**1. Azure App Service (Web Apps)**

* Host your .NET Core C# backend on Azure App Service. This provides seamless **hosting** for .NET applications with built-in features like automatic scaling, load balancing, and security.

**2. Azure SQL Database or PostgreSQL**

* You can choose between Azure SQL Database (fully managed SQL Server) or **Azure PostgreSQL**. If you are familiar with SQL Server, Azure SQL Database is a great fit, but Azure PostgreSQL is also a viable option since the project initially included PostgreSQL.

**3. Azure Maps**

* As before**, Azure Maps** will provide the necessary services for the interactive map, integrating location-based data and geospatial services with your .NET Core backend.

**4. Azure Blob Storage**

* Use Blob Storage to store media assets (e.g., **business photos**) and any files that need to be uploaded. Blob Storage is fully compatible with .NET SDKs, making it easy to integrate with your backend.

**5. Azure Active Directory (AAD)**

* Implement user authentication with Azure AD B2C to handle user account management and secure logins. Azure AD works well with .NET Core applications and supports **OAuth, OpenID, and other protocols.**

**6. Azure Cognitive Services (Content Moderator)**

* Use Azure's Content Moderator to automatically moderate user reviews, ensuring appropriate content. This service integrates easily into .NET applications via REST APIs or SDKs.

**7. Azure Functions**

* Use Azure Functions to handle background tasks such as sending notifications or processing data in real time. Azure Functions are ideal for lightweight, serverless event-driven processes that integrate well with .NET.

**8. Azure Application Insights**

* Monitor your .NET Core app **performance** using Azure Application Insights, which provides detailed telemetry, error reporting, and real-time analytics to help you track user behavior and system health.

**9. Azure Logic Apps**

* Automate workflows like sending notifications when reviews are added or **businesses are updated**. Logic Apps can be easily triggered by events in your .NET Core backend.

**10. Azure Repos:**

* Store your codebase using Git repositories for tracking changes, collaboration, and managing branches (e.g., feature, release branches).

**Azure Pipelines (CI/CD):**

1. Continuous Integration (CI):
   * Automatically build and test your .NET Core application whenever changes are pushed to the repository. It ensures that your code is always in a deployable state.
2. Continuous Delivery (CD):
   * Set up pipelines to automatically deploy your application to Azure App Service or other environments after passing tests. This ensures seamless and frequent updates.