

Azure SQL Database

Demonstration

Presenter Script  
September 2015

Final Draft—09/16/2015

Prepared for: Darmadi Komo, Tara Shankar Jana Microsoft

Prepared by: Prowess Consulting

Overview

This demonstration explores the benefits for several Microsoft Azure SQL Database features in the context of a software-as-a-service (SaaS) provider, Wingtip Tickets, which provides ticketing software to artists and groups. This demonstration centers on the tenant Julie and the Plantes (a fictitious pop-music tenant).

Other tenants that will be discussed in future labs include the following:

* The Archie Boyle Band (a fictitious rock-music tenant)
* Walla Walla Symphony (a fictitious classical-music tenant)

## Demo Architecture

## 

**Figure 1** Overall architecture of demo components

|  |  |
| --- | --- |
| **Demo Story Elements** | **Min** |
| **Scenario 0: Purchasing a Ticket** Show the Wingtip Tickets website and discuss the core concept and existing database challenges with the audience to set the scene. | 3 |
| **Scenario 1: Vertical Scaling**  Discuss how the website is anticipating a big-name artist and how that might require some increased capacity. Simulate some load using the load 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 S3 to P1) and how quickly the change is implemented. Re-run the load generator and show how the increased capacity has significantly improved the site’s ability to handle a spike in ticket sales. | 3 |
| **Scenario 2: Azure SQL Database Elastic Database Pools**  Discuss how scaling up individual databases might not be the most efficient means of dealing with spikes in usage, particularly where the exact demands of individual databases are not known and hundreds or even thousands of databases might be in play (such as for software-as-a-service [SaaS] providers). From the portal, show how easy it is to set up elastic database pools. Re-run the load generator to produce a spike in demand and show how easily the pool meets the increased demand. | 3 |
| **Scenario 3: Auditing**  Discuss how auditing can be used to keep track of infrastructure changes made to the environment in addition to database-related issues (such as failed logon attempts). In the portal, show the operation to enable and set up auditing for a specific database. Discuss how we’re going to simulate a scenario to highlight the benefits of auditing by deleting one of the concert events, which will effectively delete all events associated with that concert. Then discuss that by using the auditing dashboard in a Microsoft Excel template, we will show how we can drill down into the auditing events by showing who deleted the event and at what time. | 3 |
| **Scenario 4: Point-in-Time Restore**  Discuss how point-in-time restore can be used to restore the missing concert event. In the portal, show how easy it is to restore a database to a point in time. From the website, show how the deleted concert event is restored, along with previously purchased tickets. | 3 |
| **Scenario 5: Geographic Disaster Recovery**  Discuss how point-in-time restore is great for “oops” scenarios, but what about broader server/datacenter failures? Discuss Azure SQL Database active geographic replication. In the portal, show how easy it is to set up a geo-replicated secondary, and view the status via an interactive map. Next discuss how Traffic Manager can be used to detect a failed website and redirect traffic to a hot standby in another datacenter. | 3 |
| **Scenario 6: Azure Search**  Discuss how Azure Search enables users to search for matching terms across multiple databases or database tables. Discuss how you will walk through the search experience from the user’s point of view. | 3 |
| **Conclusion/Summary** | 2 |
| **TOTAL** | 23 |

| Screen | Click Steps |
| --- | --- |
|  | **DEMO SETUP** |
|  | The demo uses the following server names; your server names will necessarily be different: Replace *XX00* with your initials and the deployment number used in the setup script.   * *XX00*julieandtheplantesprimary.database.windows.net (Primary Database Server) * *XX00*julieandtheplantessecondary.database.windows.net (Secondary Database Server) * *XX00*julieandtheplantesprimary.azurewebsites.net (Primary Web Role) * *XX00*julieandtheplantessecondary.azurewebsites.net (Secondary Web Role) * http://*XX00*julieandtheplantes.trafficmanager.net (Traffic Manager)   Additionally, the *XX00*julieandtheplantes Microsoft Azure Storage Table is used to store audit events. |
|  | 1. Connect Microsoft SQL Server Management Studio (SSMS) to the two databases:  * *XX00*julieandtheplantesprimary.database.windows.net * *XX00*julieandtheplantessecondary.database.windows.net   User: **developer**  Password: **P@ssword1** |
|  | 1. In the Azure portal, http://portal.azure.com, make sure that:  * The Customer1 database (DB) is set to **S0 Standard SKU** and Geo-replication is disabled. Update the pricing tier using these instructions:   + Click **Customer1** database.   + Click **Pricing tier**.   + Click **S0 Standard**.   + At the bottom of the page, click **Select**. * The Customers2 DB doesn’t exist on the *XX00*julieandtheplantesprimary.database.windows.net server. |
|  | 1. Ensure that your Excel template, 01-Azure SQL DB Audit Logs Report Template, is connected to your storage account. 2. Open **01-Azure SQL DB Audit Logs Report Template**. 3. Click **Enable Content**. 4. Click the **PowerQuery** tab. 5. Make sure **Show Pane** is enabled. 6. Under Workbook Queries, on the right, right-click **AuditLogs**. 7. Click **Edit**. 8. Click the gear next to **Source**. 9. Verify the correct Azure Table Storage account is referenced. 10. Click **OK**. 11. At the top left, click **Close & Load**. |
|  | 1. If you’ve previously enabled auditing, use the **AuditLogUtility** to purge the ***XX00*julieandtheplantes** storage table, which will improve load times during the Auditing section. Switch back to the Azure Portal. 2. Browse to the Customer1 SQL Database. 3. Click the **Customer1** database on the *XX00*julieandtheplantesprimary.database.windows.net server. 4. In the Summary pane, scroll down to the **Operations** tile. 5. Click **Auditing**. 6. Click **Storage Details**. 7. Ensure that the Storage Account is listed. 8. Click **OK** until the database settings are shown. |
|  | In order to perform later sections of the demonstration, such as Auditing and Point-in-Time Restore, you will need to first buy and delete a ticket to the Julie and the Plantes concert.   1. Browse to http://*XX00*julieandtheplantes.trafficmanager.net/. 2. Select the first event, and then click the big **Find Seats** icon. |
|  | 1. Click **Sign in** to purchase tickets. |
|  | 1. Either create a new account if you haven’t browsed to the site before, or sign in using an existing account. |
|  | 1. Now that you’re signed in, click the **Purchase tickets** button.   **Note:** Take note of the location for which the tickets are purchased. |
|  | [Payment information isn’t required in this demo.]   1. Click **Purchase Now**. |
|  | [You should receive a purchase confirmation.]   1. Click **OK**. |
|  | 1. Click the **My Events** tab to verify the purchase.   [Notice the event that you just purchased.]   1. Click **Sign Off**. 2. Click **Sign In**. 3. Sign in using the administrator credentials:  * **admin@admin.com** * **P@ssword1**  1. Click the **Admin** tab. 2. In the **City**, **Venue**, **Event**, and **Artist** drop-down menus, select the values that match those of the event for which you bought tickets. 3. Verify that this event is the same event for which you purchased tickets. 4. Click **Delete Event**. |

| Screen | Click Steps | Demo Script |
| --- | --- | --- |
|  | **1. VERTICAL SCALING** |  |
|  | One of the benefits of Azure SQL Database is the ability to scale up and down as needed, with zero impact to uptime. | |
|  | 1. Open Azure SQL Database L200 - Demo Script Companion PowerPoint.pptx. 2. Show the screenshot of the Customer1 database utilization at the Basic Pricing tier. 3. Launch the Load Generator tool. 4. Insert the Primary Database Server (***XX00*julieandtheplantesprimary.database.windows.net**). 5. Select an **Event**, **Ticket Level**, and **Customer**.   For the purposes of this demo, the same event used in the previous section was chosen.   1. Change the **Tickets To Purchase** value to **2000**. 2. Change the **Bulk Purchase** value to **80**. 3. Click **Start**. | To demonstrate a scenario where an event has just gone on sale, and many users are purchasing tickets at the same time, we will use a load generator to create a usage spike. |
|  | 1. Show a screenshot of the load on a higher-pricing-tier version of the primary database server. | Our load generator simulates a real-world scenario where 2,000 tickets are purchased. As a result, the database CPU usage spikes to almost 100 percent. This would obviously cause a problem if there were multiple events that went on sale at the same time. It would also trigger a Max CPU Utilization Alert for administrators.  Let’s contrast that with how a higher-pricing-tier version of the same primary database server fared. This time, the database CPU usage spiked to around 40 percent, which demonstrates that the added capacity is already providing Wingtip Tickets with the means of coping with greater demand. |

| Screen | Click Steps | Demo Script |
| --- | --- | --- |
|  | **2. AZURE SQL DATABASE ELASTIC DATABASE POOLS** |  |
|  | The previous scenario demonstrated how Azure SQL Database can quickly and easily be used to increase the capacity of an individual database. However, often the exact demands of a database are not known, and managing the individual capacity of a database isn’t efficient. This is especially so in the case of SaaS providers, such as the one in this demo, who might have hundreds or even thousands of different tenant databases, each with its own unique mix of performance demands that rarely coincide. Elastic database pools help solve this by enabling you to set a policy for a group of elastic databases. This way you don’t have to worry about the performance of each database individually or inefficiently overprovision individual databases to meet occasional peak demand. | |
|  | 1. Open Azure SQL Database L200 - Demo Script Companion PowerPoint.pptx. 2. Go through the slide deck showing how Azure SQL Database elastic database pools can increase the capacity of the database. 3. Discuss slide 3 with the audience.   **Note:** To see or show the actual process of creating an Azure SQL Database elastic database pool, refer to the Level-300 Elastic Database Pool demo guide. | What happens when we really hammer our servers and crank up demand to 20,000 ticket orders and a 500-ticket bulk order? |
|  | 1. Discuss slide 4 with the audience. | That’s a total increase in demand of over six times, so even our upgraded server from the last example won’t be able to cope with the several thousands of tickets purchased per second. We will look to see the load on our servers as part of an elastic database pool. |
|  | 1. Discuss slide 5 with the audience. | Let’s look at the Customer1 database. Notice that, in the Resource Utilization graph, the max utilization was about 20 percent, yet the number of tickets purchased per second increased over six times.  (**Note:** This value will likely vary depending on the values that you specified in the load generator and the network from which you ran the test.) |

| Screen | Click Steps | Demo Script |
| --- | --- | --- |
|  | **3. AUDITING** |  |
|  | In addition to being able to provide excellent scale-up and scale-out capabilities, Azure SQL Database also helps database administrators gain insights into infrastructure-related issues and determine root causes.  In this scenario, we’re going to delete an event to simulate an “oops” scenario—for example, where a developer or database administrator (DBA) accidentally delete some records without realizing what they are deleting. | |
|  | For this demo, use the provided Excel template that is already connected to the Azure table.   1. Open the 01-Azure SQL DB Audit Logs Report Template. 2. Make sure the Workbook Queries pane is displaying on the right.   If it is not, then under the **Power Query** menu, click **Show Pane**.   1. Right-click **AuditLogs**. 2. Click **Refresh**. | For this demo, one of the concert events was accidentally deleted from the database. In the real world, the DBAs would want to figure out what went wrong as quickly as possible when customer calls start coming in. This is where database auditing comes in handy.  In the pane with the Auditing Events pie chart, notice the Open in Excel button. An Excel Power Query spreadsheet is provided in Azure to connect to the audit log that is saved to Azure table storage. |
|  | **Note:** The query should show how many rows have loaded once the refresh is completed.   1. Click the **Drilldown** worksheet. 2. To the right of **DataChanges**,double-click the count (in this example, **1**). | Once the refresh of the spreadsheet is complete, we will drill down into data changes. |
|  | 1. Show the audience the delete event for event ID1 or the event ID that was deleted. | This loads a new worksheet, which is populated with all rows related to data changes. Notice the delete event for event ID 1. |
|  | 1. Show the audience the time that this took place. | And notice the time that the deletion took place: precisely when we deleted the event. Now we need to restore the event. |

| Screen | Click Steps | Demo Script |
| --- | --- | --- |
|  | **4. POINT-IN-TIME RESTORE** |  |
|  | The prior scenario walked through how Azure SQL Database helps DBAs gain insights into infrastructure-related issues as and when they happen via events, which can be used to drill down further to determine root cause.  Based on the analysis, it looks like an event was deleted. To resolve this issue, point-in-time restore will be used to restore the related databases: Customers1 and ConcertTickets. | |
|  | **Note:** This section of the demo is dependent on the setup section (section 0). If you did not do that section, you will need to delete Event ID 1 from the Admin menu on the website before continuing.   1. Switch back to the Azure Portal. 2. Browse to the Customer1 SQL Database. 3. Click the **Customer1** database on the *XX00*julieandtheplantesprimary.database.windows.net server. | In order to restore the event, we need to go to the Customers1 database. |
|  | 1. Click **Restore**. 2. Change the **Restore Point** time to two minutes before the delete event occurred today. 3. Click **Create**. | We are going to restore the database to two minutes before the deletion took place. (We don’t want to go back too far before the deletion in case we accidentally undo some legitimate changes to the database in the process.) |
|  | 1. Wait while the Customer1 database is restored. | The Customer1 database is now being restored. |
|  | 1. Point out to the audience the status of the Customer1 database changing to Online. | Once restored, the status of the Customer1 database changes to Online. |

| Screen | Click Steps | Demo Script |
| --- | --- | --- |
|  | **5. GEOGRAPHIC DISASTER RECOVERY** |  |
|  | In the previous scenario, we walked through how to resolve a mishap by restoring a database using point-in-time restore. But sometimes, greater catastrophes happen. Azure SQL Database provides capabilities to recover from geographic disasters.  In this next scenario, we’re going to simulate a datacenter failure and show how active geographic replication can be used to keep a secondary environment on hot standby. | |
|  | 1. Browse to the Customer1 (previously restored) SQL Database. 2. Click the Customer1 (previously restored) database on the *XX00*julieandtheplantesprimary.database.windows.net server. | In order to take advantage of active geographic replication… |
|  | 1. In the Customer1 (previously restored) database **Summary** pane, scroll down to the **Pricing** tier. 2. Click the **Pricing** tier. 3. Click **P1 Premium**. 4. Click **Select**. | …we will first need to upgrade the Customer1 database to a Premium SKU. |
|  | 1. Point out to the audience the Updating status in the Pricing tier tile. | Notice the status in the Pricing Tier tile: Updating. |
|  | 1. Browse to the **Customer1** (previously restored) SQL Database. 2. Click the **Customer1** (previously restored) database on the *XX00*julieandtheplantesprimary.database.windows.net server. 3. In the **Summary** pane, scroll down to the **Geo-Replication** tile. 4. Click **Configure Geo-Replication**. | The Customers1 database is already an appropriate SKU, so we will enable GeoDR on that one while ConcertTickets updates. |
|  | 1. Select the recommended target region: in this example, because we’re using the Central US region, the recommended target region is East US 2. | We will select the recommended Target Region. |
|  | 1. Use an existing server: ***XX00*julieandtheplantessecondary.database.windows.net**. 2. Click **Select**. 3. Ensure that the replica is set to **Readable**. | We will use an existing server,XX00julieandtheplantessecondary.database.windows.net. |
|  | 1. Click **Create**. |  |
|  | 1. Wait for the replication link to be established between the Primary and Secondary servers. | A dotted line should animate displaying that the replication link is being established between the Primary and Secondary servers. |
|  |  | Once the replication link is established and replication has completed, the link should become solid, and check marks should appear on both servers. |

| Screen | Click Steps | Demo Script |
| --- | --- | --- |
|  | **6. AZURE SEARCH** |  |
|  | Being able to search for matching terms across multiple databases or database tables is crucial for your customers. Here, we’ll look at the Azure Search user experience. | |
|  | 1. Browse to the tenant website. (In this example, **http://*XX00*julieandtheplantes.trafficmanager.net/**). 2. In the **Search** box at the top right, type **detroit**. | The first thing to notice is the suggested auto-complete for a single event that comes up while we type in “Detroit.” |
|  | 1. Click on the event, and then click the **Search** button. | As expected, the resulting Azure returns reflects our search. In this case, it is a single result. |
|  | 1. In the **Search** box on the website, type **seattle**. | However, as we type in “Seattle,” Azure Search suggests multiple events for auto-completion on that term. |
|  | 1. Press Enter. | And when we press enter, Azure returns all of the suggested events that match up with “Seattle.”  Azure Search is powered by code developed in the web applications to drive the search results from multiple tables across multiple databases. |

|  |  |  |
| --- | --- | --- |
|  | **7. CONCLUSION** |  |
|  | Azure SQL Database enables you to create—and upgrade—servers in minutes. It also enables you to dynamically manage tens, hundreds, or even thousands of servers with Azure SQL Database elastic database pools. You can also easily and quickly activate high-availability features like geographic disaster recovery with just a handful of mouse clicks. Azure SQL Database also provides you with powerful tools to audit your databases so that you can quickly and easily find problems and restore databases to earlier points in time to resolve issues. Azure SQL Database also includes powerful user-facing features like Azure Search to help web sites and other database applications meet user needs.  This demonstration showed all these features on a high level. To see them in more depth—and to see how easy it can be to implement them even in production environments—talk with your Microsoft sales representative to schedule a demo. | |

|  |  |  |
| --- | --- | --- |
|  | Click Steps | |
|  | **8. RESET STEPS** |  |
|  | Follow the steps below to reset the Azure SQL Database demo environment. | |
|  |  | |
|  | 1. Browse to <http://portal.azure.com>. 2. Navigate to **SQL Databases**. 3. Select the Database that was restored in the Point-in-Time Restore section of the demo and on the original Server location of the primary server in the Geo-Replication section. 4. Scroll down to **Geo-Replication**. 5. Select the **Secondary** location. 6. Click **Stop Now**. 7. Click **Yes**. 8. Select the **Secondary** copy. 9. Click **Delete**. 10. Select the **Customer1** database. 11. Click **Delete**. 12. Launch SQL Server Management Studio. 13. Connect to the Primary Database Server (*XX00*julieandtheplantesprimary.database.windows.net). 14. Rename the restored database to **Customer1**. 15. Go back to <http://portal.azure.com>. 16. Select the **Customer1** database. 17. Click **Pricing Tier**. 18. Click **S0 Standard**. 19. At the bottom of the page, click **Select**. | |