

Application to Generate Data

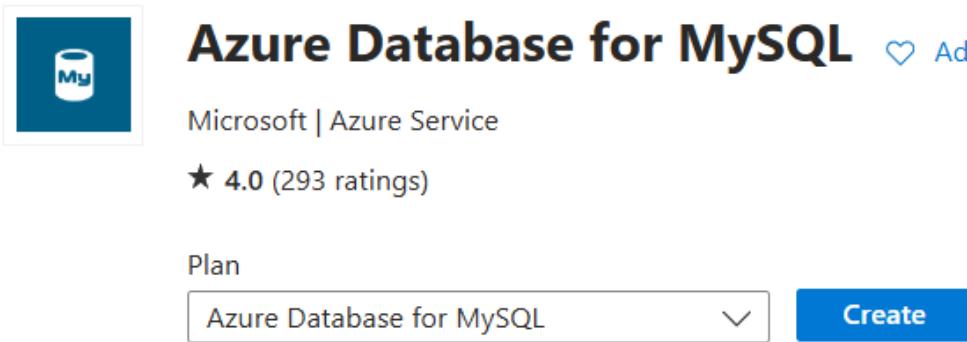
The process involves setting up an Azure Database for MySQL using a Flexible Server in the Azure Portal. After configuring authentication and firewall settings, the server is connected via Azure Data Studio with the MySQL extension. A database (demodb) and a table (logdata) are created. Next, an external program from GitHub is configured with the log file path and server name, then executed to insert data into the table. The end goal is to automate log data entry into MySQL, ensuring efficient storage and retrieval of operational records for further analysis.

1. In this lab, we are going to Generate Data using the external Application inside the MySQL Database.
2. So, first we must move to the Azure Portal marketplace and search for MySQL and we will choose Azure Database for MySQL.

[Home](#) > [Create a resource](#) > [Marketplace](#) >

Azure Database for MySQL

Microsoft



The screenshot shows the Azure Marketplace listing for 'Azure Database for MySQL'. It features a blue icon with a white cylinder and the text 'Azure Database for MySQL' in bold. Below it, it says 'Microsoft | Azure Service' and has a rating of '★ 4.0 (293 ratings)'. A 'Plan' section includes a dropdown menu set to 'Azure Database for MySQL' and a 'Create' button. There is also a 'Add to Favorites' link.

3. We also need to choose our Database from the deployment options. So, we will choose a Flexible server. Click on advanced create.

Select Azure Database for MySQL deployment option

Microsoft

Feedback

Azure Database for MySQL - Single Server is scheduled for retirement by September 16, 2024. [Learn More](#)

How do you plan to use the service?



Flexible server

Best for production workloads that require zone resiliency, predictable performance, maximum control with IOPs scaling, custom maintenance window, cost optimization controls and simplified developer experience.

Quick Create

Advanced Create



Wordpress + MySQL Flexible server

Wordpress is state of the art publishing platform with a focus on aesthetics, web standards and usability. Use this template to create Wordpress on APP Service and Azure Database for MySQL Flexible Server in a Virtual network.

Create

[Learn More](#)

- Now we need to choose the resource group, give a name to our server then choose the region with the SQL version.

Subscription * ⓘ

MSDN Platforms Subscription

Resource group * ⓘ

NewRG

[Create new](#)

Server details

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

Server name * ⓘ

mysqlserver1231



Region * ⓘ

North Europe



MySQL version * ⓘ

8.0



- We will leave workload type and compute storage to default and choose the authentication as SQL authentication. Then give a username and password.

Workload type ⓘ

For small or medium size databases

Tier 1 Business Critical Workloads

For development or hobby projects

Compute + storage ⓘ

Burstable, B1ms

1 vCores, 2 GiB RAM, 20 GiB storage, Auto scale IOPS

Geo-redundancy : Disabled

[Configure server](#)

Availability zone ⓘ

No preference



Authentication method	<input checked="" type="radio"/> MySQL authentication only <input type="radio"/> Microsoft Entra authentication only <input type="radio"/> MySQL and Microsoft Entra authentication
Administrator login *	<input type="text" value="sqladmin"/> ✓
Password *	<input type="password"/> ✓
Confirm password *	<input type="password"/> ✓

6. In the networking tab just remember to add your IP address in the firewall rules.

Firewall rules

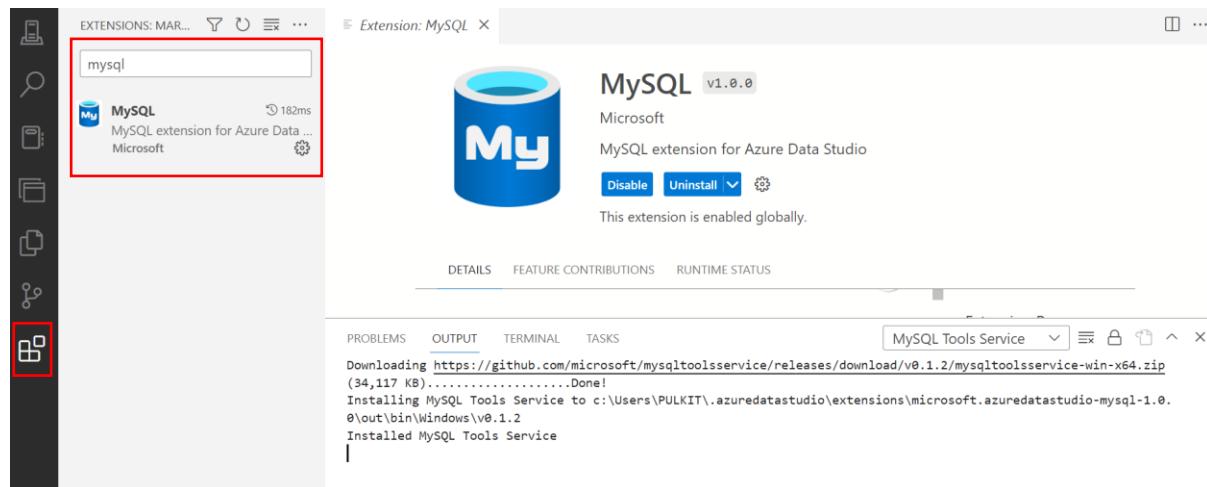
Inbound connections from the IP addresses specified below will be allowed to port 3306 on this server. [Learn more ↗](#)

Allow public access from any Azure service within Azure to this server ✓

+ Add current client IP address (103.226.203.243) + Add 0.0.0.0 - 255.255.255.255

Firewall rule name	Start IP address	End IP address	
ClientIPAddress_2025-3-5_9-41-10	103.226.203.243	103.226.203.243	Delete
<input type="text" value="Firewall rule name"/>	<input type="text" value="Start IP address"/>	<input type="text" value="End IP address"/>	

7. Once our server is up and running then we will connect our server using Azure Data Studio. So, open the Azure Data Studio.
 8. Then go to the extensions and search for MySQL then install it.



9. Now take the server's name of MySQL database and open Azure Data Studio.

mysqlserver1231

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Learning center

Resource visualizer

Settings

Power Platform

Security

Monitoring

Automation

Help

Subscription (move) : MSDN Platforms Subscription

Subscription ID : d6549a66-c45c-4979-840c-3b356da446b0

Resource group (move) : NewRG

Status : Ready

Location : North Europe

Server name : mysqlserver1231.mysql.database.azure.com

Administrator login : sqladmin

Configuration : Burstable_B1ms_1vCores_2GiB RAM_20 storage_360 IOPS

MySQL version : 8.0

Availability zone : 2

Created on : 2025-03-05 04:12:23.1676538 UTC

Add tags

Getting started Properties Recommendations Monitoring Tutorials

We've prepared a checklist to get you started

10. Here you need to choose MySQL in the connection type then paste the server's name, after that give the username and password then click on connect.

Connection Details

Connection type	MySQL
Server name *	mysqlserver1231.mysql.database.azure.com
Authentication type	Password
User name *	sqladmin
Password	*****
<input checked="" type="checkbox"/> Remember password	
Database name	<Default>
Server group	<Default>
Name (optional)	
<input style="background-color: #0078D4; color: white; padding: 5px 20px; border: none; border-radius: 5px; font-weight: bold; margin-right: 10px;" type="button" value="Connect"/> <input style="border: 1px solid #ccc; padding: 5px 20px; border-radius: 5px; font-weight: bold;" type="button" value="Cancel"/>	

11. Once the server is connected then we need to create a database in it and inside the database we will create a table.
12. Below you can see that first we created our database then we used it. After that we created an empty table.

CREATE DATABASE appdb;

USE demodb;

```

CREATE TABLE logdata
(
    Id int,
    Operationname varchar(200) NULL,
    Status varchar(100) NULL,
    Eventcategory varchar(100) NULL,
    Resourcetype varchar(1000) NULL,
    Resource varchar(2000) NULL
);

```

The screenshot shows a SQL query window in SQL Server Management Studio. The 'Database' dropdown menu is highlighted with a red box. The query itself creates a database named 'demodb' and a table 'logdata' with the specified schema. A 'Results' tab is visible at the bottom.

```

1 CREATE DATABASE demodb;
2
3 CREATE TABLE logdata
4 (
5     Id int,
6     Operationname varchar(200) NULL,
7     Status varchar(100) NULL,
8     Eventcategory varchar(100) NULL,
9     Resourcetype varchar(1000) NULL,
10    Resource varchar(2000) NULL
11 );
12
13 SELECT * FROM logdata

```

Id	Operationname	Status	Eventcategory	Resourcetype	Resource
----	---------------	--------	---------------	--------------	----------

13. Then open up the program you downloaded from GitHub and on line number 6 give the path of the log file you get with the program then on line number 10 you need to give the server name, save the changes and execute your program.

The screenshot shows a code editor with a C# file named 'sqlapp'. The code is for a MySQL database connection and parameter setup. It includes using statements for MySQL and System libraries, a connection string, and parameter definitions for Id, Operationname, Status, and Eventcategory.

```

sqlapp.cs
1  using MySql.Data.MySqlClient;
2  using System;
3  using System.Data.SqlClient;
4  using System.Reflection.Metadata;
5
6  string strFilePath = "C:\\tmp1\\Log.csv";
7  StreamReader? logReader = null;
8  int Id = 1;
9  MySqlConnection appdbConnection =
10    new MySqlConnection("Server=mysqlserver1231.mysql.database.azure.com;UserID = sqldadmin;Password=P@ssword@");
11
12  MySqlParameter paramId = new MySqlParameter();
13  paramId.ParameterName = "@Id";
14
15  MySqlParameter paramOperationname = new MySqlParameter();
16  paramOperationname.ParameterName = "@Operationname";
17
18  MySqlParameter paramStatus = new MySqlParameter();
19  paramStatus.ParameterName = "@Status";
20
21
22  MySqlParameter paramEventcategory = new MySqlParameter();
23  paramEventcategory.ParameterName = "@Eventcategory";

```

14. If you have done everything properly then you will see that records are being written onto the table and if you try to view the data inside the table then you will have information accordingly.

```
Written Record 1
Written Record 2
Written Record 3
Written Record 4
Written Record 5
Written Record 6
Written Record 7
Written Record 8
Written Record 9
Written Record 10
Written Record 11
Written Record 12
Written Record 13
Written Record 14
Written Record 15
Written Record 16
Written Record 17
Written Record 18
Written Record 19
Written Record 20
Written Record 21
Written Record 22
```

Run Cancel Disconnect Change Database: demodb

```
1 CREATE DATABASE demodb;
2
3 CREATE TABLE logdata
4 (
5     Id int,
6     Operationname varchar(200) NULL,
7     Status varchar(100) NULL,
8     Eventcategory varchar(100) NULL,
9     ResourceType varchar(1000) NULL,
10    Resource varchar(2000) NULL
11 );
12
13 SELECT * FROM logdata
```

Results Messages

	Id	Operationname	Status	Eventca
1	1	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	Update SQL database	Succ
2	2	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	Create Deployment	Star
3	3	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	Create Deployment	Accep
4	4	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	Registers the Microsoft SQL Database Resource Provider	Star
5	5	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	Registers the Microsoft SQL Database Resource Provider	Succ
6	6	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	Update SQL server	Star
7	7	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	'audit' Policy action.	Succ
8	8	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	'auditIfNotExists' Policy action.	Star

15. Then on GitHub you can find one more program and if you open up this program in Visual Studio change the server's name in Activity Services save it and then run the application.

The screenshot shows the Visual Studio IDE. On the left is the code editor with the file 'ActivityService.cs' open. The code defines a class 'ActivityService' with methods for getting a MySQL connection and listing activities from a database. On the right is the 'Solution Explorer' pane, which shows a project named 'sqlapp' containing files like 'ActivityService.cs', 'apppettings.json', and 'Program.cs'. A red box highlights the 'ActivityService.cs' file in the Solution Explorer.

```

ActivityService.cs  x
sqapp  sqapp.Services.ActivityService  GetConnection()
1  using MySql.Data.MySqlClient;
2  using sqapp.Models;
3
4
5  namespace sqapp.Services
6  {
7
8      2 references
9      public class ActivityService
10     {
11
12         1 reference
13         private MySqlConnection _getConnection()
14         {
15             return new MySqlConnection("Server=mysqlserver123.mysql.database.azure.com;UserID = sqladmin;Pass");
16
17         }
18
19         1 reference
20         public List<Activity> GetActivities()
21         {
22             List<Activity> _activity_lst = new List<Activity>();
23             string _statement = "SELECT Id,Operationname,Status,Eventcategory,Resourcetype,Resource from logde";
24             MySqlConnection _connection = _getConnection();
25
26             MySqlCommand _cmd = new MySqlCommand(_statement, _connection);
27             MySqlDataReader _reader = _cmd.ExecuteReader();
28
29             while (_reader.Read())
30             {
31                 Activity _activity = new Activity();
32                 _activity.Id = _reader.GetInt32("Id");
33                 _activity.Operationname = _reader.GetString("Operationname");
34                 _activity.Status = _reader.GetString("Status");
35                 _activity.Eventcategory = _reader.GetString("Eventcategory");
36                 _activity.Resourcetype = _reader.GetString("Resourcetype");
37                 _activity.Resource = _reader.GetString("Resource");
38
39                 _activity_lst.Add(_activity);
40             }
41
42             _reader.Close();
43             _cmd.Dispose();
44             _connection.Close();
45         }
46
47     }
48
49 }

```

16. You will see that this application is helping you to display the application onto the browser from your MySQL database.

The screenshot shows a web browser window with the URL 'localhost:7137'. The page title is 'Information in the Activity Log'. Below the title is a table with 14 rows of data, each representing an activity entry. The columns are: Id, Operation name, Status, Event category, Resource type, and Resource. The data includes various system-level operations like database updates, deployment creation, and policy changes.

Id	Operation name	Status	Event category	Resource type	Resource
1	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	Update SQL database	Succeeded	Administrative	Informational
2	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	Create Deployment	Started	Administrative	Informational
3	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	Create Deployment	Accepted	Administrative	Informational
4	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	Registers the Microsoft SQL Database Resource Provider	Started	Administrative	Informational
5	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	Registers the Microsoft SQL Database Resource Provider	Succeeded	Administrative	Informational
6	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	Update SQL server	Started	Administrative	Informational
7	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	'audit' Policy action.	Succeeded	Policy	Warning
8	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	'auditIfNotExists' Policy action.	Started	Policy	Informational
9	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	Update SQL server	Accepted	Administrative	Informational
10	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	Update SQL server firewall rules	Started	Administrative	Informational
11	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	Update SQL server firewall rules	Started	Administrative	Informational
12	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	Update Server Connection Policy Create	Started	Administrative	Informational
13	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	Update SQL database	Started	Administrative	Informational
14	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	Update Server Connection Policy Create	Succeeded	Administrative	Informational