# Problem Statement:

You work for an e-commerce company, and it generally experiences a heavy load. Due to the high load, the application component must scale to meet the demand on-peak hours. The company has decided to implement a container-based (Docker) web app development using Kubernetes. This is because the web app instance can be scaled very quickly, depending on load. The company has decided to implement an Azure pipeline to deploy the solution to Azure Kubernetes, which can be scaled fast depending on workload.

# Steps to Perform:

* Create a docker based solution and deploy to Azure Kubernetes cluster using Azure DevOps (Use DevOps Starter)
* Create an ASP.NET Web application with a Docker-based solution locally
* Check-in the solution to Azure Repos
* Create a CI/CD pipeline to deploy this application to the Container Registry
* On successful deployment to the Container Registry, let the changes propagate to the Azure Kubernetes Cluster

# Solution

1. Create a Docker-based solution locally

- Install Docker on your local machine

- Create an ASP.NET Web application

- Modify the application to use Docker containers for hosting

2. Check-in the solution to Azure Repos

- Create an Azure DevOps account if you don't have one already

- Create a new project in Azure DevOps

- Create a new repository in the project

- Clone the repository to your local machine

- Add the Docker-based solution to the repository

- Commit and push the changes to Azure Repos

3. Create a CI/CD pipeline using Azure DevOps

- In the Azure DevOps project, go to the Pipelines section

- Create a new pipeline and select "Azure Repos Git" as the source

- Select the repository and branch containing the Docker-based solution

- Choose "Docker" as the template for the pipeline

- Configure the pipeline to build the Docker image, push it to the Azure Container Registry, and deploy it to the Azure Kubernetes Cluster

- Save and run the pipeline to ensure it works correctly

4. Test the deployment

- Check the Azure Container Registry to verify that the Docker image has been pushed successfully

- Check the Azure Kubernetes Cluster to verify that the deployment has been created and is running

- Test the web application to ensure it's working correctly

By following these steps, you can create a Docker-based solution for an ASP.NET Web application and deploy it to an Azure Kubernetes Cluster using Azure DevOps. This approach enables you to quickly scale the web app instance depending on the load and helps ensure high availability for your e-commerce application during peak hours.