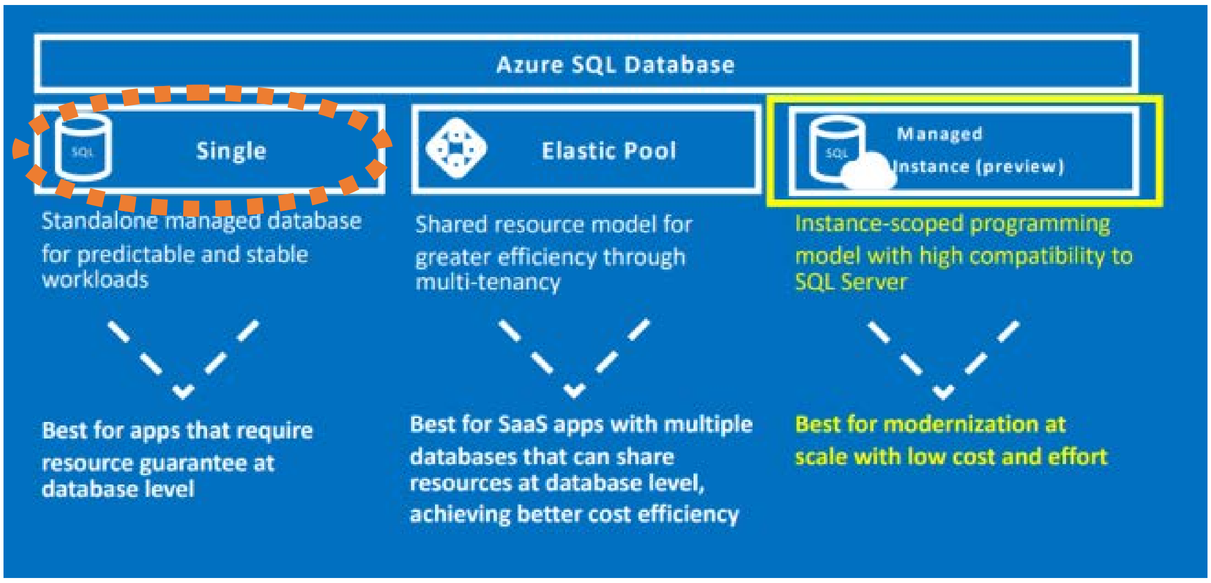
Problem Statement : Why use Azure SQL Single database?

Many enterprises or companies move their data pipelines into the cloud, they do it for the advantages of distributed storage and computing, such as scaling, cost-efficiency over outsourcing in-house IT infrastructure, and maximizing data availability. And many actually aren't aware of how simple it is to move into such a solution. All you need really is the Azure SQL Database component.

Solution


Azure SQL Database
Single
Standalone managed database for predictable and stable workloads
Best for apps that require resource guarantee at database level
Elastic Pool
Shared resource model for greater efficiency through multi-tenancy
Best for SaaS apps with multiple databases that can share resources at database level, achieving better cost efficiency
Managed Instance (preview)
Instance-scoped programming model with high compatibility to SQL Server
Best for modernization at scale with low cost and effort

A single database can be moved into or out of an elastic pool for resource sharing. For many businesses and applications, being able to create single databases and dial performance up or down on demand is enough, especially if usage patterns are relatively predictable. But if you have unpredictable usage patterns, it can make it hard to manage costs and your business model. Elastic pools are designed to solve this problem. The concept is simple. You allocate performance resources to a pool rather than an individual database and pay for the collective performance resources of the pool rather than for single database performance.

The single database resource type creates a database in Azure SQL Database with its own set of resources and is managed via a server. With a single database, each database is isolated, using a dedicated database engine. Each has its own service tier within the DTU-based purchasing model or vCore-based purchasing model and a compute size defining the resources allocated to the database engine.

Single database is a deployment model for Azure SQL Database. The other is elastic pools.

Now, let us see the steps to create a Single database in Azure SQL Database.

Implementation

Pre- requisites or conditions:

a. A live or ongoing Azure subscription. If you do not have a subscription, you can go for a free account.

b. The rearmost interpretation of Azure PowerShell or Azure CLI.

Steps to create a Database on Azure portal:

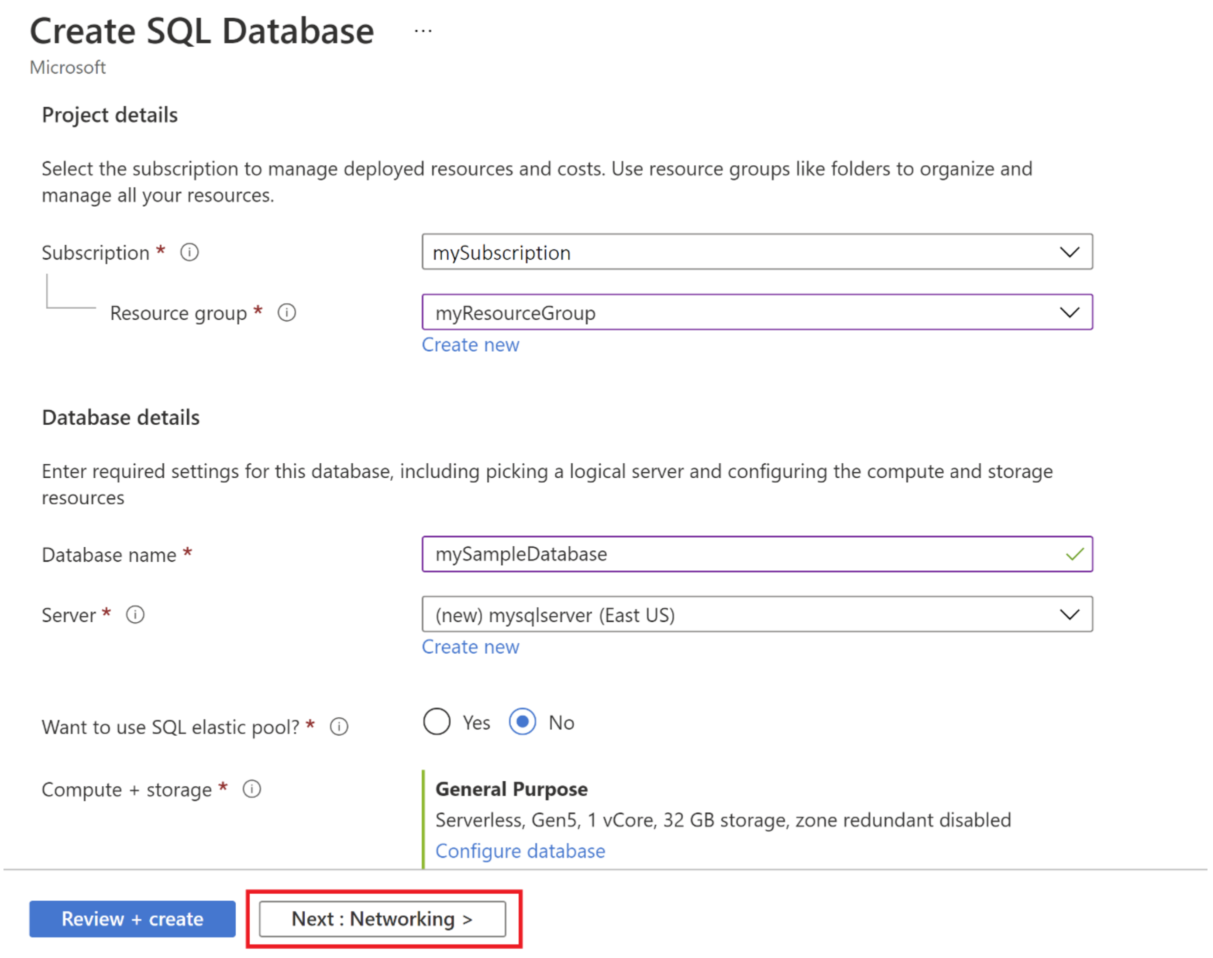
1. To create a **single database** in the Azure gate:

- Go to Select SQL Deployment option runner.
- Under SQL databases, select Single database type under resource type option, finally click on create.

2. We need to fill in a few details under the tab of the Create SQL Database, Enter the Project details, and the desired Azure Subscription.

3. In the Resource group, click on Create New, enter the name for that group let’s say ‘myResourceGroup’ and click on OK.

4. For the name of Database you can enter any name let’s enter ‘mySampleDatabase’.


Create SQL Database

Microsoft

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *	mySubscription
Resource group *	myResourceGroup

Database details

Enter required settings for this database, including picking a logical server and configuring the compute and storage resources

Database name *	mySampleDatabase
Server *	(new) mysqlserver (East US)
Want to use SQL elastic pool? *	Yes [✓ No]
Compute + storage *	General Purpose Serverless, Gen5, 1 vCore, 32 GB storage, zone redundant disabled

5. Coming to the Server, select produce new tab, fill the new details form as follows:

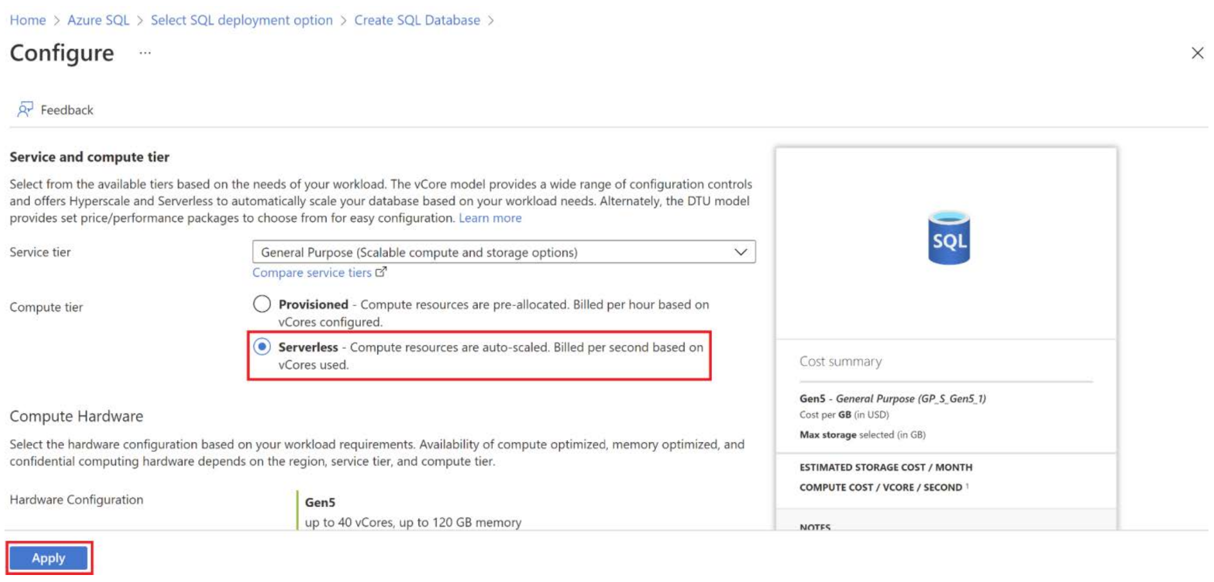
- Under name Enter mysql server, and add some characters for oneness.
- The Server name would be unique across all the Azure server, so once we enter the name portal will show us whether the name is available or it is in use already.
- Location, select from the dropdown list.

6. Authentication system, for that Select Use SQL authentication.

7. Server admin login, enter the azure user example - azureuser1. Enter a word, and Confirm the word field. Click on OK. Skip the “Want to use SQL elastic pool”, set to NO.

8. Configure Database, under Compute + Storage.

9. After that we use a serverless database, so let the Service tier be set to General Purpose and set Compute tier to Serverless and select Apply.


Home > Azure SQL > Select SQL deployment option > Create SQL Database >

Configure ...

Feedback

Service and compute tier

Select from the available tiers based on the needs of your workload. The vCore model provides a wide range of configuration controls and offers Hyperscale and Serverless to automatically scale your database based on your workload needs. Alternately, the DTU model provides set price/performance packages to choose from for easy configuration. Learn more

Service tier	General Purpose (Scalable compute and storage options)
	Compare service tiers
Compute tier	**Provisioned** - Compute resources are pre-allocated. Billed per hour based on vCores configured.
	[✓ **Serverless** - Compute resources are auto-scaled. Billed per second based on vCores used.]

Compute Hardware

Select the hardware configuration based on your workload requirements. Availability of compute optimized, memory optimized, and confidential computing hardware depends on the region, service tier, and compute tier.

Hardware Configuration	**Gen5** up to 40 vCores, up to 120 GB memory

Cost summary

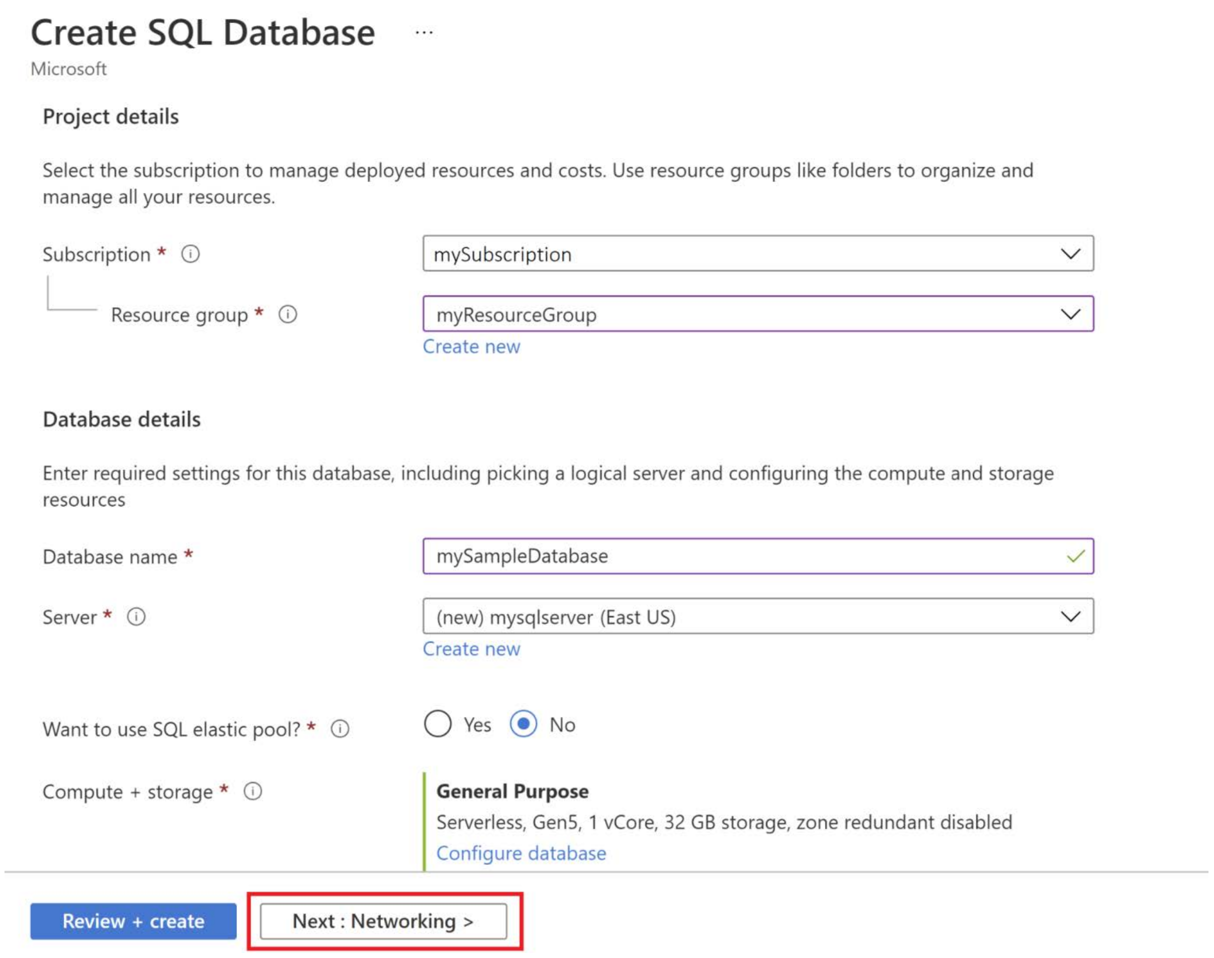
Gen5 - *General Purpose (GP_S_Gen5_1)*
Cost per **GB** (in USD)
Max storage selected (in GB)

ESTIMATED STORAGE COST / MONTH
COMPUTE COST / VCORE / SECOND

NOTES

10. In Provisory storehouse redundancy, select a redundancy option for the storehouse account where your backups will be saved.

11. Now, select Next Networking at the bottom of the page On the Networking tab, select Public endpoint, for Connectivity system.


Create SQL Database ...

Microsoft

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

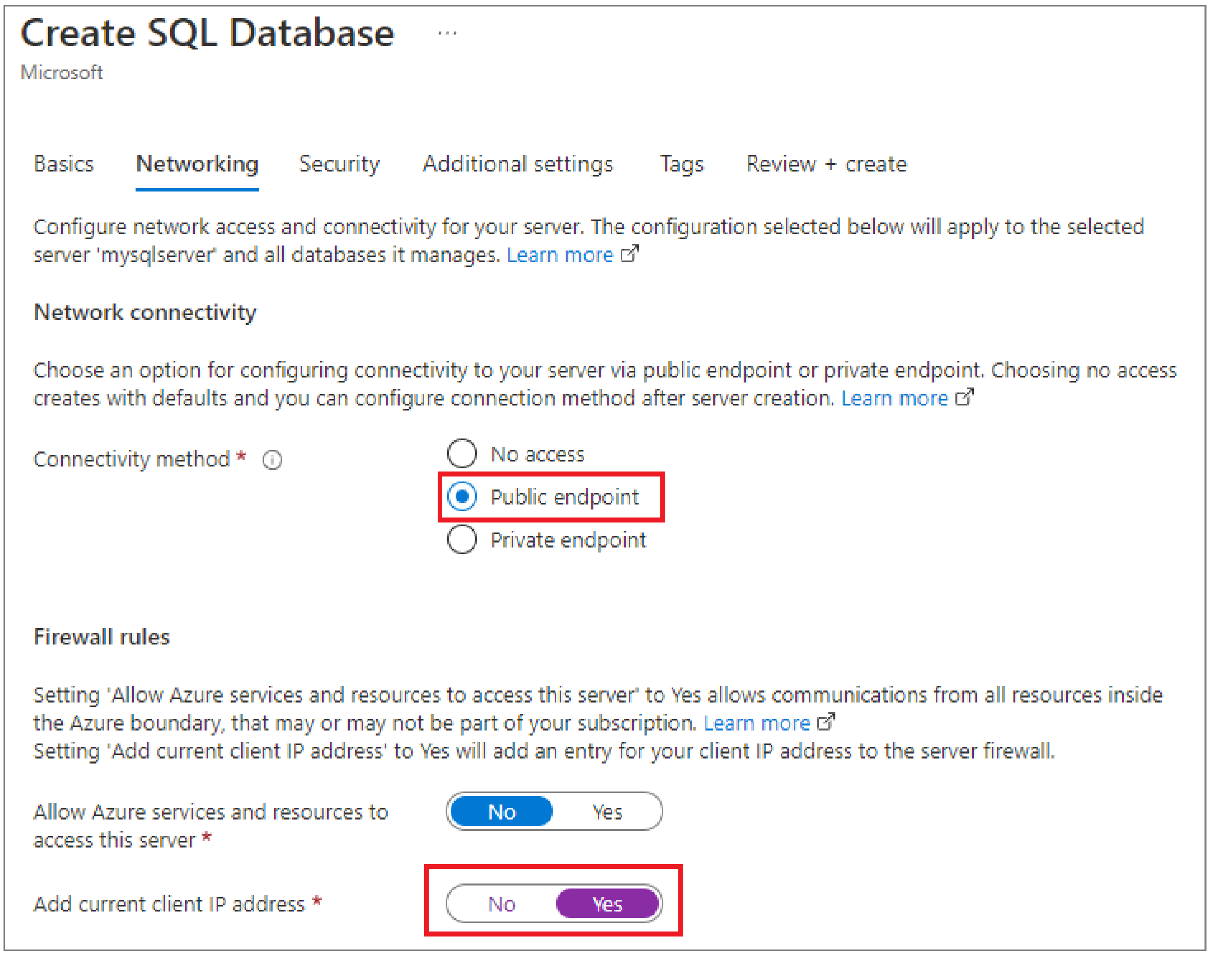
Subscription *	mySubscription
Resource group *	myResourceGroup
	Create new

Database details

Enter required settings for this database, including picking a logical server and configuring the compute and storage resources

Database name *	mySampleDatabase
Server *	(new) mysqlserver (East US)
	Create new
Want to use SQL elastic pool? *	Yes [✓ No]
Compute + storage *	**General Purpose** Serverless, Gen5, 1 vCore, 32 GB storage, zone redundant disabled
	Configure database

12. Set, “Add current customer IP address” to YES for Firewall rules. Now leave Allow Azure services and coffers to pierce this server set to No.



Create SQL Database ...

Microsoft

Basics | **Networking** | Security | Additional settings | Tags | Review + create

Configure network access and connectivity for your server. The configuration selected below will apply to the selected server 'mysqlserver' and all databases it manages. Learn more

Network connectivity

Choose an option for configuring connectivity to your server via public endpoint or private endpoint. Choosing no access creates with defaults and you can configure connection method after server creation. Learn more

Connectivity method * : No access [✓ Public endpoint] Private endpoint

Firewall rules

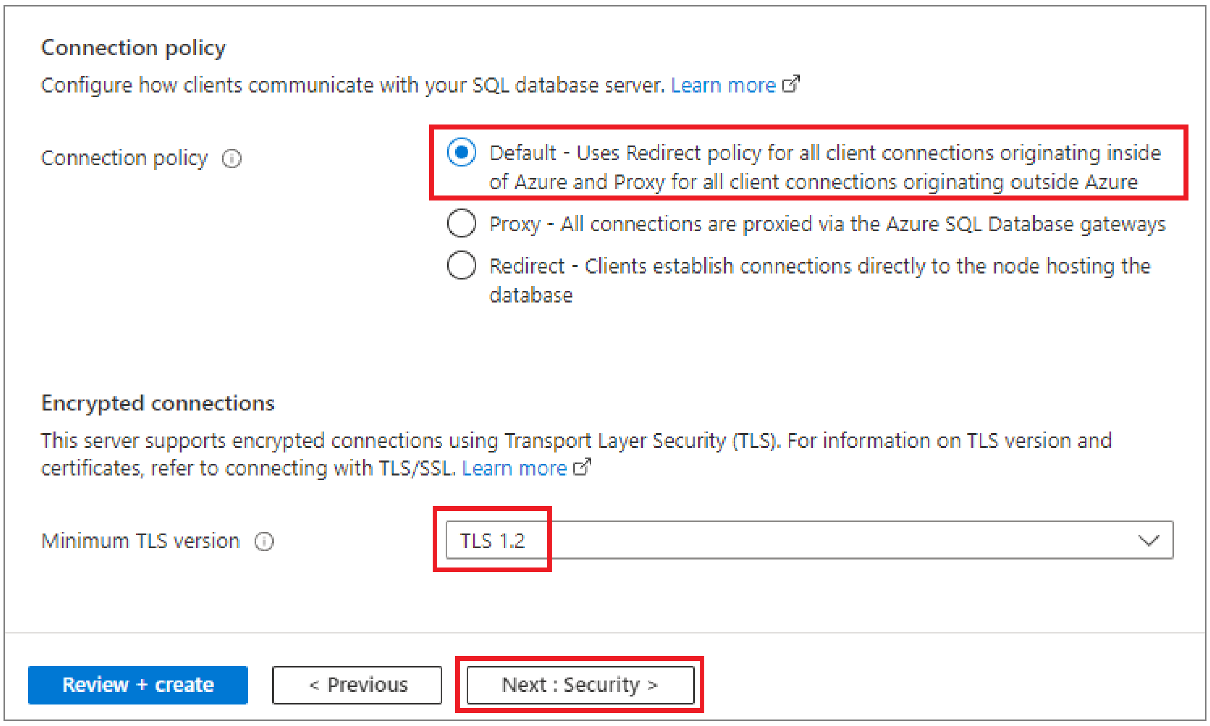
Setting 'Allow Azure services and resources to access this server' to Yes allows communications from all resources inside the Azure boundary, that may or may not be part of your subscription. Learn more
Setting 'Add current client IP address' to Yes will add an entry for your client IP address to the server firewall.

Allow Azure services and resources to access this server * : [✓ No] Yes

Add current client IP address * : No [✓ Yes]

13. Choose the connection policy in Connection policy and leave the minimal TLS interpretation at the dereliction.

14. Select Next Security at the bottom of the runner.You can choose to start a free trial of Microsoft Defender for SQL, as well as configure Ledger, Managed individualities and Transparent data encryption(TDE) if you ask on the Security runner.


Connection policy

Configure how clients communicate with your SQL database server. Learn more

Connection policy: [✓ Default - Uses Redirect policy for all client connections originating inside of Azure and Proxy for all client connections originating outside Azure] Proxy - All connections are proxied via the Azure SQL Database gateways / Redirect - Clients establish connections directly to the node hosting the database

Encrypted connections

This server supports encrypted connections using Transport Layer Security (TLS). For information on TLS version and certificates, refer to connecting with TLS/SSL. Learn more

Minimum TLS version: TLS 1.2

15. Select Coming fresh settings at the end of the runner.

16. Configure additional settings, in the Data source section, Select use existing data and select the sample. This makes an AdventureWorksLT sample database hereafter, there are some tables and data for querying and experimenting with, as opposed to an empty blank database. Also, you can configure a conservation window and database collation.

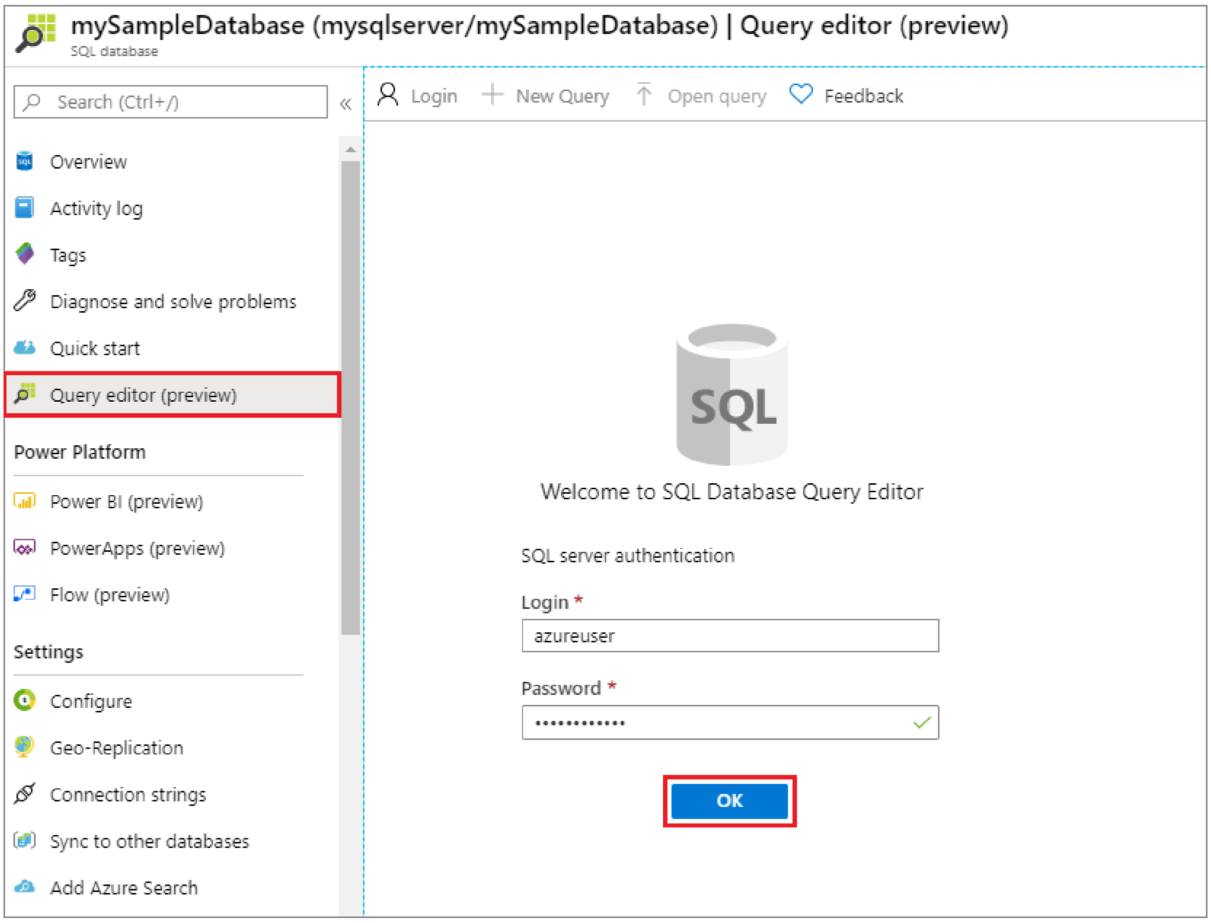
17. Select Review produce at the bottom of the runner, select produce on the Review produce runner, after reviewing.

Here, we created a Database on the Azure portal. Let's move to querying the database.

Once we have our Database up and running, we can make queries on the data using the Query editor(preview) on Azure Portal, through this we shall be able to connect to teh database and fetch the results of Queries.

Query the database:

1. Hunt for and select SQL databases in the portal, and select your database from the list.
2. Select the Query editor (exercise) in the left menu on the runner for your database.
3. Enter your server admin login information and also select OK.
4. Enter the following query in the Query editor pane.


mySampleDatabase (mysqlserver/mySampleDatabase) | Query editor (preview)
SQL database

Welcome to SQL Database Query Editor

SQL server authentication

Login *
azureuser

Password *
••••••••••••

SQL Query:

SELECT TOP 25pc.Name as CategoryName,p.name as ProductName
FROM SalesLT.ProductCategory pc
JOIN SalesLT.Product p
ON pc.productcategoryid = p.productcategoryid;

5. Select Run, and also review the query results in the Results pane.
6. Close the Query editor runner, and Select OK when asked to delete the unsaved edit.

AZURE CLI METHOD

: subscription="<subscriptionId>" # add subscription here

az account set -s $subscription # ...or use 'az login'

Setting parameters value

# Variable block

let "randomIdentifier=$RANDOM*$RANDOM"

```
location="East US"
resourceGroup="msdocs-azuresql-rg-$randomIdentifier"
tag="create-and-configure-database"
server="msdocs-azuresql-server-$randomIdentifier"
database="msdocsazuresqldb$randomIdentifier"
login="azureuser"
password="Pa$$w0rD-$randomIdentifier"
# Specify appropriate IP address values for your environment
# to limit access to the SQL Database server
startIp=0.0.0.0
endIp=0.0.0.0

echo "Using resource group $resourceGroup with login: $login, password: $password..."
```

How to Clean up Resources used?

Keep the resource group,server and single database to go on to the coming way, and learn how to connect and query your database with different styles.
When you are done using these resources, you can choose to delete the resource group you created. This will also delete the server and single database within it.

How to Clean up Resources used?

Steps :

1. In the portal, hunt for and elect Resource groups, and also elect “myResourceGroup” from the list.
2. On the resource group runner, elect cancel resource group.
3. Under Type the resource group name, enter myResourceGroup, and also select cancel.

In this way you can create the Database and query it, we also saw how we can clean up resources and delete the server and single database within it. We saw all these on Azure

Portal, we can also do it using Azure CLI and Powershell. We can connect and query the database using other tools and languages as well like SQL Server Management Studio , Azure Data studio.

Knowledge Sharing for Best Practices:

Migrating to a single database with minimal downtime

These quickstarts enable you to quickly create or import your database to Azure using a .bacpac file. However, .bacpac and .dacpac files are designed to quickly move databases across different versions of SQL Server and within Azure SQL, or to implement continuous integration in your DevOps pipeline. However, this method is not designed for migration of your production databases with minimal downtime, because you would need to stop adding new data, wait for the export of the source database to a .bacpac file to complete, and then wait for the import into Azure SQL Database to complete. All of this waiting results in downtime of your application, especially for large databases. To move your production database, you need a better way to migrate that guarantees minimal downtime of migration. For this, use the Data Migration Service (DMS) to migrate your database with the minimal downtime. DMS accomplishes this by incrementally pushing the changes made in your source database to the single database being restored. This way, you can quickly switch your application from source to target database with the minimal downtime.

Challenges :

Memory bottlenecks: Memory bottlenecks can lead to delay in application response, overall system slowness, or even application crashes.

CPU bottlenecks: Insufficient hardware usage, sudden CPU spikes, as well as complex and time-consuming queries are some of the reasons for CPU bottlenecks.

Given the complexities involved in tracking the performance of Azure SQL Database, you might need a little help.

Under some circumstances, you may need to shrink a database to reclaim unused space.

Benefits

Where do I start, really. Costs? Of course. You pay only for u used per second. Scale for demand, up or down, pause and pay only for storage during inactive periods, and seamlessly resume when workloads come back. , each database is isolated, using a dedicated database engine This is especially brilliant for databases with intermittent, unpredictable usage patterns interspersed with periods of inactivity, and lower average compute utilization over time. And last, but not the least, New single databases without usage history where compute sizing is difficult or not possible to estimate prior to deployment in SQL Database.