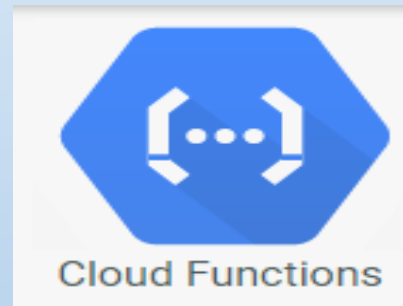
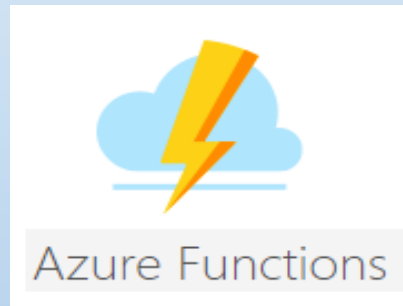


Server Less Computing



Agenda

- Overview of Server Less Computing
- Why Server Less Computing?
- Available Platforms
- Azure Functions and Azure Cosmos Synergy
- Durable Functions Overview
- Demo using Azure – Kudu, Visual Studio, CLI
- Gotchas and Limitations with Server Less

My Intro

- Baskar Rao
- Senior .Net Consultant with Compunnel Software Group.
- @baskarmib
- <https://www.linkedin.com/in/baskarrao-dandlamudi>
- baskarrao.dandlamudi@outlook.com
- www.compunnel.com
- <https://github.com/baskar3078/testazurefunctions>



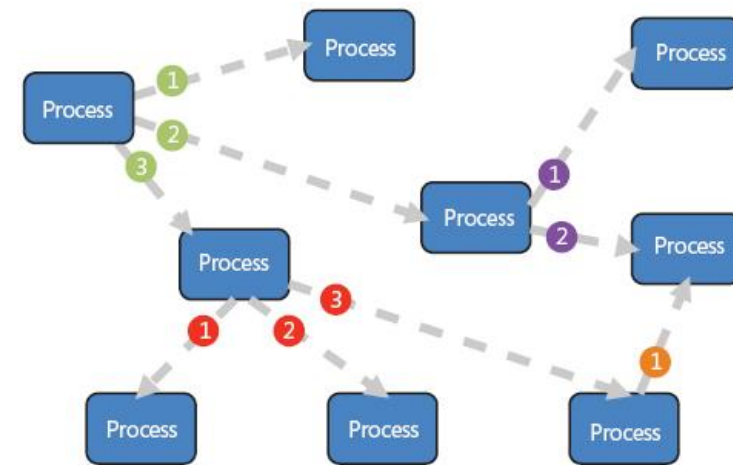
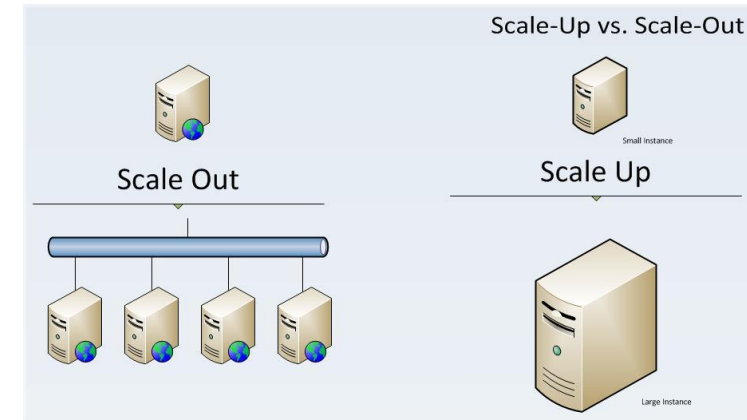
Server Less Computing - Overview

- Does not mean No Servers.
- Enables developers to develop and execute code with out server provisioning.
- “Function as Service” is mainly used to develop event driven applications or perform recurring actions with easier configuration.

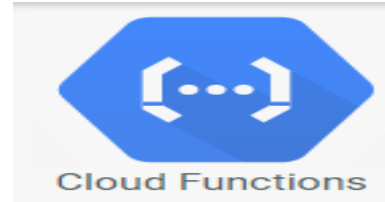


Why Server Less Computing ?

- Enables developers to focus on the functionality with out worrying on performance by automatically scaling up or down based on demand.
- Allows to pay for only the execution time along with costs for other resources like storage, network etc..
- Azure Functions, AWS Lambda, Google Cloud Functions , IBM Cloud Functions are different variations of function as service.

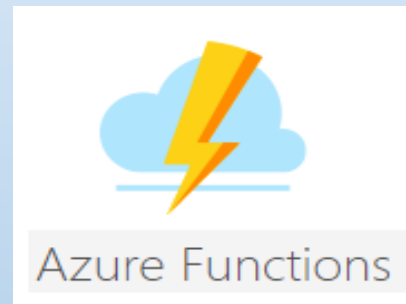


Platform Comparison



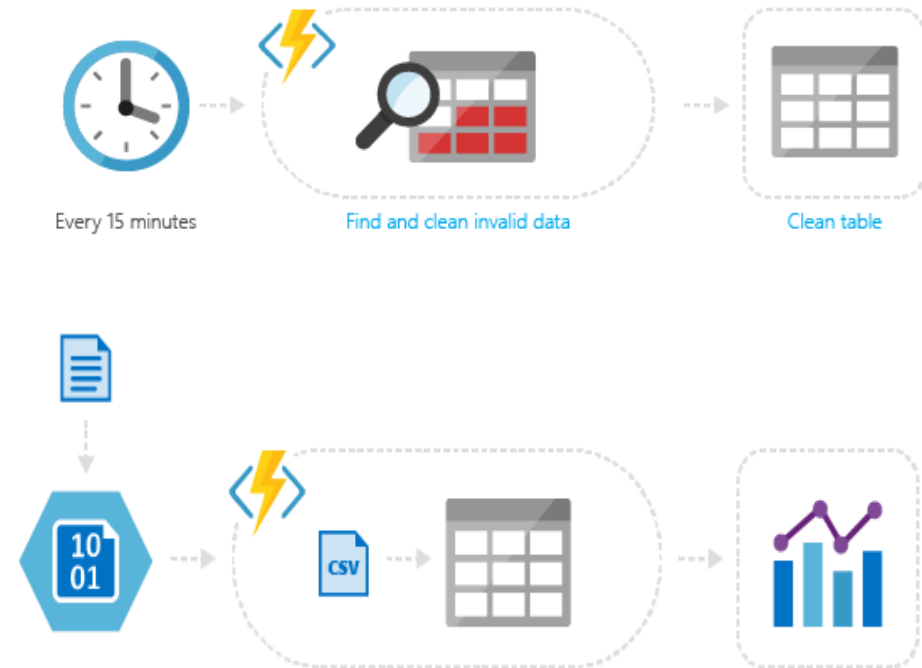
	AWS Lambda	Azure Functions	Google Function	IBM Cloud
Languages	Java , Node.js, C# , Python	C#, F#, Node.js, Python, PHP, Bash, Powershell, Custom exe	JavaScript /Node.js	Javascript/Node.js Swift, Python, Java, Docker (custom)
Triggers	HTTP, Event Based, Scheduled	HTTP, Event Based, Scheduled	HTTP , Event Based ,Scheduled	Event Driven, HTTP
Free Trial	Yes	Yes	Yes	Yes
Free Requests	1 Million Free Request per month	1 Million Executions 400,000 GB-s	400,000 GB-seconds 2 million invocations 5GB of Internet egress traffic	400,000 GB-s free

Azure Functions- Use Cases



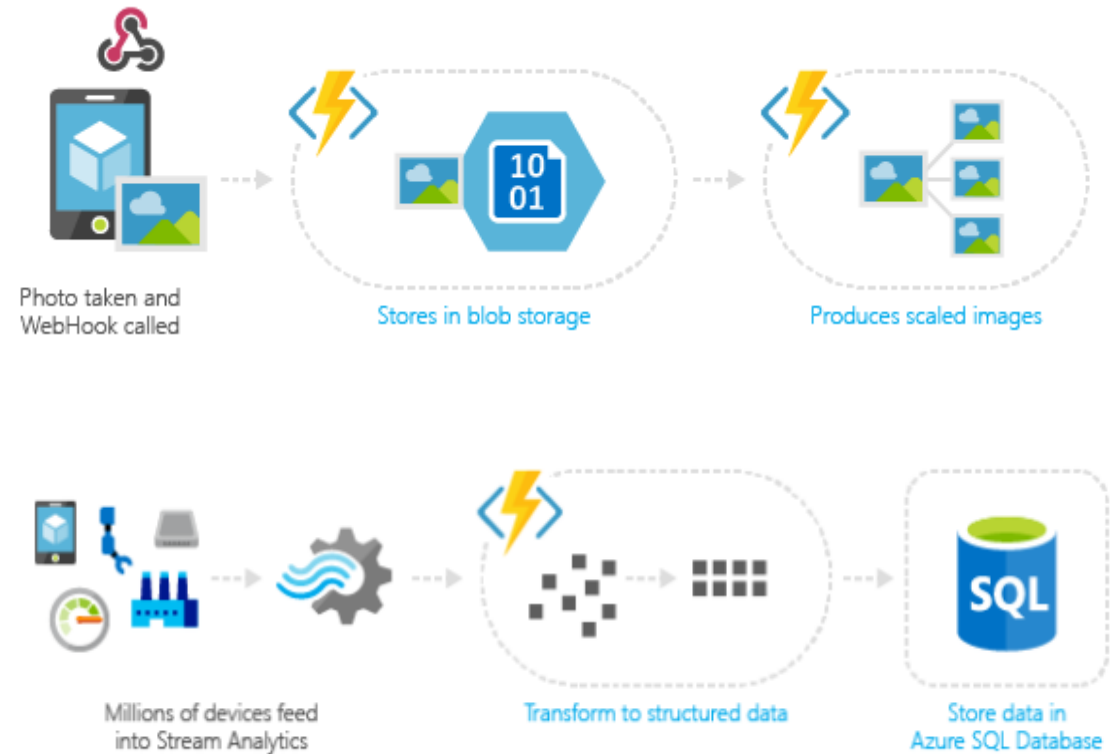
Azure Functions

- Timer Based Processing – Control the execution of function using Cron Expressions.
- Event Based Processing – Respond to Events with in Azure landscape. Blob Storage, Queues etc.
- Create Server Less APIs and integrate with Front End and Mobile Apps.



Azure Functions

- Respond to Events from IOT Hub and Process Incoming Events.
- Respond to GitHub Events using Web Hook.
- Integrate with Azure Storage Account , Twilio and SendGrid for mail communication.



Azure Cosmos



Azure Cosmos

- Support for NoSQL and MongoDB, Cassandra, Gremlin and SQL
- Easily integrate with Logic Apps and Azure Functions using built in connectors.
- Easily distribute data across any region. New Database would be available in 30 minutes anywhere in the world - 100 TBs or less
- Multi-Homing API provides support to perform read operation from nearest region.
- Index Free and Schema Agnostic.
- Control Access using Firewall.

Replicate data globally

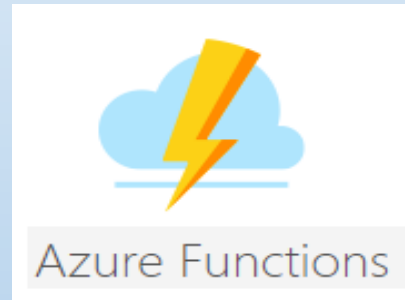
Failover Priorities

Click on a location to add or remove regions from your Azure Cosmos DB account.
* Each region is billable based on the throughput and storage for the account. [Learn more](#)

READ REGIONS	PRIORITIES
Australia East	1
Brazil South	2
Canada Central	3
Canada East	4
Central India	5
East Asia	6

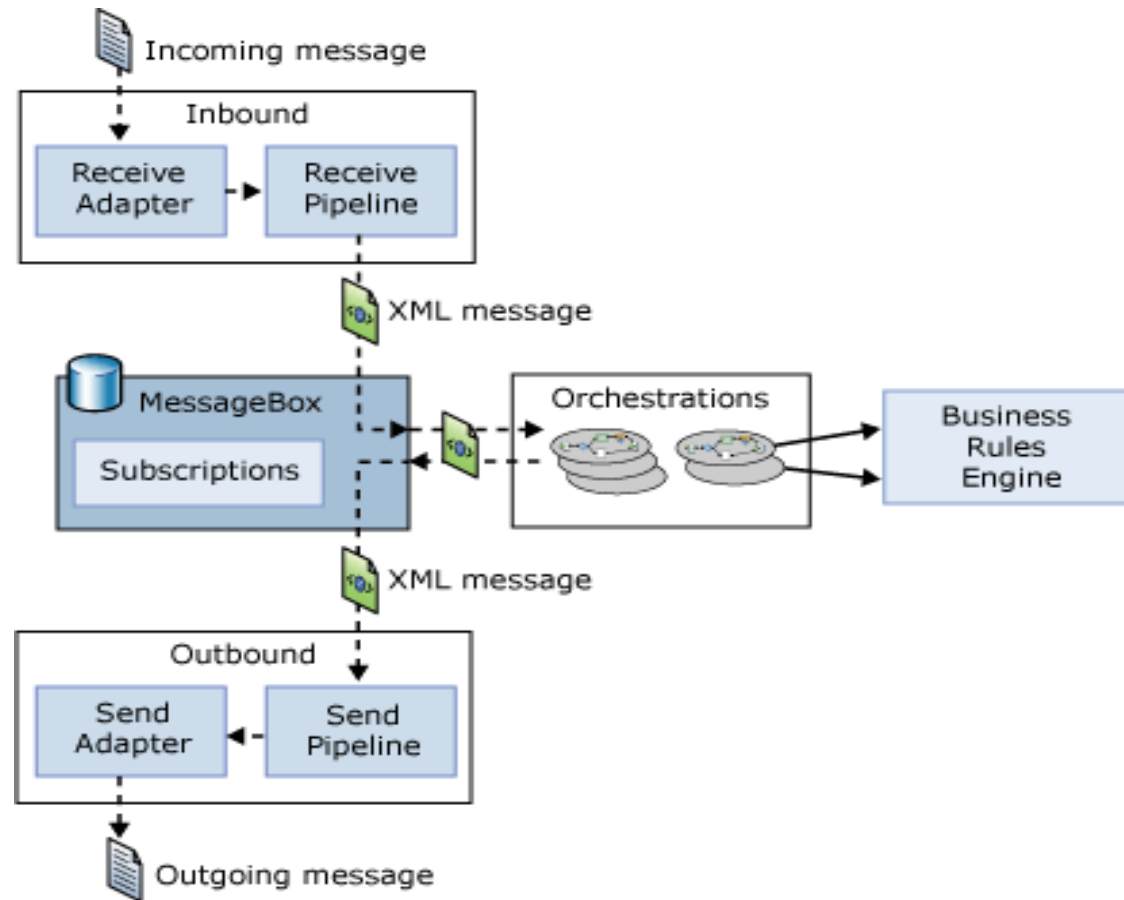
OK

Modernize EDI Operations



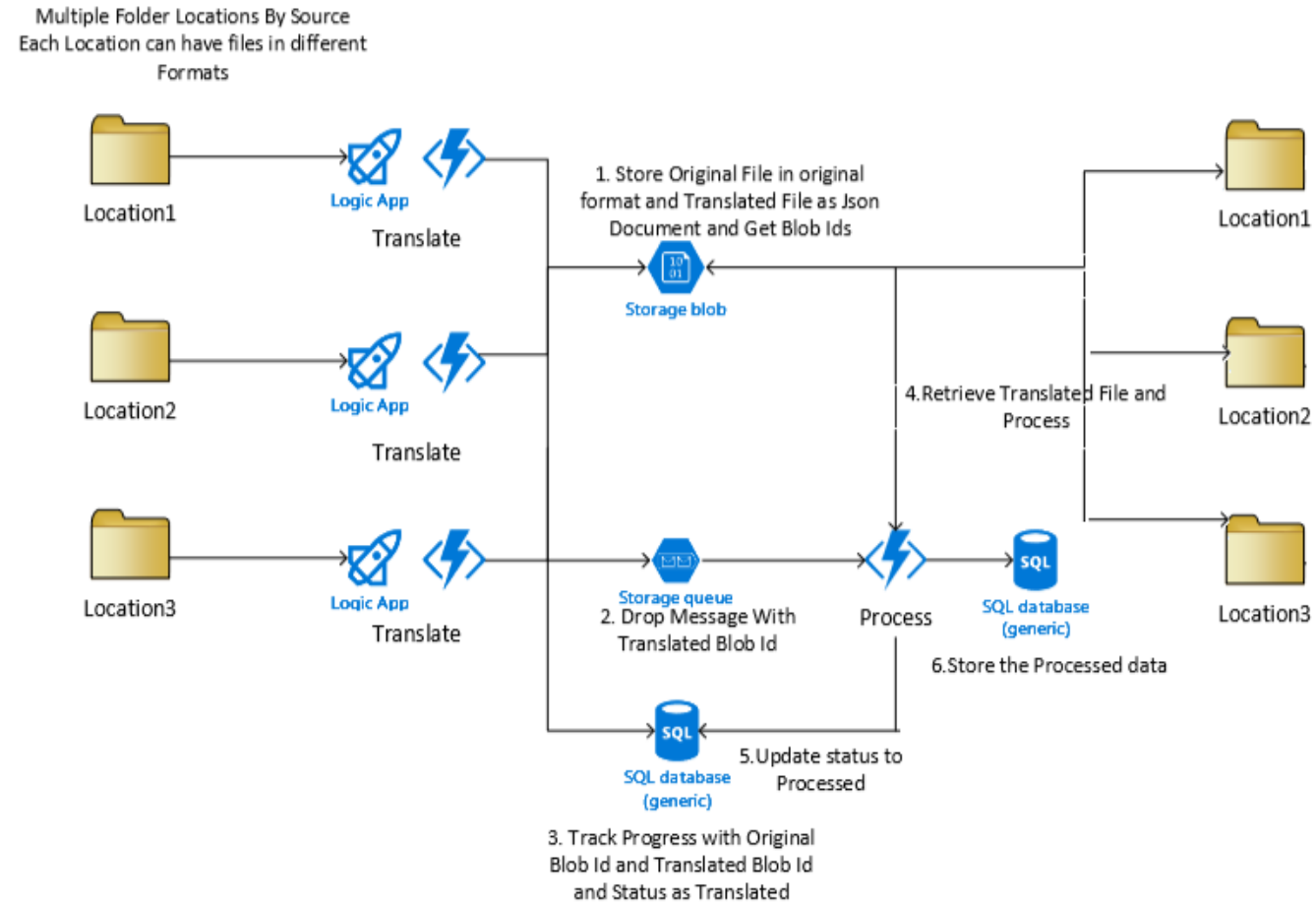
Traditional EDI Applications

- Files Received in Incoming Folder through Secure FTP or FTP.
- Incoming Folder Locations are monitored by BizTalk Server
- BizTalk Server process the incoming files through orchestrations and business rules.
- Processed Outputs are created to Outbound Folder

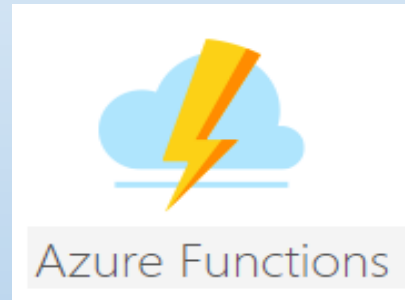


ServerLess EDI Applications

- Incoming Folder Locations will be monitored by Logic Apps.
- Logic Apps can invoke Function to perform translation
- Incoming EDI files can be stored in Blob Storage
- Process Function will be used to process incoming files
- Generate output files and transm to output location



Azure Functions and Cosmos Demo



Functions Walkthrough-Azure Portal

- Create an Http Trigger Function
- Accept order through HttpRequest
- Process the request and add an item to Azure Queue
- Create an Queue Trigger Function
- Save Processed Orders to Processed Orders Queue
- Create an Timer Trigger Function
- Send an Order Confirmation Email



Azure Functions

```
POST https://testbas.azurewebsites.net/api/AcceptOrder?code=1

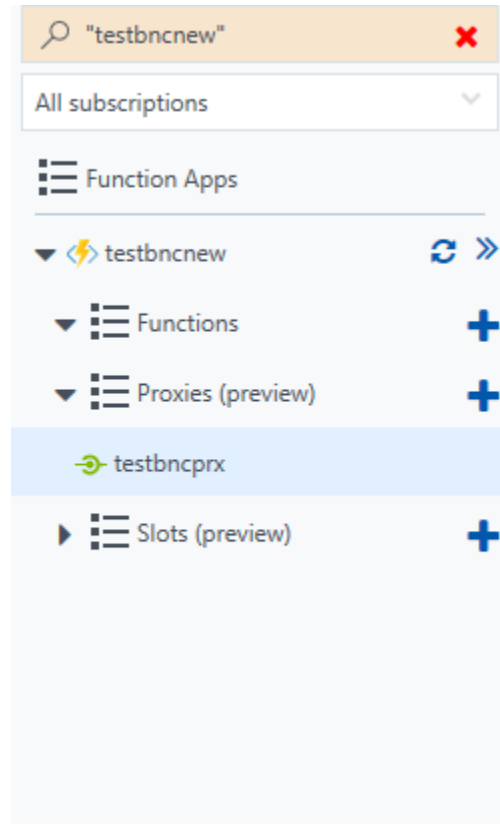
Authorization Headers (1) Body Pre-request Script Tests

form-data x-www-form-urlencoded raw binary JSON (a)

1 {
2
3
4   "ItemsList" : [
5     {
6       "ItemId" : 23,
7       "ItemName" : "testProduct",
8       "Qty" : 2,
9       "Price" : "10.00"
10    },
11   {
12     "ItemId" : 24,
13     "ItemName" : "testProduct2",
14     "Qty" : 201,
15     "Price" : "10.00"
16   }
17 ],
18 "EmailAddress" : "baskarmib@gmail.com"
19 }
```


Function Proxies-Azure Portal

- Create an Azure Function Proxy
- Define proxy route
- Configure Back End url
- Test the function proxy



The screenshot shows the Azure Portal navigation pane for a resource group named "testbncnew". The path is: All subscriptions > Function Apps > testbncnew > Proxies (preview) > testbncprx. The "testbncprx" proxy is selected and highlighted in blue.

testbncprx

[Delete proxy](#)

Proxy URL

<https://testbncnew.azurewebsites.net/test>

Route template

/test

GET

PATCH

POST

PUT

DELETE

OPTIONS

HEAD

TRACE

Backend URL

<https://testbncbas.azurewebsites.net/api/AcceptOrder?code=kcfgwE4Z/y91mmMzNkc6qm6RLxjAgCfa>

[+ Request override](#)

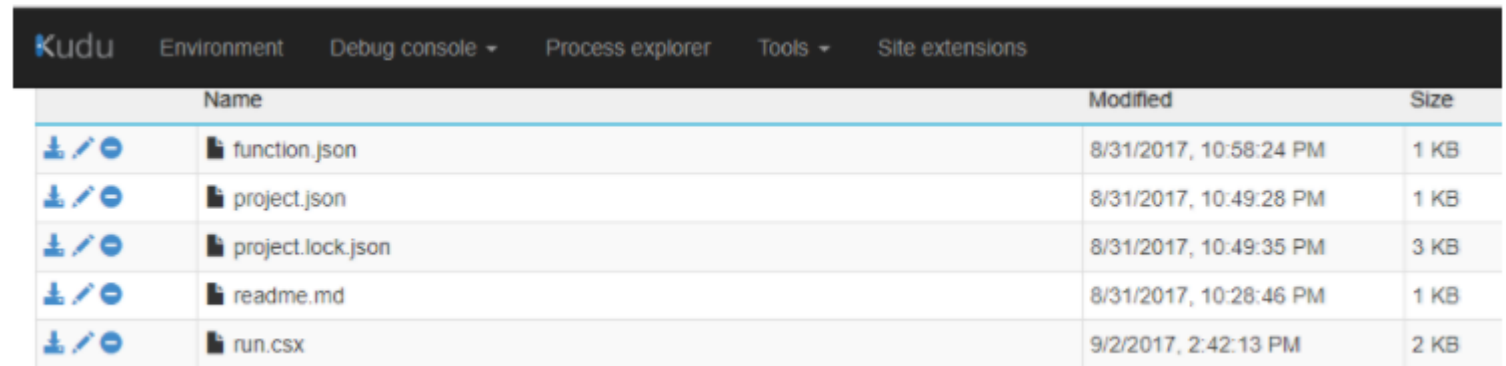


Azure Functions






Functions Walkthrough-Using Kudu

- Below url can be used to launch kudu explorer

<https://functionappname.scm.azurewebsites.net>



The screenshot shows the Kudu Explorer interface. At the top, there is a navigation bar with the following items: Kudu, Environment, Debug console (with a dropdown arrow), Process explorer, Tools (with a dropdown arrow), and Site extensions. Below the navigation bar is a table listing files. The table has three columns: Name, Modified, and Size. Each row also includes a set of icons (a person, a pencil, and a circle with a slash) to the left of the file name.

	Name	Modified	Size
	function.json	8/31/2017, 10:58:24 PM	1 KB
	project.json	8/31/2017, 10:49:28 PM	1 KB
	project.lock.json	8/31/2017, 10:49:35 PM	3 KB
	readme.md	8/31/2017, 10:28:46 PM	1 KB
	run.csx	9/2/2017, 2:42:13 PM	2 KB

- Kudu can be used to update settings and upload the files.
- Example – updating settings to update Nuget Packages.

Functions Walkthrough-Using Kudu

- Navigate to Debug Console and select cmd.
- A command line tool will be enabled.
- Using the command line tool, we can navigate to the folder pertaining to the above function. Function will be present inside wwwroot.

```
Kudu Remote Execution Console
Type 'exit' then hit 'enter' to get a new CMD process.
Type 'cls' to clear the console

Microsoft Windows [Version 6.2.9200]
(c) 2012 Microsoft Corporation. All rights reserved.

D:\home>cd site

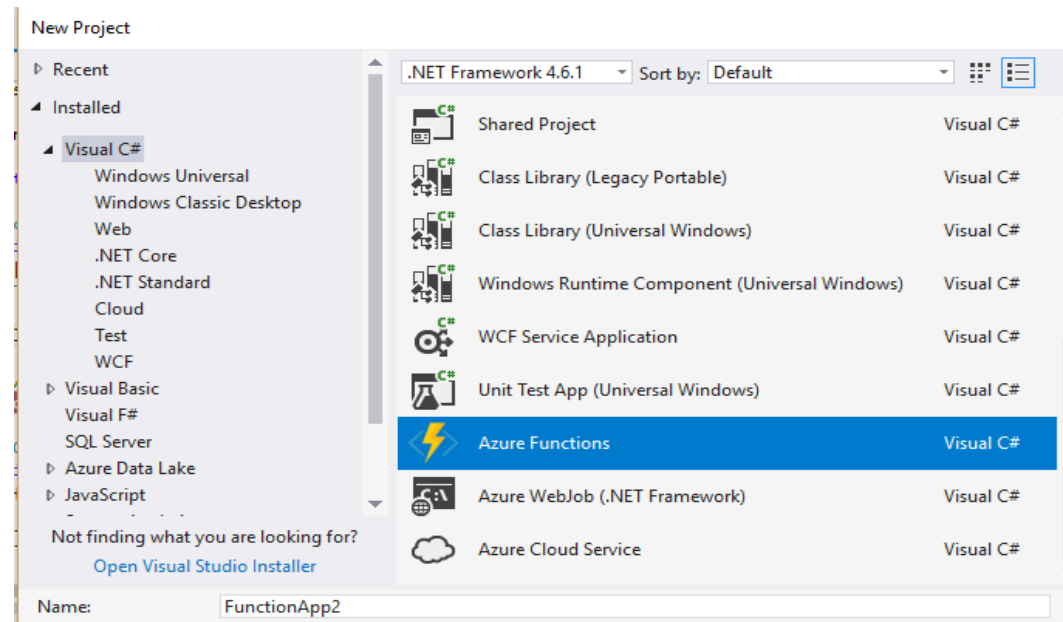
D:\home\site>cd wwwroot

D:\home\site\wwwroot>cd TimerTriggerCSharp1

D:\home\site\wwwroot\TimerTriggerCSharp1>
```

Functions Walkthrough-Visual Studio 2017

- Create a function using Visual Studio 2017
- Debug a function using Visual Studio 2017
- First time debug might take time as Visual Studio will download Azure CLI tools



- After you install or upgrade to Visual Studio 2017 version 15.3, you must manually update the Visual Studio 2017 tools for Azure Functions. You can update the tools from the Tools menu under Extensions and Updates... > Updates > Visual Studio Marketplace > Azure Functions and Web Jobs Tools > Update.

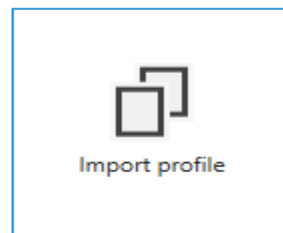
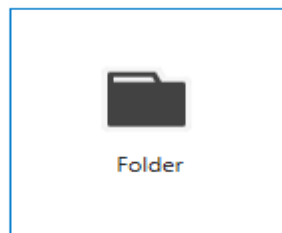
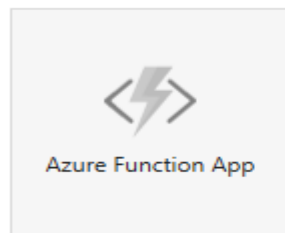
Functions Walkthrough-Visual Studio 2017

- Publish a function using Visual Studio 2017

Pick a publish target

What publishing targets can you deploy your app to?

Targets



Create New

Select Existing

Create App Service

Host your web and mobile applications, REST APIs, and more in Azure

Hosting

Services

App Name

FunctionApp20171012114938

Subscription

Resource Group

App Service Plan

Storage Account

If we publish a function to an existing function app, it is possible that the function settings will be overridden to read only mode. These can be updated in Function app settings in azure portal to make the function in read/write mode in order to use azure portal for development.

Cosmos DB Walkthrough-Azure Portal

- Create Cosmos using Azure Portal
- Create a Collection using Portal
- Add Documents to Collection using Portal
- Run Filter Queries using Portal
- Walkthrough Code Sample to Access Data

Create Azure Cosmos DB Account

PROJECT DETAILS

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

* Subscription

* Resource Group [Create new](#)

INSTANCE DETAILS

* Account Name documents.azure.com

* API

* Location

Geo-Redundancy

Multi-region Writes

Cosmos DB Walkthrough-Azure Portal

Create Azure Cosmos DB Account

Basics Network Tags Summary

NETWORK

Virtual Network ⌵
[Create a new virtual network](#)

Subnet ⌵

✓ Validation Success

Basics Network Tags Summary

BASICS

Subscription	Pay-As-You-Go
Resource Group	TestMobileNative
Location	Central US
Account Name	(new) testbasocosmos
API	DocumentDB
Geo-Redundancy	Enable
Multi-region Writes	Disable

VIRTUAL NETWORK

Virtual Network	(new) testcosmosvnet
Subnet	(new) default (10.0.0.0/24)

Cosmos DB Walkthrough-Azure Portal

The image shows two side-by-side screenshots of the Azure Portal interface for Cosmos DB.

Left Screenshot: Add Collection

- Navigation: New Database, New Collection, Open Full Screen
- API: SQL API
- Database id: Create new, Use existing. Value:
- Provision database throughput:
- Collection Id:
- Storage capacity: Fixed (10 GB), Unlimited
- Throughput (400 - 10,000 RU/s): (with +/- buttons)
- Estimated spend (USD): \$0.40 hourly / \$9.60 daily.
- Choose unlimited storage capacity for more than 10,000 RU/s.

Right Screenshot: Document View

- Navigation: New Database, New Collection, Open Full Screen, New SQL Query, New Stored Procedure, Upload
- API: SQL API
- Documents tab: New Document, Update, Discard, Delete
- Query: `SELECT * FROM c` (with Edit Filter button)
- Table view:

id
cwcitconf01

 (with Load more button)
- Code view:

```
1 {
2   "id": "cwcitconf01",
3   "sessiontitle": "Opening & KeyNote",
4   "sessionsubtitle": "Don't just show up.",
5   "sessionby": "Clark Sell, Founder of THAT Conference",
6   "sessiontype": "keynote",
7   "description": "Community is a word we started using at",
8   "time": "8:30-9:45",
9   "_rid": "C4xyALDmxkgBAAAAAAAAA==",
10  "_self": "dbs/C4xyAA=/colls/C4xyALDmxkg=/docs/C4xyALDm",
11  "_etag": "\"040022be-0000-0000-0000-5bb422120000\"",
12  "_attachments": "attachments/",
13  "_ts": 1538531858
14 }
```


Cosmos DB Walkthrough-Azure Portal

- Query Explorer can be used to query by applying filters

The screenshot displays the Azure Cosmos DB Query Explorer interface. On the left, a navigation pane shows the hierarchy: SQL API > cwitconf > conferenceSessions > Documents. The 'Documents' folder is selected. The main area shows a query: `SELECT * FROM c where c.time="10:00-10:45"` with an 'Edit Filter' button. Below the query, a table lists document IDs, with 'cwitconf02' selected. A 'Load more' link is visible. The document content is displayed in a code editor, showing a JSON document with fields like 'id', 'sessiontitle', 'sessionsubtitle', 'sessionby', 'sessiontype', 'description', 'time', '_rid', '_self', '_etag', and 'attachments'.

SQL API

Documents

New Document Update Discard Delete

SELECT * FROM c where c.time="10:00-10:45" Edit Filter

id

cwitconf02

Load more

```
1
2 {"id": "cwitconf02",
3  "sessiontitle": "Get the Boring Stuff Right: A Guide to
4  "sessionsubtitle": "",
5  "sessionby": "Dustin Ewers, Centare",
6  "sessiontype": "session",
7  "description": "Technology is a treadmill. Every year,
8  "time": "10:00-10:45",
9  "_rid": "C4xyALDmxkgCAAAAAAAAAA==",
10 "_self": "dbs/C4xyAA==/colls/C4xyALDmxkg=/docs/C4xyALDm
11 "_etag": "\"04002bbe-0000-0000-0000-5bb4251f0000\"",
12 "attachments": "attachments/"
```

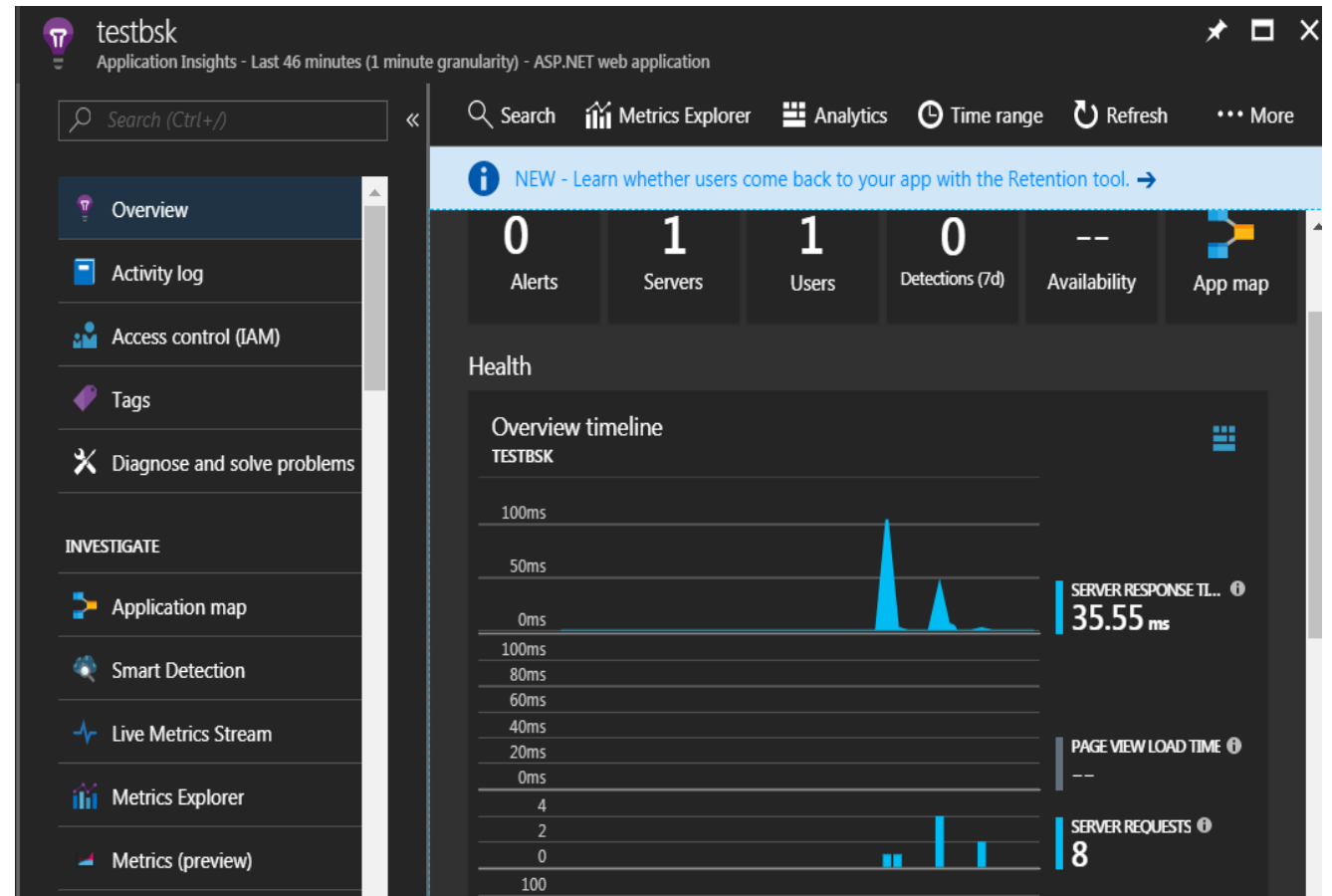
0 2 25

All Clear Notifications

10:14 PM Fetched 1 documents

Application Insights

- App Insights can be used to monitor
 - Number of Users
 - Number of Servers
 - Service Response Times
 - Total Number of Requests
 - Failed Request
 - Total Requests By Performance

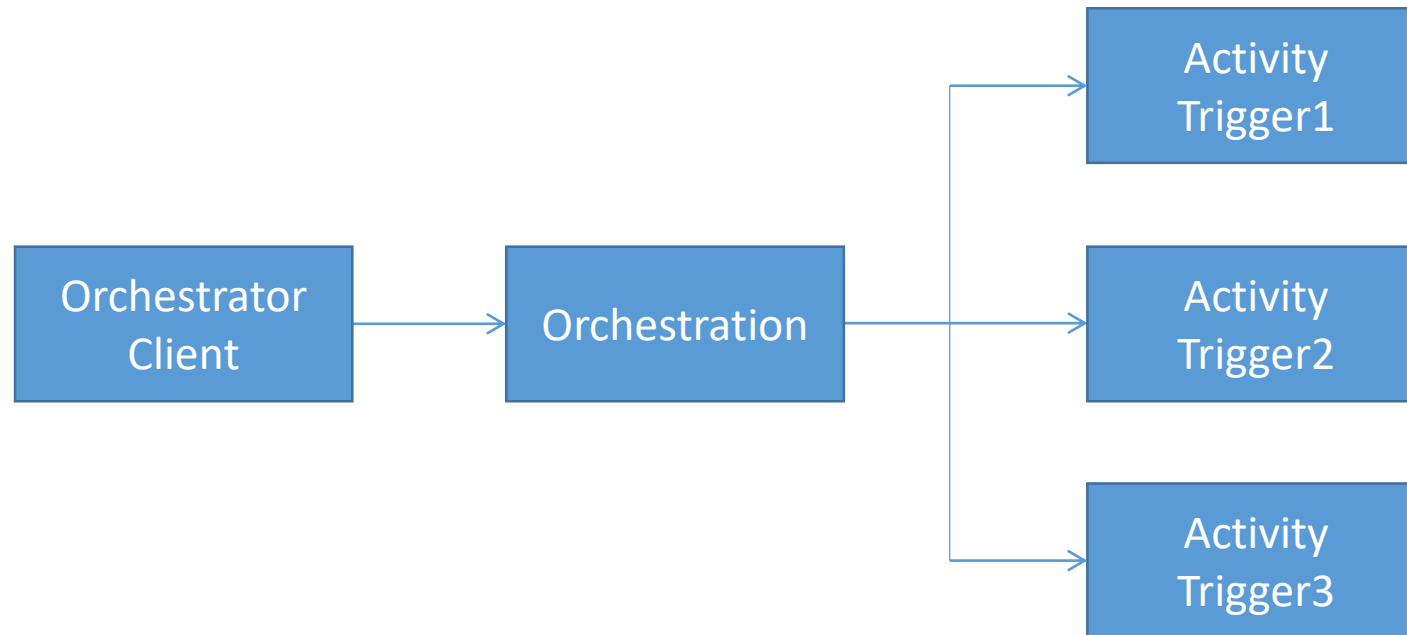


Durable Functions

- Durable Functions are still in preview mode.
- They support the development of stateful functions.
- The following patterns are recommended for Durable Functions
 - Function Chaining
 - Fan Out and Fan In Pattern
 - Async Http API and Monitoring
 - Human Interaction.
- Durable Functions are implemented using code workflows.

Demo Durable Functions

- Durable Functions demo using Function Chaining



Manage Durable Functions

- statusQueryGetUri
- sendEventPostUri
- terminatePostUri

POST ▼ http://localhost:7071/api/orchestrators/OrderProcessingSequence Params Send ▼

```
3  "ItemsList" : [  
4    {  
5      "ItemId" : 23,  
6      "ItemName" : "testProduct",  
7      "Qty" : 2,  
8      "Price" : "10.00"  
9    },  
10   {  
11     "ItemId" : 24,  
12     "ItemName" : "testProduct2",  
13     "Qty" : 2,  
14     "Price" : "10.00"  
15   }  
16 ],  
17 "EmailAddress" : ""  
18
```

Body Cookies Headers (6) Test Results Status: 202 Accepted Time: 4328 ms

Pretty Raw Preview JSON ▼ ≡

```
1 {  
2   "id": "30721c64040b42e2a69af5e8be7592a0",  
3   "statusQueryGetUri": "http://localhost:7071/admin/extensions/DurableTaskExtension/instances/30721c64040b42e2a69af5e8be7592a0?taskHub=DurableFunctionsHub&connection=Storage&code=nsJYxwZZP7GENh5G1mvHxn3g6K3n9vzr7mvUcFKjim6ZNMf10dt/dw=",  
4   "sendEventPostUri": "http://localhost:7071/admin/extensions/DurableTaskExtension/instances/30721c64040b42e2a69af5e8be7592a0/raiseEvent/{eventName}?taskHub=DurableFunctionsHub&connection=Storage&code=nsJYxwZZP7GENh5G1mvHxn3g6K3n9vzr7mvUcFKjim6ZNMf10dt/dw=",  
5   "terminatePostUri": "http://localhost:7071/admin/extensions/DurableTaskExtension/instances/30721c64040b42e2a69af5e8be7592a0/terminate?reason={text}&taskHub=DurableFunctionsHub&connection=Storage&code=nsJYxwZZP7GENh5G1mvHxn3g6K3n9vzr7mvUcFKjim6ZNMf10dt/dw=",  
6 }
```

Manage Durable Functions

- statusQueryGetUri

GET Params

Body Cookies Headers (4) Test Results Status: 400 Bad Request Time: 2977 ms

Pretty Raw Preview JSON

```
1 {
2   "runtimeStatus": "Failed",
3   "input": {
4     "ItemsList": [
5       {
6         "ItemId": 23,
7         "ItemName": "testProduct",
8         "Qty": 2,
9         "Price": "10.00"
10      },
11      {
12        "ItemId": 24,
13        "ItemName": "testProduct2",
14        "Qty": 2,
15        "Price": "10.00"
16      }
17    ],
18    "EmailAddress": ""
19  },
20   "output": "Exception while executing function: SendOrderConfirmation",
21   "createdTime": "2018-03-29T15:36:55Z",
22   "lastUpdatedTime": "2018-03-29T15:37:09Z"
23 }
```

Manage Durable Functions

- statusQueryGetUri

The screenshot displays a REST client interface for a GET request. The URL is `http://localhost:7071/admin/extensions/DurableTaskExtension/instances/6b141f29d3b747ebbbd515788e6dc4a2?taskHub=DurableFunctionsHub&connectio...`. The request headers include `Content-Type: application/json`. The response status is `200 OK` and the time taken is `192 ms`. The response body is a JSON object with the following structure:

```
1 {
2   "runtimeStatus": "Completed",
3   "input": {
4     "ItemsList": [
5       {
6         "ItemId": 23,
7         "ItemName": "testProduct",
8         "Qty": 2,
9         "Price": "10.00"
10      },
11      {
12        "ItemId": 24,
13        "ItemName": "testProduct2",
14        "Qty": 2,
15        "Price": "10.00"
16      }
17    ],
18    "EmailAddress": "baskarmib@gmail.com"
19  },
20   "output": {
21     "OrderNumber": "bad4b1e0-b38a-49db-a4ce-d5f6214e6954",
22     "ItemsList": [
23       {
24         "ItemId": 23,
25         "ItemName": "testProduct",
26         "Qty": "2",
```

Key Points to Observe

- In azure portal , all the bindings are declared through orchestration or Integrate tab.
- Function developed from Visual Studio does not have the integrate tab options in portal.
- The method signatures are different in portal and Visual Studio. Visual Studio it is required to provide the signature attributes.
- When we use a Queue Trigger or any event based triggers until the function does not exit gracefully , function will keep on trying to execute the event repeatedly.

Server Less Computing - Limitations

Below are some disadvantages

- There might be increase in response time from functions if they are not used continuously.
- There is no direct control on the CPU size or Memory allocation as they are allocated based on usage while using Consumption Plan.
- We can configure source control for functions developed in portal after initial deployment.

Code Samples

Below are the code samples in GitHub

Http Trigger, Queue Trigger and Timer Trigger Functions

<https://github.com/baskar3078/testazurefunctions>

The above functions converted as Durable Function Activities

<https://github.com/baskar3078/durableFunctionSample>

Microsoft Ignite 2018 Announcements

<https://news.microsoft.com/uploads/prod/sites/507/2018/09/IGNITEBOOKOFNEWS-5ba90f5a37c54.pdf>

Questions



<https://www.linkedin.com/in/baskarrao-dandlamudi>

baskarrao.dandlamudi@outlook.com

<https://baskarrao.wordpress.com/>

Please share your feedback at

https://docs.google.com/forms/d/1iVfmlwUoyVh_gFpIOjmz_rJ_uH1QXy17IMax_xeK2Mw