

# node Code to Cloud



ZERO



TO



HERO

with

 [GitHub Copilot](#)

[aka.ms/zero2hero](http://aka.ms/zero2hero)

Please note that this content is only informative and intended to inspire you on using GitHub Copilot to get the job done! You might get different results since it's using a nondeterministic LLM.



[github.com/vieiraae](https://github.com/vieiraae)



[linkedin.com/in/vieira](https://linkedin.com/in/vieira)



# HERO Goals

1. Generate a new nodeJS App from scratch
2. Create the VS Code workspace
3. Run/test backend and frontend
4. Deploy to Azure App Service



node Code  
to  
Cloud



Azure App  
Service





# GitHub Copilot Superpowers

- ✓ Execute complex tasks with simple prompts to your personal assistant
- ✓ Context aware environment to keep you in the flow
- ✓ Natural language powers your coding experience
- ✓ Proven impact on your productivity & happiness:

**74%** focus on more satisfying work

**88%** feel more productive

**96%** are faster with repetitive tasks

<https://github.com/features/copilot>



# 0

# Environment Setup



- [Activate GitHub Copilot](#)
- [Join the waitlist \(if needed\)](#)



- [Install Visual Studio Code](#)
- Add the extensions
  - [GitHub Copilot](#)
  - [GitHub Copilot Chat](#)
  - [Azure Tools](#)



- [Install Azure CLI \(or use the Azure Cloud Shell\)](#)

# 1

# Generate the App

- Type in the chat:

 */createWorkspace build a nodejs react app with a backend web api that screen scraps stock information from Yahoo finance and a front-end that loads the web api and displays the stocks in a table with symbol, name and last price*



GitHub Copilot



Sure, here's a proposed directory structure for a Node.js React app with a backend web API that screen scraps stock information from Yahoo finance and a front-end that loads the web API and displays the stocks in a table with symbol, name and last price:

```
✓ backend
  {} package.json
  JS server.js
  JS scraper.js
  ⓘ README.md

✓ frontend
  ✓ public
    <> index.html
  ✓ src
    ✓ components
      JS StockTable.js
      JS StockRow.js
    ✓ utils
      JS api.js
      JS App.js
      JS index.js
    {} package.json
    ⓘ README.md
    {} package.json
```

# 2

# Create the workspace

The `backend` folder contains the Node.js web API that screen scraps stock information from Yahoo finance. The `server.js` file is the entry point for the web API, while the `scraper.js` file contains the code for screen scraping the stock information. The `frontend` folder contains the React app that loads the web API and displays the stocks in a table with symbol, name and last price. The `public` folder contains the `index.html` file and the `favicon.ico` file. The `src` folder contains the `App.js` file, which is the entry point for the React app, the `components` folder, which contains the `StockTable.js` and `StockRow.js` files, and the `utils` folder, which contains the `api.js` file for making API requests to the backend. The `package.json` file in the root directory contains the dependencies for both the backend and frontend, while the `README.md` files provide information about the project.

[Create Workspace](#)

- Check the files generated. If they are empty repeat the prompt
- Click in the Create Workspace button and select a new folder to store the files



*How to test my app*



GitHub Copilot



To test your app, you can follow these steps:

1. Open a terminal window and navigate to the root directory of your project.
2. Start the backend server by running `npm start` in the `backend` directory.
3. Open another terminal window and navigate to the root directory of your project.
4. Start the frontend server by running `npm start` in the `frontend` directory.
5. Open a web browser and navigate to `http://localhost:3001` to view the app.
6. Open the developer console in your web browser by pressing F12.

# 3 Run & test the backend

- Open the backend/scrapper.js file and check the URL and the expressions to find the properties. Below is a working example:

backend > **JS** scrapper.js > ...

```
1  const cheerio = require('cheerio');
2  const fetch = require('node-fetch');
3
4  const scrapeStocks = async () => {
5      const url = 'https://finance.yahoo.com/most-active';
6      const response = await fetch(url);
7      const html = await response.text();
8      const $ = cheerio.load(html);
9      const stocks = [];
10
11     $('table tbody tr').each((i, el) => {
12         const symbol = $(el).find('td:nth-child(1)').text();
13         const name = $(el).find('td:nth-child(2)').text();
14         const lastPrice = $(el).find('td:nth-child(3)').text();
15         stocks.push({ symbol, name, lastPrice });
16     });
17
18     return stocks;
19 };
20
21 module.exports = { scrapeStocks };
```

# 3 Run & test the backend

- Open a terminal and type: cd backend; npm install; npm start
- If you get a MODULE\_NOT\_FOUND error is because there are missing modules in the backend/package.json. To fix open the backend/package.json file and add the missing module (Copilot helps with the module versions).
- Open <http://localhost:{PORT}/api/stocks> and test the app. You should get a json response like the following:

```
← → ⌂ http://localhost:5000/api/stocks [?]
```

```
[{"symbol": "TSLA", "name": "Tesla, Inc.", "lastPrice": "242.68"}, {"symbol": "AAL", "name": "American Airlines Group Inc.", "lastPrice": "11.36"}, {"symbol": "BAC", "name": "Bank of America Corporation", "lastPrice": "27.31"}, {"symbol": "NVDA", "name": "NVIDIA Corporation", "lastPrice": "421.96"}, {"symbol": "LCID", "name": "Lucid Group, Inc.", "lastPrice": "4.5300"}, {"symbol": "NIO", "name": "NIO Inc.", "lastPrice": "7.91"}, {"symbol": "AMD", "name": "Advanced Micro Devices, Inc.", "lastPrice": "102.17"}, {"symbol": "AAPL", "name": "Apple Inc.", "lastPrice": "175.84"}, {"symbol": "SWN", "name": "Southwestern Energy Company", "lastPrice": "7.25"}, {"symbol": "RIVN", "name": "Rivian Automotive, Inc.", "lastPrice": "18.09"}, {"symbol": "PLTR", "name": "Palantir Technologies Inc.", "lastPrice": "17.20"}, {"symbol": "F", "name": "Ford Motor Company", "lastPrice": "11.75"}, {"symbol": "AMZN", "name": "Amazon.com, Inc.", "lastPrice": "128.13"}, {"symbol": "T", "name": "AT&T Inc.", "lastPrice": "14.32"}, {"symbol": "CCL", "name": "Carnival Corporation & plc", "lastPrice": "11.69"}, {"symbol": "LU", "name": "Lufax Holding Ltd", "lastPrice": "0.9464"}, {"symbol": "UAL", "name": "United Airlines Holdings, Inc.", "lastPrice": "36.24"}, {"symbol": "NOK", "name": "Nokia Oyj", "lastPrice": "3.3900"}, {"symbol": "SPR", "name": "Spirit AeroSystems Holdings, Inc.", "lastPrice": "21.16"}, {"symbol": "MS", "name": "Morgan Stanley", "lastPrice": "74.88"}, {"symbol": "PLUG", "name": "Plug Power Inc.", "lastPrice": "7.00"}, {"symbol": "PBR", "name": "Petróleo Brasileiro S.A. - Petrobras", "lastPrice": "16.52"}, {"symbol": "PFE", "name": "Pfizer Inc.", "lastPrice": "31.41"}, {"symbol": "VALE", "name": "Vale S.A.", "lastPrice": "12.90"}, {"symbol": "SOFI", "name": "SoFi Technologies, Inc.", "lastPrice": "7.77"}]
```

# 3 Run & test the frontend

- ❑ Open the frontend/src/utils/api.js and fix to include the full URL for the backend API
- ❑ Open the frontend/src/components/StockRow.js and fix to reference the properties. Below is a working example:

frontend > src > components > **JS** StockRow.js > [🔗] default

```
1 import React from 'react';
2
3 function StockRow(props) {
4     const { stock } = props;
5
6     return (
7         <tr>
8             <td>{stock.symbol}</td>
9             <td>{stock.name}</td>
10            <td>{stock.lastPrice}</td>
11        </tr>
12    );
13}
14
15 export default StockRow;
```

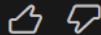
- ❑ Open a new terminal (keep the backend running) and type:  
cd backend; npm install; npm start
- ❑ If you get a MODULE\_NOT\_FOUND error is because there are missing modules in the frontend/package.json. To fix open the frontend/package.json file and add the missing module (Copilot helps with the module versions).
- ❑ If you get a ERR\_OSSL\_EVP\_UNSUPPORTED error, set the environment variable NODE\_OPTIONS with the value "--openssl-legacy-provider"

# 3 Run & test the frontend

- ❑ If successful you should open a new browser tab with the stocks
- ❑ Type in the chat:

 *change the frontend to use material ui*

 GitHub Copilot



Sure, here's an updated version of `StockRow.js` that uses Material UI:

```
import React from 'react';
import { TableRow, TableCell } from '@material-ui/core';

function StockRow(props) {
  const { stock } = props;

  return (
    <TableRow>
      <TableCell>{stock.symbol}</TableCell>
      <TableCell>{stock.name}</TableCell>
      <TableCell>{stock.lastPrice}</TableCell>
    </TableRow>
  );
}

export default StockRow;
```

Insert at

In this version, we've replaced the `tr` and `td` elements with the `TableRow` and `TableCell` components from Material UI. We've also imported these components at the top of the file using destructuring. This should give your app a more polished look and feel.

# 3 Run & test the frontend

- Type in the chat:

 add a column with a buy link to the fidelity website

 GitHub Copilot



Sure, here's an updated version of `StockRow.js` that adds a column with a buy link to the Fidelity website:

```
import React from 'react';
import { TableRow, TableCell } from '@material-ui/core';

function StockRow(props) {
  const { stock } = props;

  return (
    <TableRow>
      <TableCell>{stock.symbol}</TableCell>
      <TableCell>{stock.name}</TableCell>
      <TableCell>{stock.lastPrice}</TableCell>
      <TableCell>
        <a href={`https://www.fidelity.com/trading/order-routing/stock/quotes`}>
          {stock.symbol}
        </a>
      </TableCell>
    </TableRow>
  );
}

export default StockRow;
```

In this version, we've added a new `TableCell` component that contains an anchor tag (`<a>`) with an `href` attribute that points to the Fidelity website's order routing page for the stock symbol. We've also set the `target` attribute to `_blank` and the `rel` attribute to `noopener noreferrer` to ensure that the link opens in a new tab and to prevent security vulnerabilities. This should allow users to easily buy the

# 3 Run & test the frontend

- The browser content should automatically update and show the following:

## Stocks

Symbol	Name	Last Price	
TSLA	Tesla, Inc.	242.68	<a href="#">Buy</a>
AAL	American Airlines Group Inc.	11.36	<a href="#">Buy</a>
BAC	Bank of America Corporation	27.31	<a href="#">Buy</a>
NVDA	NVIDIA Corporation	421.96	<a href="#">Buy</a>
LCID	Lucid Group, Inc.	4.5300	<a href="#">Buy</a>
NIO	NIO Inc.	7.91	<a href="#">Buy</a>
AMD	Advanced Micro Devices, Inc.	102.17	<a href="#">Buy</a>
AAPL	Apple Inc.	175.84	<a href="#">Buy</a>
SWN	Southwestern Energy Company	7.25	<a href="#">Buy</a>
RIVN	Rivian Automotive, Inc.	18.09	<a href="#">Buy</a>
PLTR	Palantir Technologies Inc.	17.20	<a href="#">Buy</a>
F	Ford Motor Company	11.75	<a href="#">Buy</a>

# 4 Deploy backend to Azure

□ Type in the chat:

💬 *how to deploy my backend to Azure App Service using just the "az webapp up" command with the name hero-stocks-backend and basic sku in a resource group named hero-stocks in the westeurope region*



GitHub Copilot

Sure, here are the steps to deploy your backend to Azure App Service using just the `az webapp up` command with the name `hero-stocks-backend`, basic SKU, and resource group named `hero-stocks` in the `westeurope` region:

1. Open a terminal window and navigate to the root directory of your project.
2. Install the Azure CLI by following the instructions on the official website: <https://docs.microsoft.com/en-us/cli/azure/install-azure-cli>
3. Log in to your Azure account by running `az login` in the terminal and following the prompts.
4. Create a new resource group for your app by running `az group create --name hero-stocks --location westeurope` in the terminal.
5. Run the `az webapp up` command with the following parameters:
  - `--name hero-stocks-backend` to set the name of the app to `hero-stocks-backend`.
  - `--sku B1` to set the SKU to the basic tier.
  - `--location westeurope` to set the location to `westeurope`.
  - `--resource-group hero-stocks` to set the resource group to `hero-stocks`.
  - `--runtime "NODE|14-lts"` to set the runtime to Node.js 14 LTS.

The full command should look like this:

```
az webapp up --name hero-stocks-backend --sku B1 --location westeurope --resource-group hero
```

6. Wait for the deployment to complete. This may take a few minutes.
7. Verify that your backend is running by navigating to <https://hero-stocks-backend.azurewebsites.net> in a web browser. Replace `hero-stocks-backend` with the name of your app.

# 4 Deploy backend to Azure

- In a terminal go to the backend folder and run the az command to create the resource group and then the az webapp up command to create the app service and deploy the backend
- Open the URL and test your deployed backend

← ⌂ <https://hero-stocks-backend.azurewebsites.net/api/stocks>

```
1 [ [ 
2   { 
3     "symbol": "TSLA",
4     "name": "Tesla, Inc.",
5     "lastPrice": "242.68"
6   },
7   {
8     "symbol": "AAL",
9     "name": "American Airlines Group Inc.",
10    "lastPrice": "11.36"
11  },
12  {
13    "symbol": "BAC",
14    "name": "Bank of America Corporation",
15    "lastPrice": "27.31"
16  },
17  {
18    "symbol": "NVDA",
19    "name": "NVIDIA Corporation",
20    "lastPrice": "421.96"
21  },
22  {
23    "symbol": "LCID",
24    "name": "Lucid Group, Inc.",
25    "lastPrice": "4.5300"
26  },
27  {
28    "symbol": "NIO",
29    "name": "NIO Inc.",
30    "lastPrice": "7.91"
31  },
32  {
33    "symbol": "AMD",
34    "name": "Advanced Micro Devices, Inc.",
35    "lastPrice": "102.17"
36  },
37  {
38    "symbol": "AAPL",
39    "name": "Apple Inc.",
40    "lastPrice": "175.84"
41  },
42  {
43    "symbol": "SWN",
44    "name": "Southwestern Energy Company",
45    "lastPrice": "7.25"
46  }, ] ]
```

# 4 Deploy frontend to Azure

□ Copy the backend full URL and replace in the frontend/src/utils/api.js to point to the new deployed backend. An environment file or the Azure App Configuration should be used to store the URL to be more production ready and convenient.

□ Type in the chat:

 *how to deploy my frontend to Azure App Service using just the "az webapp up" command with the name hero-stocks-frontend and basic sku in the resource group named hero-stocks already created and reusing the same app service plan from the backend*

 GitHub Copilot



Sure, here are the steps to deploy your frontend to Azure App Service using just the `az webapp up` command with the name `hero-stocks-frontend`, basic SKU, and resource group named `hero-stocks` already created:

1. Open a terminal window and navigate to the root directory of your project.
2. Install the Azure CLI by following the instructions on the official website: <https://docs.microsoft.com/en-us/cli/azure/install-azure-cli>
3. Log in to your Azure account by running `az login` in the terminal and following the prompts.
4. Run the `az webapp up` command with the following parameters:
  - `--name hero-stocks-frontend` to set the name of the app to `hero-stocks-frontend`.
  - `--sku B1` to set the SKU to the basic tier.
  - `--location westeurope` to set the location to `westeurope`.
  - `--resource-group hero-stocks` to set the resource group to `hero-stocks`.
  - `--html` to specify that the app is an HTML app.
  - `--dryrun` to preview the deployment without actually deploying it.

The full command should look like this:

# 4 Deploy frontend to Azure

- In a terminal go to the frontend folder and run the az webapp up command to create the app service and deploy the frontend. If the az webapp up command has --html and --dry-run arguments, remove them.
- Open the URL and test your deployed frontend

← ⌂ ⌂ https://hero-stocks-frontend.azurewebsites.net

## My Stock App

Symbol	Name	Last Price	
TSLA	Tesla, Inc.	242.68	<a href="#">Buy</a>
AAL	American Airlines Group Inc.	11.36	<a href="#">Buy</a>
BAC	Bank of America Corporation	27.31	<a href="#">Buy</a>
NVDA	NVIDIA Corporation	421.96	<a href="#">Buy</a>
LCID	Lucid Group, Inc.	4.5300	<a href="#">Buy</a>
NIO	NIO Inc.	7.91	<a href="#">Buy</a>
AMD	Advanced Micro Devices, Inc.	102.17	<a href="#">Buy</a>
AAPL	Apple Inc.	175.84	<a href="#">Buy</a>
SWN	Southwestern Energy Company	7.25	<a href="#">Buy</a>
RIVN	Rivian Automotive, Inc.	18.09	<a href="#">Buy</a>

# Reviewing



# GitHub Copilot

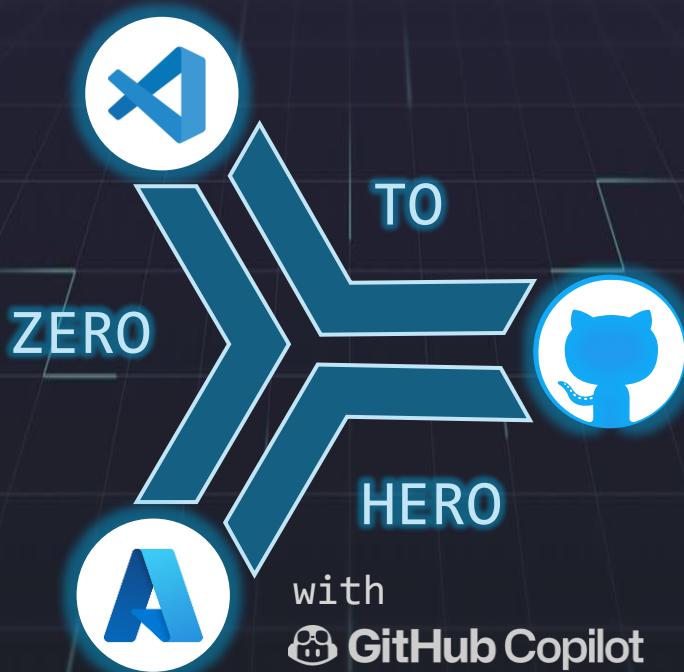
- 👍 The entire app with a backend web API and a React frontend was automatically created with the `/createWorkspace` command
- 👍 Assisted to run and test the App in the local environment
- 👍 Changed the UI and functionality
- 👍 Generated the all-in-one command to deploy the backend and the frontend to the Azure App Service



[aka.ms/zero2hero](https://aka.ms/zero2hero)



Thumbs up  
to pump  
for more



[github.com/vieiraae](https://github.com/vieiraae)



[linkedin.com/in/vieira](https://linkedin.com/in/vieira)