Wingtip SaaS demo script

The Wingtip SaaS sample and scripts provide a great platform for demonstrating SQL Database support for a range of SaaS-related scenarios and the value of the sample/tutorials as a learning resource.

While there are many pathways possible, this script describes a series of modular demos that you can compose and adapt as needed. The first demo is a quick intro to the Wingtip SaaS resources, including the app and its architecture. The other demos explore specific SaaS scenarios. Ignoring setup time, demos can take from 5 minutes to an hour or more.

While this script is not a tutorial as such, it does offer a quick look at many of the topics, each of which is explored in much more detail in the [tutorials](http://aka.ms/sqldbsaastutorial). And while the demos are very focused on database-per-tenant SaaS scenarios, most of the capabilities covered can be used in many other contexts.

# Familiarize yourself with the app, scripts, PowerShell ISE, SSMS, Portal

The script assumes you’re familiar with the Wingtip SaaS app and the patterns. Go through the tutorials and familiarize yourself with the Azure portal, SSMS, and PowerShell ISE, including its debug/trace tools.

# What’s in the demo?

This script focuses on some of the more interesting and ‘demo-able’ scenarios. It covers:

1. A tour of the Wingtip SaaS app, a look at the online tutorials and the GitHub repo
2. Deploying the app (you’ll kick off another deployment in the background)
3. A tour of the deployed app and database resources in the Azure portal
4. Monitoring pool and database performance and setting alerts in the portal,
5. Provisioning a new tenant and exploring the catalog,
6. Using distributed query across all the tenant databases for ad-hoc analytics\*
7. Deploying a schema change across all tenant databases\*

Each demo above is independent, so can be run in any order.

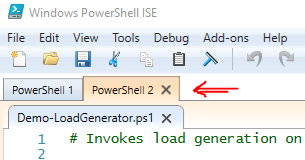
Check back for latest versions of this script which will add more demos over time.

# Setup

Allow about 15-30 minutes. Demos marked \* above need extra time.

*The Elastic Jobs feature used in 8 and 9 is in limited preview. Provide your subscription id to SaaSFeedback@microsoft.com with subject=Elastic Jobs Preview. After you receive confirmation that your subscription has been enabled, download and install the* [*latest pre-release jobs cmdlets*](https://github.com/jaredmoo/azure-powershell/releases)*. Contact SaaSFeedback@microsoft.com for any related questions or support.*

The app installs in Azure, so you need an Azure subscription. Delete the app after your demo to minimize charges. PowerShell scripts and app source code are downloaded to your demo machine, which should have the latest version of [PowerShell](https://msdn.microsoft.com/en-us/powershell/mt173057.aspx) and [SQL Server Management Studio](https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms), and [Azure PowerShell](https://github.com/Azure/azure-powershell). Also [VS Code](https://code.visualstudio.com/) or VS 2015 Community Edition or better to show app or database code.

1. Download and unblock the zip file from <https://github.com/Microsoft/WingtipSaaS> master branch to ensure you have the latest version of the scripts and code.
2. **Unblock the zip file** **BEFORE** extracting files. Right click > Properties > General > Unblock.
3. Deploy the Wingtip SaaS app from GitHub using the **Deploy to Azure** button in the readme file.
   1. Provide a resource group name, a ‘user’ value and location.   
      Always use a new resource group. The ‘user’ value is appended to pre-defined server, web site and other resource names to make them unique. Use a short value containing only lower-case letters and numbers, such as your initials plus a number, e.g. ‘abc1’. Include the user value in the resource group name, e.g. “wingtip-abc1”, to help keep track of things. Always use a new resource group and user value for each active deployment. (If you deploy a second copy of the app during the demo you might use “wingtip-abc2” and “abc2” for that deployment.)
   2. Should installation fail for any reason, try again with a new resource group and user value. Delete the resource group that failed.
4. Open …\Learning Modules\UserConfig.psm1 in PowerShell ISE and provide the resource group and <user> value used for the deployment. Pick a convenient location. These values are referenced by scripts later in the demo. Save the file. If you reinstall the app for some reason, **remember to change this file**.
5. **Wait for the deployment** **to complete** (about 5 mins). Look for the notification in the portal.
6. Open http://events.wtp.<user>.trafficmanager.net to verify. Open then close any one of the venues to start the Web app in Azure, which makes opening the apps snappier in your demo.
7. Open …\Learning Modules\Provision and Catalog\Demo-ProvisionAndCatalog.ps1 and provision the batch of tenants (demo scenario #3).   
   *This is optional, but recommended. It gives you 20+ tenant databases, which better serves to talk to the scalability of the app, and is important for the performance demos.*
8. Wait for all tenants to have deployed (typically about 5-7 mins).
9. In the PowerShell ISE use File > New PowerShell Tab (see below) or open another PowerShell ISE instance/window if you prefer.   
   
10. Open …\Learning Modules\Utilities\Demo-LoadGenerator.ps1 in the new PowerShell Tab or ISE, Adjust $DurationMinutes if needed, with a normal load (demo scenario #1) run with F5.  
    *Log in to Azure. If necessary, select the subscription for your demo. The load generator script starts a PowerShell job for each database and continues running looking for new databases. You can stop the generator using the Stop button or ctrl-c. Once stopped, you can select a different load scenario and start the generator again, which will stop prior jobs. If necessary, manage the load generation jobs using Get-Job, Receive-Job, Stop-Job and Remove-Job. Load generation mimics the unpredictable nature of many SaaS tenant workloads, so each time it is started/restarted it takes a few minutes to settle down into a new pattern. Add to that the minute or so monitoring latency, and you need to plan ahead in a demo context. Unless you start the load generator on a VM it runs locally, so if you close your laptop it will pause the jobs, which will show up as a gap in the monitoring displays later.*   
    *While the generator is running if you want to run other PowerShell scripts, switch back to the original PowerShell Tab or PowerShell ISE window.*

Extra setup tasks required for optional demos:

1. For Adhoc Analytics demo (+ 10-12 mins setup)
   1. Open …\Learning Modules\Operational Analytics\Adhoc Analytics\Demo-AdhocAnalytics.ps1
   2. Run scenario #1 to purchase tickets (can run in new PowerShell tab)
   3. Run scenario #2 to create the ad hoc analytics database (2mins)
2. For Schema Management (2-3 minutes)  
   Install the Job Account database and Job Account. This will also install and initialize the adhocanalytics database.

Open …\Learning Modules\Schema Management

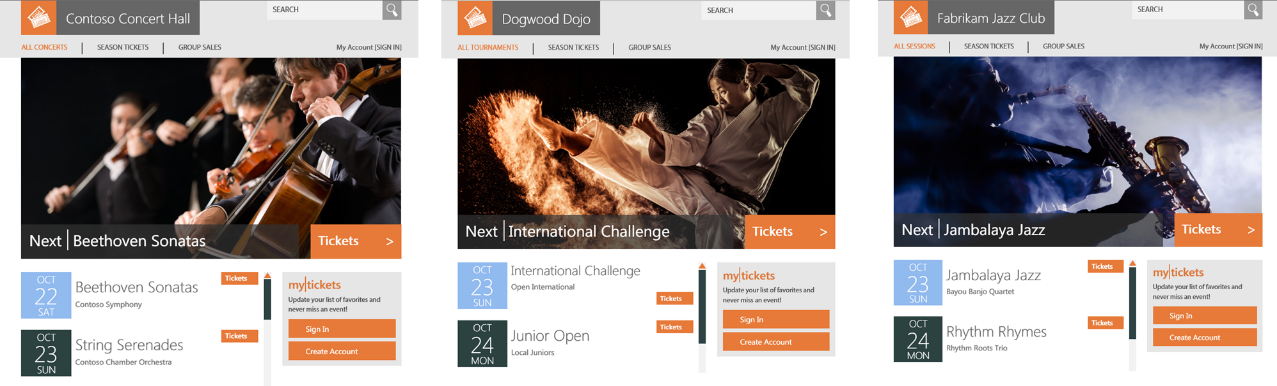
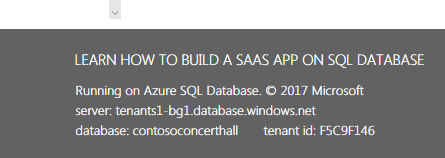
# Demos

When resource set up is complete and with load generation running, open the following for your demo

1. The [WingtipSaaS repo](https://github.com/Microsoft/wingtipSaaS) on GitHub, opened on the readme page
2. The [Introduction to the Wingtip SaaS Application](https://docs.microsoft.com/en-us/azure/sql-database/sql-database-wtp-overview), which lists available tutorials
3. The Wingtip SaaS app Events Hub page, as opened above (zoom in/out to scale display)
4. SSMS v17.0, and connect to the following using Login: developer, Password: P@ssword1
   1. catalog-<user>.database.windows.net
   2. tenants1-<user>.database.windows.net
5. Azure portal, opened on the dashboard with the subscription you’re using for the demo
6. Visual Studio (optional), opened on the Wingtip SaaS solution in the source code downloaded from GitHub if you want to show any of the app code.
7. Start [ZoomIt](https://technet.microsoft.com/en-us/sysinternals/zoomit.aspx), highly recommended for highlighting features during demos.

# Show the app

First let’s look at the app…

1. Open events.wtp*.<user>.*trafficmanager.net in a browser to open the Events hub, from where you can open individual venue experiences.
2. Open Contoso Concert Hall, Dogwood Dojo and Fabrikam Jazz Club, which each have events.  
   
3. Switch among the three venues.   
   *Highlight that this is a multi-tenant SaaS app, built using single-tenant databases (database-per-tenant). The image reflects each venue’s venue type.*
4. Click on Tickets to show that you can register as a customer of a venue, and buy a ticket.   
   *Registering and buying a ticket is not in itself important, but lets you emphasize how each venue’s data, including seating layout, event list, customer list and ticket sales data is all securely stored in a vendor-specific database.*
5. Use ZoomIt to zoom in and highlight the banner at the bottom right, where server and database names are shown. Show that each venue is connected to a different database.   
     
   *This info wouldn’t normally be exposed, but nicely highlights the database-per-tenant model*.

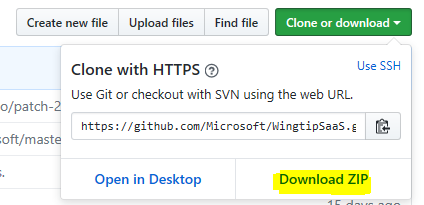
# Show the tutorials

1. Show the [Introduction to Wingtip SaaS Application](https://docs.microsoft.com/en-us/azure/sql-database/sql-database-wtp-overview) and scroll down to show the tutorial list, and the architectural overview.

*The tutorials are all scenario based, exploring different SaaS-related design and management patterns with the Wingtip SaaS sample app. None of the patterns is specific to event management/ticket selling and could be used in any domain.*   
*Tutorials are modular and can be done in any sequence.*

1. Click on the Provision

# Show the GitHub repo

1. Show the [WingtipSaaS repo](https://github.com/Microsoft/wingtipSaaS). Scroll down the readme page to show the Deploy to Azure button (don’t click it yet!)
2. Talk to how quick and easy it is to deploy (takes 5 mins and needs only two input parameters).
3. Click on the Learning Modules folder and show the next-level folders, organized by scenario, which contain PowerShell and SQL scripts, ARM templates, etc.
4. If the audience isn’t familiar with GitHub, show how to download the repo as a zip and stress the importance of unblocking the zip before extracting files.   
   

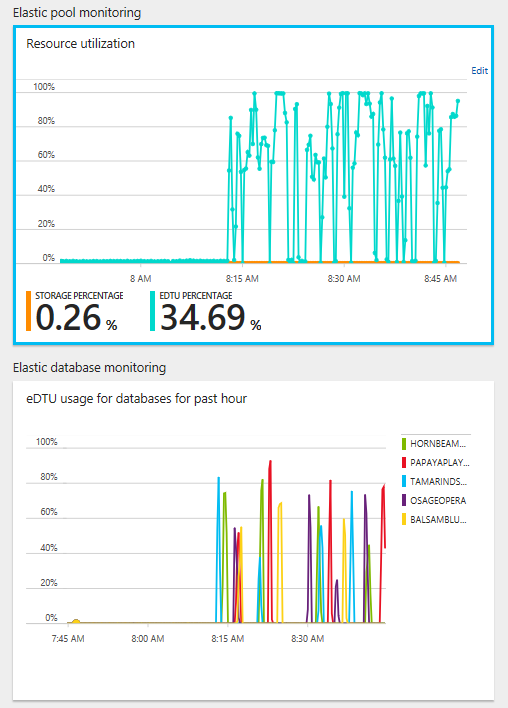
# Deploy another instance of app (optional)

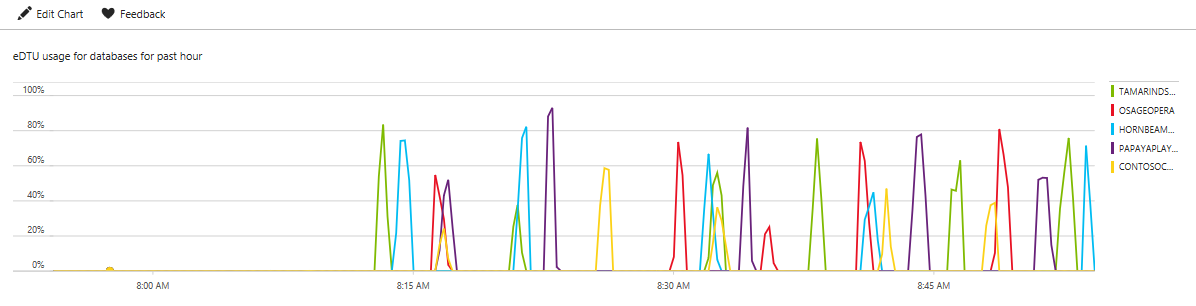
Deploy the app to show how quick and easy it is. Deploy to a new resource group with a different user value and it can be done in parallel to your main demo app. Look out for the notification in the portal later when it completes – about 5 minutes.

*Deployment uses an ARM template, which coordinates deployment of all the components and manages dependencies. All resources are deployed into one resource group; deleting this resource group once you’ve finished deletes everything and stops all billing. As it’s quick to deploy and easy to delete, you can deploy it each time you want to explore a tutorial.   
When deployment completes click on Resource Groups in the portal to show the newly created resource group and list of deployed resources.*

# Tour the Wingtip app and its architecture

1. Switch to the Azure portal and open the resource groups from the menu bar.
2. Open the wingtip-<user> resource group and show the list of application resources   
   Tip: click on the Type column header twice to sort by type, which better organizes the list
3. Highlight the catalog and tenants servers, elastic pool, databases, and web app at the bottom.
4. Click on the catalog-<user> server to open it and point out the databases.
   1. tenantcatalog, in which tenant names are mapped to databases,
   2. basetenantdb, a template database, which is copied to provision new tenants
   3. adhocanalytics (if setup), the head node db used by Elastic Query for distributed queries
   4. tenantanalytics (if setup), used as the offline analytics database
   5. jobaccount (if setup), used to hold and manage Elastic Jobs job definitions
5. Scroll left to the resources list and click on the tenants1-<user> server which hosts the default elastic pool and tenant databases.   
   *The sample installs with a single pool initially. A server is pinned to a region, and can host up to 5,000 databases, with up to 500 dbs per pool. Grouping large numbers of databases on a server allows load balancing between pools (can be demoed later).*
6. Show the list of 20 tenant databases.
7. Show the list of elastic pools – there is only one, Pool1 – and click on it to open.

*The elastic pool monitoring part shows aggregate DTU usage of all the databases in the pool.   
DTUs blend CPU, memory and read/write IO usage. If the pool shows 100% it is fully utilized, which will cause all queries to be slowed and response times to increase. Time slicing of queries is much finer than is shown in the monitoring displays, which necessarily averages this info.*   
Tip: As there is latency in reporting telemetry, start the load generator well before your demo.  
  


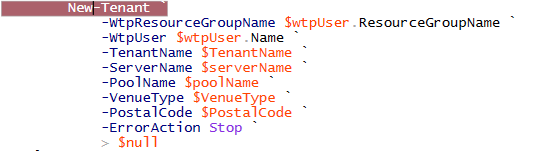
1. Discuss the database monitoring part, which shows usage by the top 5 databases, which will be different in each demo because the load is randomized. Don’t be misled by pool activity not coinciding with the apparent load on the databases shown – there are 15 other active dbs.
2. Optional. To show load on the other databases, click anywhere on the database part to open another blade that lets you select any five databases (unselect one first to select another).   
   Expand blade to full page for a more interesting graph.   
     
   *Here you can more clearly see the varied and unpredictable nature of the load, typical of many SaaS apps, and how loads from multiple databases may overlap. The elastic pool absorbs this unpredictability but can still be overloaded if many dbs burst together. Sizing the pool requires understanding the likelihood of this occurring among the specific dbs in the pool. The more dbs in the pool the greater the averaging effect.*   
   *As a reference point, an Australian accounting software vendor runs nearly 500 databases in 200 eDTU pools, with each database able to burst to 100 eDTUs.*

# Provision a new tenant

In this demo you provision a new tenant, creating a new tenant database and registering it in the catalog. If the load generator is running, it will spot the new database has been added after a few seconds and start a load running on it.

While provisioning could be integrated into a ‘registration’ app, in the sample it’s done via PowerShell.

1. Open …\Learning Modules\Provision and Catalog\Demo-ProvisionAndCatalog.ps1 in PowerShell ISE.
2. Review TenantName, VenueType parameters, and with scenario #1, press F5 to run.

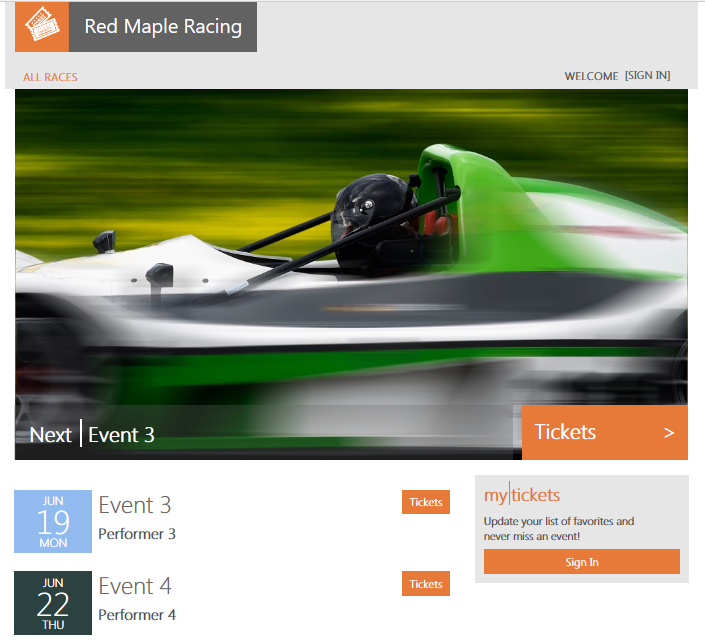
Optional: Trace the execution of the script. To do this, set a breakpoint on the call to the New-Tenant function by clicking anywhere in the line and pressing F9. Then press F5 to run.  


Use F11 to step inside the function to trace its execution. Use F10 to step through the function. Use F11 to step into called functions, shift-F11 to step out, and F5 to run to end.

Provisioning a tenant involves the following:

* Creating a tenant key from a hash of the tenant name
* Checking the tenant doesn’t already exist in the catalog
* Deploying the tenant database using an ARM template that creates a database as a copy of the basetenantdb (deployment progress is reported in the command window)
* Registering the tenant database in the catalog using the tenant key, also adding the tenant name to the Tenants table.

*The catalog is implemented using the Shard Management features of the Elastic Database Client Library. In this case it’s accessed via PowerShell cmdlets, so it can be scripted. It can also be built into a client application. You can choose to drill down during script execution to see the calls into the library. While the sample use the tenant name as a database name other strategies could be used.*

1. After the tenant is provisioned the script opens the Events page for the new tenant.  
   

# Show the tenantcatalog database in SSMS

Explore this if there is interest in how the catalog is implemented.

1. Locate the tenantcatalog database on the catalog-<user> server
2. Expand the database to show the tables

*Note the tables in the \_\_ShardManagement schema, these tables are created by and managed by the Elastic Database Client Library and which implements the catalog storage. These \*Global shard management table have a corresponding set on individual ‘shard’ (tenant) databases. Stress that although it can sometimes be insightful to look at this content never edit it manually as its likely to break the shard management and is not supported. For more info on the elastic database client library see:* <https://docs.microsoft.com/en-us/azure/sql-database/sql-database-elastic-database-client-library>

1. Right click on ShardsGlobal and select, Select Top 1000 Rows, and then scroll right to show the ServerName and DatabaseName columns.
2. Right click on the Tenants table and select, Select Top 1000 Rows  
   *This table is an extension table that holds additional metadata about tenants. In this case, it holds the Tenant’s name and the service plan they have signed up under (‘Standard’). This table could be extended to add more information about the tenant, such as contact info or links into other business systems. The primary key of this table is the internal key, used in the ShardMappingsGlobal table (in the MinValue column).*
3. Expand the Views in the object explorer. Right click on TenantsExtended and Select Top 1000 Rows.   
   *This returns tenant information joining the Tenants table via the mappings table and shards table to retrieve the database location for each tenant. This information, for example, allows devops to track down the database for a tenant (very useful if the database name is opaque)*

# Adhoc analytics - distributed query across tenant databases

In this demo, you’ll use an additional head node database, combined with SQL Database’s Elastic Query technology to run distributed queries across all tenant databases.

This demo requires the adhocanalytics database has been deployed to the catalog-<user> server during setup and the ticket generator has been run.

Note: if you have already run the schema management demo the adhocanalytics schema will already have been deployed. If so, you can review the schema in the first step, but you don’t execute it.

The first step is to initialize the schema.

1. Open …\Learning Modules\Operational Analytics\Adhoc Analytics\Initialize-AdhocAnalyticsDB.sql and ensure it is connected to the adhocanalytics database.
2. Review the schema definition script which does the following:
   1. Defines a master key
   2. Creates a database scoped credential used to authenticate access to the catalog database and to the tenant databases.
   3. Defines the shard map in the catalogdatabase as the external data source, which defines the set of databases to which queries will be distributed.
   4. Defines external tables that mirror views defined in each tenant database. Note how the distribution for each table is defined as sharded(VenueId). This indicates to the query optimizer that the data for each venue is located in a single database (this allows joins on venue id to be executed remotely).
   5. Defines a local table, VenueTypes, a reference data table that contains the same venue types that are in every tenant database.
3. Execute the script with F5 to deploy the schema

Now you can run distributed queries across all the tenant databases.

1. Open ...\Learning Modules\Operational Analytics\Adhoc Analytics\Demo-AdhocAnalyticsQueries.sql and ensure it is connected to the adhocanalytics database.
2. Select the first query SELECT \* FROM dbo.Venues and execute it with F5. This simple query demonstrates the power of Elastic Query, retrieving the Venue record from each tenant database.
3. From the Query menu select Include Actual Execution Plan.
4. Select and execute the second query, which shows the Venue Type of each Venue.
5. Show the execution plan, which shows how the external Venues table is joined to the local VenueTypes table.
6. Select and execute the third query which find the most popular venue type by selecting the count the number of tickets sold for each venue and aggregating the results by venue type.
7. Show the execution plan, and hover over the remote query to show how the join has been pushed down to be executed remotely. And although the query is aggregating over nearly 100,000 tickets, only 1 row is returned per tenant. By pushing the join down to each database, they are executed in parallel, allowing the query to execute extremely efficiently.

# Deploying a schema update to all tenant databases

This demo uses Elastic Jobs to deploy a schema change to all tenant databases, plus the basetenantdb to ensure new tenants have the same change. In this demo, the schema change involves the addition of two new venue types, *swimming* and *motorcycleracing*, to the VenueTypes reference data table, but any DDL change could be applied using the same mechanism. Once complete you can add a new venue using one of the new VenueTypes or change the VenueType of an existing one in SSMS.

***IMPORTANT: Requires your subscription has been enabled for Elastic Jobs and you have installed the extended version of Azure PowerShell incorporating the Job Account cmdlets. See Setup for details.***

Also requires the Job Account and jobaccount database are deployed first. Uses SSM pre-connected to your catalog and tenant server. See Setup.

Note: As the schema change deployed in this demo involves modifying data in the reference table VenueTypes, it also targets the adhocanalytics database, which contains a local copy of the VenueTypes table. For this reason, setup for this demo deploys the adhocanalytics database if not already deployed. You may prefer therefore to do the adhoc analytics demo first, which introduces the adhocanalytics db.

1. Open the Wingtip app Events Hub, click on Balsam Blues Club  
   *Venue types are pre-defined by the Wingtip App. Invent story for this club changing to a swimming club, a new VenueType you are going to introduce into the app.*
2. In SSMS, browse to the *balsambluesclub* database on the tenant server, open a new query and run SELECT \* FROM VenueTypes to show 10 venue types.  
   *You are going to run a job to add motorcycleracing and swimming.*
3. In SSMS, open …\Learning Modules\Schema Management\DeployReferenceData.sql
4. Modify line 6, SET @WtpUser = '<user>', replacing <user> with your User value
5. Ensure you are connected to the jobaccount database on catalog-<user>
6. Review the purpose of the script. Point out the following:
   1. *The target group contains not only all databases on the tenants1-<user> server, but also basetenantdb and the adhocanalytics database.*   
      *The target group is dynamic, so if additional databases are added to the server they will automatically be included in scope. Groups can also be based on elastic pools or arbitrary sets of databases. Scoping a job on the databases registered in a shard map (as used in the catalog) will be added to elastic jobs.*
   2. *The job is defined to be executed once but could be set up on a schedule.*
   3. *The job script uses MERGE so is idempotent. Should there be a connection failure between the job agent and the database, a job will be retried and so may run more than once.*
7. Execute the script using F5.
8. Review the results:
   1. Job definition
   2. Job step definition
   3. Job execution
   4. Job step execution
   5. Job step execution summary (state of execution instances running on target dbs)
9. Once the job has completed, re-execute the job step execution or job step execution summary until job shows as succeeded. Should take only a few seconds.
10. Re-execute the query SELECT \* FROM VenueTypes to show there are now 12 venue types
11. Expand Object Explorer to locate Venue table in balsambluesclub database. Right click and select Edit Top 200 Rows
12. Overtype the VenueType value with *swimming* (all lowercase) and press Enter to update
13. Return the Balsam Blues web page and refresh, and voila!

# Deleting the apps when you finish

As the demo environment is quick to set up, it’s recommended you delete the app(s) at the end of the demo as the last part of the demo. Doing this highlights the role of a resource group as an application container, and reinforces how easy it is to use the tutorials.

1. In the portal click on Resource groups in the left hand menu, select the resource group containing the app and then click on Delete.