Deploy Netlify Serverless Functions

with Node.js and GitHub

Serverless Functions Introduction

Are you looking to create a serverless application with Node.js and deploy it on the cloud?

In this guide, I will walk you through the steps to create a Node.js project, upload it to GitHub, and link it to Netlify for easy deployment (CI/CD). Additionally, I will explain how to get a URL endpoint for your API using Netlify Functions.

I also cover how to create a private GitHub repository and how to create a basic Node.js app.

And with Netlify's free usage for serverless functions, you can easily get started with creating and deploying your own functions without needing a credit card or worrying about usage costs.

What is Serverless Functions?

Serverless functions, also known as function-as-a-service (FaaS), are a cloud computing model that allows developers to write and deploy code without worrying about the underlying infrastructure.

In this model, the cloud provider manages the infrastructure, automatically scaling the resources up or down as needed, and the developer only needs to write the code for the function.

When a function is called, the provider runs the code, and the results are returned to the caller. This approach is useful for building scalable and highly available applications that can respond quickly to changing workloads.

Node.js App

Installations

Before we dive into creating serverless functions with Node.js, Netlify, and GitHub, there are a few things we need to set up first. Make sure you have Visual Studio Code and Node.js installed on your computer. If you don't have them installed, you can download and install them from their official websites:

- Node.js
- Visual Studio Code

Create Project

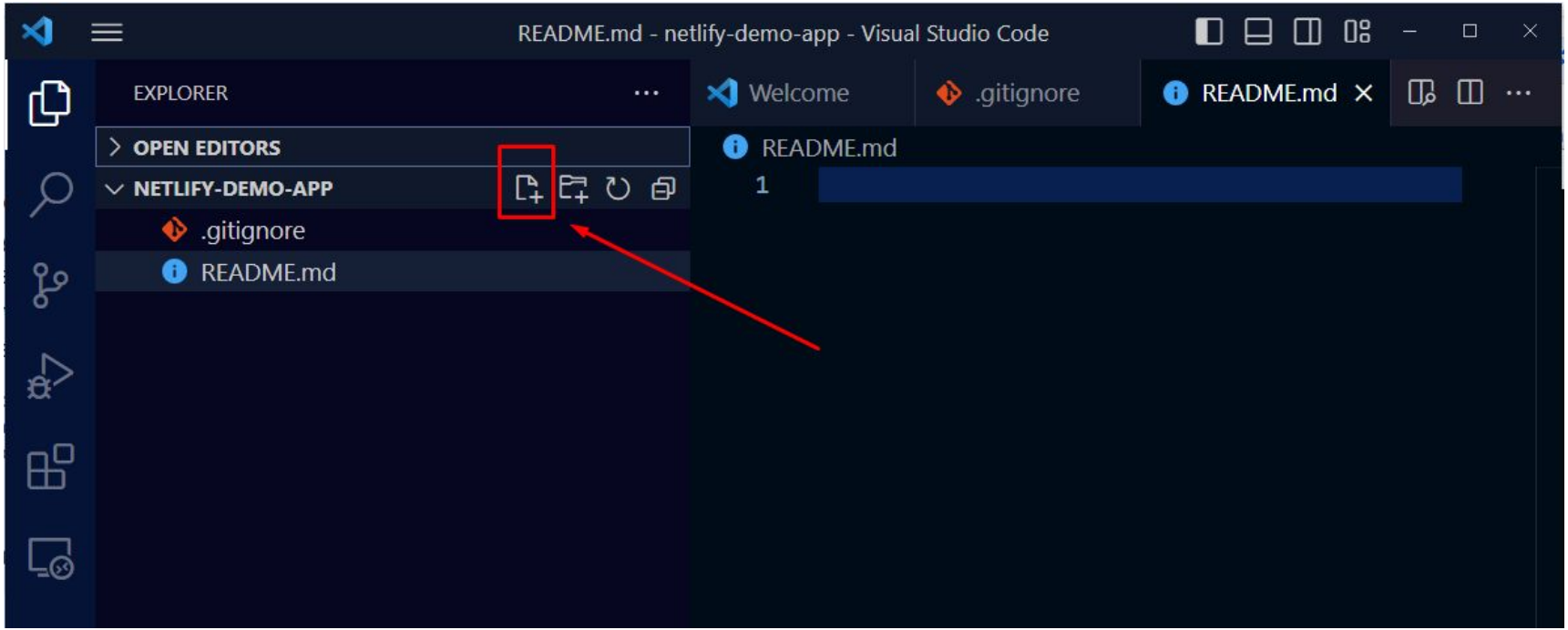
After you've installed Node.js on your computer, you'll need to create a new folder for your project, let's call it netlify-demo-app. Once you've created your new folder, you should navigate to it using the cd command from your terminal and open your project with code . command.

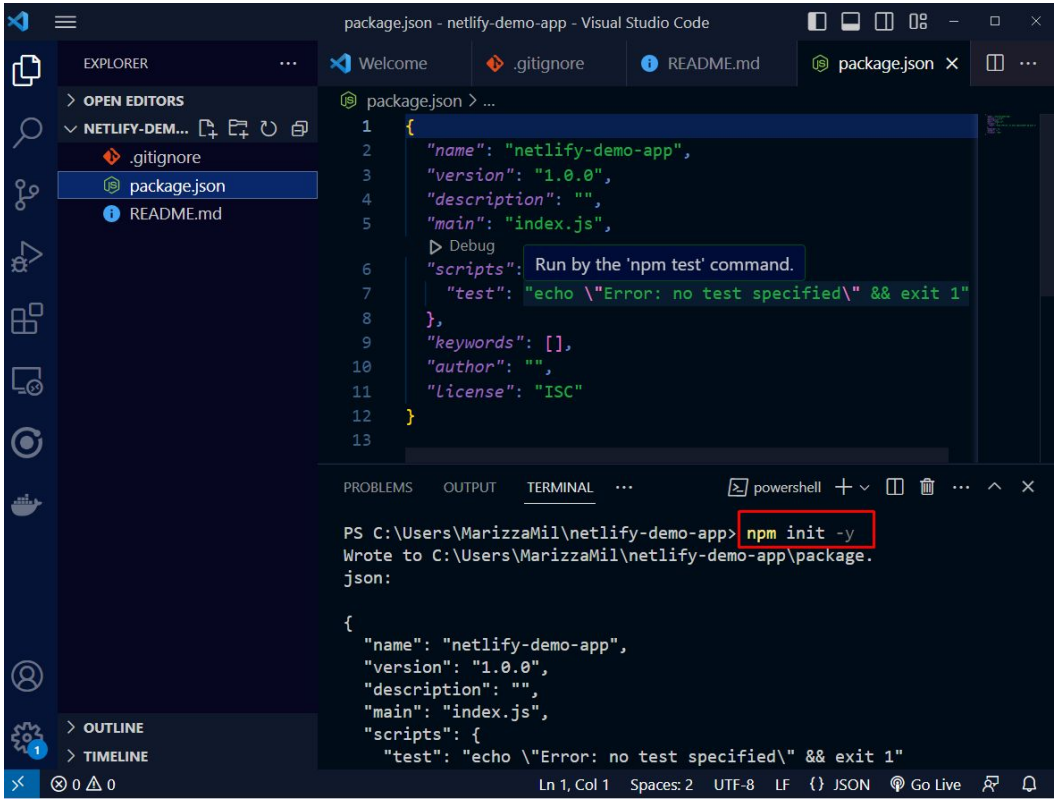
```
C:\Users\MarizzaMil>mkdir netlify-demo-app

C:\Users\MarizzaMil>cd netlify-demo-app

C:\Users\MarizzaMil\netlify-demo-app>code .
```

Now, at the root of your project, you'll need to create a `.gitignore` file to ignore certain files when you commit changes to your Git repository. And, you should create a `README.md` file to describe your project and its purpose.





When you create a new Node.js project, you need to create a 'package.json' file to manage your project dependencies and metadata. The 'npm init' command is used to create this file, but it requires you to answer a series of questions about your project, such as the project name, description, author, version, and so on.

The 'npm init -y' command skips the interactive process of creating a new package.json file and uses the default values for all the questions, creating a package.json file with the bare minimum configuration needed to get started with your project.

Using 'npm init -y' can save you a lot of time when you're starting a new project, especially if you don't need to customize the package.json file much at the beginning. However, it's still recommended to review and update the 'package.json' file as needed once it's created.

.gitignore File

The `.gitignore` file is an important part of any Git project. It allows you to specify files and directories that should be ignored by Git and not included in your repository, without cluttering it up with unnecessary files or risking including sensitive information that should not be shared.

When you create a new project, it's a good idea to create a `.gitignore` file at the root of your project before you link it to Git. This will ensure that any files or directories that should not be included in your repository are ignored from the very beginning.

For example, if you're working on a Node.js project, you might want to include the following lines in your `.gitignore` file:

.gitignore

```
# Contains all the installed Node.j
node_modules/

# Contains the built, compiled or b
build/
dist/

# This file contains sensitive envi
.env

# This directory contains test cove
coverage/

# These files are created by the ma
.DS_Store
Thumbs.db
# Local Netlify folder
.netlify

```

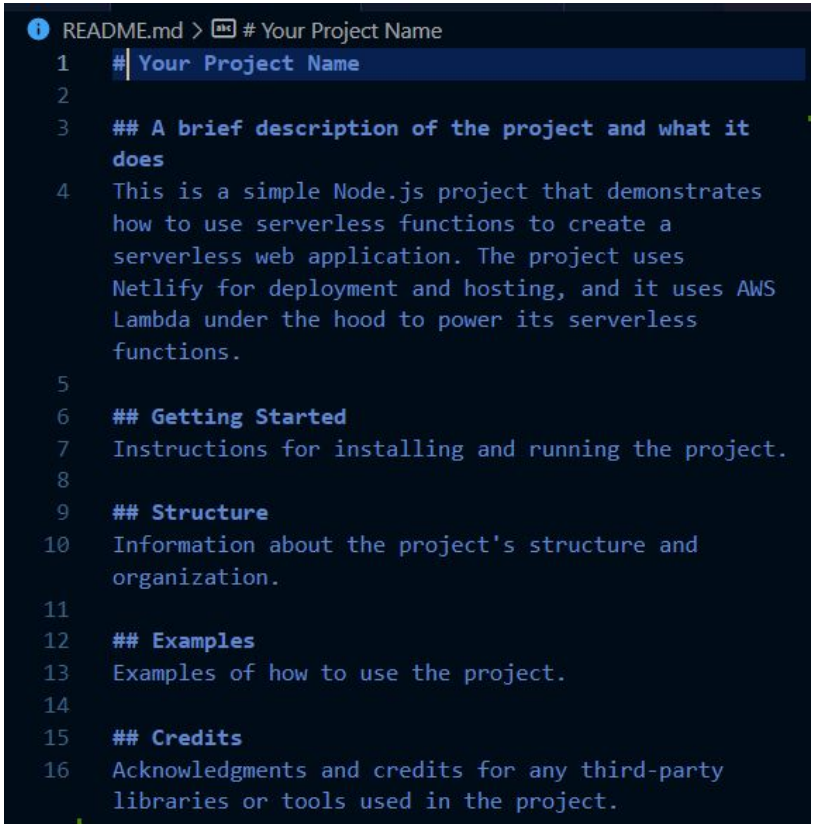
README.md File

The 'README.md' file is an essential part of many software projects, providing a high-level overview of the project and its contents. It's written in Markdown, a lightweight markup language that allows you to format text using plain text characters.

It's usually the first file that developers see when they open a project in a code editor or on a code sharing platform like GitHub. The file serves as documentation, explaining what the project is, how it works, and how to use it.

In a serverless Node.js project like the one we're working on, the 'README.md' file should provide a brief overview of the project's architecture, including how it works with Netlify's serverless functions. It should also include information on how to deploy the project to Netlify and how to use its serverless functions.

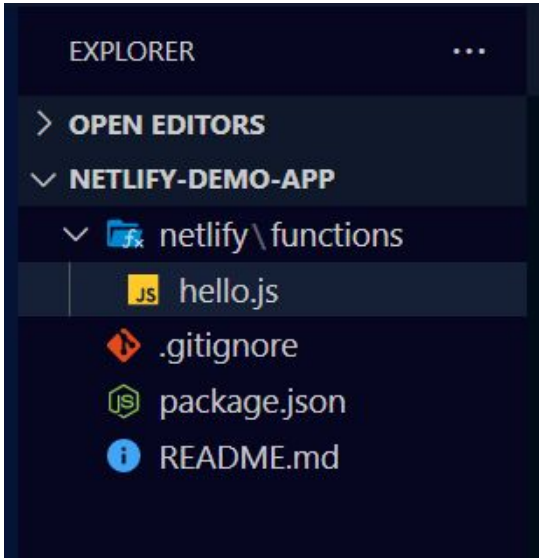
Here's an example `README.md` file for a Node.js project using serverless functions



Netlify Functions Folder

Now, at the root of your project, you'll need to create a nested folder called `netlify/functions`. This is where we'll put our Netlify serverless function code. You can simply create a new folder called `netlify`, and then create another folder called `functions` inside the netlify folder. This will give you the required folder structure.

It's a directory that contains all the serverless functions for your project. These functions can be written in different programming languages, but in this case, we are focusing on JavaScript.



Here's an example of a simple `hello.js` file for a Netlify function that returns a "Hello, world!" message:

netlify > functions > hello.js > ...

```js
exports.handler = async (event, context) => {
    return {
      statusCode: 200,
      body: "Hello, world!"
    };
  };
```

Netlify CLI

Netlify CLI

To run this project locally, you'll need to have Node.js and the Netlify CLI installed. You can install Node.js from the official website, and you can install the Netlify CLI using NPM:

```
npm install netlify-cli -g
```

```
npm install netlify-cli -g --unsafe-perm=true --allow-root
```

By default, Netlify collects data on usage of Netlify CLI commands, opt out of sharing usage data with the command line:

```
netlify --telemetry-disable
```

Netlify CLI

⚠️ Some users reporting an error related to Execution Policies. This is because the script execution policies in PowerShell may be set to a restricted mode by default, preventing the running of scripts.

```
File C:\Users\your-user-name\AppData\Roaming\npm\netlify.ps1 cannot be loaded because running scripts is disabled
on this system. For more information, see about_Execution_Policies at https:/go.microsoft.com/fwlink/?LinkID=135170.
```

To solve the "Execution Policies" error, you need to run the following command in the Terminal:

```
Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy Unrestrict
```

Run the project

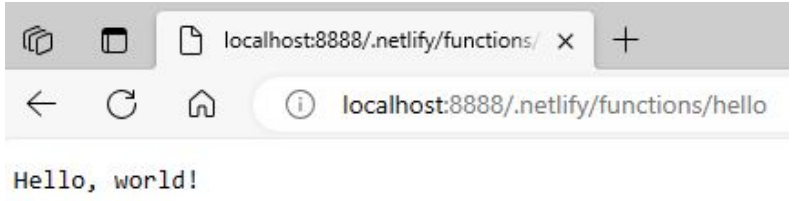
Once you have the dependencies installed, you can run the project locally using the following command:

```
netlify dev
```

Serverless Functions API Urls

When you deploy a serverless function to Netlify, it becomes available via a unique URL that includes the prefix "/.netlify/functions/".

So if you have a function called "hello" in your Netlify project, you can access it via the URL "/.netlify/functions/hello".



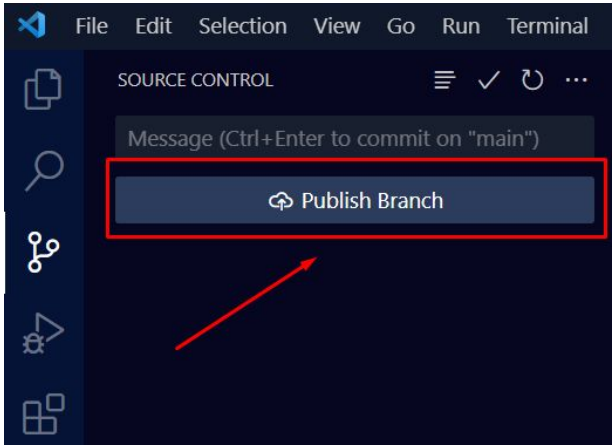
GitHub Repository

Create a Private GitHub Repository for Source Control and publish branch

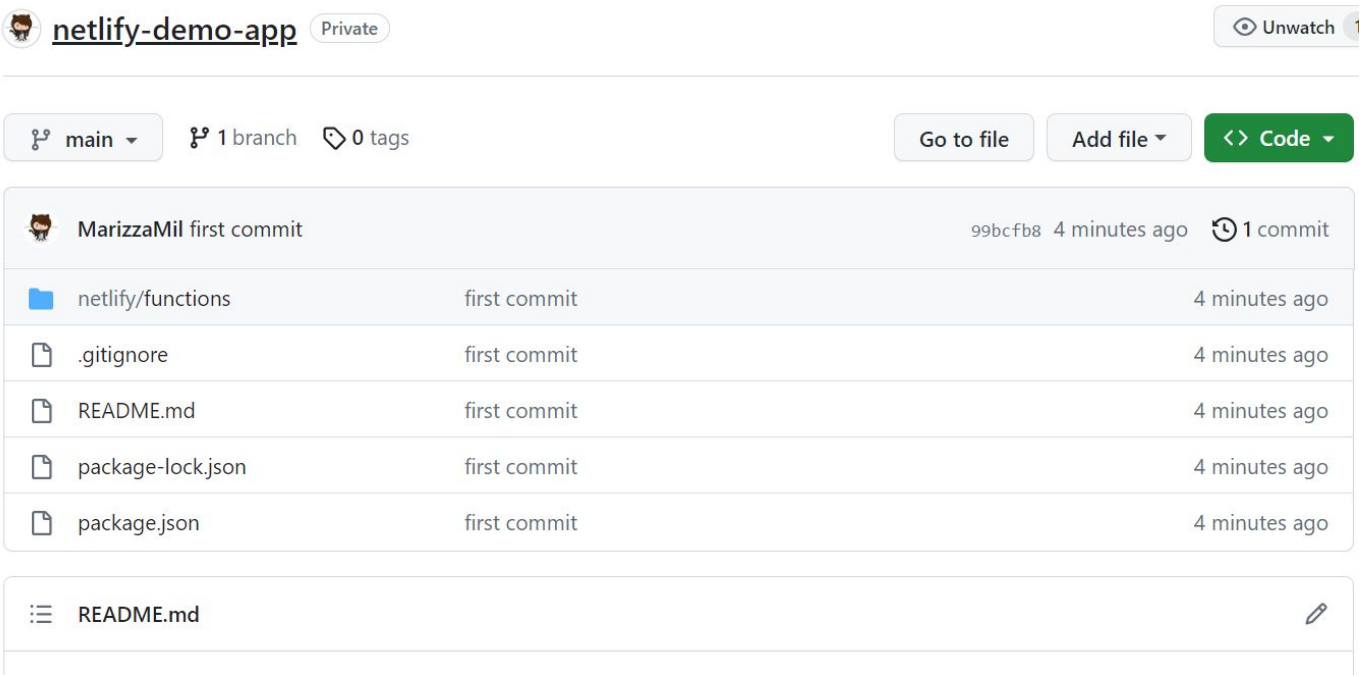
1

2

3



Create a Private GitHub Repository for Source Control



Connecting Netlify to Your GitHub Account

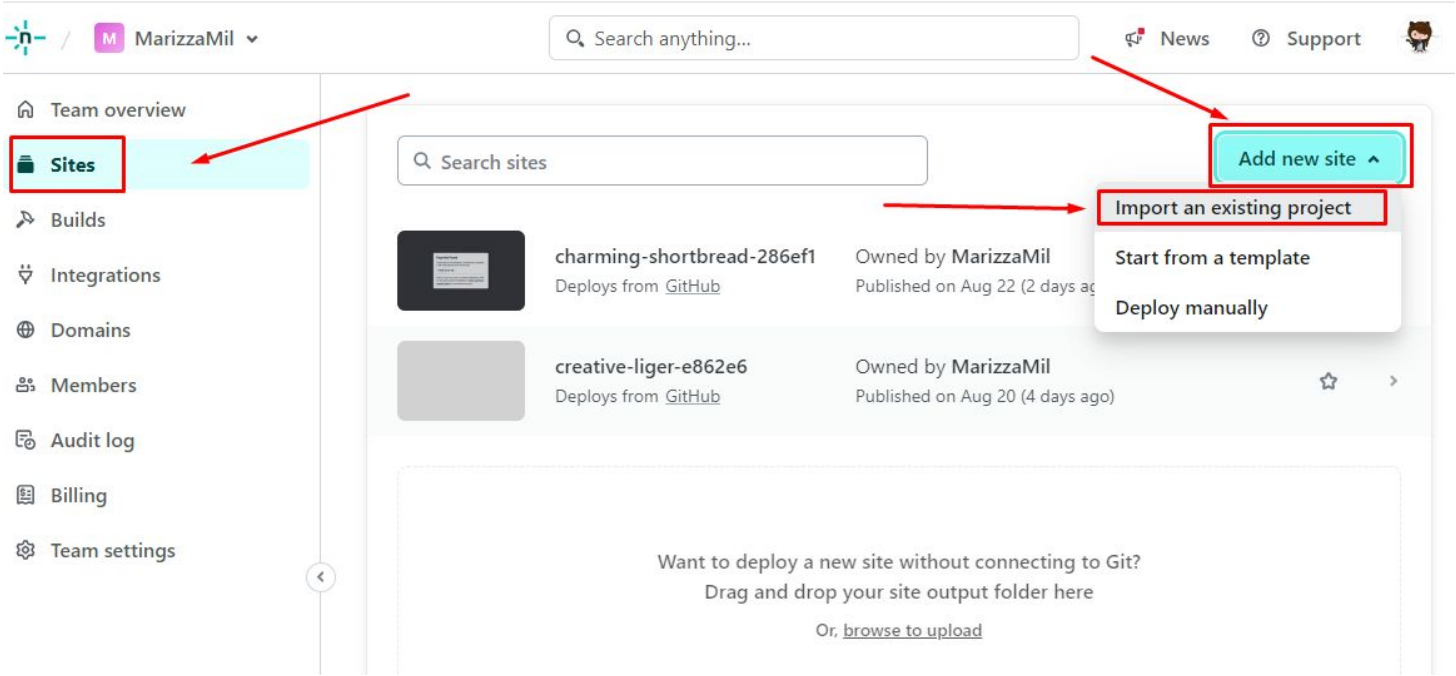
Netlify offers continuous integration and continuous deployment (CI/CD) as part of its platform. This means that when you connect your GitHub repository to Netlify, any changes you push to the repository will automatically trigger a new build and deployment of your website or application.

To connect Netlify to your GitHub account, follow these steps:

1. Go to your Netlify dashboard and select the "Sites" tab.
2. Click the "Add new site" button and "Import an existing project".

Deploy Project in the Netlify

Connecting Netlify to Your GitHub Account



Connecting Netlify to Your GitHub Account

3. Select "GitHub" as your Git provider.
4. Follow the prompts to connect your GitHub account to Netlify. You may need to grant Netlify access to your GitHub account.

Choosing "Only select repositories" option when connecting Netlify to GitHub means that you are selecting specific repositories from your GitHub account that you want to connect with Netlify. This provides more control and security to your deployment process.

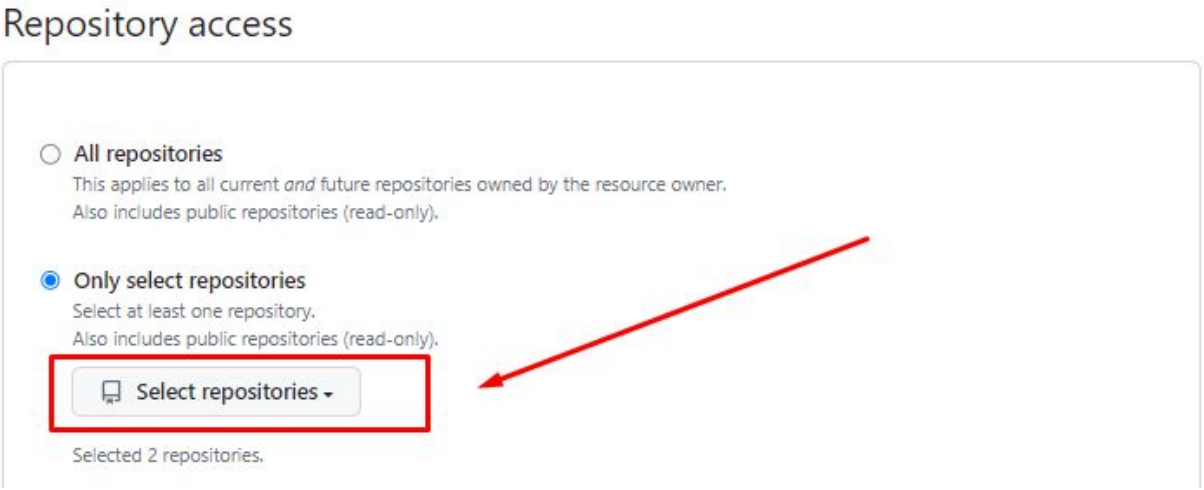
Connect to Git provider

Select repository

Once your GitHub account is connected, you can pick a repository to deploy to Netlify. In the "Pick a repository" section, select the repository you want to deploy.

If you have created a GitHub repository for your Netlify project, but it's not appearing in the list of available repositories on Netlify, you may need to configure the Netlify app on GitHub. Here's how to do it:

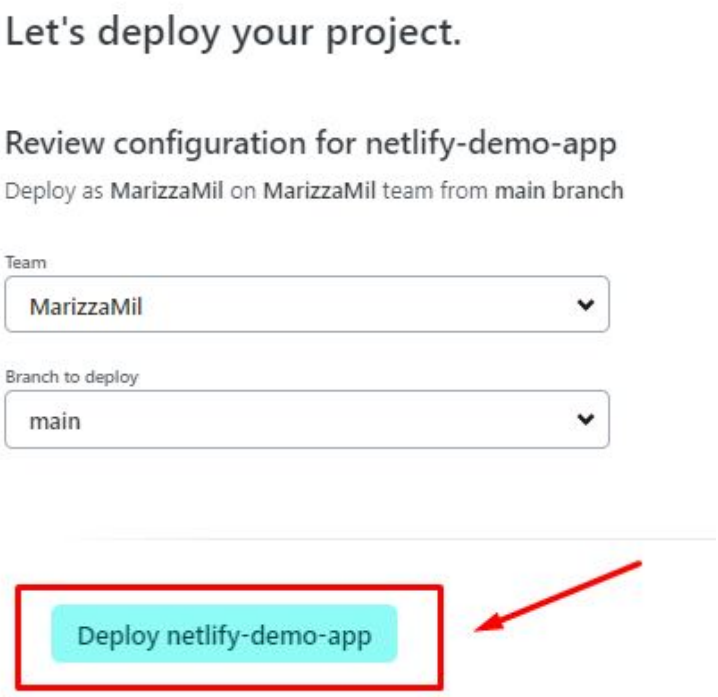
Configure site and deploy



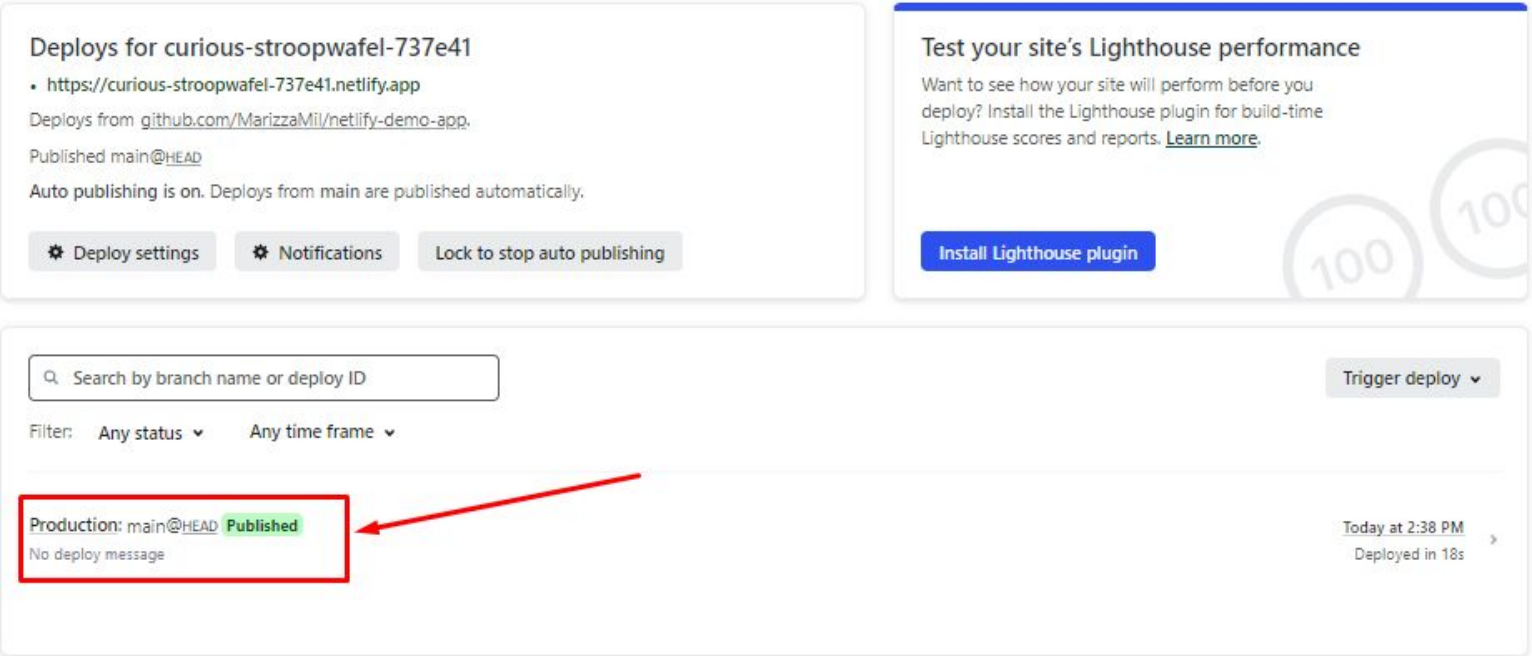
In the "Repository access" section, if you chose "Only select repositories" choose specific repositories. Select the repository you want to link to your Netlify site. Click "Save" to apply the changes.

Configure site and deploy

5. Configure your build settings. This will depend on the type of project you're deploying.
6. Click "Deploy site" to start the deployment process.



Production


Deploys for curious-stroopwafel-737e41
https://curious-stroopwafel-737e41.netlify.app
Deploys from github.com/MarizzaMil/netlify-demo-app.
Published main@HEAD
Auto publishing is on. Deploys from main are published automatically.
Deploy settings
Notifications
Lock to stop auto publishing
Test your site's Lighthouse performance
Want to see how your site will perform before you deploy? Install the Lighthouse plugin for build-time Lighthouse scores and reports. Learn more.
Install Lighthouse plugin
Search by branch name or deploy ID
Trigger deploy
Filter:
Any status
Any time frame
Production: main@HEAD Published
No deploy message
Today at 2:38 PM
Deployed in 18s

Deploy summary

Deploy summary

2 new files uploaded
2 assets changed.

No redirect rules processed
This deploy did not include any redirect rules. Learn more about redirects .

No header rules processed
This deploy did not include any header rules. Learn more about headers .

All linked resources are secure
Congratulations! No insecure mixed content found in your files.

1 function deployed
We have deployed 1 function. Visit your Functions for more information.

No edge functions deployed
This deploy did not include any edge functions. Learn more about Edge Functions .

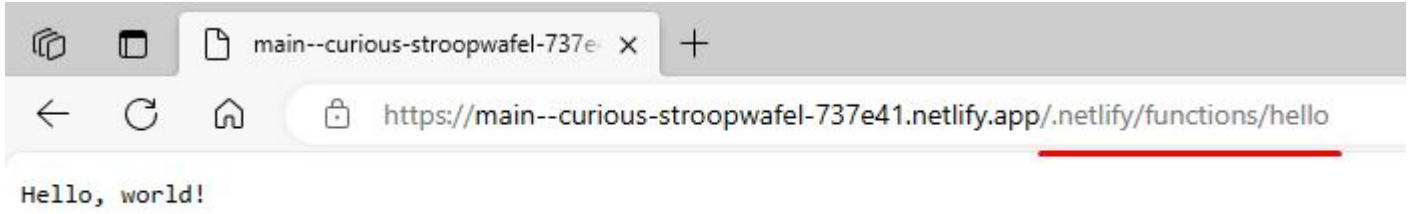
Deploy log


Deploy log
Preview
Maximize log
Initializing
Complete
Building
Complete
Deploying
Complete
Cleanup
Complete
Post-processing
Complete

Production


Published deploy for curious-stroopwafel-737e41
Permalink
Today at 2:38 PM
Production: main@HEAD
Deployed Functions
Open production deploy
Lock to stop auto publishing
Options

Production Testing



Summary

We talked about using Node.js and Netlify Functions to create serverless functions for a web application. We discussed how to organize the project and how to use it to create APIs. We also talked about connecting the code to a private GitHub repository and deploying it to Netlify.

Overall, we explored the process of creating a serverless function with Node.js, connecting it to GitHub, and deploying it to Netlify.