|  |
| --- |
| OpenHack – NoSQL |

# Overview

Microsoft’s OpenHack series is a three-day immersive, hands-on, challenge-driven hack that brings together developers from across the ecosystem and Microsoft to tackle scenarios influenced by common, real-world problems using Microsoft platform capabilities and other industry leading technologies.

Organizations are moving to the cloud in record numbers. Part of the driving force behind these cloud migrations is a desire to modernize data platforms to accommodate an ever-increasing amount of data coming in from mobile applications, social media and IoT devices as well as a desire to capitalize on the global capacity of hyper-scale cloud providers such as Azure. The OpenHack NoSQL event will challenge developers and data professionals to migrate a legacy database application to the cloud and modernize it to accommodate new business features. They will continue the path by scaling out their solution for global availability and developing new functionality to provide additional insights to the business and value to customers.

# Technologies

Developers will leverage Microsoft’s Cosmos DB, Azure SQL Database, Azure Database Migration Service, Azure Web Apps, Azure Functions, Azure Event Hubs, Global distribution with Cosmos DB and the Cosmos DB Graph API.

# Challenges

**Challenge 1: Migrate**

* Migrate an existing IIS/SQL based on-premises solution to Azure Web Apps and Azure SQL Database
* Identify and correct potential migration blockers

**Challenge 2: Choosing the Right Store**

* Update the application to accommodate new social features
* Implement new data store in Azure Cosmos DB

**Dev**

* Generate Resumes and User data for Migration form Blob Store to Cosmos
* ADF can be used to move SQL to Cosmos Json
* Any Roes from SQL will need to be de-normalized to accommodate JSON (ADF-SP)
* Application Refactor? Stored Procedures will to be converted to JavaScript in cosmos
* Connectivity modifications/Additions to accommodate Azure Cosmos SDK
* Integrate Resumes into UI – Client services and DAL to Cosmos DB with all CRUD commands
* Integrate Company Ratings into the application UI (multiple views) – Client Services and DAL/CRUD
* Create Ratings for existing comapnies
* Copy application to new repo and remove DAL Layer and test to make sure the application will run – notate where code may need to be commented out for functionality – update in resources and instructions

**Challenge 3: Integrating Realtime**

* Establish new real-time data ingestion functionality using Azure Event Hubs

**Dev**

* Generate fake job posting for companies
* Application to feed event hub job postings
* Logic app can read from event hub, or write Azure function to read from event hub and write to Cosmos, with correct referential information for each company.
* ? In Application UI – make any changes needed to observe this happening if necessary.

**Challenge 4: Extending Functionality with Serverless**

* Configure real-time updates via Cosmos DB Change Feed
* Extend the functionality of the application with Azure Functions

**Dev**

* Cosmos Change Feed is on by default
* Cosmos Change Feed is a separate SDK that will need to be implemented in Azure function to listen for changes in the feed.
* Need to determine what to monitor in the change feed and what to do about it.
* Then build AZ Function or other tools to monitor feed -Is this observable only in Portal? UI updates?

**Challenge 5: Building for Global Scale**

* Scale out the application to support customers in multiple regions
* Build out the Cosmos DB global distribution

**Dev**

* Multi-region support is done in the portal, Azure Traffic manager will need to be implemented for application
* Technically this is part of the last step

**Challenge 6: Gaining Insights from Data**

* Recommendation engine on your data with Cosmos DB.

**Dev**

* Depending on the data, this will be a Gremlin API Exercise, which is also a significant dev effort
* The data is not stored as Graph (verticies, edges, properties) so a graph database will probably have to be created with new data derived from whatever data we have in step 4 , which means the resumes, the jobs and a hiring history of past applicants will have to be valid or no connections can be made. This is likely 2 or 3 days worst case, relationships in data will need to be manually created.
* The students will ideally import graph data with ADF into a new Graph Database and just query it.