Adapting Logging Configuration for Cross-Platform Compatibility in .NET Core Application

**Summary:** This issue involves refining the logging setup in a .NET Core application to ensure compatibility with both Windows and non-Windows systems during Docker image builds.

By introducing conditional checks, the logging configuration seamlessly integrates the **AddEventLog** provider on Windows while gracefully accommodating alternative providers on other platforms, enhancing the application's cross-platform resilience.

**Issue Explanation:** The error encountered during the Docker build process is related to the logging configuration in the code. The logging setup includes a provider called **AddEventLog**, which is specific to Windows environments. However, when trying to create a Docker container image, the process was running on a non-Windows system, likely a Linux-based one commonly used for Docker containers.

**Resolution:** To address this issue, we needed to enhance the logging configuration to make it adaptable to different operating systems, ensuring seamless operation on both Windows and Linux. The updated code uses a conditional check to determine the operating system type and configures the logging accordingly.

**Technical Details:** The logging configuration is part of a broader process where we set up the application's environment using the .NET Core framework. The **CreateHostBuilder** method is a standard entry point in a .NET Core application responsible for configuring and building the application's host, which encompasses things like setting up logging, configuring services, and specifying how the application should run.

In this specific case, we're dealing with logging configuration. If the operating system is Windows, the code configures the logging to use the **AddEventLog** provider, which writes log entries to the Windows Event Log. On non-Windows systems, the code can be extended to use alternative logging providers suitable for those platforms.

**Next Steps:**

1. Ensure the corrected code is implemented in the application.
2. During Docker image builds on Windows, the logging will include the **AddEventLog** provider, capturing log entries in the Windows Event Log.
3. On other platforms (such as Linux), the code gracefully handles the absence of the **AddEventLog** provider and can be extended to include alternative logging providers, ensuring compatibility in diverse deployment environments.

This resolution not only addresses the specific Docker build issue but also enhances the application's portability and adaptability to different operating systems. It's a crucial step towards making the application more robust and deployable in various environments.

This resolution not only addresses the specific Docker build issue but also enhances the application's portability and adaptability to different operating system

Demonstrate the seamless migration of a long-standing .NET Core Interaction Services application from traditional Windows servers to AWS ECS Fargate, showcasing the potential cost savings and enhanced deployment agility through containerization. Validate the application's successful adaptation to the Fargate environment, ensuring optimal resource utilization and efficient orchestration within the AWS ecosystem. Identify and address any challenges or findings during the POC to refine the migration strategy and ensure a smooth transition