Bonus Track! Cloud, Testing & Containers

Add the following to your solution to make it more robust:

● Host your architecture in any public cloud (using the services you consider more

adequate)

● Add automated tests to the API

○ You can use whichever library that you want

○ Different tests types, if necessary, are welcome

● Containerize your application

○ Create a Dockerfile to deploy the package.

Automated tests to the API:

1. Choose Testing Framework: Azure supports various testing frameworks like NUnit (Assign,Act,Assert), MSTest, xUnit, etc.
2. Write Unit Tests: Start by writing unit tests for individual components of your REST API.
3. Write Integration Tests: Use Azure DevOps Test Plans or Azure Test Automation to automate the execution of these tests.
4. Utilize Azure Pipelines: Set up Azure Pipelines to automate the build and deployment process of your API along with running tests. Configure pipelines to trigger tests whenever changes are made to your codebase.
5. Monitor Test Results: Monitor test results using Azure Monitor or Azure Application Insights to track the health of your API and identify any regressions or performance issues.

These may include:

1. **Azure DevOps**: Azure DevOps offers robust support for continuous integration (CI) and continuous deployment (CD) pipelines. You can configure Azure Pipelines to automatically trigger NUnit tests upon code commits, providing quick feedback on the code quality.
2. **Azure App Service**: If your .NET application, including REST APIs, is deployed to Azure App Service, you can configure deployment slots to deploy your application and run NUnit tests in a staging environment before promoting it to production.
3. **Azure Container Instances (ACI)**: For containerized .NET applications, you can leverage Azure Container Instances to run NUnit tests within isolated containers, providing consistent and reproducible testing environments.

Containerize your application(Create a Dockerfile to deploy the package):

Go to Azure Container Registry to register a docker image (enabling admin user in the access keys)

Go to Azure DevOps and select Pipelines with Source GitHub and put the repo name to push the image.