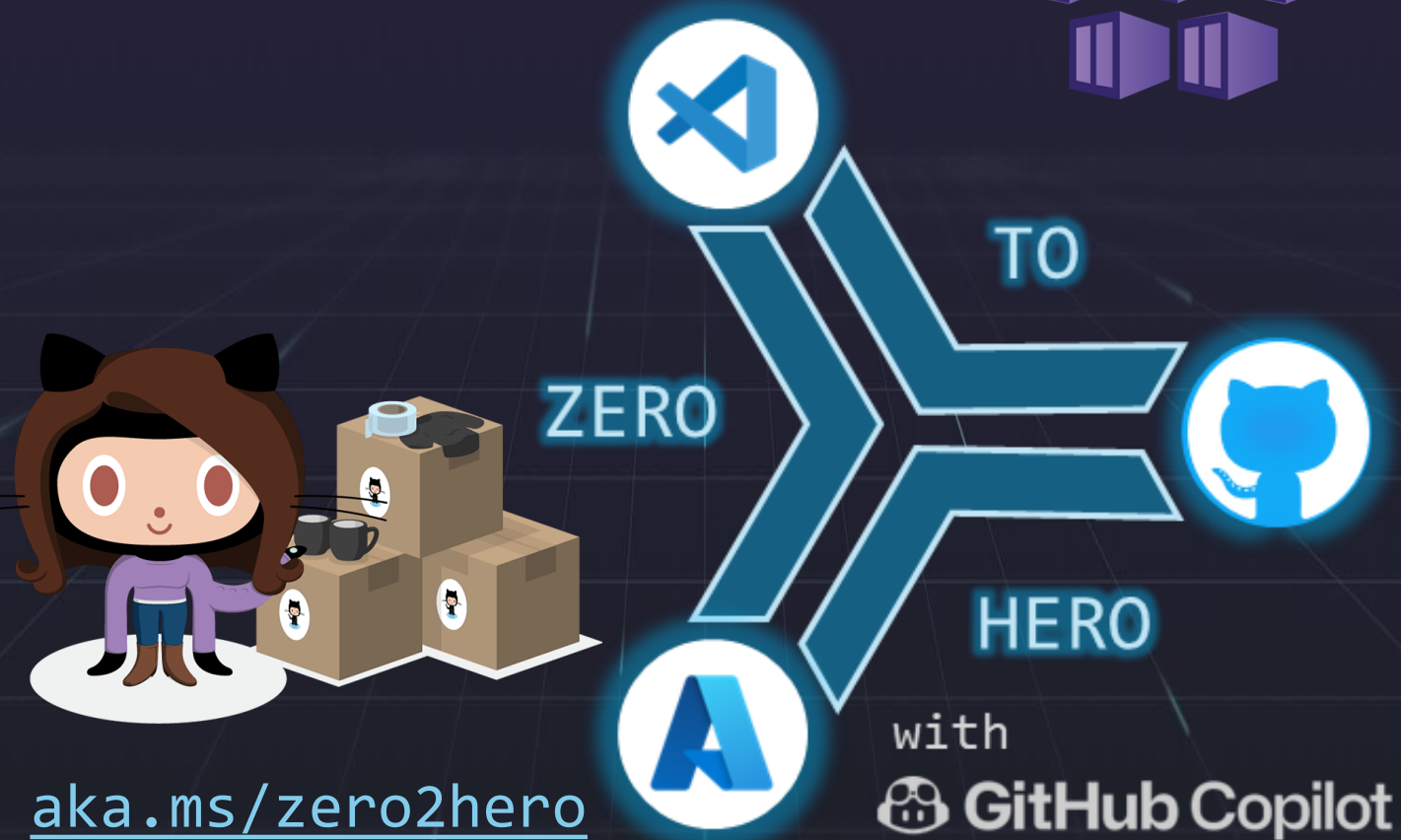


Docker & Kubernetes



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



linkedin.com/in/vieira

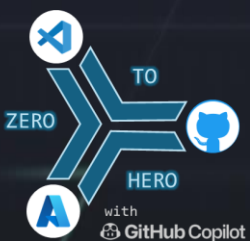


HERO Goals

1. Create a custom NGINX container
2. Create a container registry and Kubernetes cluster
3. Build the container
4. Create k8s manifest
5. Deploy the container



Docker &
Kubernetes





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 the Container

□ Type in the chat:



provide the steps and code for a nginx docker container with a custom configuration file pointing to a welcome HTML page that displays "welcome to my HERO web server" in a HeroContainer folder



GitHub Copilot



Here are the steps to create an Nginx Docker container with a custom configuration file pointing to a welcome HTML page that displays "welcome to my HERO web server" in a HeroContainer folder:

1. Create a new directory for your project and navigate into it:

```
mkdir HeroContainer
cd HeroContainer
```

2. Create a new file called **Dockerfile** and add the following code:

```
FROM nginx
COPY nginx.conf /etc/nginx/conf.d/default.conf
COPY index.html /usr/share/nginx/html/HeroContainer/index.html
```

3. Create a new file called **nginx.conf** and add the following code:

```
server {
    listen 80;
    server_name localhost;
    root /usr/share/nginx/html/HeroContainer;
    index index.html;
}
```

4. Create a new file called **index.html** and add the following code:

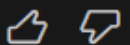
```
<!DOCTYPE html>
<html>
<head>
    <title>Welcome to my HERO web server</title>
```

- ❑ Create the HeroContainer folder and then go to "File -> Open Folder"
- ❑ Create the Dockerfile, nginx.conf and index.html files with the generated content and save all the files
- ❑ (Optional) Run the generated docker commands if you have docker installed. You don't need to do this step since we will build the container with the Azure Container Registry
- ❑ Open the Dockerfile file and select the text
- ❑ Type in the chat:

 /explain



GitHub Copilot



The code above is a Dockerfile that defines a Docker image based on the official Nginx image. The **FROM** directive specifies the base image that this image should be built on, which in this case is the Nginx image.

The **COPY** directives are used to copy files from the host machine into the Docker image. The first **COPY** directive copies the **nginx.conf** file from the host machine into the **/etc/nginx/conf.d/default.conf** file in the Docker image. This file contains the configuration for the Nginx web server, including the server block that defines how the server should respond to requests.

The second **COPY** directive copies the **index.html** file from the host machine into the **/usr/share/nginx/html/HeroContainer/index.html** file in the Docker image. This file contains the HTML content that will be served by the Nginx web server when a request is made to the server.

Overall, this Dockerfile sets up a Docker image that includes the Nginx web server and the configuration and content files needed to serve a basic web page. This image can be used to run a container that serves this web page, and can be customized further to include additional content or configuration as needed.

2

Create Azure Resources

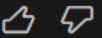
❑ Type in the chat:



use the terminal to create an Azure Resource Group named hero-container-rg located on west europe with a new Azure Kubernetes Service named heroaks attached to a new Azure Container Registry named heroacr1



GitHub Copilot



Sure! Here's the command to create an Azure Resource Group named **hero-container-rg** located in West Europe:

```
az group create --name hero-container-rg --location westeurope
```

And here's the command to create an Azure Container Registry named **heroacr1**:

```
az acr create --resource-group hero-container-rg --name heroacr1 --sku Basic
```

Finally, here's the command to create an Azure Kubernetes Service named **heroaks** attached to the **heroacr1** container registry:

```
az aks create --resource-group hero-container-rg --name heroaks --node-count 1
```

This will create a new AKS cluster with one node and attach it to the **heroacr1** container registry. You can now deploy your containerized application to the AKS cluster using the **kubect1** command-line tool.

❑ Run the commands in the terminal (bash or PowerShell)



the registry name must be unique



the AKS will be created with ACR automatically attached

3

Build the Container

❑ Type in the chat:



use the terminal to build my container in the cloud without using docker and using my Azure Container Registry instead with the name hero-container and tag v1



GitHub Copilot

To build your container in the cloud without using Docker and using your Azure Container Registry instead with the name **hero-container** and tag **v1**, you can use the **az acr build** command. Here's an example command:

```
az acr build --registry heroacr1 --image hero-container:v1 .
```

This command will build a new container image from the current directory (.) and push it to your Azure Container Registry named **heroacr1** with the name **hero-container** and tag **v1**.

- ❑ Run the commands in the terminal (bash or PowerShell)
- ❑ (Optional) browse the Container Registry and locate the container image that was just built



heroacr1 | Repositories

Container registry



Refresh



Manage Deleted Repositories



Overview



Activity log



Access control (IAM)



Tags



Quick start



Events



Repositories ↑↓

hero-container

4 Create k8s manifest

❏ Type in the chat:



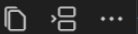
create a Kubernetes manifest file with a service and a deployment named hero-container-app with my container image and using the acrLoginServer as heroacr1.azurecr.io



GitHub Copilot



Here's an example Kubernetes manifest file that creates a deployment and a service named **hero-container-app** using your container image from your Azure Container Registry **heroacr1.azurecr.io**:



```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hero-container-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: hero-container-app
  template:
    metadata:
      labels:
        app: hero-container-app
    spec:
      containers:
        - name: hero-container
          image: heroacr1.azurecr.io/hero-container:v1
          ports:
            - containerPort: 80
          imagePullSecrets:
            - name: acr-auth
---
apiVersion: v1
kind: Service
metadata:
  name: hero-container-app
spec:
  selector:
    app: hero-container-app
  ports:
    - name: http
      port: 80
      targetPort: 80
```

- ❑ Copy the content to a new file named `deploy.yaml`
- ❑ Select all the text
- ❑ Type in the chat:

... */fix add resource limits*



GitHub Copilot

To add resource limits to the container in the deployment spec, you can add the **resources** field to the container spec. Here's an example:

```
apiVersion: apps/v1
```

...

```
resources:
  limits:
    cpu: 500m
    memory: 512Mi
  requests:
    cpu: 250m
    memory: 256Mi
```

...

- ❑ (Optional) replace the `deploy.yaml` content with the generated fix
- ❑ (Optional) go to the end of the file and press `Ctrl + [Enter]` to see a list of suggestions to add to the file

! GitHub Copilot 9+ X

```
1 Synthesizing 10/10 solutions
```

```
2
```

```
Accept Solution
```

```
3 =====
```

```
4 Suggestion 1
```

```
5
```

```
6 apiVersion: networking.k8s.io/v1
```

```
7 kind: Ingress
```

```
8 metadata:
```


```
9   name: hero-container-app
```

```
10  annotations:
```

```
11    kubernetes.io/ingress.class: nginx
```

- ❑ Select again all the text from the `deploy.yaml` file
- ❑ Type in the chat:

 */explain*

 GitHub Copilot

The code above is a Kubernetes manifest file that defines a deployment and a service for a containerized application. The `apiVersion` field specifies the version of the Kubernetes API that this manifest is using, which in this case is `apps/v1` for the deployment and `v1` for the service. The `kind` field specifies the type of Kubernetes object that this manifest is defining, which in this case is a `Deployment` for the deployment and a `Service` for the service.

The `metadata` section of the deployment contains metadata about the deployment, including the name of the deployment, which in this case is `hero-container-app`. The `spec` section contains the specification for the deployment, including the number of replicas that should be created, which in this case is 1. The `selector` section specifies the labels that the deployment should use to select the pods that are part of the deployment. In this case, the selector is `app: hero-container-app`, which means that the deployment will select pods that have the label `app` set to `hero-container-app`.

The `template` section specifies the template for the pods that should be created as part of the deployment. The `metadata` section of the template contains labels that will be applied to the pods, including the `app` label set to `hero-container-app`. The `spec` section of the template contains the specification for the containers that should be created in the pods. In this case, there is one container named `hero-container` that uses the `heroacr1.azurecr.io/hero-container:v1` image and listens on port 80.

The `imagePullSecrets` section specifies the name of the secret that should be used to pull the container image from the Azure Container Registry. In this case, the secret is named `acr-auth`.

The `metadata` section of the service contains metadata about the service, including the name of the service, which in this case is `hero-container-app`. The `spec` section contains the specification for the service, including the type of service, which in this case is a `LoadBalancer`. This means that the service will be exposed externally with a load balancer that distributes incoming traffic to the pods that are part of the service.

The `ports` section specifies the ports that the service should listen on, which in this case is port 80. The `selector` section specifies the labels that the service should use to select the pods that are part of the service. In this case, the selector is `app: hero-container-app`, which means that the service will select pods that have the label `app` set to `hero-container-app`.

Overall, this manifest file sets up a deployment and a service for a containerized application that listens on port 80 and distributes incoming traffic to pods that have the label `app` set to `hero-container-app`. This deployment and service can be used to run and expose the application externally.

5 Deploy the Container

❑ Type in the chat:



use the terminal to get the aks credentials, run the deploy.yaml manifest file and output the service endpoint



GitHub Copilot

Here are the commands to get the AKS credentials, deploy the `deploy.yaml` manifest file, and output the service address:

```
az aks get-credentials --resource-group hero-container-rg --name heroaks

kubectl apply -f deploy.yaml

kubectl get service hero-container-app -o jsonpath='{.status.loadBalancer.ingress[0].ip}'
```

The first command gets the AKS credentials for your Kubernetes cluster named `heroaks` in the resource group `hero-container-rg`.

The second command deploys the `deploy.yaml` manifest file to your Kubernetes cluster.

The third command gets the external IP address of the `hero-container-app` service and outputs it to the console. You can use this IP address to access your application.

- ❑ Run the commands in the terminal (bash or PowerShell)
- ❑ Open the command palette and type: Simple Browser: Show and then enter `http://` + the IP returned in the last command

← → ↻ `http://20.31.211.8`

Welcome to my HERO web server

- ❑ (Optional) Delete the Resource Group to stop billing

Reviewing



GitHub Copilot

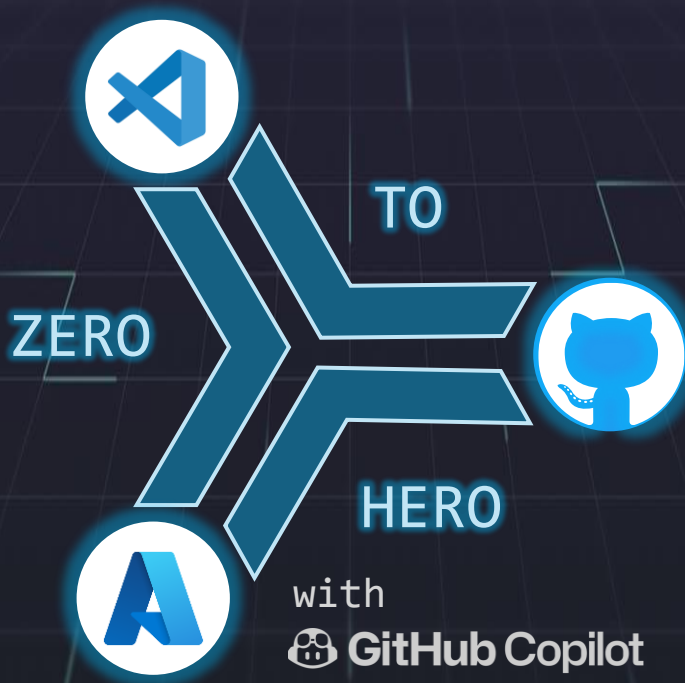
- 👍 Generated the Dockerfile, nginx configuration and custom HTML
- 👍 Generated commands to create a Resource Group, Container Registry and AKS
- 👍 Generated command to build the container in the cloud using the Container Registry
- 👍 Generated the Kubernetes manifest for the deployment and service
- 👍 Generated the commands to get AKS credentials, apply the manifest and output the service endpoint to test the end result



aka.ms/zero2hero



Thumbs up
to pump
for more



github.com/vieiraae



linkedin.com/in/vieira