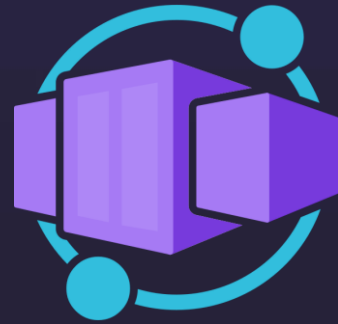
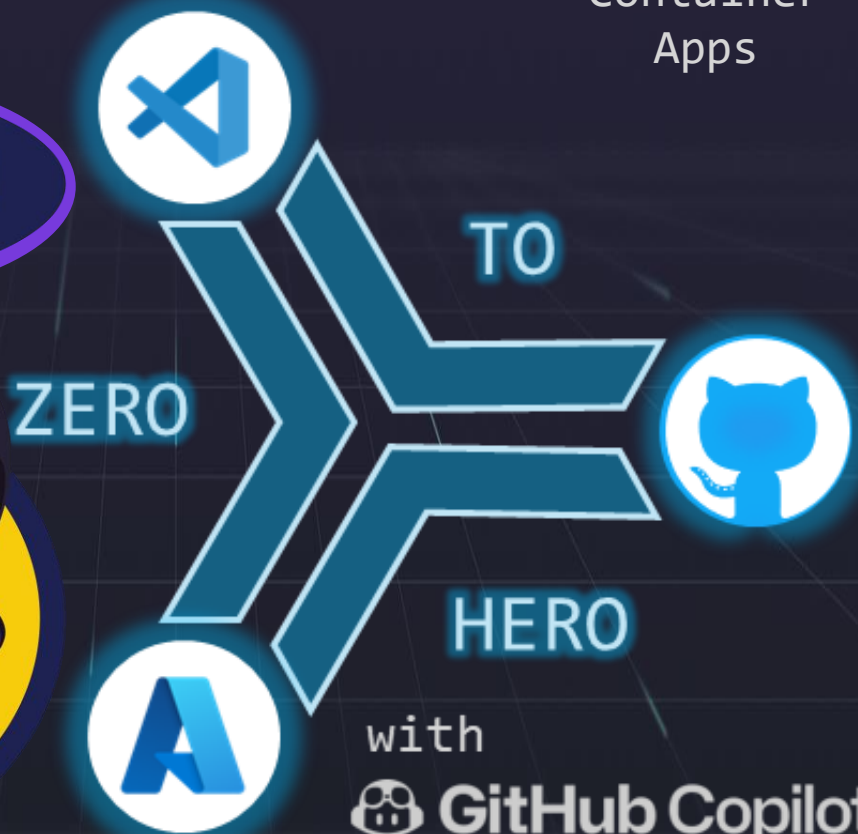


.NET Code to Cloud



Azure
Container
Apps



with
 GitHub Copilot

aka.ms/zero2hero

Please note that the views expressed
in this doc are just my own



github.com/vieiraae



linkedin.com/in/vieira



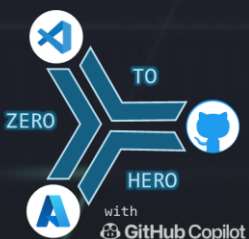
HERO Goals

1. Create a new .NET App from scratch
2. Code and test
3. Add documentation
4. Deploy to Azure Container Apps



.NET Code
to

Cloud





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

Create .NET Blazor App

❑ Type in the chat:



steps to create a .NET blazor server app called HeroBlazorApp in the terminal and include pre-requirements



GitHub Copilot

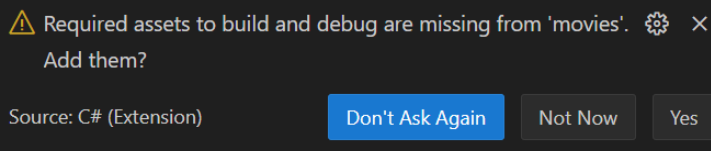
To create a .NET Blazor Server app called HeroBlazorApp, you can follow these steps:

1. Open the terminal in Visual Studio Code by pressing `Ctrl + ~` or by clicking on **Terminal** in the top menu and selecting **New Terminal**.
2. Make sure you have the .NET SDK installed on your machine. You can check this by running the command `dotnet --version` in the terminal. If you don't have it installed, you can download it from the official .NET website.
3. Navigate to the directory where you want to create your project using the `cd` command. For example, `cd Documents/Projects`.
4. Run the following command to create a new Blazor Server app: `dotnet new blazorserver -n HeroBlazorApp`.
5. This will create a new directory called **HeroBlazorApp** with the basic structure of a Blazor Server app.

❑ Install .NET SDK

❑ Run the "dotnet new" command and then go to "File -> Open Folder"

❑ Install the suggested extensions



2 Add the User entity

- ❑ Click View->Explorer and create a new file called User.cs under Data
- ❑ Open the User.cs file
- ❑ Type in the chat:



create a User class in the HeroBlazorApp.Data namespace with id, name, username, email, phone and website properties and json mapping



GitHub Copilot



Here's an example of how you can create a `User` class in the `HeroBlazorApp.Data` namespace with the specified properties and JSON mapping:

```
using System.Text.Json.Serialization;

namespace HeroBlazorApp.Data
{
    public class User
    {
        [JsonPropertyName("id")]
        public int Id { get; set; }

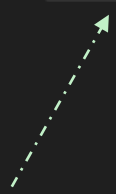
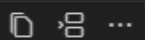
        [JsonPropertyName("name")]
        public string Name { get; set; }

        [JsonPropertyName("username")]
        public string Username { get; set; }

        [JsonPropertyName("email")]
        public string Email { get; set; }

        [JsonPropertyName("phone")]
        public string Phone { get; set; }

        [JsonPropertyName("website")]
        public string Website { get; set; }
    }
}
```



- ❑ Click on "Insert at Cursor"

2 Add the User Service

- ❑ Create a new file called UserService.cs under Data
- ❑ Open the UserService.cs file
- ❑ Type in the chat:



create a user service class that fetches users from the jsonplaceholder web site



GitHub Copilot



Sure! Here's an example of a `UserService` class that fetches users from the JSONPlaceholder website using `HttpClient`:

```
using System.Collections.Generic;
using System.Net.Http;
using System.Text.Json;
using System.Threading.Tasks;

namespace HeroBlazorApp.Data
{
    public class UserService
    {
        private readonly HttpClient _httpClient;

        public UserService(HttpClient httpClient)
        {
            _httpClient = httpClient;
        }

        public async Task<List<User>> GetUsersAsync()
        {
            var response = await _httpClient.GetAsync("https://jsonplaceholder.typicode.com/users");
            response.EnsureSuccessStatusCode();
            var responseStream = await response.Content.ReadAsStreamAsync();
            return await JsonSerializer.DeserializeAsync<List<User>>(responseStream);
        }
    }
}
```

❑ Click on "Insert at Cursor"

This class uses `HttpClient` to make a GET request to the JSONPlaceholder website's `/users` endpoint, deserializes the response into a `List<User>`, and returns it as an asynchronous task.

2 Add the new Service

- ❑ Open the Program.cs file
- ❑ Before the builder.Build instruction add the following comment and accept the code suggestion

```
11 // Add builder for user service singleton
12 builder.Services.AddSingleton<UserService>();
```

- ❑ Add the following comment and accept the code suggestion

```
13 // Add builder for http client
14 builder.Services.AddHttpClient();
```

- ❑ Open the Pages/FetchData.razor file and select all the content
- ❑ Type in the chat:



create a razor page with the router /fetchdata that display user information from the user service



GitHub Copilot



Sure! Here's an example of a Razor page with the router `/fetchdata` that displays user information from the `UserService`:

```
@page "/fetchdata"
@using HeroBlazorApp.Data
@inject UserService UserService
```

```
<PageTitle>Users</PageTitle>
```

```
<h1>Users</h1>
```

```
<p>This component demonstrates fetching data from a service.</p>
```

```
@if (users == null)
{
    <p><em>Loading...</em></p>
}
```

```
else
```

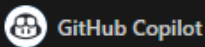
```
{
    <table class="table">
```

- ❑ Click on "Insert at Cursor"

2 Test it locally

❑ Type in the chat:

 *run my app*



To run your Blazor app, you can use the built-in run command in Visual Studio Code. Here are the steps to run your app:

1. Open the terminal in Visual Studio Code by clicking on "Terminal" in the top menu and selecting "New Terminal".
2. In the terminal, navigate to the root directory of your Blazor app.
3. Run the following command to start your app: `dotnet run`
4. Once the app has started, open a web browser and navigate to `https://localhost:5001` to view your app.

Alternatively, you can use the built-in debugging features of Visual Studio Code to run your app. Here are the steps to run your app in debug mode:

1. Open the Debug view in Visual Studio Code by clicking on the "Debug" icon in the Activity Bar on the side of the VS Code window.
2. Click on the "Create a launch.json file" button and select ".NET Core" as the environment.
3. In the generated `launch.json` file, set the `program` property to the path of your app's executable file (usually `bin/Debug/net5.0/{appname}.dll`).
4. Set any additional configuration options as needed.
5. Click on the "Start Debugging" button to run your app in debug mode.

❑ Save all the files

❑ Run "dotnet run" on the terminal

❑ Open the browser and test the App

← ↺ ⓘ localhost:5116/fetchdata

HeroBlazorApp

🏠 Home

⛶ Counter

☰ Fetch data

Users

This component demonstrates fetching data from a service.

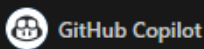
Id	Name	Username	Email
1	Leanne Graham	Bret	Sincere@april.biz
2	Ervin Howell	Antonette	Shanna@melissa.tv

3 Add documentation

- ❑ Click View->Explorer and create a new file called README.md
- ❑ Open the README.md file
- ❑ Type in the chat:



create a readme file in markdown format for this project



Sure! Here's an example of a README.md file for your Blazor app:

HeroBlazorApp

This is a sample Blazor app that demonstrates how to fetch and display data from an external API.

Getting Started

To get started with this app, follow these steps:

1. Clone this repository to your local machine.
2. Open the `HeroBlazorApp` folder in Visual Studio Code.
3. Open a terminal in Visual Studio Code and navigate to the `HeroBlazorApp` folder.
4. Run the following command to start the app: `dotnet run`
5. Once the app has started, open a web browser and navigate to `https://localhost:5001` to view the app.

Features

This app includes the following features:

- Fetching and displaying data from an external API
- Integration with the JSONPlaceholder API
- Use of the `HttpClient` class to make HTTP requests
- Use of the `JsonSerializer` class to deserialize JSON data
- Use of the `User` class to represent user data

Contributing

If you would like to contribute to this app, please follow these steps:

1. Fork this repository to your own GitHub account.
2. Create a new branch for your changes.
3. Make your changes and commit them to your branch.

❑ Click on "Insert at Cursor"

4 Deploy to Azure

❑ Type in the chat:



use the `az containerapp up` command to deploy my app named `heroblazorapp` with external ingress on an environment named `heroenv`, a resource group named `zero2hero-blazor` located in west europe with the source parameter as `"."`



GitHub Copilot

Here's an example command that you can use to deploy your app using the `az containerapp up` command:

```
az containerapp up --name heroblazorapp --resource-group zero2hero-blazor --location westeurope --
```

Insert Into New File

Run in Terminal

This command will deploy your app named `heroblazorapp` with external ingress on an environment named `heroenv`, in a resource group named `zero2hero-blazor` located in `west europe`. The `source` parameter is set to `"."` which means that the current directory will be used as the source for the deployment.

❑ Click on "Run in Terminal"



The "`az containerapp up`" automates the following tasks:

- Provision your Azure resources, including: an Azure Container Apps environment, a container app, an Azure Container Registry, and an Azure Log Analytics workspace
- Build your container image and push it to the container registry (without the need to have Docker installed)
- Deploy your container to the container app

❑ Run the `az` command and wait until it finishes

Your container app `heroblazorapp` has been created and deployed! Congrats!

Your app is running image `ca71903624b5acr.azurecr.io/heroblazorapp:20230801013234037779` and listening on port `8080`
Browse to your container app at: <http://heroblazorapp.jollypond-74e694c8.westeurope.azurecontainerapps.io>

Stream logs for your container with: `az containerapp logs show -n heroblazorapp -g zero2hero-blazor`

See full output using: `az containerapp show -n heroblazorapp -g zero2hero-blazor`

PS C:\Projects\HeroBlazorApp> |

❑ Open the URL and test your App

4 Monitor deployed resources

Open the Resource Group in the Azure Portal to check the resources created

Resources

Recommendations

Filter for any field...

Type equals all X





Location equals all X

+ Add filter

Showing 1 to 4 of 4 records.

☐ Show hidden types ⓘ


No grouping


<input type="checkbox"/> Name ↑↓	Type ↑↓
<input type="checkbox"/>  ca71903624b5acr	Container registry
<input type="checkbox"/>  heroblazorapp	Container App
<input type="checkbox"/>  heroenv	Container Apps Environment
<input type="checkbox"/>  workspace-zero2heroblazorK73D	Log Analytics workspace


Open the Log Analytics Workspace and browse to the ContainerAppConsoleLogs_CL table


▶ Run

Time range : Last 24 hours

 Save ▾

 Share ▾

 New alert rule

 Export ▾

1 ContainerAppConsoleLogs_CL

Results		Chart		
TimeGenerated [UTC] ↑↓	...	_timestamp_d	RevisionName_s	Stream_s
> 8/1/2023, 12:41:43.404 AM		1690850243.38979	heroblazorapp--b38gxdn	stdout
> 8/1/2023, 12:41:43.404 AM		1690850243.38983	heroblazorapp--b38gxdn	stdout

Reviewing



GitHub Copilot

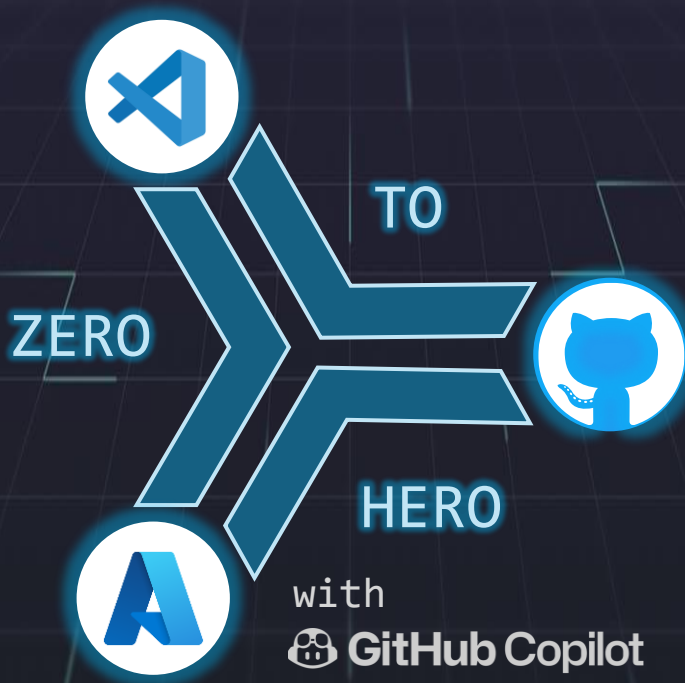
- 👍 Identified requirements and generated the command to create the App
- 👍 Generated entire new C# classes to extend the App and assisted to improve the code
- 👍 Generated documentation in markdown format based on the project context
- 👍 Generated the all-in-one command to deploy the App in Azure Container Apps



aka.ms/zero2hero



Thumbs up
to pump
for more



github.com/vieiraae



linkedin.com/in/vieira