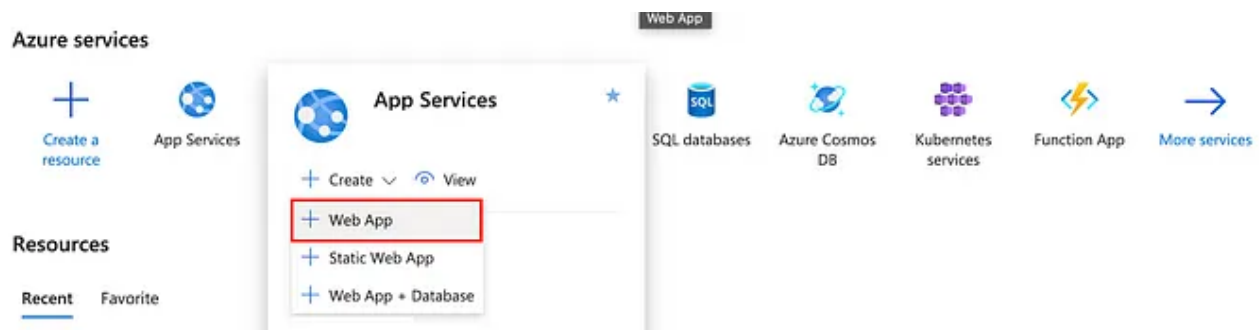


Figure 1.2

3. Create a new Web App, providing a unique name, subscription, resource group, and other necessary details as follows. Select the Runtime stack as Node 18 LTS .

Home > Create Web App

Basics Deployment Networking Monitoring Tags Review + create

App Service Web Apps lets you quickly build, deploy, and scale enterprise-grade web, mobile, and API apps running on any platform. Meet rigorous performance, scalability, security and compliance requirements while using a fully managed platform to perform infrastructure maintenance. [Learn more](#)

Project Details

Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *

Resource Group *
[Create new](#)

Instance Details

Need a database? Try the new Web + Database experience. [🔗](#)

Name * .azurewebsites.net

Publish * ☒ Code ☐ Docker Container ☐ Static Web App

Runtime stack *

Operating System * ☒ Linux ☐ Windows

Region *
🔗 Not finding your App Service Plan? Try a different region or select your App Service Environment.

Figure 1.3

4. Then, choose the pricing plan that is best for you. Because this is for educational purposes, I've chosen a free-price plan. You can find a suitable pricing plan by clicking on the [Explore pricing plans](#)

Pricing plans

App Service plan pricing tier determines the location, features, cost and compute resources associated with your app. [Learn more](#)

Linux Plan (East US) *
[Create new](#)

Pricing plan
[Explore pricing plans](#)

Zone redundancy

An App Service plan can be deployed as a zone redundant service in the regions that support it. This is a deployment time only decision. You can't make an App Service plan zone redundant after it has been deployed. [Learn more](#)

Zone redundancy ☐ Enabled: Your App Service plan and the apps in it will be zone redundant. The minimum App Service plan instance count will be three. ☒ Disabled: Your App Service Plan and the apps in it will not be zone redundant. The minimum App Service plan instance count will be one.

[Review + create](#) < Previous Next: Deployment >

Figure 1.4

5. When you finish these steps, click the Review + Create button, and the web app with the name you entered will be created. The web app will take some time to complete the deployment.

6. After the deployment is complete, navigate to your Web app service by clicking on the service name you specified.

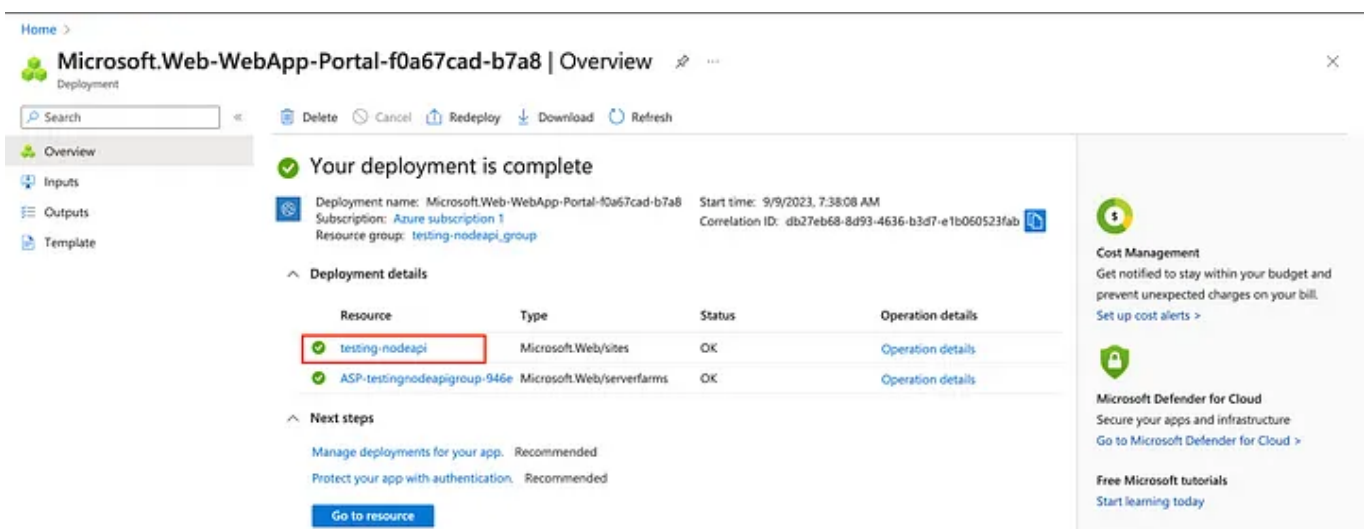


Figure 1.5

7. After that you will be able to see the Web App service overview page for your project. Next Search Deployment Center From the sidebar click on that to configure the Azure continuous deployments.

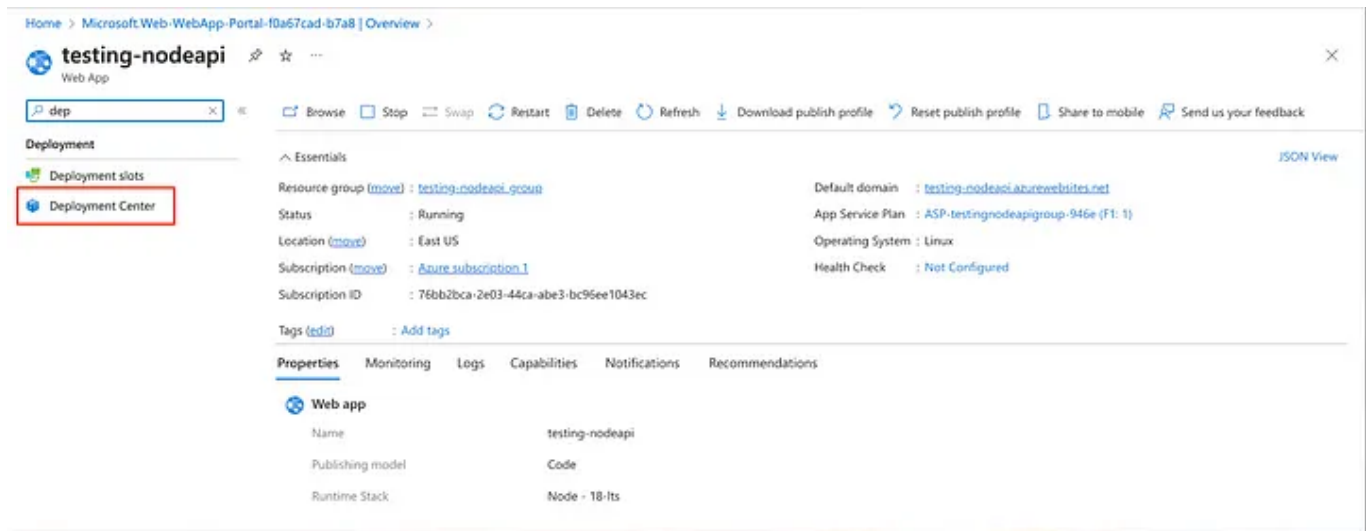


Figure 1.6

Step 5: Configure Azure Deployment Center

1. On that go to `settings` tab and select the Source as GitHub as follows.

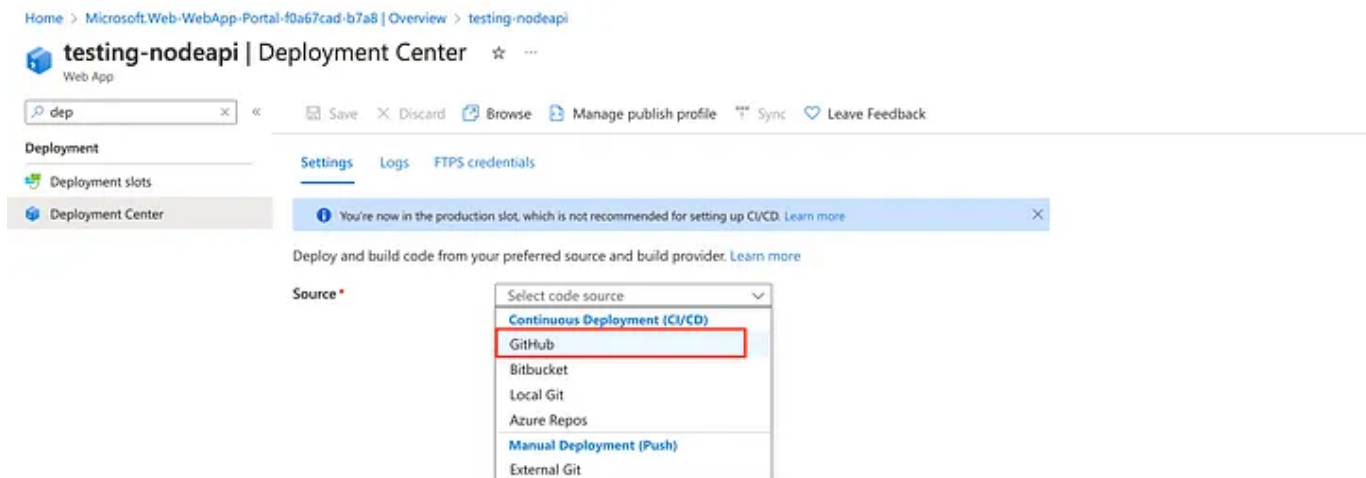


Figure 1.7

2. After that you have to link your GitHub account with your Azure account by following the necessary steps.

3. Next, you'll need to choose the repository where your Node Express API source code is hosted and specify the branch for setting up the CI/CD pipeline as follows.

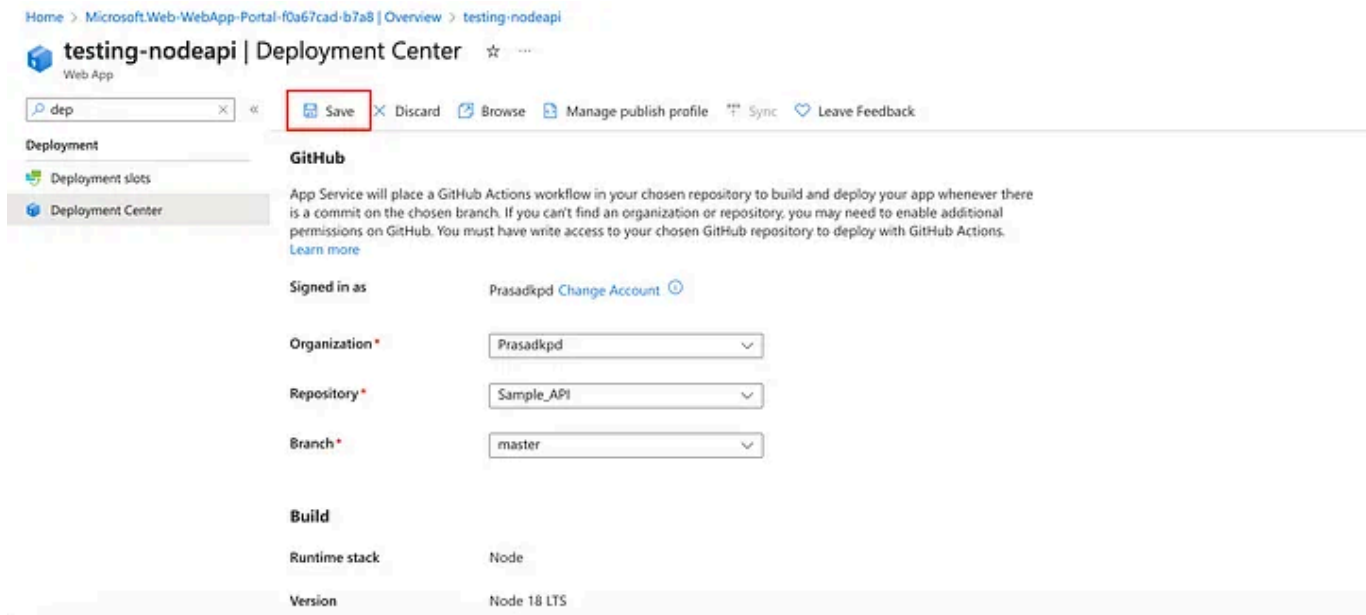


Figure 1.8

4. When all the steps are done click on the save button. This will create a GitHub action on your repository. And deploy your Node.js app on Azure Web app services.

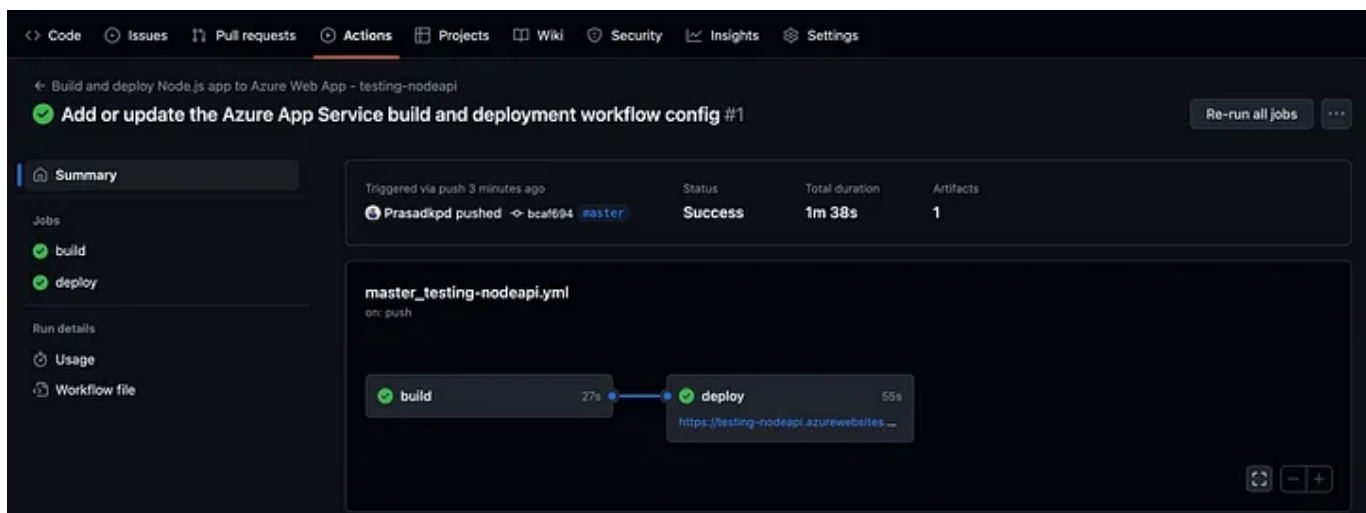


Figure 1.9

5. Azure will automatically trigger a deployment whenever you push changes to the specified branch (master) in your GitHub repository.

Step 6: Complete the Configuration

When the deployment is finished, Search **Configuration** from the sidebar and click on that to configure the startup command.

Then go to the general Settings tab and enter the Startup command to start our Node.js application. After that click on the Save button. This will start the Node.js application.

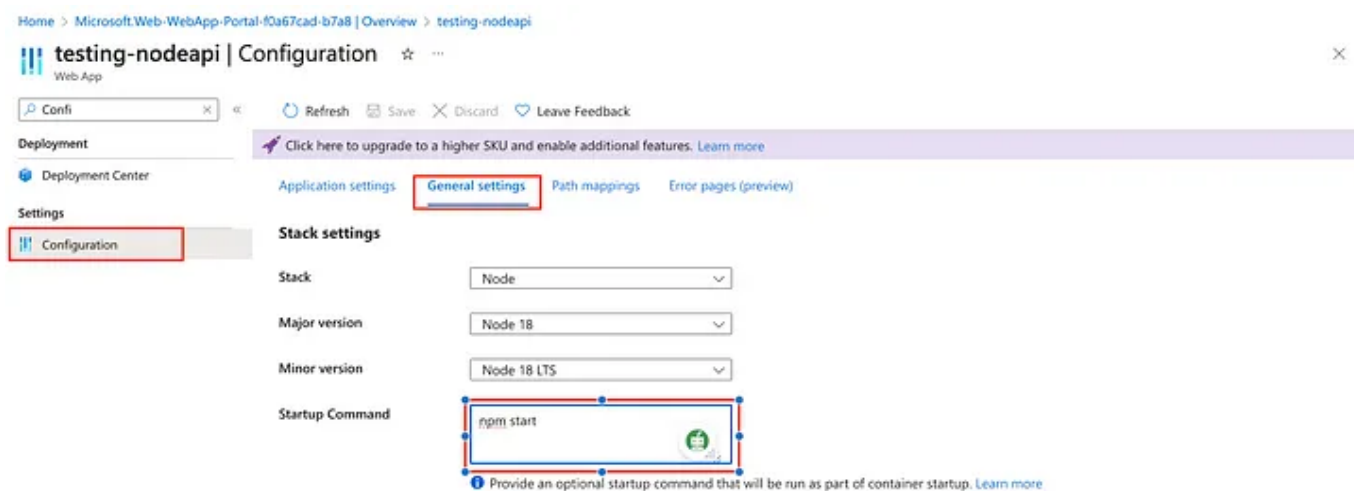
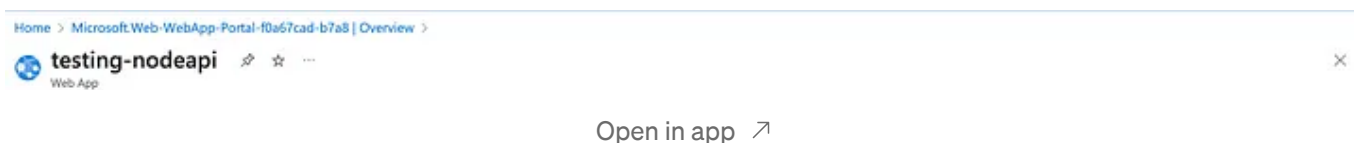


Figure 1.10

Step 7: Access Your API

After a successful deployment, your Node.js Express API should be accessible through the Azure Web App's URL. You can find this URL in the Azure portal's overview page for your Web App.



Search

Write



Microsoft Defender for Cloud

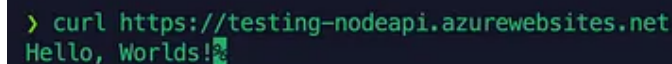
Subscription ID

Events (preview)

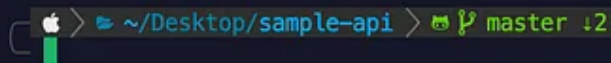
Tags (edit) Add tags

Figure 1.11

You can test your Node app by using the curl command in your terminal.



```
> curl https://testing-nodeapi.azurewebsites.net
Hello, Worlds!
```



```
~/Desktop/sample-api > master +2
```

Figure 1.12

Step 8: Configure Custom Domains (Optional)

If you have a custom domain for your API, you can configure it to point to your Azure Web App. This step involves updating your DNS settings in your domain registrar's control panel.

Conclusion

Congratulations! You have successfully deployed your Node Express API in Azure App Service with GitHub. Using GitHub for version control and continuous deployment to deploy a simple Node.js Express API to Azure Web App streamlines the deployment process, making it easy to manage and update your API. You can efficiently host and maintain your API on the cloud if you follow these procedures.

Remember to treat any Azure environment-specific configurations (e.g., database connection strings) correctly. With this setup in place, you can concentrate on designing your API while Azure handles the deployment.

Resources

Example Node API Code Repository -

https://github.com/Prasadkpd/Sample_API

[Azure](#)[Deployment](#)[Cloud Services](#)[Hosting](#)[Azure Services](#)

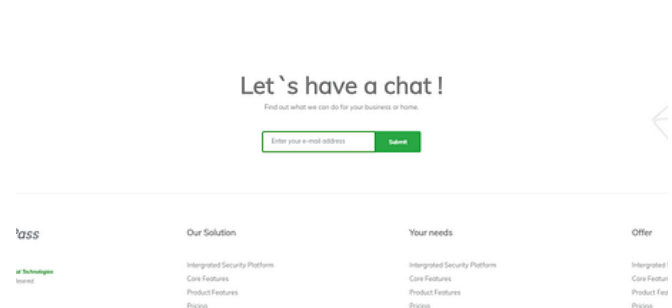
Written by Prasad Lakshan

[Follow](#)

416 Followers · Writer for Enlear Academy

Tech enthusiast, passionate about exploring opportunities to learn, teach, help, and take experiences. <https://howtocodes.com>

More from Prasad Lakshan and Enlear Academy



Prasad Lakshan in UCSC ISACA Student Group



Niemvuilaptrinh in Enlear Academy

OpenID vs. OAuth vs. SAML: Understanding the Key Differences

Choosing the Right Protocol for Secure
Access Management

4 min read · Mar 6, 2023



139



1



29 footer examples for website

Welcome back to my blog. Today we are
going to learn Footer snippet by combining...

5 min read · Dec 28, 2021



137



1



Niemvuilaptrinh in Enlear Academy

Best 31 Login Form Templates For Website

Free Login Form Templates

5 min read · Sep 28, 2021



154



Prasad Lakshan in AWS in Plain English

Deploying a Node Express API on AWS Lambda

Launching a Node Express API with AWS
Lambda

5 min read · Dec 27, 2023



54



3



See all from Prasad Lakshan

See all from Enlear Academy

Recommended from Medium

Azure App Service



=

Web Apps

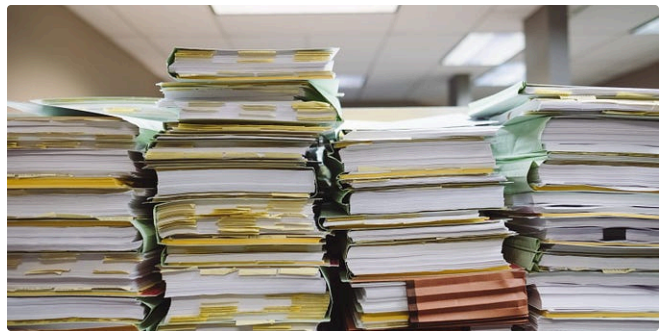


Mobile Apps



API Apps

Logic Apps



Ivan Scoles

Deployment with Azure, Node and GitHub Actions

Let's start at the beginning. If you are doing your first steps with nodejs and want to...

4 min read · Dec 27, 2023



Aashish Peepra in Python in Plain English

Folder Structure for Your Backend and How to Keep it Clean

Structuring your code base seems like the last thing standing between you and scalability...

9 min read · Nov 11, 2023



Lists



ChatGPT

21 stories · 626 saves



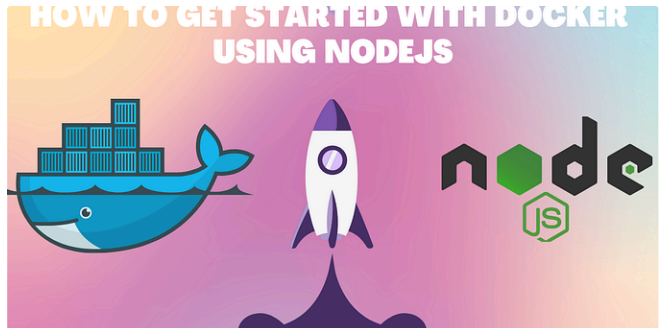
Generative AI Recommended Reading

52 stories · 1018 saves



Natural Language Processing

1438 stories · 935 saves





Jean F Beaulieu

How to Deploy a Dockerized React App Using Azure DevOps Pipelines

A Comprehensive Guide to Efficiently Containerize and Deploy React Applications...

10 min read · Dec 16, 2023



54



1



Vishal Sharma

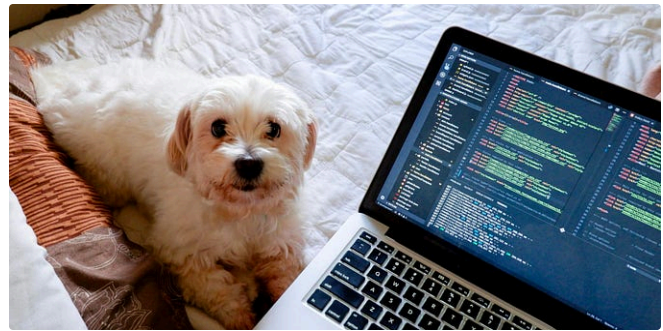
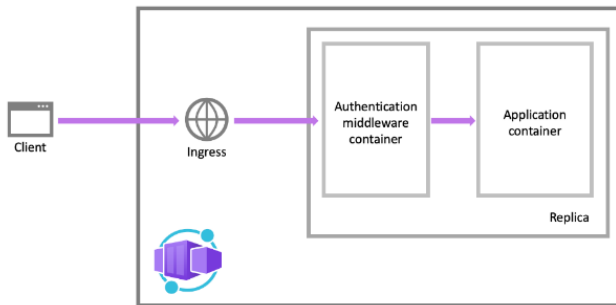
Simple Node.js Application with Docker and Docker-Compose

Learn how to build a Docker image for a simple NodeJS application and deploy the...

4 min read · Dec 1, 2023



2



Takumi Seo

Azure Container Apps Authentication Functionalities

Introduction

8 min read · Dec 16, 2023



10



HolaSoyMalva

How to make your first REST API in Node.js

In less than 5 minutes

3 min read · Jan 20, 2024



19



1



See more recommendations