Scenario Guidance: SaaS Azure SQL Database Elastic Pools

Scenario Guidance 1

Balance costs vs tenant performance 1

Freemium Models and Elastic Pools 1

When to Create New Pools 2

When to Adjust eDTU’s Allocated to a Pool 3

Pool Management Options 3

Performing Schema Maintenance on Pooled Databases 3

Monitoring & Alerting 4

Database Recovery 5

# Scenario Guidance

Before exploring Elastic Pools “hands-on” by running the load generator and Monitoring Web App in the ShopKeeper scenario, you should review the following guidance.

## Balance costs vs tenant performance

In a multi-tenant scenario, appropriately balancing the cost of resources versus tenant performance is critical. An imbalance in one direction means that too much is being spent on resources and costs are high. An imbalance in the other direction means that tenants experience unacceptably slow performance.

### Freemium Models and Elastic Pools

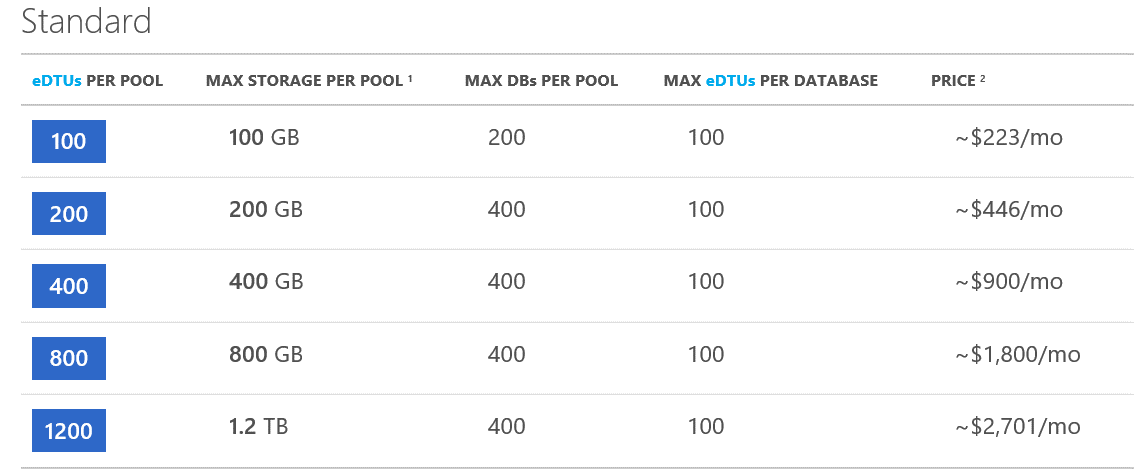
In most SaaS application scenarios, like ShopKeeper, there is a notion of a freemium subscription model. In this model, the solution as it is sold to end-customers is priced in different tiers, such as free and paid, with the notion that there is a friction free upgrade path from the free subscription (where most customers start out) to the paid subscription.

In the ShopKeeper scenario, for example, assume Contoso has a Free Subscription and a Paid Subscription. Contoso’s goal for the Free Subscription is to keep the per tenant costs as low as possible because the tenants are not paying for this consumption directly (it might be paid from revenue generated by paying customers, or by other means such as a transaction fees or advertising). The Paid Subscription should still aim for cost-effectiveness, but because tenants are paying for usage in this Subscription, it is likely the per tenant costs can be higher (and provide improved peak performance or storage capacity).

So how does this map to how Contoso might leverage Elastic Pools? The database cost per tenant is effectively the cost of the pool divided by the number of tenant databases in the pool. The number of databases that can be added to any given pool is limited by the pricing tier of the pool. For example, a Standard 200 pool supports up to 400 databases. Currently this tier is priced at $446 USD per month. If Contoso were to fully utilize the pool by adding 400 databases, the cost per tenant would near $1.12 per tenant per month. Similarly, if they used the Basic 200 tier (which currently is priced at $298 USD per month), then the cost per tenant- month would near $0.75.

**Elastic Database Pricing**

The latest pricing for Elastic Databases is available from <https://azure.microsoft.com/en-us/pricing/details/sql-database/>



Naturally, they might consider using Pools in the Basic Tier for their Free Tier merchants in order to realize the lowest cost per tenant-month for that set of tenants. They might then consider reserving Pools in the Standard Tier for their Paid Tier merchants. Alternately, they may choose to use standardize on just one Pool service tier for both Free and Paid Subscriptions. The goal of getting the lowest cost per tenant month would remain the same.

An important consideration here is that once a Pool is created, the service tier selected cannot be changed. To change the service tier of a Pool amounts to creating a new Pool with the desired tier, removing the databases from the existing Pool and the adding them into the new Pool. While this can be done without any down-time with respect to database access, it is not something you would want to perform on a regular basis on account of the process being time consuming.

### When to Create New Pools

Given Contoso’s freemium model, the optimal configuration of an Elastic Pool from a per tenant-month cost perspective is to maximize the number of databases it contains, and the least desirable situation is creating a Pool for only a single database, because the Pool has a fixed price regardless of whether there is 1 database or 400 databases within it. As their tenant count grows, at some point they will reach the limit of the number of databases the pool allows—this when they should consider creating a new Pool.

Contoso could also consider moving selected databases into a new pool, especially when a particular database has more consistent demand than others, when a pool is hitting its eDTU capacity limits. Also you may want to split out the databases into separate pools, when multiple databases in the pool show a pattern of consistent spiking at the same time

### When to Adjust eDTU’s Allocated to a Pool

There are many reasons why Contoso might consider adjusting the number of eDTU’s allocated to a Pool. For example, when they first create a Pool to handle the situation when other existing pools are at capacity, they might create the new Pool with the minimum number of eDTU’s to reduce the all-up cost of the Pool and then scale up the number of eDTU’s on the Pool as more databases are added.

Another reason Contoso would adjust the number of eDTU’s allocated to a pool are if their merchants are encountering seasonal fluctuations or other such fluctuations. It is important to note, that while this capability is supported by Elastic Pools, it is not one that should be used frequently as the scaling up or down of eDTU’s takes time to complete—on the order of hours.

### Pool Management Options

The provisioning of new Pools, adjusting the eDTU’s assigned to a Pool or the migration of databases between Pools can be accomplished manually using the Azure Portal or it can be automated via C# (using the SQL Database Library for .NET) or PowerShell cmdlets (using Azure PowerShell 1.0 or higher).

**NOTE: Examples of Pool Management**

For examples of management with the above options, see <https://azure.microsoft.com/en-us/documentation/articles/sql-database-elastic-pool-manage-portal/>

## Performing Schema Maintenance on Pooled Databases

The Contoso ShopKeeper solution demonstrates an example of the challenge faces by SaaS solutions with regards to managing schemas. In the Quick Start, it deploys an instance of the AdventureWorks database for each tenant. Therefore, Contoso would have many copies of a database with the same schema, albeit different data. This raises the question, how should Contoso roll out schema updates when required (e.g., because of application updates), without affecting the per tenant data?

Using the Azure Portal, as illustrated in this Quick Start, Contoso can use Elastic Jobs to coordinate the execution of a T-SQL script against all of the databases in an Elastic Pool. The important consideration when taking this approach is that the T-SQL script must be written so it is idempotent. That is, running the script multiple times against a single database does not corrupt the target database or raise errors. The approach shown in this Quick Start is to perform checks that examine if the script has already been run against the target database, and to gracefully complete (instead of executing any changes) if the script has previously completed.

Besides using the Portal, Contoso may consider using PowerShell to control the execution of Elastics Database Jobs. In addition to the automation opportunities this allows for the deployment of updates, using PowerShell has one characteristic that is not present in the Portal: custom groups. With custom groups Contoso can target its T-SQL script to execute on a specific set of databases, instead of all the databases within the Pool.

NOTE: For examples of using PowerShell to create and manage Elastic Database jobs, see <https://azure.microsoft.com/en-us/documentation/articles/sql-database-elastic-jobs-overview/>

## Monitoring & Alerting

For Contoso, balancing performance versus cost is critical, and for their ShopKeeper application the cost is directly correlated with the number of Elastic Pools they have allocated. Therefore, they want to monitor their Pools closely so they know when to take actions such as creating new Pools or moving databases between Pools.

This begs the question, how can they setup notifications if their load is overwhelming the pool? In the Quick Start, we demonstrate configuring alerts on the Pool using the Azure Portal. Contoso can configure alerts for metrics such CPU %, eDTU %, sessions %, storage %, workers % which when can be tracked in the portal when they trigger, or they can be used to send an email out to admins.

If the capacity concern has more to do with specific tenant database instances, Contoso can also configure alerts on a per database level, in a similar fashion as they do for Pools, by using the Azure Portal.

For both Pool alerts and per Database alerts, Contoso can configure Web Hooks that will perform an HTTP POST to the endpoint of their choosing when the alert is triggered, in addition to having an email sent out to administrators. The payload of this HTTP POST contains the information about the alert that was configured (e.g., the alert name, description and metric configuration), but also the value that caused the alert to trigger. This enables Contoso to extend their ability to react to alerts by using their own management web app or to send out notifications (e.g. using Azure Notification Services) or SMS text messages (e.g., using Twilio).

When it comes to monitoring the Pool and the databases Contoso can choose to use the Azure Portal as well as T-SQL, as we show in this Quick Start with the Monitoring Web App. They can use the T-SQL options to collect the telemetry from Azure and store it in their own log analytics solution. This would enable them to perform analysis on the telemetry that spans much longer periods of time than that permitted by the retention policy of the data when it is managed by Azure—for example, enabling them to review pool usage over the course of months instead of the 14 days that is maintained by Azure.

## Database Recovery

By utilizing Elastic Pools, Contoso gets an improved ability to juggle cost versus tenant performance, but does not lose any of the features supporting availability and disaster recovery that are available to SQL Databases outside of a Pool. For example, they can use Point in Time Restore to recover from user error, such as a DBA accidentally dropping the customers table in a tenant’s database. To accomplish this with minimal down-time, they would ensure Pool to which they will restore has capacity for another database instance, rename the original Database to a temporary name and then restore to new database with same name as original. In this fashion, they could restore the database without having to make any application level changes, such as altering connection strings, because the restored database name would be unchanged.

They also get to benefit from the restore deleted database feature to provide a window of time during which their system could actually delete the database when a merchant cancels, but be able to restore that database should the customer re-join within the retention window. This retention window is controlled by the service tier of the pool: 7 days for Basic, 14 days for Standard and 35 days for Premium.