# Problem Statement:

You have been hired in an e-commerce company Flipso that wants to deploy the code frequently, depending on customer feedback. As part of the deployment, the database needs to be updated with customer reviews and feedback. The database connection string is a confidential parameter which even the developer should not know. The company has decided to store the database connection string in Key vault, which only the application can access to connect to the database. This way, the developer or any other person cannot see the database credential, and thus database access will remain restricted.

# Steps to Perform:

* Create an Azure Key vault and store some parameters with values in it
* Create a sample ASP.NET application with some parameters taken from Azure key vault
* Check in the code to Azure Repos
* Create a CI/CD pipeline using YAML
* Deploy the application through the CI/CD pipeline to the Azure Web app

Sample Application Source

* Program.cs
* appsettings.json
* ValuesController.cs

# Solution

1. Create an Azure Key Vault and store parameters:

- Log in to the Azure portal (portal.azure.com)

- Create a new Azure Key Vault resource

- Configure access policies to allow the application to access the Key Vault

- Add the required parameters with their values to the Key Vault, such as the database connection string

2. Create a sample ASP.NET application with Key Vault integration:

- Open Visual Studio

- Create a new ASP.NET Web Application project

- Install the Azure Key Vault NuGet package to enable Key Vault integration

- Modify the application code to retrieve the database connection string from the Key Vault

3. Check-in the code to Azure Repos:

- Create a new repository in Azure Repos

- Clone the repository to your local machine

- Copy the project files to the local repository directory

- Use Git commands to commit and push the changes to Azure Repos

4. Create a CI/CD pipeline using YAML:

- In Azure DevOps, go to the Pipelines section

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

- Choose "Existing Azure Pipelines YAML file" and specify the YAML file path in your repository

- Configure the pipeline YAML file to build and package the application

- Configure the pipeline to retrieve the Key Vault secrets and pass them as environment variables during the deployment

5. Deploy the application through the CI/CD pipeline to the Azure Web app:

- Set up the Azure Web App resource to host your application

- Configure the deployment step in the pipeline YAML file to deploy the application to the Azure Web App

- Save and run the pipeline to trigger the deployment process

- Monitor the pipeline logs for any errors and ensure the application is successfully deployed

By following these steps, you should be able to create an Azure Key Vault, store confidential parameters in it, create an ASP.NET application with Key Vault integration, check-in the code to Azure Repos, and create a CI/CD pipeline using YAML to deploy the application to an Azure Web App while securely retrieving the database connection string from the Key Vault.