Take your Azure Functions

to the next Level

with Durable Functions !

Marc Duiker

Lead Azure Consultant @ Xpirit

ServerLess → Control flow

↙

Durable Functions

↙

Local Development → Use cases & Demos

Rise of the "Serverless"

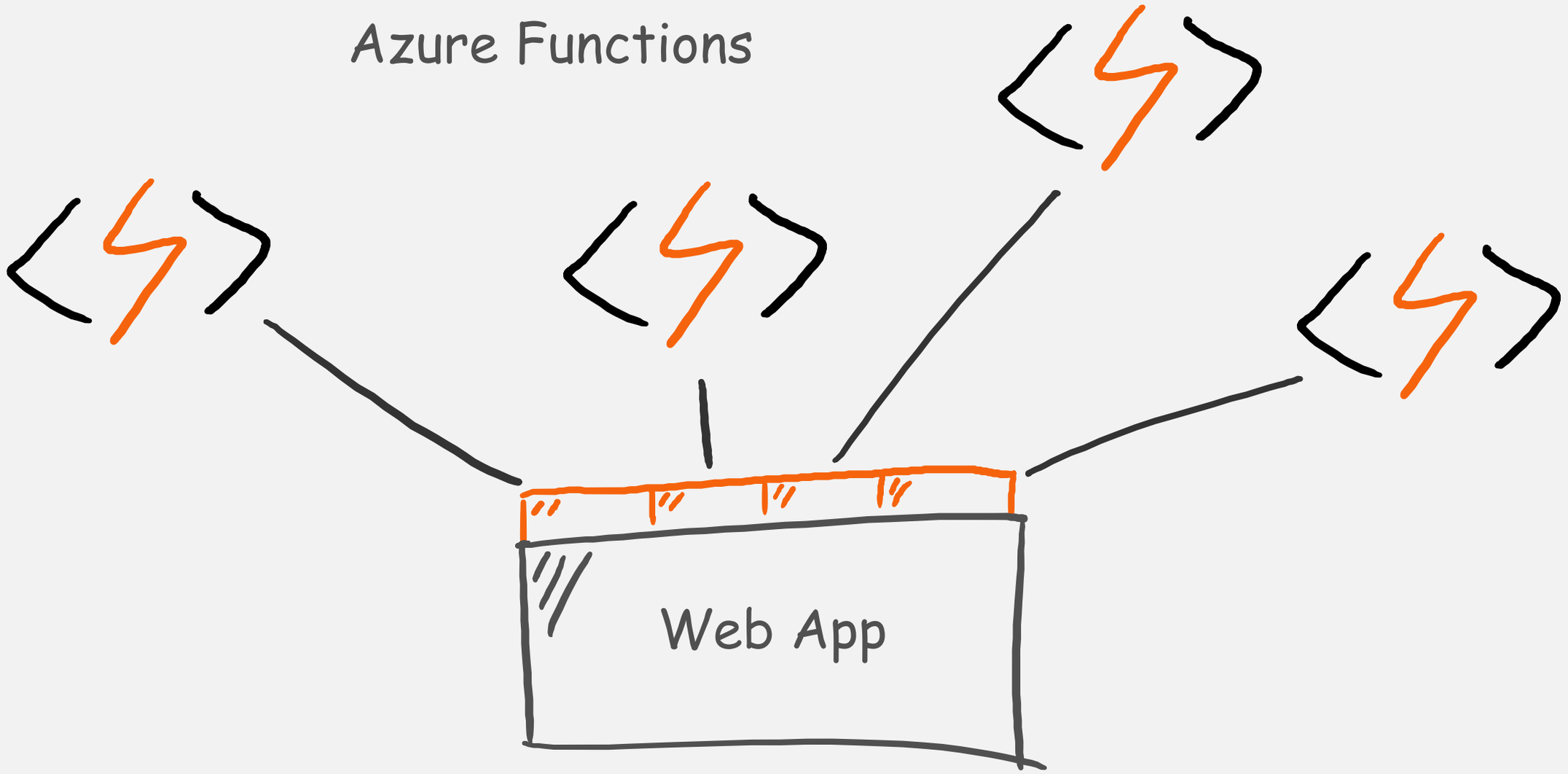
Troy Hunt
@troyhunt

Follow

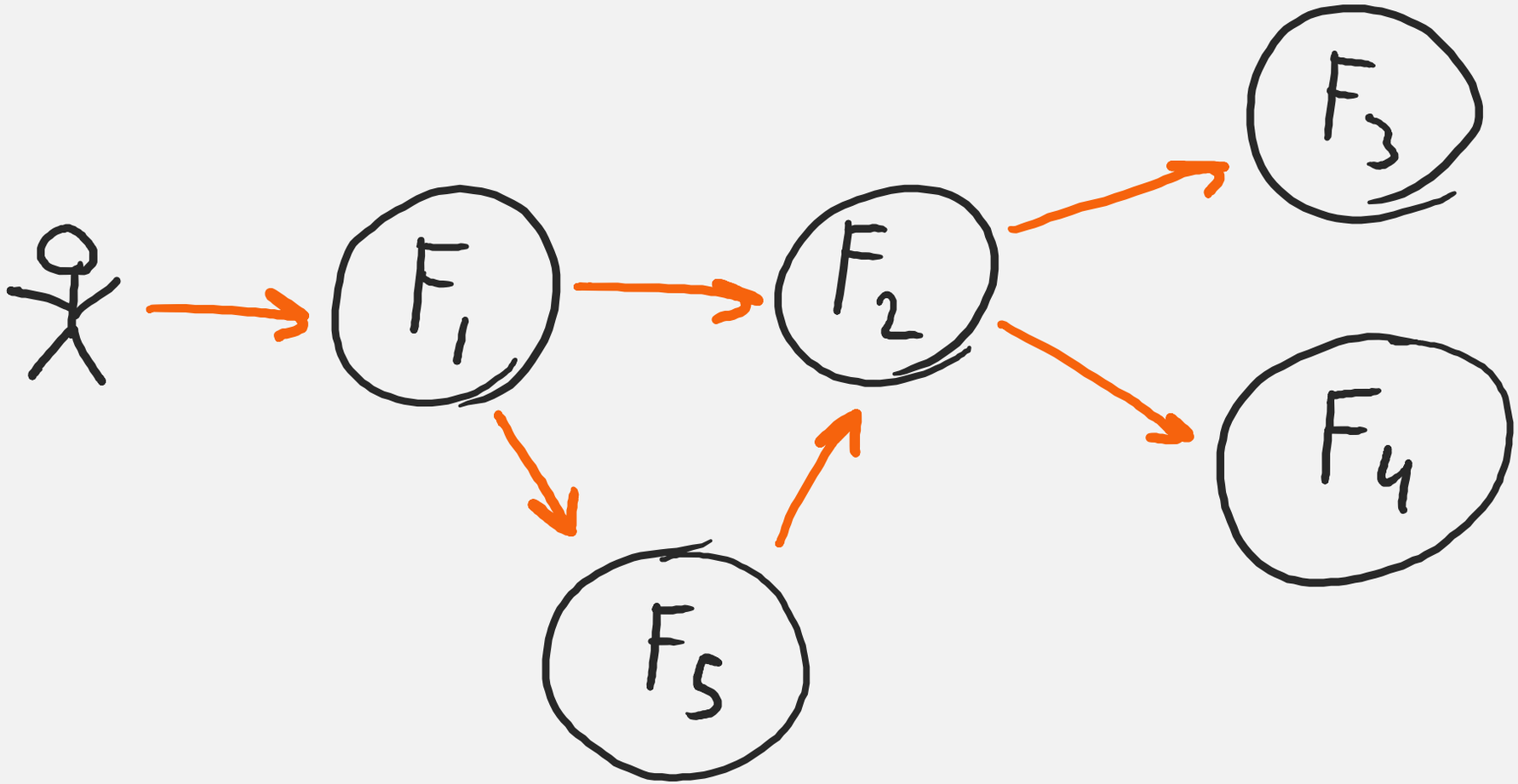
I've been moving *a lot* of @haveibeenpwned stuff to @AzureFunctions lately which has massively cut my app services costs (hundreds per month). Just checked my Function usage for the last month: 77,500,000 executions and 1,580,000,000,000 execution units which costs... $33.59 😎

12:27 AM - 26 Oct 2018

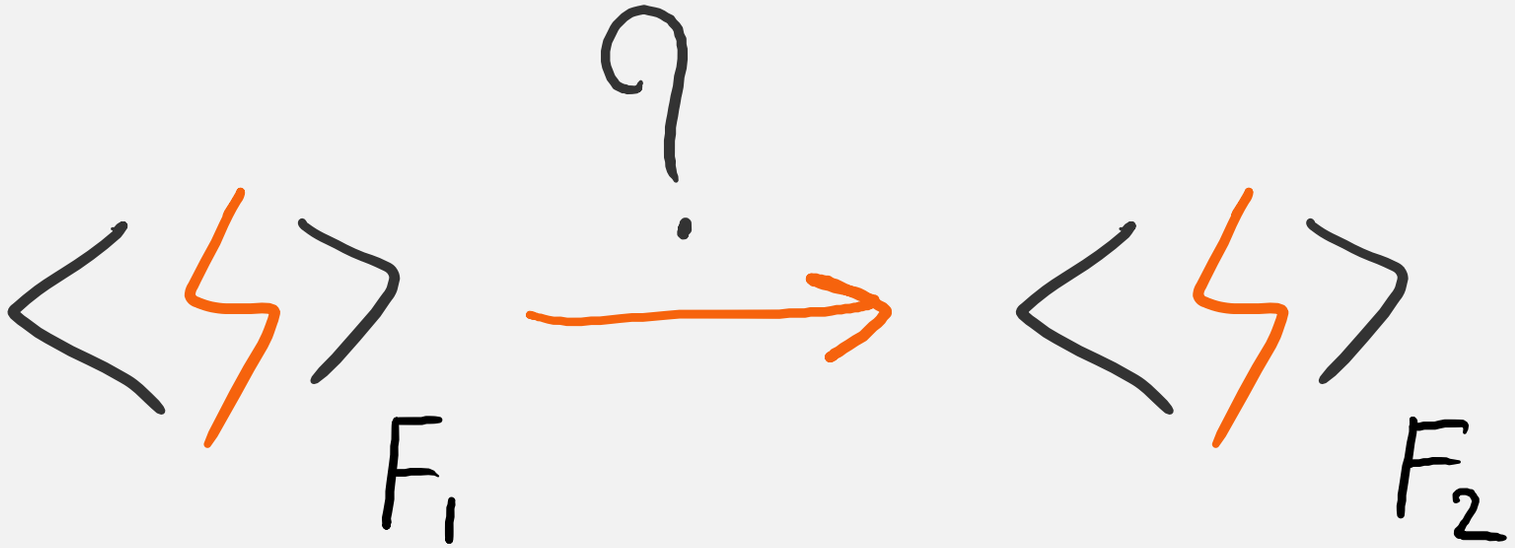
211 Retweets **953** Likes

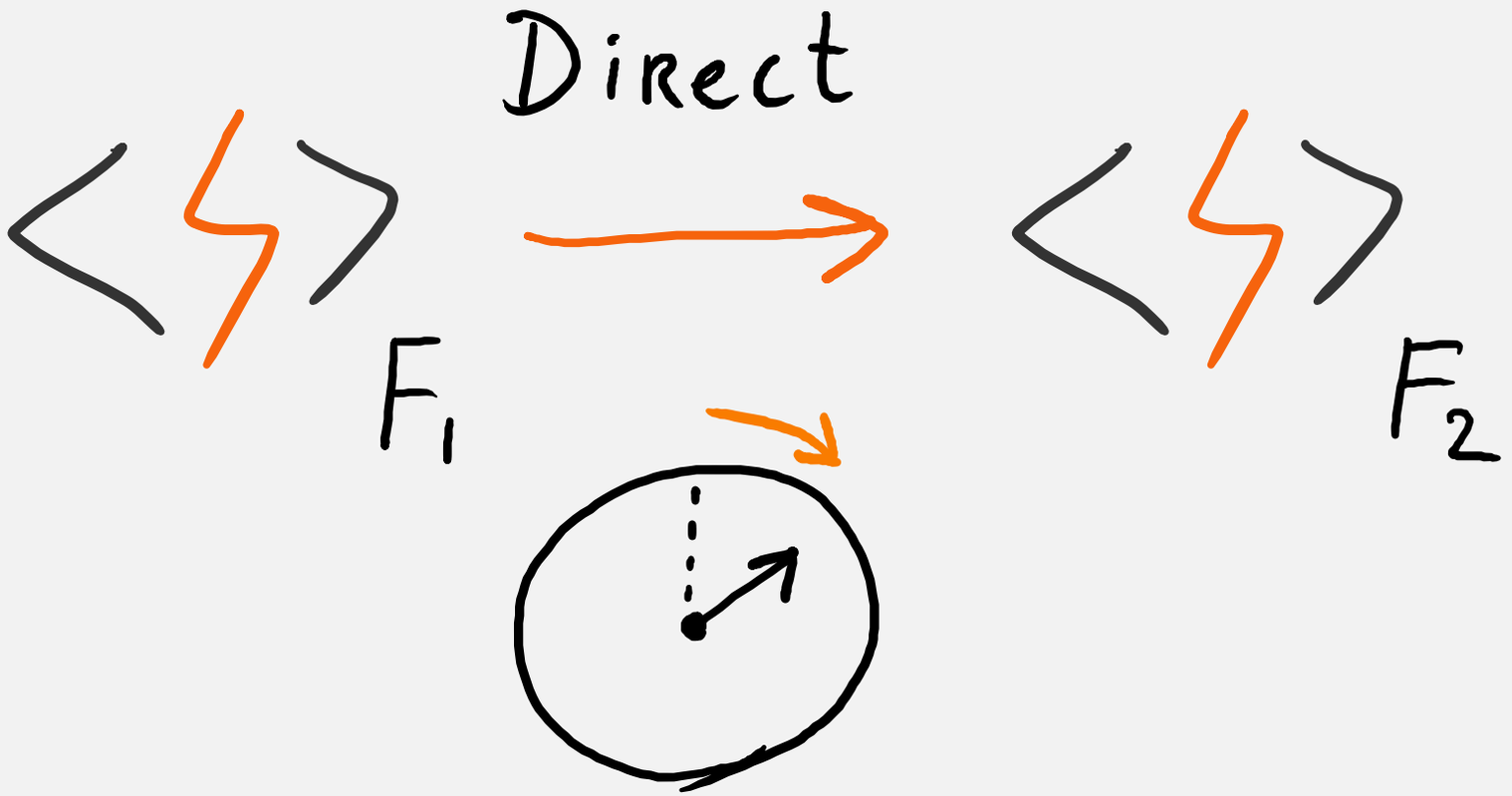

Azure Functions
Web App

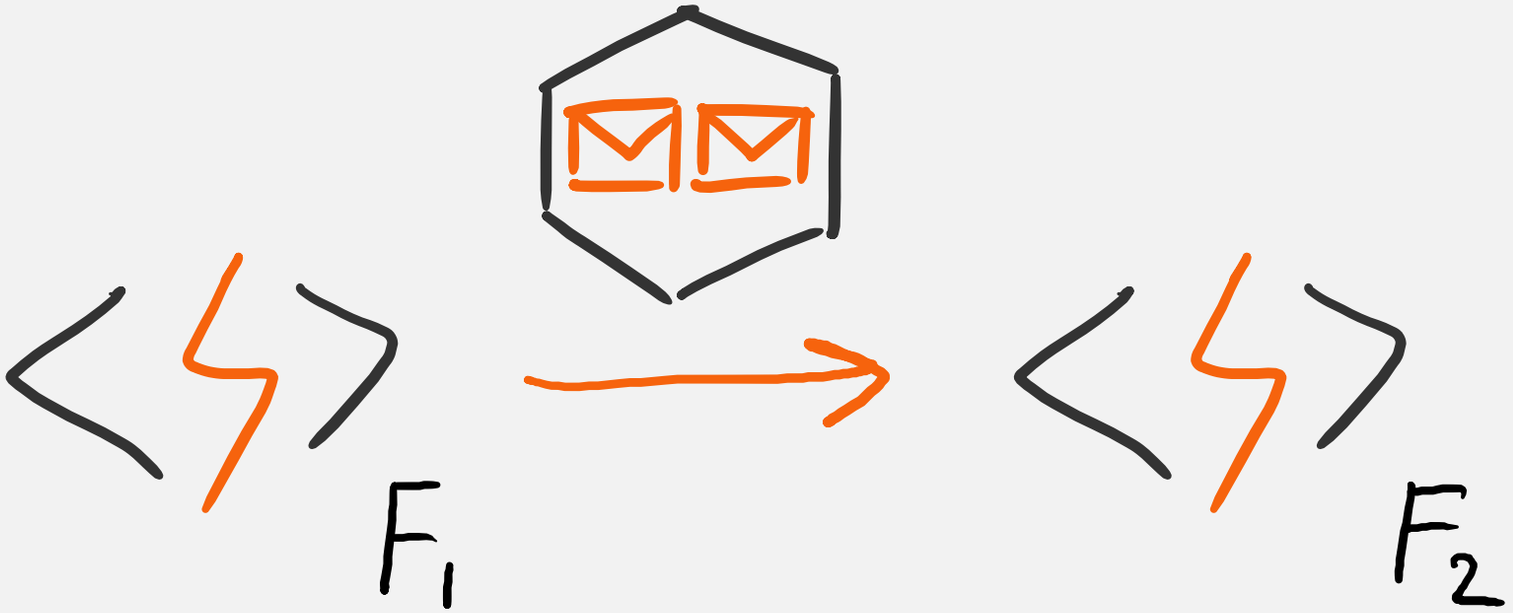
Multiple Functions & Control Flow

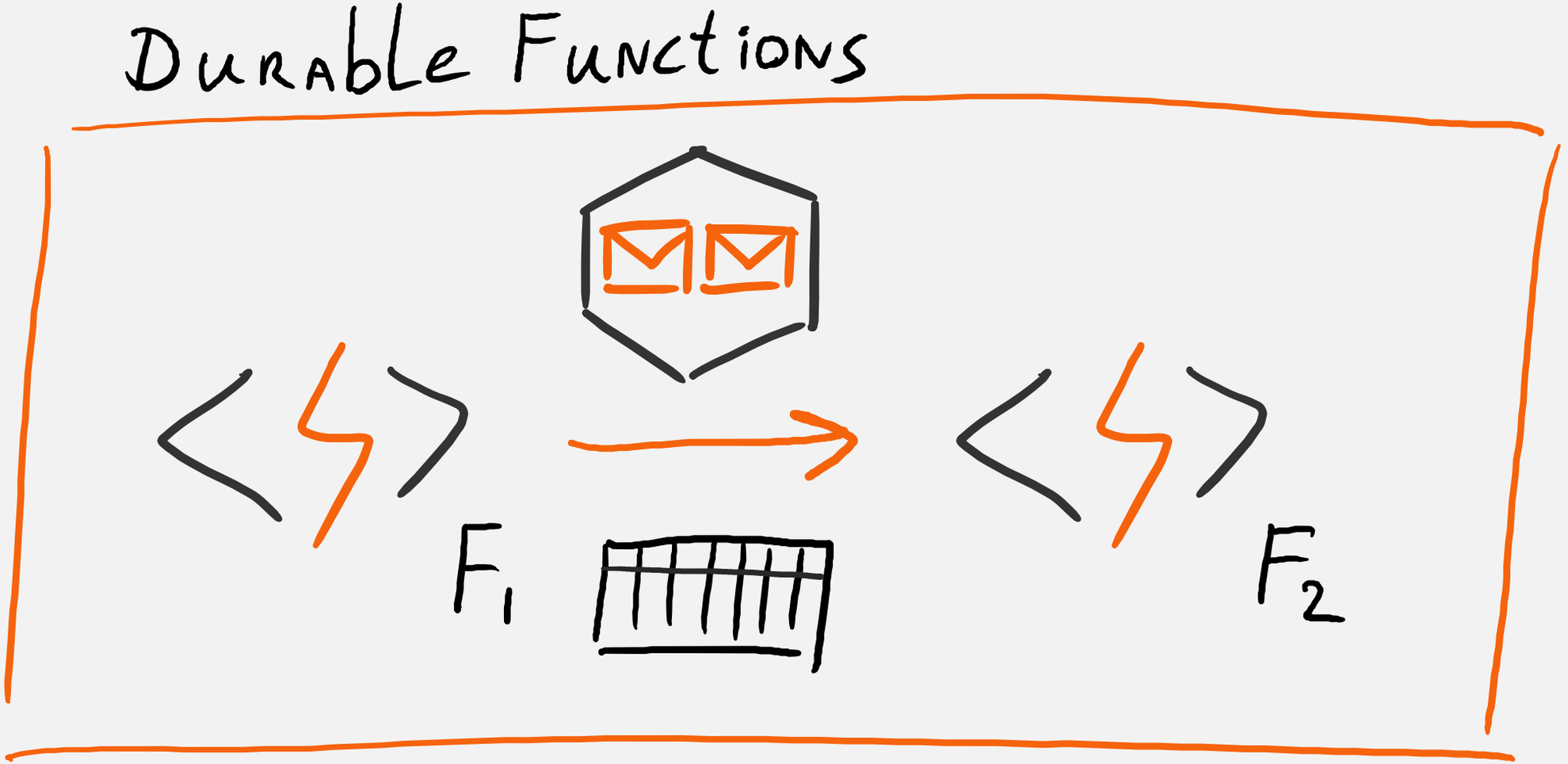


How to define the control flow?

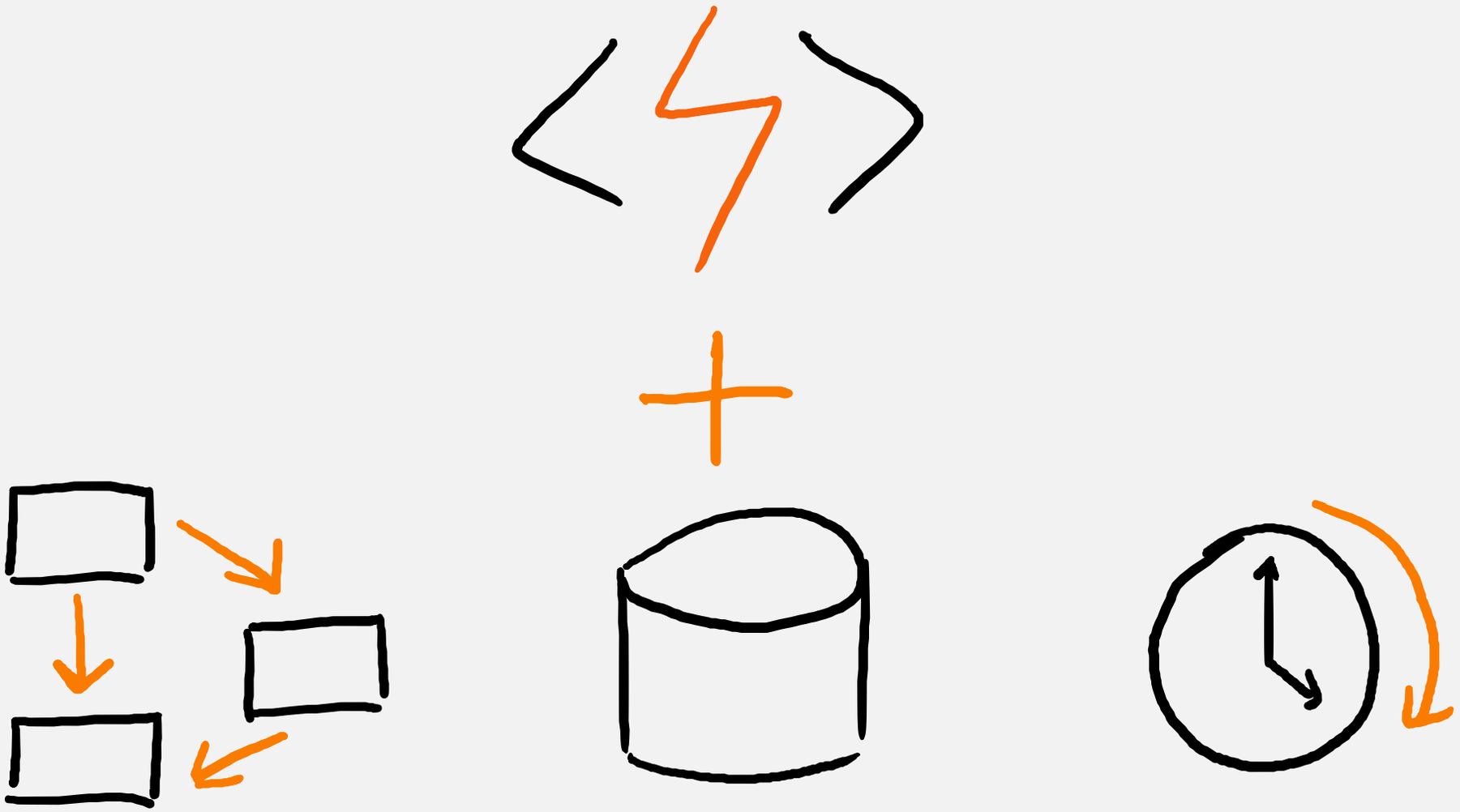

?
F1
F2


Direct
F1
F2


F1
F2


Durable Functions
F1
F2

What is Durable Functions?



Durable Functions is based on

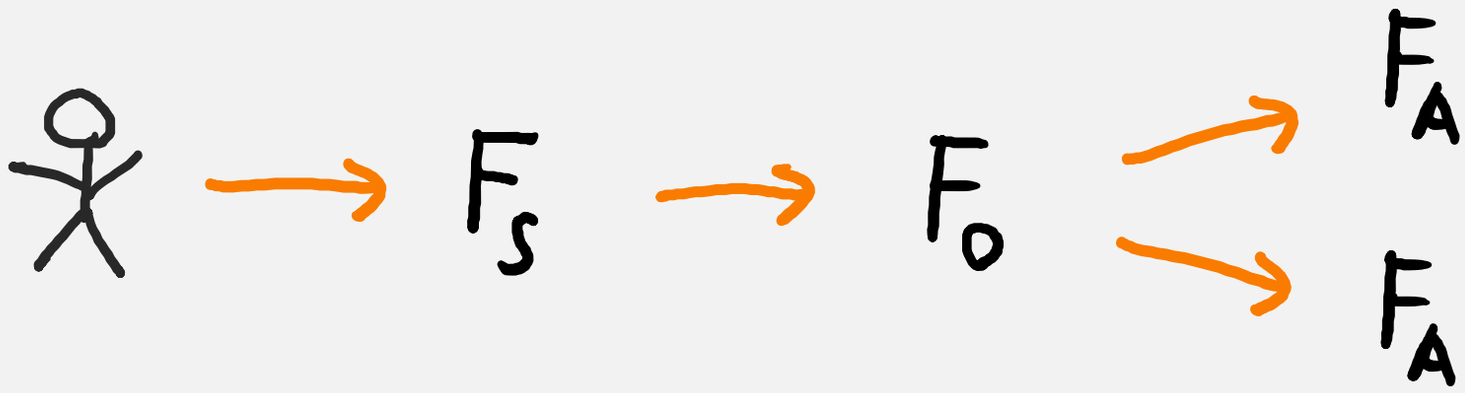
Durable Task Framework

gitter | join chat

This framework allows users to write long running persistent workflows in C# using the async/await capabilities.

It is used heavily within various teams at Microsoft to reliably orchestrate long running provisioning, monitoring and management operations. The orchestrations scale out linearly by simply adding more worker machines.

Function Roles

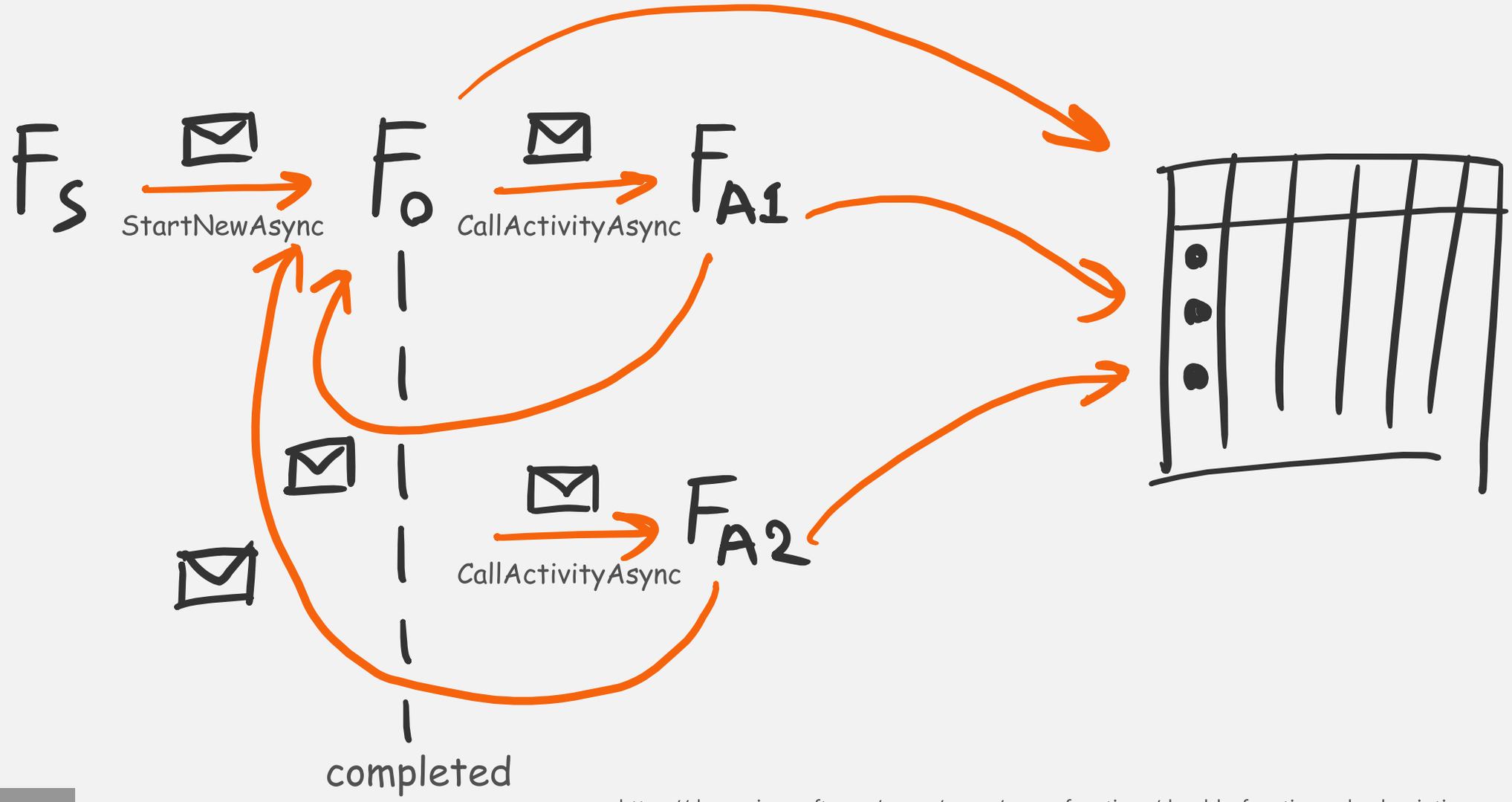


Orchestration Starter Function

Orchestration Function

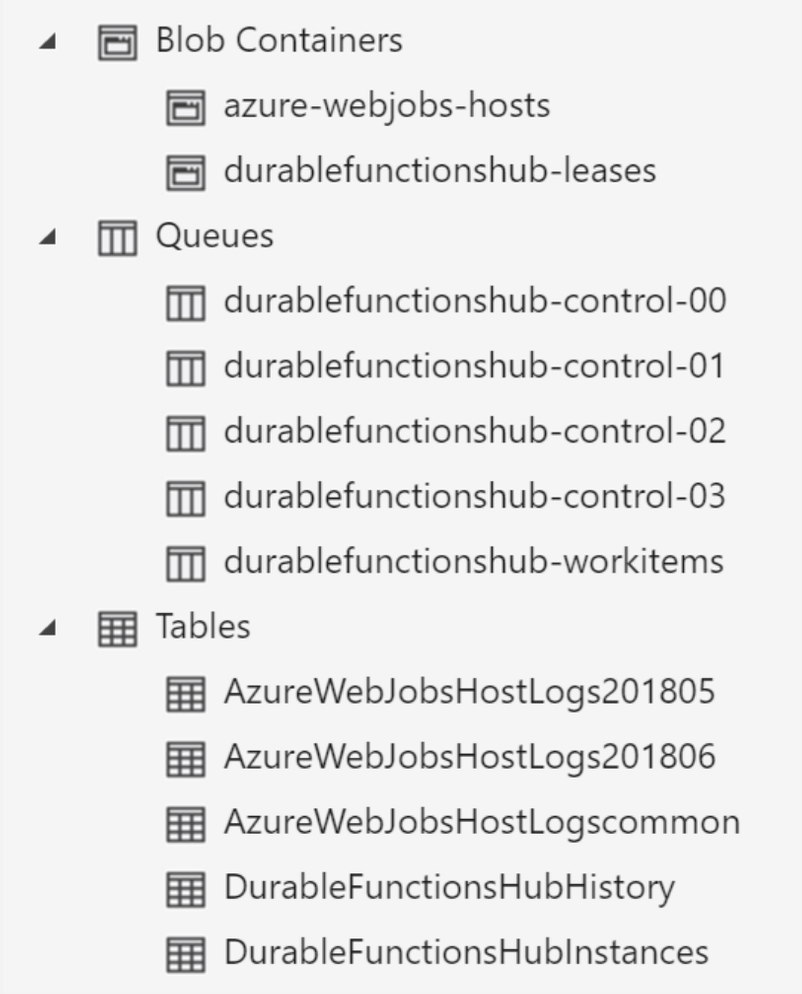
Activity Functions

Invocations, Checkpoints & Replay



https://docs.microsoft.com/en-us/azure/azure-functions/durable-functions-checkpointing-and-replay

Storage Types used by Durable Functions



State is Persisted in Table Storage

PartitionKey	RowKey	Timestamp	EventId	IsPlayed	_Timestamp
498f22754ad849a7be9f559ba1ee2a38	0000000000000000	2018-06-03T21:33:44.297Z	-1	false	2018-06-03T21:33:43.974Z
498f22754ad849a7be9f559ba1ee2a38	0000000000000002	2018-06-03T21:33:44.307Z	0	false	2018-06-03T21:33:44.151Z

EventType	ExecutionId	Name	Version	Input
OrchestratorStarted	0575aa6a829a4716beea4b259c58224d	*null*	*null*	*null*
TaskScheduled	0575aa6a829a4716beea4b259c58224d	HelloNameActivity		*null*
OrchestratorCompleted	0575aa6a829a4716beea4b259c58224d	*null*	*null*	*null*
OrchestratorStarted	0575aa6a829a4716beea4b259c58224d	*null*	*null*	*null*
TaskCompleted	0575aa6a829a4716beea4b259c58224d	*null*	*null*	*null*
ExecutionCompleted	0575aa6a829a4716beea4b259c58224d	*null*	*null*	*null*
OrchestratorCompleted	0575aa6a829a4716beea4b259c58224d	*null*	*null*	*null*
null	0575aa6a829a4716beea4b259c58224d	*null*	*null*	*null*
ExecutionStarted	0575aa6a829a4716beea4b259c58224d	HelloName		"Experts Live 2018"

TaskScheduledId	Result
null	*null*
null	*null*
null	*null*
null	*null*
0	"Hello Experts Live 2018!"
null	"Hello Experts Live 2018!"
null	*null*
null	*null*
null	*null*

{"InstanceId":"498f22754ad849a7be9f559ba1ee2a38","ExecutionId":"0575aa6a829a4716beea4b259c58224d"}

New Trigger Bindings

Function Role: **Orchestration Starter**


```
[OrchestrationClient] DurableOrchestrationClient(Base)
```

```
...public abstract class DurableOrchestrationClientBase
{
    protected DurableOrchestrationClientBase();

    ...public abstract string TaskHubName { get; }

    ...public abstract HttpResponseMessage CreateCheckStatusResponse(HttpRequestMessage request, string ins
    ...public abstract HttpManagementPayload CreateHttpManagementPayload(string instanceId);
    ...public virtual Task<DurableOrchestrationStatus> GetStatusAsync(string instanceId);
    ...public virtual Task<DurableOrchestrationStatus> GetStatusAsync(string instanceId, bool showHistory);
    ...public abstract Task<DurableOrchestrationStatus> GetStatusAsync(string instanceId, bool showHistory,
    ...public abstract Task<IList<DurableOrchestrationStatus>> GetStatusAsync(CancellationToken cancellatio
    ...public abstract Task<IList<DurableOrchestrationStatus>> GetStatusAsync(DateTime createdTimeFrom, Dat
    ...public abstract Task RaiseEventAsync(string instanceId, string eventName, object eventData);
    ...public abstract Task RaiseEventAsync(string taskHubName, string instanceId, string eventName, object
    ...public abstract Task RewindAsync(string instanceId, string reason);
    ...public virtual Task<string> StartNewAsync(string orchestratorFunctionName, object input);
    ...public abstract Task<string> StartNewAsync(string orchestratorFunctionName, string instanceId, objec
    ...public abstract Task TerminateAsync(string instanceId, string reason);
    ...public virtual Task<HttpResponseMessage> WaitForCompletionOrCreateCheckStatusResponseAsync(HttpReque
    ...public abstract Task<HttpResponseMessage> WaitForCompletionOrCreateCheckStatusResponseAsync(HttpRequ
    ...public virtual Task<HttpResponseMessage> WaitForCompletionOrCreateCheckStatusResponseAsync(HttpReque
}
```

New Trigger Bindings

Function Role: **Orchestration Function**


```
[OrchestrationTrigger] DurableOrchestrationContext(Base)
```

```
public abstract class DurableOrchestrationContextBase
{
    protected DurableOrchestrationContextBase();

    public virtual bool IsReplaying { get; internal set; }
    public virtual string ParentInstanceId { get; internal set; }
    public virtual string InstanceId { get; internal set; }
    public abstract DateTime CurrentUtcDateTime { get; }

    public virtual Task CallActivityAsync(string functionName, object input);
    public abstract Task<TResult> CallActivityAsync<TResult>(string functionName, object input);
    public virtual Task CallActivityWithRetryAsync(string functionName, RetryOptions retryOptions, object input);
    public abstract Task<TResult> CallActivityWithRetryAsync<TResult>(string functionName, RetryOptions retryOptions,
    public virtual Task CallSubOrchestratorAsync(string functionName, object input);
    public virtual Task CallSubOrchestratorAsync(string functionName, string instanceId, object input);
    public virtual Task<TResult> CallSubOrchestratorAsync<TResult>(string functionName, object input);
    public abstract Task<TResult> CallSubOrchestratorAsync<TResult>(string functionName, string instanceId, object in
    public abstract Task<TResult> CallSubOrchestratorWithRetryAsync<TResult>(string functionName, RetryOptions retryO
    public virtual Task<TResult> CallSubOrchestratorWithRetryAsync<TResult>(string functionName, RetryOptions retryOp
    public virtual Task CallSubOrchestratorWithRetryAsync(string functionName, RetryOptions retryOptions, string inst
    public virtual Task CallSubOrchestratorWithRetryAsync(string functionName, RetryOptions retryOptions, object inpu
    public abstract void ContinueAsNew(object input);
    public abstract Task<T> CreateTimer<T>(DateTime fireAt, T state, CancellationToken cancelToken);
    public virtual Task CreateTimer(DateTime fireAt, CancellationToken cancelToken);
    public abstract T GetInput<T>();
    public abstract void SetCustomStatus(object customStatusObject);
    public abstract Task<T> WaitForExternalEvent<T>(string name);
    public abstract Task<T> WaitForExternalEvent<T>(string name, TimeSpan timeout);
    public abstract Task<T> WaitForExternalEvent<T>(string name, TimeSpan timeout, T defaultValue);
}
```

New Trigger Bindings

Function Role: **Activity Function**


```
[ActivityTrigger] DurableActivityContext
```

DurableActivityContext

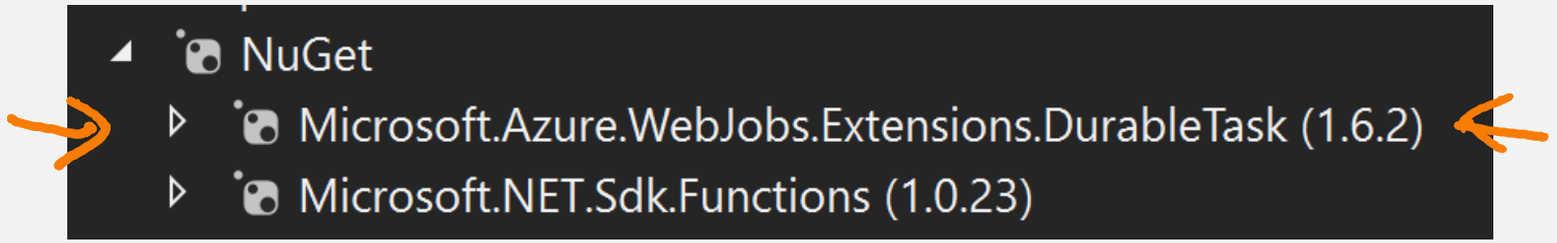
```
...public class DurableActivityContext
{
    ...public string InstanceId { get; }

    ...public T GetInput<T>();
}
```

Local Development

Local development with VS2017

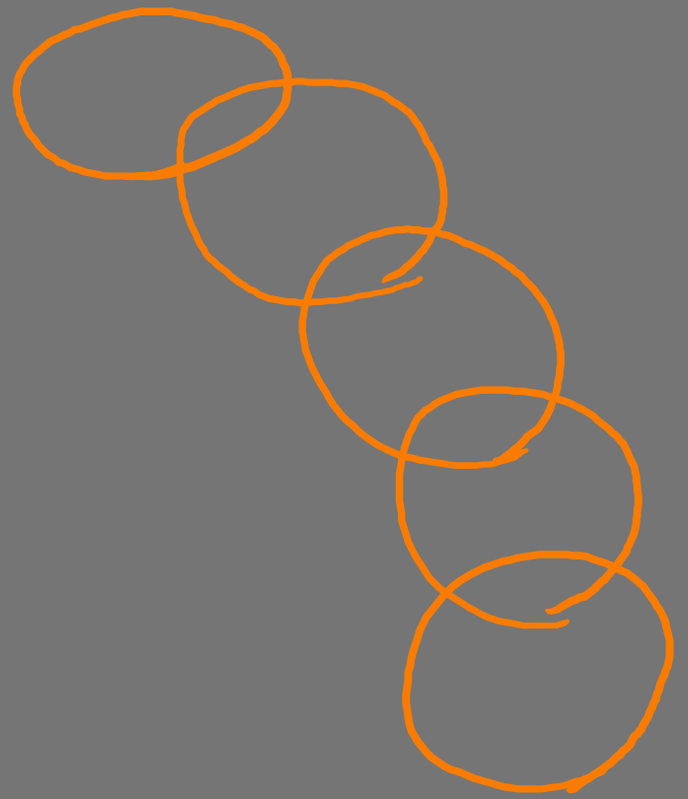
- Visual Studio 2017 15.7+
- Azure Workload
  - Extension: Azure Functions & Web Jobs Tools
  - Azure Storage Emulator
  - Microsoft.Azure.Webjobs.Extensions.DurableTask



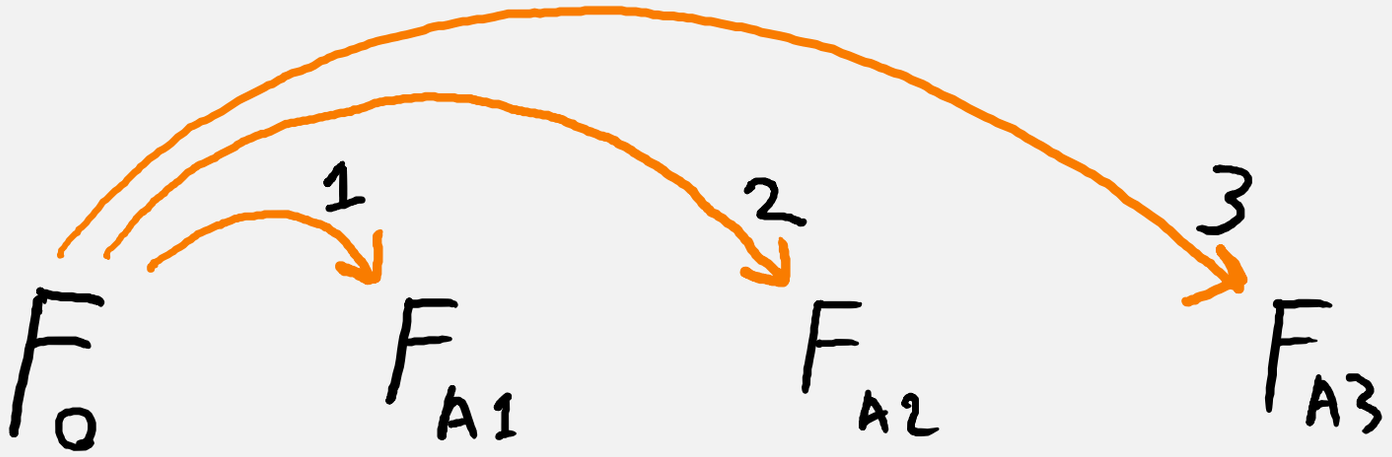
Hello {name} Demo!

HttpStart HelloName HelloNameActivity

Function Chaining



Function chaining

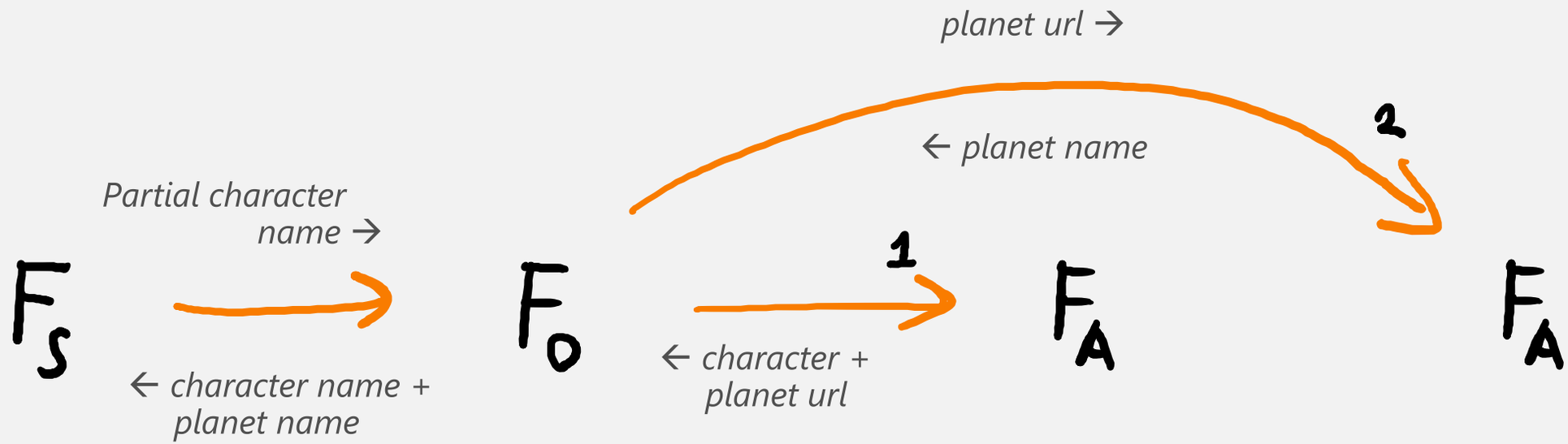


Demo: Function Chaining

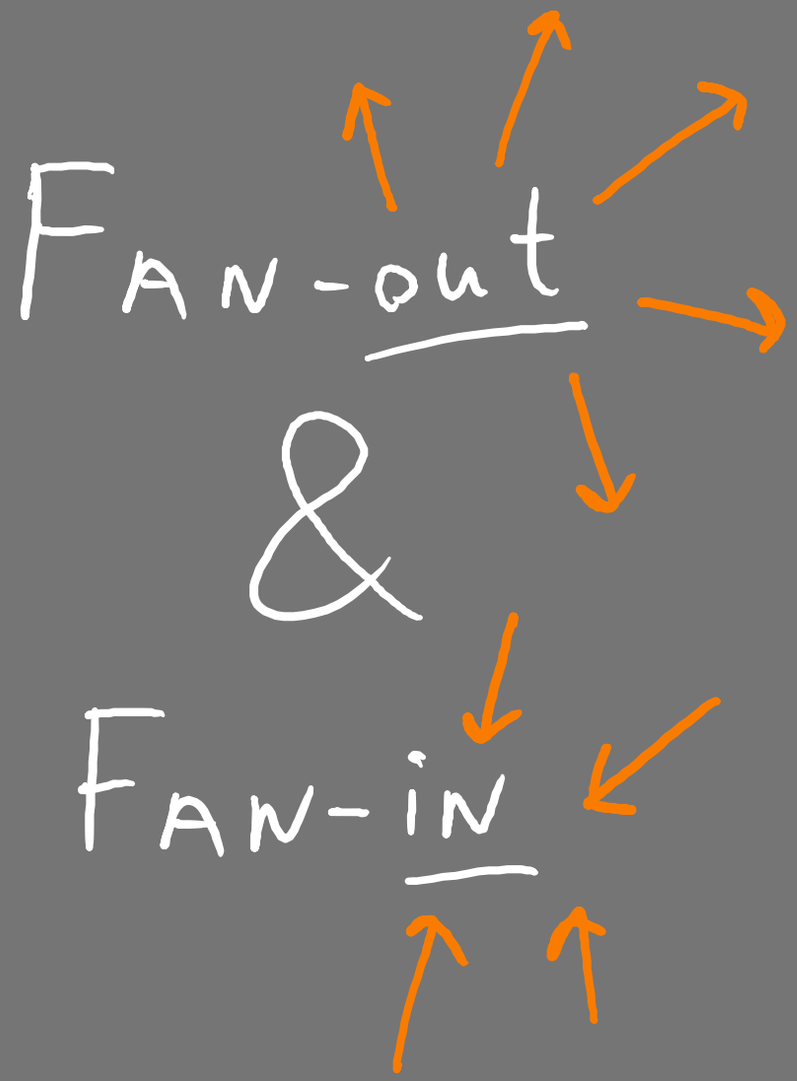
Find a Star Wars character and their home planet.

https://swapi.co/

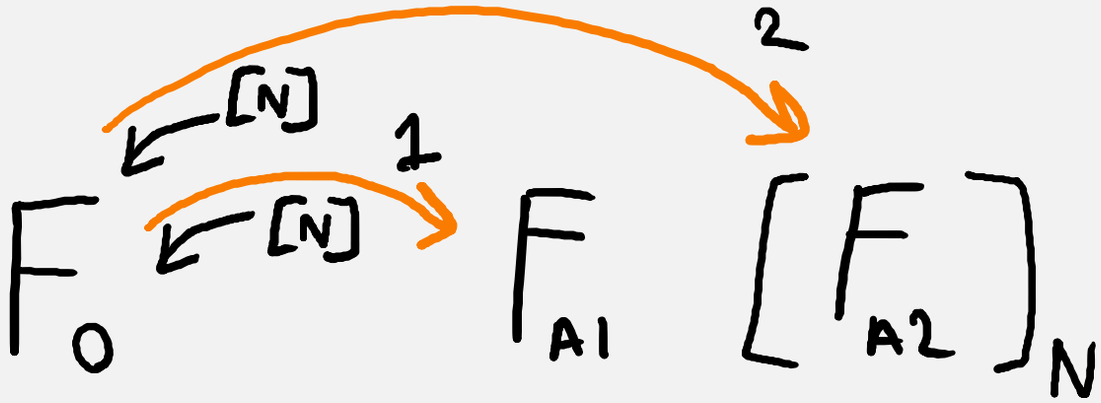
Demo: Function Chaining



HttpStart GetCharacterInfo SearchCharacter GetPlanet


Fan-out
&
Fan-in

Fan-out/fan-in



F_{A1} returns a collection of items

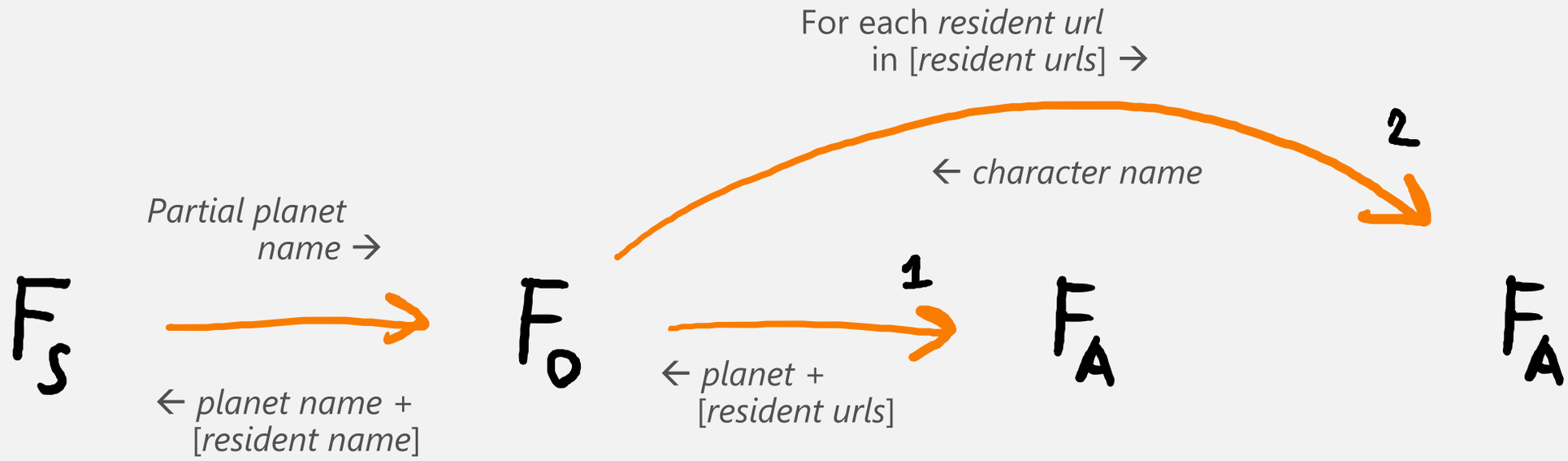
F_{A2} is called for each of the items from F_{A1}

Demo: Fan-out/Fan-in

Find a Star Wars planet and their residents.

https://swapi.co/

Demo: Fan-out/Fan-in



HttpStart GetPlanetResidents SearchPlanet GetCharacter

Unit Testing

Unit Testing Demo

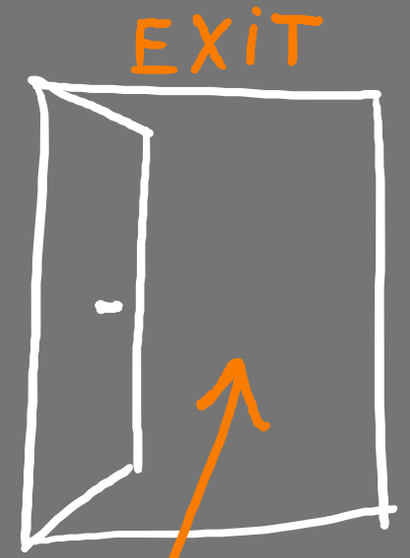
Testing the GetPlanetResidents orchestration using:

- xUnit
- Moq
- AutoFixture

Unit Testing

- Use DurableOrchestrationContextBase in the orchestration.
- Use serializable types with [ActivityTrigger] for the activity input (because DurableActivityContext can't be instantiated/mocked due to internal constructor ☹).

Closing Remarks



Closing remarks I

- In and output of functions should be serializable.
- Orchestration Functions should be deterministic:
  - `DateTime.Now`, use `CurrentUtcDateTime`
  - `Guid.NewGuid()`
  - Random generated data

Closing remarks II

- Orchestration Functions can only call Activity Functions in the same Function App.
- Keep your orchestrations small.
- What changes together should be deployed together.

GitHub: demos-azure-durable-functions

mduiker@xpirit.com

blog.marcduiker.nl

@marcduiker