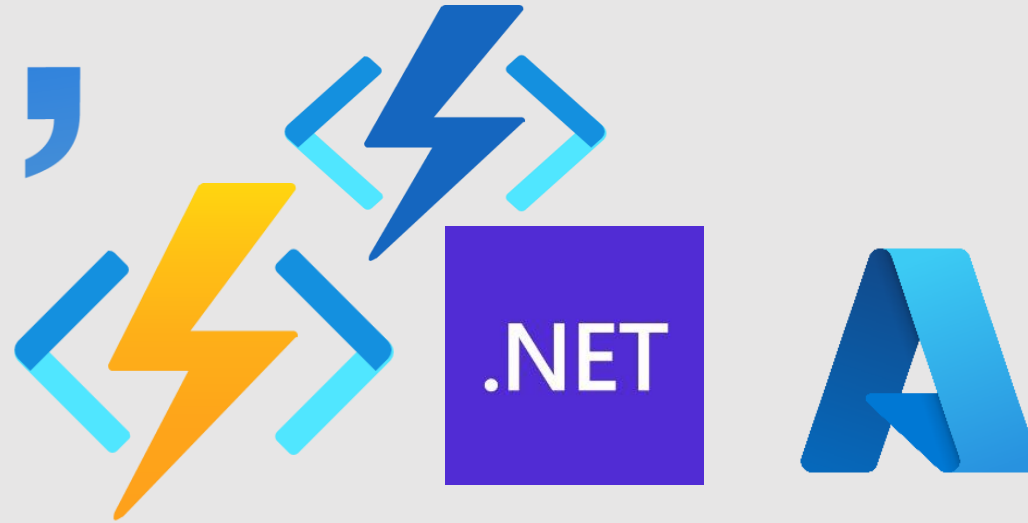




Jonah Andersson

Azure Durable Functions for Developers:

Developing and Solving Complex
Event-Driven Workflows



LET US CONNECT!

goto;
copenhagen





Sarah

Thomas

JONAH

Get on board.

To Stockholm



5 min



20 min



50 min

Arlanda express

» Fastest way from Arlanda to Stockholm C.

18 minutes non-stop
to Stockholm C,
that's express.



Arlanda
express



VANDERLANDE

Convenient
Speed
Delivery
Using AI

+ Cute Dog!



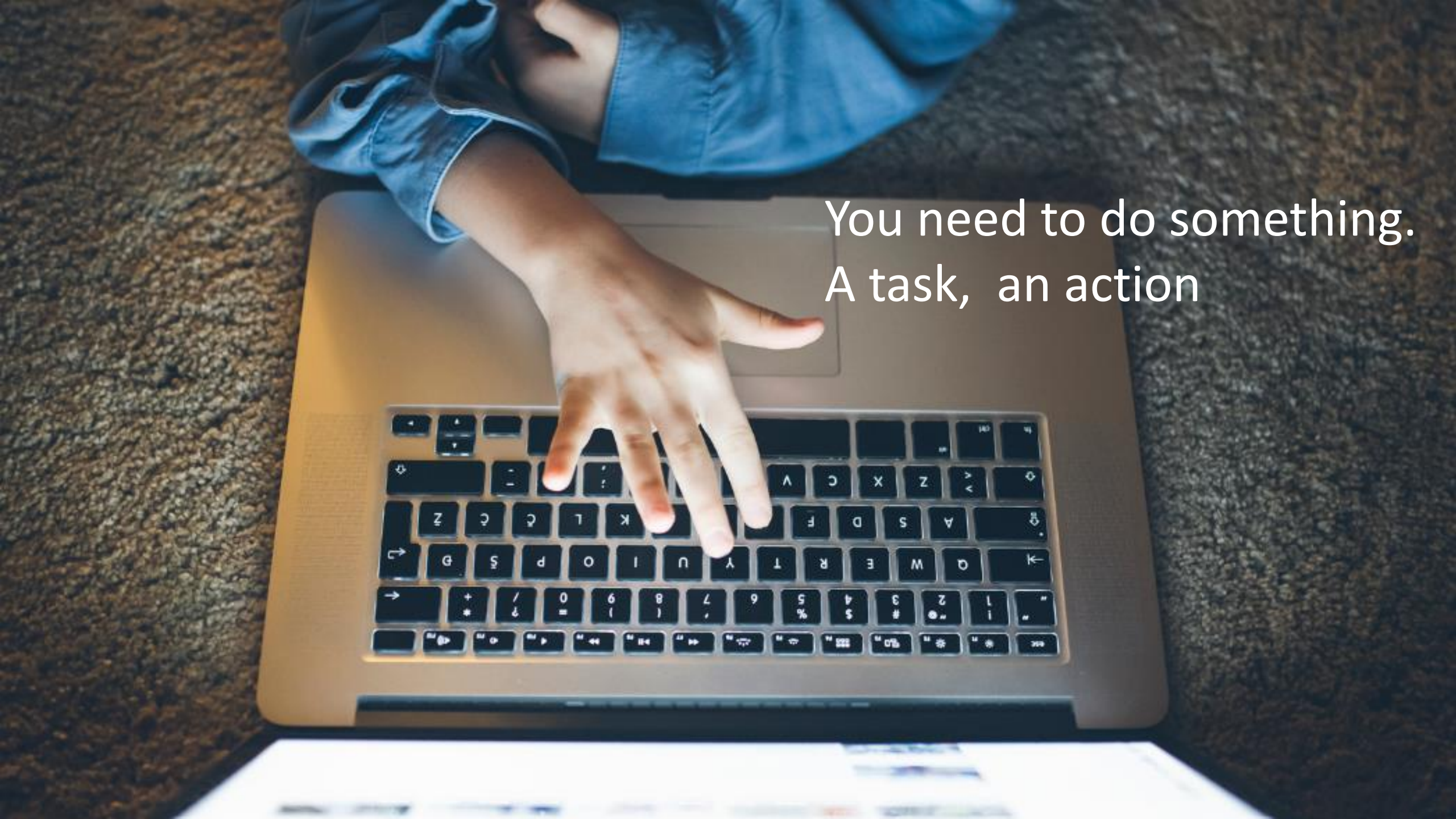
Imagine ...



You have an idea...

You want to jot it down...

You click the mouse...

A top-down photograph of a person's hand hovering just above the keyboard of a silver laptop. The person is wearing a blue long-sleeved shirt. The laptop is open, and the keyboard is clearly visible. The background is a dark, textured surface. The text "You need to do something. A task, an action" is overlaid in white on the right side of the image.

You need to do something.
A task, an action



... and you have to wait



Relax





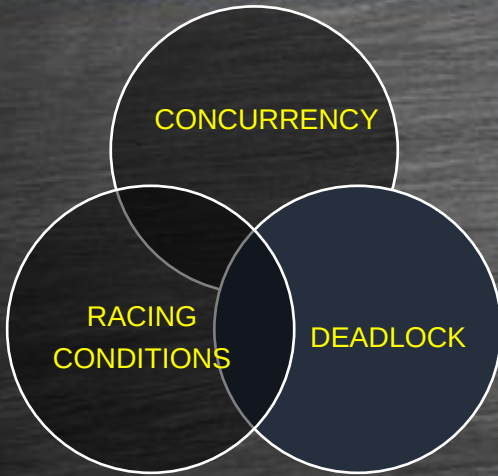
Or...
Kill your
computer!

Serverless computing 
*can help solve
complex
event-driven workflows.* 

PROBLEM



SOLUTION



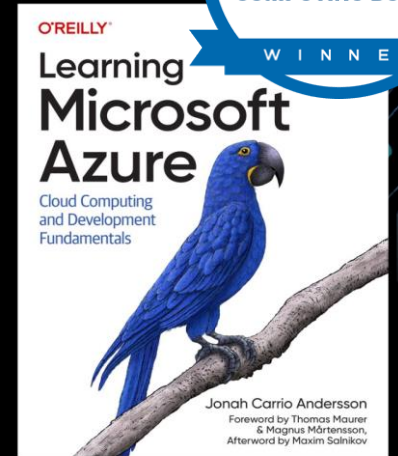


Jonah Andersson

@cjkodare

“Inspire tech to others”

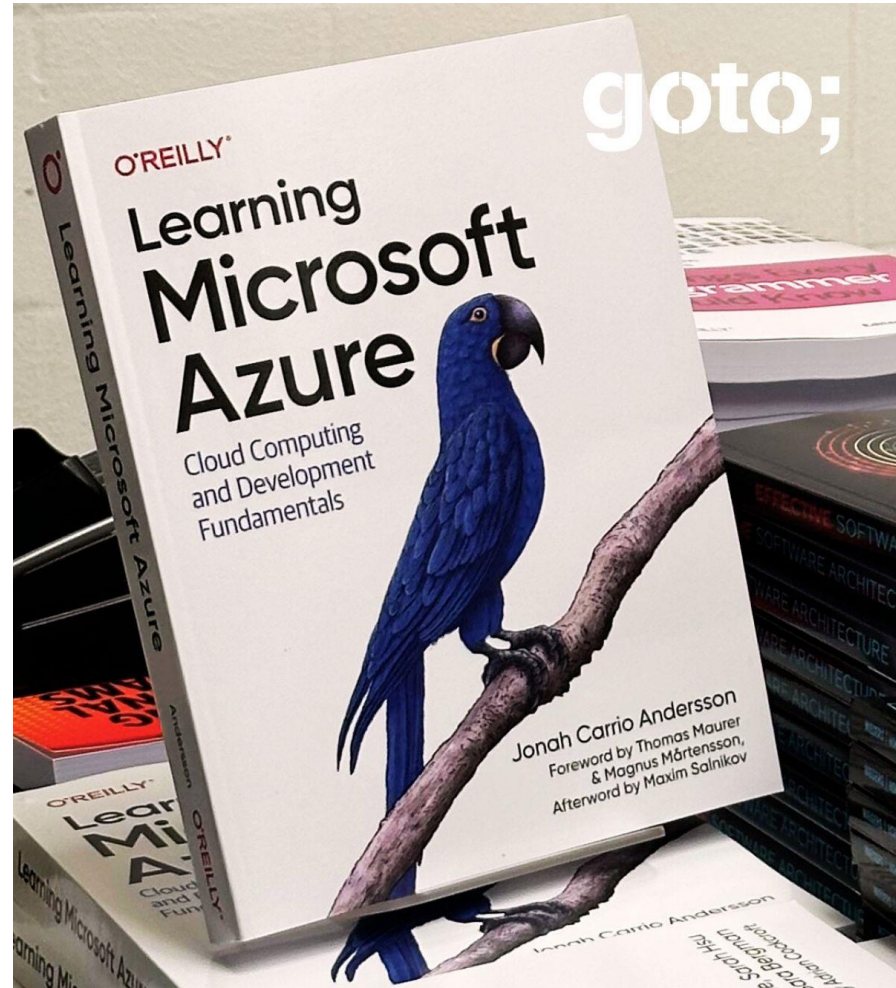
- Senior Azure Consultant
- Developer (Fullstack, .NET, Cloud)
- Lead Cloud DevOps Engineer Architect
- Author of Learning Microsoft Azure (O'Reilly)
- Tech Community Leader
- Microsoft MVP, MCT, MS Learn Expert
- Developer of the Year 2023, Årets Utvecklare 2024



Signed Books Raffle



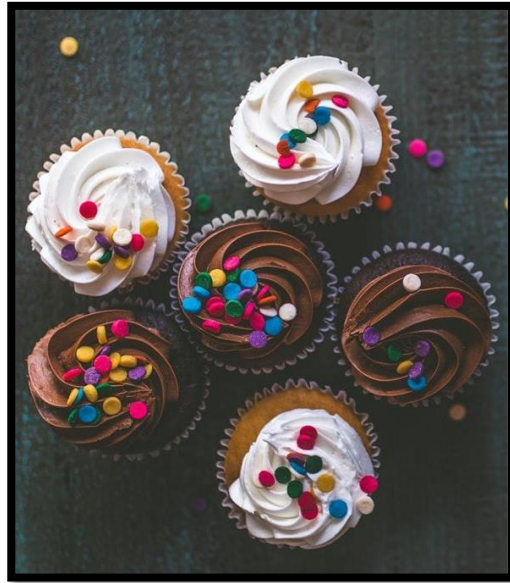
Drop By Factum Books Booth and say Hi to John



Serverless Sweet Swedish Fika Comparison



MONOLITH



MICROSERVICES



CONTAINERIZATION

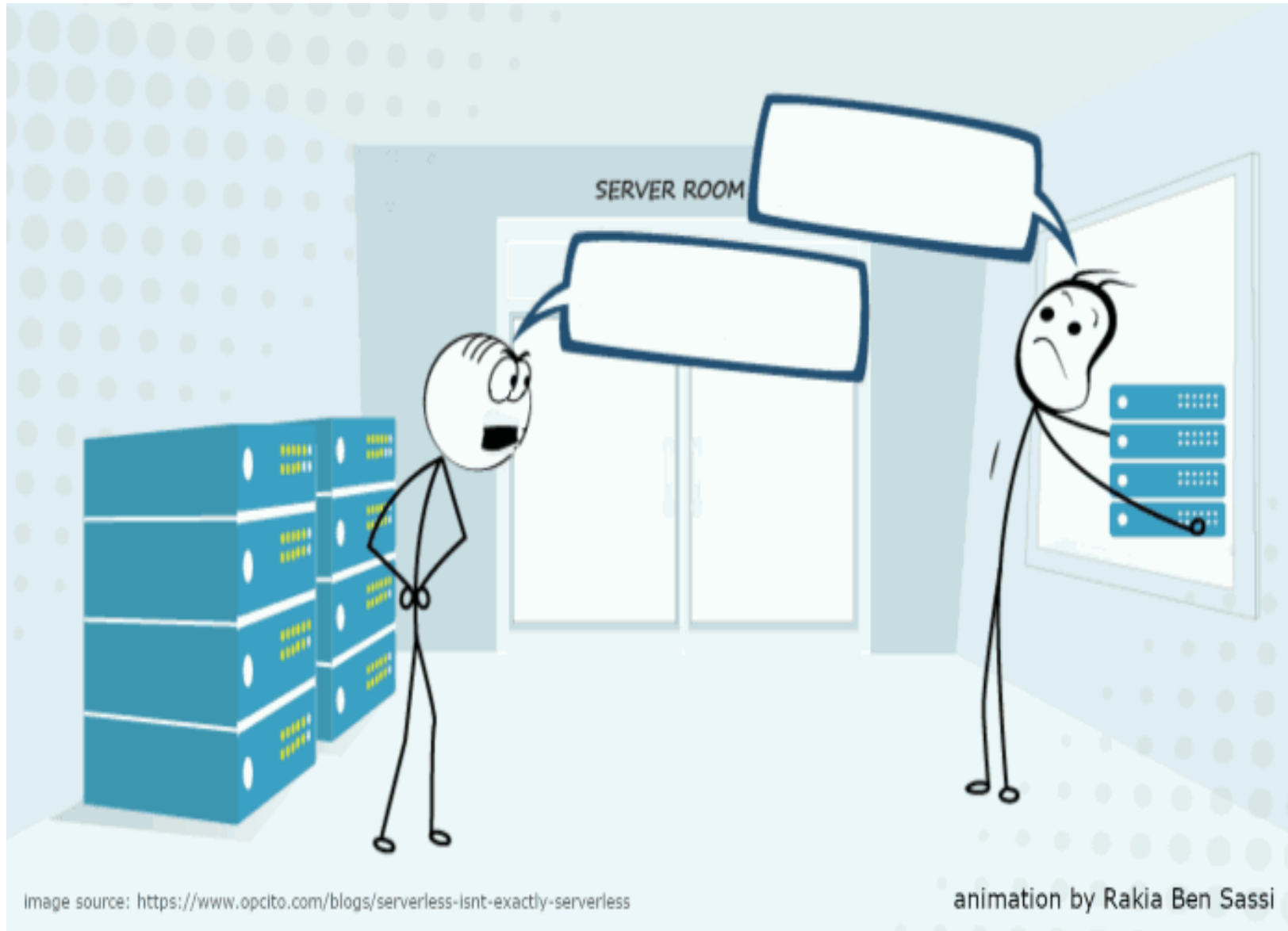


SERVERLESS

CONTROL

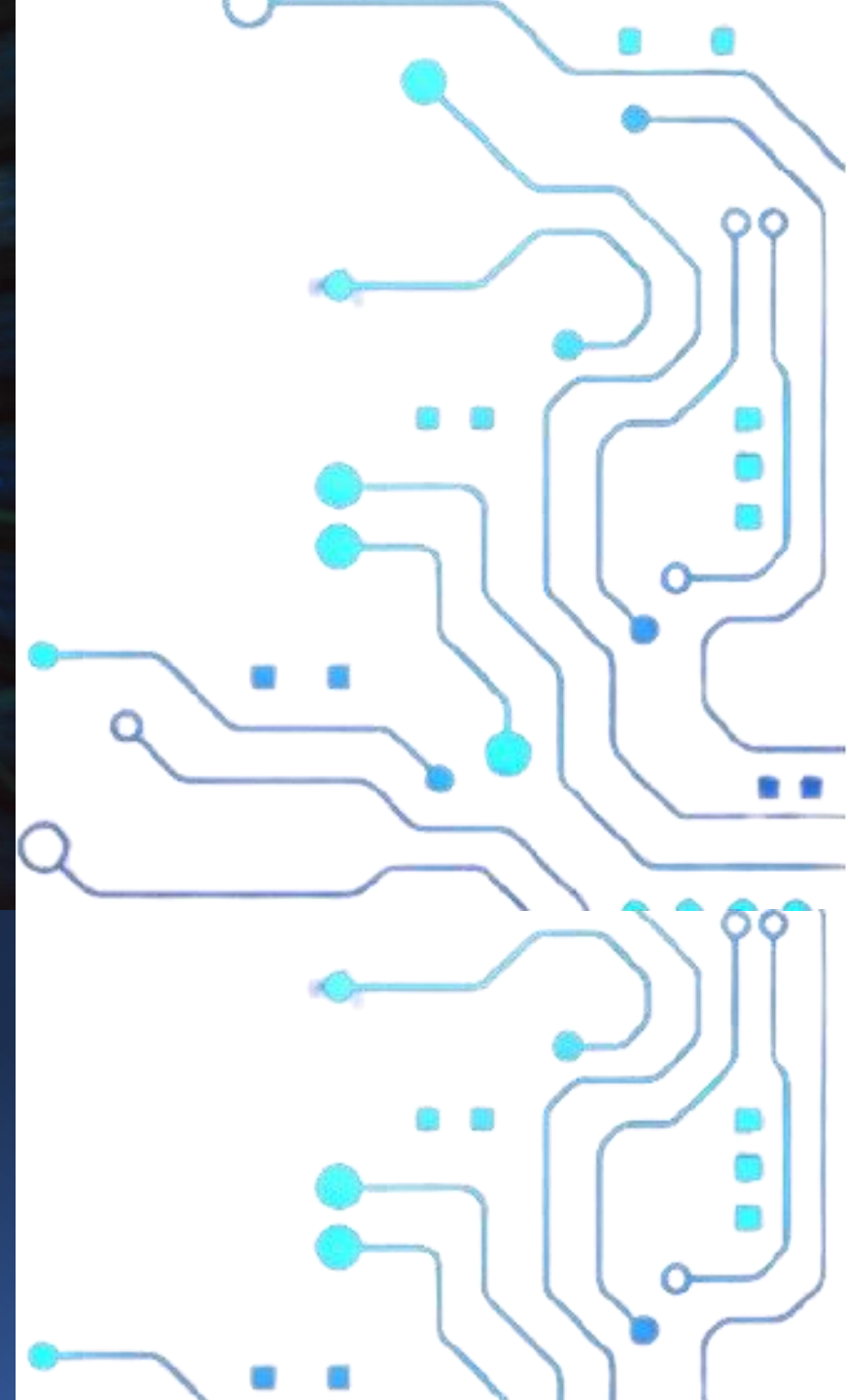
PRODUCTIVITY

Making Sense of Serverless





Backend Infrastructure and Servers
are managed by the Cloud Provider



```
delay_sec(1);  
idle = 0; /* wait a second before processing data */  
digits = 0;  
data_read = 0;  
while (1) {  
    data = RdPortA();  
    if (data & VALID) { /* if valid line set */  
        idle = 0;  
        if (data_read == 0) { /* if digit not processed yet */  
            data &= 0xf;  
            if (data == 12) { /* if B key */  
                if (digits == 4 && /* check code */  
                    digit1 == CODE1 &&  
                    digit2 == CODE2 &&  
                    digit3 == CODE3 &&  
                    digit4 == CODE4) { /* set B(4) if correct code */  
                    delay_sec(2);  
                    /* reset */  
                    digits = 0;  
                    data_read = 0;  
                } else { /* save the digit */  
                    digits++;  
                    data_read = 1;  
                }  
            }  
        }  
    }  
}
```

Developer Productivity

Azure Durable Functions

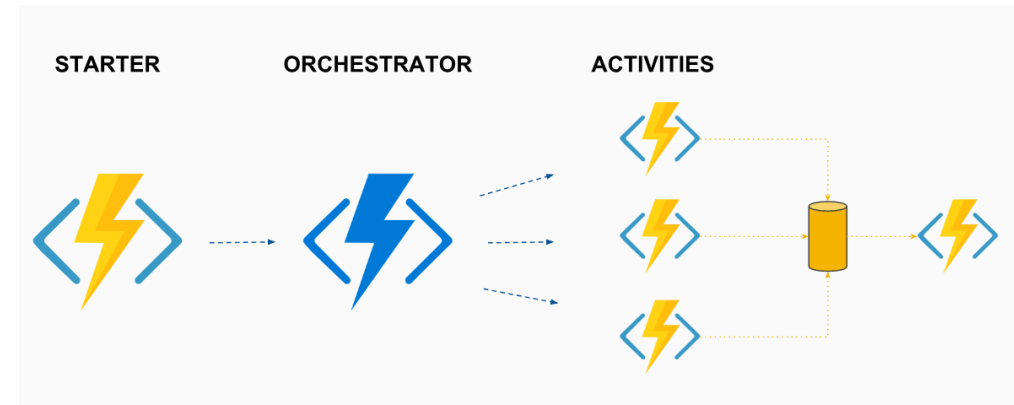
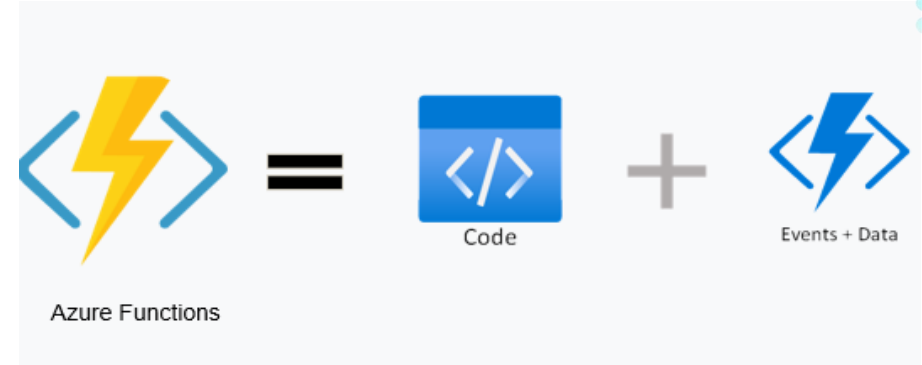
Extension of Azure Functions

Stateful Workflows in Serverless Architecture

Manages, Checkpoints and Restart – State Management

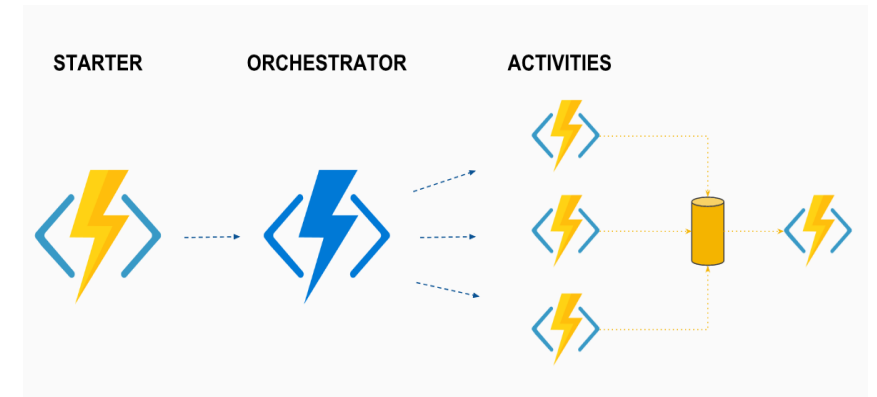
C# .NET, Javascript, Python, F#, Powershell, Java, Go/Rust

Durable Task Framework <https://github.com/Azure/durabletask>

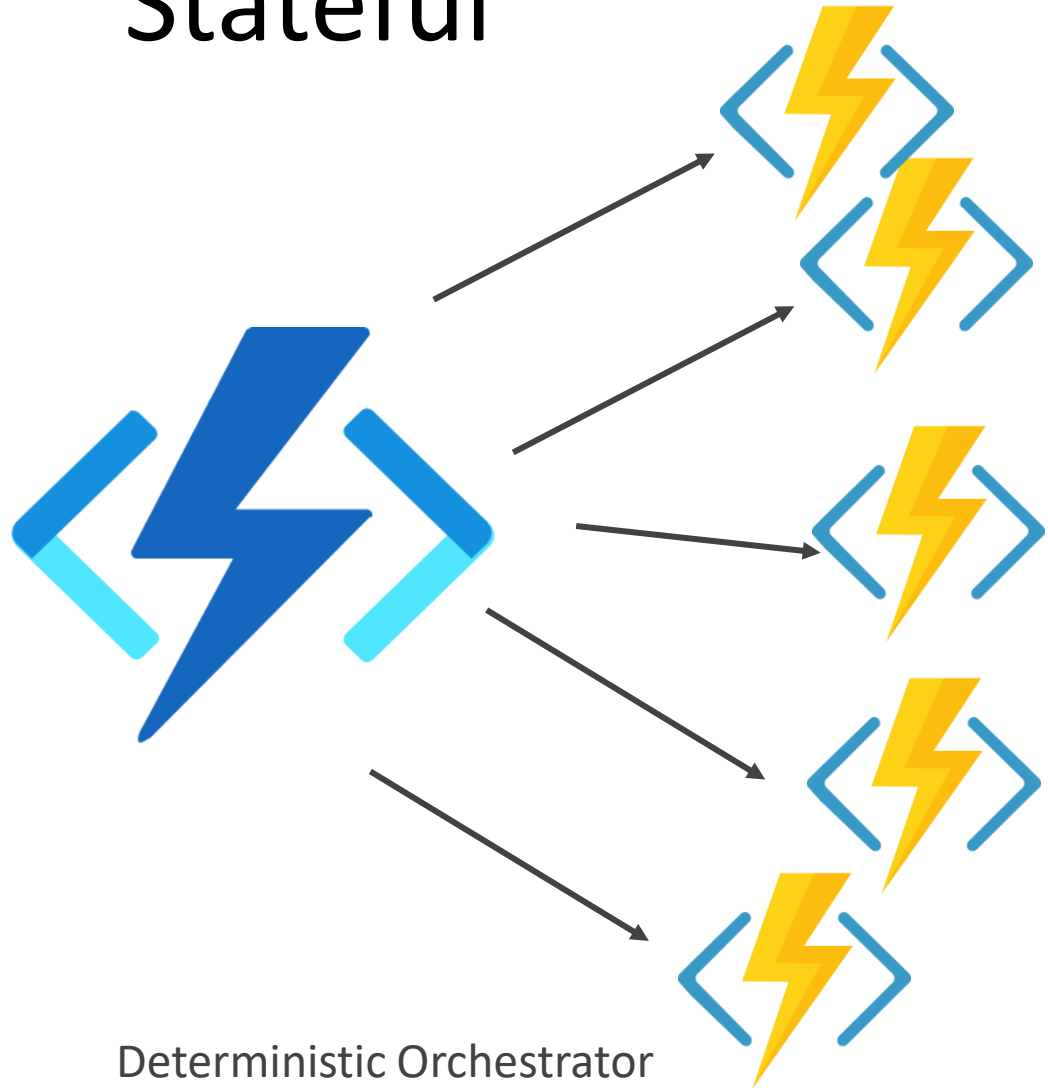


Durable Functions - .NET Isolated Process

- Separate processes, fewer conflicts
- Full control of the process (start-up, configuration, middleware, etc.)
- Standard dependency injection
- .NET version flexibility (.NET runtime and framework)



Stateful

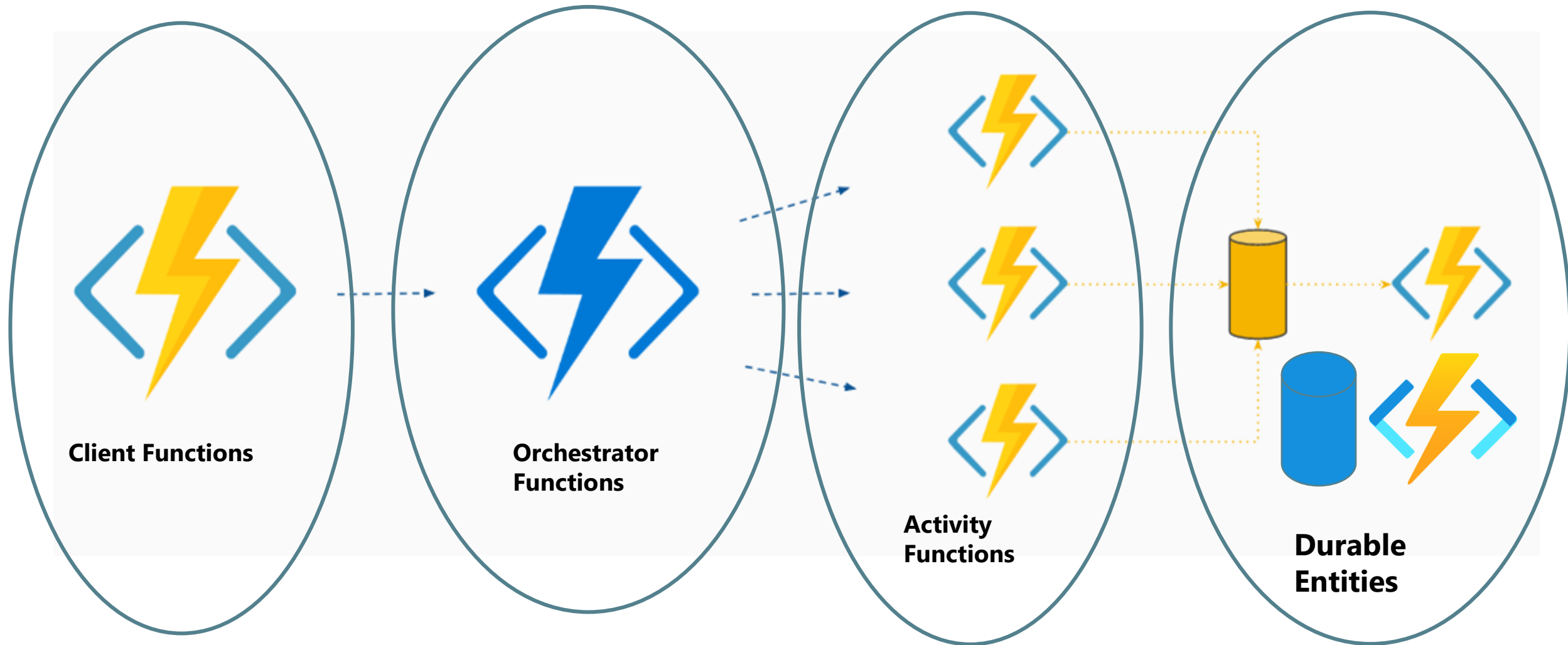


vs.

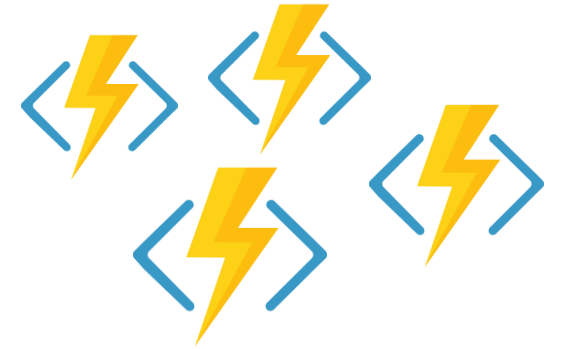
Stateless



Durable Functions - Types



Orchestrator = Tinikling Dance Orchestration



Tinikling is a traditional Philippine folk dance that originated during the Spanish colonial era.



Orchestrator is our helpful yet strict friend!



Durable Functions –Donts & Dos!

Dont generate random numbers
or GUIDs

NewGuid() method

Dont access data stores like
databases and configurations

Pass configurations into
Activity function

Blocking APIs - Threads
like Sleep, Run or Delay

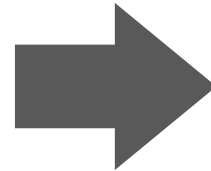
Use DurableOrchestrationContext

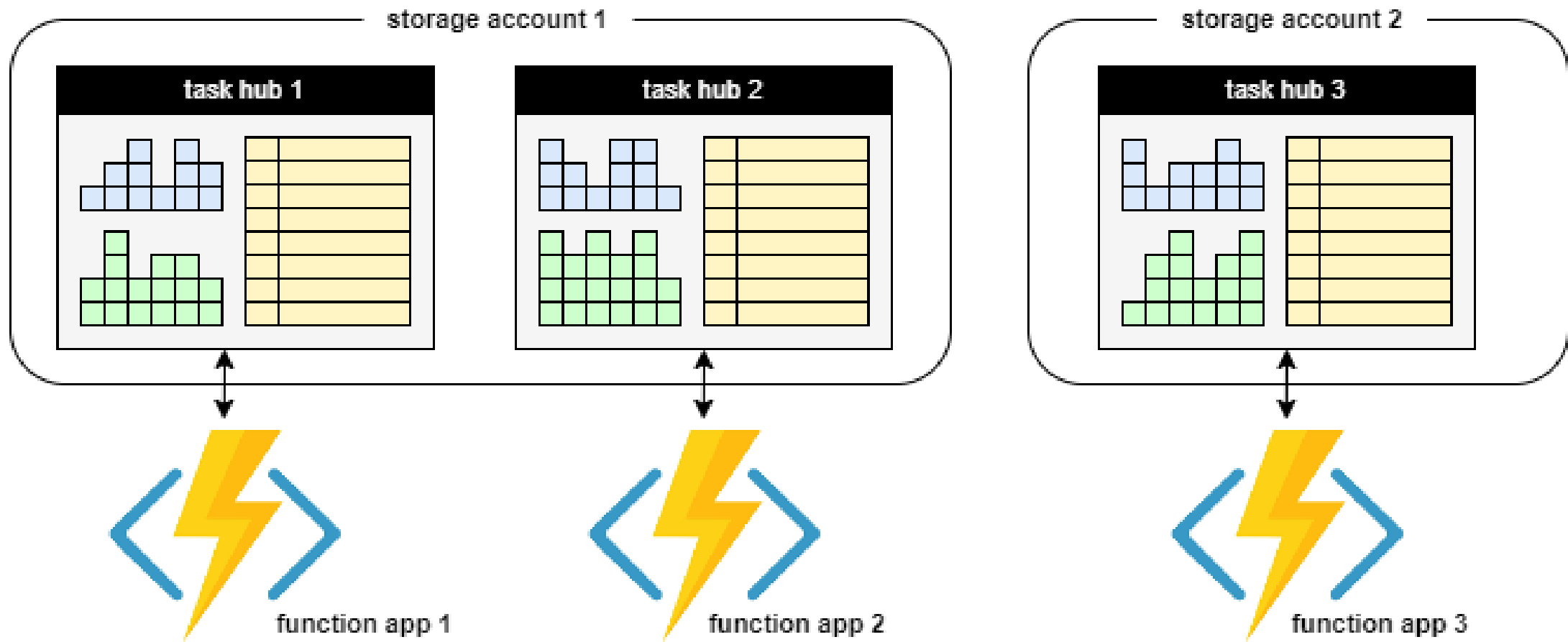
Dont create Infinite Loops!

CreateAsNew()

Dont use CurrentDateTime

Use.CurrentUtcDateTime()

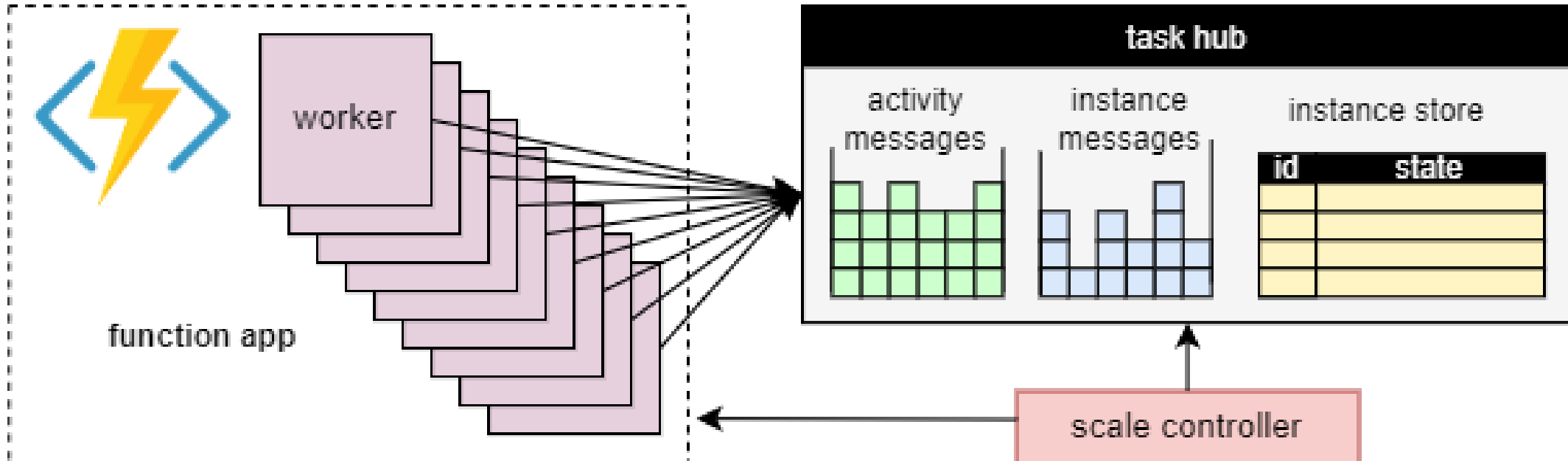




Task Hub – a logical container for durable storage resources used for **orchestrations** and **entities**.

Worker scaling

- Apps can add more workers (**scale out**) if the work needs to be processed more quickly
- It can remove workers (**scale in**) if there is not enough work to keep the workers busy.
- Possible to *scale to zero* if the task hub is completely idle



Concurrency in Azure Functions



Static

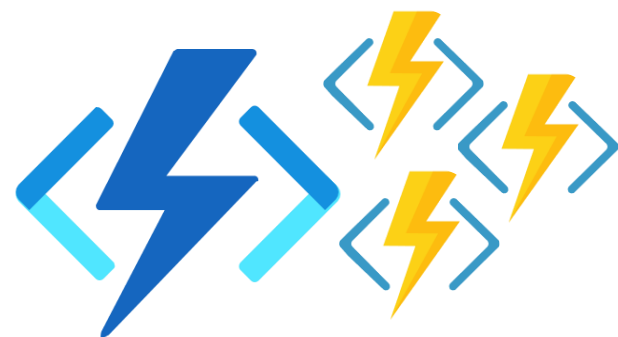
You can configure **host-level limits** on concurrency, which are specific to individual triggers. This is the default concurrency behavior for all Azure Functions.



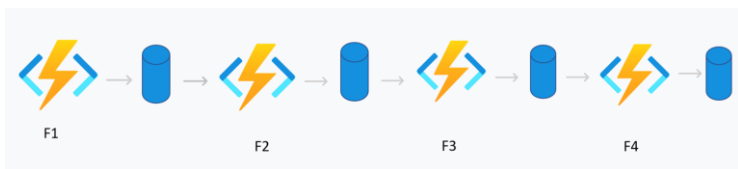
Dynamic

For certain trigger types, the Functions host can **automatically determine** the best level of concurrency for that trigger in your app

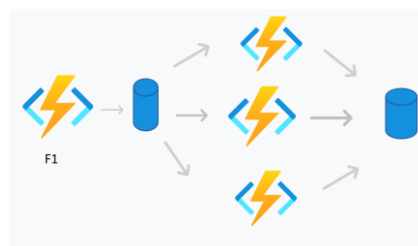




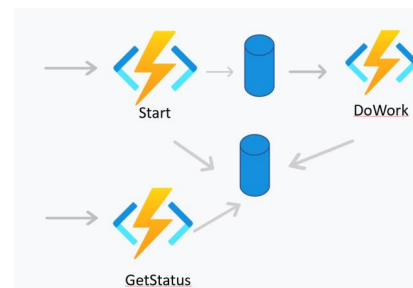
Application Design Patterns of Azure Durable Functions



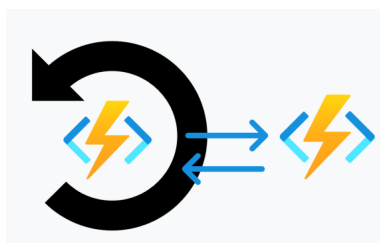
Function Chaining



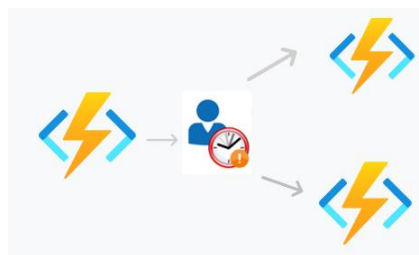
Fan out / Fan in



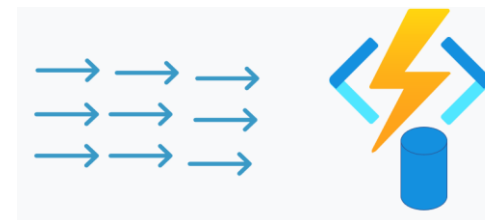
Async HTTP API



Monitor

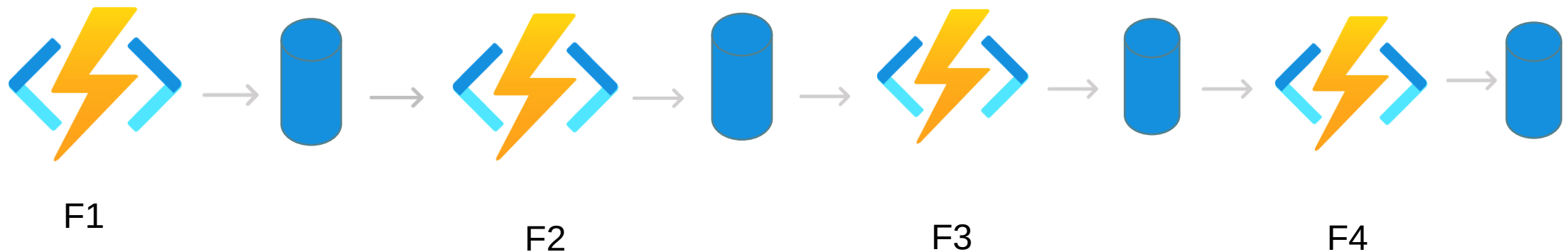


Human Interaction



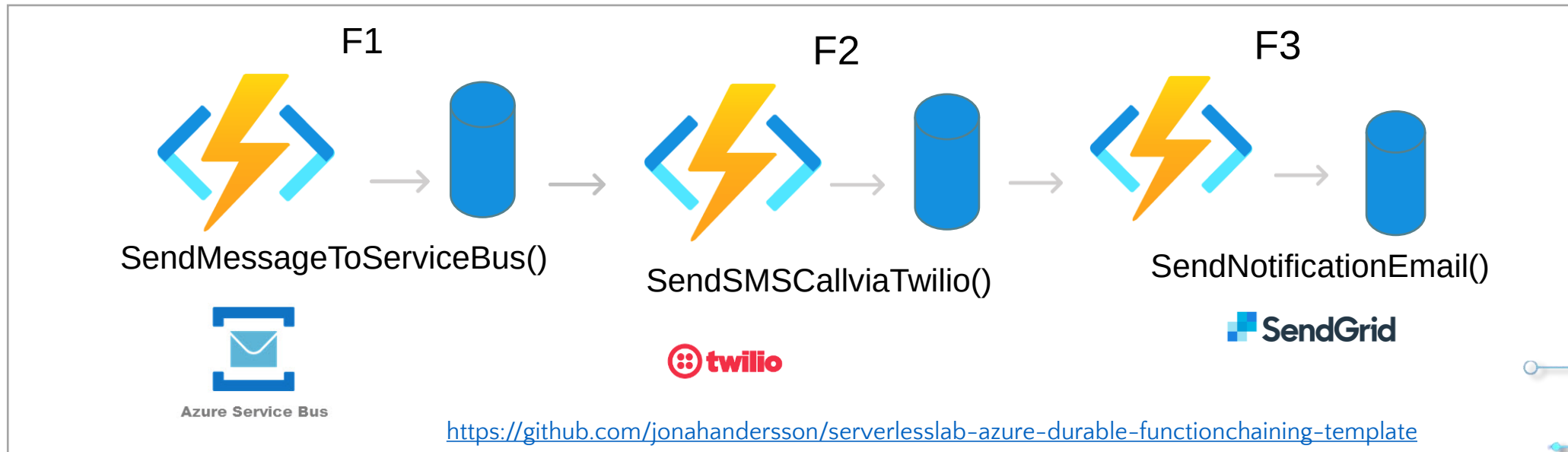
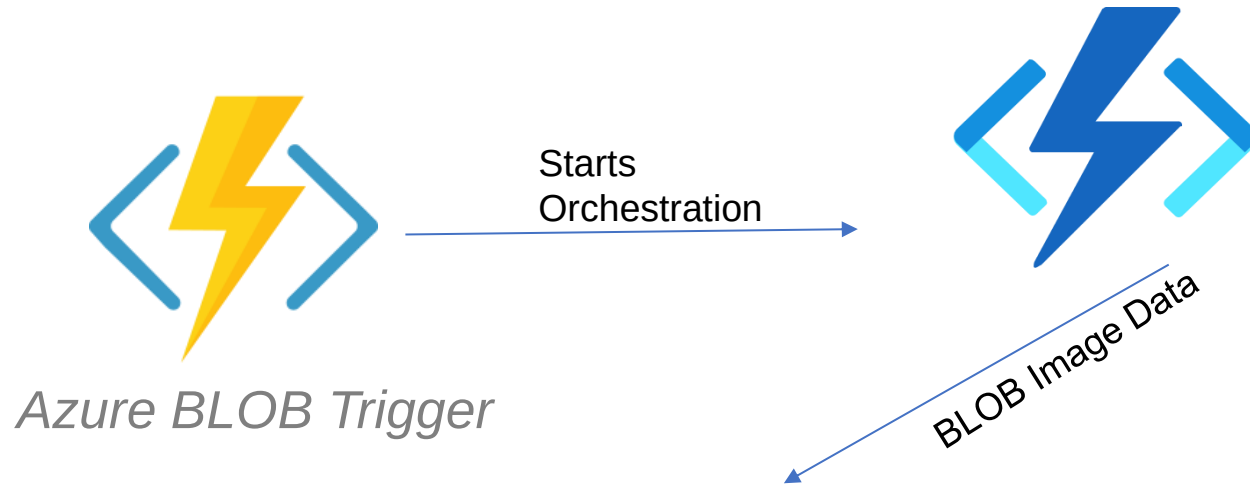
Aggregator

Function Chaining

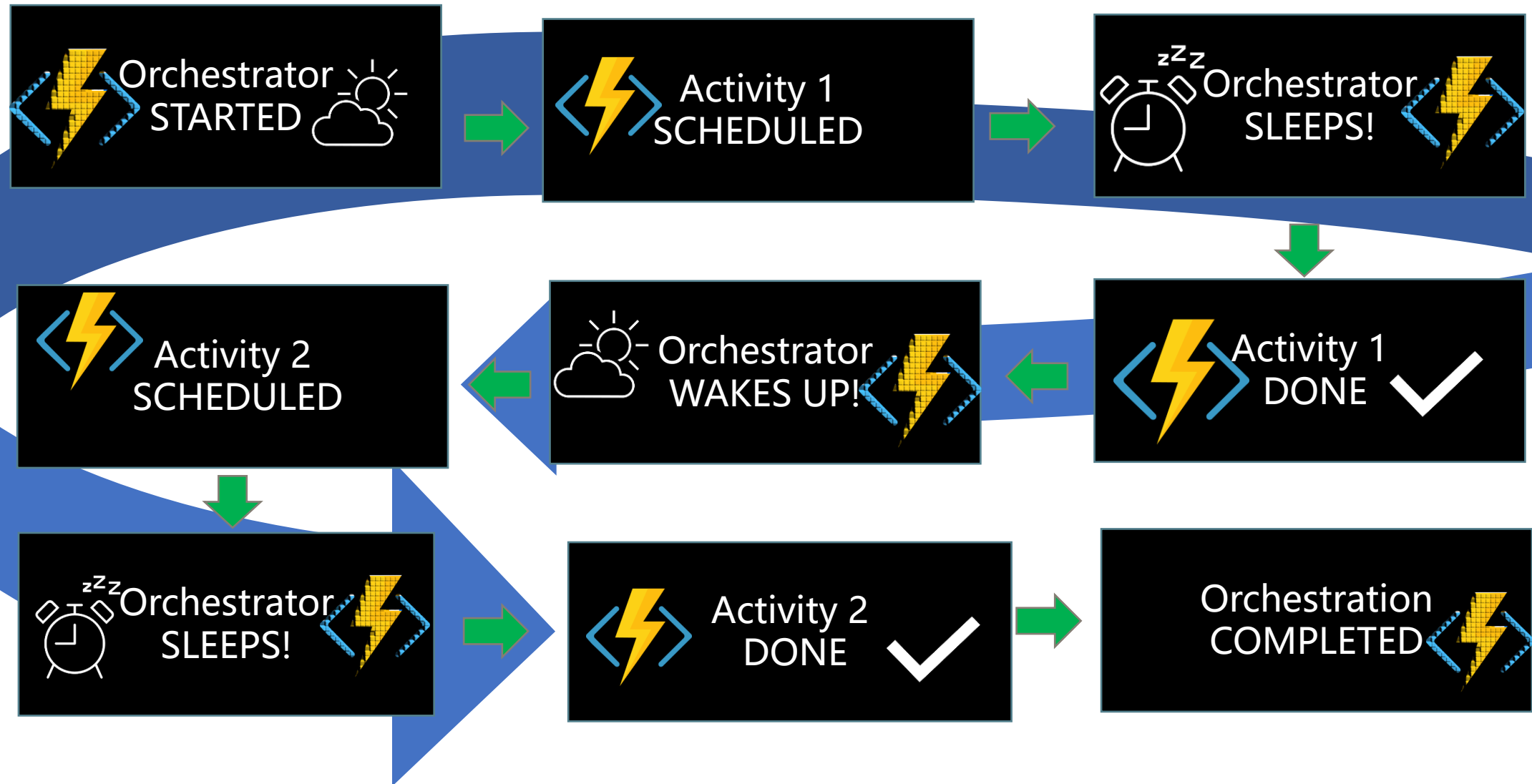


Execute functions in a particular order and sequence.

Example Function Chaining



Function Chaining in Action

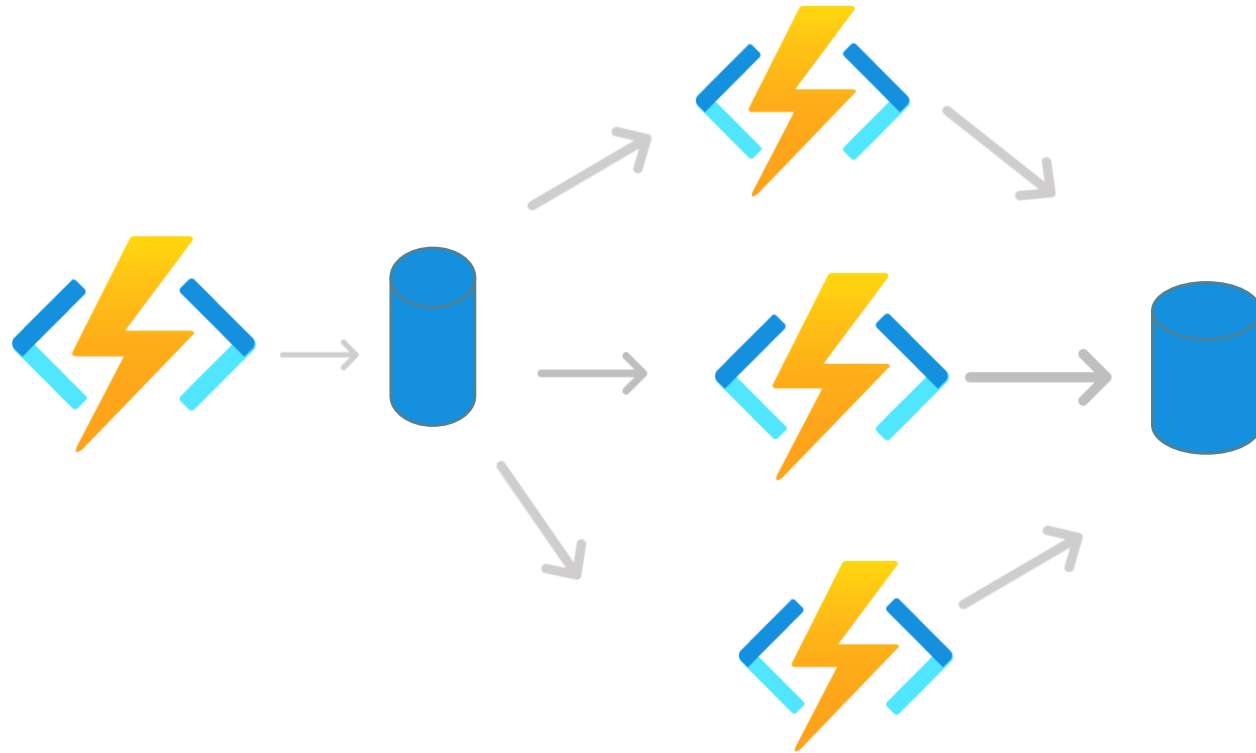


The 
Orchestrator
can be a
sleepyhead
but it is **GOOD!**





Fan-Out/Fan In Pattern



Execute multiple functions in parallel and then wait for all functions to finish and aggregate the results when done.

C#

JavaScript

Python

PowerShell

Java

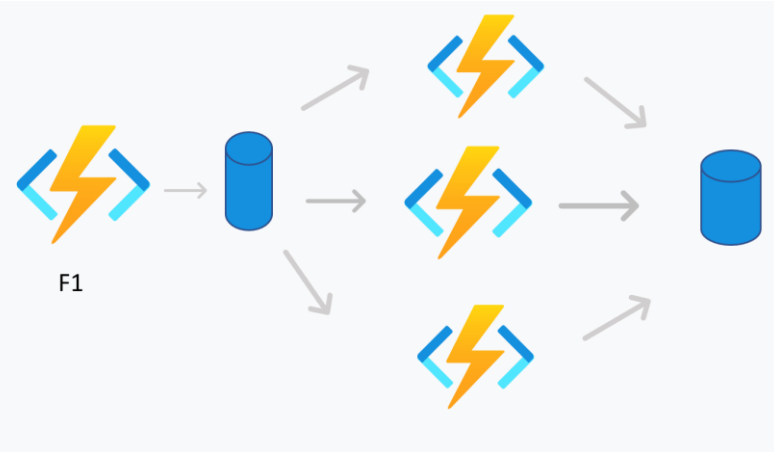
C#

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

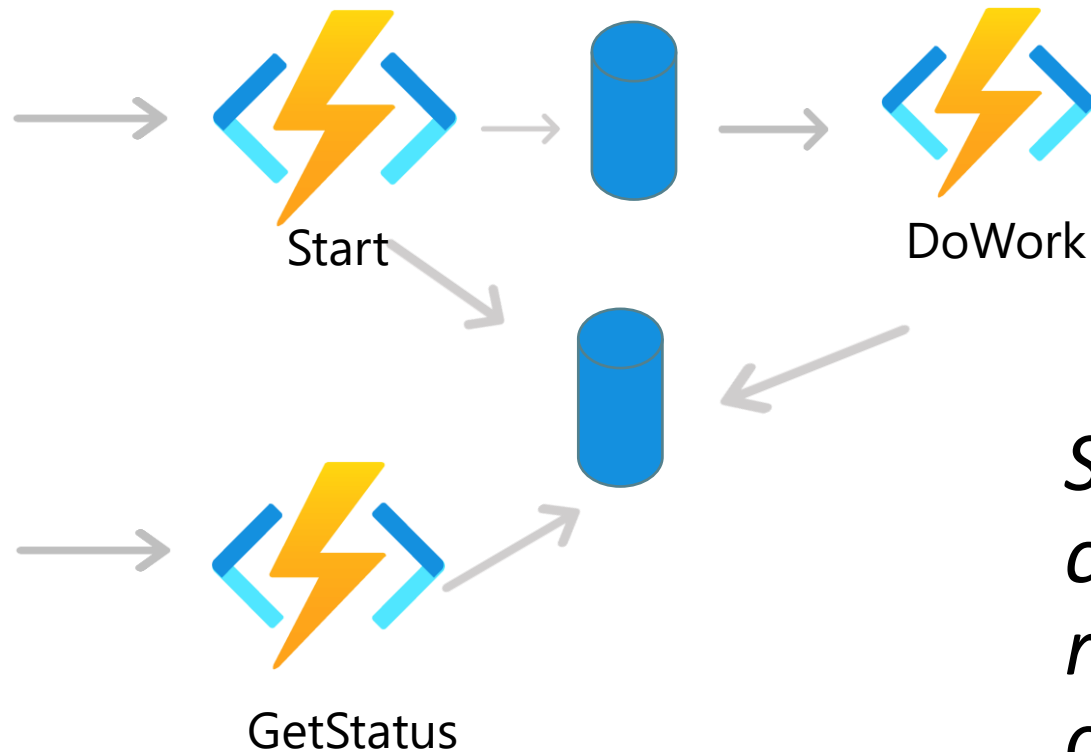
    // Get a list of N work items to process in parallel.
    object[] workBatch = await context.CallActivityAsync<object[]>("F1", null);
    for (int i = 0; i < workBatch.Length; i++)
    {
        Task<int> task = context.CallActivityAsync<int>("F2", workBatch[i]);
        parallelTasks.Add(task);
   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

    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);
}
```

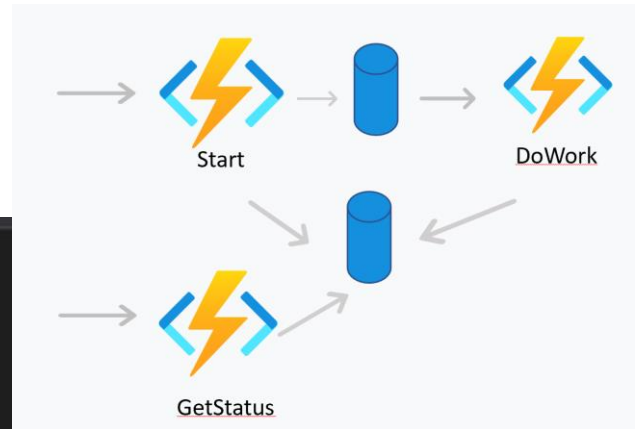


HTTP Async APIs



Solves the problem of coordinating the state of long-running operations with external clients

HTTP Async APIs - Example

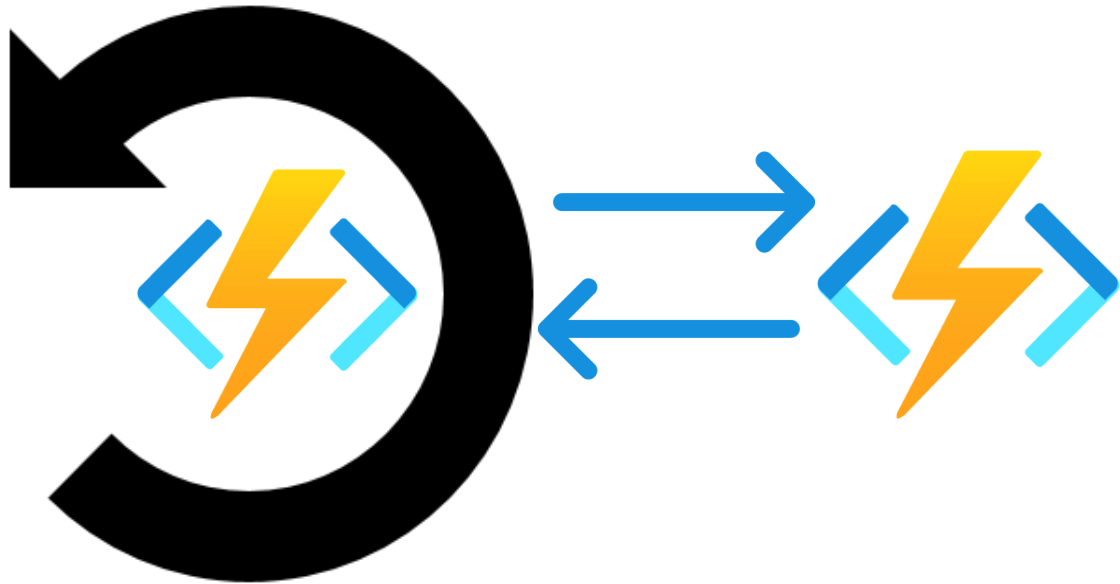


```
namespace HttpAsyncAPIExample
{
    0 references | 0 changes | 0 authors, 0 changes
    public static class CheckIfMySiteIsAvailable
    {
        [FunctionName("CheckWebsiteIsAvailable")]
        0 references | 0 changes | 0 authors, 0 changes
        public static async Task CheckWebsiteIsAvailable([OrchestrationTrigger] IDurableOrchestrationContext context, ILogger log)
        {
            try
            {
                Uri jonahsUrl = new Uri("https://jonahandersson.tech");

                // Makes an HTTP GET request to the specified endpoint like a url
                DurableHttpResponse httpResponse = await context.CallHttpAsync(HttpMethod.Get, jonahsUrl);

                if ((int)httpResponse.StatusCode >= 500 && (int)httpResponse.StatusCode < 600)
                {
                    throw new HttpRequestException("Hey, there is a server error! Something wrong with your website. Fix it Jonah!");
                }
                else
                {
                    log.LogInformation($"Completed: {httpResponse.StatusCode} {httpResponse.Content}");
                }
            }
            catch (Exception ex)
            {
                log.LogError($"Error at CheckIfMySiteIsAvailable(): { ex.Message} { ex.InnerException}");
                throw;
            }
        }
    }
}
```

Monitor Pattern



- *Recurring process or stateful workflow that is flexible*
- *Polling until specific conditions are met*

```

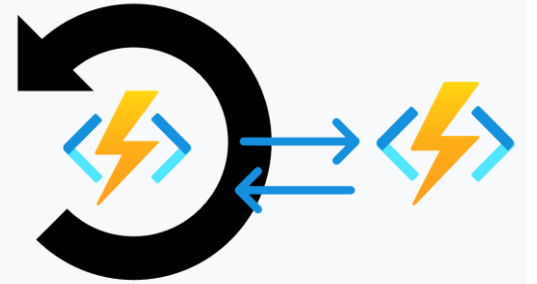
[FunctionName("MonitorJobStatus")]
public static async Task Run(
    [OrchestrationTrigger] IDurableOrchestrationContext context)
{
    int jobId = context.GetInput<int>();
    int pollingInterval = GetPollingInterval();
    DateTime expiryTime = GetExpiryTime();

    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;
        }

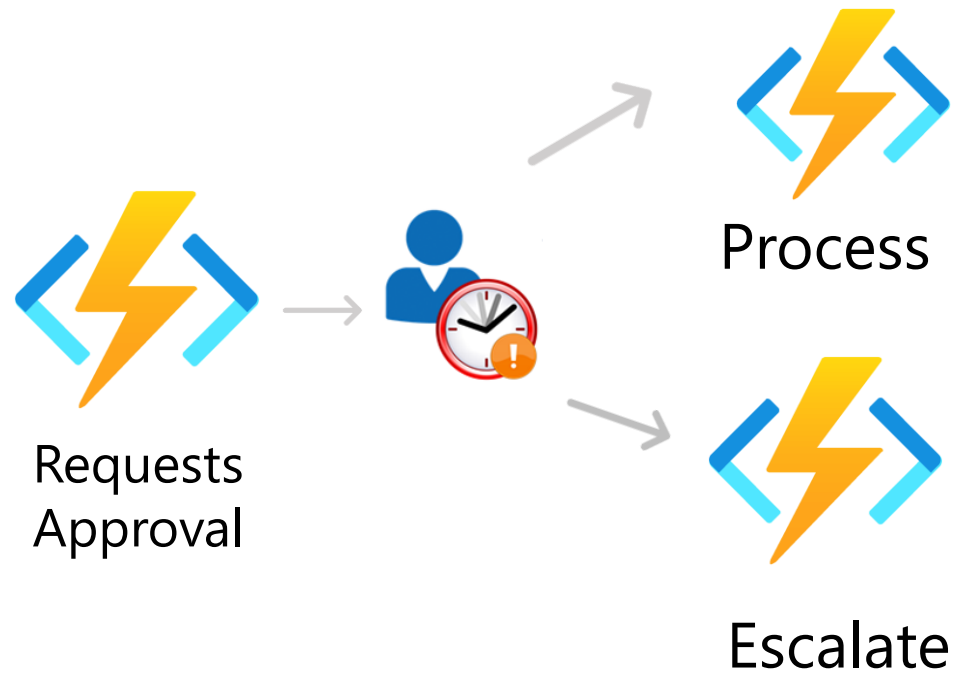
        // Orchestration sleeps until this time.
        var nextCheck = context.CurrentUtcDateTime.AddSeconds(pollingInterval);
        await context.CreateTimer(nextCheck, CancellationTokens.None);
    }

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

```



Human-Interaction Pattern



Automate business process that require human interaction

C#

JavaScript

Python

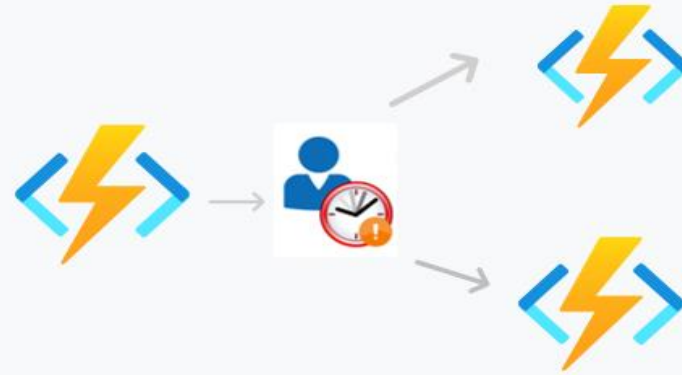
PowerShell

Java

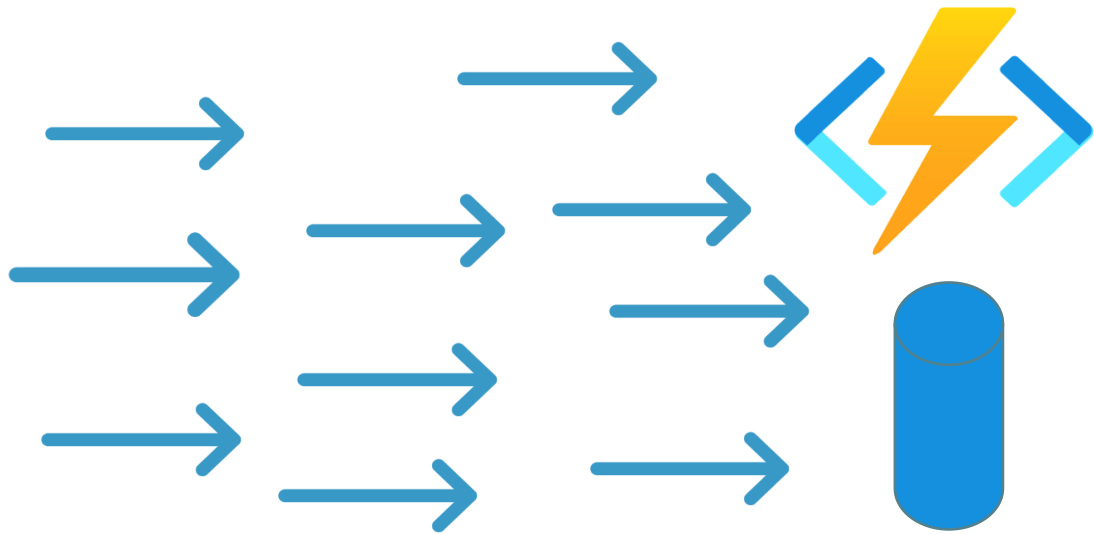
C#

```
[FunctionName("ApprovalWorkflow")]
public static async Task Run(
    [OrchestrationTrigger] IDurableOrchestrationContext context)
{
    await context.CallActivityAsync("RequestApproval", null);
    using (var timeoutCts = new CancellationTokenSource())
    {
        DateTime dueTime = context.CurrentUtcDateTime.AddHours(72);
        Task durableTimeout = context.CreateTimer(dueTime, timeoutCts.Token);

        Task<bool> approvalEvent = context.WaitForExternalEvent<bool>("ApprovalEvent");
        if (approvalEvent == await Task.WhenAny(approvalEvent, durableTimeout))
        {
            timeoutCts.Cancel();
            await context.CallActivityAsync("ProcessApproval", approvalEvent.Result);
        }
        else
        {
            await context.CallActivityAsync("Escalate", null);
        }
    }
}
```



Aggregator Pattern (Stateful Entities)



C#

```
[FunctionName("EventHubTriggerCSharp")]
public static async Task Run(
    [EventHubTrigger("device-sensor-events")] EventData eventData,
    [DurableClient] IDurableEntityClient entityClient)
{
    var metricType = (string)eventData.Properties["metric"];
    var delta = BitConverter.ToInt32(eventData.Body, eventData.Body.Offset);

    // The "Counter/{metricType}" entity is created on-demand.
    var entityId = new EntityId("Counter", metricType);
    await entityClient.SignalEntityAsync(entityId, "add", delta);
}
```



Durable Entities

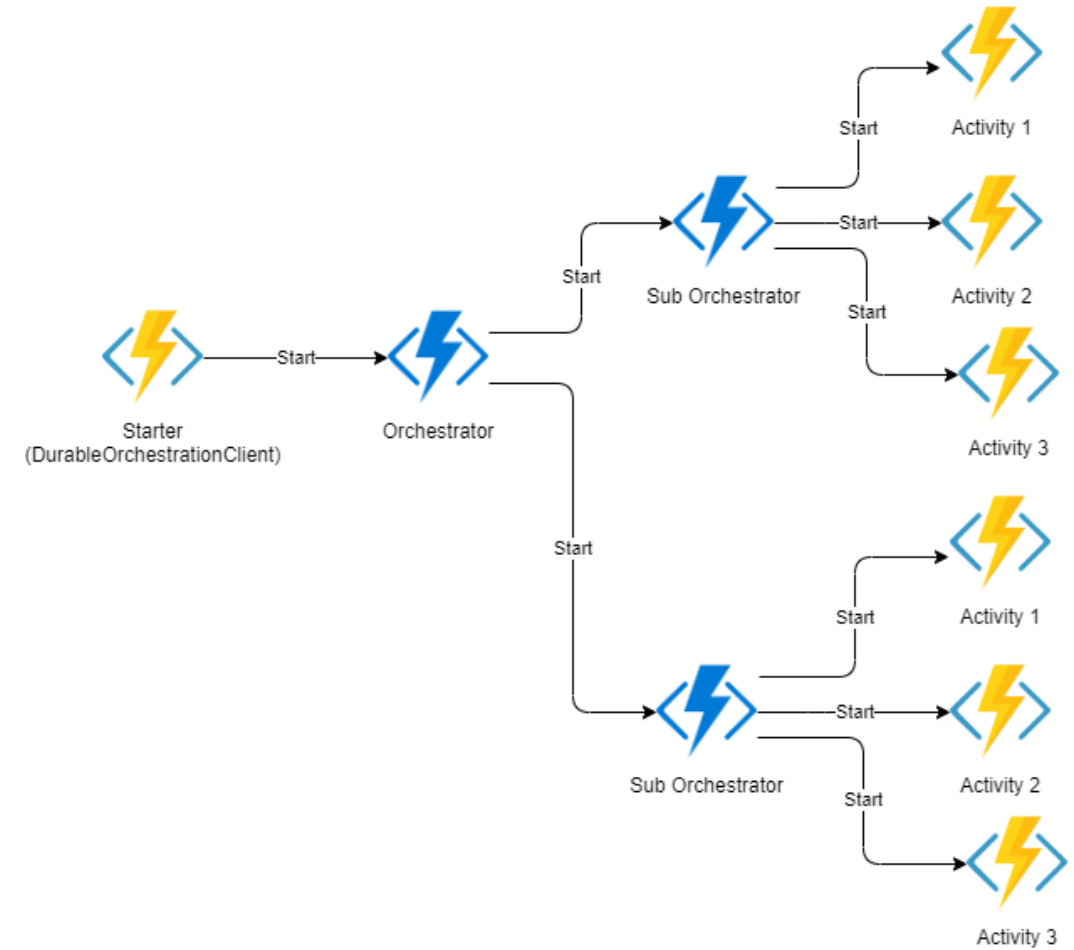
Aggregate event data over a period of time
from different points and data sources

Sub-orchestrations in an orchestration

Sub-orchestrations are a feature of Durable Functions that allow you to further compose and reuse functions.

Essentially sub-orchestrations allow you to call **an orchestration from within another orchestration**.

Supported .NET, Javascript, Python



Sub-orchestrations (Example)

C#

JavaScript

Python

C#

Copy

```
[FunctionName("ProvisionNewDevices")]
public static async Task ProvisionNewDevices(
    [OrchestrationTrigger] IDurableOrchestrationContext context)
{
    string[] deviceIds = await context.CallActivityAsync<string[]>("GetNewDeviceIds");

    // Run multiple device provisioning flows in parallel
    var provisioningTasks = new List<Task>();
    foreach (string deviceId in deviceIds)
    {
        Task provisionTask = context.CallSubOrchestratorAsync("DeviceProvisioningOrchestration", deviceId);
        provisioningTasks.Add(provisionTask);
    }

    await Task.WhenAll(provisioningTasks);

    // ...
}
```

Sub-orchestrations - Pros

Clean Code

Error handling and ability to retry the orchestration

Run sub-orchestrations in parallel

Reuse across workflows



Azure Durable Functions
Sub-orchestrations in different languages

Eternal Orchestration

C#

JavaScript

Python

PowerShell

Java

C#

 Copy

```
[FunctionName("Trigger_Eternal_Orchestration")]
public static async Task<HttpResponseMessage> OrchestrationTrigger(
    [HttpTrigger(AuthorizationLevel.Function, "post", Route = null)] HttpRequestMessage request,
    [DurableClient] IDurableOrchestrationClient client)
{
    string instanceId = "StaticId";

    await client.StartNewAsync("Periodic_Cleanup_Loop", instanceId);
    return client.CreateCheckStatusResponse(request, instanceId);
}
```

Developing Azure Durable Functions



Azure Portal



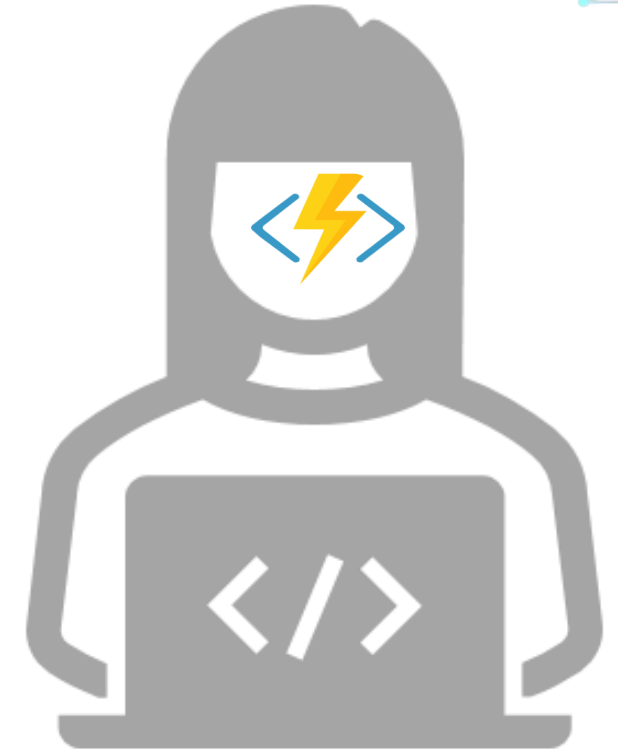
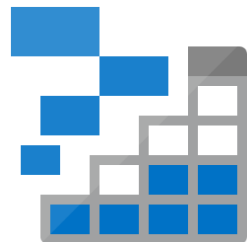
Visual Studio, VS Code, Supported IDEs



Programming Language of Choice!
C#, Java, Powershell, Python, Typescript, etc.



Latest Version of Azure Functions Core tools



Error Handling - Durable Functions

Call another function activity

Durable Timers

Use Automatic Retry on Failure –
`CallActivityWithRetryAsync()`

App Logging / Application Insights

Try-Catch Blocks

Durable Functions Monitor

func durable delete-task-hub
func durable get-history
func durable get-instances
func durable get-runtime-status
func durable purge-history
func durable raise-event
func durable rewind
func durable start-new



Azure Function Tools for Durable Functions

Azure Functions

C#

Azure

Cloud

Functions worker ⓘ

.NET 8.0 Isolated (Long Term Support)

Function ⓘ

Durable Functions Orchestration

- ☒ Use Azurite for runtime storage account (AzureWebJobsStorage) ⓘ
- ☐ Enable container support ⓘ
- ☐ Enlist in .NET Aspire orchestration ⓘ

Azurite, Container, .NET Aspire Support

2024-10-03

Overview **Platform features** Log streaming

Search features

General Settings

- Function app settings
- Application settings
- Properties
- Backups
- All settings

Code Deployment

- Deployment Center

Development tools

- Logic Apps
- Console (CMD / PowerShell)
- Advanced tools (Kudu)
- App Service Editor
- Resource Explorer
- Site Extensions

Networking

- Networking
- SSL
- Custom domains
- Authentication / Authorization
- Identity
- Push notifications

Monitoring

- D diagnostic logs
- Log streaming**
- Process explorer
- Metrics

API

- API definition
- CORS

App Service plan

- App Service plan
- Scale up
- Scale out
- Quotas

Resource management

- Diagnose and solve problems
- Activity log
- Access control (IAM)
- Tags
- Locks
- Automation script

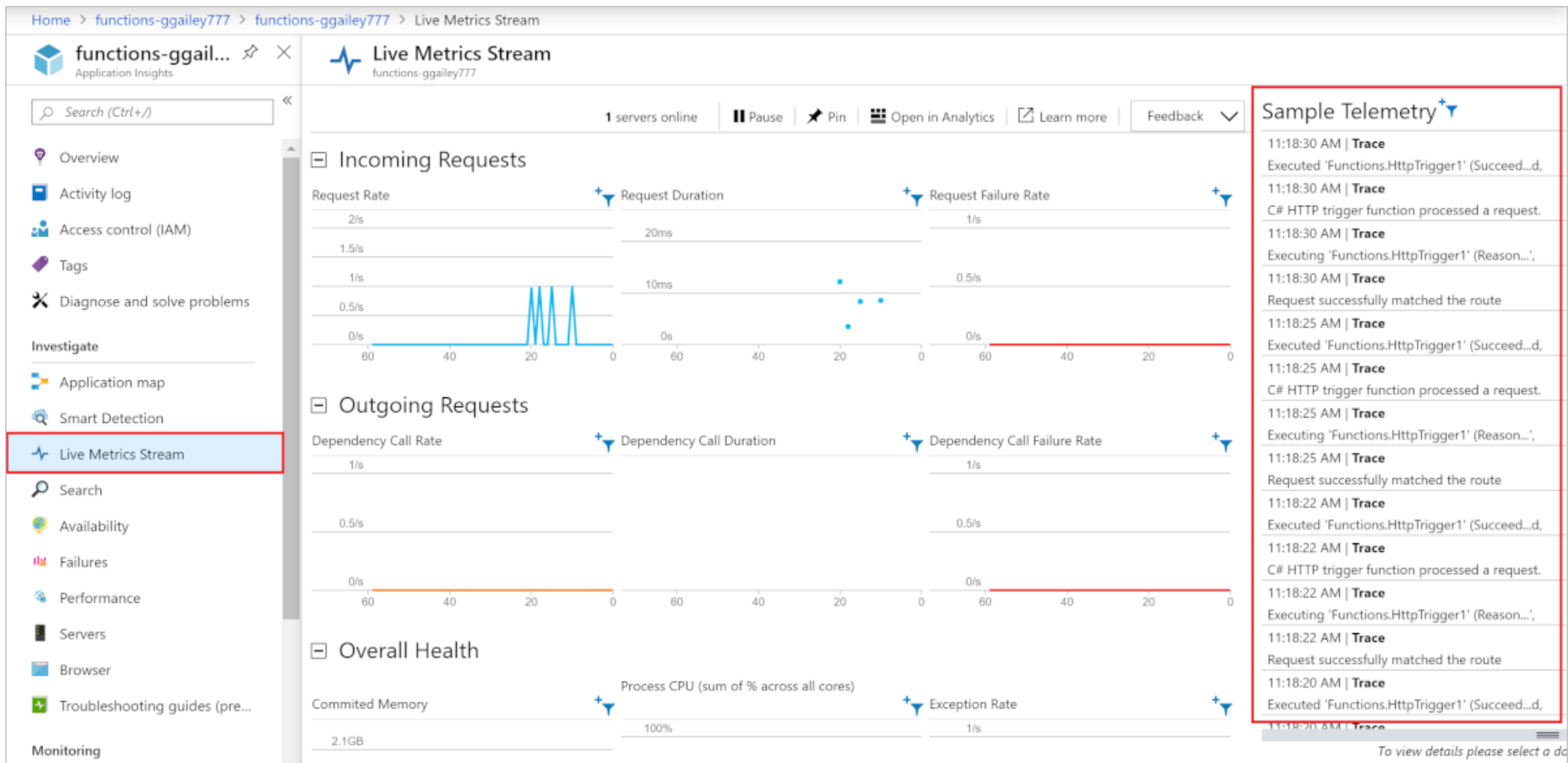
Overview Platform features Log streaming

Reconnect Copy logs Pause Clear

Application logs Web Server logs

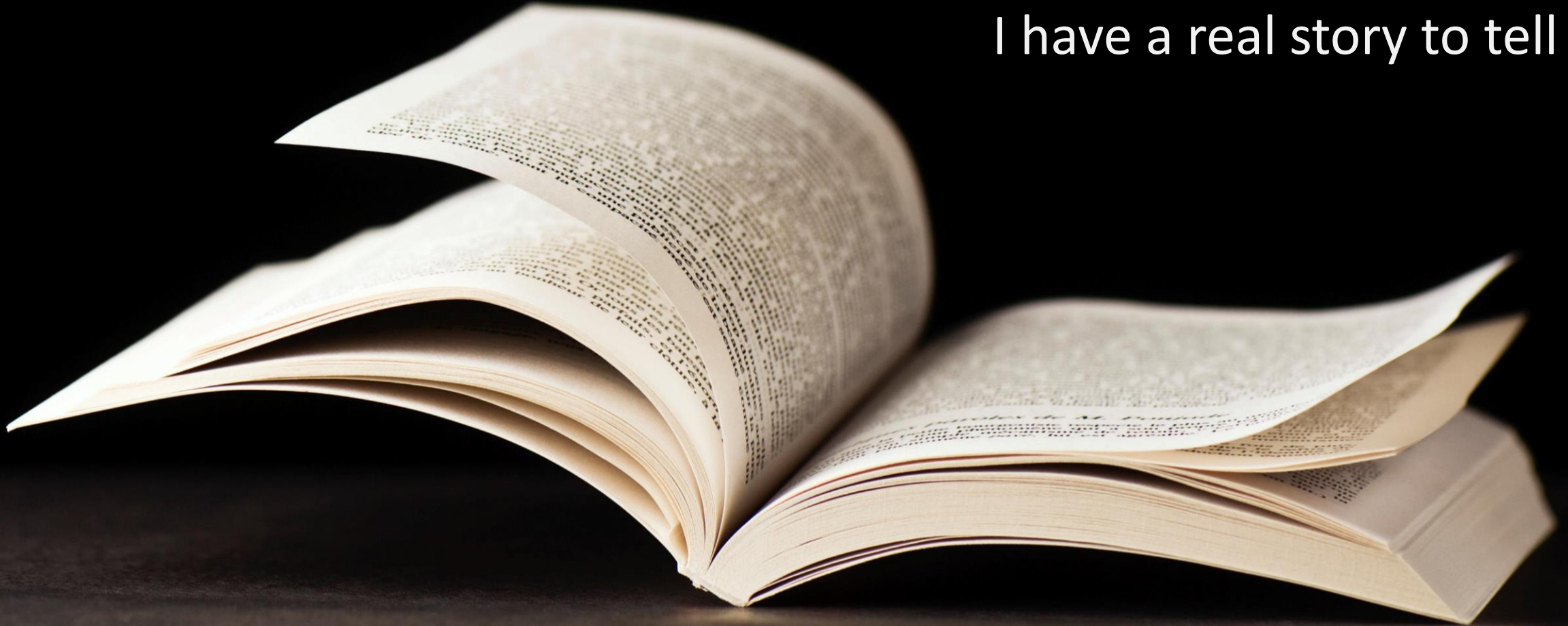
```
2019-04-08T07:15:19.398 [Information] Executing 'Functions.HttpTriggerCSharp1'
(Reason='This function was programmatically called via the host APIs.', Id=62b1969f-
b568-4291-83ef-691e11d0d2e8)
2019-04-08T07:15:19.399 [Information] C# HTTP trigger function processed a request.
2019-04-08T07:15:19.399 [Information] Executed 'Functions.HttpTriggerCSharp1'
(Succeeded, Id=62b1969f-b568-4291-83ef-691e11d0d2e8)
2019-04-08T07:15:19.398 [Information] Executing 'Functions.HttpTriggerCSharp1'
(Reason='This function was programmatically called via the host APIs.', Id=62b1969f-
b568-4291-83ef-691e11d0d2e8)
2019-04-08T07:15:19.399 [Information] C# HTTP trigger function processed a request.
2019-04-08T07:15:19.399 [Information] Executed 'Functions.HttpTriggerCSharp1'
(Succeeded, Id=62b1969f-b568-4291-83ef-691e11d0d2e8)
2019-04-08T07:16:05.008 [Information] Executing 'Functions.TimerTriggerCSharp1'
(Reason='Timer fired at 2019-04-08T00:16:05.0074271-07:00', Id=3f5f11b1-f243-454f-
9e58-6ab7916d68dc)
2019-04-08T07:16:05.009 [Information] C# Timer trigger function executed at: 4/8/2019
12:16:05 AM
2019-04-08T07:16:05.010 [Information] Executed 'Functions.TimerTriggerCSharp1'
(Succeeded, Id=3f5f11b1-f243-454f-9e58-6ab7916d68dc)
2019-04-08T07:16:05.008 [Information] Executing 'Functions.TimerTriggerCSharp1'
(Reason='Timer fired at 2019-04-08T00:16:05.0074271-07:00', Id=3f5f11b1-f243-454f-
9e58-6ab7916d68dc)
2019-04-08T07:16:05.009 [Information] C# Timer trigger function executed at: 4/8/2019
12:16:05 AM
2019-04-08T07:16:05.010 [Information] Executed 'Functions.TimerTriggerCSharp1'
(Succeeded, Id=3f5f11b1-f243-454f-9e58-6ab7916d68dc)
2019-04-08T07:16:10.768 [Information] Executing 'Functions.HttpTriggerCSharp1'
(Reason='This function was programmatically called via the host APIs.', Id=7be06fb2-
cb0f-4825-8e76-500dc7766891)
2019-04-08T07:16:10.769 [Information] C# HTTP trigger function processed a request.
2019-04-08T07:16:10.769 [Information] Executed 'Functions.HttpTriggerCSharp1'
(Succeeded, Id=7be06fb2-cb0f-4825-8e76-500dc7766891)
2019-04-08T07:16:10.768 [Information] Executing 'Functions.HttpTriggerCSharp1'
```

Monitoring – Live Log Streaming



Monitoring – Live Metrics Stream

I have a real story to tell



EXPERIENCED SELF-ORGANIZING DEVELOPERS

DEVOPS / CLOUD INFRASTRUCTURE ENGINEERS

CONSULTANTS

BUSINESS STAKEHOLDERS

DATABASE ADMINISTRATORS

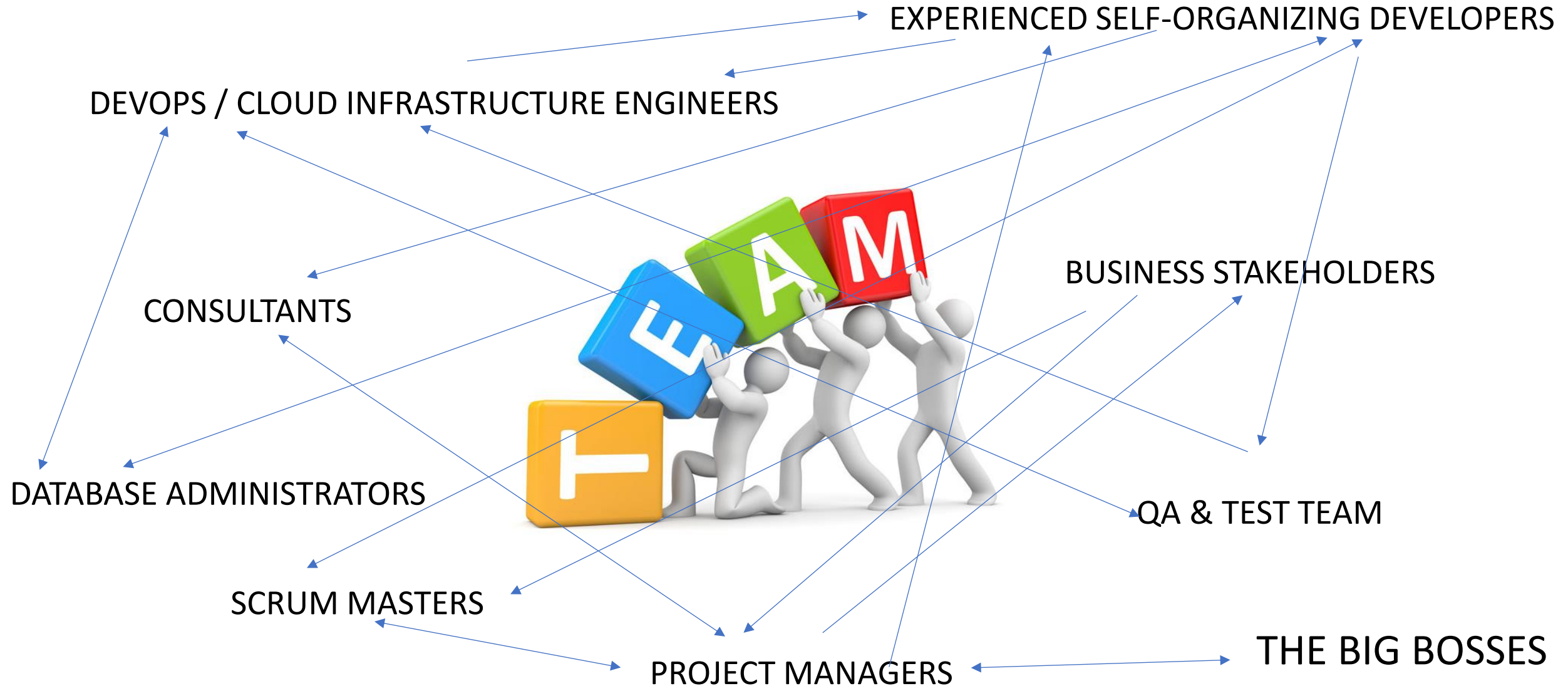
QA & TEST TEAM

SCRUM MASTERS

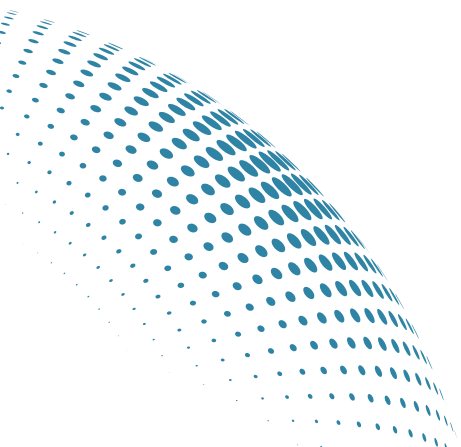
PROJECT MANAGERS



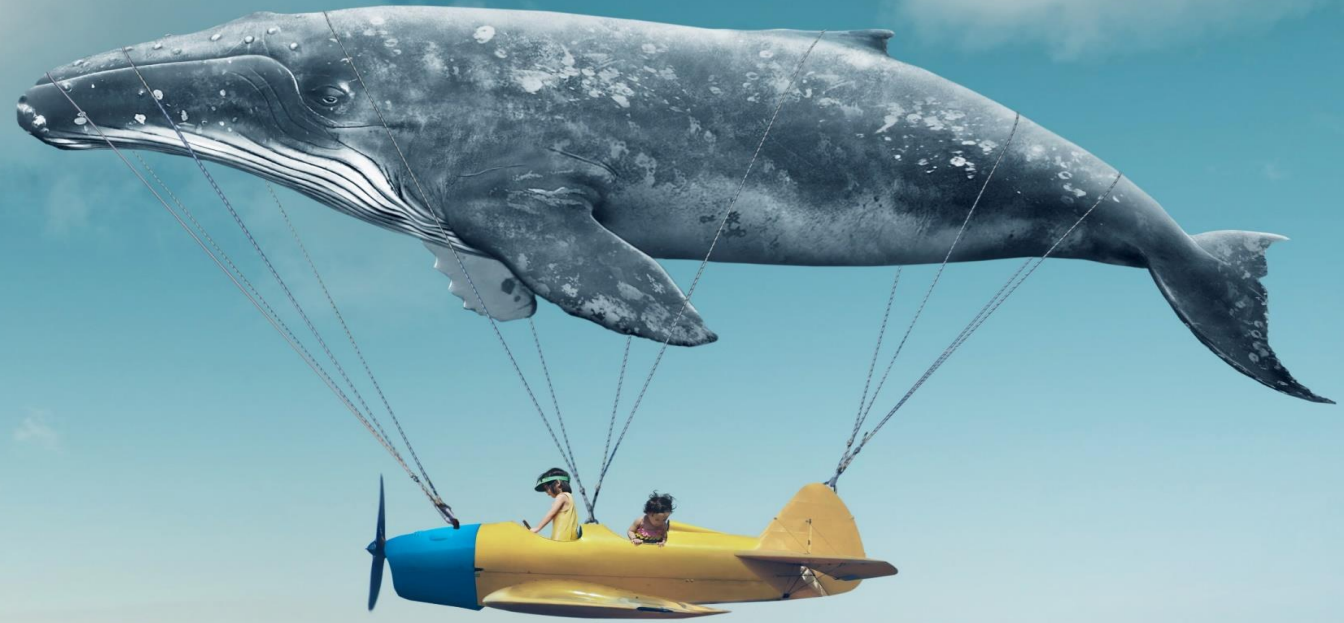
Jonah's Deep Mind Map Thinking in Action! 😊



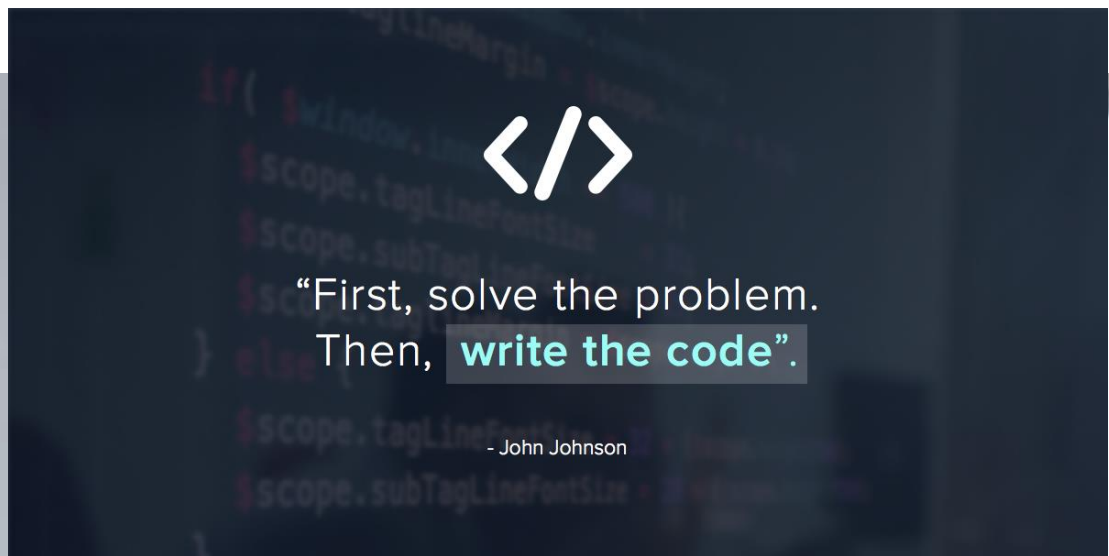
THE DILEMMA



ARE WE REALLY IN THE CLOUD?









Proposed Solution



Azure
Durable Functions

 Microsoft Azure



 **ATLASSIAN** Developer Bitbucket Cloud

[REST APIs](#) [Modules](#) [JavaScript API](#)

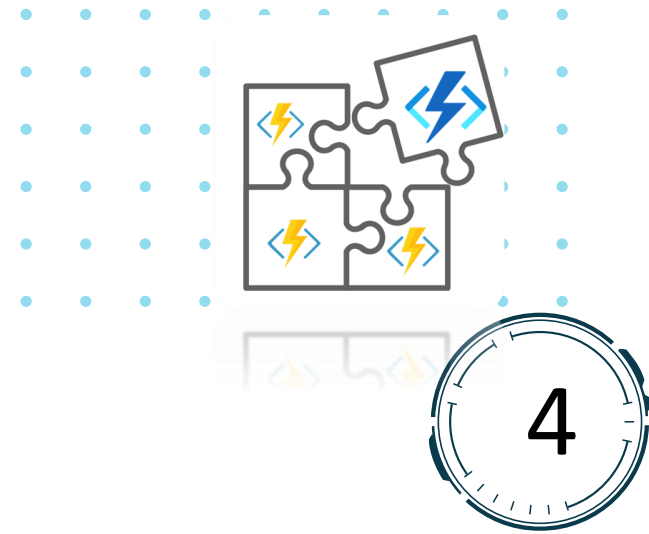
Bitbucket (Cloud) REST API

<https://developer.atlassian.com/>



Azure DevOps Services REST API

<https://learn.microsoft.com/en-us/rest/api/azure/devops>



Azure Service Bus

AZURE SERVICE BUS

Messaging service on cloud

Connect any applications,
devices, and services

A messaging broker



FEATURES

- Scheduling
- Dead-lettering
- Transactional processing
- Message ordering
- Expiration (TTL)
- Duplicate detection
- Deferring
- Pub/Sub
- Filtering

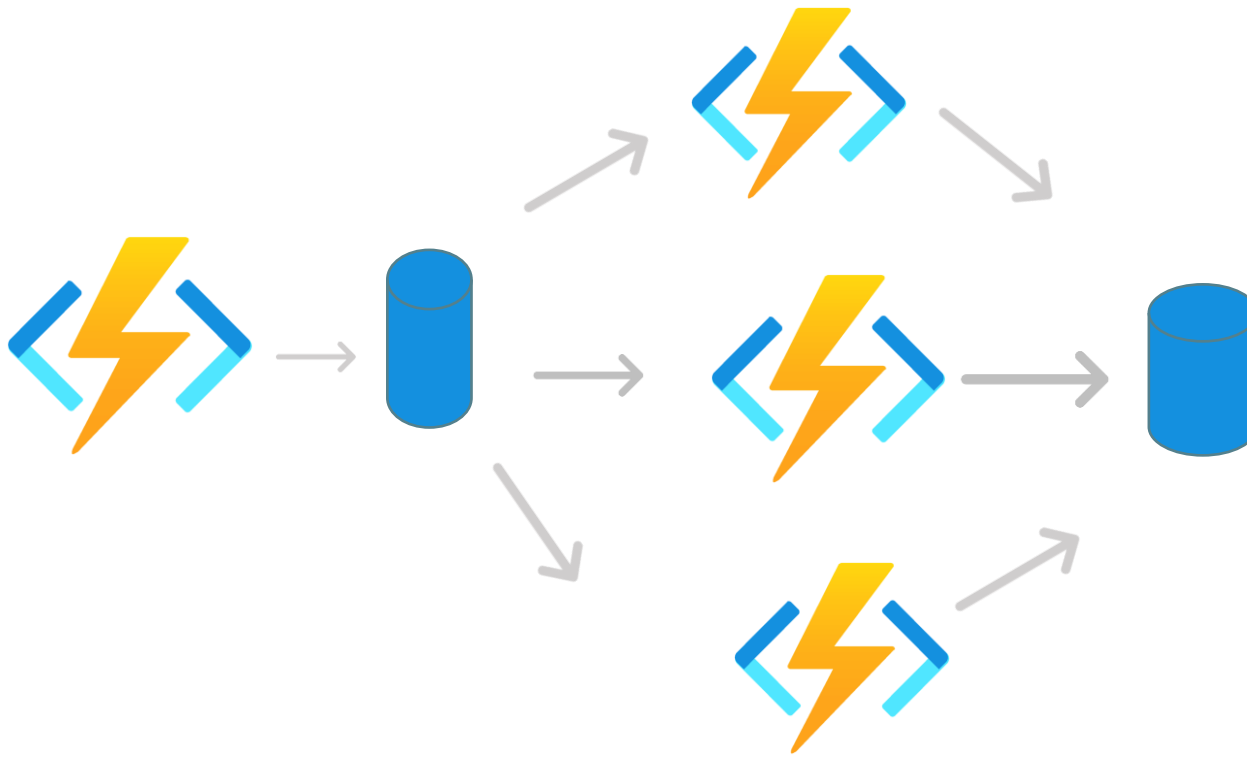
YES

LET'S DO IT!



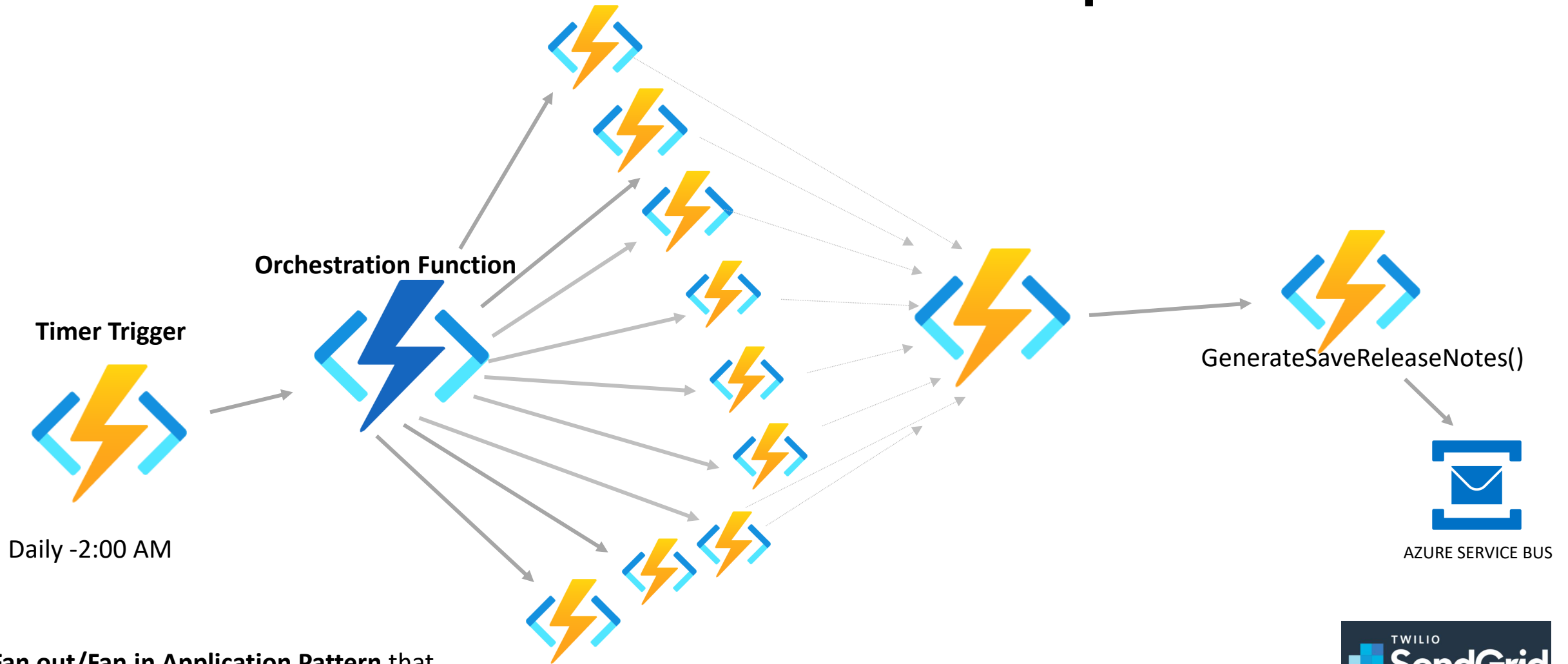
JUST DO IT.





Fan-Out/Fan In Pattern

#1 Getting the commits from Bitbucket Cloud via Azure DevOps Rest API



Fan out/Fan in Application Pattern that gets all Commits from Bitbucket Cloud Repositories from 10-15 applications/web services. Checks commit difference from since last deploy date.



Cont. #2 Auto Scheduling of Release Notes Email



Azure Service Bus Queue Trigger

Scheduled Messaging

`ScheduledEnqueueTimeUtc()`

*Release Notes as
Queue Messages in Batch*

`ComposeSendEmailToBusinessStakeHolders()`

`CompleSendEmailToITDevTeam()`

`SendEmailToReleaseManager()`

Orchestrator Function

Scheduled release notes
sendouts based on
release events

`ComposeSendEmailToExternalTeams()`



TIME CHECK FOR DEMO



Good to know before you get started!

Azure Application
Insights / Azure
Monitor

Concious
Versioning

Side-by-Side
Deployment

Deployment with
Breaking Change

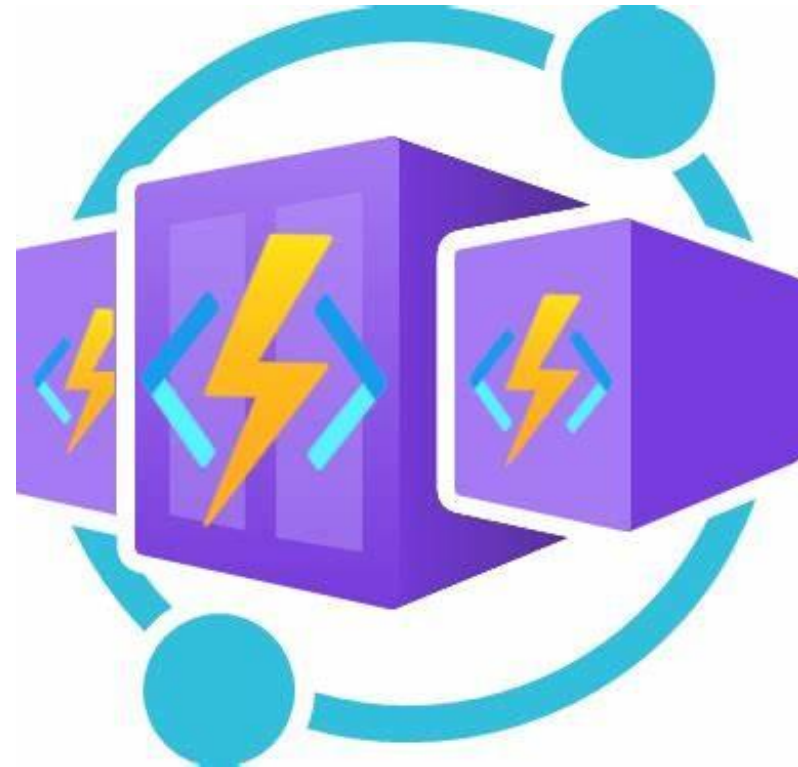
Azure Storage
Explorer / Azurite

Deployment Slots

Azure Key Vault /
Serverless Security

Isolated vs.
In Process (.NET)

*Serverless
computing
into cloud
containers*





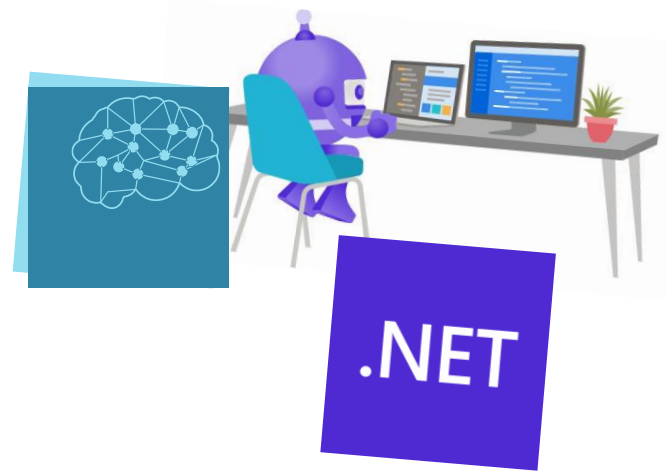
**Solve problems
by automating manual tasks using
Serverless or good technologies if possible**

WORK LESS
WITH SERVERLESS 😊

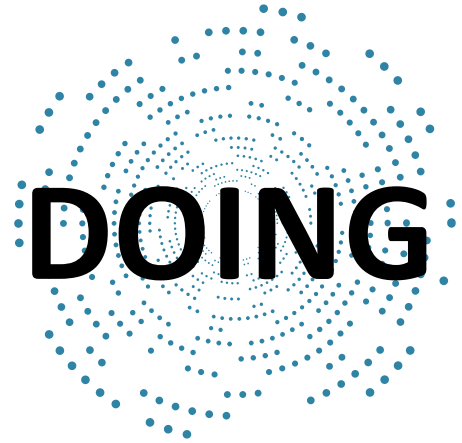
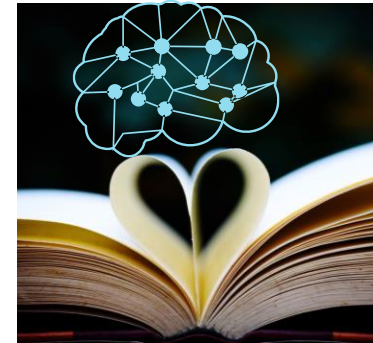




AZURE DURABLE FUNCTIONS



LEARN BY DOING



18 minutes non-stop
to Stockholm C,
that's express.



Buy tickets here.

Arlanda
express



After 30
mins,

My luggage
finally came
out!



**Rate this session
in the GOTO Guide app**
and claim your reward



goto;
copenhagen

THANK YOU

Feel free to connect



Any questions?



Learn and Have Fun like my dog, Lorentz!

