NoSQL OpenHack: Content Refresh

**Game Plan**

Contents

[Background 3](#_Toc23322400)

[Deliverable 3](#_Toc23322401)

[Deliverable requirements: Performance Metrics & other measurements 3](#_Toc23322402)

[Availability SLA 3](#_Toc23322403)

[Cost Sensitivities and Factors 3](#_Toc23322404)

[Rollout Strategy 3](#_Toc23322405)

[Disaster Recovery 4](#_Toc23322406)

[Proposed Content Structure, Architecture & Technology Decisions 4](#_Toc23322407)

[Overview 4](#_Toc23322408)

[Challenge 1: Pick a noSQL DB 4](#_Toc23322409)

[Challenge 2: Migrate Your Data and Model for noSQL 4](#_Toc23322410)

[Challenge 3: Stream Data to noSQL 5](#_Toc23322411)

[Challenge 4: Change Feed and Materialized Views 5](#_Toc23322412)

[Challenge 5: Azure Search Integration 5](#_Toc23322413)

[Challenge 6: Global Expansion, Indexing, and Logging 6](#_Toc23322414)

[Challenge 7: Insights from Graph 6](#_Toc23322415)

[Engineering Fundamentals 6](#_Toc23322416)

[Leveraging Prior Work 7](#_Toc23322417)

[Sharing Plan 7](#_Toc23322418)

[Sprint Schedule 7](#_Toc23322419)

[Success Criteria 7](#_Toc23322420)

[Risks, Issues 8](#_Toc23322421)

[Additional Asks 8](#_Toc23322422)

[Team 8](#_Toc23322423)

[Ready to Start? 8](#_Toc23322424)

# Background

The NoSQL OpenHack was created originally in FY19 and delivered in July (of that fiscal year). The content was re-reviewed by the CSE Data Tech Domain and the CSE Programs Team and P0 issues were raised with the quality of the content against OpenHack content quality standards/requirements. A v-team was developed to re-scope and re-develop the NoSQL content so that it could be delivered at high-quality to S500 customers across the globe.

# Deliverables

Our deliverable will be a finished repo in the OH project within ADO with the following finished, high-quality items:

* Challenges
* Coach materials (i.e. coach solution guide, at the least)
* Any setup scripts required (i.e. ARM template for Azure sub resource provisioning)
* Any infrastructure required (i.e. DB for catalog)
* OH Summary

Our second-tier deliverable will be a test classroom that successfully spins up this necessary hacking environment in the OpenHack portal web application.

# Deliverable requirements: Performance Metrics & other measurements

The deliverable standards will adhere to Slide 14 in [this PPT deck](https://microsoft-my.sharepoint.com/:p:/p/nirshah/EcEeg9wRs0BMoXGm-X9rsy0Bk5iGZwaGZMxFKxVvlTn74Q?e=k6YgEl).

## 

## Cost Sensitivities and Factors

What are the costs of having any infrastructure, i.e. the database, up and running for the OpenHack? Are there any unusual resources that will be used in Azure subs that will result in high costs?

<<Basu>> Once we identify all the services and infrastructure needs based on the challenge content it becomes easier to estimate for this.

## Rollout Strategy

New minor features (i.e. typos updated in the content) will be pushed through set-up build and release pipelines into the OpenHack portal automatically. If major features are needed (i.e. new JSON file for challenge structure) then the Tech Lead will reach out to [openhacks@opsgility.com](mailto:openhacks@opsgility.com) that a new major version change is being released.

## Disaster Recovery

<<Basu>> Basically we need to handle DR in two ways..

* + Automated deployments so that we can deploy new instances easily.
  + Any pre-configured data, databases etc. Will need to be backed up so that they can be restored using automated deployment.

# Proposed Content Structure, Architecture & Technology Decisions

Attendees will have the option to leverage several services available in Azure such as Cosmos DB including the core SQL and graph APIs, Azure SQL Database, Azure Data Factory, Azure Functions, Azure Event Hubs, and Azure Search. All attendees will be given an Azure SQL Database to start with. They are at liberty to choose any technology they would like that meets the success criteria for each challenge.

### Overview



Contoso Movies, Ltd. has expressed their desire to move to a more modern and cloud-based approach to their online e-commerce presence. They currently have an ASP.NET web application that provides a store front allowing users to navigate their movie catalog, view movie details and purchase movies. The movie catalog, user profiles and e-commerce transactions are all stored in a Microsoft SQL Server database.

They would like to unlock new, innovative capabilities in the cloud, starting with migrating their relational database to a NoSQL database that can provide them greater scalability, more flexibility in the handling of concurrency, a simplified ability to scale geo-graphically and benefit from greater schema agility.

### Challenge 1: Plan and prepare for migration

* Open the deployed website(The website and database will be deployed for them in the provided environment)
* (The goal is to provide an environment with all of these artifacts provisioned and pre-loaded)

Create and document an overall plan for the target state of the migrated solution that meets the business and technical requirements and addressing:

* Where will the website be deployed?
* What NoSQL database will be used?
* High-level plan for how to map the relational tables to NoSQL document collections.

requirements

* The database needs to support the event sourcing pattern where changes to the data store trigger events that can be processed by any number of listening components in near real-time

Create the initial resources in Azure to host the web app (e.g., Web App Plan) and database (e.g. Cosmos DB Account).



### Challenge 2: Migrate the catalog database

Given some business requirements (query patterns, usage, data model etc.) migrate the movies catalog and transactions to the NoSQL database to establish a baseline model.

Verify that you have a proof of concept solution working, albeit not optimized. You will be provided with conceptual queries that validate migration success. You will perform deeper schema design and optimization in a subsequent challenge as you prepare your solution for production.

At this point:

* Have selected appropriate partitioning strategy based on initial observations of data models and query patterns



**Artifacts provided to attendees for this challenge:**

* No additional artifacts

### Challenge 3: Optimize NoSQL design

We provide a set of specific query patterns that Contoso has explained they want to make sure continue to work in the new design, are performant and are cost-optimized for the anticipated query volumes.

* Catalog browsing:
  + Retrieve list of movies
  + Paging thru lists of movies
  + Retrieve details for a specific movie
  + Filtering movies by category
* Purchase transactions
  + Adding a movie to the shopping cart
  + Completing a purchase transaction

Contoso currently has X million concurrent users during peak time, but would like to plan for Y million concurrent users.

Perform sizing exercise to determine anticipated scale requirements. Optimize schema design to reduce scale requirements if possible.

For example, attendees might:

* Explore Autopilot for Cosmos DB.
* Denormalize datasets
* Duplicate data and store with different sort orders or keys
* Adjust indexing
* Adjust partition key selected
* Evaluate their choice of consistency model

**Artifacts provided to attendees for this challenge:**

* No additional artifacts



### Challenge 4: Events are the center of our universe

Contoso would like to add new feature to the solution that collect events from user behavior, as users interact with the web app. This telemetry would initially be used for understanding patterns of behavior, but in the longer term they foresee using it to build a movie recommendation system.

These events will stream will be streamed in near-real time from the website to the back end that collects them, which should be able to handle high volumes of ingest with minimal latency and, if possible, no loss of events.

Contoso would like to create report style views summarizing the activity of users on the website. As new events arrive they would like to see some of the statistics update in near real time. Other statistics they want to update during specific windows, like hourly and daily.

Examples of these statistics include:

* Count of activity (number of “view details”, “add to cart” and “purchase” events) for this hour
* Summary of activity (number of “view details”, “add to cart”, “purchase” events and revenue) for this day so far
* Top 10 most popular movies purchased of all time
* Most popular movie categories
* Others – use your imagination on what data points would be interesting for Contoso

Additionally, Contoso would like ensure they have a layer of caching the reduces reads against the NoSQL store when the commonly retrieved movies are accessed. In their current deployment, they had explored doing this but got stuck figuring out how they could invalidate movies in the cache whose details had been changed, and do that in response to the change. They were particularly concerned with ensuring that the cache would be properly adjusted, no matter how the change to the movie details was made (e.g., thru an admin website or a direct edit against the database). This can be tested by attendees by executing queries against the cache, update one or more catalog items, then re-execute their cache query to ensure the cache was refreshed.

For example, attendees might:

* Use the Change Feed from Cosmos DB, responding with an Azure Function that forwards the data point to an Event Hub instance. The Event Hub is consumed by an Azure Stream Analytics windowed query whose summarized output is written back to Cosmos DB, to Power BI or to another endpoint that supports the display of the aggregated data.
* Alternately, they might create a materialized view for the aggregate statistics. Again, using the Change Feed on Cosmos DB, they would use an Azure Function to process each batch of events. This batch of events is grouped and aggregated. The current statistics are queried and the aggregated results are added to the current values and saved back to Cosmos DB in a materialized view.
* They could create visualizations using Notebooks.
* Use the Change Feed to invalidate or update items in the cache when movie details are changed.

**Artifacts provided to attendees for this challenge:**

* Event generator: We provide a data generator (as a simple, easy to customize script) that simulates a user load and writes the events to Event Hubs.



### Challenge 5: Improving the search experience

Contoso has mentioned that their customers have expressed frustration that their interface only allows them to find movies by browsing the recent or top movies, or browsing by category. Customers have indicated they would love to be able find movies by title or even better by their synopsis. Contoso developers have mentioned they could easily accomplish this with the full text search capabilities Microsoft SQL Server, but when it comes to NoSQL databases they are uncertain how to proceed. Beyond full-text search, they are interested in seeing if they could add other filters to help users quickly narrow the results.

For example, attendees might:

* Integrate Azure Cognitive Search (or ElasticSearch) to index the title and description fields of the movie data, and allow for full text search against these fields.
* Add faceted search by enhancing the Azure Search Index to index other fields like release date and category.
* Test search API via service interface or a tool like Postman.



**Artifacts provided to attendees for this challenge:**

* No additional artifacts

### Challenge 6: Taking over the world (MUAHAHAHA)

With the improved search experience, orders are coming in faster than ever and Contoso wants to expand globally. They would like to start by having their solution available in a performant way in two regions (e.g., the United States and Europe). They do not want users in any region to experience any form of latency resulting from making network requests that span an ocean. They also, do not want to have to explicitly manage how the two regions stay in sync.

For example, attendees might:

* Verify the that the account is configured with multi-master.
* Add a new region to Cosmos DB.
* Create new resource groups and deploy any related services in the new region.
* Configure the services to only use the Cosmos DB instance within the preferred region.

**Artifacts provided to attendees for this challenge:**

* Generator that can be customized to send events to their application. Sends the same 100 events to each region. At the conclusion, there should be 100 events in each region. Not 200 in one region, as measured using the Azure portal and Cosmos DB metrics.







# Engineering Fundamentals

Briefly describe how your project will address:

|  |  |
| --- | --- |
| Version Control (including data science artifacts, if applicable) | Using OH project in ADO with new branch of pre-existing repo (branch off of master) ([link](https://dev.azure.com/cseeest/openhack/_git/NoSql_App_Modernization?path=%2F&version=GBrefresh)) |
| Work Item Tracking (user stories, acceptance criteria, tasks, bugs) | Using Kanban Board in ADO ([link](https://dev.azure.com/cseeest/OpenHack/_boards/board/t/NoSQL%20Team/Features/)) |
| CI/CD (including measuring velocity) | Using ADO |
| Code Reviews (Frequency, Triggers) (including data science artifacts, if applicable) | Weekly sync set-up where code reviews will take place |
| Agile/Scrum (who’s the scrum master? Who is the product owner?) | Basu Nagarahalli (Scrum Master), Nirali Shah (Product Owner) |
| Retrospectives (Frequency, Audience) | \*only for the Dry Run/Preview\* 1 Retro every day of the Dry Run/Preview (3 total) using the Retrospectives tool in ADO |

# Leveraging Prior Work

We are using the previous NoSQL project as a baseline, but starting from scratch in development ([link](https://dev.azure.com/cseeest/openhack/_git/NoSql_App_Modernization?path=%2F&version=GBrefresh)). We will branch off of this pre-existing project so that we can use pre-existing JSON lab definition files (and just modify them, rather than build from ground-zero).

# Sharing Plan

The content will be shared through the OpenHack knowledge transfer platform – MSFT-wide and with customers.

# Sprint Schedule

|  |  |  |  |
| --- | --- | --- | --- |
| **Sprint** | **Estimated start/end date** | **Type** | **User stories/Focus/Deliverable** |
| 0 | 10/28-11/4 | Planning | Solliance Onboarded, Finished Game Plan, Kanban board set for Sprint 1 |
| 1 | 11/4-11/11 | Dev. | Start dev. Challenges/Coach Materials |
| 2 | 11/11-11/18 | Dev. | Continue dev. Challenges/Coach Materials |
| 3 | 11/18-11/25 | Dev. | Continue dev. Challenges/Coach Materials |
| 4 | 11/25-11/28 | Dev./Break | Challenges + Coach Materials, Setup Test Classroom and complete initial testing/QA by EOW |
| 5 | 12/2-12/9 | Testing | Dry Run/Preview |
| 6 | 12/9-12/16 | Dev. | Scope/plan necessary updates to the Challenges/Coach Materials |
| N/A | 12/16-1/3 | Break | N/A |
| 7 | 1/6-1/13 | Dev. | Make changes to Challenges/Coach Materials, Finalize OH Summary |
| 8 | 1/13-1/29 | Deliver | First Delivery |

# Success Criteria

Describe how and when we measure success, both as defined by the customer and by ourselves. Some measures may be realized after the Dev Crew engagement end.

|  |  |  |
| --- | --- | --- |
| **Metric** | **When** | **Value (range)** |
| *MBOM completed/published* | By sprint 4 (before Thanksgiving 11/28) | Test classroom created and MD files showing in correct format on the OH Portal |
| *London NoSQL OH is delivered* | Week of 1/27 (1/27-1/29) | Positive feedback is received from customers who attend |

# Risks, Issues

|  |  |
| --- | --- |
| **Risk / Issue** | **Mitigation** |
| Content quality is not up to par for customers due to a rushed delivery (not enough Sprints) | London 12/2 delivery is pushed back to later date and if NoSQL must be used (old content used and labelled as a “workshop”) |

# Additional Asks

Would you benefit from deep dives on roadmaps and limitations with service engineering teams?

N/A for now

# Team

Who’s doing what? (include tech lead, shadow tech lead, technical focus/responsibility; scrum master, product owner and other process roles; and members of customer’s engineering team)

|  |  |  |
| --- | --- | --- |
| Name | Role | Technical focus/responsibility |
| Justine Cocchi | Tech Lead | Manage dev. team, help with technical blockers in dev., ensure quality of content is at high-quality |
| Nirali Shah | Product Owner/PM | Overall PM coordinating efforts from Content Creation to > Dry Run > delivery |
| Solliance | Engineers | Develop challenges/coach materials, activate content in portal, and test content in functional/QA testing + Dry Run capacities |
| Basu Nagarahalli | Scrum Master | Manage sprints to ensure Partner is on track for content development |
| Tim Sander | Shadow Tech Lead | Reviews content quality and technical caliber, helps with planning and coach sourcing |
| Cosmos DB Product team | Reviewers | Reviews content quality and technical caliber |

# Ready to Start?

|  |  |
| --- | --- |
| Criteria | Status |
| Partner/vendor committed to content development | Done |
| SOW/PO Signed | In Progress |
| SE/I or SE/DWR committed to be engaged through customer handoff | N/A |
| Customer engineering team committed and ready to go | N/A |
| Customer Azure subscriptions deployed and tested | N/A |
| Code repo accessible to project engineers | N/A |
| Customer network/security provisions for project engineers enabled | N/A |
| Project data is accessible and usable | N/A |
| AETHER/Ethics review completed | N/A |