

Polyglot Storage

Lab Document

Overview

In this scenario based HOL, you will learn how to build a 'polyglot persistence' data pattern that is common in modern cloud hosted applications. Requirements of modern applications, such as, greater scale and availability, have driven the industry to begin using a much broader range of technologies for storing data within an application. Microsoft Azure provides a range of storage technologies that support these architectures and this HOL provides an example of the use of these in the well understood scenario of e-Commerce. With data services in Microsoft Azure, you can quickly design, deploy, and manage highly-available apps that scale without downtime and that enable you to rapidly respond to security threats. Features built into services like Azure SQL Database, Azure Search, and Azure DocumentDB help your apps scale smartly, run faster, and stay available and secure.

In this HOL you will see a browser based e-commerce application running under the LCA approved sample company name 'AdventureWorks'. It has been created to demonstrate functionality provided by the following data storage technologies (SQL Database, DocumentDB, Search, Table Storage). In a real application, decisions will need to be made as to where data is stored. In this HOL we wish to highlight; how using multiple Azure data service technologies allows you to take a modern approach to data in your applications.

Note- The website (which gets built out of this HOL) is not intended to be a fully functioning site. It is not designed to be a reference e-commerce implementation nor a starting point for a customer's implementation of an e-commerce site on Azure, rather it will provide the following functionality in order to demonstrate the selected storage technologies.

In the course of this lab, you will gain greater familiarity with Azure SQL Database, DocumentDB, Azure Search and Table Storage through performing the following tasks:

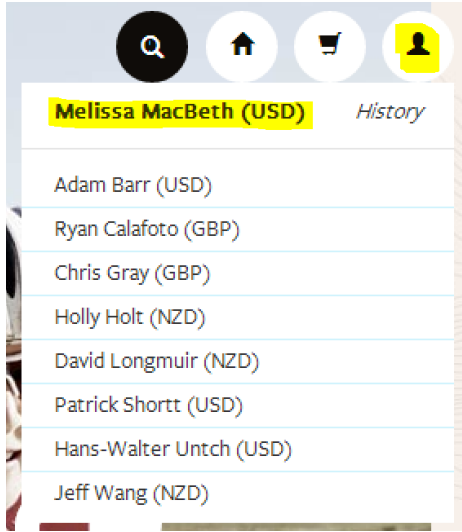
- Familiarize yourself with one of the tenant-company's websites and its Azure SQL Database backend.
- Create a new database using the Azure portal.
- Configure and implement vertical scaling by increasing the capacity of a database.
- Use Azure SQL Database auditing features to track down an erroneous deletion from a database.
- Use Azure SQL Database point-in-time restore to correct the deletion (Optional)
- Configure and implement Azure SQL Database geographic disaster recovery to prevent large-scale data loss.
- Locate data using Azure Search.
- Modernize and create an interactive experience using DocumentDB.
-

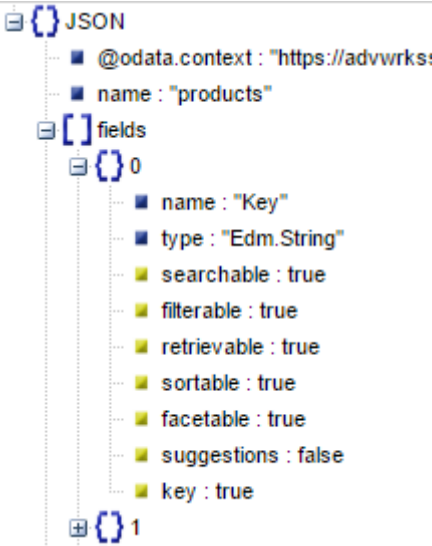
Solution overview




Demo Story Elements	Completed (Yes/No)
Scenario 0: Azure Search – Index Definition Discuss how we maintain a product index within Azure Search. Contained within this index, we store everything to show you the products on this home page. Setting up a new index and getting started is simple.	
Scenario 1: Azure Search – Indexing Documents Discuss how we index documents to Azure Search, and how we can define indexers to do this for us.	
Scenario 2: Azure Search – Search Suggestions Discuss how we use out of the box Azure Search to provide search suggestions to the user.	
Scenario 3: Azure Search - Facets Discuss we use Facets provided by Azure Search to provide us with refiner items.	
Scenario 4: Azure Search – Scoring Profiles Discuss we can use scoring profiles to manipulate the priority of search results.	
Scenario 5: DocumentDB Discuss how we can rapidly produce new applications using DocumentDB.	
Scenario 6: DocumentDB – Schema Free Query Discuss how DocumentDB is well suited to query across schema free attributes.	
Scenario 7: DocumentDB- Nested Documents/Consistency Discuss how DocumentDB supports tunable consistency.	
Scenario 8: DocumentDB - Transactions Discuss how DocumentDB provides transaction processing over multiple documents.	
Scenario 9: Table Storage Discuss Azure Table storage.	
Scenario 10: Azure SQL – Transaction Data Store Discuss how we can use Azure SQL to store transactional records.	
Scenario 11: Azure SQL – Predictable Performance Discuss how we can easily upscale or downscale the performance of Azure SQL.	
Scenario 12: Azure SQL – Database Recovery Discuss how Azure SQL automatically stores Point in Time restore points, and how we can leverage them to undo mistakes.	
Scenario 13: Azure SQL - Auditing Discuss how we can store the audit log to an Azure Storage account.	
Scenario 14: Azure SQL - Vertical Scaling Discuss how the website is anticipating a big sale and how that might require some increased capacity. Simulate some load using the mass order generator, and then show the spike in resource utilization. From the portal, show how easy it is to scale up by changing the size of an instance (for example, from B1 to S3) and how quickly the change is implemented. Re-run the mass order generator and show how the increased capacity has significantly improved the site's ability to handle a spike in sales.	

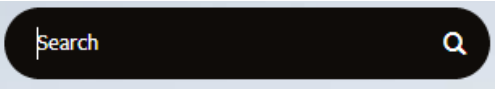
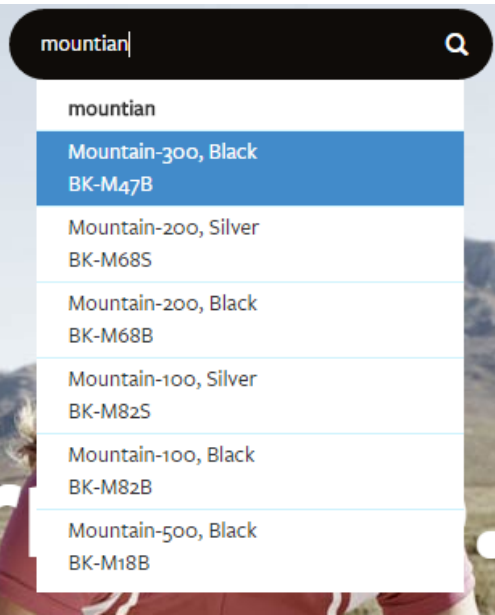
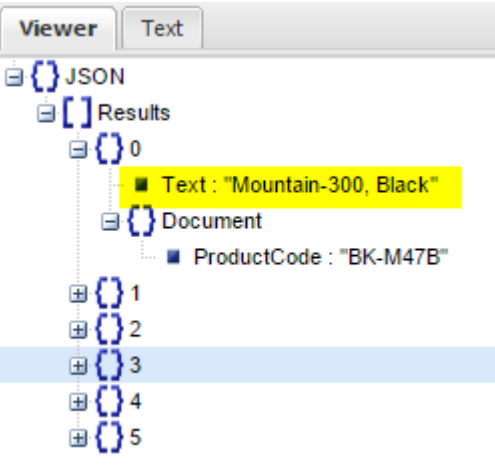
Conclusion/Summary	
	TOTAL

Screen	Click Steps
DEMO SETUP	
	<p>Open Browser to http://inbuildxxxx.azurewebsites.net (inbuildxxxx is what you used in the deployment. "in" is your initials and buildxxxx is you azure pass ID).</p>
	<ul style="list-style-type: none"> • Select Melissa MacBeth as the customer by <ul style="list-style-type: none"> ○ clicking on the person icon ○ selecting Melissa MacBeth



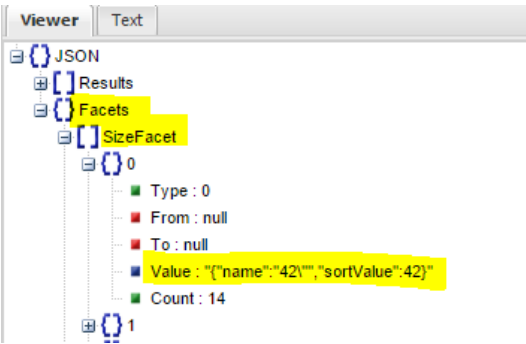
Screen	Click Steps	Demo Script
Scenario 0: Azure Search – Index Definition		
 <pre> { "@odata.context": "https://advwrks...", "name": "products", "fields": [{ "name": "Key", "type": "Edm.String", "searchable": true, "filterable": true, "retrievable": true, "sortable": true, "facetable": true, "suggestions": false, "key": true }] } </pre>	<ol style="list-style-type: none"> 1. Open browser http://inbuildxxxx.azurewebsites.net 2. Open a new browser window and navigate to http://jsonviewer.stack.hu/#http://inbuildxxxx.azurewebsites.net/Api/Discover/AzureSearchSchema/Products 3. Click Viewer in the top left hand corner 4. Expand Out Fields Collection 	<p><i>This is a JSON that defines the schema of our search index for Products.</i></p>
	<ol style="list-style-type: none"> 5. Navigate to http://inbuildxxxx.azurewebsites.net/Api/Discover/Index 	<p><i>Let's see how we create the index.</i></p> <p><i>This is a JSON representation of the Index object that is created in the AdventureWorks code. This is then passed into the Azure Search Client.</i></p>
<pre> public async Task<bool> CreateIndex(params ScoringProfile[] scoringProfiles) { var index = BuildIndex(scoringProfiles); if (await CheckIfIndexExists()) { return true; } var response = await _searchClient.Indexes.CreateAsync(index); return response.StatusCode == HttpStatusCode.OK; } </pre>		<p><i>Build Index creates the index object as seen above. This is then passed into CreateAsync.</i></p> <p><i>This index is created by using some custom attributes on our models, which then builds up the index object to pass to Azure Search.</i></p> <p>https://msdn.microsoft.com/en-us/library/azure/microsoft.azure.search.searchserviceclient.aspx</p> <p>Source Code: MSCorp.AdventureWorks.Core->AzureSearchClient.cs->Line 80</p>


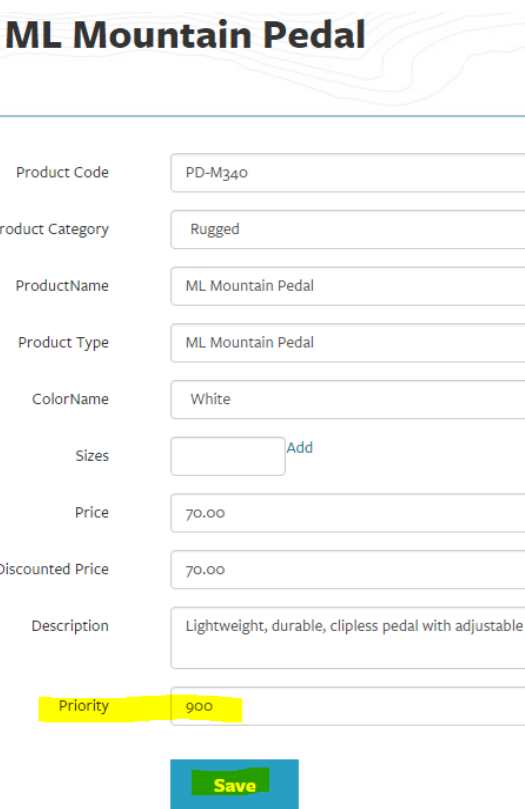
Screen	Click Steps	Demo Script
Scenario 1: Azure Search – Indexing Documents		
	1. Navigate to the Search Box.	
	2. Type Mountain into the search box and press enter	<p><i>You will notice that there currently are no products being displayed on the website.</i></p> <p><i>We need to index some documents to the search index. There are two ways to do this.</i></p>
<pre>public async Task<bool> PostDocumentsAsync(IEnumerable<T> documents) { using (var indexClient = _searchClient.Indexes.GetClient(_indexName)) { var response = await indexClient.Documents.IndexAsync(IndexBatch.Create(documents.Select(doc => IndexAction.Create(doc)))); if (response.StatusCode == HttpStatusCode.OK) { return true; } } }</pre>		<p><i>We can upload documents as and when we need to, we are passing an object that represents the document and then indexing it through the Azure Search Client.</i></p> <p><i>You can either use a full model (class) which matches your index type. Or you can instantiate a Document object which allows you to build up the document with key value pairs. (This allows for partial updates where you only need to update one field).</i></p> <p><i>To do an update. You would call <code>indexClient.Documents.ReplaceAsync</code>. The key must be provided in the object.</i></p> <p><i>But there is a more automated way, where no code is required.</i></p> <p><i>Source Code: <code>MSCorp.AdventureWorks.Core->AzureSearchClient.cs->Line 200</code></i></p>
	3. Navigate to http://inbuildxxxx.azurewebsites.net/Api/Discover/Indexer	<p><i>We can create an indexer which is scheduled task that will look at a datasource and every x minutes and index them into the Search Index.</i></p> <p><i>To create the Indexer. You must first define a datasource, and then define an indexer that uses that data source.</i></p> <p><i>This returns a JSON that represents the DataSource and Indexer objects used in AdventureWorks for the products indexer.</i></p>

Screen	Click Steps	Demo Script
	<ol style="list-style-type: none"> Navigate to http://inbuildxxxx.azurewebsites.net/Indexer/Create Navigate to http://inbuildxxxx.azurewebsites.net/Indexer/Run 	<p>Let's create the Indexer by running this command. First we are creating it. Then we are setting it to run.</p> <p>Note: This is creating a predefined indexer which will link the existing DocumentDB database for products with the existing azure search service.</p>
<pre> public async Task<bool> CreateIndexer() { if (!await _searchClient.DataSources.ExistsAsync(_dataSourceName)) { var dataSource = BuildDataSource(); if (dataSource != null) { await _searchClient.DataSources.CreateAsync(dataSource); } } if (!await _searchClient.Indexers.ExistsAsync(_indexerName)) { var indexer = BuildIndexer(); await _searchClient.Indexers.CreateAsync(indexer); } return true; } </pre>		<p>While we wait for the Indexer to run, let's observe how we created the data source and indexer.</p> <p>BuildDataSource creates a DataSource object (as shown in step 3). This is a datasource that looks at the DocumentDB product database.</p> <p>BuildIndexer creates an Indexer object (as shown in step 3).</p> <p>After a couple of minutes, we should start getting results (shown in Scenario 2).</p> <p>Source Code: MSCorp.AdventureWorks.Core->AzureSearchClient.cs->Line 114</p>

Screen	Click Steps	Demo Script
Scenario 2: Azure Search – Search Suggestions		
	<ol style="list-style-type: none"> 1. Navigate to the search box. 	
	<ol style="list-style-type: none"> 2. Type 'Mountian' as a misspelling of 'Mountain' into the search box. (Don't press enter yet) 	<p>We use out of the box Azure Search to provide search Suggestions to the user. Suggestions is the term Azure Search uses to retrieve suggestions based on partial search input. They can only be set for fields of type Edm.String or Collection(Edm.String)</p> <p>One of the options for Search suggestions is fuzzy which takes a Boolean parameter.</p> <p>When fuzzy is turned on the Search service will find suggestions even if there's a substituted or missing character in the search text.</p> <p>Worth noting that while this provides a better experience, in some scenarios it comes at a performance cost as fuzzy suggestion searches are slower and consume more resources.</p>
	<ol style="list-style-type: none"> 3. Open a new browser window and navigate to http://jsonviewer.stack.hu/#http://inbuildxxxx.azurewebsites.net/Api/Search/Suggest/Mountian - Notice that we are sending the misspelling of mountain to the Suggest endpoint of the service. 4. Expand out the element 0 of the Results collection. 	<p>See that we are getting the correct spelling of mountain when passing in the incorrect spelling.</p>


Screen	Click Steps	Demo Script
<pre> [Route("Api/Search/Suggest/{searchText}")] [HttpGet] public async Task<HttpResponseMessage> Suggest(string searchText) { var result = await _searchClient.ExecuteSuggest(searchText, new SuggestParameters() { UseFuzzyMatching = true, Top = 6, Select = new[] { "ProductCode" } }); return WebApiHelper.ReturnRawJson(JsonConvert.SerializeObject(result)); } public async Task<dynamic> ExecuteSuggest(string query, SuggestParameters suggestParameters) { using (var indexClient = _searchClient.Indexes.GetClient(_indexName)) { var suggestResponse = await indexClient.Documents.SuggestAsync(query, SearchSchemaGenerator.Sug- gesterName, suggestParameters); return new { suggestResponse.Results }; } } </pre>		<p><i>Let's see how this suggest was executed.</i></p> <p><i>Notice UseFuzzyMatching = true. This is what allows misspellings to still be found.</i></p> <p><i>Source Code: MSCorp.AdventureWorks.Web->ApiSearchControllert.cs->Line 48</i></p> <p><i>Source Code: MSCorp.AdventureWorks.Core->AzureSearchClient.cs->Line 300</i></p>

Screen	Click Steps	Demo Script
Scenario 3: Azure Search – Facets		
	<ol style="list-style-type: none"> 1. Navigate to the search box. 2. Type 'Mountain' into the search box and press Enter. 	<p>Azure search service provides us both the Search results and the refiner lists you see on the left hand side of the screen. Notice that we get other products which contain the word 'Mountain', these can be filtered using Azure Search Facets.</p>
	<ol style="list-style-type: none"> 3. Select these Facets (Filters) <ul style="list-style-type: none"> • Black • 40" 	<p>As we select and un-select facets, we are going back to Azure Search service for the facet filtering including the text search passed (in this case Mountain). We do this by filtering against the facets using the word values and piped filters E.g. facet=color,values:Black Silver White</p>
	<ol style="list-style-type: none"> 4. Open a new browser window and navigate to http://jsonviewer.stack.hu/#http://inbuildxxx.azurewebsites.net/Api/Search/Products/Mountain 	<p>Here you can see the Facets collection which provides us with our "groups" of filters that we can use to refine the results. (We provided the facet collections we wanted in Scenario 2.)</p>
<pre>[Route("Api/Search/ProductsByCategory/{categoryName}/{count}")] [HttpGet] public async Task<HttpResponseMessage> GetProductsByCategory(string categoryName, int count) { var result = await _searchClient.ExecuteSearch("", new SearchParameters() { Top = count, Facets = new[] { "ProductCategory" }, Filter = "ProductCategory eq '{0}'".FormatWith(categoryName) }); return WebApiHelper.ReturnRawJson(JsonConvert.SerializeObject(result)); }</pre>		<p>Let's see how we can use Facets with Azure Search.</p> <p>See that we have provided a Filter with this search. In this case we are asking to match where the product category is equal to the one passed into the function.</p> <p>Source Code: MSCorp.AdventureWorks.Web->ApiSearchControllert.cs->Line 176</p>

Screen	Click Steps	Demo Script
Scenario 4: Azure Search – Scoring Profiles		
	<ol style="list-style-type: none"> 1. Navigate to http://inbuildxxx.azurewebsites.net/Search/Mountain (or type mountain in the search box and click enter) 2. Click a product image which is not at the top of a list. (but has mountain written in the name) 3. Click Edit Product Details. 	<p><i>This will open up a new browser window from here to review the product definition. When looking at product detail page you are viewing the full product representation which is stored in DocumentDB</i></p>
	<ol style="list-style-type: none"> 4. Update the product priority to be significantly larger than it is right now and click save. Note: the priority scoring profile has been setup for values between 1 and 10,000. 	<p><i>As the product is updated we persist the changes into product storage and then update the representation within Azure Search.</i></p>
	<ol style="list-style-type: none"> 5. Navigate back to the original browser window with the search results and refresh the page. 	<p><i>By updating the products priority we have altered the order in which the products are returned. Note: This may take 5 minutes. Or we can force it straight away by running http://inbuildxxx.azurewebsites.net/Indexer/Run</i></p>

Scenario 5: DocumentDB

Mountain-500, Black



Suitable for any type of riding, on or off-road. Fits any budget. Smooth-riding with a comfortable ride.

Reviews

Adventure (1) ★★★★★

This is the best mountain bike that fictitious money can buy.

I agree I have one of these and they are brilliant :)

Comment by Adam Bart's review

\$799 (USD)

Size: M18B

Black

48"

1

+

+

Add to cart

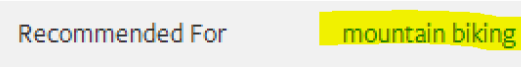
Specifications

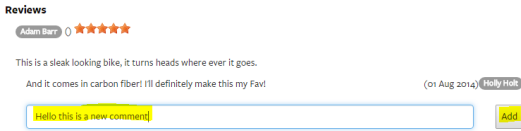
Cable Routing	Internal
Head Tube Diameter	Standard
Headset Included	Yes
Manufacturer Warranty	2 years
Material	Carbon Fiber
Rear Shock	HL Shock
Rear Travel	5 in
Recommended For	Mountain Biking
Reversible Rear Derailleur Hanger	Yes
Saddle Diameter	33.0 mm
Suspension	Full Suspension

1. Navigate to <http://inbuildxxxx.azurewebsites.net/Product/BK-M18B>

In AdventureWorks we have two product stores, DocumentDB which is the master product store and the product index from within Azure Search. The product information which you see on this product details page is surfaced directly from DocumentDB and passed to the browser for display.



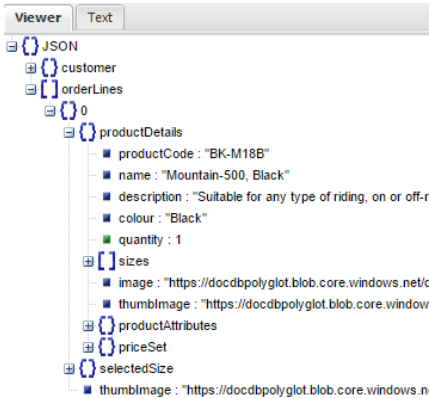
Screen	Click Steps	Demo Script																								
Scenario 6: DocumentDB – Schema Free Query																										
<div>Specifications</div> <table><tr><td>Cable Routing</td><td>internal</td></tr><tr><td>Head Tube Diameter</td><td>standard</td></tr><tr><td>Headset Included</td><td>yes</td></tr><tr><td>Manufacturer Warranty</td><td>2 years</td></tr><tr><td>Material</td><td>Carbon fiber</td></tr><tr><td>Rear Shock</td><td>HL Shocks</td></tr><tr><td>Rear Travel</td><td>5 in</td></tr><tr><td>Recommended For</td><td>mountain biking</td></tr><tr><td>Replaceable Rear Derailleur Hanger</td><td>yes</td></tr><tr><td>Seatpost Diameter</td><td>32.0 mm</td></tr><tr><td>Suspension</td><td>full-suspension</td></tr><tr><td>Weight</td><td>3,320g</td></tr></table>	Cable Routing	internal	Head Tube Diameter	standard	Headset Included	yes	Manufacturer Warranty	2 years	Material	Carbon fiber	Rear Shock	HL Shocks	Rear Travel	5 in	Recommended For	mountain biking	Replaceable Rear Derailleur Hanger	yes	Seatpost Diameter	32.0 mm	Suspension	full-suspension	Weight	3,320g	<div><div>1.</div><div>Navigate to http://inbuildxxxx.azurewebsites.net/Product/BK-M18B</div></div> <div><div>2.</div><div>View the specifications of the product.</div></div>	<div>Each product has some level of specifications which will be unique to each product. E.g. Bike "Frame Material : Carbon Fiber" VS Water bottle "Material : Eastman Tritan copolyester"</div>
Cable Routing	internal																									
Head Tube Diameter	standard																									
Headset Included	yes																									
Manufacturer Warranty	2 years																									
Material	Carbon fiber																									
Rear Shock	HL Shocks																									
Rear Travel	5 in																									
Recommended For	mountain biking																									
Replaceable Rear Derailleur Hanger	yes																									
Seatpost Diameter	32.0 mm																									
Suspension	full-suspension																									
Weight	3,320g																									
<div><div><div><div></div></div><div>ProductAttributes</div><div><div>■ Material : "Carbon fiber"</div><div>■ Suspension : "full-suspension"</div><div>■ Rear Travel : "5 in"</div><div>■ Rear Shock : "HL Shocks"</div><div>■ Head Tube Diameter : "standard"</div><div>■ Headset Included : "yes"</div><div>■ Seatpost Diameter : "32.0 mm"</div><div>■ Cable Routing : "internal"</div><div>■ Replaceable Rear Derailleur Hanger : "yes"</div><div>■ Recommended For : "mountain biking"</div><div>■ Manufacturer Warranty : "2 years"</div><div>■ Weight : "3,320g"</div></div></div></div>	<div><div>3.</div><div>Open a new browser window and navigate to http://jsonviewer.stack.hu/#http://inbuildxxxx.azurewebsites.net/Api/Search/Product/BK-M18B</div></div> <div><div>4.</div><div>Expand out the Product Attributes Collection.</div></div>	<div>These specifications show the schema-free nature of the DocumentDB, this is not particularly special but next we will talk about query over schema-free data which really separates DocumentDB out from the competition.</div>																								

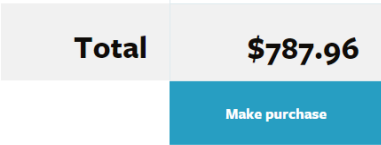
Screen	Click Steps	Demo Script
	<ol style="list-style-type: none"> Go back to the original page - http://inbuildxxxx.azurewebsites.net/Product/BK-M18B Click on the text 'mountain biking' under the specification 'recommended for' 	<p><i>In other areas of the application, we've used Azure Search service for predictable searching across known product attributes E.g. name, description, category, and manufacturer. DocumentDB is well suited to query across schema free attributes. DocumentDB provides a Rich query over schema-free design which allows us to query over these product specifications. Specifically:</i></p> <ul style="list-style-type: none"> <i>Schema-free design allows for agile development and continuous iteration</i> <i>Differentiated querying through no limitation to the number of ways you can query your data because there is no forced, pre-defined set of indices</i> <i>Query hierarchical JSON data through a succinct query grammar, including transforms and inline JavaScript</i>

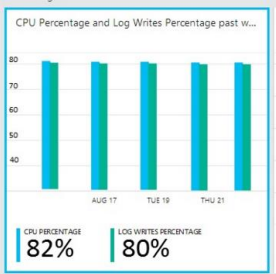
Screen	Click Steps	Demo Script
Scenario 7: DocumentDB – Nested Documents / Consistency		
	<ol style="list-style-type: none"> 1. Navigate to http://inbuildxxxx.azurewebsites.net/Product/BK-R50B 2. Add a comment underneath the existing review and click add. 	<p><i>The product review document holds the original review and any related comments are stored as sub documents. This makes up the hierarchy.</i></p>
<pre> {id": "d73749af-6308-4cd9-90f5-27b8589a4984", "_rid": "GfFsAPfwsAADAAAAAAAAA==", "_self": "dbs/GfFsAA==/colls/GfFsAPfwsAA=/docs/GfFsAPfwsAADAA", "_ts": 1452548624, "_etag": "\"000006e00-0000-0000-0000-569422100000\"", "IsForIndex": false, "Date": "2014-07-31T21:58:51.229Z", "ProductCode": "BK-R50B", "ReviewText": "This is a sleek looking bike, it turns heads w "Rating": 5, "ReviewingCustomer": { "Name": "Adam Barr", "CustomerKey": 100 }, "Responses": [{ "ResponseText": "And it comes in carbon fiber! I'll defi "Date": "2014-07-31T22:07:44.744Z", "CustomerResponding": { "Name": "Holly Holt", "CustomerKey": 103 } }, { "ResponseText": "Hello this is a new comment", "Date": "2016-01-11T21:43:44.4121639Z", "CustomerResponding": { "Name": "Adam Barr", "CustomerKey": 100 } }], "_attachments": "attachments/" , </pre>	<ol style="list-style-type: none"> 3. Navigate to http://jsonviewer.stack.hu/#http://inbuildxxxx.azurewebsites.net/Api/Search/ProductReview/BK-R50B 	<p><i>Looking at the JSON you see that the responses collection has a set of comments displaying the nested document structure.</i></p>

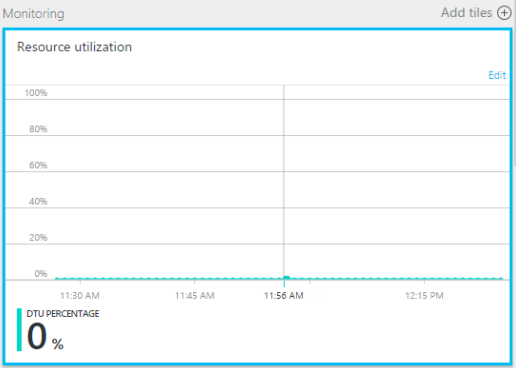
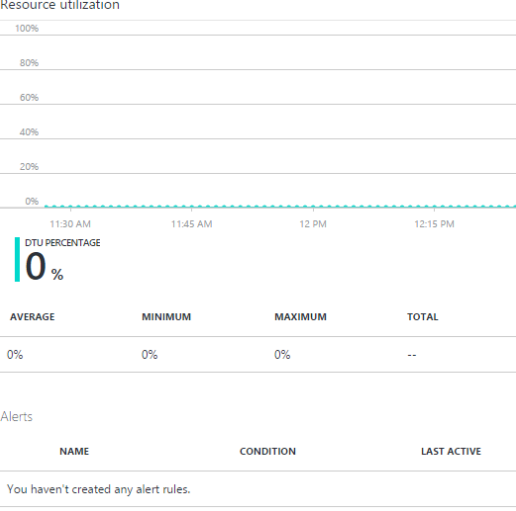
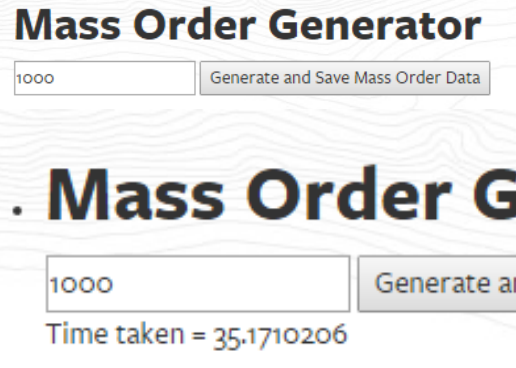
Screen	Click Steps	Demo Script
Scenario 8: DocumentDB – Transactions		
<p>Select products</p> <div> <div>Recommended For ▼</div> <div>mountain biking ▼</div> <div>Apply Filter</div> </div> <p>Set discount</p> <div> <div>15</div> <div>Apply Discount</div> </div> <p>Discount has been applied successfully.</p> <p><input type="checkbox"/> Prevent this page from creating additional dialogs.</p> <div>OK</div>	<ol style="list-style-type: none"> 1. Navigate to http://inbuildxxxx.azurewebsites.net/Admin/Product/Discount 2. Select Recommended For in the first drop down. 3. Select Mountain Biking in the second drop down. 	<p>Now that we have found all the products which we want to update let's give all the products a 15% discount.</p>
<div>Edit Product Details</div>	<ol style="list-style-type: none"> 4. Type 15 into the set discount. 5. Click Apply Discount (Wait for confirmation). 	<p>Under the covers we are updating all the product documents with that discounted price within a single ACID transaction. DocumentDB provides a full-featured NoSQL document database service including out of the box transactional processing over multiple documents through application defined JavaScript / stored procedures.</p>
<div> <div>Unable to apply discount. One or more products may have been modified, please refresh this view to try again.</div> <div><input type="checkbox"/> Prevent this page from creating additional dialogs.</div> <div>OK</div> </div>	<ol style="list-style-type: none"> 6. Right click any product in the list and Open in a new tab. 7. Click the Edit Product Details. 8. Save the product and go back to original discount tab. (Do not refresh). 9. Click Apply Discount (wait for confirmation). 	<p>See that the update discount has failed. Because we edited one of the products in the set, it was unable to be updated as it was out of date.</p> <p>We have leveraged the ETag on the product, using a concurrency check we have demonstrated one scenario where we wanted to abort the underlying transaction. We achieve this behavior through further use of stored procedures.</p> <p>Stored procedures, triggers and UDFs are modelled after the concepts supported by RDBMS. All JavaScript logic is executed within an ambient ACID transaction with snapshot isolation.</p>

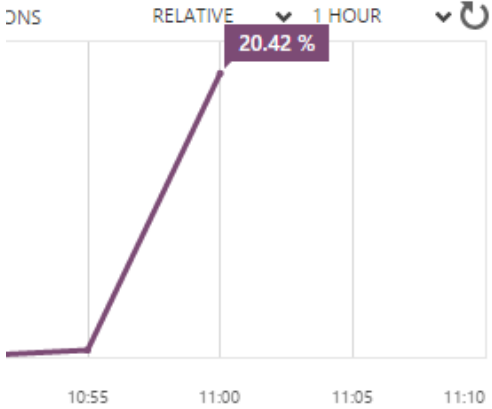
Screen	Click Steps	Demo Script
		<i>During the course of its execution, if the JavaScript throws an exception, then the entire transaction is aborted.</i>
	<p>10. Navigate to http://inbuildxxxx.azurewebsites.net/Api/Discover/StoredProcedures/ProductCollection</p> <p>11. Scroll to procedure ApplyDiscountToProducts.</p>	<i>This is the stored procedure responsible for apply discounts and checking the Etag.</i>



Screen	Click Steps	Demo Script
Table Storage Scenario 9: Table Storage		
	<ol style="list-style-type: none"> 1. Make sure you are signed in as Jeff Wang. 2. Navigate to http://inbuildxxxx.azurewebsites.net/Product/BK-M18B 3. Click Add to Cart. 	
	<ol style="list-style-type: none"> 4. Click on the cart icon. 	
	<ol style="list-style-type: none"> 5. Navigate to http://jsonviewer.stack.hu/#http://inbuildxxxx.azurewebsites.net/Api/Cart/108 	<p>As you can see by the representation of what is in the shopping cart, we are storing an un-schematized JSON representation of the order. If the order schema changes in any we don't need to consider changing anything about table storage. All information pertaining to the information is stored in a single document, so no links or hierarchy are required.</p> <p>We store the shopping cart based on the customer who has added the order to the cart. All the information is stored in a single document so no links or hierarchy is required.</p> <p>Table storage is cheap and accessible as a simple RESTful service. The shopping cart is a canonical use of key value pair - simple to retrieve based on the customer key and not required after the users session is completed.</p>





Screen	Click Steps	Demo Script
Azure SQL Scenario 10: Azure SQL – Transaction Data Store		
	<ol style="list-style-type: none"> 1. Navigate to http://inbuildxxxx.azurewebsites.net/Checkout 2. Click Make Purchase 	<p>As part of making the purchase we take what is in the shopping cart and push it into an Azure SQL database as the transactional store.</p>
	<ol style="list-style-type: none"> 3. Open up visual studio and add a data the connection to the database (Azure SQL Database Instance), then open up a T-SQL query window 	<p>Let's head over to the database and validate that we see a new order in the database. Run the following SQL looking at the order and lines.</p>
	<ol style="list-style-type: none"> 4. Execute these queries <pre>SELECT * FROM [dbo].[Orders] ORDER BY Id DESC SELECT * FROM [dbo].[OrderLines] ORDER BY Order_ID DESC</pre> 	<p>By using Azure SQL, we remove virtually all infrastructure maintenance with SQL Database which provides automatic software patching as part of the service. We are also able to leverage SQL Server skills across on-premises and cloud environments with a familiar relational foundation and T-SQL functions.</p> <p>These two aspects of Azure SQL make it both familiar and self-managed which have enabled AdventureWorks as a company to drive fast time-to-market and unprecedented efficiencies with near-zero maintenance service.</p>

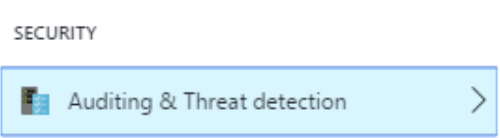
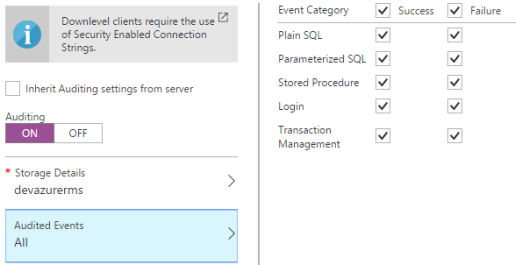
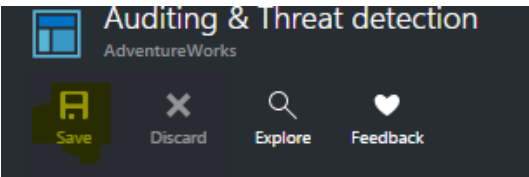
Screen	Click Steps	Demo Script												
Scenario 11: Azure SQL – Predictable Performance and Vertical Scaling														
	<ol style="list-style-type: none"> 1. Login to the Azure portal at https://portal.azure.com/ with credentials which has access to the database hosting the application. 2. Click Browse and select SQL databases. 3. Select the AdventureWorks, Click on Monitoring in the Database blade. 	<p>As you can see from the monitoring on the AdventureWorks database, at the moment with the current load we are in no way stressing the database. The database is configured on the premium service tier. This is exactly where we want to be with the sale event starting this weekend. But it was a different story last year.</p>												
 <table border="1"> <caption>CPU Percentage and Log Writes Percentage past w...</caption> <thead> <tr> <th>Date</th> <th>CPU Percentage</th> <th>Log Writes Percentage</th> </tr> </thead> <tbody> <tr> <td>AUG 17</td> <td>82%</td> <td>80%</td> </tr> <tr> <td>TUE 19</td> <td>82%</td> <td>80%</td> </tr> <tr> <td>THU 21</td> <td>82%</td> <td>80%</td> </tr> </tbody> </table>	Date	CPU Percentage	Log Writes Percentage	AUG 17	82%	80%	TUE 19	82%	80%	THU 21	82%	80%	<ol style="list-style-type: none"> 4. Navigate to http://inbuildxxxx.azurewebsites.net/images/LastYearMonitoring.jpg 	<p>Here are the metrics from last year's Black Friday sale event. We were caught out with the amount of traffic generated from so many customers placing orders.</p> <p>We can see that we were reaching to the 80% for Log writes per second and the CPU percentage. Based on these telemetry results we can clearly see our database was getting close to the upper threshold limit of the service tier, we had to quickly react and upgrade to a higher performance level to maintain great performance and thus a great customer experience.</p> <p>This was a good problem to have, but this year we are prepared in advance as we have already scaled our database service tier up to premium.</p>
Date	CPU Percentage	Log Writes Percentage												
AUG 17	82%	80%												
TUE 19	82%	80%												
THU 21	82%	80%												

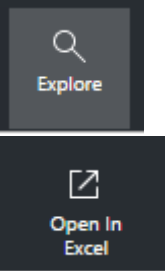
Screen	Click Steps	Demo Script
	<p>5. Close the tab and navigate back to the open portal window, Right click on the graph and select edit chart.</p>	<p><i>The Azure Portal has built in dashboard views showing telemetry data. This allows us to get an understanding of what's been happening on our databases over time. Note as developers we also have the ability to programmatically obtain this.</i></p> <p><i>We can easily include or exclude metrics to plot on the graph and change the charts time range. Options for telemetry are Storage, CPU percentage, Physical Data reads, Log writes per second, blocked by firewall, deadlocks, failed connections and successful connections.</i></p>
	<p>6. Click on Monitoring Graph.</p>	<p><i>Notice when I click on the monitoring graph we have the ability to add an alert. We can configure alerts on performance metrics. For example, if you expect the workload on your database to grow, you can chose to configure an email alert whenever your database reaches 80% on any of the log writes percentage. You can use this as an early warning to figure out when you might have to switch to the next higher service tier. Alternatively you could use it to monitor when it falls below a certain percentage to downgrade the service tier.</i></p> <p><i>Let's do an experiment.</i></p>
	<p>7. Navigate to http://inbuildxxxx.azurewebsites.net/MassOrdering</p> <p>8. Click Generate and Save Mass Order Data (and wait).</p>	<p><i>This will generate 1,000 orders simulating a big sale event with big usage spikes.</i></p> <p><i>You will see the time it took for this process.</i></p>

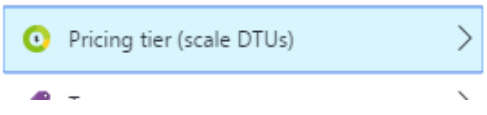
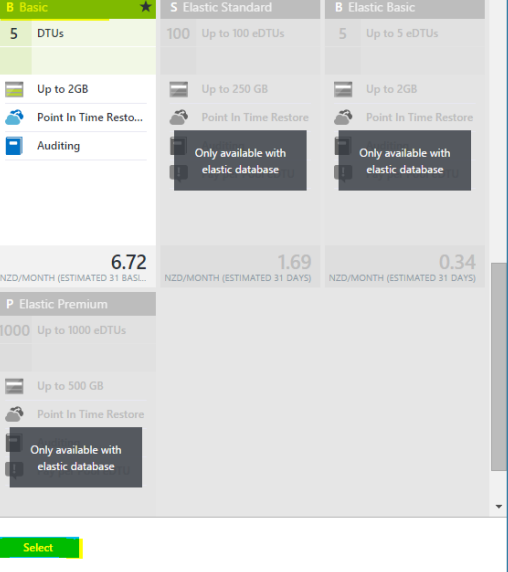

Screen	Click Steps	Demo Script
 <p>The graph displays a purple line representing DTU percentage. The line starts at a low value at 10:55, remains flat until 11:00, and then rises sharply to 20.42% at 11:05. The x-axis is labeled with times 10:55, 11:00, 11:05, and 11:10. The y-axis is labeled 'RELATIVE' and '1 HOUR'. There are dropdown arrows and a refresh icon at the top right of the graph area.</p>	<p>9. Go back to the monitoring graph.</p>	<p><i>Note: This may take 5-10 minutes to update.</i></p> <p><i>Notice that DTU percentage is at 20% (your results may differ slightly).</i></p> <p><i>We will compare the results when we scale down in Scenario 19.</i></p>

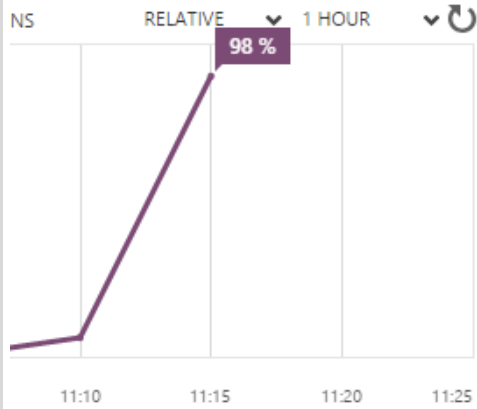
Screen	Click Steps	Demo Script
Scenario 12: Azure SQL – Database Recovery		
<div> <div> <div>★ Database name</div> <div>AdventureWorks_2016-01-11T23-25Z ✓</div> </div> <div> <div>★ Target server</div> <div>moffmo (Australia Southeast) </div> </div> <div> <div>Current time</div> <div>2016-01-11 23:34 UTC</div> </div> <div> <div>Oldest restore point</div> <div>2016-01-07 20:26 UTC</div> </div> <div> <div>Restore point</div> <div>Date</div> <div>2016-01-11 </div> <div>Hour</div> <div>23 ▾</div> <div>Minute</div> <div>: 25 ▾</div> <div>UTC</div> </div> <div> <div>Create</div> </div> </div>	<ol style="list-style-type: none"> 1. Sign in to the Azure Management Portal using your Microsoft account. 2. From the Databases list, select the database you want to restore - <i>Only the new service tiers Basic, Standard and Premium service tiers support Point in Time.</i> 3. At the bottom of the page in the command bar, click Restore. This launches the Specify Restore Settings dialog box. 4. Select the Restore Point. 5. Specify a new database name in the Database Name field. 6. Click create to submit the restore request. 	<p><i>Point in Time Restore is designed to recover a database to a specific point in time within the backup retention period supported by the service tier of the database. Restoring creates a new database with the same service tier that was in use at the chosen restore point and the lowest performance level supported by that tier. For example, if you restore a database which was set to Premium at the specified restore point in time, the new database will also be a Premium database.</i></p> <p><i>The Restore Point is used to specify the point in time to which your database should be restored. The restore point must be within the retention period supported by the database service tier and the Point in Time setting must be set to ON.</i></p> <p><i>The backup retention period is a maximum of seven days for Basic, 14 days for Standard, and 35 days for Premium. Requests to restore to a point in time outside of the supported retention period for the service tier will fail.</i></p> <p><i>Restore operations may take a long time to complete. You can monitor the status of the restore operation on the Databases list. Databases that are being restored are visible in the Databases list once the restore is in progress. The row is populated with the database settings and the status of the restore operation which is displayed as Restoring until the operation is completed.</i></p>

Screen	Click Steps	Demo Script								
<div><div>Operations</div><div><p>Deleted databases</p></div></div> <div>Restorable deleted databases</div> <table><thead><tr><th>DATABASE</th><th>DELETION TIME (UTC)</th><th>CREATION TIME (UTC)</th><th>EDITION</th></tr></thead><tbody><tr><td> polygot</td><td>2016-01-07 04:20</td><td>2015-12-21 03:13</td><td>Standard</td></tr></tbody></table>	DATABASE	DELETION TIME (UTC)	CREATION TIME (UTC)	EDITION	 polygot	2016-01-07 04:20	2015-12-21 03:13	Standard	<div><div>7. Navigate to the SQL Server.</div><div>8. On the side tab, scroll down to deleted databases.</div></div>	<p>We can also restore a deleted database. This list includes a list of Restorable Deleted Databases available to restore. Similarly to above, the restore point must be within an appropriate retention period supported by the service tier.</p> <p>Point in Time Restore enables restoring a database to a point in time in the past within the retention period for the given service tier. However, the retention period for the service tier may not be long enough to meet business needs. In this case, consider creating an archived database using methods such as automated export through Azure.</p> <p>Point in Time restore offers self-service control over data restoration from available backup data, putting the power to restore in the hands of customers in the case of a human or programmatic data deletion scenario.</p> <p>Using Azure there are other options for database recovery depending on the tier you are using. We've already mentioned Automated Export above, but there is also Geo restore, standard Geo replication and a database copy and manual restore.</p>
DATABASE	DELETION TIME (UTC)	CREATION TIME (UTC)	EDITION							
 polygot	2016-01-07 04:20	2015-12-21 03:13	Standard							

Screen	Click Steps	Demo Script
Scenario 13: Azure SQL – Auditing		
	<ol style="list-style-type: none"> 1. Sign in to the Azure Management Portal using your Microsoft account. 2. Select a database from the Databases list. 3. Click Auditing and Threat Detection. 	<p><i>This will enable the auditing and launch the auditing configuration blade.</i></p>
	<ol style="list-style-type: none"> 4. In the auditing configuration blade, select the Azure storage account where logs will be saved. 	<p><i>If we had several audited databases, we would use the same storage account for all audited databases to get the most out of the preconfigured reports templates.</i></p> <p><i>Audit logs are aggregated in a single Azure Store Table named AuditLogs in the Azure storage account.</i></p>
	<ol style="list-style-type: none"> 5. Under Auditing Event, click All, then click OK. 	<p><i>Depending on the requirements for auditing, we can configure what type of events are audited. Access to data, Schema changes (DDL), data changes (DML), Accounts, roles, and permissions (DCL), Security exceptions.</i></p> <p><i>We can also save this configuration as default. This means that these settings would apply to all future databases on the same server, and any that don't already have auditing set up. You can override the settings later for each database by following these same steps.</i></p>
 <p>Saving Saving Auditing settings...</p>	<ol style="list-style-type: none"> 6. Click Save. 	

Screen	Click Steps	Demo Script
	<ol style="list-style-type: none"> 7. Click Explore. 8. Click Open in Excel. 	<p>To use the template on your audit logs, you need Excel 2013 or later, and Power Query. The template here has fictional sample data in it which we can use to show you the powerful capabilities.</p> <p>Let's have a look through some of the sheets.</p> <p>Orientation: Documentation links and brief description of the visible interactive reports</p> <p>Anomalies: Events that usually should not appear in normal operation. Double-click a Count value for the list of events</p> <p>Drill Down: Ability to slice to the desired specific set of rows. Double-click a Count value for the list of events</p> <p>Event Type Distribution: Distribution of Audit Logs Event Types by several dimensions: Database, Month --> Day, Weekday, Hour in the Day</p> <p>Event Time Analysis: Analysis over time by Event Type. The two vertically adjacent charts enable correlating events with different scale of appearance. Double-click a count value for the list of events</p> <p>With the Database Auditing capability, Microsoft Azure SQL Database offers a tool to streamline compliance-related activities, gain knowledge about what is happening in your database, and identify trends, discrepancies, and anomalies. These can serve to enhance business visibility, or potentially indicate business concerns or suspected security violations.</p>

Screen	Click Steps	Demo Script
Scenario 14: Azure SQL - Vertical Scaling		
	<ol style="list-style-type: none"> 1. Login into the Azure Portal. 2. Select SQL Databases and select the AdventureWorks database. 3. Click on the pricing tier tab. 	
	<ol style="list-style-type: none"> 4. Click on Basic then click Select. 	<p><i>This will downgrade our database to five DTUs.</i></p>
	<ol style="list-style-type: none"> 5. Navigate to http://inbuildxxxx.azurewebsites.net/MassOrdering 6. Click Generate and Save Mass Order Data and wait. 	<p><i>This will generate 1,000 orders, simulating a big sale event with big usage spikes.</i></p> <p><i>You will see the time it took for this process.</i></p>

Screen	Click Steps	Demo Script
 <p>NS RELATIVE 1 HOUR</p> <p>98 %</p> <p>11:10 11:15 11:20 11:25</p>	<p>7. Go back to the monitoring graph (Database Page).</p>	<p><i>Note: This may take 5-10 minutes to update.</i></p> <p><i>Notice that DTU percentage is at 98%. Your results may differ slightly.</i></p> <p><i>This has drastically reduced the performance of the SQL Server. Luckily with Azure, it is easy to vertically scale the services on an adhoc or full time basis.</i></p>

