EUROPEAN COLLABORATION

SUMMIT 2024

Durable Azure functions using PnP.PowerShell cmdlets awesomeness!

Rodrigo Pinto

ENTERPRISE ARCHITECT

PERSPECTIVE DRAGON

M365 MVP

EUROPEAN COLLABORATION SUMMIT

Microsoft run.events

AURUM AvePoint CoreView dox42

EasyLife 365 resco veeam

adesso business. people. technology. Allied Global ASCENT BCC DE CIX devoteam

empowerID FPT Software glueck kanja Jabra GN kaspersky LightningTools

nintex Rencore ShareGate: Spot by NetApp SysCloud Syskit WEBCON LOW-CODE, BUT BETTER.

Use ECS Coins for Swag!

Top 3 win an Atari 2600+

1. Get the app
2. Visit sessions and sponsors, rate sessions
3. Earn ECS Coins
4. Spend ECS Coins

csmmt.eu/app

run events

#evangelize

Microsoft SharePoint

#expertise

#community

#Software Engineering (>20 years)

#M365 Enterprise Architect

#Speaker (TechDays, European SharePoint Conference, SharePointNA, European Collaboration Summit, CollabDaysLisbon)

RODRIGO PINTO

#M365 MVP

@scoutmanpt

www.scoutman.pt

rpinto@pdragon.co

perspective dragon · Microsoft · sharept · collabdays | lisbon · MVP Microsoft Most Valuable Professional · SUPER DAD awarded since 2008 old JEDI saying…

POWERSHELL CONSULTANTS

… are natural born communicators …

Who already tried …

- AZURE FUNCTIONS
  C# ? PowerShell?
- AZURE DURABLE FUNCTIONS
  C# ? PowerShell?

OVERVIEW

OVERVIEW

Events

Code

Outputs

React to timers, HTTP, or events from Azure services

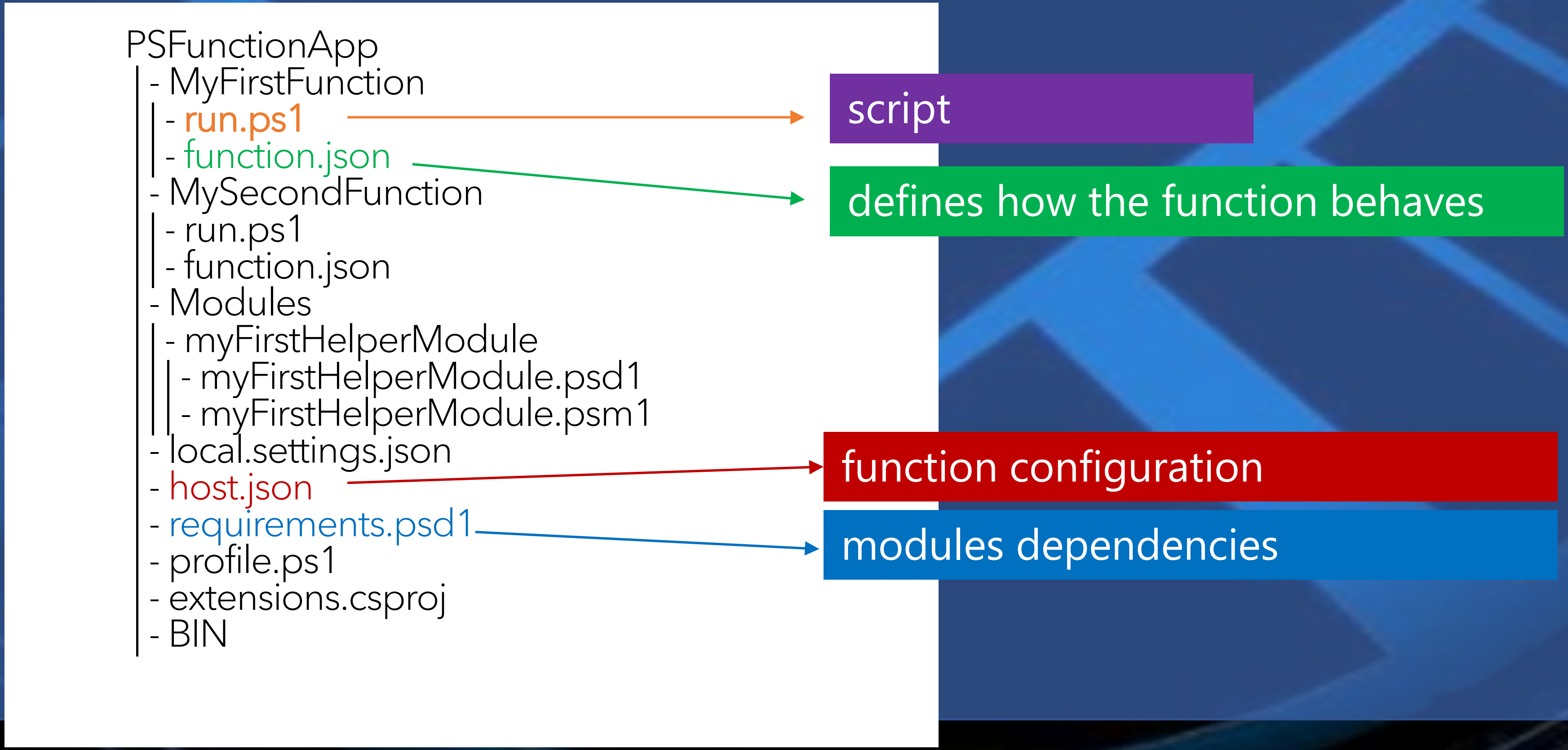
Author functions in C#, F#, Node.JS, Java, PowerShell and more

Send results to an ever-growing collection of services

OVERVIEW

PowerShell Azure function

PowerShell script executed when triggered

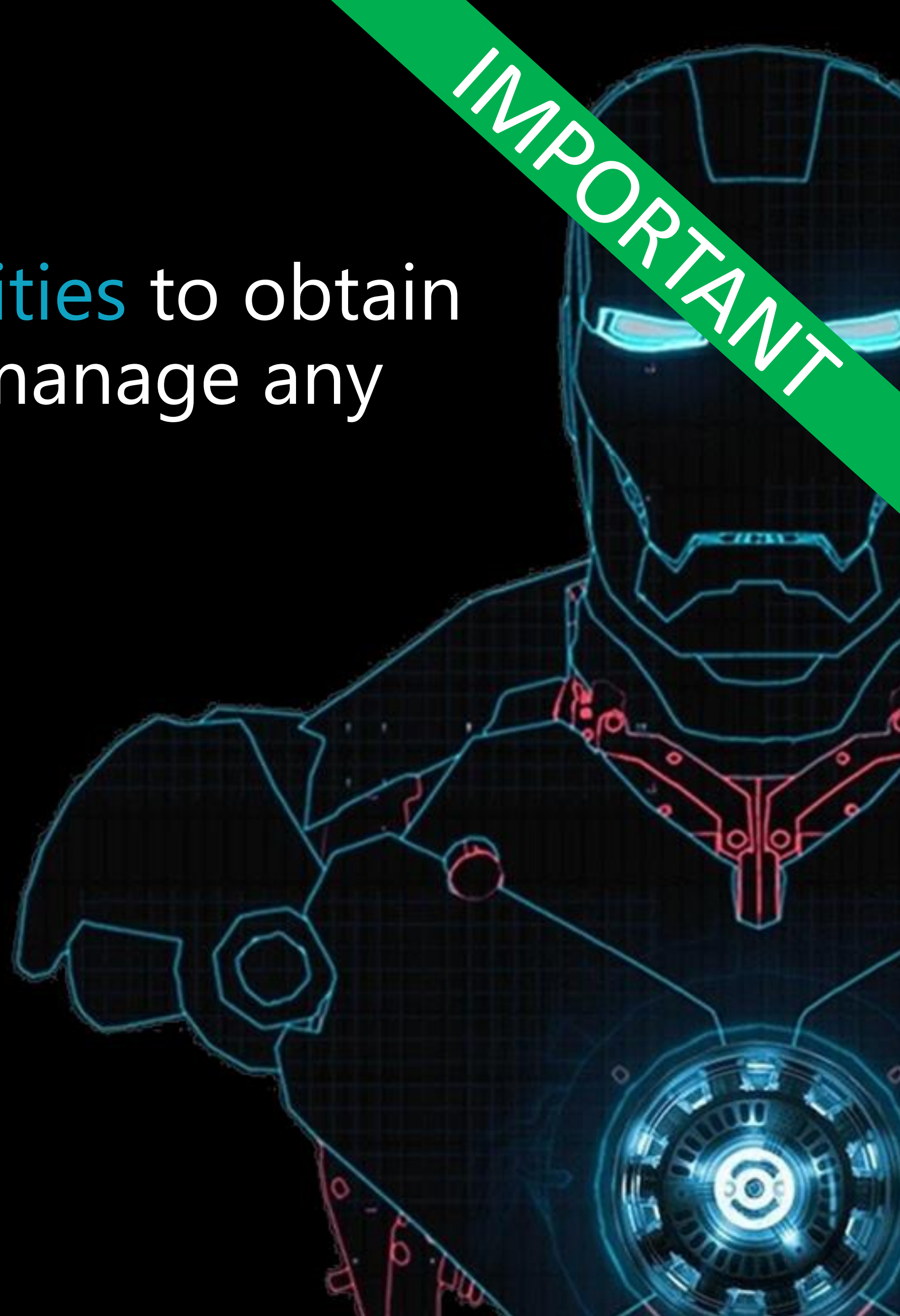


```
PSFunctionApp
 | - MyFirstFunction
 | | - run.ps1
 | | - function.json
 | - MySecondFunction
 | | - run.ps1
 | | - function.json
 | - Modules
 | | - myFirstHelperModule
 | | | - myFirstHelperModule.psd1
 | | | - myFirstHelperModule.psm1
 | - local.settings.json
 | - host.json
 | - requirements.psd1
 | - profile.ps1
 | - extensions.csproj
 | - BIN
```

Managed Identities

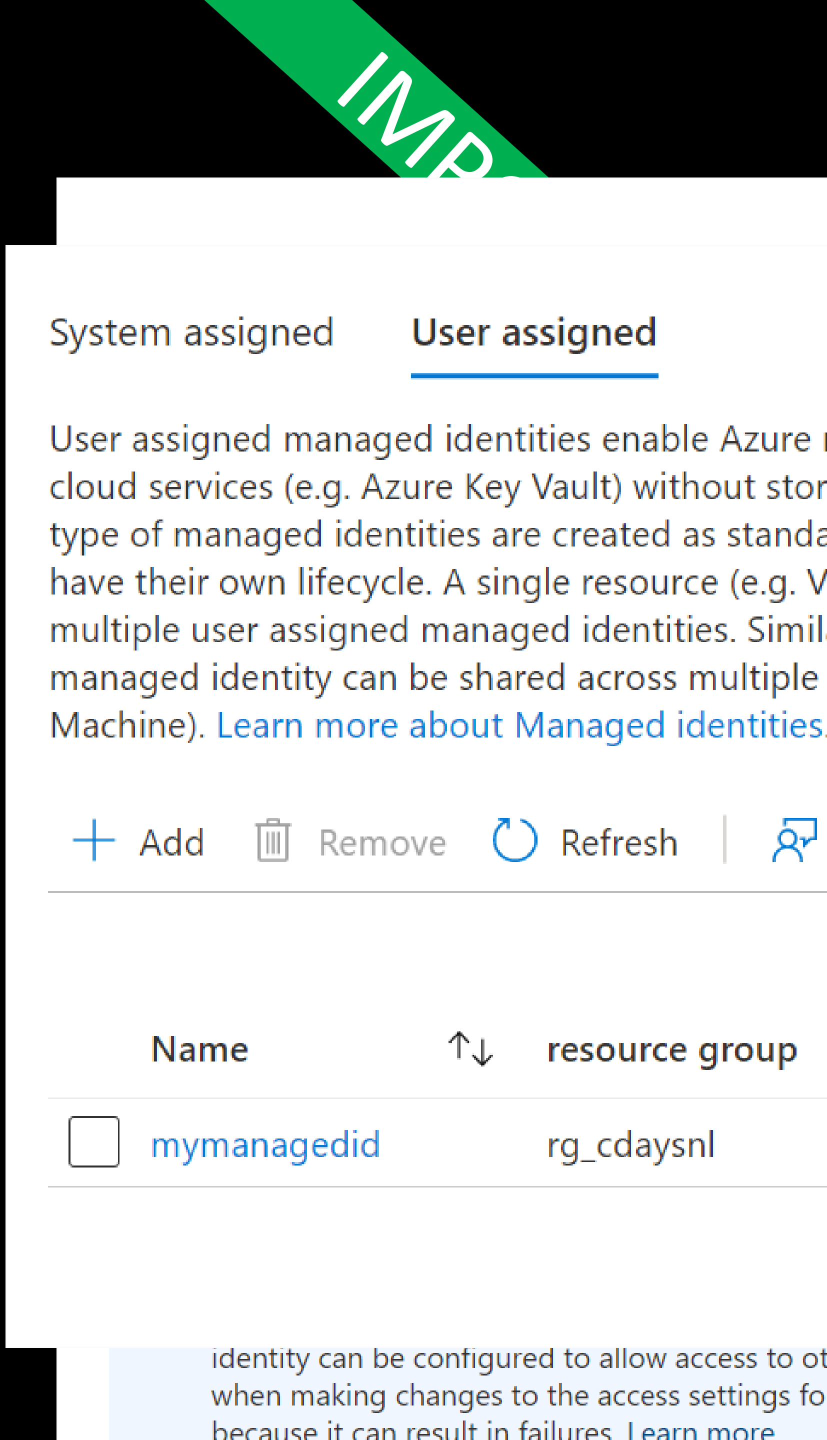
- Applications can use managed identities to obtain Azure AD tokens without having to manage any credentials.

  - System-assigned.
  - User-assigned.



Managed Identities

- System-assigned.
  - Enabling a system-assigned managed identity, an identity is automatically created in Azure AD.
  - The identity is tied to the lifecycle of that service instance.
  - When the resource is deleted, Azure automatically deletes the identity for you.
- User-assigned.
  - Manual managed identity
  - Can be assigned it to one or more instances of an Azure service. Identity is managed separately from the resources that use it.



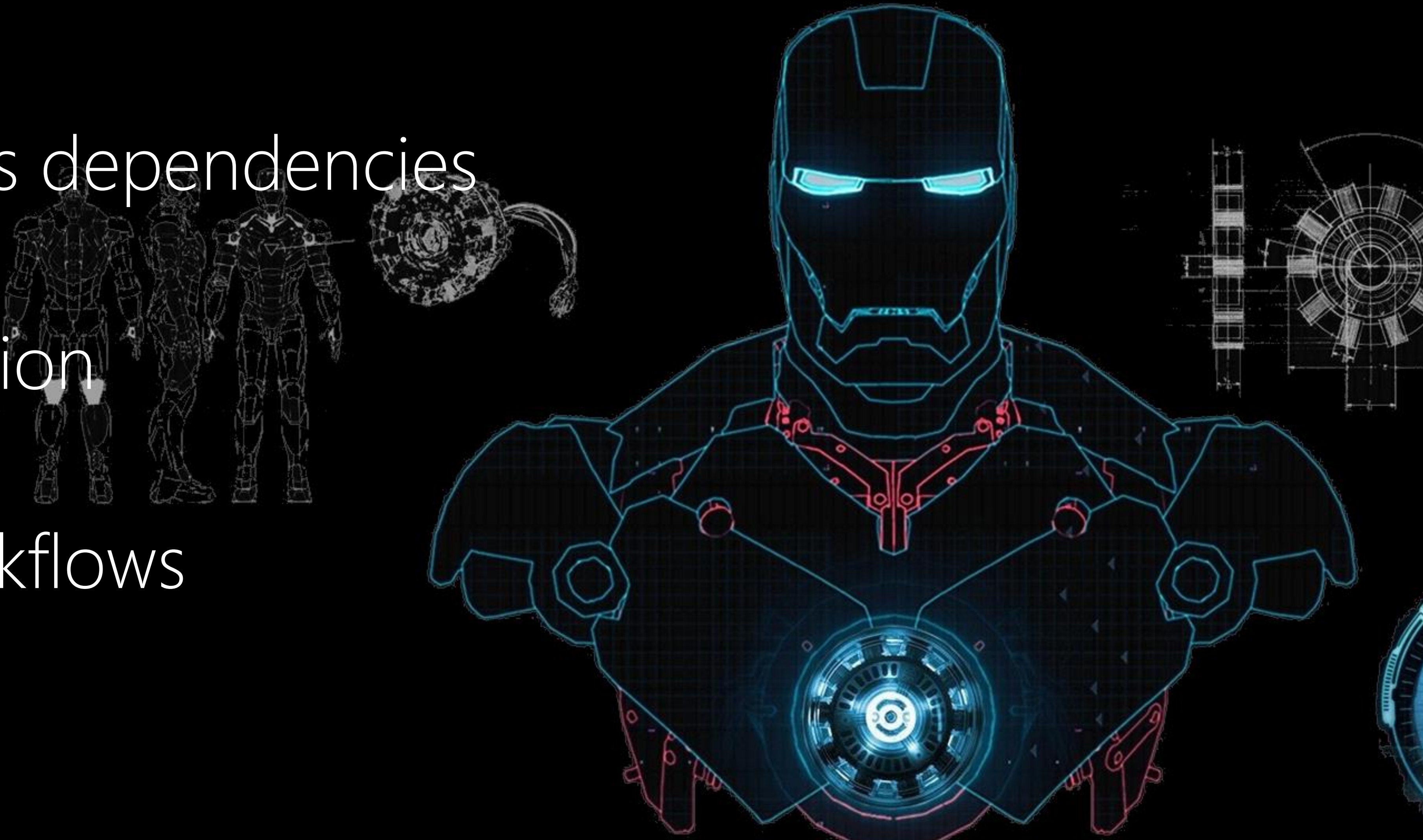
User assigned managed identities enable Azure
cloud services (e.g. Azure Key Vault) without stor
type of managed identities are created as standa
have their own lifecycle. A single resource (e.g. V
multiple user assigned managed identities. Simil
managed identity can be shared across multiple
Machine). Learn more about Managed identities

Name	resource group
mymanagedid	rg_cdaysnl

identity can be configured to allow access to ot
when making changes to the access settings fo
because it can result in failures. Learn more

Some challenges with serverless …

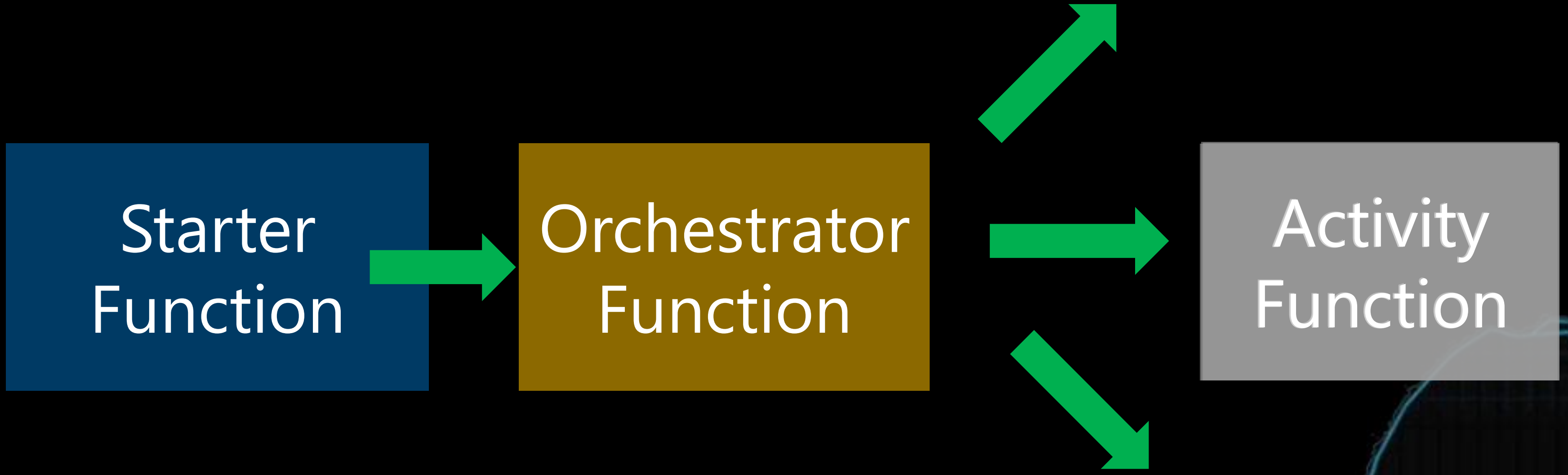
- No state
- Asynchronous dependencies
- No state
- Shorter duration
- No state
- Complex workflows
- No state



Durable Functions

Stateful workflows in a serverless compute environment that will simplify complex, stateful coordination requirements

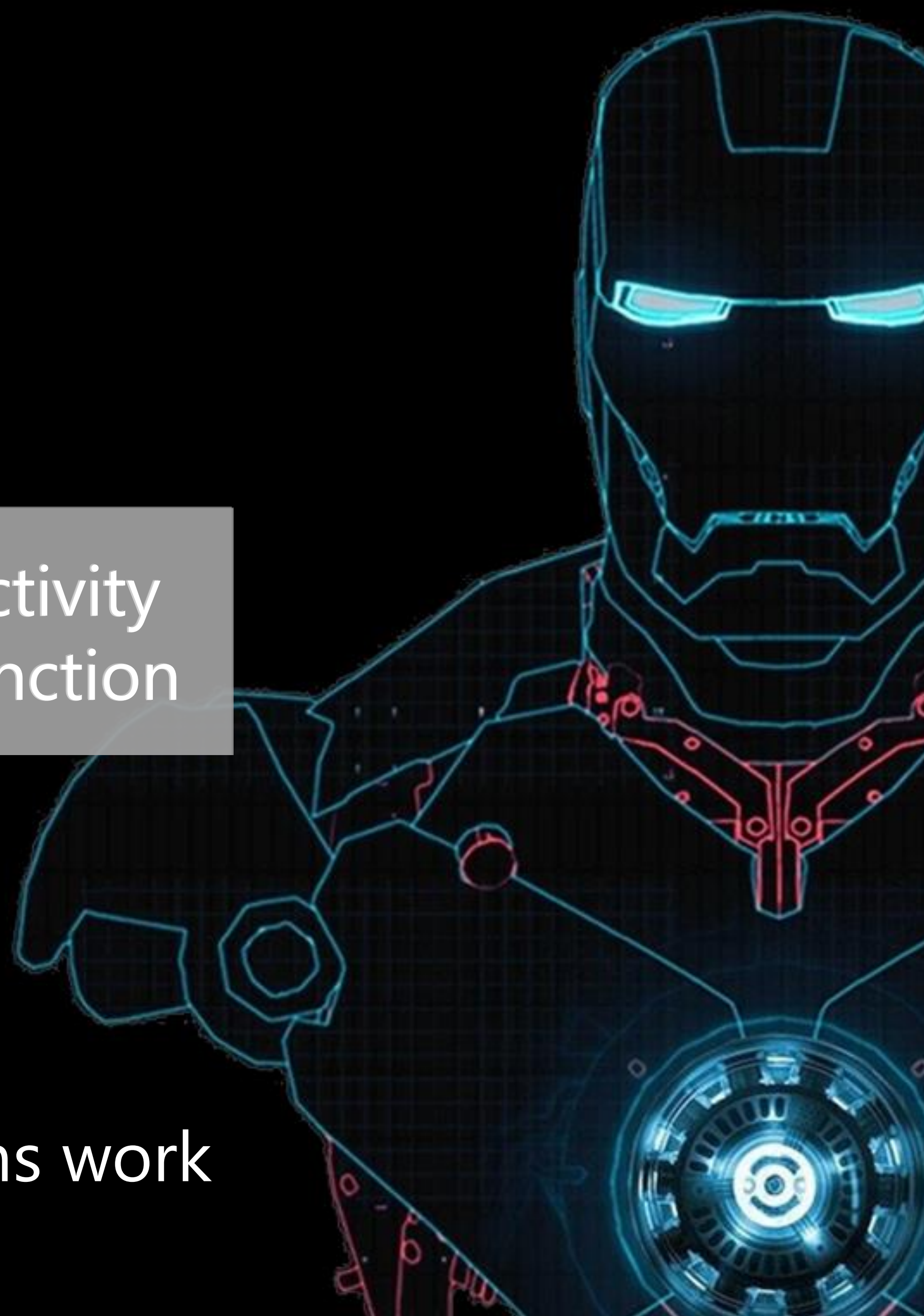
How does it work ?



Starts orchestrations

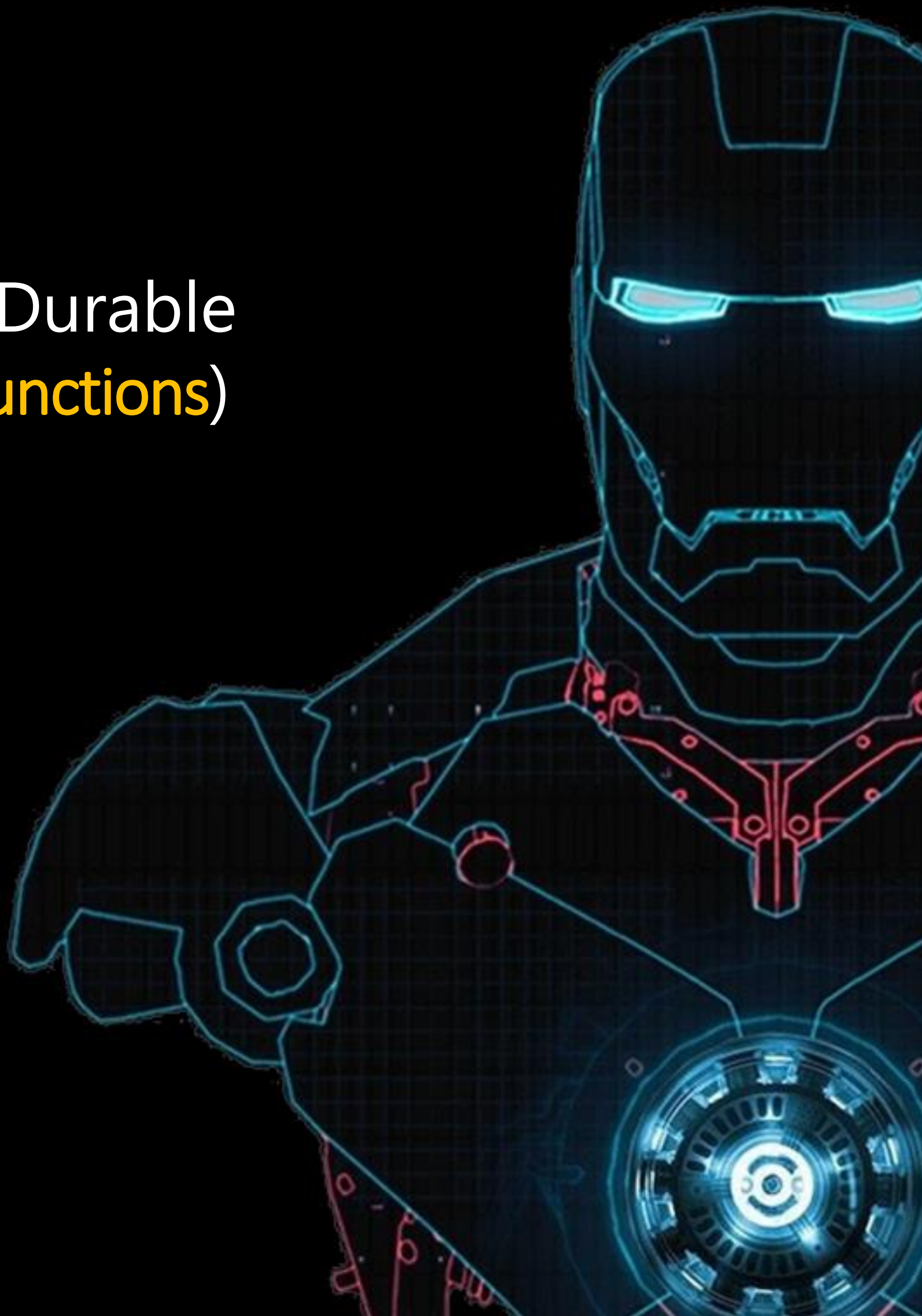
Coordinates activities

Performs work



Durable Functions

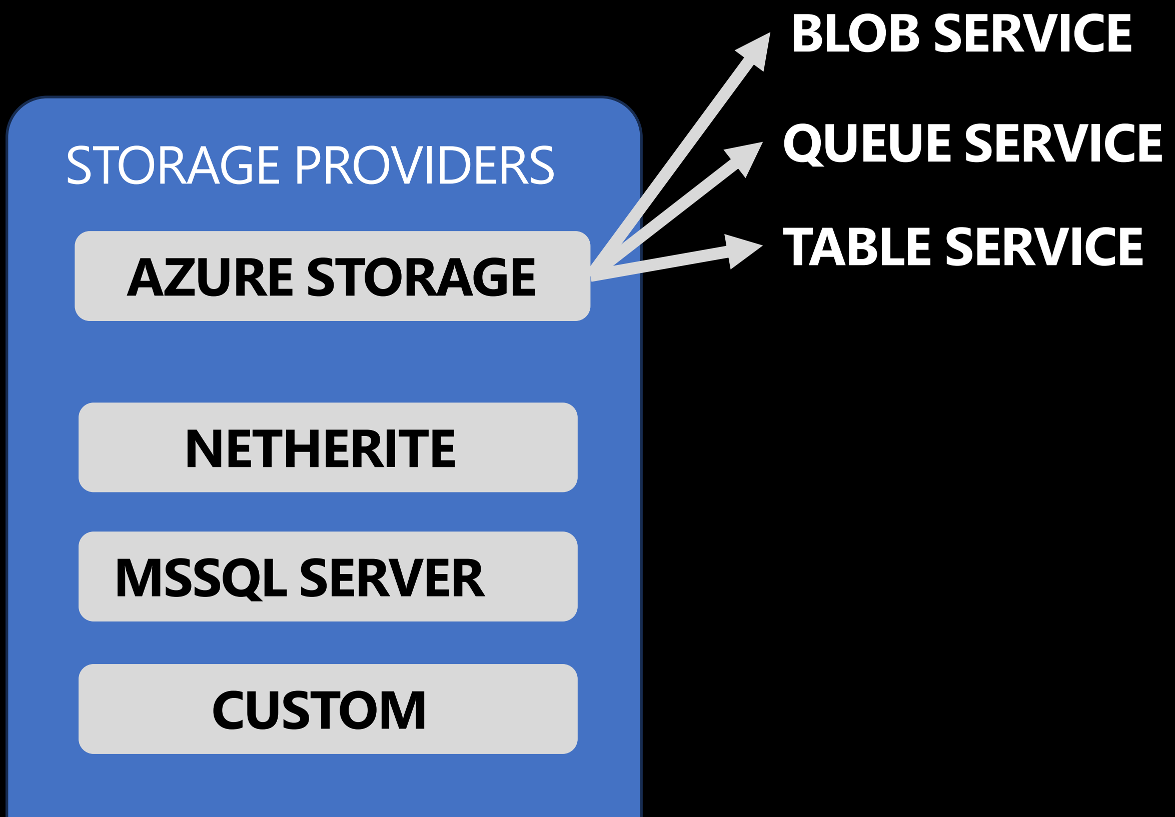
- Decompose sequential workflow into a Durable Functions orchestration (multiple shorter functions)
- Orchestration :
  - Duration : hours or longer
  - Retries, custom error handling
  - CheckPoints

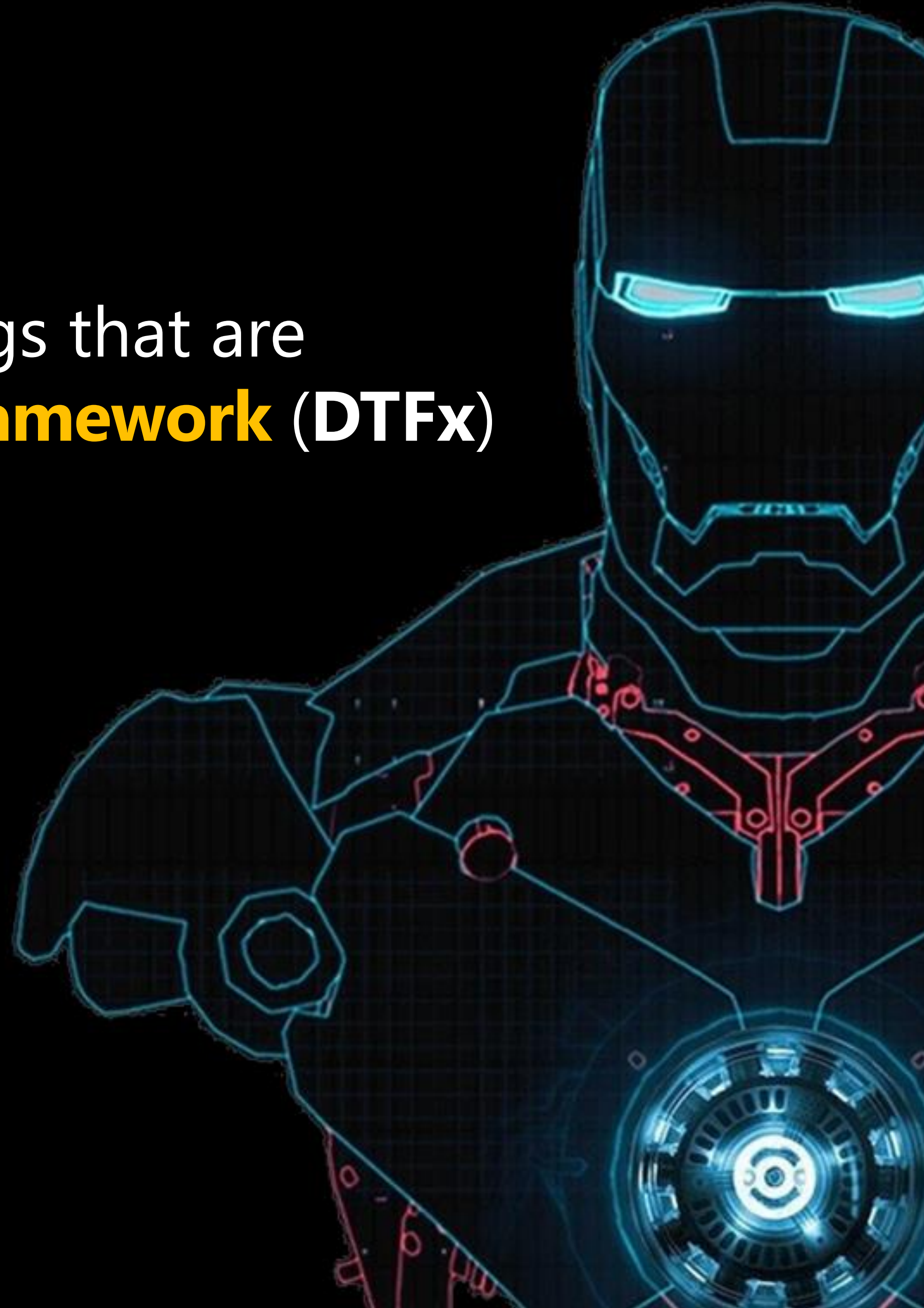


Durable Functions

What's beneath ....

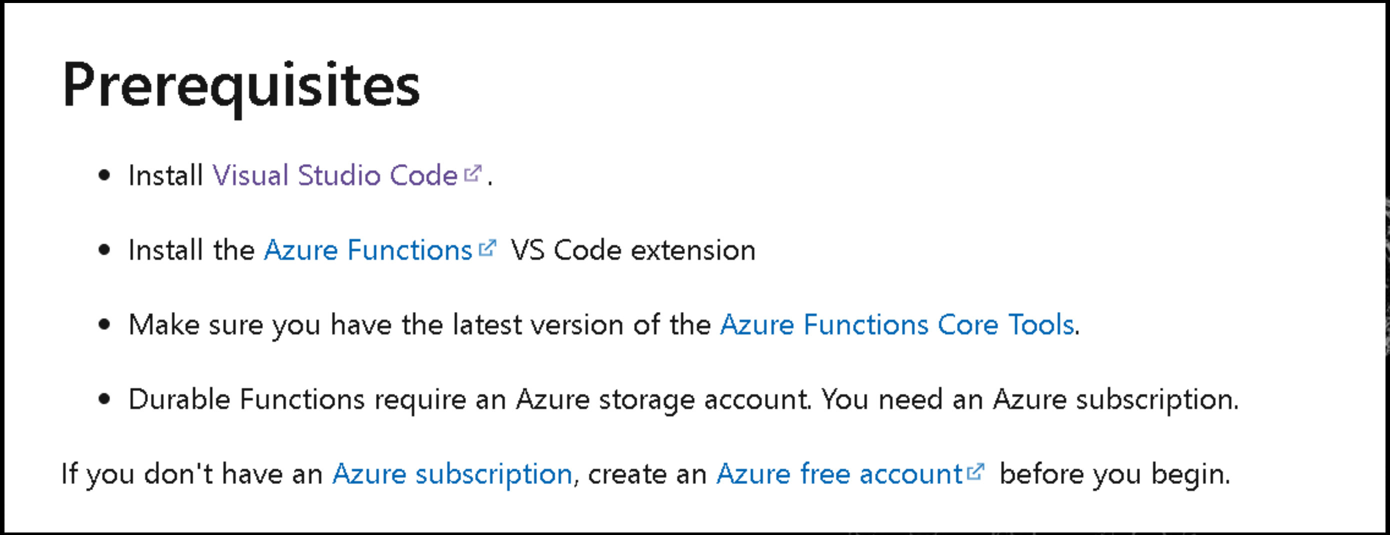
- Set of Azure Functions triggers and bindings that are internally powered by the **Durable Task Framework** (**DTFx**)





PREREQUISITES

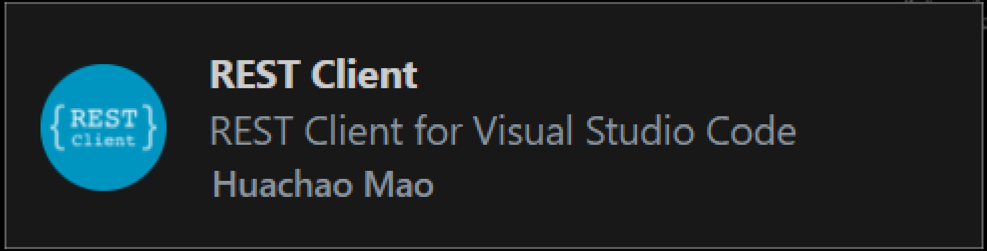
PREREQUISITES



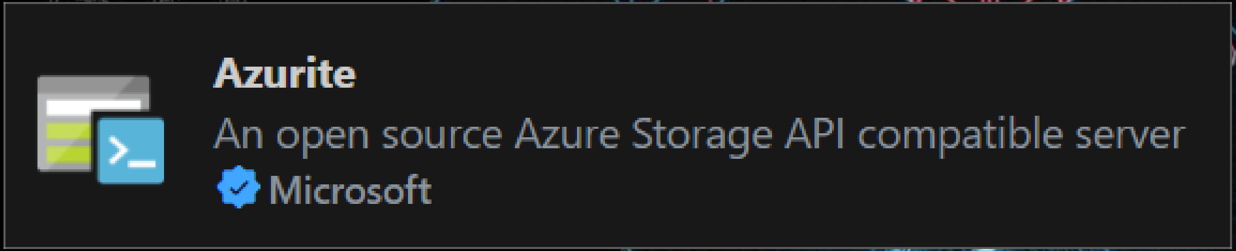
Prerequisites

- Install Visual Studio Code.
- Install the Azure Functions VS Code extension
- Make sure you have the latest version of the Azure Functions Core Tools.
- Durable Functions require an Azure storage account. You need an Azure subscription.

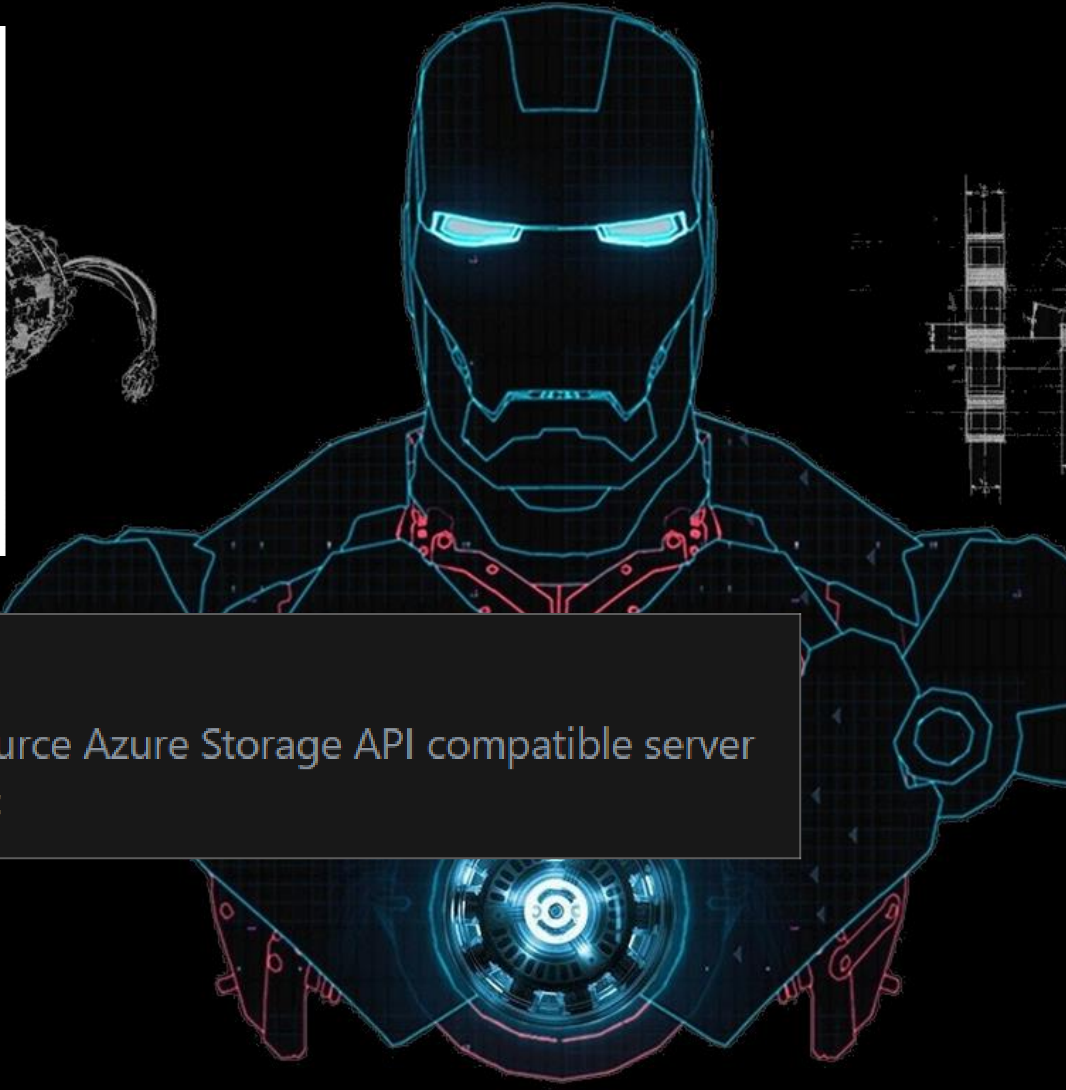
If you don't have an Azure subscription, create an Azure free account before you begin.



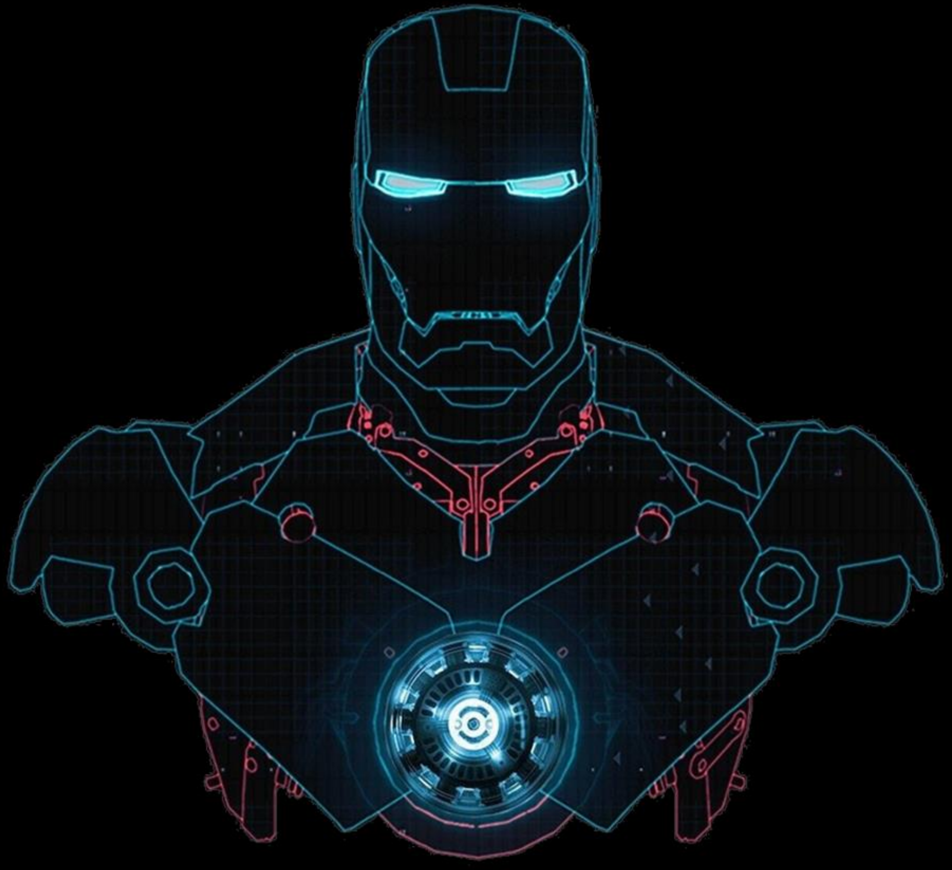
REST Client
REST Client for Visual Studio Code
Huachao Mao



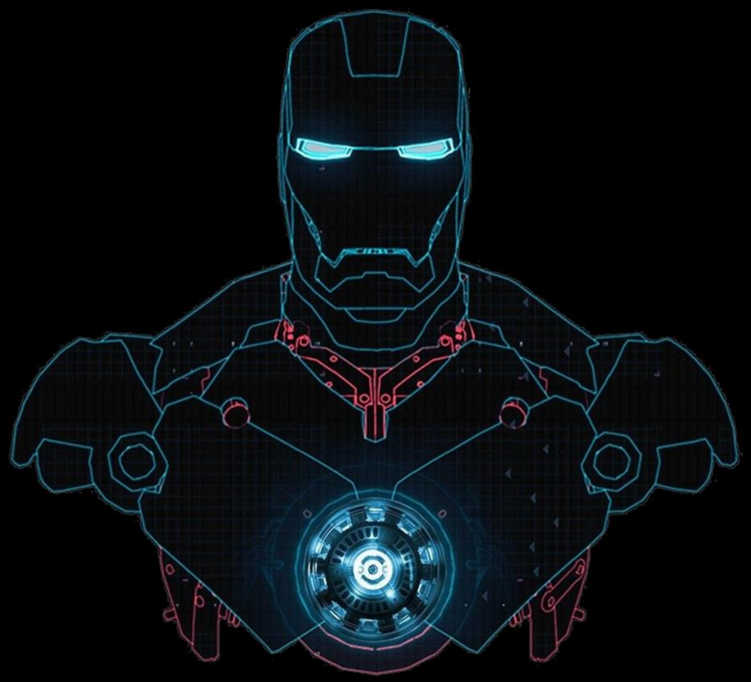
Azurite
An open source Azure Storage API compatible server
Microsoft



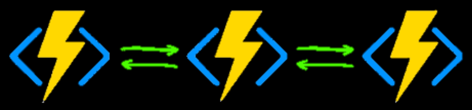
TOOLING DEMO



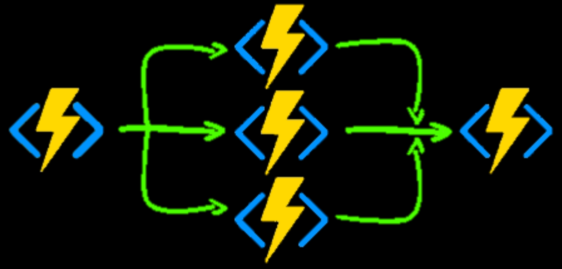
Patterns



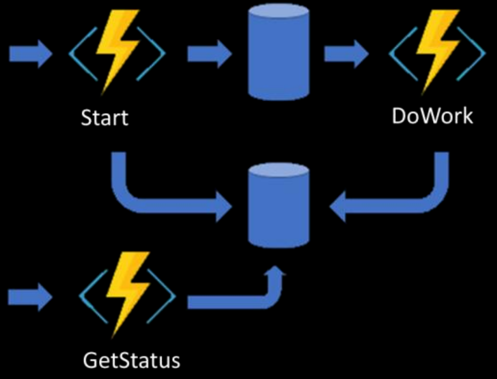
Patterns



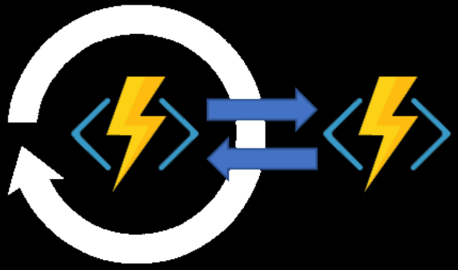
Function chaining



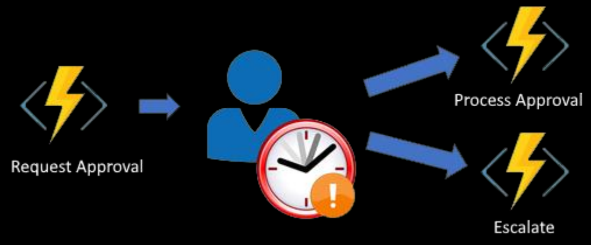
Fan-out \ Fan-In



Async HTTP APIs



Monitoring



Human interaction

Aggregator

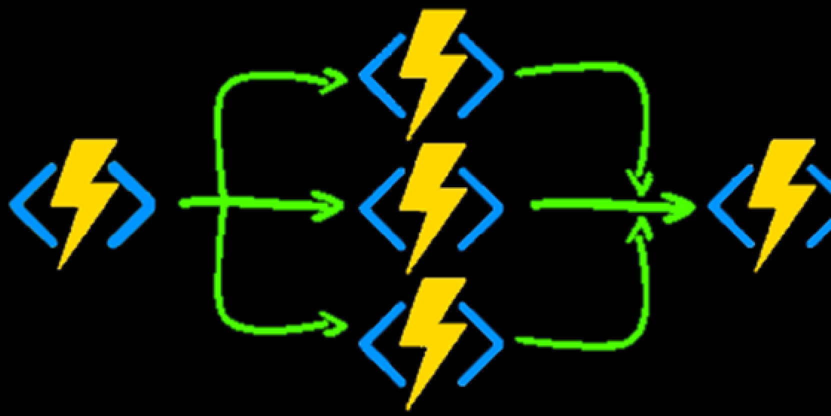
Durable Functions

Function Chaining

- Execute functions in a specific order
- Output of one function can be the input of the next one

```
param($Context)

$site = Invoke-ActivityFunction -FunctionName ‘CreateHRSite’
$DeployAssets = Invoke-ActivityFunction -FunctionName ‘DeployAssets’ -Input $site

Invoke-ActivityFunction -FunctionName ‘ApplyConfiguration’ -Input $DeployAssets
Invoke-ActivityFunction -FunctionName ‘AddApps’ -Input $site
Invoke-ActivityFunction -FunctionName ‘SetSensitiveLabels’ -Input $site
```

Durable Functions

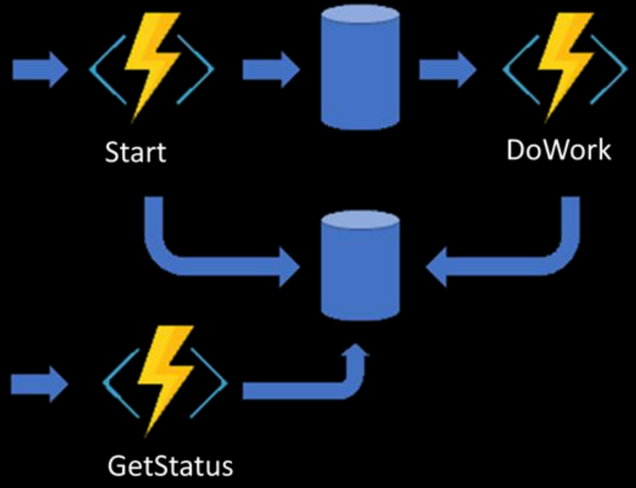


Fan-out/Fan-in

- Execute multiple functions in parallel (asynchronously)
- Wait for each function to ends and then continue with the workflow.

```
param($Context)

# Get a list of work items to process in parallel.
$WorkBatch = Invoke-ActivityFunction -FunctionName ‘GetO365Groups'

# Fan out
$ParallelTasks =
    foreach ($WorkItem in $WorkBatch) {
        Invoke-ActivityFunction -FunctionName 'ProcessListItem' -Input $WorkItem -NoWait
    }

$Outputs = Wait-ActivityFunction -Task $ParallelTasks
# Fan in
Invoke-ActivityFunction -FunctionName 'AggregateResults' -Input $Outputs
```

Durable Functions



Async HTTP APIs

- Workflow composed of long-running operations
- As soon as the client function ends its execution, an endpoint is provided through which the status of the execution can be checked

```
param($Context)

# Get a list of work items to process in parallel.
$WorkBatch = Invoke-ActivityFunction -FunctionName ‘GetO365Groups'

# Fan out
$ParallelTasks =
    foreach ($WorkItem in $WorkBatch) {
        Invoke-ActivityFunction -FunctionName 'ProcessListItem' -Input $WorkItem -NoWait
    }

$Outputs = Wait-ActivityFunction -Task $ParallelTasks
# Fan in
Invoke-ActivityFunction -FunctionName 'AggregateResults' -Input $Outputs
```

Durable Functions

Monitor

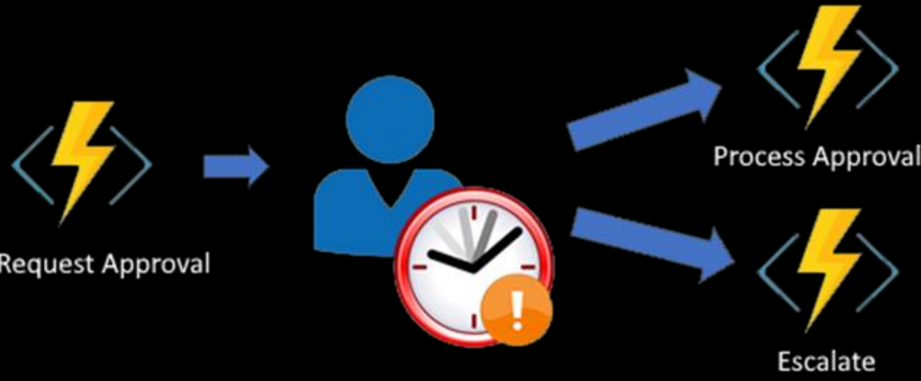
- Through a timer, you can define a polling rate and check at regular intervals when a specific condition is met.

```
param($Context)
# set polling interval + expiry time

$pollingInterval = New-TimeSpan -Seconds $Context.Input.PollingInterval.Value
$expiryTime = $Context.Input.ExpiryTime.Value
# execute while waiting
while ($Context.CurrentUtcDateTime -lt $expiryTime) {

    $jobStatus = Invoke-DurableActivity -FunctionName ‘GetJobStatus’
    if ($jobStatus -eq "Completed") {
            # Perform an action when a condition is met.
            $output += Invoke-DurableActivity -FunctionName ‘SendAlert’
            break
        }
}
    Start-DurableTimer -Duration $pollingInterval
```

Durable Functions

Human Interaction

- Asking the user for a challenge to approve the execution and continue the workflow.



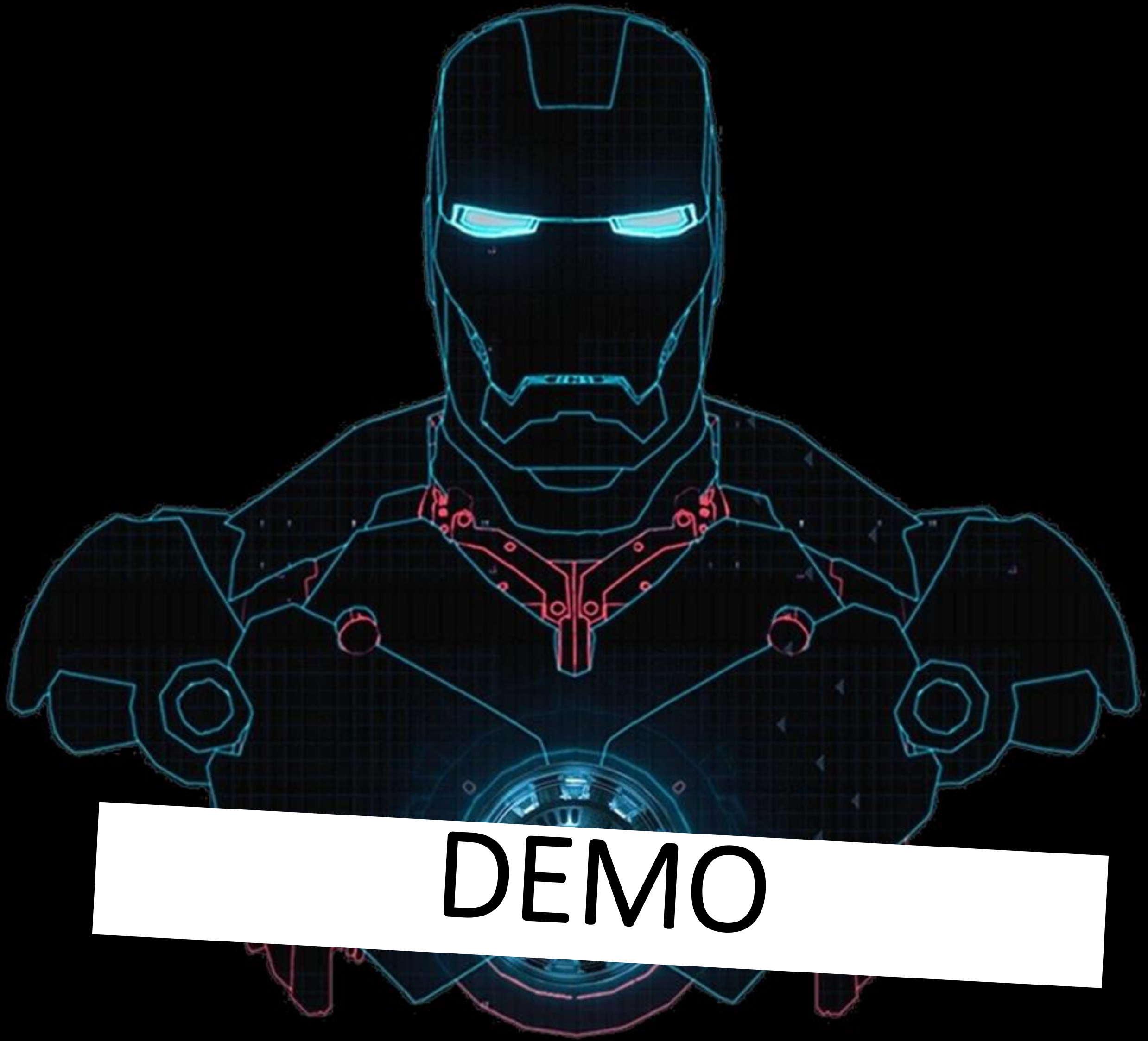
```
param($Context)
# set polling interval + expiry time
$duration = New-TimeSpan -Seconds $Context.Input.Duration
$managerId = $Context.Input.ManagerId
$skipManagerId = $Context.Input.SkipManagerId

$output += Invoke-DurableActivity -FunctionName "RequestApproval" -Input $managerId

$durableTimeoutEvent = Start-DurableTimer -Duration $duration -NoWait
$approvalEvent = Start-DurableExternalEventListener -EventName "ApprovalEvent" -NoWait

$firstEvent = Wait-DurableTask -Task @( $approvalEvent, $durableTimeoutEvent , $declineEvent) –Any

if ( $approvalEvent -eq $firstEvent)  {
    Stop-DurableTimerTask -Task $durableTimeoutEvent
    $output += Invoke-DurableActivity -FunctionName "ProcessApproval“ }
else {
    $output += Invoke-DurableActivity -FunctionName "EscalateApproval" -Input $skipManagerId
```

Durable Functions

Aggregator

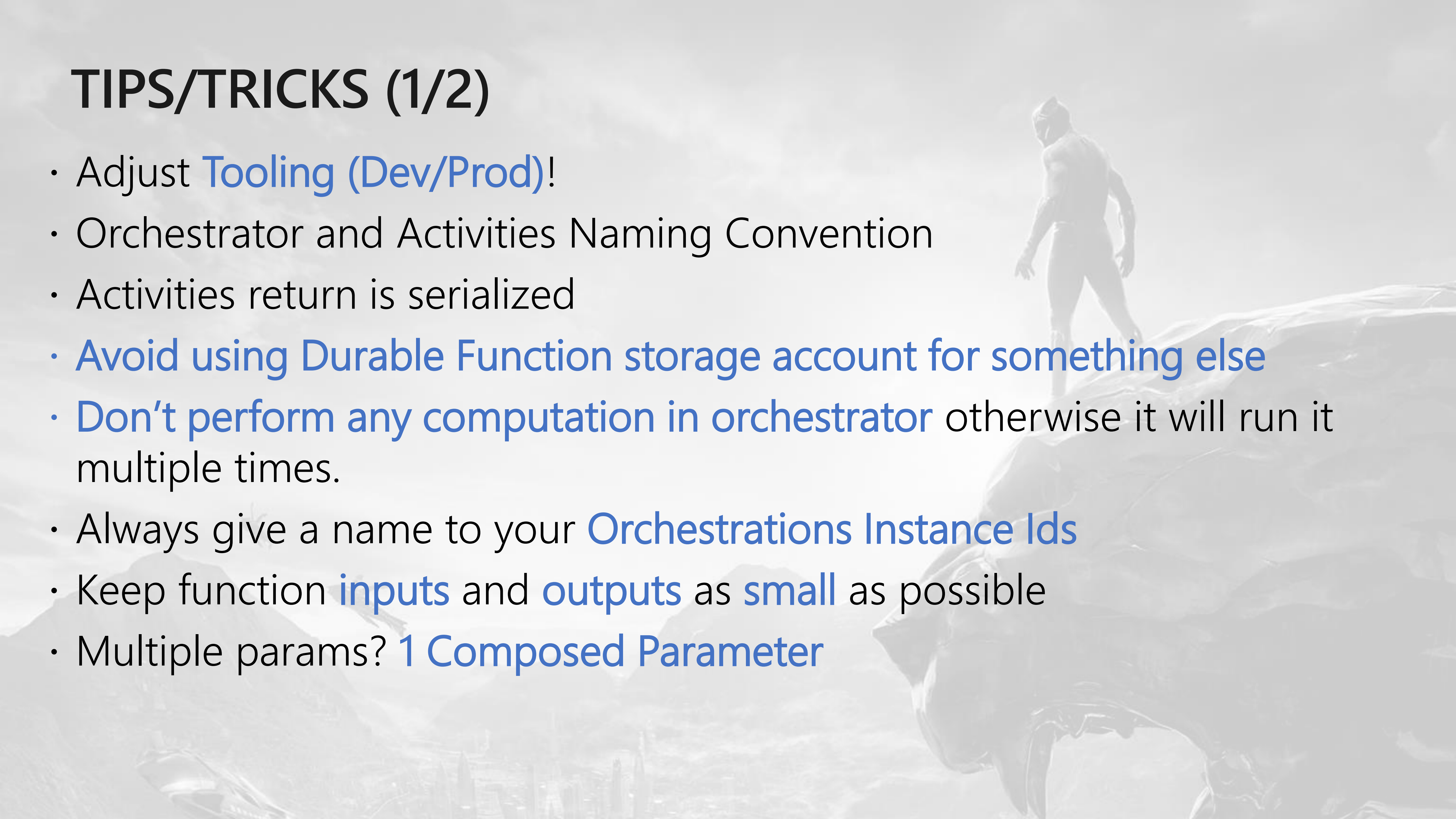
- Query ggregate values

```
param($Context)

# Get a list of work items to process in parallel.
$WorkBatch = Invoke-ActivityFunction -FunctionName ‘GetO365Groups'

# Fan out
$ParallelTasks =
    foreach ($WorkItem in $WorkBatch) {
        Invoke-ActivityFunction -FunctionName 'ProcessListItem' -Input $WorkItem -NoWait
    }
$Outputs = Wait-ActivityFunction -Task $ParallelTasks

# Fan in
Invoke-ActivityFunction -FunctionName 'AggregateResults' -Input $Outputs
```



DEMO

TIPS & TRICKS



TIPS/TRICKS (1/2)

- Adjust Tooling (Dev/Prod)!
- Orchestrator and Activities Naming Convention
- Activities return is serialized
- Avoid using Durable Function storage account for something else
- Don't perform any computation in orchestrator otherwise it will run it multiple times.
- Always give a name to your Orchestrations Instance Ids
- Keep function inputs and outputs as small as possible
- Multiple params? 1 Composed Parameter

TIPS/TRICKS (2/2)

- Think Microservices !
- Deployment ! (#bicep)
- Lock your PnP.PowerShell Version
- Use $env:PNPPOWERSHELL_UPDATECHECK="false"
- When possible, use managed identities

GITHUB LINK WITH ALL CODE

https://bit.ly/azdrlfzpws

THANK YOU,
YOU ARE AWESOME ❤️

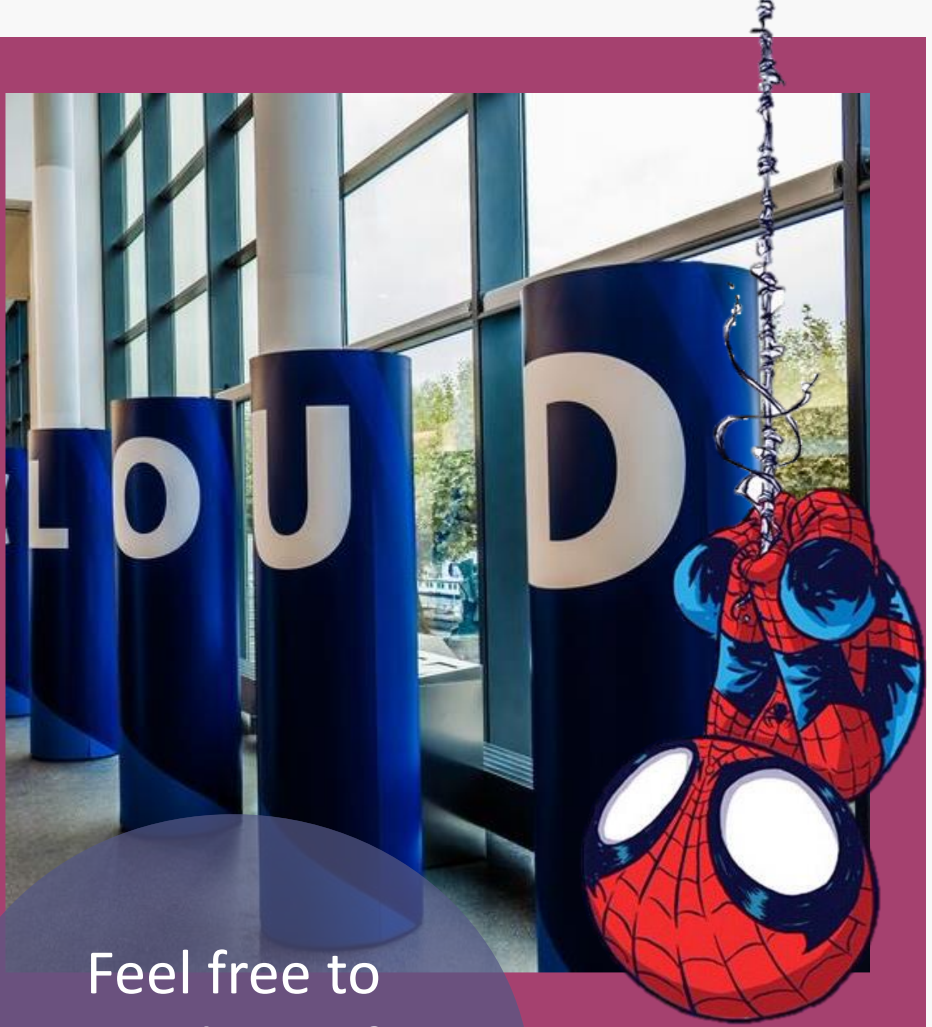
PLEASE RATE THIS SESSION IN THE MOBILE APP.

@scoutmanpt

www.scoutman.pt

rpinto@pdragon.co

PERSPECTIVE DRAGON



Feel free to reach me if you need any help ! 🤠