

Deploy a Backend API (Firebase) to Netlify





Building a Scalable Backend API with Serverless Functions

Building a backend API with Netlify functions (on top of AWS Lambda functions), Node.js, Express, and PostgreSQL can be a powerful and efficient way to create scalable and cost-effective applications.

With the right tools and techniques, developers can create robust and flexible APIs that can be easily deployed and managed on the cloud.



Project Description

In this project, we'll create and deploy a Node.js web application on Netlify that incorporates CRUD (Create, Read, Update, Delete) functionality. We'll leverage Firebase Firestore, a flexible and scalable NoSQL database, as our data storage solution. This app will serve as a practical demonstration of building a dynamic web application, showcasing how to interact with a real-time database and providing a seamless user experience.



Project Structure

This project follows a structured layout to efficiently organize its components, backend logic, and deployment setup. Here's a breakdown of the main directories and files within the project:

.netlify: This directory contains configuration files and settings specific to Netlify deployment.

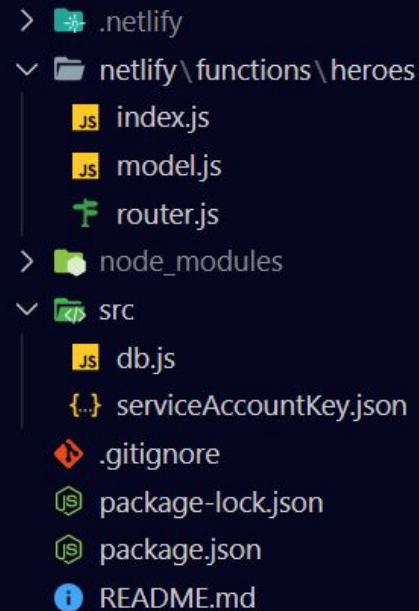
netlify/functions/heroes: This directory houses the backend functionalities of the application related to managing heroes. It's organized into separate files to ensure modular and maintainable code.

- **index.js:** This file serves as the entry point for the Netlify serverless functions related to heroes. It's responsible for routing incoming requests to appropriate routes using the Express.js framework.
- **model.js:** Here, you define the data model and interact with the database. This file encapsulates functions for CRUD operations on heroes, interacting with the database, and handling Firestore queries.
- **router.js:** This file defines the API routes that correspond to different CRUD operations. It uses Express.js to create a router that maps HTTP requests to specific model functions.

```
> .netlify
  netlify\functions\heroes
    index.js
    model.js
    router.js
  node_modules
  src
    db.js
    serviceAccountKey.json
  .gitignore
  package-lock.json
  package.json
  README.md
```

Project Structure

- **node_modules:** This directory contains the external libraries and dependencies required for the project. It's managed by npm and stores packages used in the application.
- **src/db.js:** This file establishes the connection to the Firebase Firestore database. It initializes the Firebase Admin SDK with the provided service account key, allowing the app to communicate with the Firestore database.
- **src/serviceAccountKey.json:** This JSON file contains the service account key required to authenticate the app with Firebase. It provides the necessary credentials to securely access the Firestore database.



```
> .netlify
└─ netlify\functions\heroes
   ├── index.js
   ├── model.js
   └── router.js
> node_modules
└─ src
   ├── db.js
   └── serviceAccountKey.json
.gitignore
package-lock.json
package.json
README.md
```



Project Structure

- **.gitignore:** The .gitignore file specifies which files and directories should be excluded from version control. It ensures sensitive or unnecessary files aren't committed to the repository.
- **package-lock.json:** This file is generated by npm and includes metadata about the project's dependencies and their versions. It ensures consistent installations across different environments.
- **package.json:** The package.json file holds metadata about the project and its dependencies. It also includes scripts, version information, and project settings.
- **README.md:** The README file provides essential information about the project. It explains how to set up, install dependencies, run the application, and provides a brief overview of its structure and features.

A screenshot of a file explorer window with a dark theme. The file tree is expanded to show the following structure:

- > .netlify
- ▼ netlify\functions\heroes
 - JS index.js
 - JS model.js
 - router.js
- > node_modules
- ▼ src
 - JS db.js
 - {...} serviceAccountKey.json
- .gitignore
- package-lock.json
- package.json
- README.md

Set Up Firebase Project

- Go to the Firebase Console (<https://console.firebase.google.com/>).
- Create a new project



Create a new project

× Create a project (Step 1 of 3)

Let's start with a name for
your project[®]

Project name

node-demo-app

node-demo-app-7755f

Continue

× Create a project (Step 2 of 3)

Google Analytics for your Firebase project

Google Analytics is a free and unlimited analytics solution that enables targeting, reporting, and more in Firebase Crashlytics, Cloud Messaging, In-App Messaging, Remote Config, A/B Testing, and Cloud Functions.

Google Analytics enables:

- A/B testing ⓘ
- Crash-free users ⓘ
- User segmentation & targeting across Firebase products ⓘ
- Event-based Cloud Functions triggers ⓘ
- Free unlimited reporting ⓘ

☒ Enable Google Analytics for this project
Recommended

Previous

Continue

× Create a project (Step 3 of 3)

Configure Google Analytics

[Choose or create a Google Analytics account](#) ⓘ

Demo

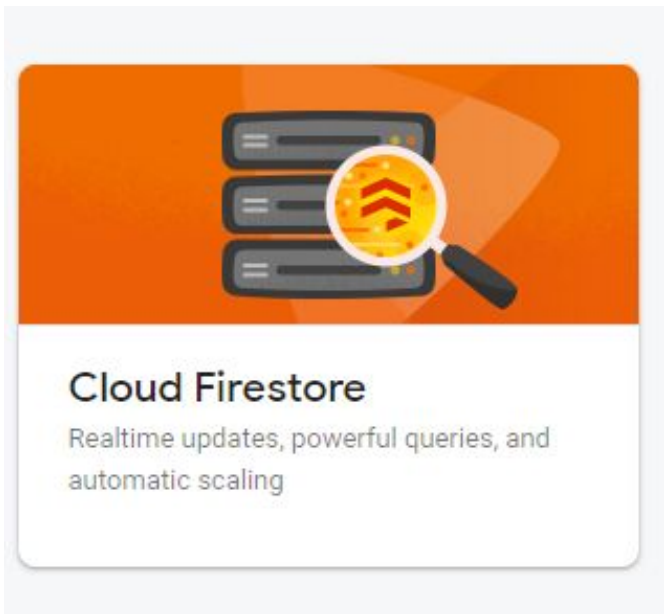
Automatically create a new property in this account ⓘ

Upon project creation, a new Google Analytics property will be created in your chosen Google Analytics account and linked to your Firebase project. This link will enable data flow between the products. Data exported from your Google Analytics property into Firebase is subject to the Firebase terms of service, while Firebase data imported into Google Analytics is subject to the Google Analytics terms of service. [Learn more](#) ⓘ

Previous

Create project

Choose a Database



Cloud Firestore is a flexible, scalable, and fully managed NoSQL database service provided by Google Cloud. It allows developers to easily store, manage, and synchronize data for web and mobile applications. With real-time capabilities, Cloud Firestore enables seamless data synchronization across devices and provides a reliable and secure solution for building modern and responsive applications. Its flexible data model, automatic scaling, and comprehensive querying capabilities make it suitable for a wide range of use cases, from small applications to large-scale, globally distributed projects.

Create database

Cloud Firestore

Realtime updates, powerful queries, and automatic scaling

Create database



Is Cloud Firestore right for you?

[Compare Databases](#)



Create database

Create database

1 Secure rules for Cloud Firestore

2 Set Cloud Firestore location


After you define your data structure, you will need to write rules to secure your data.
[Learn more](#)


☒ **Start in production mode**
Your data is private by default. Client read/write access will only be granted as specified by your security rules.

☐ **Start in test mode**
Your data is open by default to enable quick setup. However, you must update your security rules within 30 days to enable long-term client read/write access.

```
rules_version = '2';

service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if false;
    }
  }
}
```



 All third party reads and writes will be denied

Enabling Cloud Firestore will prevent you from using Cloud Datastore with this project

Cancel **Next**



Create database

Create database



Secure rules for Cloud Firestore



Set Cloud Firestore location

Your location setting is where your Cloud Firestore data will be stored.



After you set this location, you cannot change it later. Also, this location setting will be the location for your default Cloud Storage bucket.

[Learn more](#)

Cloud Firestore location

nam5 (United States)

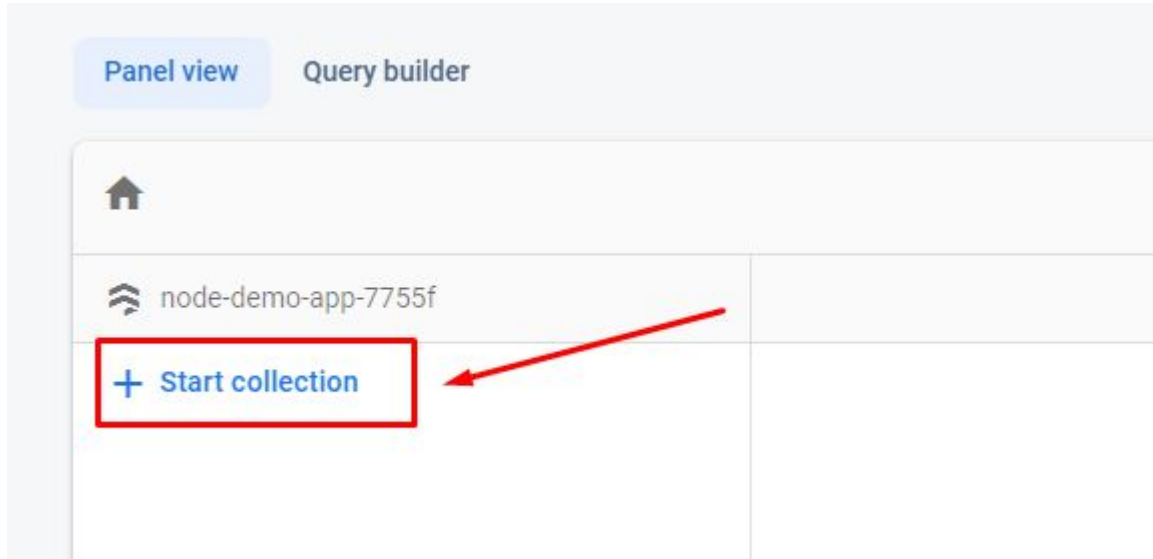
Enabling Cloud Firestore will prevent you from using Cloud Datastore with this project

Cancel

Enable



Add new collection





Add new collection

Start a collection

- 1 Give the collection an ID
- 2 Add its first document

Parent path

/

Collection ID ?

heroes

Cancel

Next



Add new Document

Start a collection



Give the collection an ID

2

Add its first document

Document parent path

/heroes

Document ID [?](#)

1

Field	Type	Value
name	= string	Wonder Woman

Field	Type	Value
superpower	= string	Amazonian powe



Cancel

Save



Add new Document

Panel view

Query builder

🏠 > heroes > 1

More in Google Cloud ▾

🏠 node-demo-app-7755f

📁 heroes

📄 1

+ Start collection

+ Add document

+ Start collection

heroes >

1 >

+ Add field

name : "Wonder Woman"

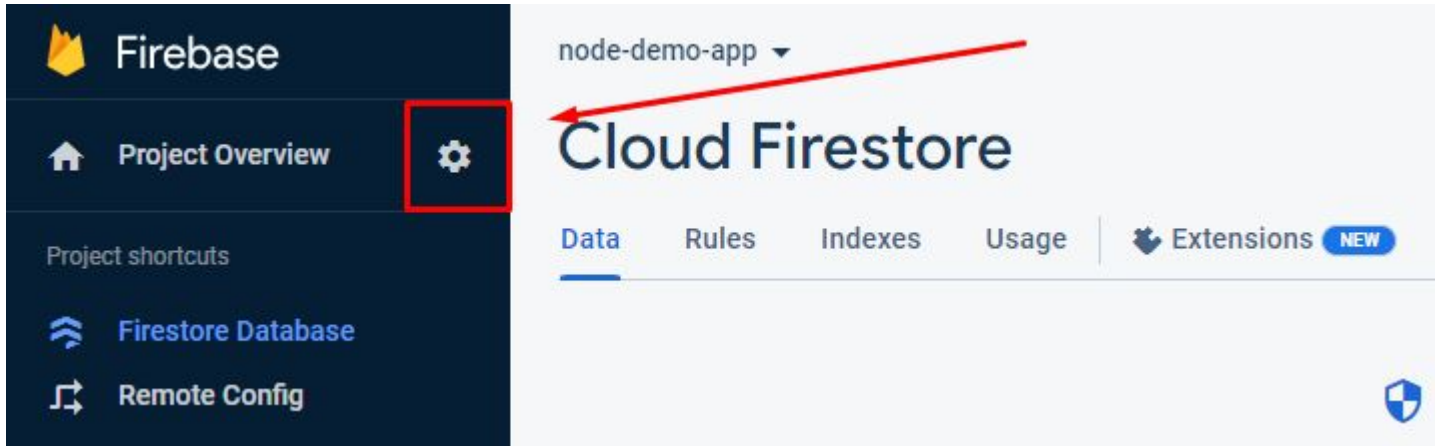
superpower : "Amazonian powers"

Generate and upload serviceAccountKey.json file

Go to the Firebase Console: Open the Firebase Console using the link:

<https://console.firebase.google.com/>

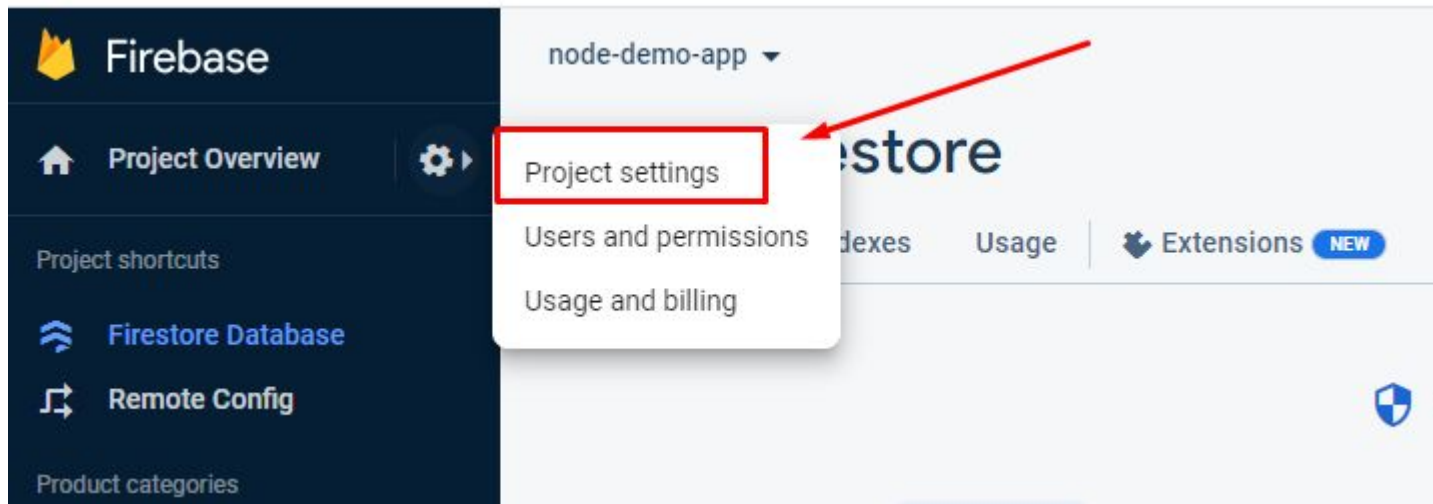
Select Your Project: If you have multiple projects, make sure you're in the project where you want to integrate Firebase.





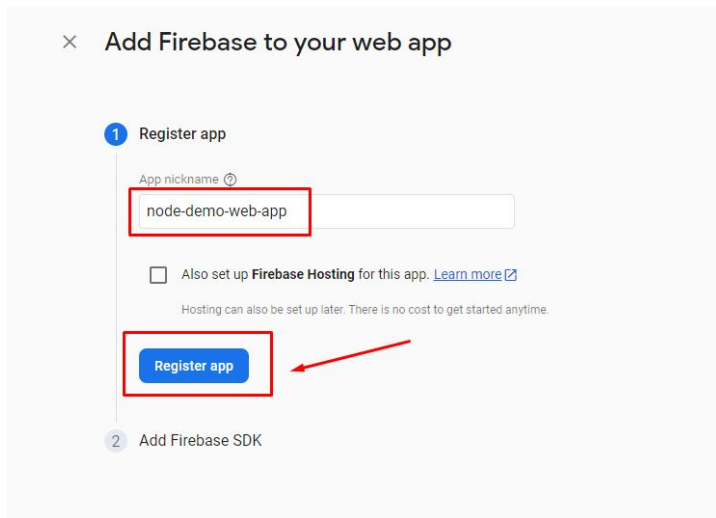
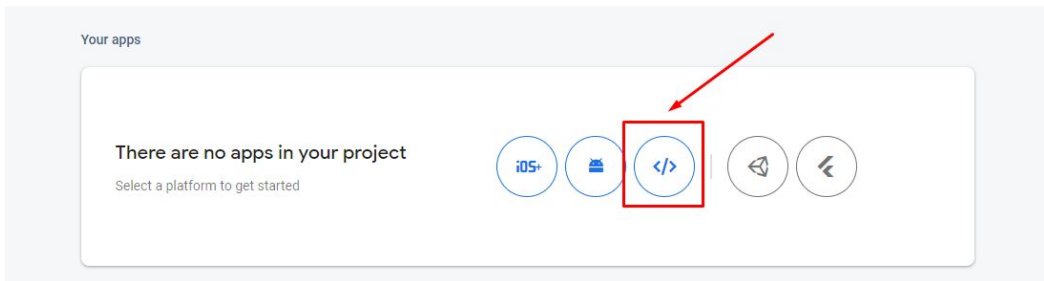
Generate and upload serviceAccountKey.json file

Navigate to Project Settings: Click on the gear icon (settings) at the top of the left sidebar to access your project settings.



Generate and upload serviceAccountKey.json file

General Tab: In the General tab, you'll see a section called "Your apps" with your Firebase project's Web App. If you haven't created a web app yet, you can do so by clicking "Add app" and selecting "Web."





Generate and upload serviceAccountKey.json file

[Manage service account permissions](#)

Firestore Admin SDK

Legacy credentials

Database secrets

All service accounts

[5 service accounts](#)

Firestore Admin SDK

Your Firebase service account can be used to authenticate multiple Firebase features, such as Database, Storage and Auth, programmatically via the unified Admin SDK. [Learn more](#)

Firebase service account
firebase-adminsdk-gsygu@node-demo-app-7755f.iam.gserviceaccount.com

Admin SDK configuration snippet

☒ Node.js ☐ Java ☐ Python ☐ Go

```
var admin = require("firebase-admin");  
  
var serviceAccount = require("path/to/serviceAccountKey.json");  
  
admin.initializeApp({  
  credential: admin.credential.cert(serviceAccount)  
});
```

Generate new private key

Go to the Firebase Console, navigate to your project settings, and then to the "Service accounts" tab. There, you can generate a new private key, which will give you a JSON file containing the credentials.

Full Project Code





Creating a Simple Express App for Netlify Serverless Function

This code sets up a simple Express app with two route handlers and exports it as a serverless function that can be deployed on Netlify. When a request is made to one of the defined routes, the app will respond with the appropriate message or data.

To use the code, you need to have the dependencies "express", "serverless-http" and "firebase-admin" installed. To install these dependencies, you can run the following command in your terminal while in the root directory of your project:

```
npm install express serverless-http firebase-admin
```



index.js

netlify > functions > heroes > **JS** index.js > ...

```
1  const express = require('express')
2  const serverless = require('serverless-http')
3  const router = require('./router');
4
5  require('dotenv').config();
6
7  const app = express()
8
9  app.use(function (req, res, next) {
10     const allowedHosts = ['charming-shortbread-286ef1.netlify.app',
11       'main--charming-shortbread-286ef1.netlify.app',
12       'localhost:8888'];
13     const host = req.headers.host;
14     console.log(`host: ${host}`)
15
16     if (allowedHosts.includes(host)) {
17       next();
18     }
19     else {
20       return res.status(405).send('Host Not Allowed');
21     }
22   });
23
24   app.use(express.json());
25   app.use('/.netlify/functions', router)
26
27   module.exports.handler = serverless(app)
```



model.js

```
netlify > functions > heroes > JS model.js > getHeroesModel
1  const firestore = require('../../../../src/db'); // Adjust the path as needed
2
3  const getHeroesModel = () => {
4    return [
5      findAll: async (callback) => {
6        try {
7          const snapshot = await firestore.collection('heroes').get();
8          const heroes = snapshot.docs.map((doc) => {
9            const data = doc.data();
10            data.id = doc.id; // Add the id field to the data
11            return data;
12          });
13          callback(heroes);
14        } catch (err) {
15          console.error('Firestore query error:', err.message);
16          callback([]);
17        }
18      },
19      create: async (name, superPower, callback) => {
20        try {
21          await firestore.collection('heroes').add({
22            name,
23            superPower,
24          });
25          callback(null, 'Hero created successfully');
26        } catch (err) {
27          console.error('Firestore query error:', err.message);
28          callback('Error creating hero');
29        }
30      },
31    ],
32  }
```




model.js

```
33   update: async (id, name, superPower, callback) => {
34     try {
35       await firestore.collection('heroes').doc(id).update({
36         name,
37         superPower,
38       });
39       callback(null, 'Hero updated successfully');
40     } catch (err) {
41       console.error('Firestore query error:', err.message);
42       callback('Error updating hero');
43     }
44   },
45
46   delete: async (id, callback) => {
47     try {
48       await firestore.collection('heroes').doc(id).delete();
49       callback(null, 'Hero deleted successfully');
50     } catch (err) {
51       console.error('Firestore query error:', err.message);
52       callback('Error deleting hero');
53     }
54   },
55 };
56 };
57
```



router.js

```
netlify > functions > heroes > router.js > router.put('/heroes/:id') callback
1  const express = require('express')
2  const getHeroesModel = require('./model');
3  const router = express.Router()
4
5  router.get('/heroes', async (req, res) => {
6    const HeroModel = getHeroesModel();
7
8    HeroModel.findAll((heroes) => {
9      res.status(200).json(heroes);
10    });
11  });
12  router.post('/heroes', async (req, res) => {
13    const { name, superPower } = req.body;
14
15    if (!name || !superPower) {
16      return res.status(400).json({ message: 'Name and superPower are required' });
17    }
18
19    const HeroModel = getHeroesModel(); // Initialize the model
20
21    HeroModel.create(name, superPower, (err, message) => {
22      if (err) {
23        console.error(err);
24        return res.status(500).json({ message: 'Error creating hero' });
25      }
26
27      return res.status(201).json({ message: 'Hero created successfully' });
28    });
29  });
```



router.js

```
31 router.put('/heroes/:id', async (req, res) => {
32   const { id } = req.params;
33   const { name, superPower } = req.body;
34   if (!name || !superPower) {
35     return res.status(400).json({ message: 'Name and superPower are required' });
36   }
37   const HeroModel = getHeroesModel(); // Initialize the model
38   HeroModel.update(id, name, superPower, (err, message) => {
39     if (err) {
40       console.error(err);
41       return res.status(500).json({ message: 'Error updating hero' });
42     }
43     return res.status(200).json({ message: 'Hero updated successfully' });
44   });
45 });
46
47 router.delete('/heroes/:id', async (req, res) => {
48   const { id } = req.params;
49   const HeroModel = getHeroesModel(); // Initialize the model
50
51   HeroModel.delete(id, (err, message) => {
52     if (err) {
53       console.error(err);
54       return res.status(500).json({ message: 'Error deleting hero' });
55     }
56     return res.status(200).json({ message: 'Hero deleted successfully' });
57   });
58 });
59
60 module.exports = router;
```



db.js

```
src >  db.js > ...  
 1  const fs = require('firebase-admin');  
 2  const serviceAccount = require('./serviceAccountKey.json');  
 3  fs.initializeApp({  
 4    credential: fs.credential.cert(serviceAccount)  
 5  });  
 6  
 7  const firestore = fs.firestore();  
 8  module.exports = firestore;
```

Move the previously downloaded
serviceAccountKey.json file to the
src folder

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

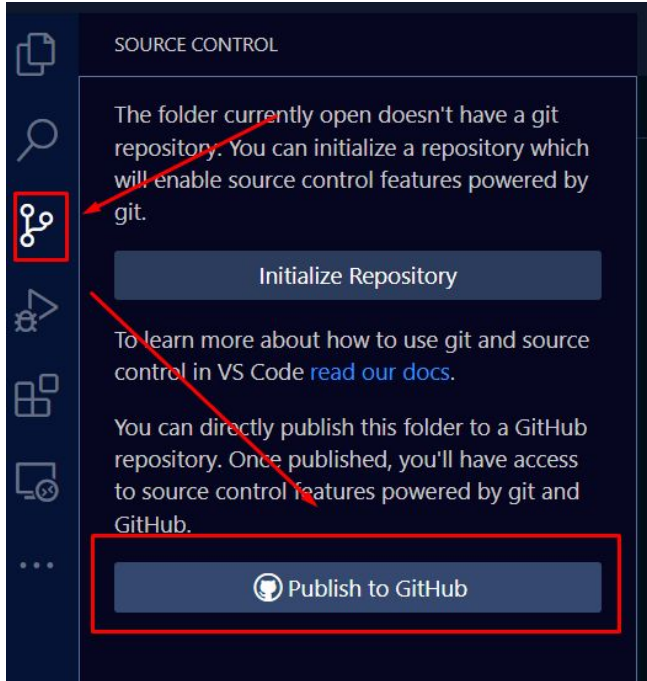
Once you have the dependencies installed, you can run the project locally using the following command:

```
netlify dev
```


GitHub Repository

Create a Private GitHub Repository for Source Control and Publish Branch

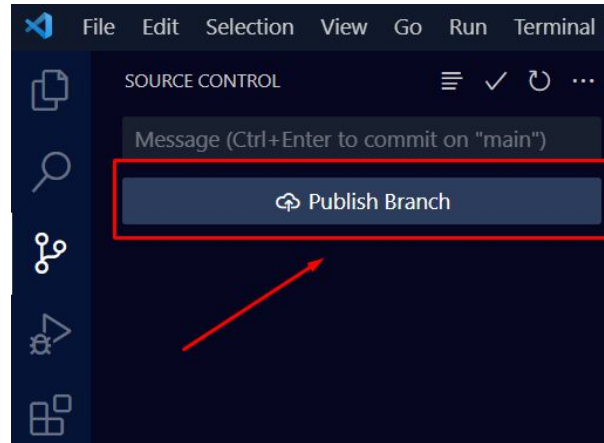
1



2



3



Deploy Project in the Netlify

Connecting Netlify to Your GitHub Account

The screenshot shows the Netlify dashboard interface. On the left, a sidebar contains navigation links: Team overview, Sites (highlighted with a red box and a red arrow), Builds, Integrations, Domains, Members, Audit log, Billing, and Team settings. The main content area has a search bar labeled 'Search sites'. Below it, two site cards are visible: 'charming-shortbread-286ef1' and 'creative-liger-e862e6', both owned by 'MarizzaMil'. A red arrow points from the 'Add new site' button (highlighted with a red box) to a dropdown menu. The dropdown menu contains four options: 'Import an existing project' (highlighted with a red box and a red arrow), 'Start from a template', 'Deploy manually', and 'Add new site ^'. At the bottom of the main content area, there is a dashed box with the text: 'Want to deploy a new site without connecting to Git? Drag and drop your site output folder here Or, [browse to upload](#)'.

Netlify / M MarizzaMil

Search anything...

News Support

Team overview

Sites

Builds

Integrations

Domains

Members

Audit log

Billing

Team settings

Search sites

charming-shortbread-286ef1
Deploys from [GitHub](#)
Owned by MarizzaMil
Published on Aug 22 (2 days ago)

creative-liger-e862e6
Deploys from [GitHub](#)
Owned by MarizzaMil
Published on Aug 20 (4 days ago)

Add new site ^

Import an existing project

Start from a template

Deploy manually

Want to deploy a new site without connecting to Git?
Drag and drop your site output folder here
Or, [browse to upload](#)



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

Let's deploy your project.

 Deploy with GitHub

 Deploy with GitLab

 Deploy with Bitbucket

 Deploy Azure DevOps



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.

Let's deploy your project.



MarizzaMil ▾



🔍 Search your repos



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:

Can't see your repo here? [Configure the Netlify app on GitHub.](#)





Configure site and deploy

Repository access

☐ All repositories

This applies to all current *and* future repositories owned by the resource owner.
Also includes public repositories (read-only).

☒ Only select repositories

Select at least one repository.
Also includes public repositories (read-only).

 Select repositories ▾

Selected 2 repositories.



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.

Let's deploy your project.

Review configuration for netlify-demo-app

Deploy as MarizzaMil on MarizzaMil team from main branch

Team

MarizzaMil



Branch to deploy

main



Deploy netlify-demo-app





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

- i** 2 new files uploaded
2 assets changed.
- i** No redirect rules processed
This deploy did not include any redirect rules. [Learn more about redirects](#) ↗
- i** No header rules processed
This deploy did not include any header rules. [Learn more about headers](#) ↗
- i** All linked resources are secure
Congratulations! No insecure mixed content found in your files.
- i** 1 function deployed
We have deployed 1 function. [Visit your Functions](#) for more information.
- i** 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

Today at 2:38 PM

Production: main@HEAD ↕

Deployed [Functions](#)

Open production deploy ↗

Lock to stop auto publishing

Options ▾

[Permalink](#)

Test the API using Postman

GET REQUEST

GET

https://<your_address>/.netlify/functions/heroes

Send

ParamsAuthorizationHeaders (6)BodyPre-request ScriptTestsSettingsCookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
	Key	Value	Description		

BodyCookiesHeaders (10)Test Results

Status: 200 OKTime: 3.00 sSize: 422 BSave Response

PrettyRawPreviewVisualizeJSON

```
1 {  
2   {  
3     "name": "Wonder Woman",  
4     "superpower": "Amazonian powers",  
5     "id": "1"  
6   }  
7 }
```

POST REQUEST

POST

https://<your_address>/.netlify/functions/heroes

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Cookies

☐ none

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

☐ GraphQL

☒ JSON

Beautify

```
1 {
2   "name": "New Hero",
3   "superPower": "Super Strength"
4 }
```

Body

Cookies

Headers (10)

Test Results

Status: 201 Created

Time: 374 ms

Size: 400 B

Save Response

Pretty

Raw

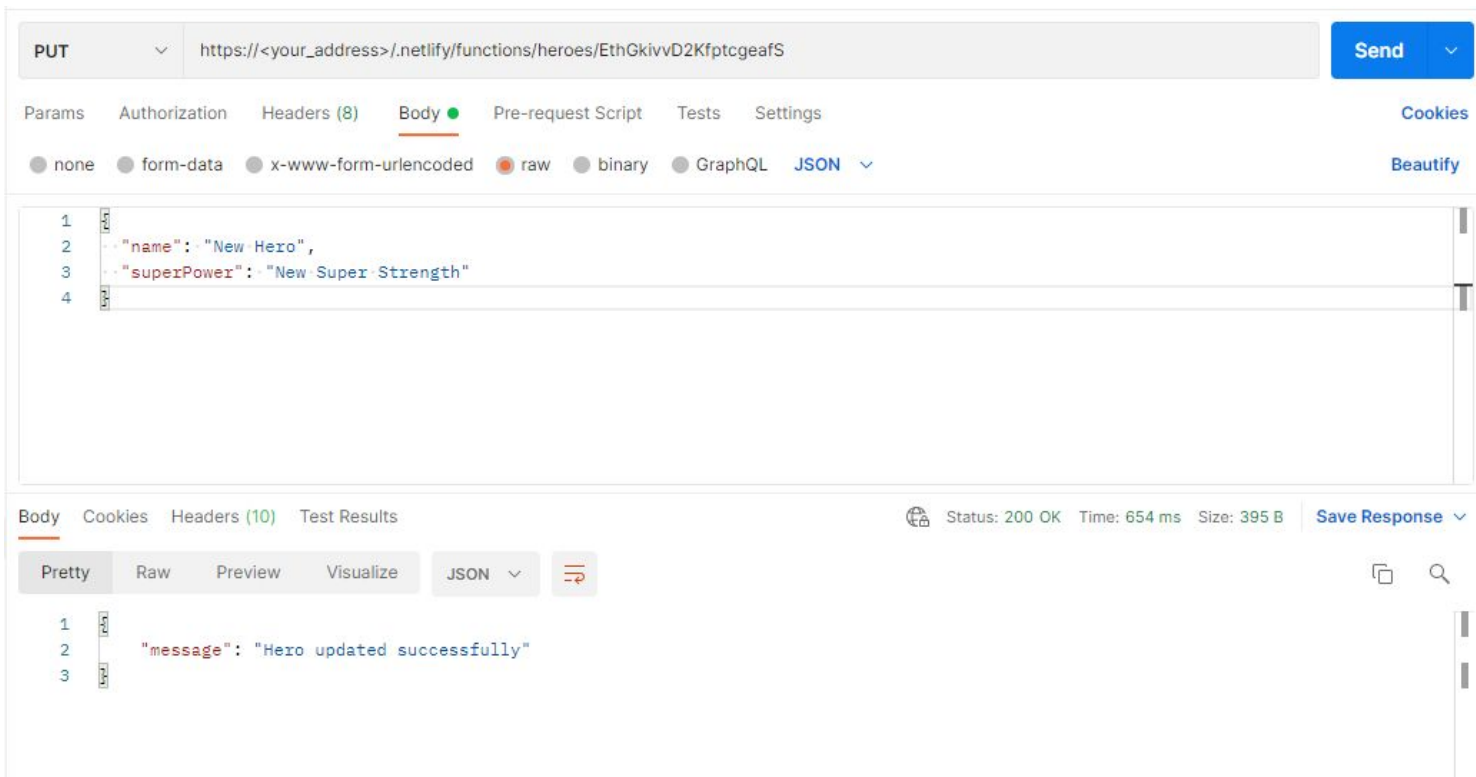
Preview

Visualize

JSON

```
1 {
2   "message": "Hero created successfully"
3 }
```


UPDATE REQUEST



The image shows a REST client interface with a PUT request and its response.

Request:

- Method: PUT
- URL: `https://<your_address>/netlify/functions/heroes/EthGkivvD2KfptcgeafS`
- Body:

```
1 {  
2   "name": "New Hero",  
3   "superPower": "New Super Strength"  
4 }
```

Response:

- Status: 200 OK
- Time: 654 ms
- Size: 395 B
- Body:

```
1 {  
2   "message": "Hero updated successfully"  
3 }
```

DELETE REQUEST

DELETE

https://<your_address>/.netlify/functions/heroes/EthGkivvD2KfptcgeafS

Send

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body

Cookies

Headers (10)

Test Results

Status: 200 OK

Time: 397 ms

Size: 395 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

```
1 {
2   "message": "Hero deleted successfully"
3 }
```



Conclusion

In conclusion, you've successfully navigated through the process of creating a Node.js application deployed on Netlify, equipped with a Firebase Cloud Firestore database to facilitate CRUD operations. This tutorial aimed to provide a comprehensive guide, covering everything from setting up your development environment to building a functional application with real-time data capabilities.

By following the steps outlined in this tutorial, you've gained valuable insights into:

- **Environment Setup:** You've configured your development environment by creating a Firebase project and setting up the necessary dependencies.
- **Database Creation:** You've established a Firestore database, a scalable and NoSQL solution, to store your application's data.
- **API Development:** You've constructed an API using Express.js, enabling your application to perform Create, Read, Update, and Delete operations seamlessly.
- **Model-Router Architecture:** You've structured your project with a clear separation of concerns, using models and routers for enhanced code organization and maintainability.
- **Real-Time Data Handling:** By integrating Firestore, you've harnessed the power of real-time data synchronization, ensuring your application remains responsive and up-to-date.
- **Deployment on Netlify:** You've successfully deployed your Node.js application on Netlify, allowing it to be accessible to users online.

Throughout this journey, you've encountered and overcome challenges, such as configuring Firestore, implementing authentication, and ensuring your code adheres to best practices. This hands-on experience has provided you with practical insights that can be extended to future projects.

As you continue to develop your skills, remember that the journey of learning is ongoing. You now possess a foundation that enables you to build upon this project, explore more advanced features of Firestore and Netlify, and further refine your application based on evolving needs. Armed with the knowledge gained here, you're better equipped to embark on new coding endeavors and create innovative solutions that address real-world challenges.