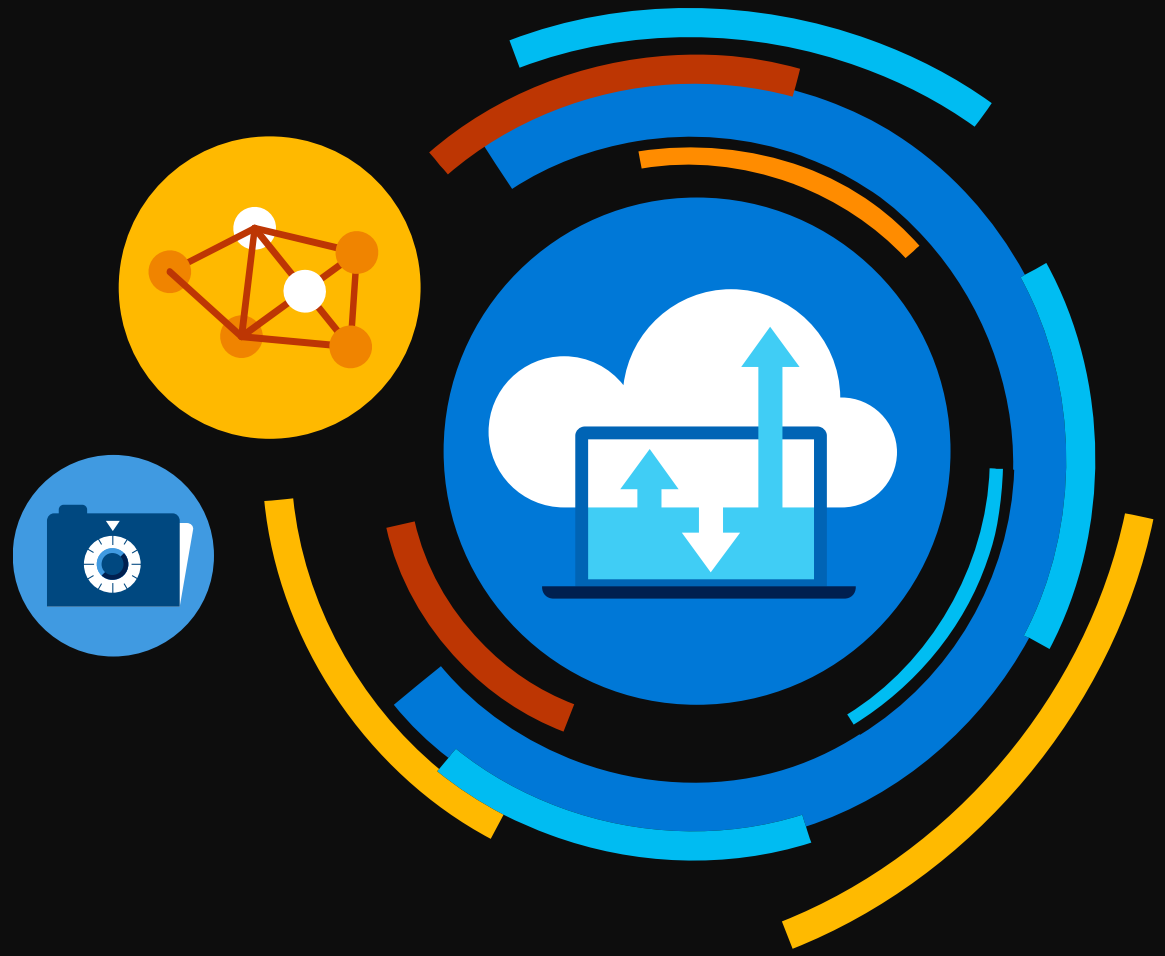
Microsoft

http://bit.ly/AzureSPC-FuncJS
http://bit.ly/AzureSPC-FuncCS
http://bit.ly/AzureSPC-FuncSPFx

Mastering Azure Functions

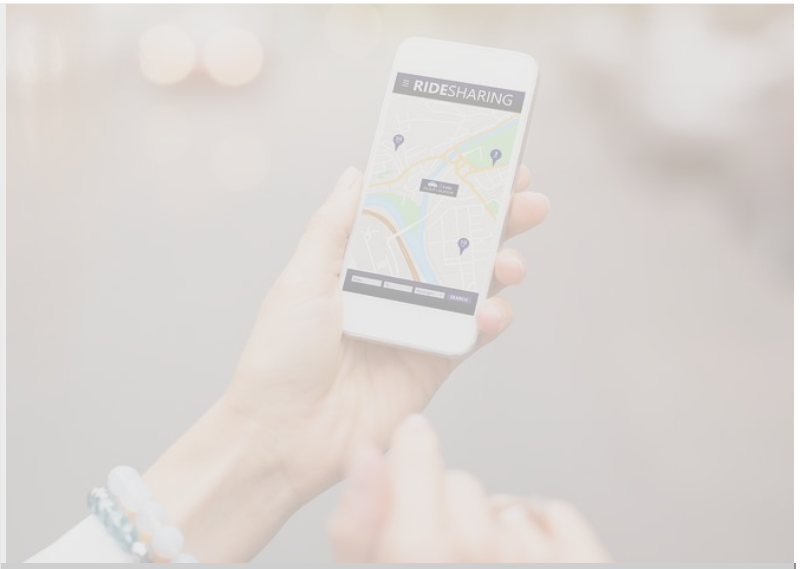


Bob German
Partner Technical Architect

ocp | One Commercial Partner

IaaS

PaaS



Serverless

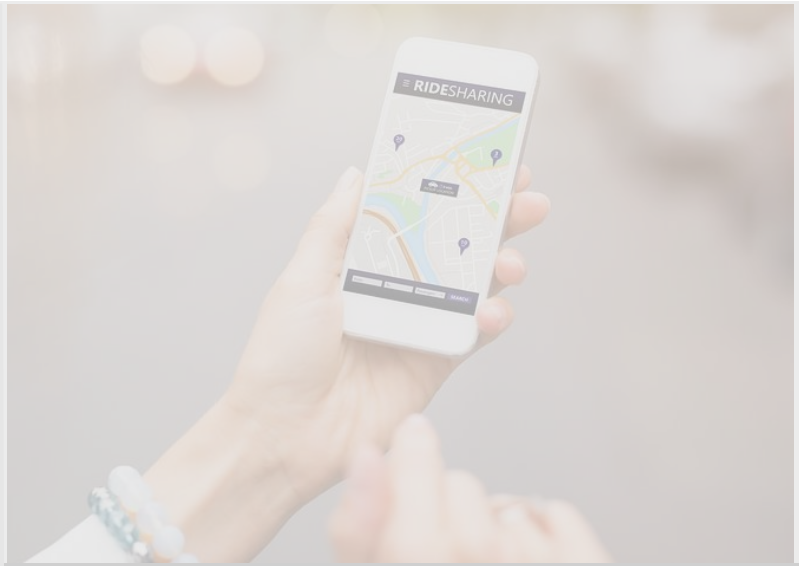
...you can buy or lease a car
and maintain it yourself

IaaS

...you can buy or lease a car and maintain it yourself

PaaS

...you can rent a car and pay for having it around even when you are not driving



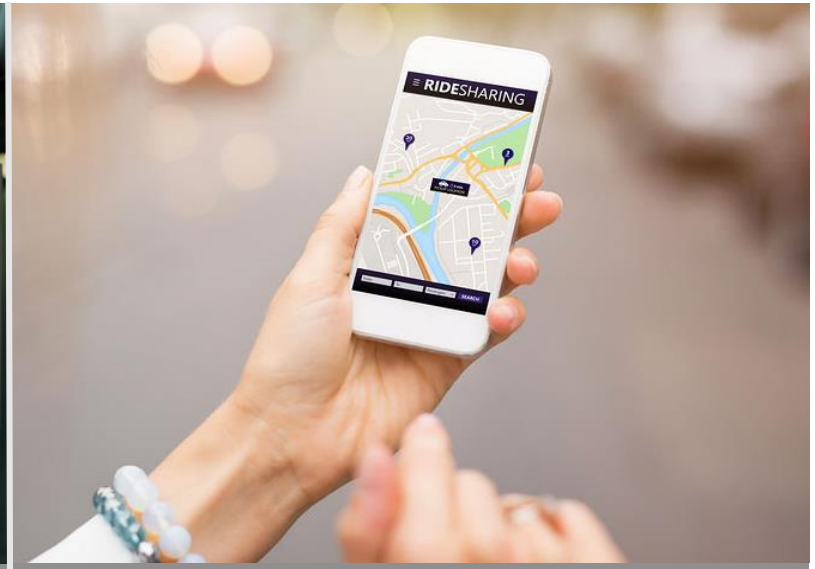
Serverless

IaaS

...you can buy or lease a car and maintain it yourself

PaaS

...you can rent a car and pay for having it around even when you are not driving



Serverless

...you can use a ride sharing app pay only for transportation

Serverless application platform components

Development

- IDE support
- Visual debug history
- Local development
- Verbose debugging

Platform

Functions

- Developer tooling
- Bindings and triggers
- Open source

Logic apps

- Visual designer
- 100+ connectors
- Functions orchestration

Data/storage

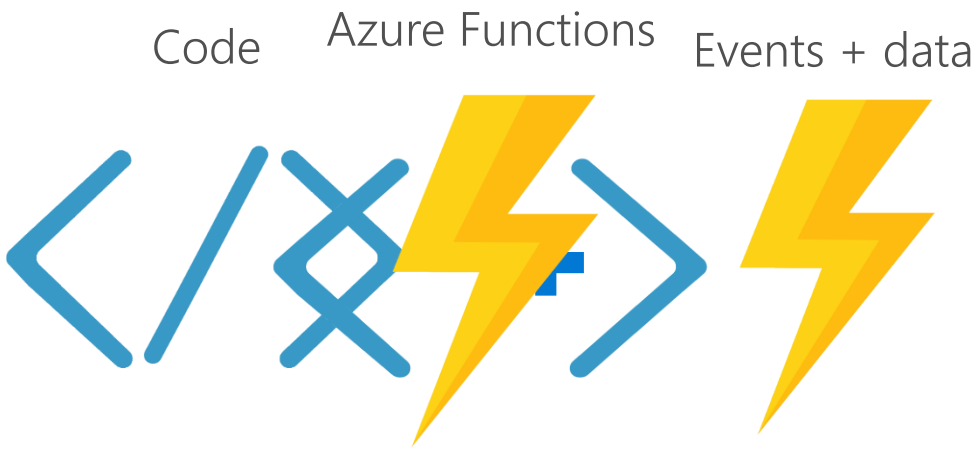
Messaging

Gateway Connectors

Intelligence

Bots

Azure Functions



Process events with Serverless code.

Make composing Cloud Apps insanely easy

Develop Functions in C#, Node.js, F#, Python, PHP, Batch and more

Easily schedule event-driven tasks across services

Expose Functions as HTTP API endpoints

Scale Functions based on customer demand

Easily integrate with Logic Apps

The runtime is open source:
https://github.com/Azure/Azure-Functions

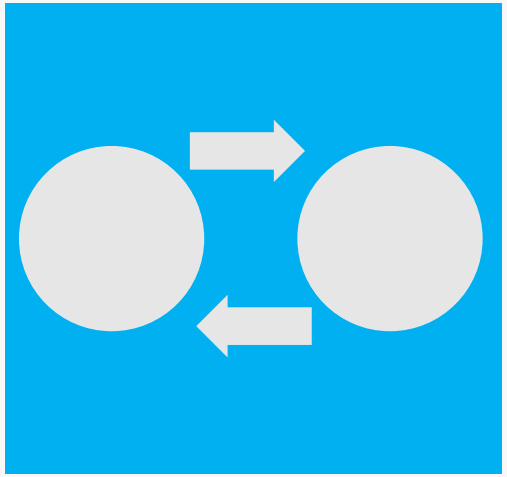
“Functions” programming model

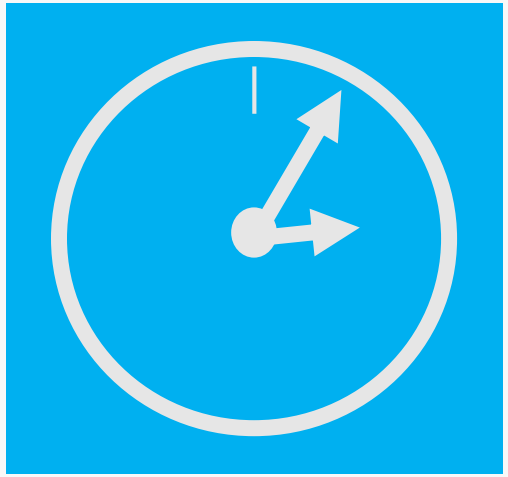
- Function as the unit of work
- Functions are executed; they start and finish
- Functions have inputs and outputs

```
public async static Task ProcessQueueMessageAsyncCancellationToken(
    [QueueTrigger("blobcopyqueue")] string blobName,
    [Blob("textblobs/{queueTrigger}",FileAccess.Read)] Stream blobInput,
    [Blob("textblobs/{queueTrigger}-new",FileAccess.Write)] Stream blobOutput,
    CancellationToken token)
{
    await blobInput.CopyToAsync(blobOutput, 4096, token);
}
```

Best practices for the “Functions” programming model

- Functions *should* “do one thing”
- Functions *should* be stateless and idempotent
- Functions *should* finish as quickly as possible



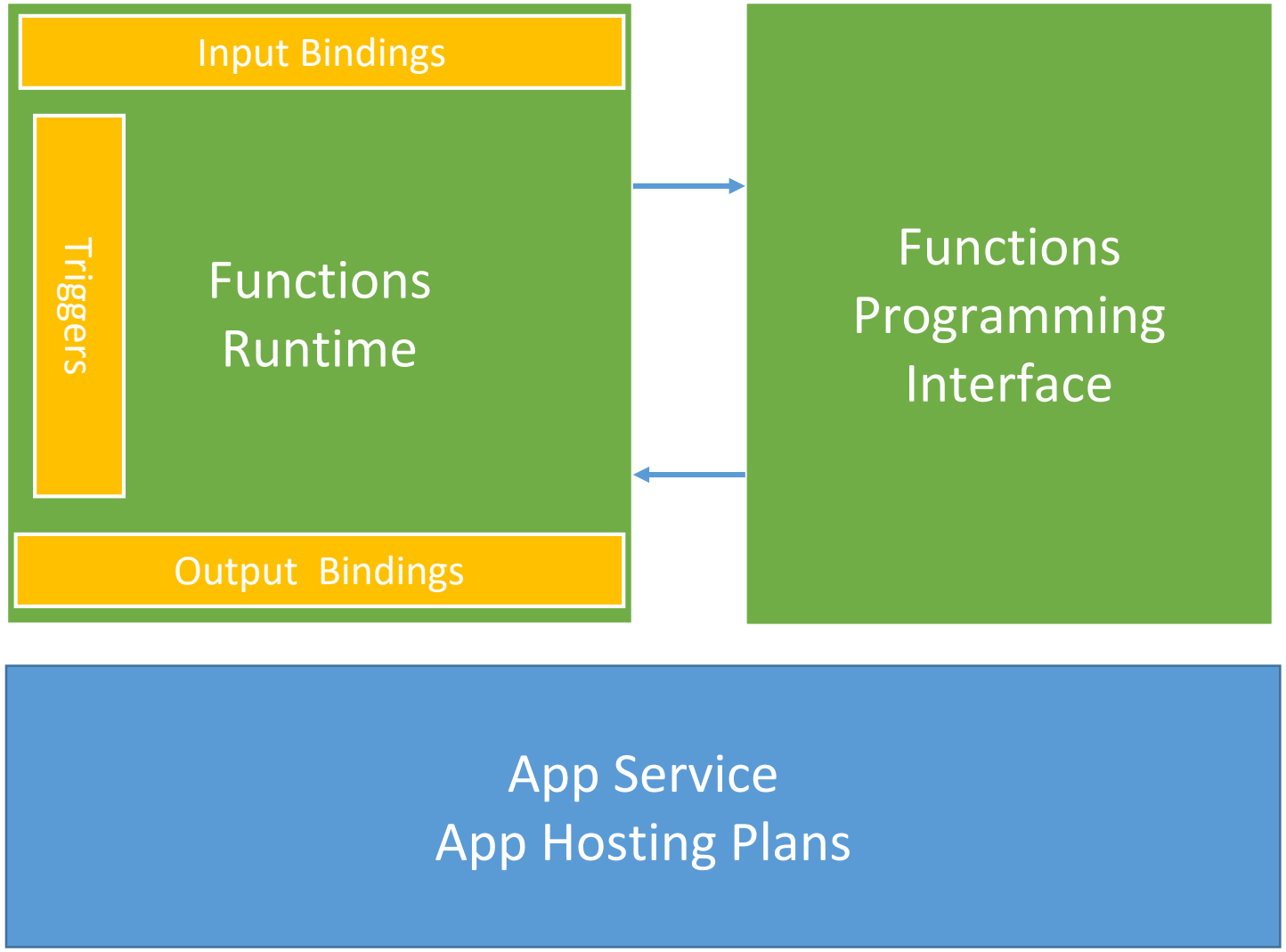


How to use the “Functions” programming model

1. What is you need to do? (business logic wise)
2. Am I solving more than one business problem? Split up, go back to 1.
3. What will trigger the function?
4. Is there additional data I need?
5. Is there output I should produce?

“When ___, get ___, do ___, and output ___”

Azure Functions Architecture



Azure Functions - Languages

	Functions 1.x	Functions 2.x
C#	GA	Preview
JavaScript	GA	Preview
F#	GA	Preview
Java		
PowerShell	Experimental	
Python	Experimental	
PHP	Experimental	
TypeScript	Experimental	
Batch (.cnd, .bat)	Experimental	
Bash	Experimental	

Experimental options

- Don’t scale well
- Don’t support all bindings
- v1 only (there will be no v2 Experimental options)
- Not supported – officially not for production use!
- Yet … widely used as if these constraints didn’t exist!

Triggers and bindings
configured in function.json

Trigger

Invokes the function

- Always exactly 1

Input binding

Handles incoming data

- Optional

Output binding

Handles outgoing data

- Optional

Azure Function Bindings

Type	1.x	2.x	Trigger	Input	Output
Blob Storage	✓	✓[1]	✓	✓	✓
Cosmos DB	✓	✓	✓	✓	✓
Event Grid	✓	✓	✓		
Event Hubs	✓	✓	✓		✓
External File[2]	✓			✓	✓
External Table[2]	✓			✓	✓
HTTP	✓	✓[1]	✓		✓
Microsoft Graph Excel tables		✓		✓	✓
Microsoft Graph OneDrive files		✓		✓	✓

Type	1.x	2.x	Trigger	Input	Output
Microsoft Graph Outlook email		✓			✓
MS Graph Events		✓	✓	✓	✓
MS Graph Auth tokens		✓		✓	
Mobile Apps	✓	✓		✓	✓
Notification Hubs	✓				✓
Queue storage	✓	✓[1]	✓		✓
SendGrid	✓	✓			✓
Service Bus	✓	✓	✓		✓
Table storage	✓	✓[1]		✓	✓
Timer	✓	✓	✓		
Twilio	✓	✓			✓
Webhooks	✓		✓		✓

Azure Function Authorization Types

Function (pass a function key)

Admin (pass function app's host key)

System (pass function app's master key)

Anonymous (can be used in conjunction with App Service authN)

Coming soon: User – this will be token based

Platform and scaling

- App Service offers dedicated and dynamic tiers.
- Dedicated is the existing App Service plan tiers
  - Basic, Standard, Premium
  - Pay based on # of reserved VMs
  - You’re responsible for scale
- Dynamic
  - Pay on number of executions
  - Platform responsible for scale

Dynamic tier pricing

Pay per execution model - two meters, three units

- Number of executions
- Duration of execution x reserved memory

Use bindings in your code

run.csx

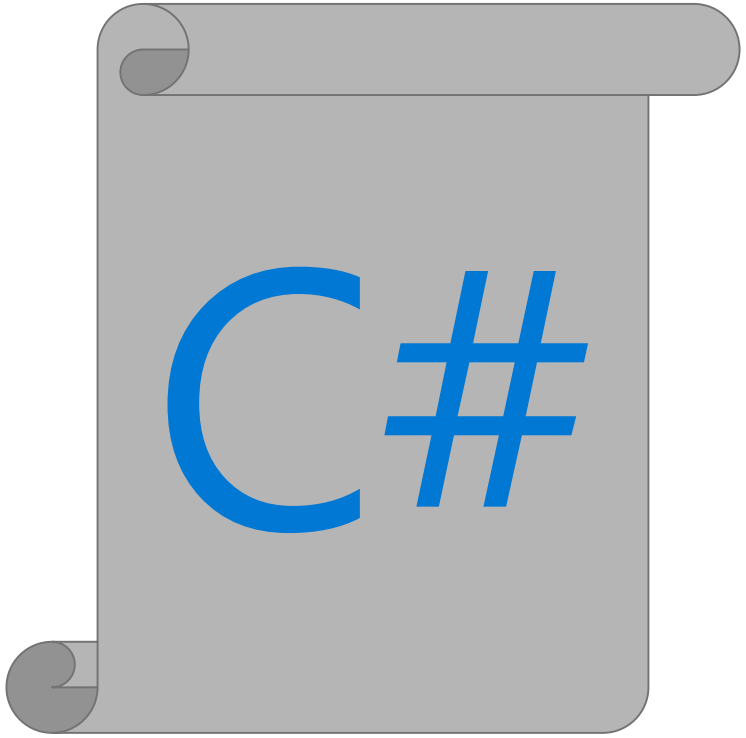
```
public static void Run(byte[] image, string filename,
                       Stream outputBlob, TraceWriter log)
{
    log.Info($"Processing image: {filename}");

    var imageBuilder = ImageResizer.ImageBuilder.Current;

    imageBuilder.Build(
        image, outputBlob,
        new ResizeSettings(640, 400, FitMode.Max, null), false);
}
```

function.json

```
{
  "bindings": [
    {
      "name": "image",
      "type": "blobTrigger",
      "direction": "in",
      "path": "card-input/{filename}.jpg",
      "connection": "AzureWebJobsStorage"
    },
    {
      "type": "blob",
      "name": "outputBlob",
      "path": "card-output/{filename}.jpg",
      "connection": "AzureWebJobsStorage",
      "direction": "out"
    }
  ]
}
```

Roslyn: Scripting for C#

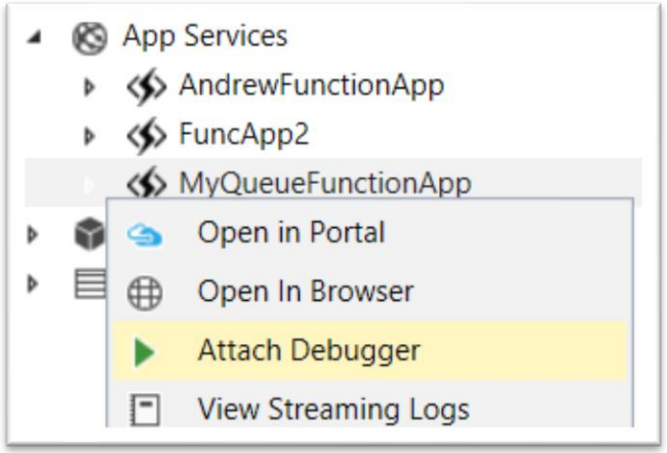
- Dynamically evaluated at runtime
- .csx filename extension
- Relaxed C# syntax
  - No namespaces, project or solution file
  - Global functions allowed
  - **`#r`** pragma to reference external assemblies
  - **`using`** statements and references can be imported implicitly by the hosting application (ie. Azure functions runtime)

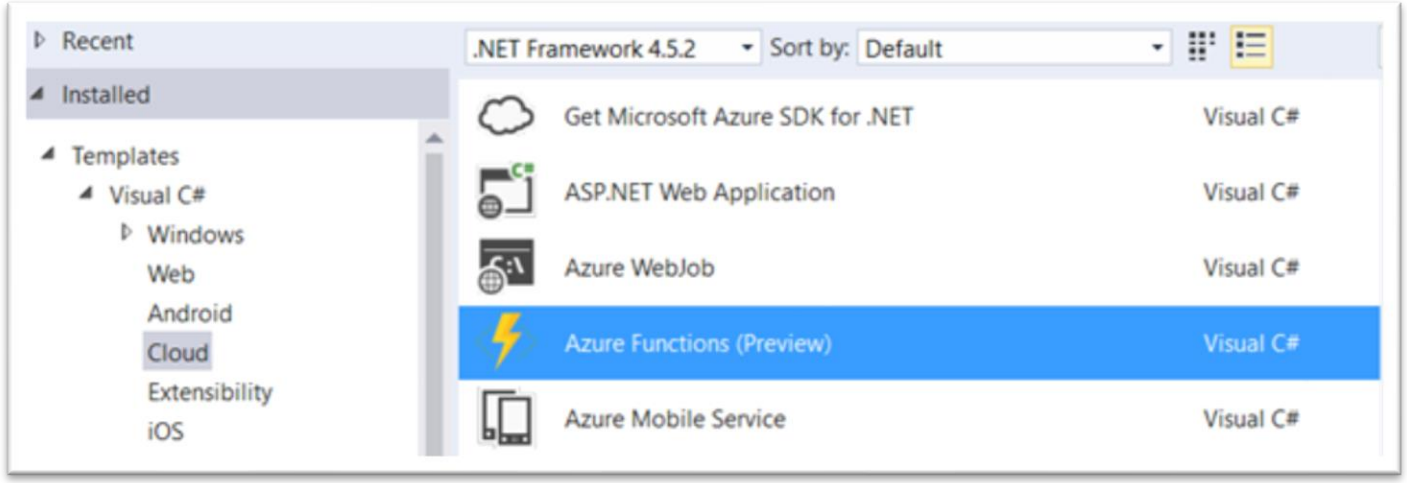


- OR – use compiled C# for faster execution

Azure Function Visual Studio Integration

- Functions project and templates
- Uses Azure Functions Developer Tools
- Run and debug locally or in Azure
- Trigger on events in Azure
  - Example: add a new queue item in Azure and hit a breakpoint within your function code!





DEMO:
Function in C# elevates permission in SharePoint
Using Function key
Consuming from SPFx

Azure Functions Developer Tools*

- Create/ Develop / Manage Azure Functions
- Run and debug locally
  - Run the Functions runtime on your local machine
  - Functions invoked (triggered) based on events in Azure
  - **Not** an emulator or simulator –
    same code that runs your Function Apps in Azure!

Do try this at home!
http://bit.ly/AzFuncTools

Want to *not* elevate permissions? See this great blog series by Vardhaman Deshpande
http://bit.ly/AzFuncOnBehalfOf

DEMO:
Function in Node elevates permission in SharePoint
Using App Services authentication and deployment
Consuming from SPFx

* Formerly known as Azure Functions CLI

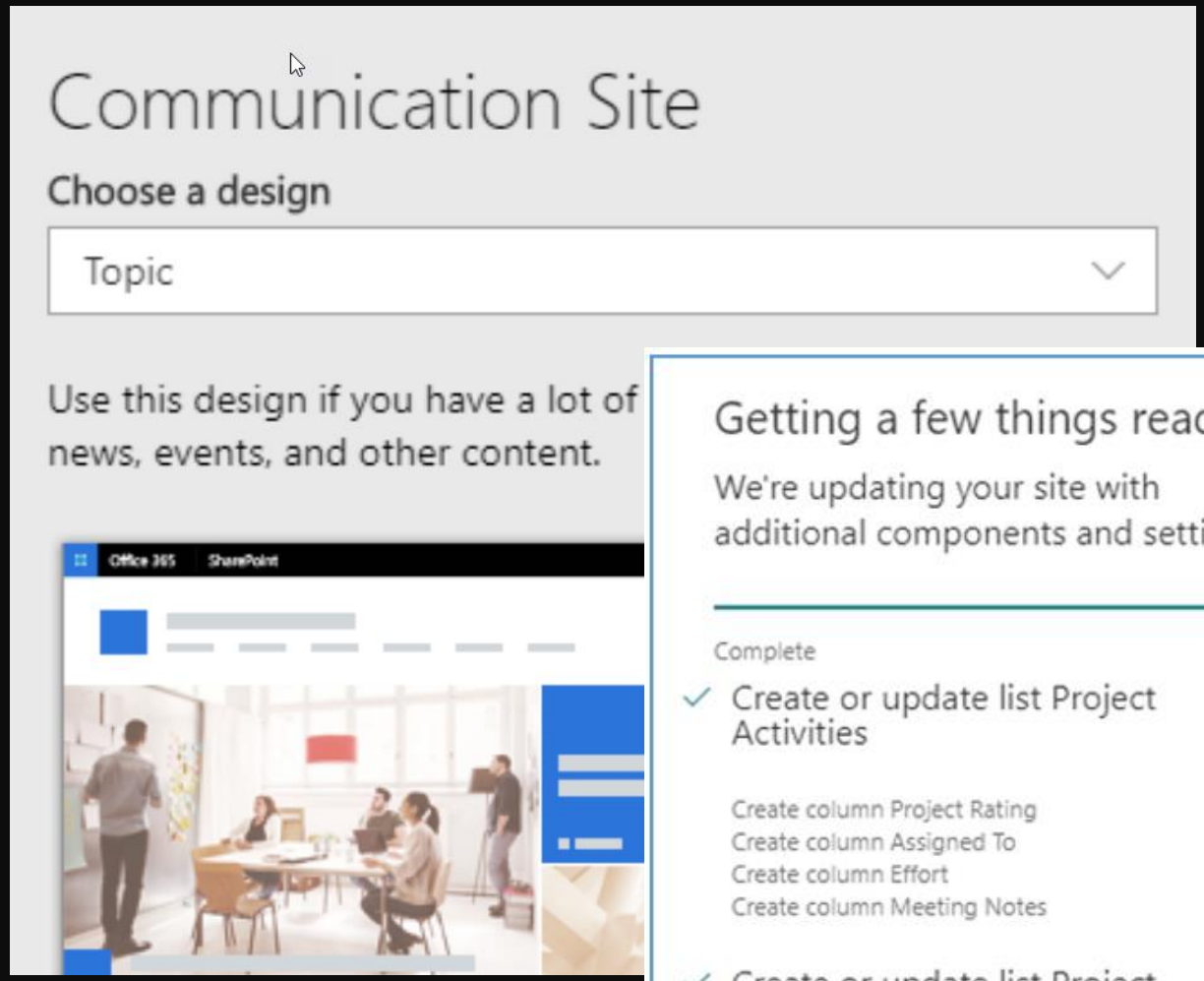
Azure Function Proxies

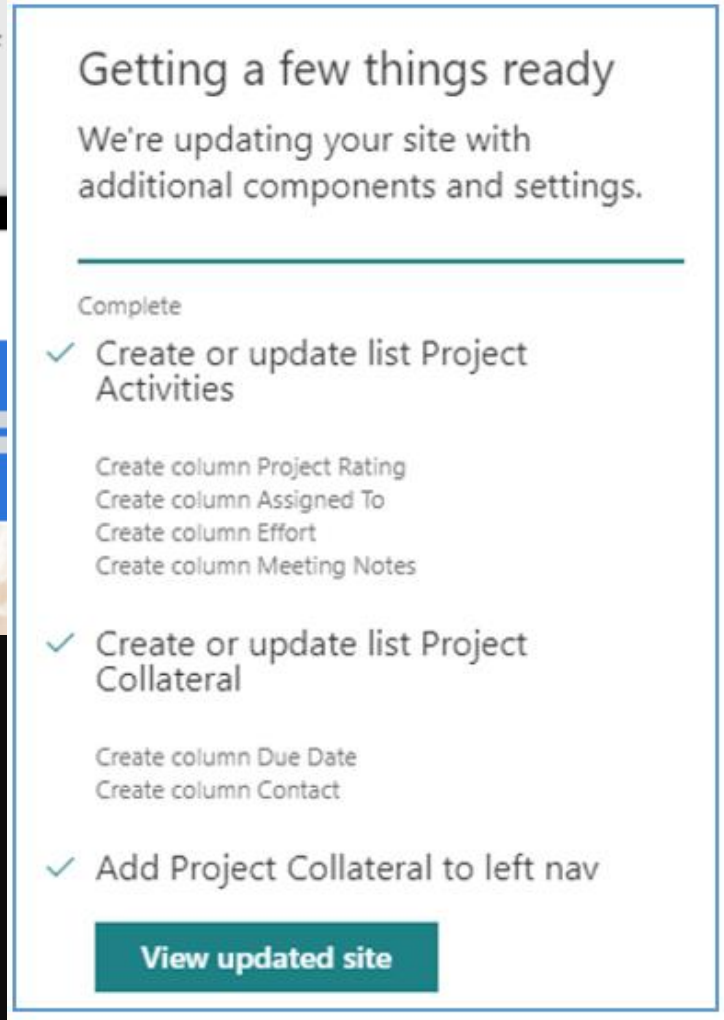
- Light-weight API management
- Change URL, manipulate request and response
- Inherits the configuration of your Function App
  (e.g. authentication)

DEMO:
Hide API key in an Azure Function Proxy

Site Designs and Scripts

- Site Design defines a name, web template, site script, and who can provision
- Site Script carries out a series of actions on a site
  - Run on the server – faster than remote provisioning
  - Idempotent – run again and again





Anatomy of a site script

```
var listRecipe = {
  "$schema": "schema.json",
  "actions": [
    {
      "primary_verb": "List.CreateOrOpen",
      "target": "Customer Tracking",
      "templateType": 100,
      "verbs": [
        {
          "verb": "SetDescription",
          "description": "List of Customers and Orders"
        },
        {
          "verb": "AddFieldIfNotExist",
          "fieldType": "Text",
          "displayName": "Customer Name",
          "isRequired": false,
          "addToDefaultView": true
        },
            {
          "verb": "AddFieldIfNotExist",
          "fieldType": "DateTime",
          "displayName": "Date of Delivery",
          "isRequired": true
        },
      ]
    },
    {
      primary_verb: "Theme.Apply",
      target: "Contoso Travel Green"
    }
  ],
  "bindata": {},
  "version": 1
};
```

Idempotent syntax of common site configuration actions

Script actions can be concatenated in single file or multiple files can be used (and reused)

Site Design attributes designate display characteristics and target template

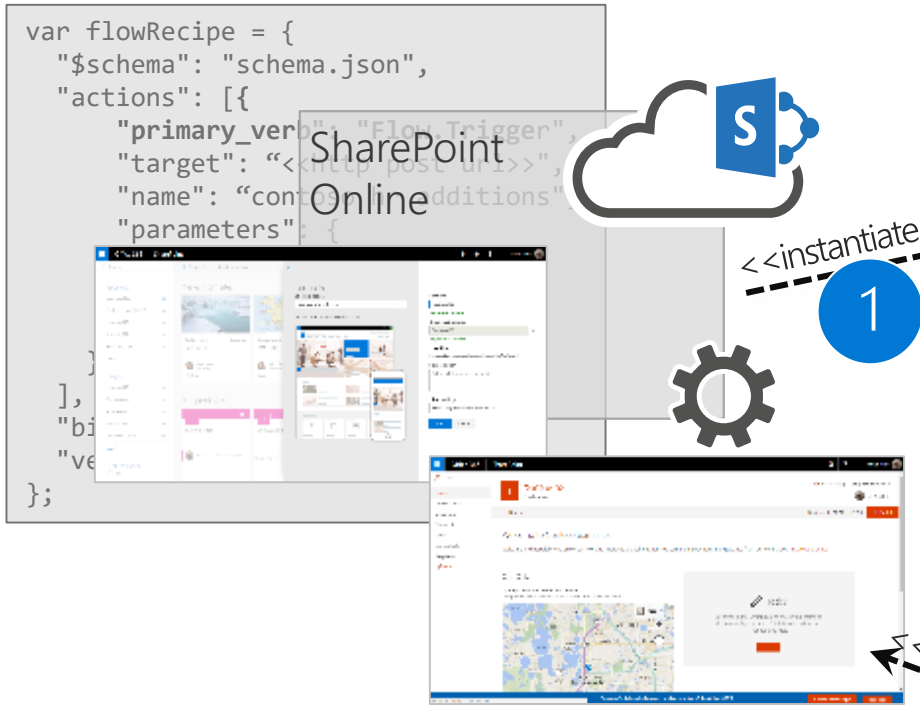
```
RestRequest("/_api/Microsoft.Sharepoint.Utilities.WebTemplateExtensions.ScriptUtility.CreateFormula", {info:{Title:"Contoso Travel - Legal Case Book", Description:"Restricted site design to create a legal case book site", ScriptGuids:["b432a1cd-7e1f-4fb1-9829-633d8MaG1C"], Targets:["CN=GUID,OU=GUID,OU=Tenants,OU=MSOnline,DC=<value>,DC=msoprd,DC=msft,DC=net"], IsDefault:true, WebTemplate:"68"}});
```

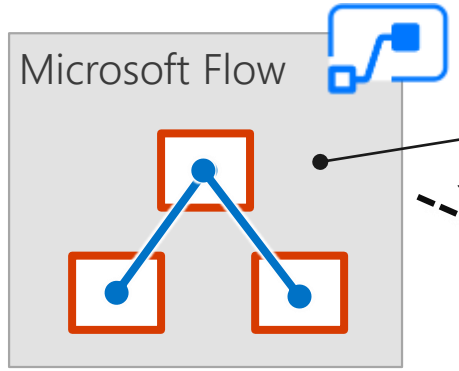
Site Script Actions

as of May, 2018

- Create lists, fields, views, custom formatters
- Create site columns, content types
- Create navigation links
- Apply a theme
- Set site logo
- Join a hub site
- Install an add-in or SPFx solution
- Configure regional settings
- Manage guest access
- Run a Flow

What if there's no site script action for what you need?





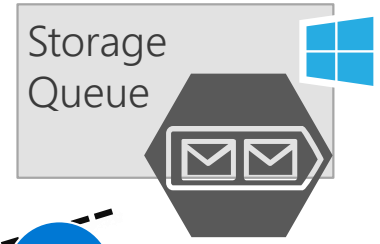
Script configured to start previously configured flow, which will receive the newly created site URL as parameter. Flow includes a step to pass site URL to Azure storage queue for delegating processing to Azure side

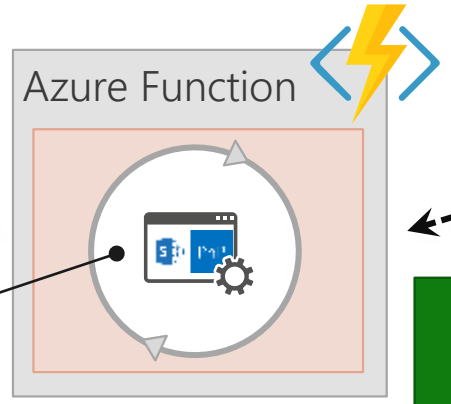
<<instantiate with URL details>>

1

<<add message with site URL>>

2





<<instantiate>>

3

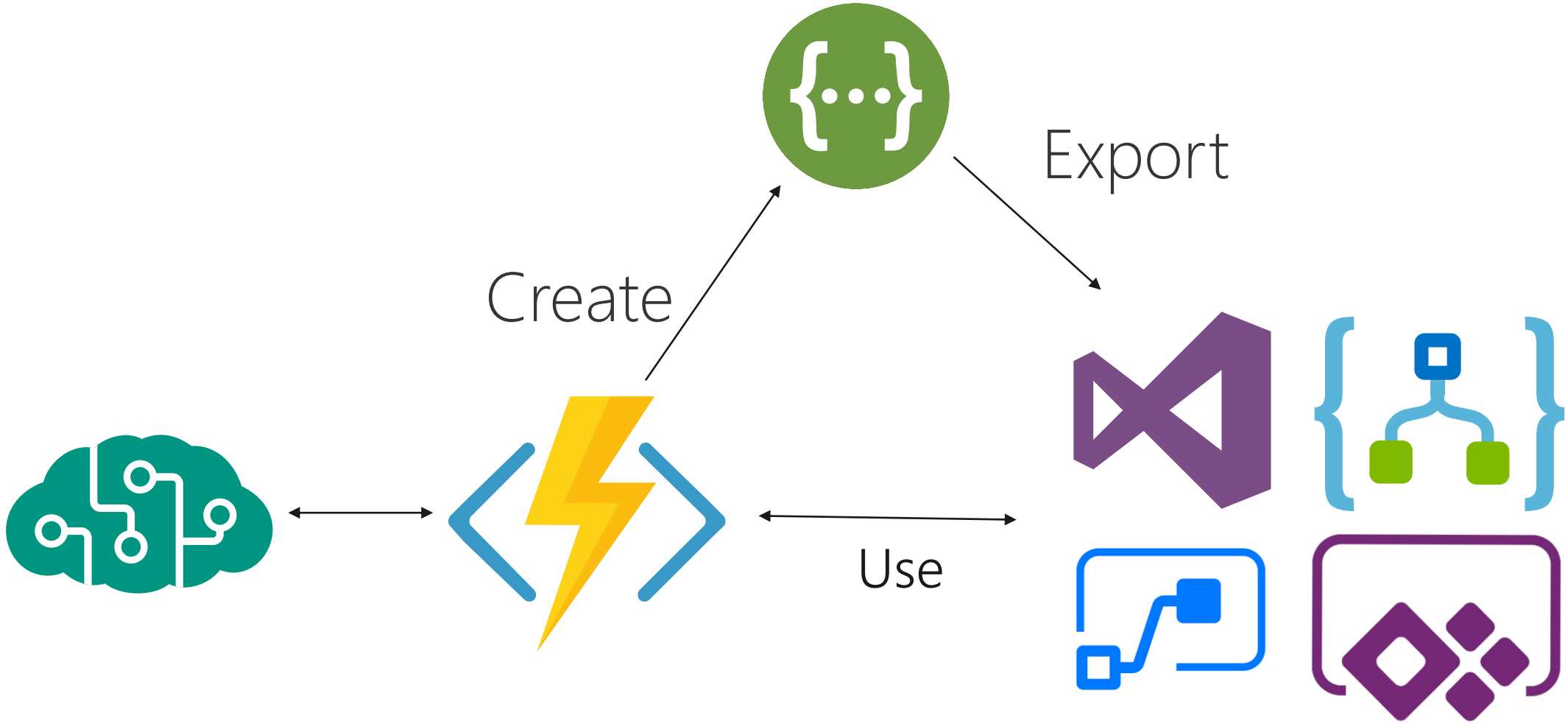
<<apply configurations>>

4

Business logic will handle the message processing. Site URL is used as a parameter to connect to site using remote APIs. Applies needed customization using remote provisioning techniques, for example with PnP remote provisioning template model.

DEMO:
Azure Function in PowerShell
Called from Site Script

Invoke Azure Functions from PowerApps and Flow



Introduction to Custom APIs

- Connect Flow, PowerApps, Logic Apps to Azure Functions (or any API)
- Express API in Swagger format – or
  - Describe via connector UI
  - Import from Postman
- Authentication options
  - No authentication
  - Basic authentication
  - OAuth 2.0 Authentication (AAD, Facebook, Google)
  - API Key

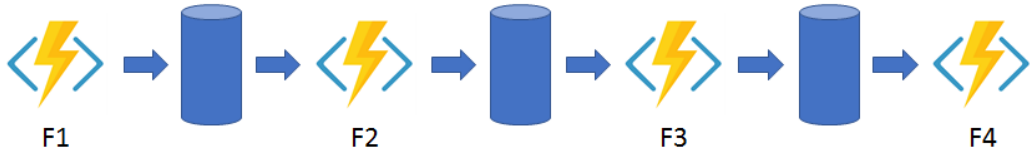
DEMO:
Connect to Azure Function from Flow and Logic Apps

Coming soon …

- Durable Functions
- Azure Functions 2.0

Durable Functions

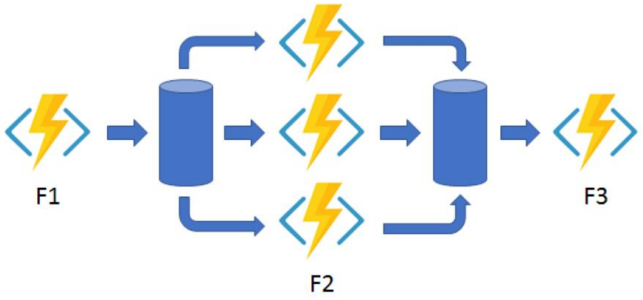
Stateful functions – workflow in code



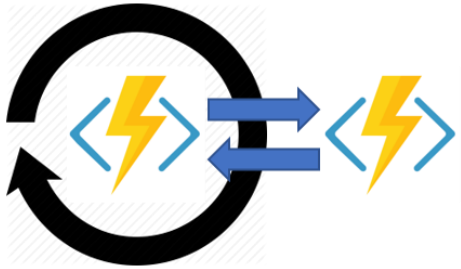
Function Chaining

Long running transaction

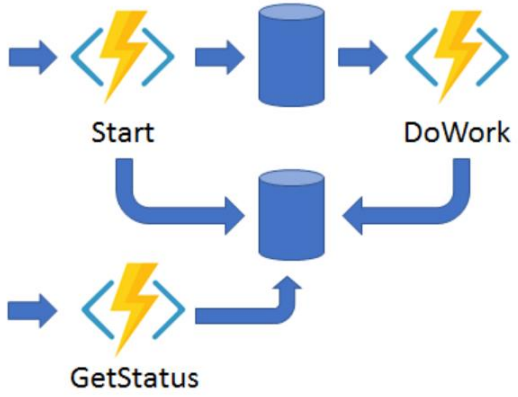
```
public static async Task<object> Run(DurableOrchestrationContext ctx)
{
    try
    {
        var x = await ctx.CallActivityAsync<object>("F1");
        var y = await ctx.CallActivityAsync<object>("F2", x);
        var z = await ctx.CallActivityAsync<object>("F3", y);
        return  await ctx.CallActivityAsync<object>("F4", z);
    }
    catch (Exception)
    {
        // error handling/compensation goes here
    }
}
```



Fan Out / Fan In



Monitoring



Async HTTP APIs

In preview … Functions v2

Version 1

- Generally available
- .NET Core
- Develop and run on Windows only

Version 2

- Preview
- .NET Core
- Develop and run on all Windows, macOS, Linux
- Binding extensibility
- Language extensibility

DEMO:
v2 Function with Graph AuthN extension

Microsoft

http://bit.ly/AzureSPC-FuncJS
http://bit.ly/AzureSPC-FuncCS
http://bit.ly/AzureSPC-FuncSPFx

Mastering Azure Functions

Bob German
Partner Technical Architect

ocp | One Commercial Partner