

Azure SQL Database – Geographic Disaster Recovery

Level-300 Demonstration

Script

Version 1.0

9/3/2015

Prepared for: Darmadi Komo

Prepared by: Prowess Consulting

|  |  |
| --- | --- |
| **Form Factor** | Demo script (Microsoft Word) |
| **Target Audience** | TDMs and database administrators |
| **Goals/Objectives** | Increase confidence among Microsoft SQL Server administrators that Azure SQL Database is easy to configure and can still meet the needs of their production databases. |

Overview

This demonstration explores geographic disaster recovery in Microsoft Azure SQL Database 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, will 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

## Dependencies

This demonstration requires running the deployment and configuration PowerShell scripts from the Level-200 demonstration in order to work.

|  |  |  |
| --- | --- | --- |
|  | **Section 1: Geographic Disaster Recovery** |  |
|  | In this demonstration, we will simulate a datacenter failure and see how active geographic replication can be used to keep a secondary environment on hot standby. In this demo you will:   * See how easy it is to set up a geo-replicated secondary from the Azure SQL Database portal and view its status by using an interactive map. * Explore how Traffic Manager can be used to detect a failed website and redirect traffic to a hot standby in another datacenter. * Watch a failover of the website, and show how Traffic Manager has detected the site has failed and is redirecting traffic to the hot standby to limit interruption. * See a simulated database server failover by stopping replication to the secondary databases, effectively promoting the secondary to a primary.   From the Tenant website, you will also see how everything continues to work once Traffic Manager has rerouted requests to the passive Tenant site (which is now active, and the database secondary has become the primary). | |
|  | 1. Browse to the **Customer1** database. 2. In the **Customer1** database summary pane, scroll down to the Pricing tier. 3. Click **Pricing tier**. | In order to take advantage of active geographic replication, we will first need to upgrade the Customer1 database to a Premium SKU. |
|  | 1. Click **P1 Premium**. 2. Click **Select**. | Notice the status in the Pricing Tier tile: Updating. |
|  | 1. Once completed, in the Summary pane of the Customer1 database, scroll down to the Geo Replication tile. 2. Click **Configure Geo-Replication**. |  |
|  | 1. Select the recommended target region: in this example, because we’re using the West Europe region, the recommended target region is **North Europe.** | We will select the recommended Target Region: because we’re using the West Europe region, the recommended target region is North Europe.. |
|  | 1. Use an existing server: Secondary Database Server (In this example *mbjulieandtheplantessecondary.database.windows.net* is used.) 2. Click **Select**. 3. Ensure the replica is set to **Readable**. | We will use an existing server. |
|  | 1. Click **Create**. | The animated dotted line should displays that the replication link is being established between the primary and secondary database servers. |
|  |  | Once the replication link is established and replication has completed, the link switches to solid and there are check marks on both servers. |
|  | 1. Switch back to the Tenant website. (In this example *mbjulieandtheplantes.trafficmanager.net*). 2. Refresh the *Tenant* website. 3. Scroll down to the bottom of the page. 4. Notice that PrimaryDbServer = *Primary Database Server (In this example: mbjulieandtheplantesprimary*.) 5. Notice that SiteHost = *Primary WebApp*, which is the destination endpoint to which Traffic Manager is routing all web traffic. (In this example: *mbjulieandtheplantesprimary.azurewebsites.net)* | We can see some of these changes propagated to the web set.  Right now, the primary database server and the site host show up as the primary server and destination for those (which is what we would expect). |
|  | 1. Switch back to the Azure portal. 2. At the lower right, click **Azure Portal** to launch the old portal. | Now let’s simulate an outage. |
|  | 1. Once the old portal (manage.windowsazure.com) loads, from the left column, click **Traffic Manager**. | We will switch to the old Azure portal in order to interact directly with the Traffic Manager. |
|  | 1. Select the Traffic Manager Profile that was created for you. (In this example *mbjulieandtheplantes)*. 2. Click **Endpoints**. 3. Notice there are two sites configured:  * *Primary WebApp* * *Secondary WebApp* | The Traffic Manager endpoints are configured in failover mode, because each site is mapped 1:1 to their respective database server. For example, *Primary WebApp* is mapped to *Primary Database Server* and *Secondary WebApp* is mapped to the *Secondary Database Server*. |
|  | 1. Select the Primary WebApp Endpoint (in this example mbjulieandtheplantesprimary.azurewebsites.net). 2. Click on **Disable** on the bottom right, and click **Yes** to confirm the Primary Endpoint is being disabled. | The primary reason for this configuration is specifically to address a datacenter disaster.  Let’ see this in action. To simulate a regional outage, we will manually stop the *Primary WebApp* Endpoint. |
|  | 1. Select the Secondary WebApp Endpoint (in this example mbjulieandtheplantessecondary.azurewebsites.net). 2. Click on **Enable** on the bottom right, and click **Yes** to confirm the Secondary Endpoint is being enabled. | We will now enable the secondary endpoint to move traffic there. |
|  | **Note:** It should take around 30 seconds for Traffic Manager to check the endpoint and set its status to Online. You may need to refresh your browser if the status doesn’t update. | After a few second, the Traffic Manager is back online. But what does this look like from the customer point of view? |
|  | 1. Return to the Tenant’s website. (In this example *mbjulieandtheplantes.trafficmanager.net*) 2. Refresh the page. 3. Scroll down to the bottom. 4. Verify that the failover has taken effect:    * Notice the PrimaryDbServer = *Secondary Database Server*.    * Notice the SiteHost = *Secondary WebApp*.   **Note:** Occasionally, browser caching prevents the PrimaryDbServer and SiteHost settings from refreshing, and these will continue to show Primary instead of Secondary, even though the Primary Endpoint has been disabled. To resolve this, either try close your browser and open a new instance, or open a new browser window in private/incognito mode if that still doesn’t work. | Going back to the web site, we notice that the secondary database server and site host are now supporting the web site. |
|  | 1. Click **Sign In**. | Let’s also check to make sure that data such as customers’ ticket orders replicated over to the secondary server. |
|  | 1. Sign in using the account you previously created. | We sign in using customer credentials… |
|  | 1. Click **My Events**. 2. Verify that the ticket previously purchased is still listed under My Events. | …and the purchased ticket is still there. |
|  | The ticket purchase now being retrieved by the Tenant’s Secondary WebApp is from the Secondary Database Server, thanks to the active geo replication.  At this point you could terminate the replication links between the primary and secondary database servers, effectively making the secondary server primary as it switches from read-only to read/write.  Or you could start the *Primary WebApp* Endpoint in Traffic Manager, and stop the Secondary WebApp Endpoint in Traffic Manager, and resume reading and writing to the Customer1 database on the *Primary Database Server*, while keeping the replication enabled to the Secondary Database Server. For the purposes of the demo, this is the recommended option. |  |

|  |  |  |
| --- | --- | --- |
|  | **Section 2: Conclusion** |  |
|  | Geo-DR provides protection to help ensure that your databases are available to you and your customers even when large, region-scale outages strike. Powerful as this capability is, it can also be overkill for dealing with “oops” scenarios (like and inadvertanlty deleted database record). To see the Azure tool designed for these localized, one-off mistakes, ask your Microsoft sales representative about seeing the Azure SQL Database point-in-time restore demonstration. | |