Serverless application with .NET

Azure Functions with .NET

Azure Functions enables serverless computing in the Azure cloud, supporting multiple triggers (e.g., HTTP, Timer, Queue).

Steps:

1. **Set Up Azure CLI & Tools**:
   - Install the Azure CLI.
   - Install the Azure Functions extension for Visual Studio or Visual Studio Code.
2. **Create an Azure Function**:
   - Use the Azure Functions template in Visual Studio (Azure Functions Project).
   - Select a trigger type (e.g., HTTP trigger).

3.i) HTTP Trigger

An **HTTP Trigger** allows your function to be invoked via an HTTP request. It is commonly used to build APIs or webhooks.

- **Use Case**: This trigger is useful when you want your function to respond to web requests. For example, you can create an API endpoint that responds to GET or POST requests, processes incoming data, and returns a result.

```
        public static class Function1
{
  [FunctionName("AzureFunction")]
  public static async Task<IActionResult> Run(
    [HttpTrigger(AuthorizationLevel.Function, "get", "post", Route = null)] HttpRequest req,
    ILogger log)
  {
    log.LogInformation("C# HTTP trigger function processed a request.");

    string name = req.Query["name"];
```

```
        string requestBody = await new StreamReader(req.Body).ReadToEndAsync();
        dynamic data = JsonConvert.DeserializeObject(requestBody);
        name = name ?? data?.name;


        string responseMessage = string.IsNullOrEmpty(name)
            ? "This HTTP triggered function executed successfully. Pass a name in the query string or in the request body for a personalized response."
            : $"Hello, {name}. This HTTP triggered function executed successfully.";


        return new OkObjectResult(responseMessage);
    }
}
```

ii. **Timer Trigger**

A **Timer Trigger** allows you to run your function on a schedule. You can configure it using a cron expression or a predefined schedule in your function's configuration.

- **Use Case**: Timer triggers are often used for background jobs, such as cleaning up old data, sending periodic emails, or performing regular data processing tasks.
- **Example**: Suppose you want a function to run at midnight every day:

```
public static class TimerTriggeredFunction
{
    [FunctionName("TimerTriggeredFunction")]
    public static void Run([TimerTrigger("0 0 0 * * *")] TimerInfo myTimer, ILogger log)
    {
        log.LogInformation($"Timer trigger function executed at: {DateTime.Now}");
    }
}
```

iii) **Queue Trigger**

A **Queue Trigger** allows your function to execute when a message is added to an Azure Storage Queue. This is useful for processing tasks asynchronously.

- **Use Case**: Queue triggers are commonly used for background processing, such as processing orders, sending emails, or handling long-running tasks. When an event occurs that doesn't need immediate processing, you can queue it, and the function will handle it later.
- **Example**: Suppose you have a queue named myqueue in Azure Storage, and you want to process each message:

```
public static class QueueTriggeredFunction
{
   [FunctionName("QueueTriggeredFunction")]
   public static void Run([QueueTrigger("myqueue", Connection = "AzureWebJobsStorage")] string myQueueItem, ILogger log)
   {
      log.LogInformation($"Queue trigger function processed: {myQueueItem}");
   }
}
```

4. **Deploy to Azure**:

- Right-click the project in Visual Studio and choose "Publish."
- Follow the prompts to publish your function to Azure.

5. **Set Up Triggers & Bindings**:

- Azure Functions can be triggered by HTTP requests, timers, queues, etc. You can also bind inputs/outputs to services like Azure Storage.

Benefits of Serverless with .NET

- **Scalability**: Both AWS Lambda and Azure Functions automatically scale based on the number of requests.
- **Cost-Effective**: You only pay for the execution time of your code, making it more cost-effective than traditional servers.