



Azure Durable Functions

S. Kyle Korndoerfer

About Me



- Co-Leader of .NET Rochester (formerly VDUNY)
- RIT Graduate - BioMedical Computing (2000)
- Working in software for 20+ yrs, primarily .NET & Web
- Part-time Pyrotechnician (23+ yrs)
- Assistant Scoutmaster, Troop 750

Azure Functions

A brief re-cap

Azure Functions

- "Serverless" computing offering for Azure
 - You write the code and upload it; Azure does the rest
- Available in several languages (C#, Java, JavaScript, Python, PowerShell)
- Cross platform development & hosting
 - Windows, MacOS, Linux
- Available in 2 different pricing models
 - Premium Plan (pre-warmed instances always running + scale up)
 - App Service Plan (alongside existing App Service instance w/o added \$)
 - Consumption (pay-per-use; generous free allotment; 1M/400,000GB-s)
- Stateless
- Time-limited!
 - (5min; extendable to 10)
- Functions runtime is Open Source on GitHub

Azure Functions Triggers

- **HTTP**
 - web-requests; micro-services
- **Timer**
 - Scheduled runs
- **Azure CosmosDB**
 - New/updated documents
- **Blob storage**
 - New/modified blobs
- **Queue storage**
 - New messages
- **Event Grid**
 - Events via subscriptions/filters
- **Event Hub**
 - High volumes of events
- **Service Bus Queue/Topic**
- **Many others**



Quick Demo

Basic Azure Function

Azure Durable Functions

What is a durable function?

Azure Durable Functions

- Stateful functions in a serverless environment
 - Create workflows using **Orchestrator** functions that invoke single purpose **Activity** functions
 - Stateful Entities using **Entity** functions
- *Not time limited!*
- Language support
 - C#, JavaScript, F#
 - (goal to support same as Functions)

Azure Durable Functions

- Consists of 3 types of functions:
 - **Starter** function(s)
 - Entry point that starts an orchestrator
 - Http, Queue, etc.
 - Can have more than 1
 - **Orchestrator** function
 - Main processing loop
 - Must be deterministic (no calls to `DateTime.Now()`, `Guid.New()`, etc.)
 - Re-entrant (executes multiple times)
 - Composes Activity functions into a workflow
 - **Activity** function(s)
 - 'Units of work' such as calling external services, non-deterministic code, etc.

Starter Functions

- Entry point for the Orchestrator
- Triggered like a standard Azure Function
- Kicks off the Orchestrator function
- Optionally returns info for monitoring the Orchestrator progress
 - REST service that can be queried given an Instance ID

Sample Starter Function

```
[FunctionName("Function1_HttpStart")]
```

0 references

```
public static async Task<HttpResponseMessage> HttpStart(  
    [HttpTrigger(AuthorizationLevel.Anonymous, "get", "post")] HttpRequestMessage req,  
    [DurableClient] IDurableOrchestrationClient starter,  
    ILogger log)  
{  
    // Function input comes from the request content.  
    string instanceId = await starter.StartNewAsync("Function1", null);  
  
    log.LogInformation($"Started orchestration with ID = '{instanceId}'.");  
  
    return starter.CreateCheckStatusResponse(req, instanceId);  
}
```

Orchestrator Functions

- Workflow management
 - Which Actions to invoke
 - What order to run them in
 - Invoking *sub*-orchestrations
 - Waiting for external input
- Re-entrant
 - Orchestrator runs multiple times, picking up where it left off each time

Sample Orchestrator Function

[FunctionName("Function1")]

0 references

```
public static async Task<List<string>> RunOrchestrator(
    [OrchestrationTrigger] IDurableOrchestrationContext context)
{
    var outputs = new List<string>();

    // Replace "hello" with the name of your Durable Activity Function.
    outputs.Add(await context.CallActivityAsync<string>("Function1_Hello", "Tokyo"));
    outputs.Add(await context.CallActivityAsync<string>("Function1_Hello", "Seattle"));
    outputs.Add(await context.CallActivityAsync<string>("Function1_Hello", "London"));

    // returns ["Hello Tokyo!", "Hello Seattle!", "Hello London!"]
    return outputs;
}
```

Activity Functions

- Basic 'unit of work' in a workflow
- Can perform network calls, CPU intensive operations, etc.
- Can return data back to the Orchestrator
- Durable Task framework ensure activity will be run *at least once*
 - Activities should be **idempotent**
 - Multiple invocations should have the same outcome (similar to REST guidelines)

Sample Activity Function

```
[FunctionName("Function1_Hello")]
```

0 references

```
public static string SayHello([ActivityTrigger] string name, ILogger log)
{
    log.LogInformation($"Saying hello to {name}.");
    return $"Hello {name}!";
}
```



Quick Demo

Basic Durable Function

Patterns

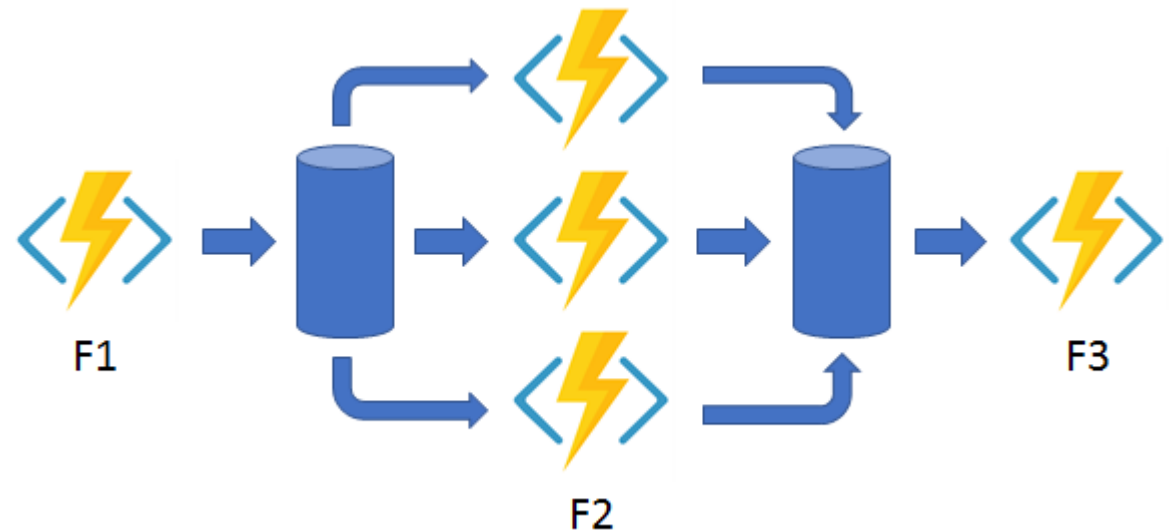
Application patterns made easier with Durable Functions

Supported Application Patterns

- Durable Functions make the following patterns easier:
 - Function Chaining (previous demo)
 - Fan-out/Fan-in
 - Async HTTP APIs
 - Monitoring
 - Human Interaction
 - Aggregator (stateful entities)
 - Etc.

Patterns: Fan-out/in

- Execute a series of Activities in parallel
- Collate results before moving on





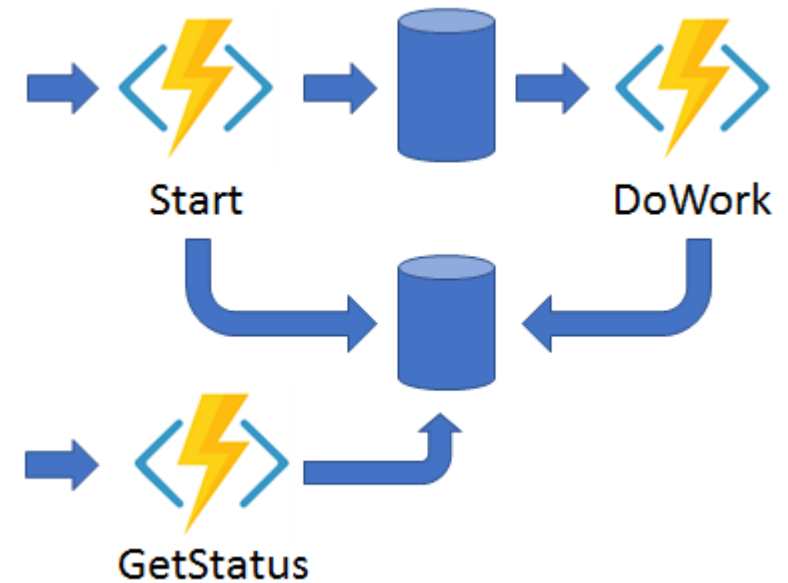
Quick Demo

Fan Out / Fan In Pattern

Async HTTP APIs

- REST API to trigger orchestrations directly
- WebHooks to query the status, cancel, etc.
- HTTP Trigger returns URLs that can be used for this
- Ensure there is an expiration!

POST <https://myfunc.azurewebsites.net/orchestrators/DoWork>





Quick Demo

Async HTTP APIs

Pattern: Human Interaction

- Pauses the orchestration while waiting for an external event
- REST endpoint for providing the response
- Can be raised from another function
 - `await client.RaiseEventAsync(instanceId, "EventName", eventData);`
- Ensure there is a timeout!





Quick Demo

Human Interaction

Closing Thoughts

- Durable Entities
 - Define operations for reading and updating small pieces of state
 - All operations on a single entity are guaranteed to execute serially
 - Behave like eternal orchestrations
- Cleanup
 - History for orchestrators/entities is maintained in Azure Storage Account (tables, blobs, queues)
 - Manual process currently to clean this up using `DurableOrchestrationClient.PurgeInstanceHistoryAsync()`

Resources

- Documentation
 - <https://docs.microsoft.com/en-us/azure/azure-functions/durable/>
- Durable Function Quick start
 - <https://docs.microsoft.com/en-us/azure/azure-functions/durable/durable-functions-create-first-csharp?pivots=code-editor-visualstudio>
- Making sense of Durable Functions
 - <https://mikhail.io/2018/12/making-sense-of-azure-durable-functions/>
- Understanding Durable Functions
 - [Part 1](#), [Part 2](#), [Part 3](#), [Part 4](#), [Part 5](#), [Part 6](#), [Part 7](#), [Part 8](#), [Part 9](#), [Part 10](#), [Part 11](#), [Part 12](#)
- Azure Durable Functions: Before and After
 - <https://medium.com/@ThisisZone/azure-durable-functions-before-and-after-b7266d51ed4d>