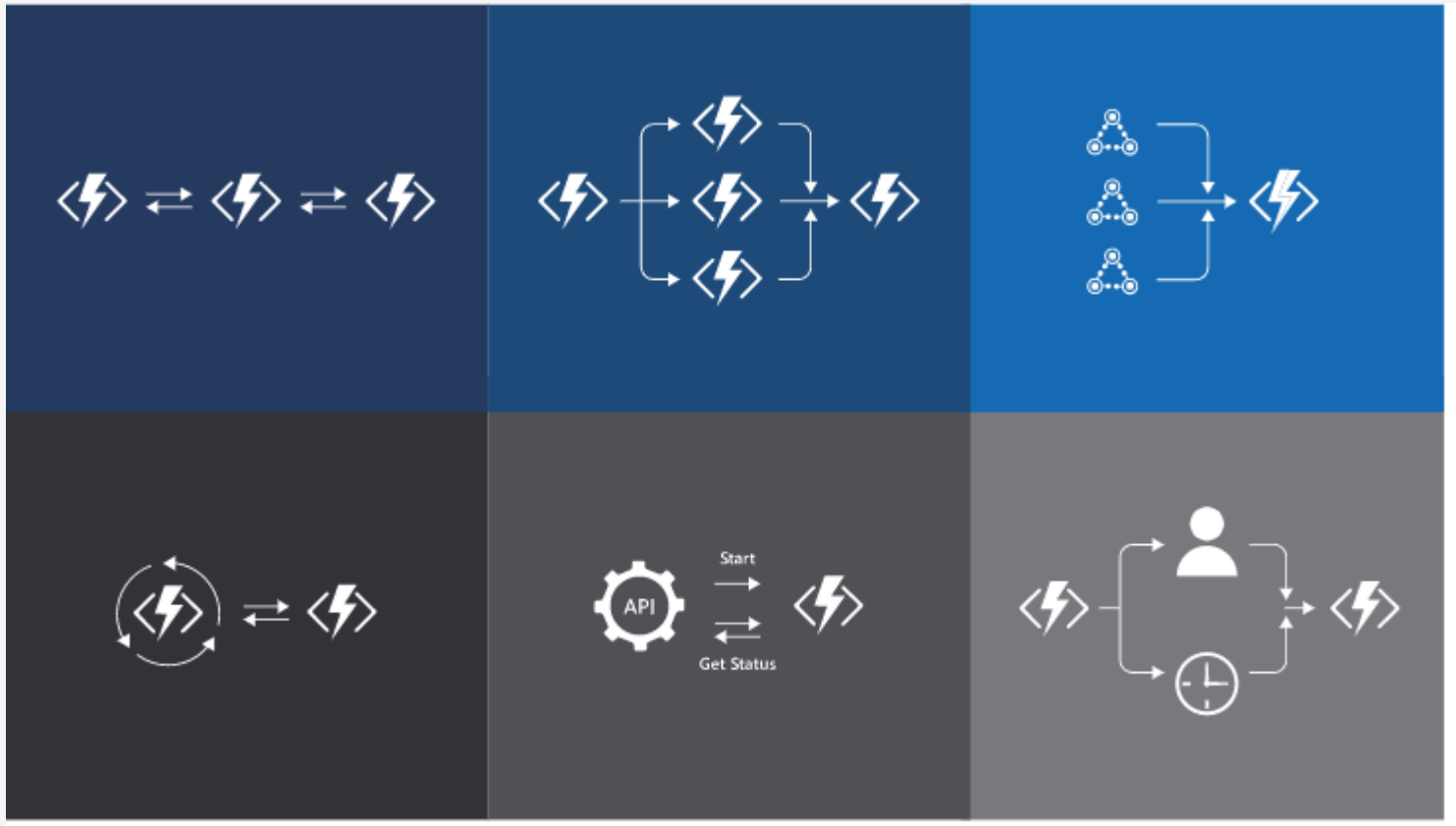
Azure functions

Rajesh Kolla
Full-stack developer ,Azure Architect

Email: razesh.kolla@gmail.com
Twitter: @RajeshKolla18
LinkedIn: https://be.linkedin.com/in/razeshkolla

Agenda

- Overview
- Durable functions
- Best Practices
- Common Scenarios
- Demos
- Pricing
- Wrap up
- Q&A



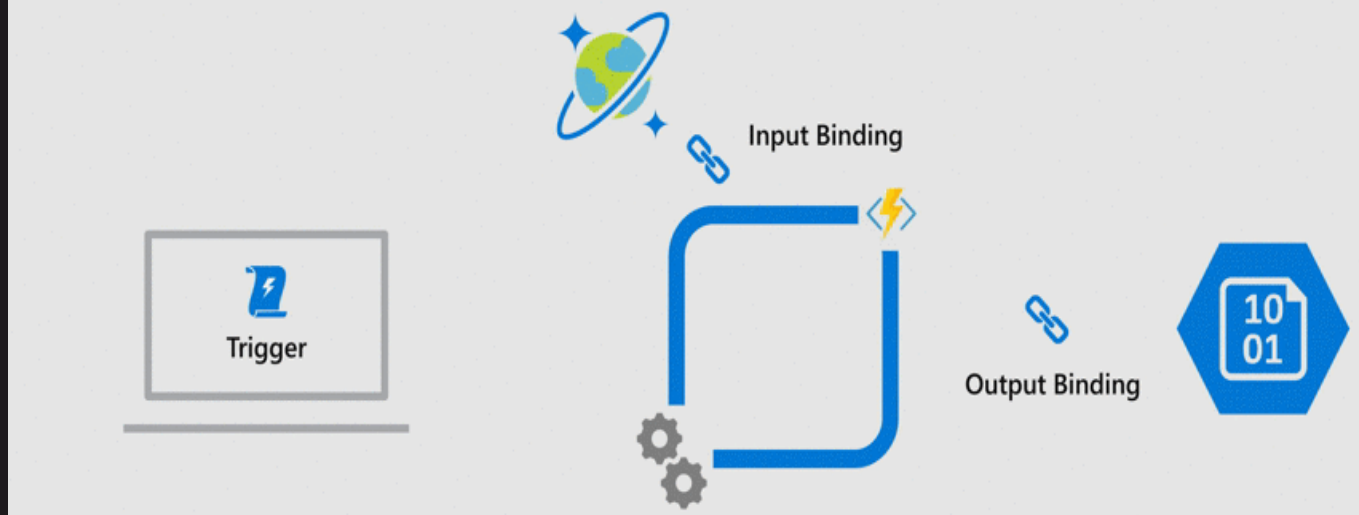
Microsoft Azure serverless compute

- It's now becoming easier than ever to create small, targeted microservice architecture using a variety of services
- Azure provides many services that can help you achieve a low-friction, high-throughput and low-cost solution
- Azure Functions is the newest service in the serverless architecture family
  - Per-second Billing model
  - only pay when your code runs '
- Automatic scale
  - Simpler
  - Cheaper
  - More scalable

Azure function

Azure Function is small pieces of code without worrying about underlying infrastructure .

It is triggered by a specific type of event or running on schedule or as the result of an Http request



Triggers & Binding

- A trigger is action (or) event (or) time-based schedule which defines how function is invoked.
- Binding to the function is a way of connecting another resource to function in input \output \both.
- Triggers and Bindings let developer avoid hardcoding access to other services

- Function must have exactly one trigger
- Triggers have associated data which is often provided

Service	Trigger	Input	Output
Blob Storage	☑	☑	☑
Azure Cosmos DB	☑	☑	☑
Event Grid	☑		☑
Event Hubs	☑		☑
IOT Hub	☑		☑
Http	☑		
Queue Storage	☑		☑
Send Grid			☑
Service Bus	☑		☑
Signal R	☑	☑	☑
Table Storage	☑		☑
Timer	☑		
Twilio			☑

Hosting Models

Consumption Plan (Server less)

- Per-sec billing
- 1m executions
- 400,000 GB-s

App Service Plan

- Reserved servers
- Predictable monthly costs

Premium Plan

- Pre-warmed instances
- VNet integration
- Longer run duration

Docker Container

- Run Anywhere
- On premises
- Other cloud providers

Development Environments

Azure portal

- https://portal.azure.com
- Experiments
- Proof of concepts

Visual Studio

- Powerful IDE
- Azure function extension
- Debug and test locally

Azure Function Core Tools

- Cross Platform
- Visual Studio code
- Azure function extension

Supported Languages

- Azure functions can be created in C# , Node/JavaScript, Python, F# ,PHP and scripting languages like PowerShell ,Batch and Bash.

https://www.npmjs.com/package/azure-functions-core-tools

Additional features

Concept of Function App

- Unit of Deployment
- Share common configuration
- Scale together
- Logically related

Security

- API Keys
- Identity Provider integration such as AAD , third party identity providers

Durable Functions

- Define workflows
- Orchestration of functions
- Run tasks in parallel
- Retries and error handling

Proxies

- Route incoming requests
- Visual Studio code
- Transform Request and Response

Best Practices

Performance &Storage Consideration

- Avoid long running functions
- Cross function communication
- Write functions to be stateless
- Write defensive functions
- Avoid sharing storage accounts
- Don't mix test and production code in the same function app
- Use async code but avoid blocking calls\bottle necks
- use multiple worker processes
- Configure host behaviors to better handle concurrency

Durable Function

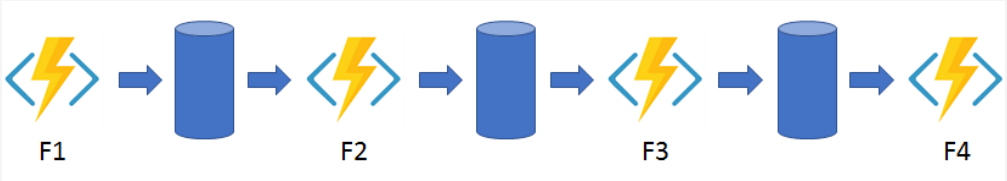
Durable function is an extension of Azure function. Allow developer write stateful function.

Define Stateful workflows by writing orchestrator functions and stateful entities

The main purpose of durable functions is simplifying complex, stateful requirement in serverless applications.

- **Pattern #1: Function chaining:-**
  a sequence of function executes in specific order.

  The output of one function is input of another function.




```
[FunctionName("Chaining")]
public static async Task<object> Run(
    [OrchestrationTrigger] IDurableOrchestrationContext context)
{
    try
    {
        var x = await context.CallActivityAsync<object>("F1", null);
        var y = await context.CallActivityAsync<object>("F2", x);
        var z = await context.CallActivityAsync<object>("F3", y);
        return  await context.CallActivityAsync<object>("F4", z);
    }
    catch (Exception)
    {
        // Error handling or compensation goes here.
    }
}
```

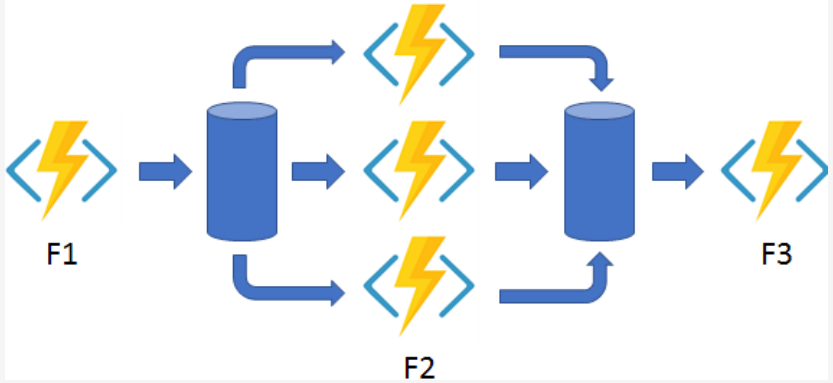
Durable Function

Durable function is an extension of Azure function. Allow developer write stateful function.

Define Stateful workflows by writing orchestrator functions and stateful entities

➢ Pattern #2: Fan out/fan in

In this pattern, multiple functions execute in parallel and then wait for all functions to finish and then aggregate the result.




```
[FunctionName("FanOutFanIn")]
public static async Task Run(
    [OrchestrationTrigger] IDurableOrchestrationContext context)
{
    var parallelTasks = new List<Task<int>>();

    // Get a list of N work items to process in parallel.
    object[] workBatch = await context.CallActivityAsync<object[]>("F1", null);
    for (int i = 0; i < workBatch.Length; i++)
    {
        Task<int> task = context.CallActivityAsync<int>("F2", workBatch[i]);
        parallelTasks.Add(task);
    }

    await Task.WhenAll(parallelTasks);

    // Aggregate all N outputs and send the result to F3.
    int sum = parallelTasks.Sum(t => t.Result);
    await context.CallActivityAsync("F3", sum);
}
```

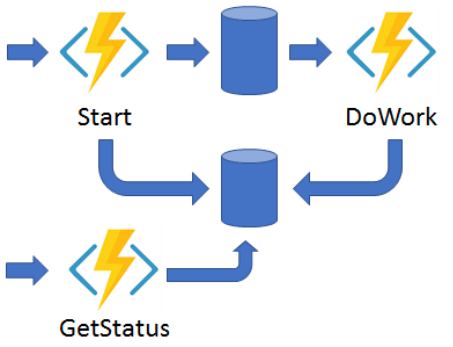
Durable Function

Durable function is an extension of Azure function. Allow developer write stateful function.

Define Stateful workflows by writing orchestrator functions and stateful entities

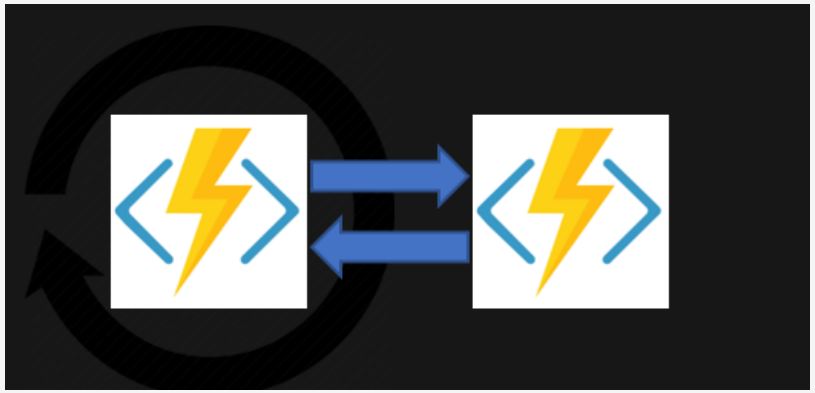
➢ **Pattern #3: Async HTTP APIs**

The async HTTP API pattern addresses the problem of coordinating the state of long-running operations with external



➢ **Pattern #4: Monitor**

recurring process in a workflow. An example is polling until specific conditions are met.



Durable Function

Durable function is an extension of Azure function. Allow developer write stateful function.

Define Stateful workflows by writing orchestrator functions and stateful entities

```
[FunctionName("MonitorJobStatus")]
public static async Task Run(
    [OrchestrationTrigger] IDurableOrchestrationContext context)
{
    int jobId = context.GetInput<int>();
    int pollingInterval = GetPollingInterval();
    DateTime expiryTime = GetExpiryTime();

    while (context.CurrentUtcDateTime < expiryTime)
    {
        var jobStatus = await context.CallActivityAsync<string>("GetJobStatus", jobId);
        if (jobStatus == "Completed")
        {
            // Perform an action when a condition is met.
            await context.CallActivityAsync("SendAlert", machineId);
            break;
        }

        // Orchestration sleeps until this time.
        var nextCheck = context.CurrentUtcDateTime.AddSeconds(pollingInterval);
        await context.CreateTimer(nextCheck, CancellationToken.None);
    }

    // Perform more work here, or let the orchestration end.
}
```

Durable Function

Durable function is an extension of Azure function. Allow developer write stateful function.

Define Stateful workflows by writing orchestrator functions and stateful entities

➢ Pattern #5: Human interaction

this pattern is mainly used in automated process involve human interaction


```
[FunctionName("MonitorJobStatus")]
public static async Task Run(
    [OrchestrationTrigger] IDurableOrchestrationContext context)
{
    int jobId = context.GetInput<int>();
    int pollingInterval = GetPollingInterval();
    DateTime expiryTime = GetExpiryTime();

    while (context.CurrentUtcDateTime < expiryTime)
    {
        var jobStatus = await context.CallActivityAsync<string>("GetJobStatus", jobId);
        if (jobStatus == "Completed")
        {
            // Perform an action when a condition is met.
            await context.CallActivityAsync("SendAlert", machineId);
            break;
        }

        // Orchestration sleeps until this time.
        var nextCheck = context.CurrentUtcDateTime.AddSeconds(pollingInterval);
        await context.CreateTimer(nextCheck, CancellationToken.None);
    }

    // Perform more work here, or let the orchestration end.
}
```

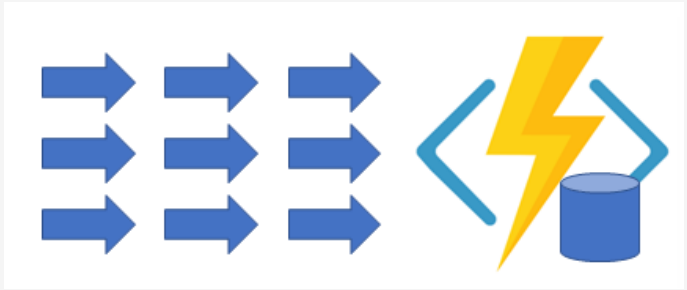
Durable Function

Durable function is an extension of Azure function. Allow developer write stateful function.

Define Stateful workflows by writing orchestrator functions and stateful entities

➢ Pattern #6: Aggregator (stateful entities)

this is used for aggregating event data over a period of time into single entity



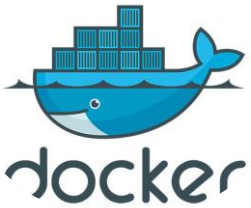
```
[FunctionName("Counter")]
public static void Counter([EntityTrigger] IDurableEntityContext ctx)
{
    int currentValue = ctx.GetState<int>();
    switch (ctx.OperationName.ToLowerInvariant())
    {
        case "add":
            int amount = ctx.GetInput<int>();
            ctx.SetState(currentValue + amount);
            break;
        case "reset":
            ctx.SetState(0);
            break;
        case "get":
            ctx.Return(currentValue);
            break;
    }
}
```

Demos

Create Function in Portal

Create Function in Visual studio

Create Function in Visual code



Running Functions in Containers

Working with Triggers and Binding

Q&A

Email: razesh.kolla@gmail.com
Twitter: @RajeshKolla18
LinkedIn: https://be.linkedin.com/in/razeshkolla

Thank you!

Email: razesh.kolla@gmail.com
Twitter: @RajeshKolla18
LinkedIn: https://be.linkedin.com/in/razeshkolla