

Requirement Doc

Globally Distributed App

September 6, 2017

Click2Cloud Inc.

Email: contact@click2cloud.net

Document Version – V1.0

Contents

[Introduction 3](#_Toc492496385)

[Overview 3](#_Toc492496386)

[Scope 3](#_Toc492496387)

[Out of Scope 4](#_Toc492496388)

[Assumptions 5](#_Toc492496389)

[Prerequisites 5](#_Toc492496390)

[LAB 1: Globally-distributed apps using Azure Cosmos DB Service 5](#_Toc492496391)

[Details for Immersion Labs 5](#_Toc492496392)

[Problem Statement 6](#_Toc492496393)

[Solution 6](#_Toc492496394)

[Work Flow 6](#_Toc492496395)

[Scenario 6](#_Toc492496396)

[Technology Used 8](#_Toc492496397)

[LAB 2: Predictive analysis using Cosmos DB, Machine Learning and Power BI 8](#_Toc492496398)

[Details for Immersion Labs 8](#_Toc492496399)

[Problem Statement 9](#_Toc492496400)

[Solution 9](#_Toc492496401)

[Workflow 9](#_Toc492496402)

[Scenario 10](#_Toc492496403)

[Technology Used 11](#_Toc492496404)

[LAB 3: Connect Bot Service with Cosmos DB 11](#_Toc492496405)

[Details for Immersion Labs 11](#_Toc492496406)

[Problem Statement 12](#_Toc492496407)

[Solution 12](#_Toc492496408)

[Workflow 13](#_Toc492496409)

[Scenario 13](#_Toc492496410)

[Technology Used 15](#_Toc492496411)

[LAB 4: Using Graph (Gremlin) API of Azure Cosmos DB Service 15](#_Toc492496412)

[Details for Immersion Labs 15](#_Toc492496413)

[Problem Statement 15](#_Toc492496414)

[Solution 15](#_Toc492496415)

[Workflow 16](#_Toc492496416)

[Scenario 16](#_Toc492496417)

[Technology Used 17](#_Toc492496418)

[LAB 5: Submit Feedback using Bing Speech API, Text Analytics API, Emotion API, Azure Function and Cosmos DB 17](#_Toc492496419)

[Details for Immersion Labs 17](#_Toc492496420)

[Problem Statement 18](#_Toc492496421)

[Solution 18](#_Toc492496422)

[Work flow 18](#_Toc492496423)

[Scenario 19](#_Toc492496424)

[Technology Used 20](#_Toc492496425)

[LAB 6: Azure notification service for Flying solo service(minor’s) 20](#_Toc492496426)

[Details for Immersion Labs 20](#_Toc492496427)

[Problem Statement 21](#_Toc492496428)

[Solution 21](#_Toc492496429)

[Workflow 22](#_Toc492496430)

[Scenario 22](#_Toc492496431)

[Technology Used 23](#_Toc492496432)

[LAB 7: Azure notification service for tracking luggage status 23](#_Toc492496433)

[Details for Immersion Labs 23](#_Toc492496434)

[Problem Statement 24](#_Toc492496435)

[Solution 24](#_Toc492496436)

[Workflow 24](#_Toc492496437)

[Scenario 25](#_Toc492496438)

[Technology Used 25](#_Toc492496439)

[Deliverables and Milestones 26](#_Toc492496440)

[Disclaimer 27](#_Toc492496441)

[Appendix A: Glossary 28](#_Toc492496442)

[Appendix B: Document Change History and Sign-off 28](#_Toc492496443)

[Point of Contact 28](#_Toc492496444)

# Introduction

## Overview

The objective of this document is to describe the technical requirement of the proposed labs for immersion platform. Click2Cloud will work with Microsoft to build the Hands-on Labs largely around Azure Cosmos DB service. The Azure Cosmos DB is Microsoft's globally distributed, multi-model database. With the click of a button, Azure Cosmos DB enables us to elastically and independently scale throughput and storage across any number of Azure's geographic regions.

Proposed labs would be called as; Globally Distributed Apps, one of the innovative strategy of Microsoft designed to deliver apps for FY18. Once the App would get ready, it would be used across the globe to validate Microsoft’s vision and strategy.

The globally distributed app would be an extension of the “ContosoAir” app demo, which is a customer facing app. The app would be modified to store the data in Cosmos DB. There would be 6 different labs taking about Cosmos DB and its features. Each lab would present different aspects of global distributed applications.

## Scope

We are dividing scope in to three major piece:

Immersion Labs:

1. Click2Cloud will develop below 6 labs using the Contosoair app theme. Each lab will portray the different capabilities of Cosmos DB and talk about the “Develop Globally Distributed Apps”. Each lab would run in isolation with reference to another lab.

*[NOTE: Microsoft needs to select the 6 labs out of the 7 labs below. The selected 6 labs would be developed as a part of scope]*

|  |  |  |
| --- | --- | --- |
| # | Story | Participating Technologies |
| 1 | Globally-distributed apps using Azure Cosmos DB Service | App Service, Cosmos DB, ARM Templates, Ansible Playbooks |
| 2 | Predictive analysis using Cosmos DB, Machine Learning and Power BI | Cosmos DB, Machine Learning, Power BI, VSTS |
| 3 | Connect Chat Bot with Cosmos DB | Cosmos DB, Bot Services, Bing Map, Jenkins |
| 4 | Using Graph (Gremlin) API of Azure Cosmos DB Service | Cosmos DB  Cosmos DB Graph API  Azure Functions |
| 5 | Submit Feedback using Bing Speech API, Text Analytics API, Emotion API, Azure Function and Cosmos DB | Cognitive Service (Emotion API)  Cognitive Service (Bing Speech API)  Cognitive Service (Text Analytics API)  Azure Function  Cosmos DB |
| 6 | Azure notification service for Flying solo service(minor’s) | Cosmos DB  Notification Hub API  Text Analytics API |
| 7 | Azure notification service for tracking luggage status | Cosmos DB  Logic App  Azure Functions  Notification Hub API |

1. Click2Cloud will work on designing the content pack including labs documentations as per immersion platform standard and deploy and setup the Hands-on Lab for Cosmos DB on immersion platform.
2. UAT and feedback implementation that can be accomplish within a specified time period.
3. The lab will be developed for Web platform (Node.js version) and few scenarios will be developed for Xamarin platform.

MTC-0 Demo extension:

1. Click2Cloud will work on designing the deep technical document for MTC-0 for the finalized labs. The documents include –

|  |  |  |
| --- | --- | --- |
| # | Document | Doc Format |
| 1 | Cosmos DB Briefing Overview | Word |
| 2 | Cosmos DB Pitch Short | PPT |
| 3 | Cosmos DB Presenter Script | Word |
| 4 | Cosmos DB Architecture | Visio |
| 5 | Cosmos DB Deployment Guide | Word |
| 6 | Cosmos DB Re-Deployment Guide | Word |

1. Document changes and incorporate feedback

Support:

1. Bug fixing in labs within specified period
2. Content amendments

## Out of Scope

1. Anything task are not listed in the [Scope](#_Scope) section will be treated as out of scope for the project.
2. The lab will be developed for Web platform only (Node.js version). The Xamarin platform will not be considered as a part of the scope.
3. All labs will be tested manually, there will not be any automation work considered in the project scope.
4. All the supporting documents and labs documents will be provided in English language only.

# Assumptions

1. All the documents and guideline for creating immersion platform will be provided by Microsoft.
2. Change in any functional requirement, shall be treated as CR (Change Request).
3. This document need to be freeze and sign-off before implementation start.
4. To deliver project within stipulated time, we assume that Microsoft provide timely feedback.
5. Tasks/items mentioned in the [Out of Scope](#_Out_of_Scope) are not part of requirement.
6. All the supporting documents and labs documents will be provided in English language only.
7. Incorporating the changes as an impact of any update or additional feature in services will be treated as different CR.
8. During the project lifecycle, the feedback or changes that can be incorporated within the specified period, will be considered. If priority changes, the CR will be raised.

# Prerequisites

1. Remote Desktop (RDP) Client to connect with labs
2. Access to VSTS where Contoso Air application code will be available

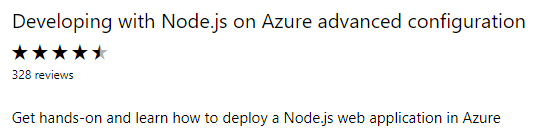
# LAB 1: Globally-distributed apps using Azure Cosmos DB Service

## Details for Immersion Labs

Lab Title: Globally-distributed apps using Azure Cosmos DB Service

Lab Description: Get hands-on and learn how to build and deploy globally distributed application using Cosmos DB in Azure and explore Cosmos DB capabilities

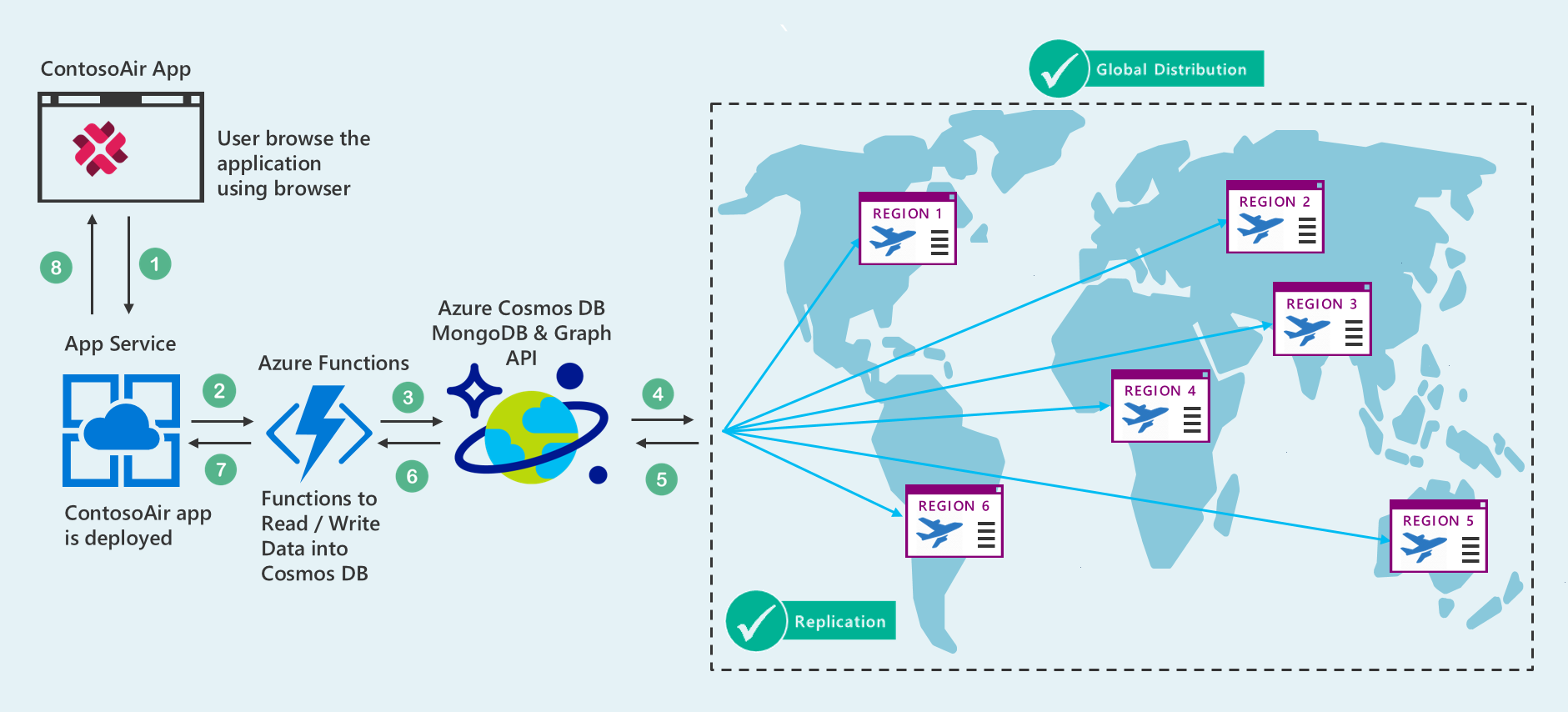
NOTE: The lab title and description will be displayed on Hand-on lab as specified in below screen. Please review and approve the same.



## Problem Statement

It was taking a long time and a lot of tedious effort to build a globally distributed database traditionally and there were some issue of latency and consistency in case of the data stored in centralized location. So, to solve this we will use Azure Cosmos Database.

## Solution



## Work Flow

1. Using ARM Templates, Kevin can deploy his application and database in different regions with the help of Azure App.
2. While browsing the Contoso Air application, Kevin will see the flight details which are coming from the Cosmos DB using MongoDB API database.
3. After searching the relevant flight, he will book the flight ticket. All the data related to flight booking will be stored in Cosmos DB using MongoDB API and Azure Functions.
4. Azure functions will have a logic to store the data in the Cosmos DB using MongoDB API, this can be used in another lab too.

## Scenario

|  |  |
| --- | --- |
| 1 | Scenario walkthrough – Setup lab environment using DevOps Tools\*  The lab environment setup will include, deploying the front-end application, service API and Cosmos DB.  Prerequisites:   1. Lab will be setup to run the Ansible Playbooks. 2. ARM Templates to setup application resources. 3. Access to VSTS.   Instructions:   1. To setup the lab environment, developer will launch Ubuntu bash shell and update the required information such as Subscription details, Resource group etc. in Ansible playbook. 2. Then he will run Ansible playbook to setup the environment. 3. Now, he will login to Azure Portal and verify the application resource in provided   Resource group.  NOTE: We are currently evaluating possibility of this scenario. |
| 2 | Scenario walkthrough – A single Azure Cosmos DB collection partitioned and distributed across multiple Azure regions  Latency is the delay between a client request, probably a request made by you at your computer, and a response to that request.   1. Kevin will run the Contoso Air application. 2. The application which he deployed, is available in only one region, for ex. East US. 3. He will then try to access the application and do all other operations like searching, booking a ticket and so on. 4. As in our example the Cosmos DB is initially available in Singapore region only, so he will notice the high latency; as both Cosmos DB and Web Service, which request for data, are hosted on different regions. 5. After that, Kevin will again change the region, in the ARM template, and extend the support of Cosmos DB in East US region. 6. Kevin will again repeat the same process of searching and booking a ticket and this time he will get a low latency. |
| 3 | Scenario walkthrough – Tunable data consistency levels in Azure Cosmos DB  Different regions have their own copy of database, and they may not always be consistent if read / write operations happen frequently.   1. There is a possibility where another person books the same seat from another region which is already booked recently. 2. So, the person from this region should be aware about these changes. 3. To test this scenario Kevin can deploy this application in two different regions and will try to book an air ticket for a seat by browsing the application. 4. After that, Kevin will open the application deployed in another region and will try to book an air ticket for a same seat. 5. Kevin will set the different consistency level of the Cosmos database to see the impact of how frequently the data updates. |
| 4 | Scenario walkthrough – Multi-Model API Support   1. From the Contoso Air application, Kevin will search the flight records which will retrieve the data using MongoDB API. 2. Kevin can see the graphical data based on the feedback provided by different users in LAB-4. In Lab-4 we have stored all the data using Cosmos DB (Graph API). |
| 5 | Scenario walkthrough – Partitioning in Azure Cosmos DB  In Azure Cosmos DB, the number of partitions is determined by Cosmos DB based on the storage size and the provisioned throughput of container.  Every partition in Cosmos DB has a fixed amount of SSD-backed storage associated with it, and is replicated for high availability.  1. Kevin can set the throughput in the application.  2. In this application we are using ID of the flight as a partition key.  3. He can change this partition key to test and understand the importance of partition key |

## Technology Used

App Service: For hosting front end and service layer

ComosDB with MogoDB API: It is used to store and retrieve data of different countries for plotting it into the Bing Map

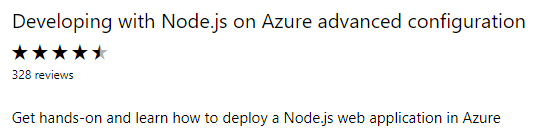
# LAB 2: Predictive analysis using Cosmos DB, Machine Learning and Power BI

## Details for Immersion Labs

Lab Title: Predictive analysis using Cosmos DB, Machine Learning and Power BI

Lab Description: Get hands-on and learn how to do predictive analysis using Cosmos DB, Machine Learning and Power

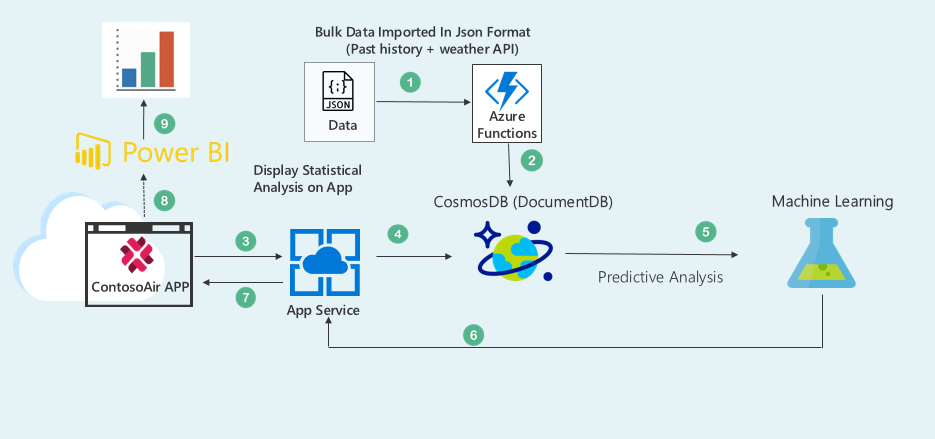
NOTE: The lab title and description will be displayed on Hand-on lab as specified in below screen. Please review and approve the same.



## Problem Statement

Flight delay is one of the most remembered performance indicator of any transportation system. Kevin wants to travel from Seattle to Barcelona for a very important business meeting, he books the ticket with one the of Airline companies. He reaches to the Airport on the day of his flight and finds that the flight with which he was supposed to travel will be delayed or cancelled and came to know that the Airline company with which he booked the ticket is having bad records of flight delay. Another reason for the delay was bad weather. Such kind of situation may put a person into a trouble to avoid such situations the Contoso Air app has developed a solution to perform the Flights delay predictive analysis.

## Solution



## Workflow

* + - 1. Bulk data will be imported through Azure Functions into Cosmos DB in JSON format.

Note: Bulk data includes the data retrieved from weather API and airlines historical data of delay and cancellation.

* + - 1. Request generated to view the flight status will be sent through Contoso Air App which is hosted on App service.
      2. Data will be retrieved from Cosmos DB which will be then sent to Machine learning service for Predictive Analysis.
      3. Using Contoso Air website user can view the analytical report generated through Power BI, before booking a ticket or after booking a ticket.

## Scenario

|  |  |
| --- | --- |
| 1 | Scenario Walkthrough – Contoso Air Website   1. Kevin goes to Contoso Air website. 2. He selects the Source - Destination and Airline with which he wants to travel. 3. And will check the past delay/cancellation history of that Airline with Destination on clicking the option provided to view the status. 4. On getting positive results after analyzing the past delay/cancellation history status, he will book the ticket with that Airline. 5. He can even get the flight delay/cancellation status a day before boarding his flight. 6. By clicking on View flight status button present on the Contoso Air website. |
| 2 | Scenario walkthrough – Cosmos DB (Document DB API)   1. After clicking on the View flight status button, it will fetch the data from Cosmos DB (Document DB API). 2. Data present in the Cosmos DB (Document DB API) will be fetched in JSON format and processed through Azure Functions. |
| 3 | Scenario walkthrough – Machine Learning   1. Fetched data from the Cosmos DB (Document DB API) will be processed through Machine Learning. 2. Machine learning service will give the predictive analysis for flight delay, cancellation depending on historical data. |
| 4 | Scenario walkthrough – Power BI   1. Delay prediction or flight cancellation analytics will be displayed through Power BI. 2. On clicking the URL present on Contoso Air website, it redirects the user on the page of analytics generated through Power BI. |
| 5 | Configure Continuous Integration and Continues Delivery using VSTS\*  Developer wants to add the new functionality in existing Contoso Air application. While developing solution, he needs to manually deploy the bits on Azure cloud and test the changes. Also, he wants to continuously check, if there is any impact of code commit on any other functionality. So, he will configure the CI/CD using the Visual Studio Team Services.  Prerequisites:   1. Access to VSTS.   Instructions:   1. To setup the CI/CD, launch the Visual Studio Team Services web portal in browser. 2. Select the Build & Release tab and configure the build steps as per provided instructions [here](https://www.visualstudio.com/en-us/docs/build/apps/node/nodejs-to-azure). 3. Now, once the setup is done, developer will commit his changes and from the VSTS checks if new build is started in Build & Release tab. 4. The VSTS will first run the unit test scripts and validate the code commit. If test script scenario failed, developer will receive notification. Or else, the new code bits will be deployed on Azure cloud and Contoso Air app will be updated.   NOTE: We are currently evaluating possibility of this scenario. |

## Technology Used

Cosmos DB: Cosmos DB is used to store and retrieve the Airline data.

Machine Learning: Predictive analysis is done by machine learning service with the help of Cosmos DB database.

Power BI: It generates the analytics depending on the result of Machine learning module.

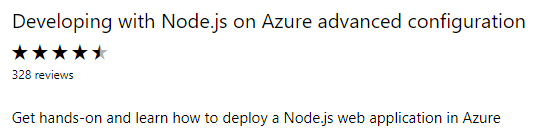
# LAB 3: Connect Bot Service with Cosmos DB

## Details for Immersion Labs

Lab Title: Connect Bot service with Cosmos DB

Lab Description: Get hands-on and learn how to connect Bot service with Cosmos DB

NOTE: The lab title and description will be displayed on Hand-on lab as specified in below screen. Please review and approve the same.

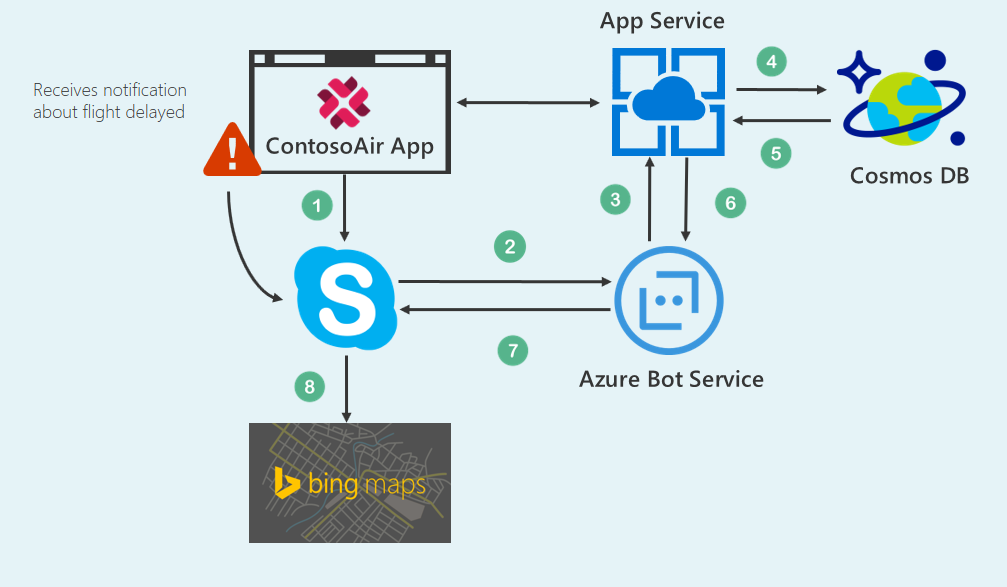


## Problem Statement

Kevin is driving towards the airport and due to some reason, he would get late to catch up his flight. So, he wants immediate help regarding flights scheduling, alternate flights options and E check-in to get boarding pass. He also need to get the airport terminal map, so that he can save time to find out the route to the boarding gate. He can use the bot chat service to get flight assistance.

The bot chat window will display the alternate flight options, so he can choose anyone option and confirm ticket for same. The customer also get the boarding pass, travel information and terminal map from bot chat window.

## Solution



## Workflow

The architecture describes the communication between all Azure services. In this architecture,

* + - 1. the Contoso Air application communicates with Azure Cosmos DB for data exchange.
      2. All the flight detail information is stored in Azure Cosmos DB.
      3. The notifications related to flight status will be displayed on the application, so user click on notification and open Skype Bot service.
      4. The Skype bot service communicate with Azure Bot service which exchange the traveler information data with Azure Cosmos DB.
      5. The Bing Map API is used to display the airport map view to find the shortest path to reach the airport.

## Scenario

|  |  |
| --- | --- |
| 1 | Scenario walkthrough – Flight status using Chat Bot   1. Kevin has previously booked flight ticket from Contoso Air app. 2. While traveling to the airport, he may get late to reach the airport because of traffic. 3. Kevin will launch the Contoso Air app and then open Skype bot service. 4. Kevin first type “Hi” on the Skype chat window. 5. On receiving request from Kevin, the Bot service will response back to Kevin and asks to send travel details. 6. Kevin will send its travel details using Bot service. With this travel details, skype bot will interact with Cosmos DB and get back flight details. 7. The Bot service will answer back about the flights information such as flight status, terminal number, gate number etc. 8. Incase Kevin finds that he can reach the airport on time, he can do the E Check-in through Bot services. 9. The bot service will pass the request to the Cosmos DB and send the boarding pass to Kevin using Bot service. 10. After reaching the airport, Kevin can now directly board the flight. |
| 2 | Scenario walkthrough – Alternate flights   1. If Kevin finds that he can’t reach the airport on time, he will open Skype bot chat window from Contoso Air app. 2. He will query for alternative flights through Skype bot. 3. Skype bot will check flight data in Cosmos DB and send alternate flights options for which Kevin has requested for. 4. Kevin will select suitable alternative flight and get confirmed ticket through Skype bot chat window and cancel his booking for current flight. 5. The bot service will pass the request to the Cosmos DB and save the record in Cosmos DB. 6. Kevin will receive the boarding pass from Cosmos DB for the alternative flight through skype bot. |
| 3 | Scenario walkthrough – Terminal map   1. Kevin will get the boarding pass from Cosmos DB through Skype bot, he will then request for terminal map or image. 2. In response to the Kevin’s query, the bot will send the terminal map or image to him which is stored in Cosmos DB. 3. Kevin will use this Map or image to reach the boarding gate as early as possible. |
| 4 | Configure Continuous Integration and Continues Delivery using Jenkins\*  Developer wants to add the new functionality in existing Contoso Air application. While developing solution, he needs to manually deploy the bits on Azure cloud and test the changes. Also, he wants to continuously check, if there is any impact of code commit on any other functionality. So, he will configure the CI/CD using the Jenkins  Prerequisites:   1. Access to VSTS. 2. Jenkins Server   Instructions:   1. To setup the CI/CD, launch the Jenkins console in a web browser and log in using provided credentials. 2. Under the Build tab, configure the new build as per provided instructions [here](https://www.visualstudio.com/en-us/docs/marketplace/integrate/service-hooks/services/jenkins). 3. Now, once the setup is done, developer will commit his changes and from the Jenkins checks if new build is started. 4. The Jenkins will first run the unit test scripts and validate the code commit. If test script scenario failed, developer will receive notification. Or else, the new code bits will be deployed on Azure cloud and Contoso Air app will be updated.   NOTE: We are currently evaluating possibility of this scenario. |

## Technology Used

Cosmos DB: Cosmos DB is used to store and retrieve the Airline data.

Bot Services: For knowledge based answering to the user regarding travel details.

Bing Map: Bing Map service is used to share possible shortest location to airport

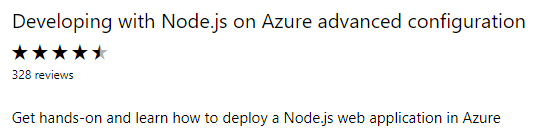
# LAB 4: Using Graph (Gremlin) API of Azure Cosmos DB Service

## Details for Immersion Labs

Lab Title: Using Graph (Gremlin) API of Azure Cosmos DB Service

Lab Description: Explore Graph (Gremlin) API of Azure Cosmos DB Service

NOTE: The lab title and description will be displayed on Hand-on lab as specified in below screen. Please review and approve the same.

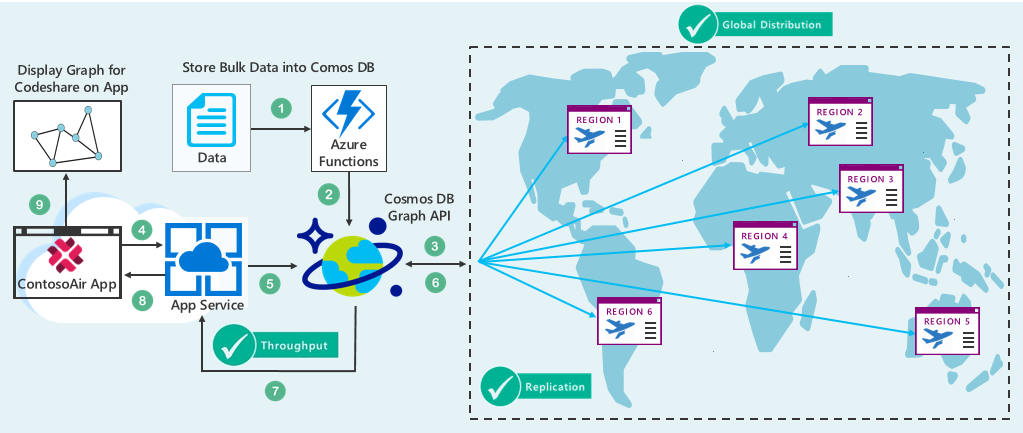


## Problem Statement

Kevin wants to travel by air, he books ticket for specific airline. But, in a situation, he may experience that he is not going to travel with same airline, which he has booked for, but with a different airline. This happens because of codeshare, which is a business arrangement where two or more airlines shares the same flight. To avoid customer inconvenience, they can check codeshare for booked airline by clicking Check Codeshare button and this will show all codeshare airline details of booked flight.

## Solution

The below architecture provides overview of Lab 2, which displays the airline codeshare using ContosoAir app, Cosmos DB Graph (Gremlin) API, App service and Azure functions.



## Workflow

1. In first steps, JSON data is read from JSON file through data store utility and forward it to Azure functions.
2. In the next step, Azure functions will process data and store it into the Cosmos DB database.
3. Cosmos DB supports Global Distribution feature by storing data into available multiple physical location.
4. Next, when end user visit Contoso Air app for flight searching or booking its request transfer to App Service.
5. App Service is the place where our Contoso Air app is hosted and responsible for communicating with the Cosmos DB database.
6. Cosmos DB will retrieve required data from nearest physical location and achieve max throughput using Cosmos DB.
7. Retrieved codeshare data, then transferred to Contoso Air app for further process through App Service.
8. As per codeshare data, graph plotting will perform to display codeshare to end user.

## Scenario

|  |  |
| --- | --- |
| 1 | Scenario walkthrough – Cosmos DB and Azure functions   1. To make Contoso Air display relevant airline data to the end users, developer needs to build and connect the app with Cosmos DB. 2. Developer should able to store the bulk of data into Cosmos DB database. 3. To do this, developers will run the utility to read bulk of data from JSON file and will store it into the Cosmos DB using Azure functions. 4. The utility will store data into Cosmos DB and after the completion of data storage process customer can access Contoso Air app. |
| 2 | Scenario walkthrough – Cosmos DB   1. For ticket booking Kevin will go on Contoso Air website. 2. Kevin will fill all the details required for ticket booking; like Trip Type, From Date, To Date, Depart Date, Return Date and Passengers count. 3. Kevin clicks on *Find Flight* button based on the above criteria set. Then App will internally interact with Cosmos DB and will bring the desired flights and fares. 4. Kevin will select the appropriate flight as per his requirement. |
| 3 | Scenario walkthrough – Cosmos DB Graph (Gremlin) API   1. Then after, Kevin can click on *Check Codeshare* button to get more information about airlines partners. Application will implicitly fetch the data from Cosmos DB for the selected Airlines. (Scenario 2.2) 2. Kevin will be able to see the result in Graphical format through Cosmos DB Graph (Gremlin) API feature. 3. Finally, Kevin will click on *Book Ticket* Button to complete his Ticket booking process. |

## Technology Used

Cosmos DB: Cosmos DB is used to store and retrieve the Airline Codeshare data.

Cosmos DB Graph API: Cosmos DB Graph API is used for display the codeshare airlines in graphical format.

Azure Functions: Azure functions used in combination with Cosmos DB for Bulk data storage.

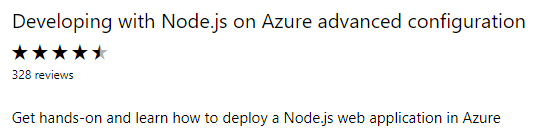
# LAB 5: Submit Feedback using Bing Speech API, Text Analytics API, Emotion API, Azure Function and Cosmos DB

## Details for Immersion Labs

Lab Title: Submit Feedback using Bing Speech API, Text Analytics API, Emotion API, Azure Function and Cosmos DB

Lab Description: Get hands-on and learn how to create application that supports submission of feedback using Bing Speech API, Text Analytics API, Emotion API, Azure Function and Cosmos DB

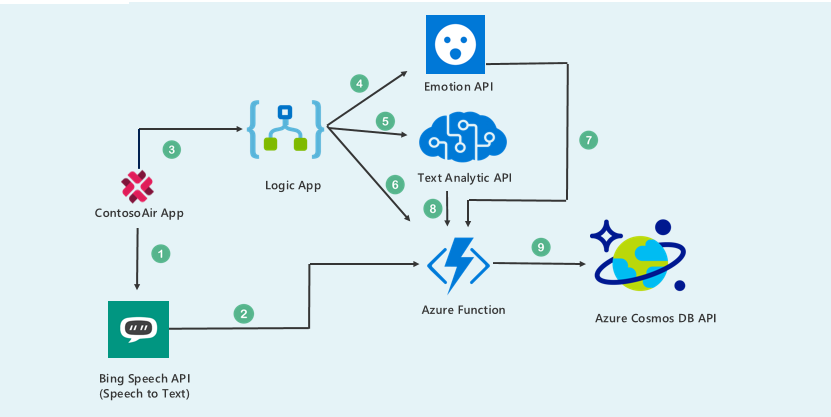
NOTE: The lab title and description will be displayed on Hand-on lab as specified in below screen. Please review and approve the same.



## Problem Statement

Traditionally, it takes a long time to provide ratings and feedback, but Contoso Air always innovates and recently built a fun and easy way to provide feedback.

## Solution



## Work flow

1. Use the voice command for providing the input to the application using Cognitive Service (Bing Speech API) which will convert user voice into the text.
2. Emotional API analyze the image and gives rating based on their facial expression.
3. The Text analytics API will perform the detailed analysis around this comment and provide a Score and broadly categorized it into Good, Bad and Normal rating.
4. Azure Functions is a solution for easily running small pieces of code or functions in the cloud. We can collect the data from different services and store it into the Cosmos DB.

## Scenario

|  |  |
| --- | --- |
| 1 | Scenario walkthrough –Upload the picture and analyze it using Azure Cognitive Service (Emotion API)   1. Kevin booked flight using Contoso Air application. 2. Kevin is travelling from location Seattle to Barcelona, after completing the journey the user wants to provide feedback about his/her recent flight journey. 3. For giving the feedback must open the Contoso Air application form. 4. Instead of following the traditional approach to provide feedback, In Contoso Air application Kevin needs to upload his selfie, after uploading the selfie the Emotion API will analyze that uploaded picture and gives rating based on his facial expression. 5. Emotion API consider his facial expression as rating and provides a point (between 1 to 5) according to his facial expression. |
| 2 | Scenario walkthrough – Provide comments in your voice and Cognitive Service (Bing Speech API) will convert your voice into text.   1. After uploading the picture Kevin can also add comments about his flight journey. 2. Contoso Air application provides a way of adding comments in his own voice instead of manually typing it. He just need to click on the mic symbol and start speaking and once done, again click on the same to finish. 3. Here in this scenario we are using the Cognitive Service (Bing Speech API) which will convert his voice into text. |
| 3 | Scenario walkthrough – Analyze the comment provided by customer using Cognitive Services (Text Analytics API)   1. After providing the feedback using voice command (speech), the text passed to Cognitive service (Text Analytics API) for analysis. 2. The Text analytics API will perform the detailed analysis around this comment and provide a score and broadly categorized it into Good, Bad and Normal rating. 3. All the data related to feedback, rating and score will be stored in database for future analysis. |
| 4 | Scenario walkthrough – Use Azure Function to insert the data of cognitive services and picture.   1. Azure function is a solution for easily running small pieces of code or function in the cloud. 2. In this scenario, we are writing the logic to store the data in the Cosmos DB inside the Azure Function. The advantage of using the Azure Function is we can use the same Function in another lab too, we just need a URL of that azure function for using in another lab. 3. All the data related to the feedback like Customer Image, Comment, Ratings and the score provided by the Cognitive service will be stored in Cosmos database for doing analysis and generation of reports. |

## Technology Used

**Cognitive Service (Emotion API): -** The Emotion API beta takes an image as an input, and returns the confidence across a set of emotions for each face in the image as well as bounding box for the face from the Face API. The detected emotions are happiness, sadness, surprise, anger, fear, contempt, disgust or neutral.

**Cognitive Service (Bing Speech API): -** APIs convert human speech to text that can be used as input.

**Cognitive Service (Text Analytics API): - Text Analytics API** is a cloud-based service that provides advanced natural language processing over raw text.

**Azure Function: -** Azure Functions is a solution for easily running small pieces of code or "functions" in the cloud. You can write just the code you need for the problem at hand, without worrying about a whole application or the infrastructure to run it.

**Cosmos DB: -** Azure Cosmos DB is Microsoft's globally distributed, multi-model database. With the click of a button, Azure Cosmos DB enables you to elastically and independently scale throughput and storage across any number of Azure's geographic regions.

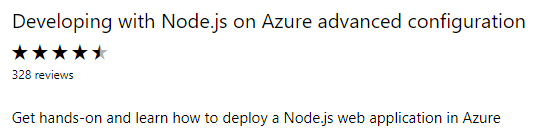
# LAB 6: Azure notification service for Flying solo service(minor’s) (New)

## Details for Immersion Labs

Lab Title: Azure notification service for Flying solo service(minor’s)

Lab Description: Get hands-on and learn how to configure notification using Azure Notification APIs and Cosmos DB

NOTE: The lab title and description will be displayed on Hand-on lab as specified in below screen. Please review and approve the same.

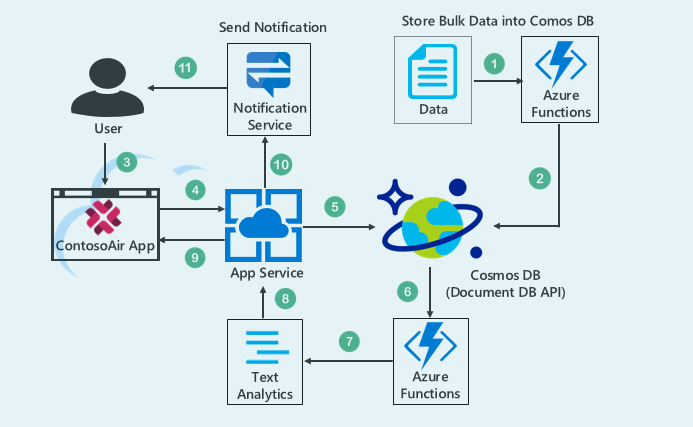


## Problem Statement

Alice, the daughter of Kevin, is a minor and willing to travel from Seattle to Barcelona and wants to avail “Flying Solo Service” provided by Airline. But as a parent, Kevin needs to know and be assured that the child will be taken care of. To solve this problem, we have developed a solution using Azure Notification Hub API.

## Solution

The below architecture provides an overview of Lab 6, which displays the Flying solo service of airline through ContosoAir app, Notification Hub API, Text Analytics API, CosmosDB, Azure Function services.



## Workflow

* + - 1. In first steps, JSON data is fetched from JSON file through data store utility and will be forwarded to Azure functions.
      2. In the next step, Azure functions will process data and store it into the Cosmos DB database.
      3. User will book the ticket with “Flying solo service” of airline through ContosoAir application.
      4. Guardians will be provided with the review and personal details of the ground staff by sending a notification through Notification API**,** 24 hours before boarding the flight.
      5. The review of the ground staff will be calculated through Analytics API based on the feedback provided by past passengers. Personal details will be fetched through Azure Functions from CosmosDB.
      6. Time-to-time check-in notification during departure of minor will be sent through Notification API to the guardian.
      7. The same workflow will be followed during transit and arrival at the destination.
      8. The review and personal details of the ground staff at destination will also be provided to guardian through Notification Hub API.
      9. Guardians can even give the feedback of the ground staffs (Boarding point, Transit point and Arrival point) depending on their personal experience.
      10. The provided feedback will be analyzed through Text Analytics API and will be stored into CosmosDB database.

## Scenario

|  |  |
| --- | --- |
| 1 | Scenario walkthrough – Cosmos DB and Azure functions   1. To make ContosoAir display relevant airline data to the end users, developer needs to build and connect the app with Cosmos DB. 2. Developer should able to store the bulk of data into Cosmos DB database. 3. To do this, developers will run the utility to read bulk of data from JSON file and will store it into the Cosmos DB using Azure functions. 4. The utility will store data into Cosmos DB and after the completion of data storage process customer can access ContosoAir app. |
| 2 | Scenario Walkthrough – ContosoAir App   1. Kevin goes to ContosoAir app and books a ticket with “Flying solo service” 2. He selects the Source, Destination and Airline with which the Alice has to travel. 3. Thus, completes his ticket booking process, the booking details will be stored into Cosmos DB |
| 3 | Scenario walkthrough – Notification API   1. Kevin will receive the notification of ground staff information involved in Flying solo service. 2. 24 hours before Alice(minor) boarding the flight, at Transit point and at Arrival point Kevin will receive the notification of Alice’s status at the Airport through Notification API. |
| 4 | Scenario walkthrough – Text Analytics API   1. All the reviews of ground staff will be calculated through Text Analytics API and will be sent to Kevin. 2. Kevin will also be able to give the feedback of their personal experience which will be calculated through Text Analytics API and will be stored into CosmosDB and the same data (review) will be used for future reference. |

## Technology Used

Cosmos DB: Cosmos DB is used to store and retrieve the data.

Notification Hub API: It is a mobile push notification engine for quickly sending millions of notifications to iOS, Android, Windows or Kindle devices.

Text Analytics API: It is a service that provides advanced natural language processing over raw text.

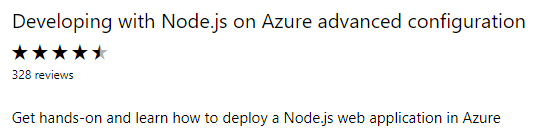
# LAB 7: Azure notification service for tracking luggage status (New)

## Details for Immersion Labs

Lab Title: Azure notification service for tracking luggage status

Lab Description: Get hands-on and learn how to configure notifications using Azure Functions, Notification Service APIs and Cosmos DB

NOTE: The lab title and description will be displayed on Hand-on lab as specified in below screen. Please review and approve the same.

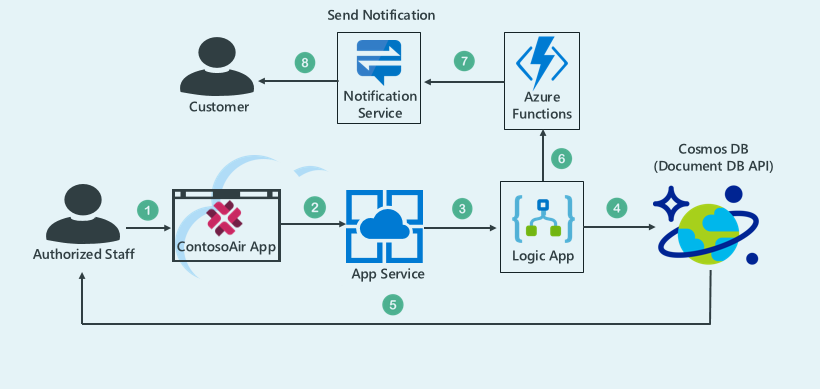


## Problem Statement

Kevin was travelling from Manchester Airport to Delhi Airport by using connecting flights where two different airlines were involved, both the airlines had a tie-up. The first flight landed at the Dubai Airport which was hopping airport from which Kevin had to board the second flight, he was intimated about the fact that once he drops his luggage for the first flight that is from Manchester airport he will get the luggage at the destination that is Delhi Airport, and there is no need for him to individually collect his luggage from the first flight and re-drop the luggage for the second flight at the Dubai airport. But when he reached at the Delhi airport he did not receive his luggage, then he contacted the concerned official about the loss of his luggage, there was no track about his luggage and was declared as lost, as both airlines involved were blaming each other for the loss of luggage. We have solved this problem by using Azure Function which updates the luggage data into CosmosDB at every point in his travel.

## Solution

The below architecture provides an overview of Lab 7, which is used to track passenger’s luggage and sends notification to user through ContosoAir app Notification Hub API, Logic App, CosmosDB, Azure Function and App service.



## Workflow

1. User has booked the Ticket from ContosoAir website.
2. After reaching the Airport, the baggage is scanned and the airport authority check-ins his luggage through their system. This process continues at every point of his journey.
3. Every single time when the luggage gets checked-in the Logic App triggers the action and updates the luggage status into CosmosDB database.
4. Once the action is triggered in the Logic App, Azure Functions fetches the latest updated data and sends it to the Notification API.
5. As soon as the data is received by the Notification API, the updated status of luggage is send to the user.
6. This process of sending notifications also continues at every point of his journey.

## Scenario

|  |  |
| --- | --- |
| 1 | Scenario walkthrough – Logic app   1. Kevin arrives at the airport and the Check-in of the luggage is done through the airport authority. 2. Once the Check-in is done, Logic App triggers an event to store the updated data into the CosmosDB database. |
| 2 | Scenario walkthrough – Azure Functions   1. Azure Functions fetches the updated data from the CosmosDB database once the Logic App is triggered. 2. This updated data is then send to Notification Services. |
| 3 | Scenario walkthrough – Notification Services   1. Every time the Logic App is triggered the updated data in the Notifications Services will be send to Kevin for tracking the details about his luggage status. |

## Technology Used

Cosmos DB: Cosmos DB is used to store passenger’s luggage details.

Logic App: Logic API is used to consume Azure functions.

Azure Functions: Azure functions used to trigger action of sending notification

Notification Hub API: Notification Hub API service is used to send notification.

NOTE: The above stories may change, if scenarios does not fit the suggested technologies. Click2Cloud will inform Microsoft, if there is any change in the scope due to said technology limitations.

# Deliverables and Milestones

Microsoft must complete sign this document before Sep 08 to meet the below milestones on time. The milestone delivery schedule for the services, will be as follows:

| # | Milestone | Due Date | Deliverables |
| --- | --- | --- | --- |
| 1 | Requirement Analysis and requirement documentation | Sep. 06, 2017 | Requirement Document |
| 2 | Immersion Lab development complete, deployment & Setup | Sep. 30, 2017 | Contoso Air code modified to support labs  Cosmos DB data and data definition  ARM templates as per immersion labs standards |
| 3 | Testing, Lab documentation | Oct. 06, 2017 | Labs documentations |
| 4 | Immersion Lab UAT & bug Fixing | Oct. 13, 2017 | Test environment setup for UAT  Bug fixing and stabilization |
| 5 | MTC-0 Release | Oct. 13, 2017 | All documents specified in MTC-0 Demo extension under [Scope](#_Scope) section |
| 6 | Support | Oct. 20, 2017 | As mentioned in support of [Scope](#_Scope) section |

# Disclaimer

This **DRAFT** specification is being forwarded to you strictly for informational purposes and sign-off requirement before development starts. This document covers functional and technical requirement of the proposed labs for immersion platform. The specification is "AS IS," "WITH ALL FAULTS" and Click2Cloud makes no warranties, and disclaims all warranties, express, implied, or statutory related to the specifications. THE CORPORATIONS ARE NOT LIABLE FOR ANY INCOMPLETENESS OR INACCURACIES. THE CORPORATIONS ARE NOT LIABLE FOR ANY CONSEQUENTIAL, INCIDENTAL, OR INDIRECT DAMAGES RELATING TO THE SPECIFICATIONS OR THEIR USE.

# Appendix A: Glossary

|  |  |  |
| --- | --- | --- |
| No. | Initial Name | Description |
| 1 | CR | Change Request |
| 2 | FY | Fiscal Year |
| 3 | VSTS | Visual Studio Team Services |

# Appendix B: Document Change History and Sign-off

|  |  |  |  |
| --- | --- | --- | --- |
| Version No. | Date | Name | Sign-Off by |
| V1.0 | 09/06/2017 | Tara Shankar Jana <tarasha@microsoft.com> |  |

# Point of Contact

|  |  |  |
| --- | --- | --- |
| Name | Designation | Email |
| Shrutika Mendhe | Sr. Lead Software Design Engineer | [shrutika.mendhe@click2cloud.net](mailto:shrutika.mendhe@click2cloud.net) |
| Gaurav Dhuwe | Lead Software Design Engineer | [gaurav.dhuwe@click2cloud.net](mailto:gaurav.dhuwe@click2cloud.net) |
| Sandeep Thakre | Project Manager | [sandeept@click2cloud.net](mailto:sandeept@click2cloud.net) |
|  |  |  |
|  |  |  |