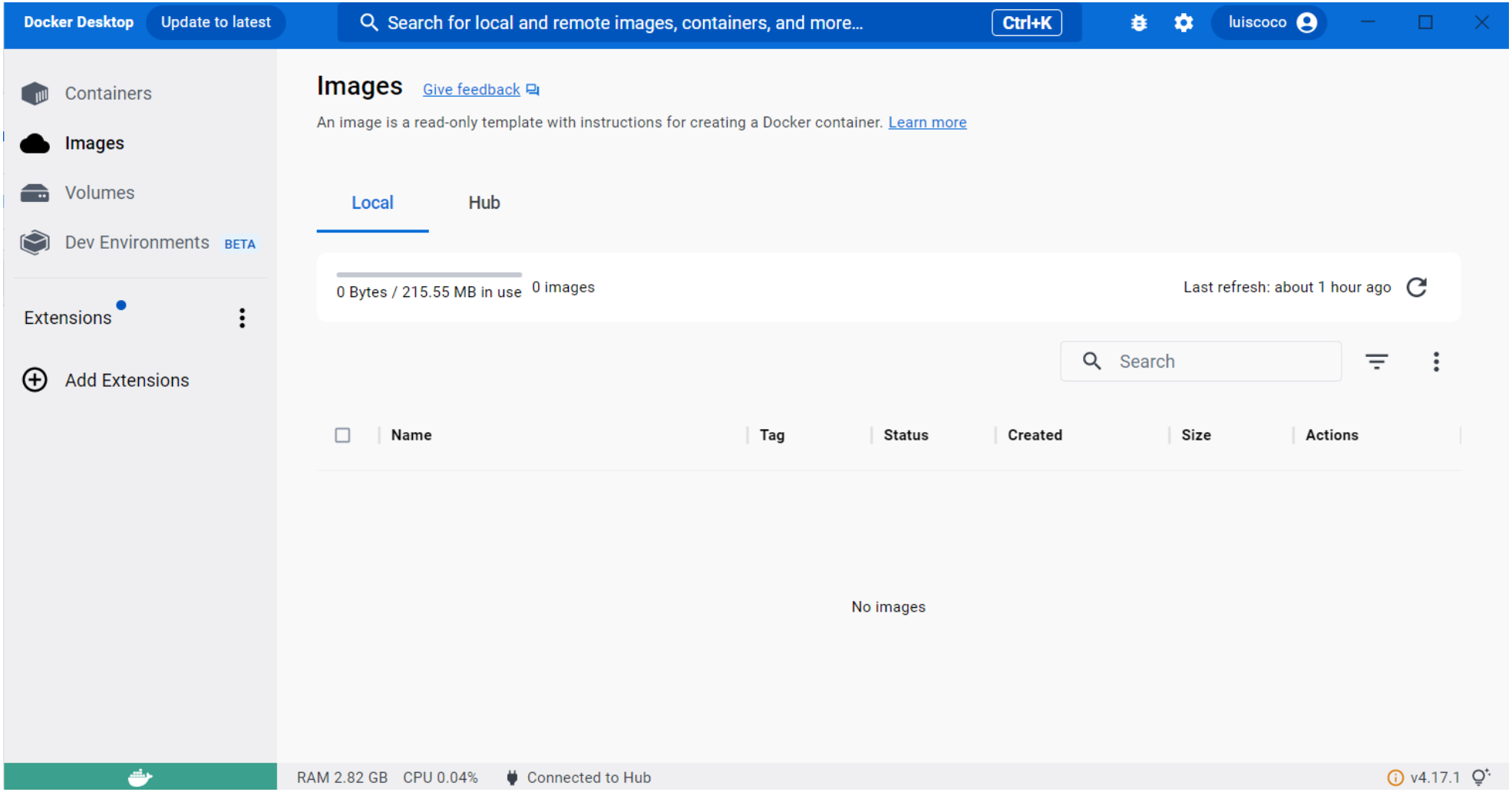
How to deploy to Kubernetes a .NET 8 Web API

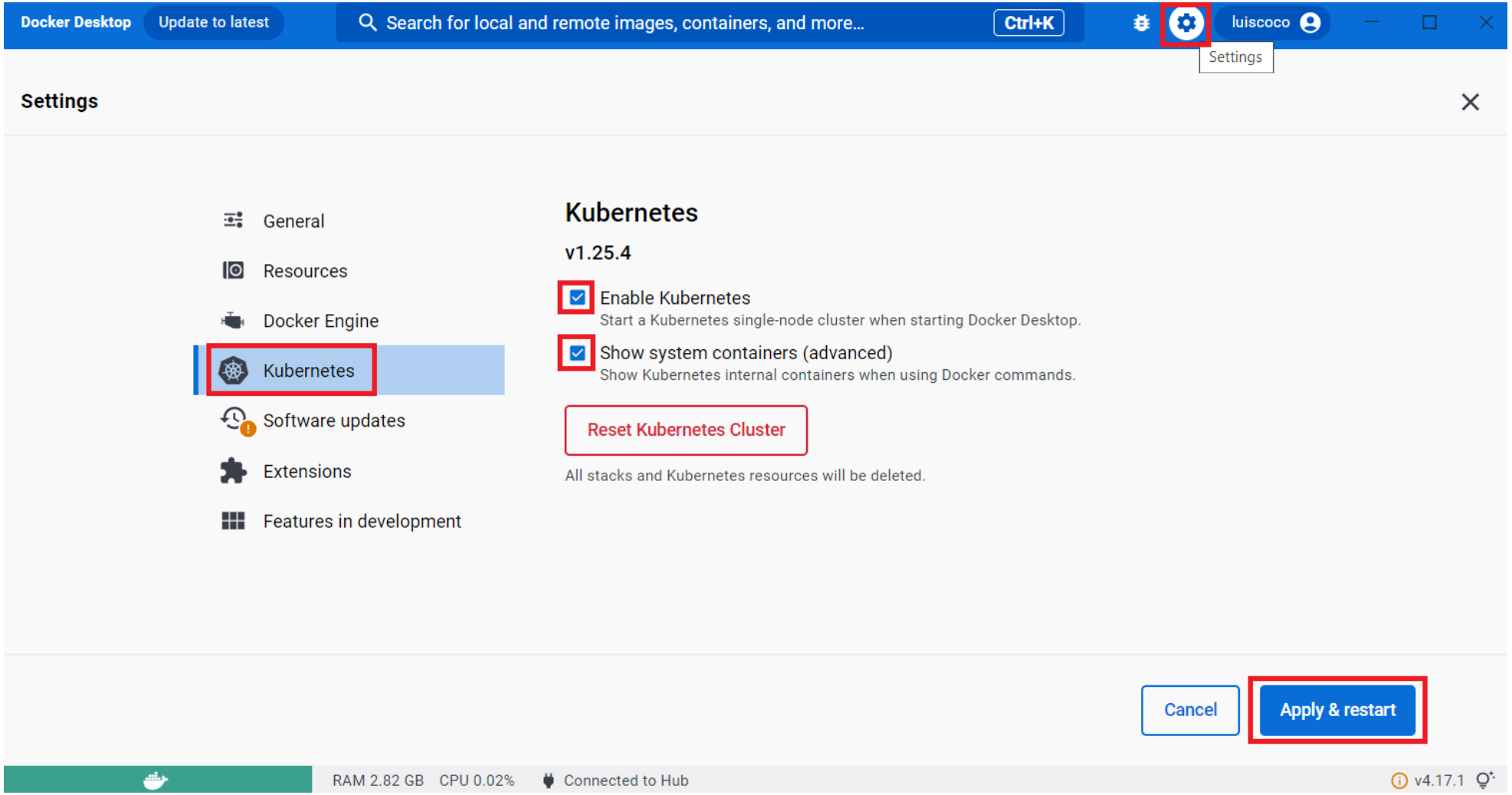
1. Prerequisites

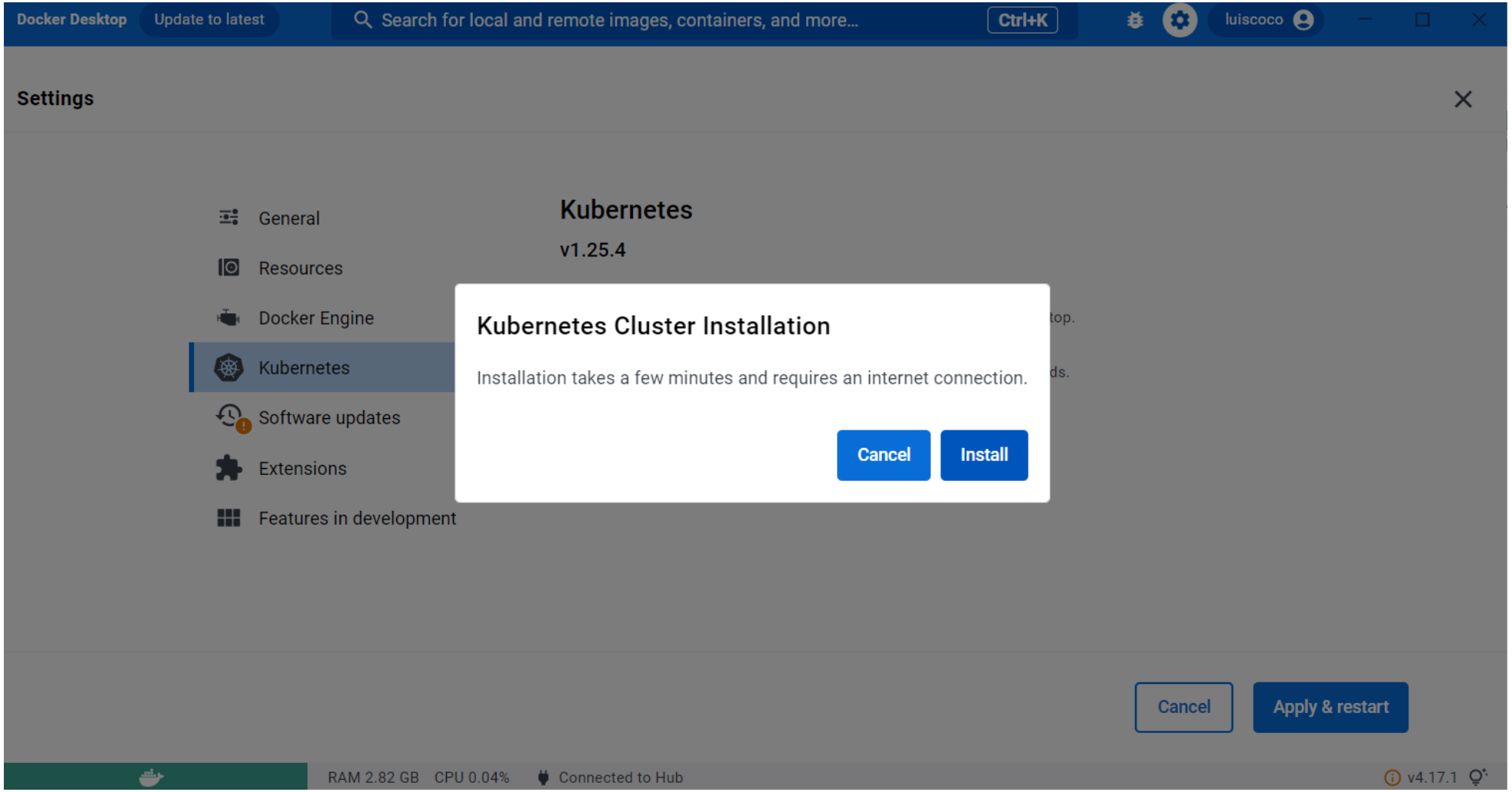
Download and Install Docker Desktop: https://docs.docker.com/desktop/install/windows-install/

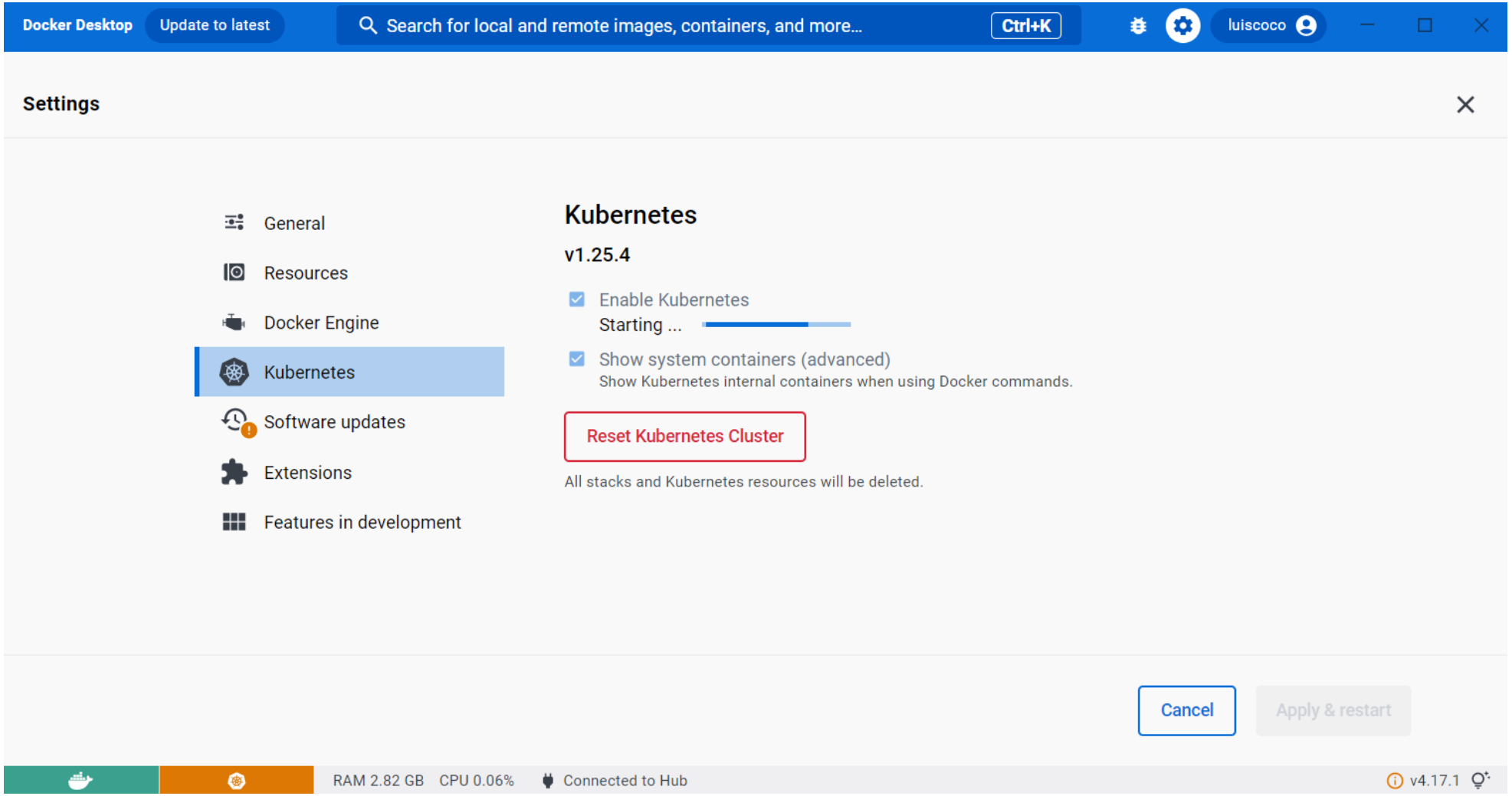
Run Docker Desktop

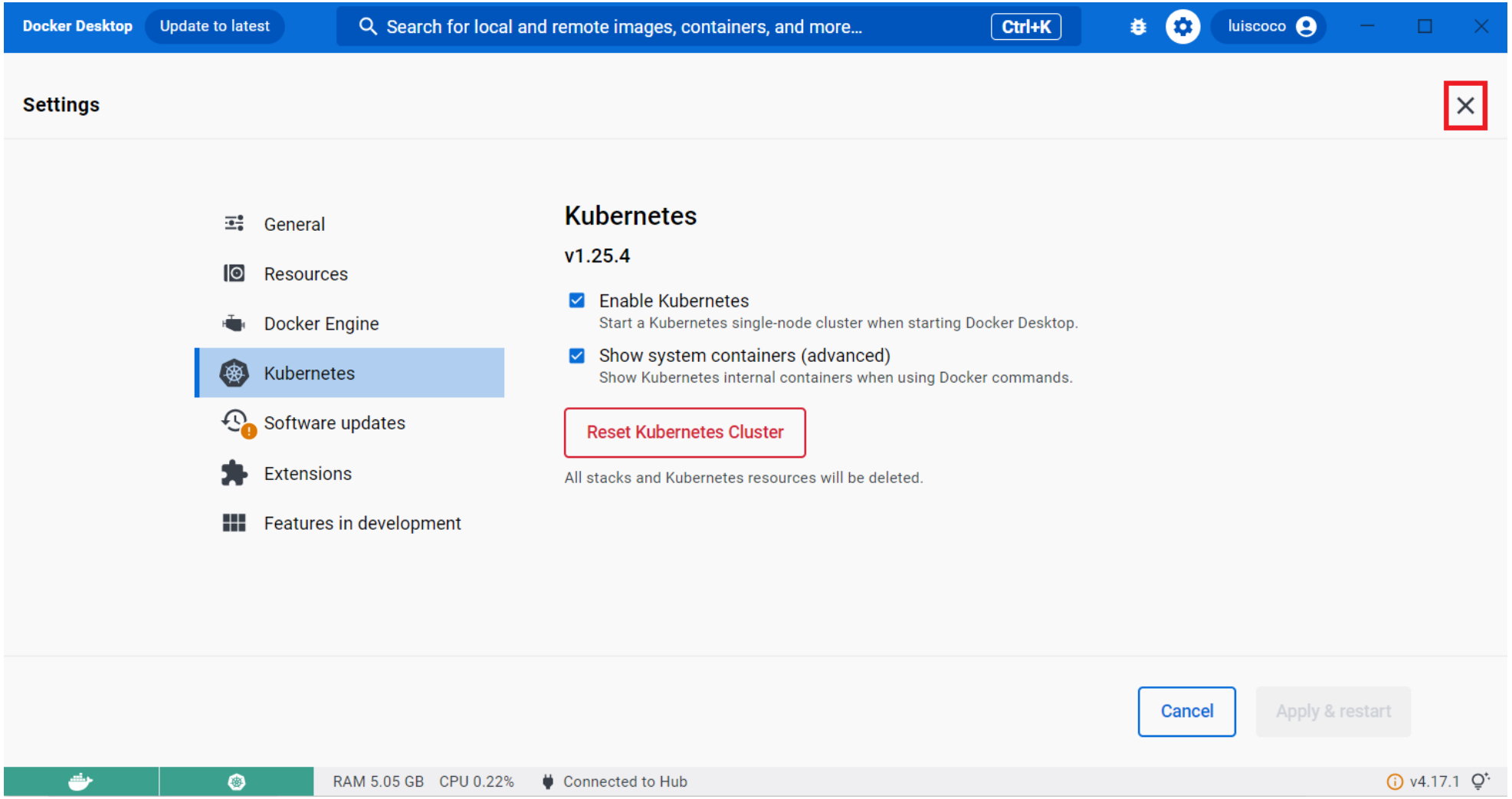


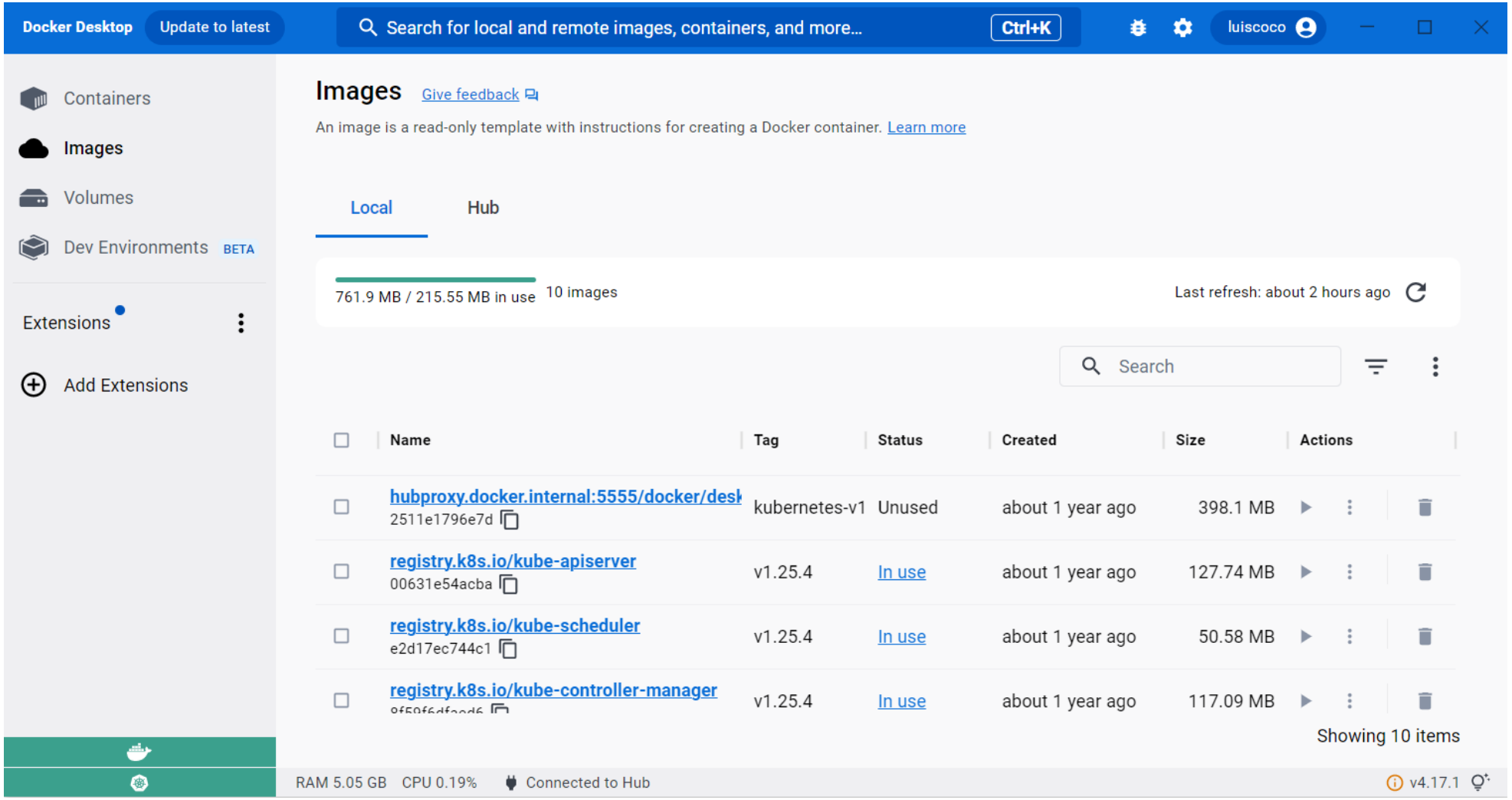
Navigate to enable the Kubernetes option in Docker Desktop:


Docker Desktop
Update to latest
Search for local and remote images, containers, and more...
Ctrl+K
luiscoco
Settings
Settings
General
Resources
Docker Engine
Kubernetes
Software updates
Extensions
Features in development
Kubernetes
v1.25.4
Enable Kubernetes
Start a Kubernetes single-node cluster when starting Docker Desktop.
Show system containers (advanced)
Show Kubernetes internal containers when using Docker commands.
Reset Kubernetes Cluster
All stacks and Kubernetes resources will be deleted.
Cancel
Apply & restart
RAM 2.82 GB
CPU 0.02%
Connected to Hub
v4.17.1


Docker Desktop
Update to latest
Search for local and remote images, containers, and more...
Ctrl+K
luiscoco
Settings
General
Resources
Docker Engine
Kubernetes
Software updates
Extensions
Features in development
Kubernetes
v1.25.4
Kubernetes Cluster Installation
Installation takes a few minutes and requires an internet connection.
Cancel
Install
Cancel
Apply & restart
RAM 2.82 GB
CPU 0.04%
Connected to Hub
v4.17.1


Docker Desktop
Update to latest
Search for local and remote images, containers, and more...
Ctrl+K
luiscoco
Settings
General
Resources
Docker Engine
Kubernetes
Software updates
Extensions
Features in development
Kubernetes
v1.25.4
Enable Kubernetes
Starting ...
Show system containers (advanced)
Show Kubernetes internal containers when using Docker commands.
Reset Kubernetes Cluster
All stacks and Kubernetes resources will be deleted.
Cancel
Apply & restart
RAM 2.82 GB
CPU 0.06%
Connected to Hub
v4.17.1


Docker Desktop
Update to latest
Search for local and remote images, containers, and more...
Ctrl+K
luiscoco
Settings
General
Resources
Docker Engine
Kubernetes
Software updates
Extensions
Features in development
Kubernetes
v1.25.4
Enable Kubernetes
Start a Kubernetes single-node cluster when starting Docker Desktop.
Show system containers (advanced)
Show Kubernetes internal containers when using Docker commands.
Reset Kubernetes Cluster
All stacks and Kubernetes resources will be deleted.
Cancel
Apply & restart
RAM 5.05 GB
CPU 0.22%
Connected to Hub
v4.17.1



2. Summarizing the steps for deploying your application to Kubernetes

Here are the general steps to deploy your .NET 8 Web API to Kubernetes:

1. Build and Push the Docker image to the Docker Hub registry/repo

2. Create Kubernetes Deployment YAML file. This file defines how your application is deployed in Kubernetes.

3. Create Kubernetes Service YAML file. This file defines how your application is exposed, either within Kubernetes cluster or to the outside world.

4. Apply the YAML files to your Kubernetes Cluster: use the command "**kubectl apply**" to create the resource defined in your YAML file in your Kubernetes cluster.

3. Build and Push the Docker image to the Docker Hub registry/repo

For more details about this section see the repo: https://github.com/luiscoco/Docker_Create_and_run_Image-_for_dotNET_8_Web_API

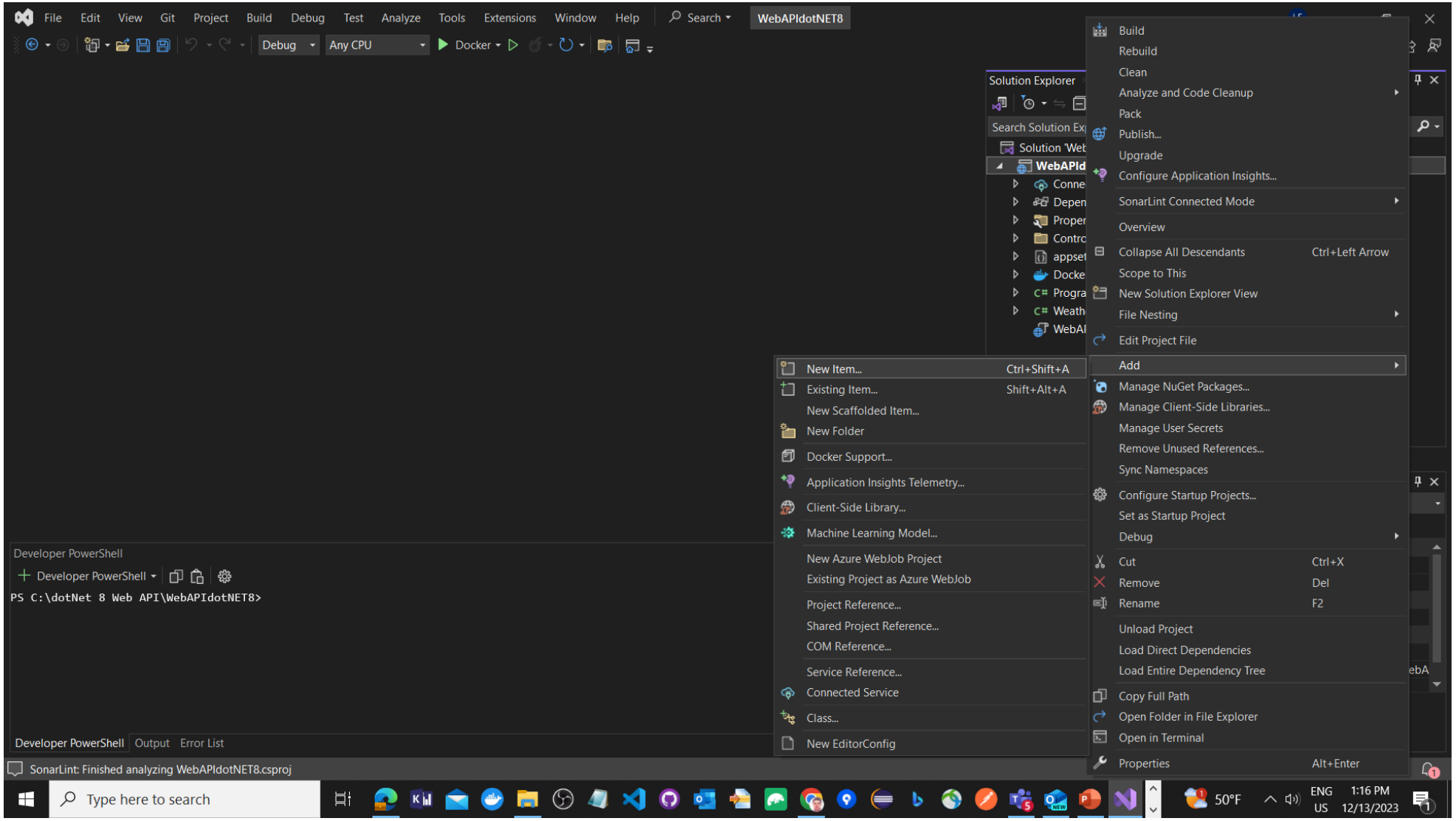
```
docker build -t luiscoco/webapidotnet8:latest .
```

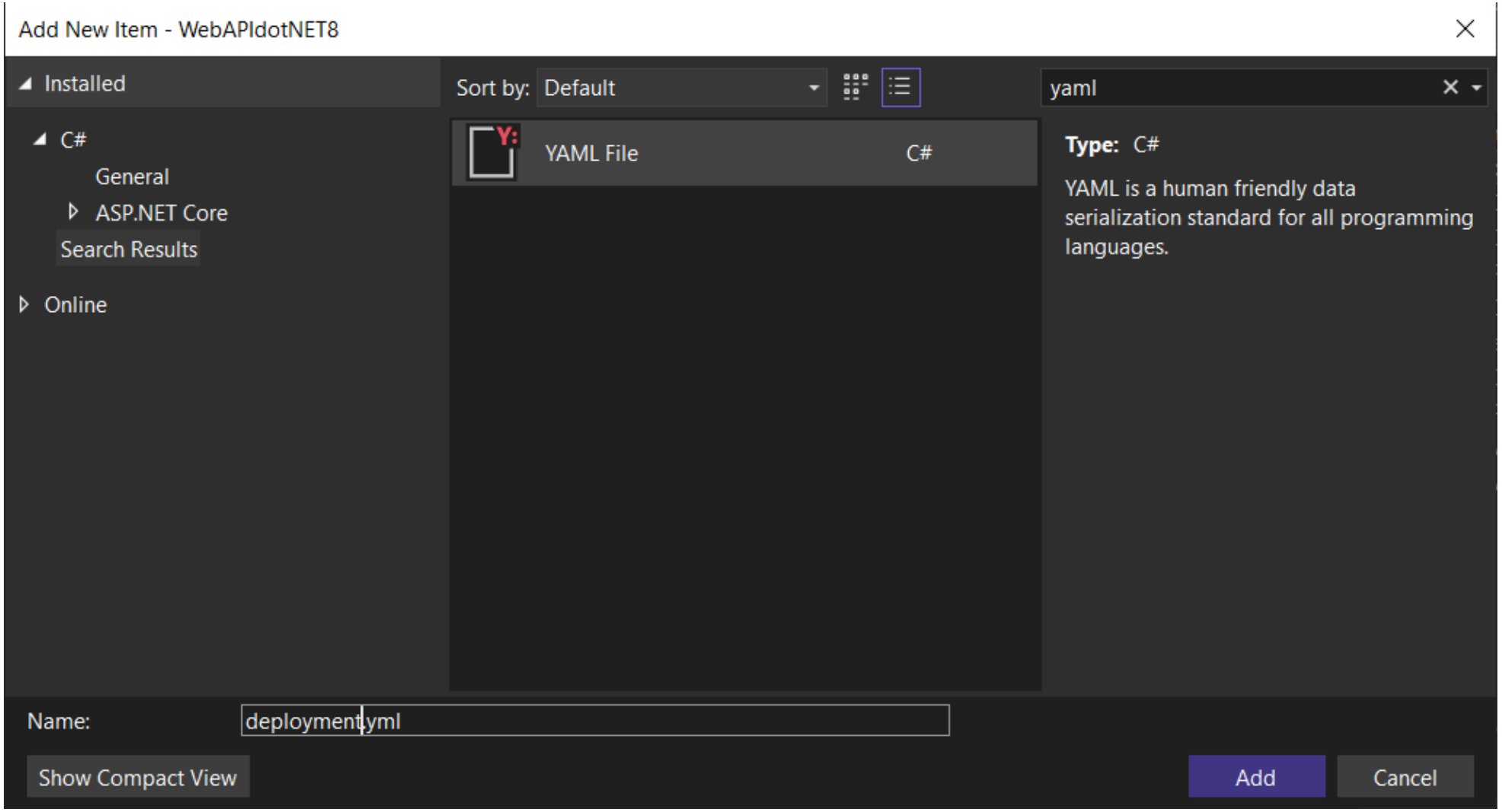
Then we use the docker push command to upload the image to the Docker Hub repository:

```
docker push luiscoco/webapidotnet8:latest
```

4. Create Kubernetes Deployment YAML file

In Visual Studio we add a new yaml file to our project


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Search
WebAPIdotNET8
Debug
Any CPU
Docker
Solution Explorer
Search Solution Ex
Solution 'Web
WebAPId
Conne
Depen
Proper
Contrc
appset
Docke
Progra
Weath
WebAI
New Item... Ctrl+Shift+A
Existing Item... Shift+Alt+A
New Scaffolded Item...
New Folder
Docker Support...
Application Insights Telemetry...
Client-Side Library...
Machine Learning Model...
New Azure WebJob Project
Existing Project as Azure WebJob
Project Reference...
Shared Project Reference...
COM Reference...
Service Reference...
Connected Service
Class...
New EditorConfig
Build
Rebuild
Clean
Analyze and Code Cleanup
Pack
Publish...
Upgrade
Configure Application Insights...
SonarLint Connected Mode
Overview
Collapse All Descendants Ctrl+Left Arrow
Scope to This
New Solution Explorer View
File Nesting
Edit Project File
Add
Manage NuGet Packages...
Manage Client-Side Libraries...
Manage User Secrets
Remove Unused References...
Sync Namespaces
Configure Startup Projects...
Set as Startup Project
Debug
Cut Ctrl+X
Remove Del
Rename F2
Unload Project
Load Direct Dependencies
Load Entire Dependency Tree
Copy Full Path
Open Folder in File Explorer
Open in Terminal
Properties Alt+Enter
Developer PowerShell
Developer PowerShell
PS C:\dotNet 8 Web API\WebAPIdotNET8>
Developer PowerShell Output Error List
SonarLint: Finished analyzing WebAPIdotNET8.csproj
Type here to search
50°F
ENG US
1:16 PM 12/13/2023



This is the source code for the **deployment.yaml** file:

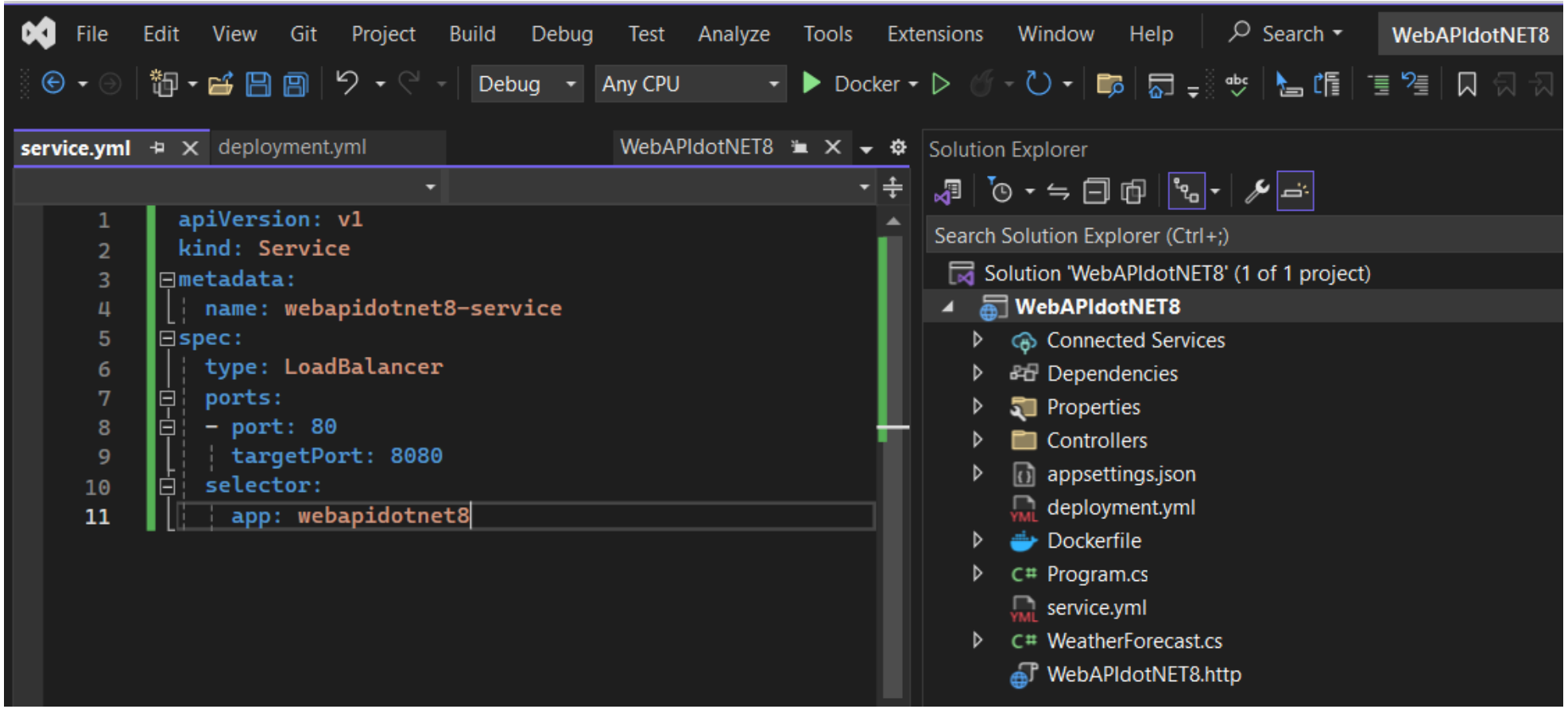
deployment.yml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: webapidotnet8-deployment
```

```
  spec:
    replicas: 2
    selector:
      matchLabels:
        app: webapidotnet8
    template:
      metadata:
        labels:
          app: webapidotnet8
      spec:
        containers:
        - name: webapidotnet8
          image: luiscoco/webapidotnet8:latest  # Replace with your image path
          ports:
          - containerPort: 8080
```

5. Create Kubernetes Service YAML file

In Visual Studio we add a new yaml file to our project. See the steps explained in the section 4.



service.yml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: webapidotnet8-service
spec:
  type: LoadBalancer
  ports:
  - port: 80
    targetPort: 8080
```

```
    selector:
      app: webapidotnet8
```

6. Applying the YAML Files

```
kubectl apply -f deployment.yml
```

```
PS C:\dotNet 8 Web API\WebAPIdotNET8> kubectl apply -f deployment.yml
deployment.apps/webapidotnet8-deployment created
```

```
kubectl apply -f service.yml
```

```
PS C:\dotNet 8 Web API\WebAPIdotNET8> kubectl apply -f service.yml
service/webapidotnet8-service created
```

We can use the command "**kubectl get services**" to check the IP and port your application is accessible on, if using a LoadBalancer.

Verify the Deployment with the command:

```
kubectl get deployments
```

```
PS C:\dotNet 8 Web API\WebAPIdotNET8> kubectl get deployments
NAME                     READY   UP-TO-DATE   AVAILABLE   AGE
webapidotnet8-deployment   2/2     2            2           2m8s
```

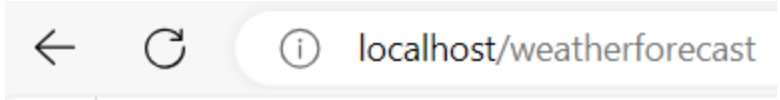
Verify the service status with the command:

```
kubectl get services
```

```
PS C:\dotNet 8 Web API\WebAPIdotNET8> kubectl get services
NAME                    TYPE           CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
kubernetes              ClusterIP      10.96.0.1      <none>        443/TCP        12m
webapidotnet8-service   LoadBalancer   10.98.224.65   localhost     80:31558/TCP   87s
```

7. Navigate to the Web API endpoint

http://localhost/weatherforecast




```
[
    {
        "date": "2023-12-14",
        "temperatureC": 20,
        "temperatureF": 67,
        "summary": "Scorching"
    },
    {
        "date": "2023-12-15",
        "temperatureC": 5,
        "temperatureF": 40,
        "summary": "Balmy"
    },
    {
        "date": "2023-12-16",
        "temperatureC": 11,
        "temperatureF": 51,
        "summary": "Chilly"
    },
    {
        "date": "2023-12-17",
        "temperatureC": -3,
        "temperatureF": 27,
        "summary": "Sweltering"
    },
    {
        "date": "2023-12-18",
        "temperatureC": 48,
        "temperatureF": 118,
        "summary": "Scorching"
    }
]
```