

There are several ways that you can deploy your app so that you can share it with the world, including using GitHub pages or using one of many service providers. A really excellent way to do this is to use Azure Static Web Apps. In this assignment, build your web app and deploy it to the cloud by following [these instructions](#) or watching [these videos](#). A benefit of using Azure Static Web Apps is that you can hide any API keys in the portal, so take this opportunity to refactor your subscriptionKey as a variable and store it in the cloud.

### Answer

- In this assignment, I will use [app.netlify.com](https://app.netlify.com) instead of Azure because there are some problems with creating an account on Azure.

## 1. Tools & Technologies Used

**Netlify:** Used for deploying the frontend and serverless backend functions

**GitHub:** Used for hosting source code and version control

**HTML/CSS/JavaScript:** Used to create the frontend

**Netlify Functions:** Used to create serverless functions that securely access API keys

**Environment Variables:** Used to securely store the **SUBSCRIPTION\_KEY**

## 2. Project Structure

```
/my-project
├─ index.html
├─ netlify.toml
└─ netlify/
    └─ functions/
        └─ callapi.js
```

**index.html:** A simple webpage with a button to call the API

**netlify/functions/callapi.js:** A Netlify serverless function that reads the **SUBSCRIPTION\_KEY** from environment variables

**netlify.toml:** Configuration file telling Netlify how to build and where to find functions

## 3. Environment Variable Setup

In Netlify dashboard:

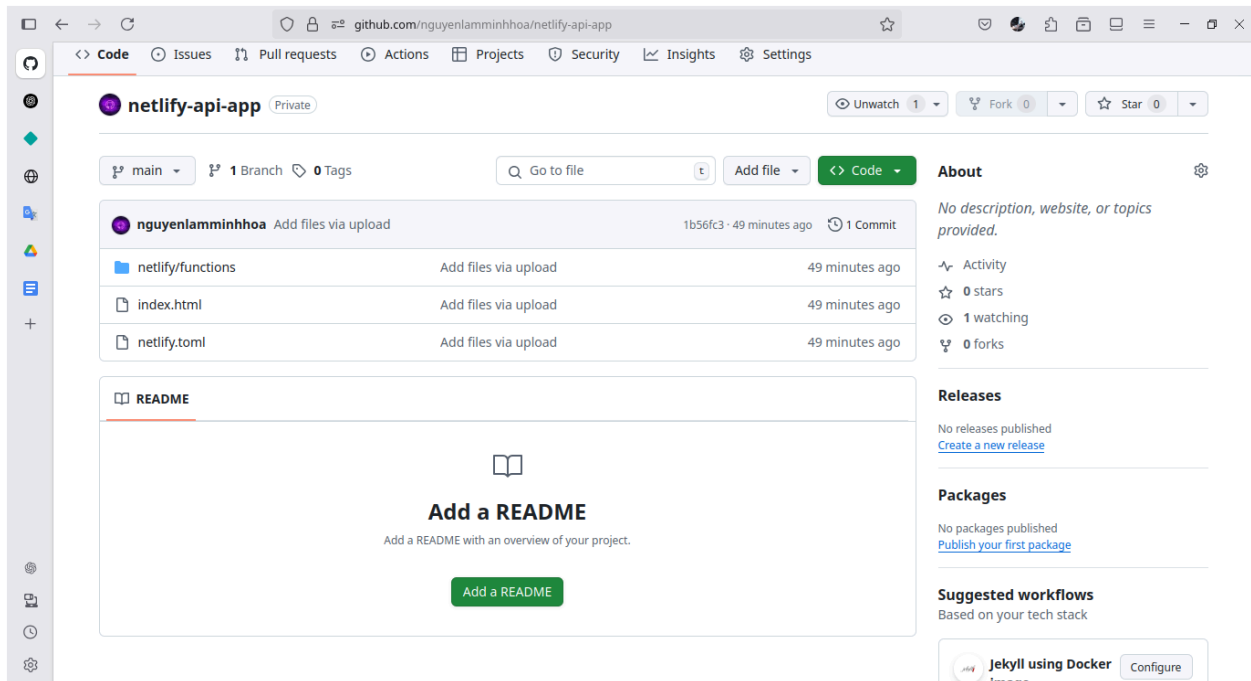
- **Key:** **SUBSCRIPTION\_KEY**
- **Value:** [actual API key or dummy for testing]

This key is never stored in the GitHub repository

## 4. Deployment Process

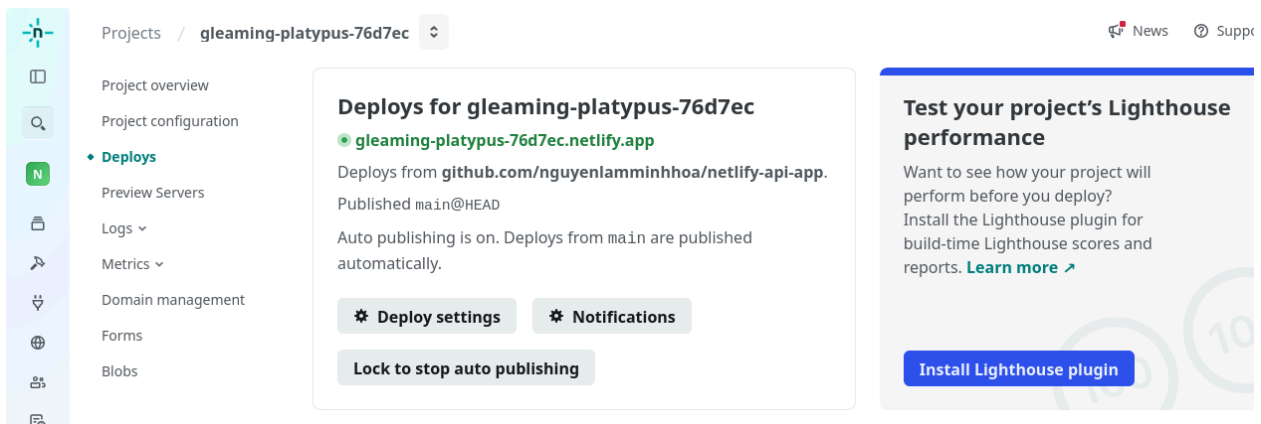
### 4.1 Create GitHub repository

- Upload project files via GitHub web interface

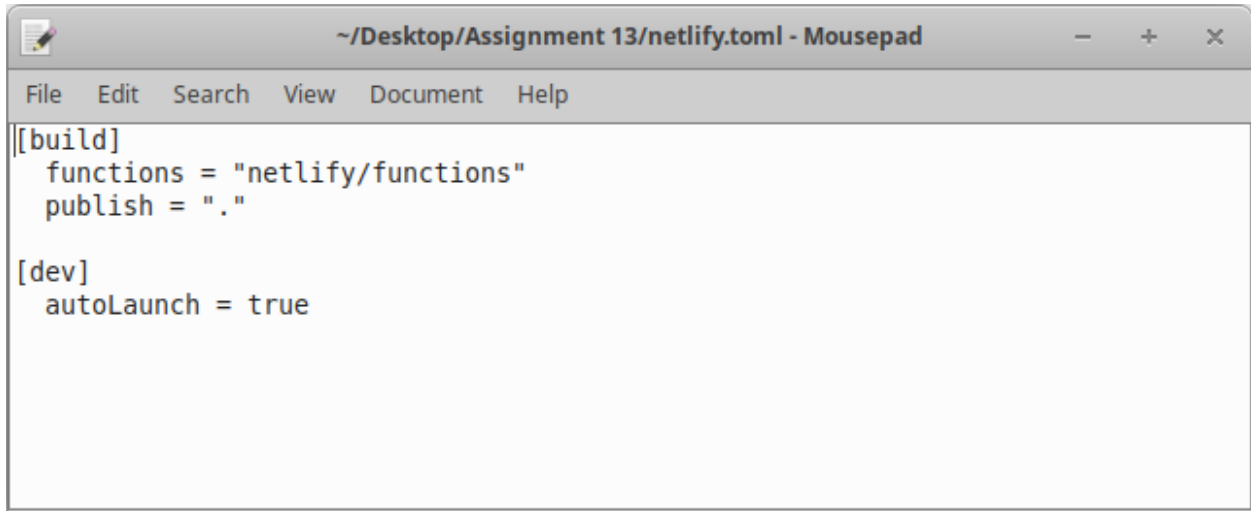


### 4.2 Deploy to Netlify

- Connect GitHub repo to Netlify



- Configure **build** and **publish** settings using **netlify.toml**



```
[[build]
  functions = "netlify/functions"
  publish = "."

[dev]
  autoLaunch = true
```

#### 4.3 Add environment variable

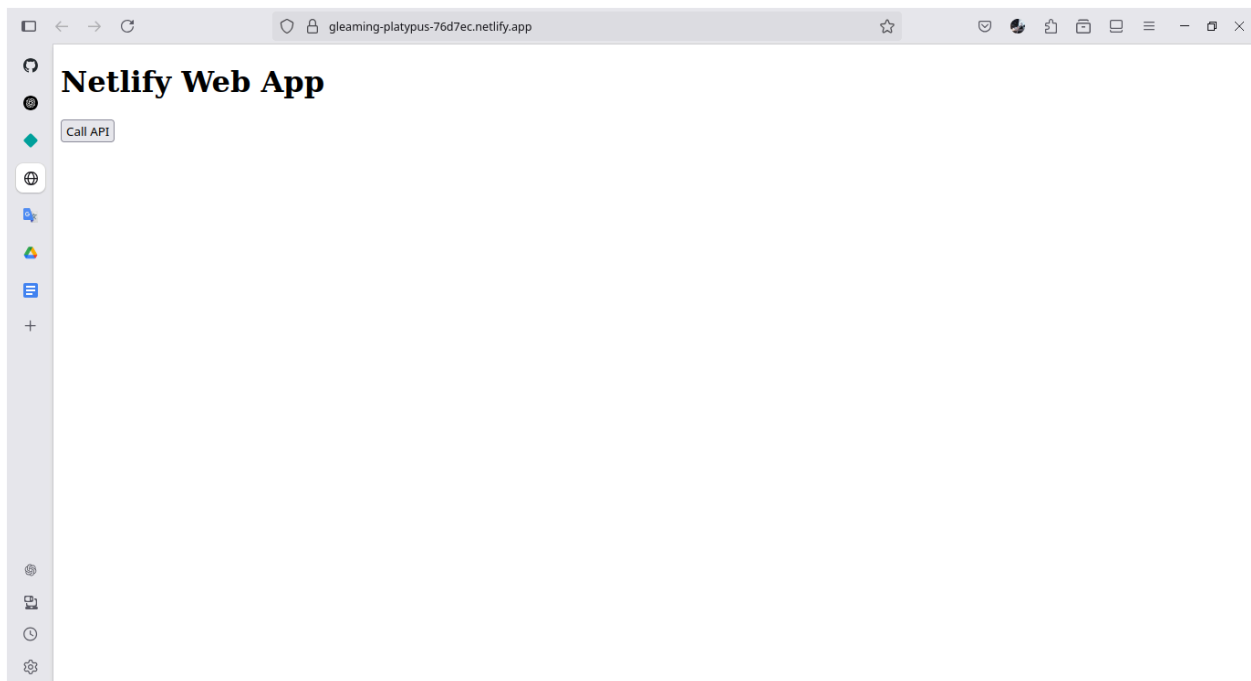
- Inside Netlify's **Site settings** → **Build & deploy** → **Environment**

#### 4.4 Trigger redeploy

- To make sure Netlify uses updated environment variables

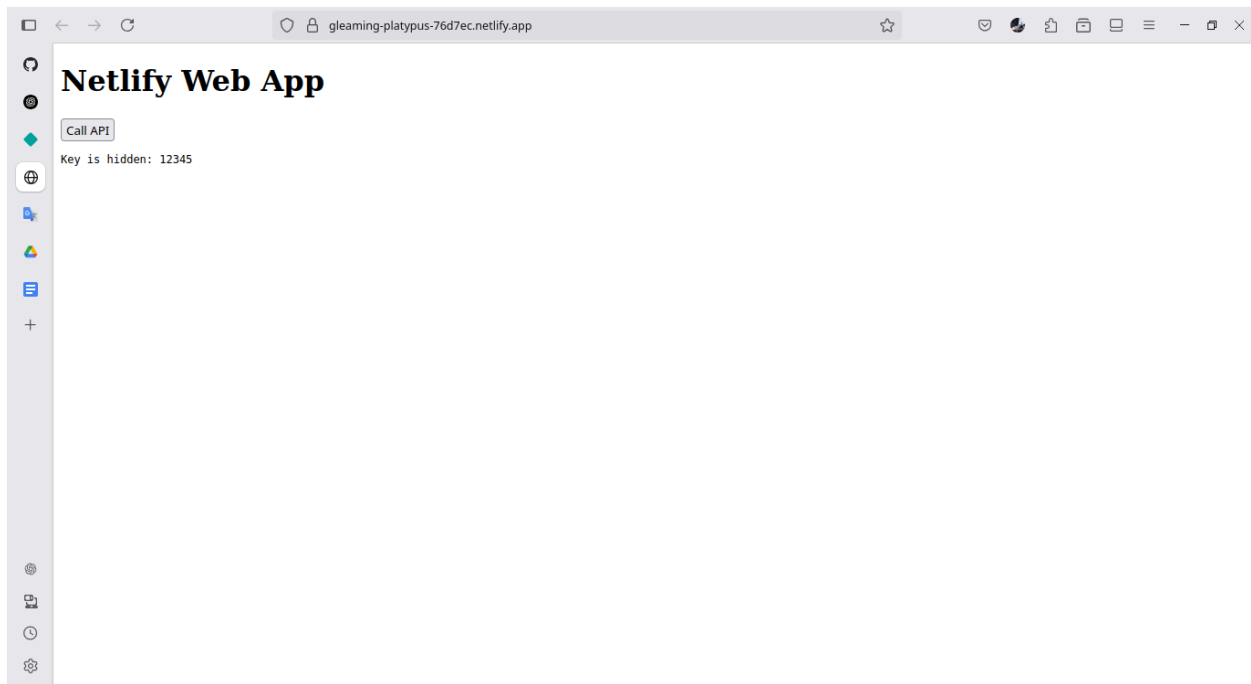
#### 4.5 Test the application

- Visit live site → Click button “**Call API**” → See result from API
- Link: <https://gleaming-platypus-76d7ec.netlify.app/>



## 5. Result

- The web app is successfully deployed on Netlify.
- When the button is clicked, the `callapi.js` function is triggered and reads the `SUBSCRIPTION_KEY` from environment variables.
- The subscription key is hidden from the client side, fulfilling security requirements.



## 6. Conclusion

In conclusion, this project demonstrates a successful deployment of a web application using **Netlify**, integrating both frontend and backend logic through serverless functions. By utilizing **environment variables**, sensitive information such as the API `subscriptionKey` is securely stored in the cloud, preventing exposure in the public GitHub repository or frontend code.

This approach not only ensures security but also showcases best practices in modern web development and cloud deployment. The ability to trigger backend logic via a frontend button click using Netlify Functions proves the flexibility and power of serverless architecture for small to medium-scale applications.