ISSN: 2754-6705

Journal of Mathematical & Computer Applications

Review Article

Open Access

Composing Serverless Azure Functions in NodeJS with MySQL Database Integration

Bhargav Bachina

USA

ABSTRACT
This paper delves into the concept and implementation of serverless computing, particularly focusing on Azure Functions as a platform for deploying serverless applications. It introduces the notion of a Function app, a container that amalgamates functions into a single logical unit, thereby simplifying management, deployment, and resource sharing. Serverless computing is elucidated as Function as a Service (FaaS), a paradigm where business logic is executed in the form of functions without the need for manual infrastructure management. Azure Functions enables developers to host and execute business logic seamlessly, without the concerns of provisioning infrastructure. The platform supports a multitude of programming languages, including C#, Java, JavaScript, TypeScript, and Python, offering versatility in development approaches. The paper further explores the utilization of triggers, specifically HTTP triggers, for executing code in response to HTTP requests. It highlights the development process within Visual Studio Code, facilitated by the Azure Functions extension, which allows for the comprehensive creation and deployment of projects. Additionally, the distinction between Consumption service plans and App Service Plans is examined, along with the prerequisite of linking every function app to a storage account. Deployment methodologies, both locally and through the Azure portal, are outlined, along with the capabilities for monitoring logs via the Azure portal's Monitor section. This comprehensive overview provides insights into leveraging Azure Functions for efficient, scalable, and manageable serverless application development.

***Corresponding author**
Bhargav Bachina, USA.

Received: March 15, 2024; **Accepted:** March 21, 2024, **Published:** March 28, 2024

Keywords: Serverless, Azure, Cloud Computing, Programming, Software Development

Azure Functions represent an efficient method for executing code snippets in the cloud, liberating developers from the complexities of managing the infrastructure traditionally required for code hosting. This platform supports a diverse array of programming languages, including C#, Java, JavaScript, PowerShell, Python, and others as documented in the Azure Functions supported languages guide. A notable feature of Azure Functions is the consumption plan, which offers a cost-effective payment model based solely on the execution time of the code, ensuring that resources are automatically scaled according to user demand.

The practice of writing serverless code has gained widespread popularity, with Azure Functions leading the way by providing a flexible and economical approach to code execution. This serverless model allows developers to focus on their code, written in various programming languages such as Java, Python, NodeJS, and more, without worrying about the operational aspects of server management. This paper will specifically explore the development of NodeJS Azure Functions, integrating MySQL as the database backend. Through this exploration, we aim to demonstrate the versatility and efficiency of Azure Functions in supporting serverless application development, emphasizing the ease of deployment, scalability, and cost-effectiveness inherent in the serverless computing paradigm.

Prerequisites
You need to know a lot of things as prerequisites if you want to write serverless NodeJS REST API. First, you need to create two accounts: a GitHub account to store the source code and Microsoft Account to deploy that code using Function App Service. Let's create these accounts by following the below links. You can start both for free.
- GitHub Account
- Microsoft Azure Account

Since we are building the NodeJS REST API application you need to be familiar with nodejs, and javascript. You need to install NodeJS on your local machine.
- NodeJS
- Javascript
- Azure Data Studio
- MySQL

- Sequelize
- VSCode

NodeJS
As an asynchronous event-driven JavaScript runtime, Node.js is designed to build scalable network applications.

Sequelize
Sequelize provides various methods to assist in querying your database for data.

Azure Data Studio
Azure Data Studio offers additional experiences available as optional extensions. It's built for data professionals who use SQL Server and Azure databases on-premises or in multi-cloud environments.

MySQL Community Edition
Community Edition relational Database

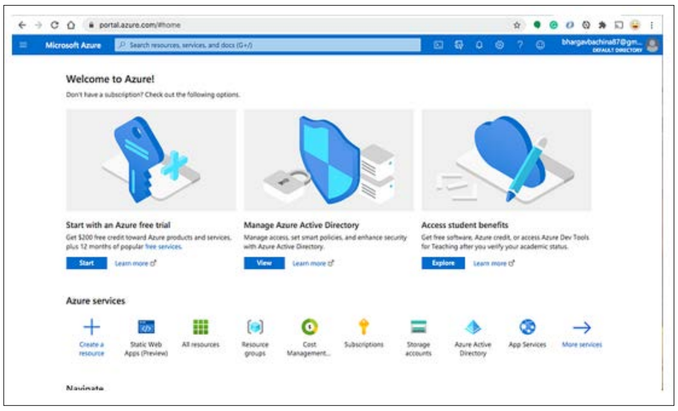
VSCode
The editor we are using for the project. It's open-source and you can download it here.

Postman
Manual testing of your APIs
If you are new to NodeJS and don't know how to build REST API with it, I would recommend going through the below article. We used the project from this article as a basis for this post.

- How to write production-ready Node.js Rest API — Javascript version
- How To Write Serverless NodeJS REST API With Azure Functions

Microsoft Azure Account
You should have a Microsoft Azure Account. You can get a free account for one year. You should see the below screen after you log in.

- Azure Account



Welcome to Azure!
Don't have a subscription? Check out the following options.

Start with an Azure free trial
Get $200 free credit toward Azure products and services, plus 12 months of popular free services.
Start | Learn more

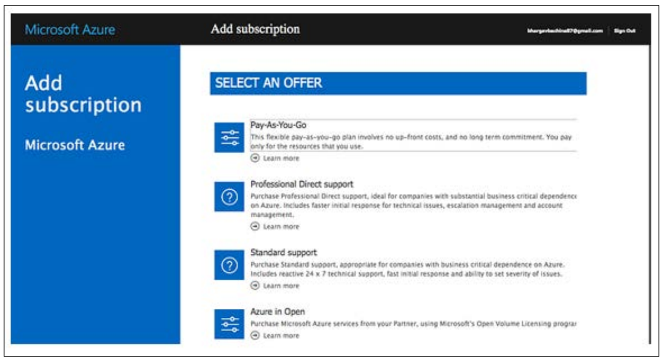
Manage Azure Active Directory
Manage access, set smart policies, and enhance security with Azure Active Directory.
View | Learn more

Access student benefits
Get free software, Azure credit, or access Azure Dev Tools for Teaching after you verify your academic status.
Explore | Learn more

Azure services

Azure Home Screen

You need to create a subscription for your account. The most common is the Pay as You Go subscription.



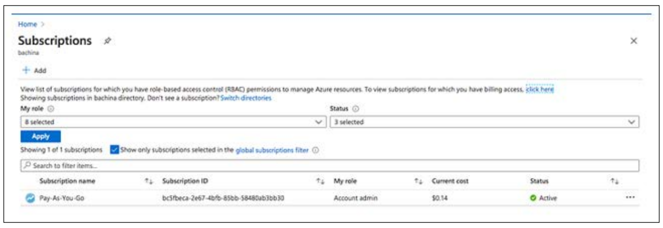
Pay-As-You-Go
This flexible pay-as-you-go plan involves no up-front costs, and no long term commitment. You pay only for the resources that you use.
Learn more

Professional Direct support
Purchase Professional Direct support, ideal for companies with substantial business critical dependence on Azure. Includes faster initial response for technical issues, escalation management and account management.
Learn more

Standard support
Purchase Standard support, appropriate for companies with business critical dependence on Azure. Includes reactive 24 x 7 technical support, fast initial response and ability to set severity of issues.
Learn more

Azure in Open
Purchase Microsoft Azure services from your Partner, using Microsoft's Open Volume Licensing prograr
Learn more

Subscription Offers



View list of subscriptions for which you have role-based access control (RBAC) permissions to manage Azure resources. To view subscriptions for which you have billing access, click here
Showing subscriptions in bachina directory. Don't see a subscription? Switch directories

My role: 8 selected
Status: 3 selected
Apply
Showing 1 of 1 subscriptions ✓ Show only subscriptions selected in the global subscriptions filter
Search to filter items...

Subscription name	Subscription ID	My role	Current cost	Status
Pay-As-You-Go	bc5fbeca-2e67-4bfb-85bb-58480ab3bb30	Account admin	$0.14	Active

Pay-As-You-Go Subscription

You need a subscription to be associated with your tenant so that all the cost is billed to this subscription. If you are new to Azure, please go through the below article on how to get started with Azure.

• How To Get Started with Azure

Example Project
Here is the Github link for the example project you can just clone and run on your machine.

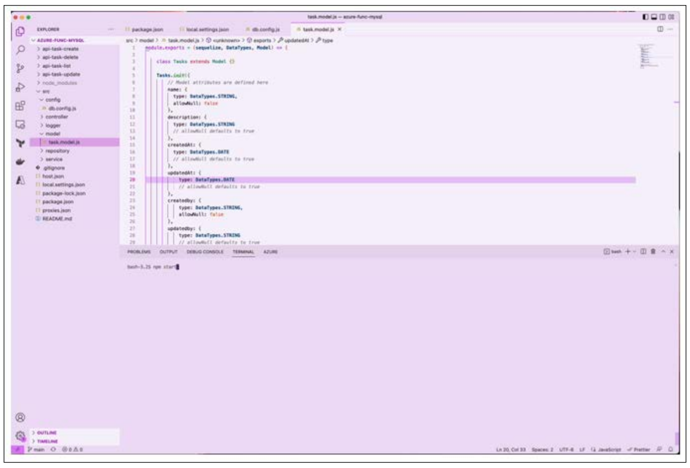
// clone the project
git clone https://github.com/bbachi/azure-func-mysql.git

There are four endpoints in this project. Let's look at one endpoint. Each endpoint has index.js and function.json files as below. The function.json has details such as route, method type, etc. The index.js file is the main file and starting file of the API and the controller is called from here.

• Files (https://gist.github.com/bbachi/0ffa11a2457301db6081c6994c28d95b#file-index-js)

We are using azure function-core tools for the development phase that speeds up your development. You just need to run this command after installing all dependencies.

```
// install dependencies
npm install

// start the server in development phase
npm start
```



```
module.exports = (sequelize, DataTypes, Model) => {

  class Tasks extends Model {}

  Tasks.init({
    // Model attributes are defined here
    name: {
      type: DataTypes.STRING,
      allowNull: false
    },
    description: {
      type: DataTypes.STRING
      // allowNull defaults to true
    },
    createdAt: {
      type: DataTypes.DATE
      // allowNull defaults to true
    },
    updatedAt: {
        type: DataTypes.DATE
        // allowNull defaults to true
    },
    createdby: {
        type: DataTypes.STRING,
        allowNull: false
    },
    updatedby: {
        type: DataTypes.STRING
        // allowNull defaults to true
```

```
bash-3.2$ npm start
```

Running the Application

Project Structure

Let's understand the project structure that we have here. The starting point of the application is folders as shown below and we have all the scripts, dependencies, etc in the package.json.

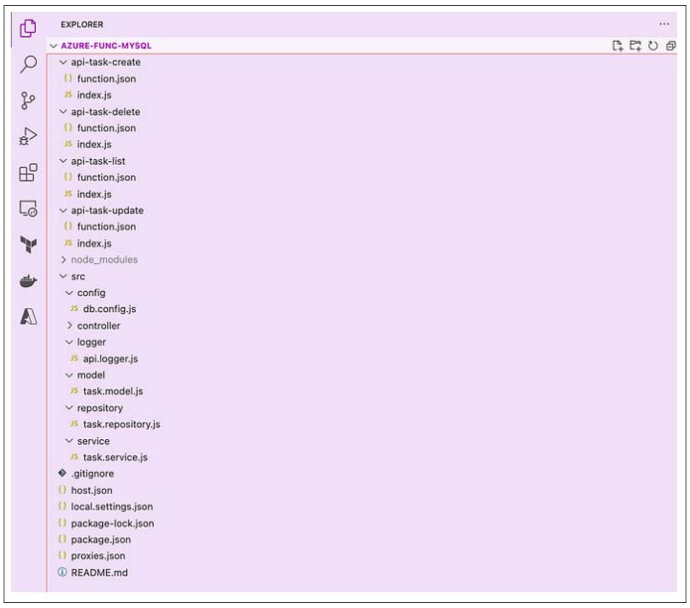
// endpoints/folders

api-tasks-delete

api-tasks-edit

api-tasks-list

api-tasks-save



```
AZURE-FUNC-MYSQL
  api-task-create
    function.json
    index.js
  api-task-delete
    function.json
    index.js
  api-task-list
    function.json
    index.js
  api-task-update
    function.json
    index.js
  node_modules
  src
    config
      db.config.js
    controller
    logger
      api.logger.js
    model
      task.model.js
    repository
      task.repository.js
    service
      task.service.js
  .gitignore
  host.json
  local.settings.json
  package-lock.json
  package.json
  proxies.json
  README.md
```

Project Structure

Each endpoint has index.ts and function.json files as below. The ***function.json*** has details such as route, method type, etc. The ***index.js*** file is the main file and starting file of the API and the controller is called from here.

• Files (https://gist.github.com/bbachi/0ffa11a2457301db6081c6994c28d95b#file-index-js)

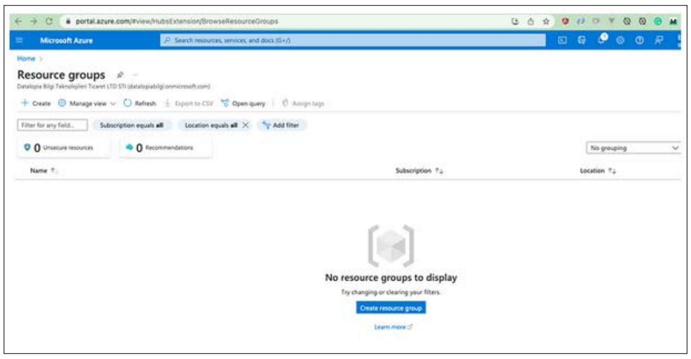
We have a controller, service, and repository in the src folder. All the logging-related configuration goes into api.logger.js under the folder logger. We have the local.settings.json file for all the environment-related configurations. We have a db.config.js file for all the database configurations.

Create MYSQL Server on Azure

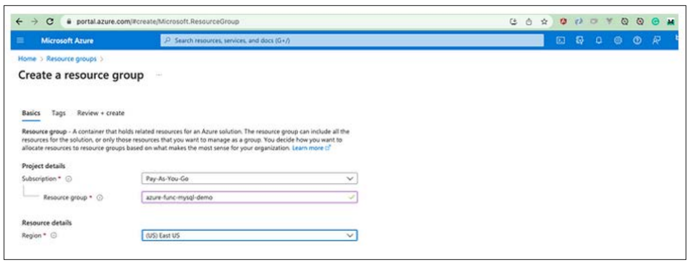
There are so many ways we can create and deploy MySQL servers; we will use Azure to create one for this post. Let's login into the Azure portal and create Azure Database for MySQL. You need an Azure Subscription and Account to create one. Please go through the below article on how to create those if you are new to Azure.

• How To Get Started with Azure

Let's Create a Resource Group for The Demo Project



Creating a resource group



portal.azure.com/#create/Microsoft.ResourceGroup

Create a resource group

Basics | Tags | Review + create

Resource group - A container that holds related resources for an Azure solution. The resource group can include all the resources for the solution, or only those resources that you want to manage as a group. You decide how you want to allocate resources to resource groups based on what makes the most sense for your organization. Learn more

Project details

Subscription *: Pay-As-You-Go

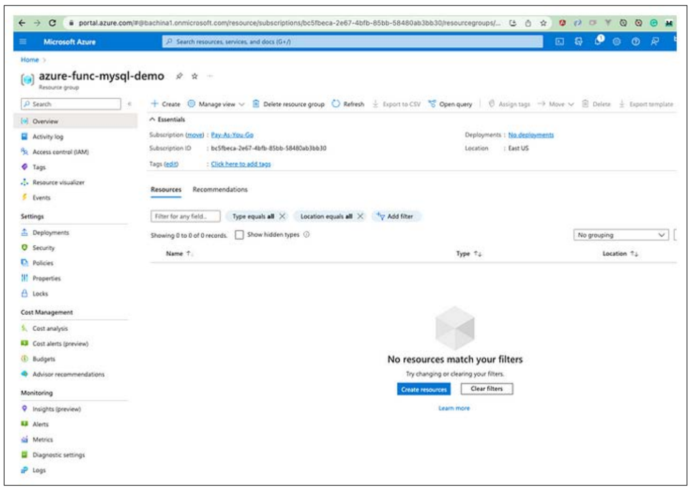
Resource group *: azure-func-mysql-demo

Resource details

Region *: (US) East US

Creating a resource group

Once it is done, you can see the overview page below.



Subscription (move) : Pay-As-You-Go

Subscription ID : bc5fbeca-2e67-4bfb-85bb-58480ab3bb30

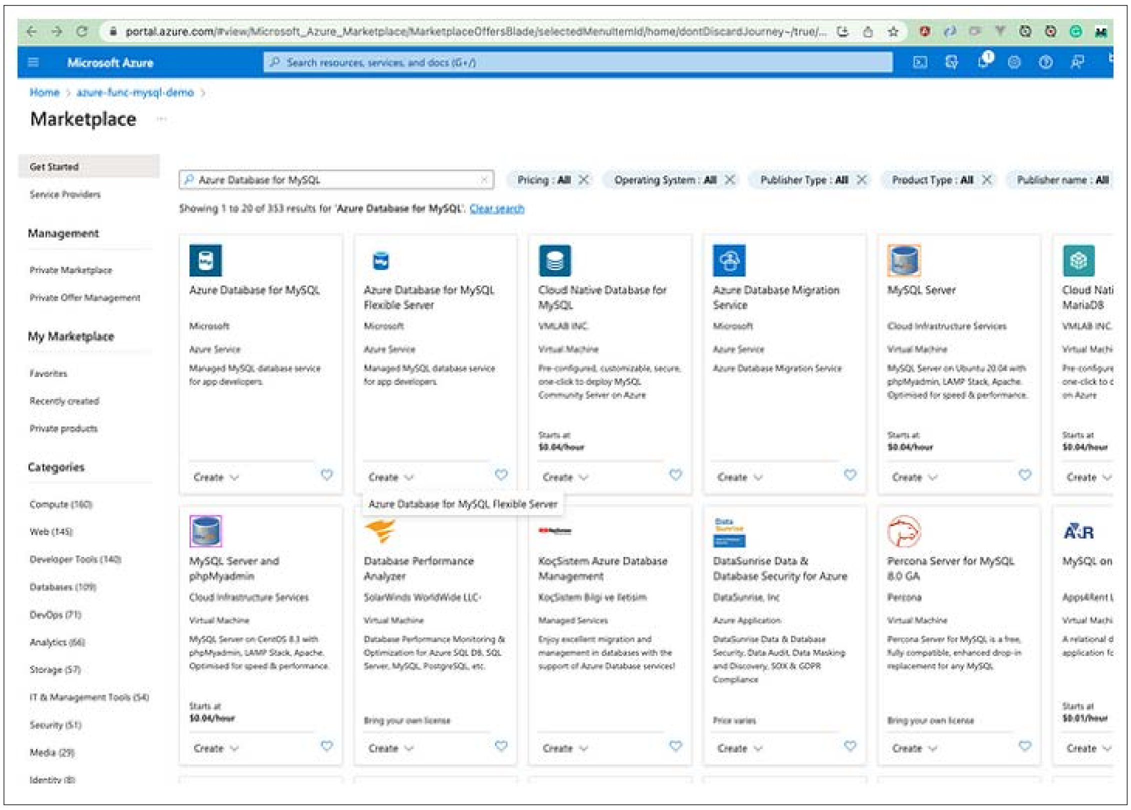
Tags (edit) : Click here to add tags

Deployments : No deployments

Location : East US

Resource Group

Let's create resources under this resource group. We can select from the marketplace below. Let's select Azure Database for MySQL Flexible Server.


portal.azure.com/#view/Microsoft_Azure_Marketplace/MarketplaceOffersBlade/selectedMenuItemId/home/dontDiscardJourney~/true/...

Microsoft Azure — Search resources, services, and docs (G+/)

Home > azure-func-mysql-demo >

Marketplace

Get Started
Service Providers
Management
Private Marketplace
Private Offer Management
My Marketplace
Favorites
Recently created
Private products
Categories
Compute (160)
Web (145)
Developer Tools (140)
Databases (109)
DevOps (71)
Analytics (66)
Storage (57)
IT & Management Tools (54)
Security (51)
Media (29)
Identity (8)

Search: Azure Database for MySQL — Pricing : All — Operating System : All — Publisher Type : All — Product Type : All — Publisher name : All

Showing 1 to 20 of 353 results for 'Azure Database for MySQL'. Clear search

Azure Database for MySQL — Microsoft — Azure Service — Managed MySQL database service for app developers.

Azure Database for MySQL Flexible Server — Microsoft — Azure Service — Managed MySQL database service for app developers.

Cloud Native Database for MySQL — VMLAB INC. — Virtual Machine — Pre-configured, customizable, secure, one-click to deploy MySQL Community Server on Azure — Starts at $0.04/hour

Azure Database Migration Service — Microsoft — Azure Service — Azure Database Migration Service

MySQL Server — Cloud Infrastructure Services — Virtual Machine — MySQL Server on Ubuntu 20.04 with phpMyadmin, LAMP Stack, Apache. Optimised for speed & performance. — Starts at $0.04/hour

Cloud Nati MariaDB — VMLAB INC. — Virtual Machi — Pre-configure one-click to d on Azure — Starts at $0.04/hour

Azure Database for MySQL Flexible Server

MySQL Server and phpMyadmin — Cloud Infrastructure Services — Virtual Machine — MySQL Server on CentOS 8.3 with phpMyadmin, LAMP Stack, Apache. Optimised for speed & performance. — Starts at $0.04/hour

Database Performance Analyzer — SolarWinds WorldWide LLC· — Virtual Machine — Database Performance Monitoring & Optimization for Azure SQL DB, SQL Server, MySQL, PostgreSQL, etc. — Bring your own license

KoçSistem Azure Database Management — KoçSistem Bilgi ve İletisim — Managed Services — Enjoy excellent migration and management in databases with the support of Azure Database services!

DataSunrise Data & Database Security for Azure — DataSunrise, Inc — Azure Application — DataSunrise Data & Database Security, Data Audit, Data Masking and Discovery, SOX & GDPR Compliance — Price varies

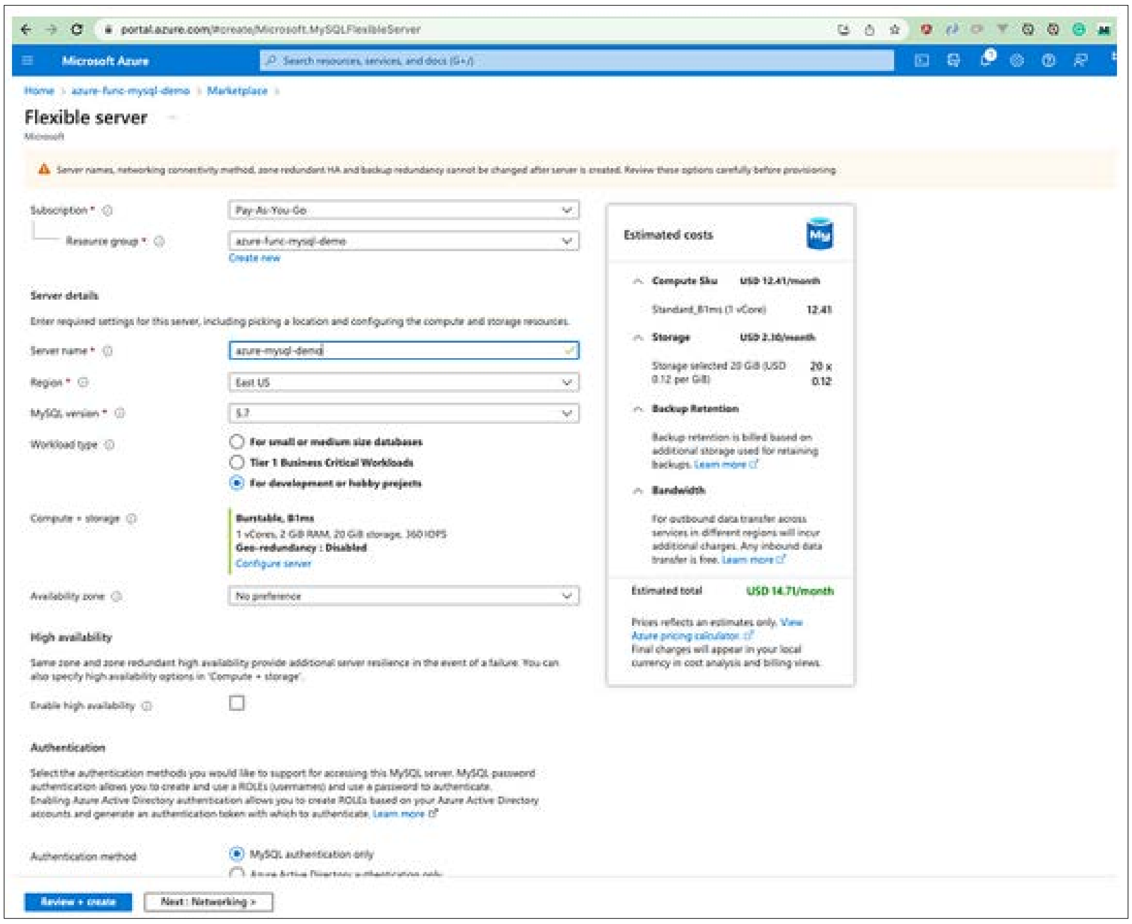
Percona Server for MySQL 8.0 GA — Percona — Virtual Machine — Percona Server for MySQL is a free, fully compatible, enhanced drop-in replacement for any MySQL. — Bring your own license

MySQL on — Apps4Rent L — Virtual Machi — A relational d application fc — Starts at $0.01/hour

Create

Azure Database for MySQL Flexible Server

You can give a unique name on the creation page.


portal.azure.com/#create/Microsoft.MySQLFlexibleServer

Microsoft Azure — Search resources, services, and docs (G+/)

Home > azure-func-mysql-demo > Marketplace >

Flexible server
Microsoft

Server names, networking connectivity method, zone redundant HA and backup redundancy cannot be changed after server is created. Review these options carefully before provisioning.

Subscription * : Pay-As-You-Go
Resource group * : azure-func-mysql-demo
Create new

Server details

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

Server name * : azure-mysql-demo
Region * : East US
MySQL version * : 5.7
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, 360 IOPS **Geo-redundancy : Disabled** Configure server
Availability zone : No preference

High availability

Same zone and zone redundant high availability provide additional server resilience in the event of a failure. You can also specify high availability options in 'Compute + storage'.

Enable high availability : ☐

Authentication

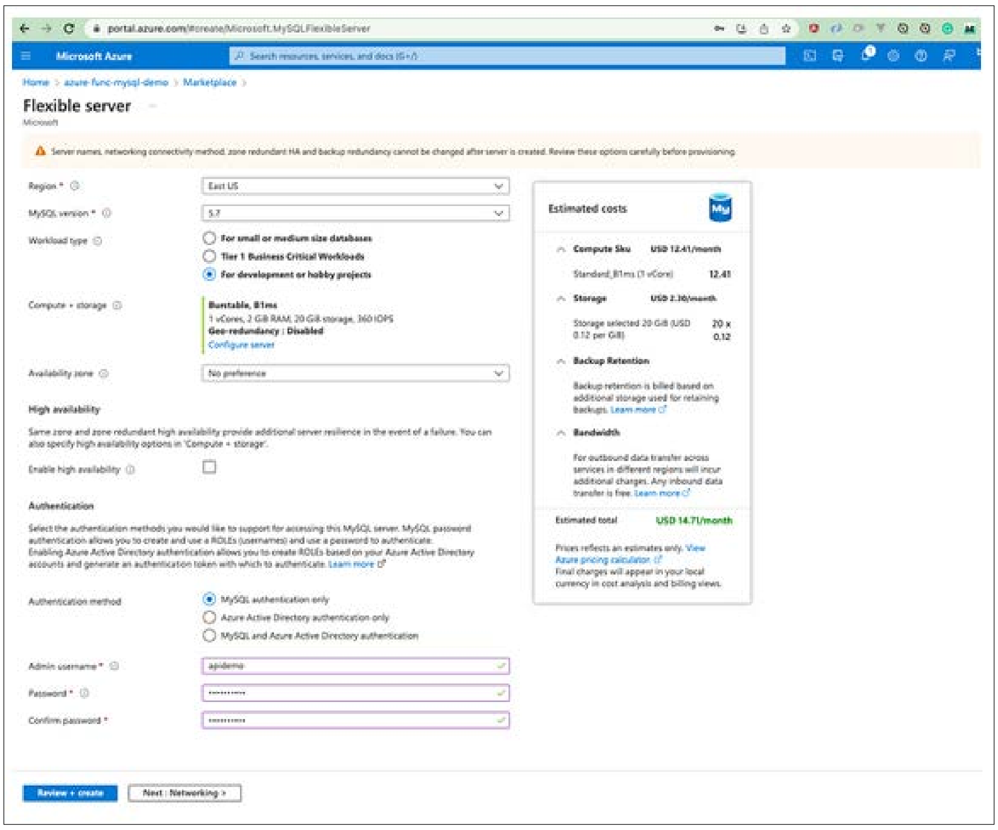
Select the authentication methods you would like to support for accessing this MySQL server. MySQL password authentication allows you to create and use a ROLEs (usernames) and use a password to authenticate. Enabling Azure Active Directory authentication allows you to create ROLEs based on your Azure Active Directory accounts and generate an authentication token with which to authenticate. Learn more

Authentication method : [✓ MySQL authentication only] / Azure Active Directory authentication only

Estimated costs

Compute Sku — USD 12.41/month
Standard_B1ms (1 vCore) — 12.41
Storage — USD 2.30/month
Storage selected 20 GiB (USD 0.12 per GiB) — 20 x 0.12
Backup Retention
Backup retention is billed based on additional storage used for retaining backups. Learn more
Bandwidth
For outbound data transfer across services in different regions will incur additional charges. Any inbound data transfer is free. Learn more
Estimated total — USD 14.71/month
Prices reflects an estimates only. View Azure pricing calculator
Final changes will appear in your local currency in cost analysis and billing views.

Flexible Server
You must select the Authentication method. I am selecting the MySQL authentication-only option and gave it an admin username and password.



portal.azure.com/#create/Microsoft.MySQLFlexibleServer

Microsoft Azure | Search resources, services, and docs (G+/)

Home > azure-func-mysql-demo > Marketplace >

Flexible server
Microsoft

Server names, networking connectivity method, zone redundant HA and backup redundancy cannot be changed after server is created. Review these options carefully before provisioning.

Region *: East US
MySQL version *: 5.7
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, 360 IOPS
Geo-redundancy : Disabled
Configure server
Availability zone: No preference

High availability
Same zone and zone redundant high availability provide additional server resilience in the event of a failure. You can also specify high availability options in 'Compute + storage'.
Enable high availability: ☐

Authentication
Select the authentication methods you would like to support for accessing this MySQL server. MySQL password authentication allows you to create and use a ROLEs (usernames) and use a password to authenticate. Enabling Azure Active Directory authentication allows you to create ROLEs based on your Azure Active Directory accounts and generate an authentication token with which to authenticate. Learn more

Authentication method: [✓ MySQL authentication only] / Azure Active Directory authentication only / MySQL and Azure Active Directory authentication
Admin username *: apidemo
Password *: ••••••••••••
Confirm password *: ••••••••••••

Review + create | Next : Networking >

Estimated costs
Compute Sku	**USD 12.41/month**
Standard_B1ms (1 vCore)	12.41
Storage	**USD 2.30/month**
Storage selected 20 GiB (USD 0.12 per GiB)	20 x 0.12

Backup Retention
Backup retention is billed based on additional storage used for retaining backups. Learn more

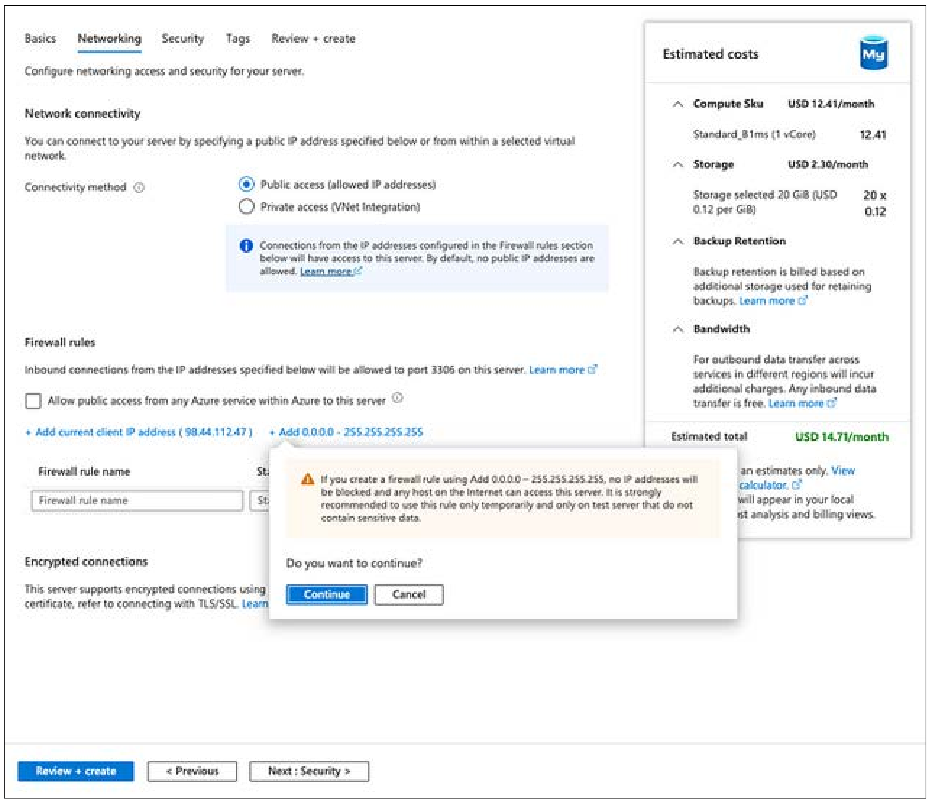
Bandwidth
For outbound data transfer across services in different regions will incur additional charges. Any inbound data transfer is free. Learn more

Estimated total: USD 14.71/month

Prices reflects an estimates only. View Azure pricing calculator.
Final charges will appear in your local currency in cost analysis and billing views.

Authentication Method

We can enable public access for this demo on the networking tab.



Basics | Networking | Security | Tags | Review + create

Configure networking access and security for your server.

Network connectivity
You can connect to your server by specifying a public IP address specified below or from within a selected virtual network.

Connectivity method: [✓ Public access (allowed IP addresses)] / Private access (VNet Integration)

Connections from the IP addresses configured in the Firewall rules section below will have access to this server. By default, no public IP addresses are allowed. Learn more

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 (98.44.112.47) + Add 0.0.0.0 - 255.255.255.255

Firewall rule name | St
Firewall rule name | St

Encrypted connections
This server supports encrypted connections using certificate, refer to connecting with TLS/SSL. Learn

If you create a firewall rule using Add 0.0.0.0 – 255.255.255.255, no IP addresses will be blocked and any host on the Internet can access this server. It is strongly recommended to use this rule only temporarily and only on test server that do not contain sensitive data.

Do you want to continue?

Continue | Cancel

Review + create | < Previous | Next : Security >

Estimated costs
Compute Sku	**USD 12.41/month**
Standard_B1ms (1 vCore)	12.41
Storage	**USD 2.30/month**
Storage selected 20 GiB (USD 0.12 per GiB)	20 x 0.12

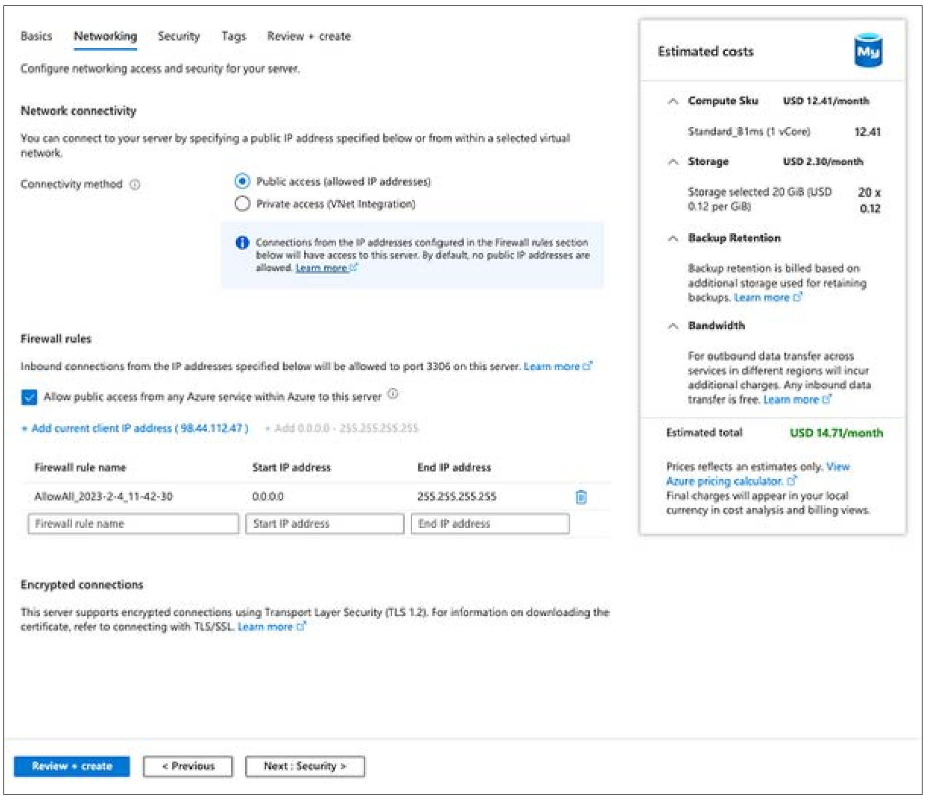
Backup Retention
Backup retention is billed based on additional storage used for retaining backups. Learn more

Bandwidth
For outbound data transfer across services in different regions will incur additional charges. Any inbound data transfer is free. Learn more

Estimated total: USD 14.71/month

an estimates only. View calculator.
will appear in your local st analysis and billing views.

Allowing Public Access



Configure networking access and security for your server.

Network connectivity

You can connect to your server by specifying a public IP address specified below or from within a selected virtual network.

Connectivity method: [✓ Public access (allowed IP addresses)] Private access (VNet Integration)

Connections from the IP addresses configured in the Firewall rules section below will have access to this server. By default, no public IP addresses are allowed. Learn more

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 (98.44.112.47)   + Add 0.0.0.0 - 255.255.255.255

Firewall rule name	Start IP address	End IP address
AllowAll_2023-2-4_11-42-30	0.0.0.0	255.255.255.255
Firewall rule name	Start IP address	End IP address

Encrypted connections

This server supports encrypted connections using Transport Layer Security (TLS 1.2). For information on downloading the certificate, refer to connecting with TLS/SSL. Learn more

Estimated costs

Compute Sku	USD 12.41/month
Standard_B1ms (1 vCore)	12.41
Storage	USD 2.30/month
Storage selected 20 GiB (USD 0.12 per GiB)	20 x 0.12

Backup Retention

Backup retention is billed based on additional storage used for retaining backups. Learn more

Bandwidth

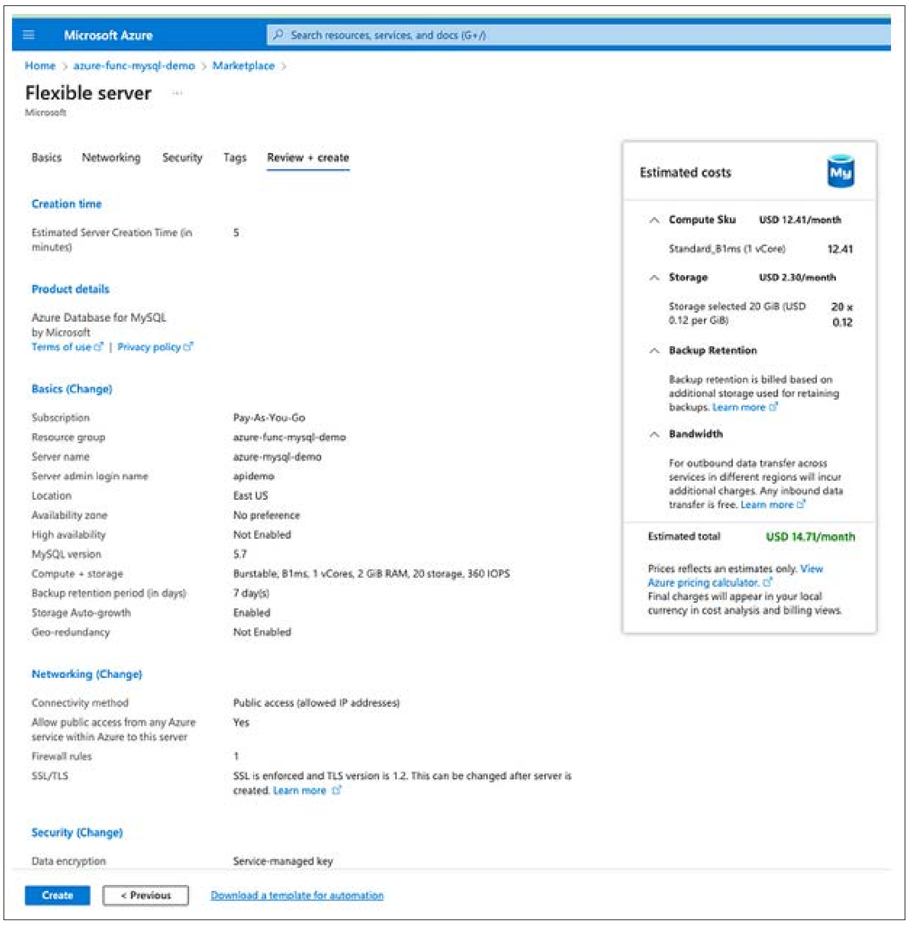
For outbound data transfer across services in different regions will incur additional charges. Any inbound data transfer is free. Learn more

Estimated total: USD 14.71/month

Prices reflects an estimates only. View Azure pricing calculator. Final charges will appear in your local currency in cost analysis and billing views.

Public Access

Let’s go to the review page and create.



Microsoft Azure
Search resources, services, and docs (G+/)
Home > azure-func-mysql-demo > Marketplace >

Flexible server

Microsoft

Basics
Networking
Security
Tags
Review + create

Creation time

Estimated Server Creation Time (in minutes)	5

Product details

Azure Database for MySQL by Microsoft
Terms of use | Privacy policy

Basics (Change)

Subscription	Pay-As-You-Go
Resource group	azure-func-mysql-demo
Server name	azure-mysql-demo
Server admin login name	apidemo
Location	East US
Availability zone	No preference
High availability	Not Enabled
MySQL version	5.7
Compute + storage	Burstable, B1ms, 1 vCores, 2 GiB RAM, 20 storage, 360 IOPS
Backup retention period (in days)	7 day(s)
Storage Auto-growth	Enabled
Geo-redundancy	Not Enabled

Networking (Change)

Connectivity method	Public access (allowed IP addresses)
Allow public access from any Azure service within Azure to this server	Yes
Firewall rules	1
SSL/TLS	SSL is enforced and TLS version is 1.2. This can be changed after server is created. Learn more

Security (Change)

Data encryption	Service-managed key

Estimated costs

Compute Sku	USD 12.41/month
Standard_B1ms (1 vCore)	12.41
Storage	USD 2.30/month
Storage selected 20 GiB (USD 0.12 per GiB)	20 x 0.12

Backup Retention

Backup retention is billed based on additional storage used for retaining backups. Learn more

Bandwidth

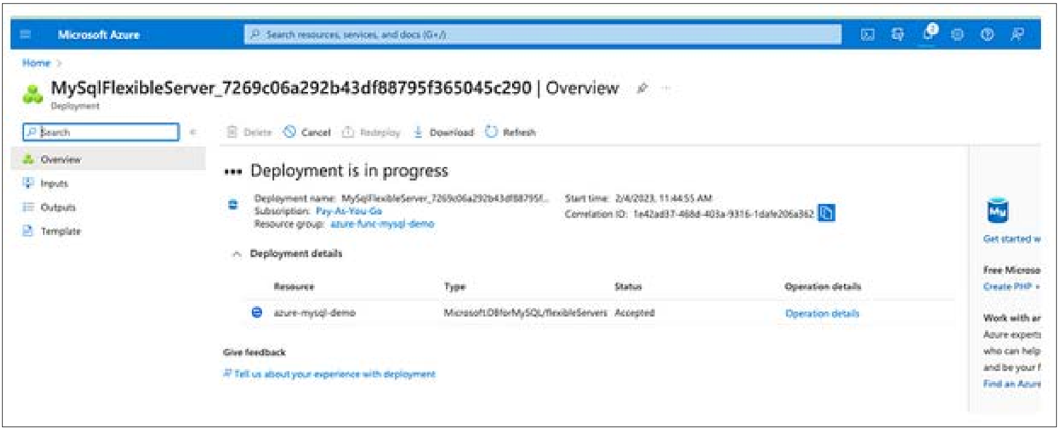
For outbound data transfer across services in different regions will incur additional charges. Any inbound data transfer is free. Learn more

Estimated total: USD 14.71/month

Prices reflects an estimates only. View Azure pricing calculator. Final charges will appear in your local currency in cost analysis and billing views.

Creating Server

It takes some time to create the server.


MySqlFlexibleServer_7269c06a292b43df88795f365045c290 | Overview
Deployment

Deployment is in progress

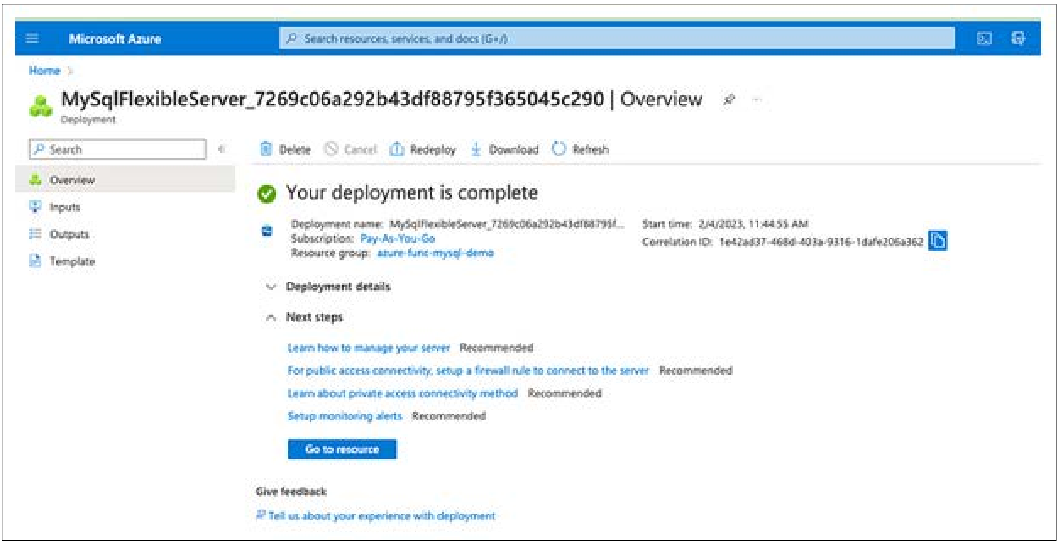
Deployment name: MySqlFlexibleServer_7269c06a292b43df88795f...
Subscription: Pay-As-You-Go
Resource group: azure-func-mysql-demo
Start time: 2/4/2023, 11:44:55 AM
Correlation ID: 1e42ad37-468d-403a-9316-1dafe206a362

Deployment details

Resource	Type	Status	Operation details
azure-mysql-demo	Microsoft.DBforMySQL/flexibleServers	Accepted	Operation details

Give feedback
Tell us about your experience with deployment

Deployment in Progress


MySqlFlexibleServer_7269c06a292b43df88795f365045c290 | Overview
Deployment

Your deployment is complete

Deployment name: MySqlFlexibleServer_7269c06a292b43df88795f...
Subscription: Pay-As-You-Go
Resource group: azure-func-mysql-demo
Start time: 2/4/2023, 11:44:55 AM
Correlation ID: 1e42ad37-468d-403a-9316-1dafe206a362

Deployment details

Next steps

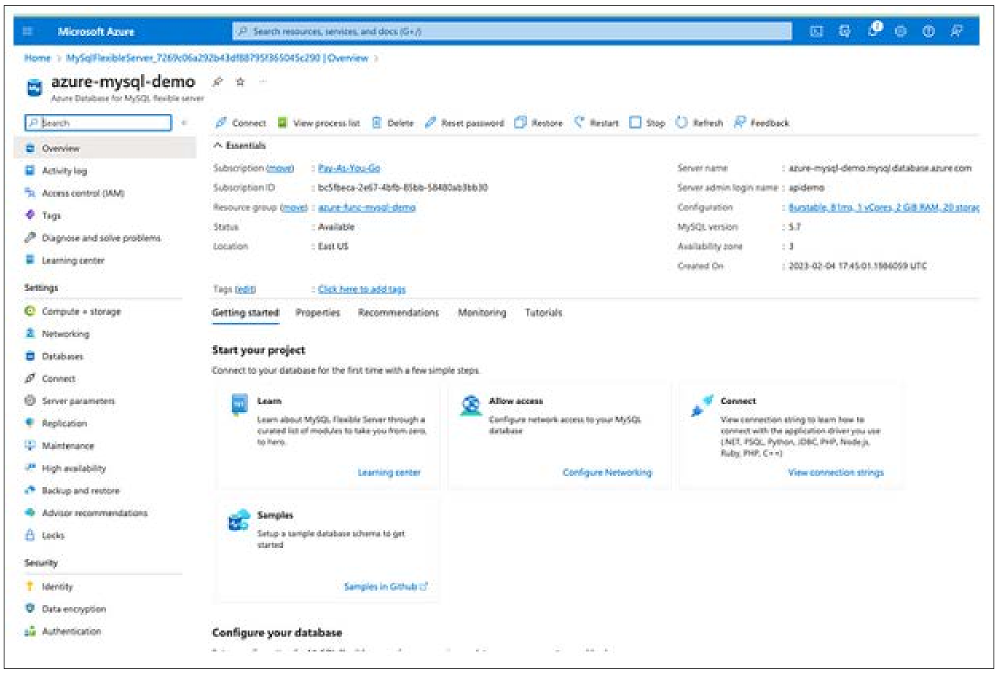
Learn how to manage your server Recommended
For public access connectivity, setup a firewall rule to connect to the server Recommended
Learn about private access connectivity method Recommended
Setup monitoring alerts Recommended

Go to resource

Give feedback
Tell us about your experience with deployment

Deployment Completed

You can go to the overview page after the deployment.


Home > MySqlFlexibleServer_7269c06a292b43df88795f365045c290 | Overview >

azure-mysql-demo
Azure Database for MySQL flexible server

Essentials

Subscription (move)	: Pay-As-You-Go	Server name	: azure-mysql-demo.mysql.database.azure.com
Subscription ID	: bc5fbeca-2e67-4bfb-85bb-58480ab3bb30	Server admin login name	: apidemo
Resource group (move)	: azure-func-mysql-demo	Configuration	: Burstable, B1ms, 1 vCores, 2 GiB RAM, 20 storag
Status	: Available	MySQL version	: 5.7
Location	: East US	Availability zone	: 3
		Created On	: 2023-02-04 17:45:01.1986059 UTC

Tags (edit) : Click here to add tags

Start your project
Connect to your database for the first time with a few simple steps.

Learn
Learn about MySQL Flexible Server through a curated list of modules to take you from zero to hero.
Learning center

Allow access
Configure network access to your MySQL database
Configure Networking

Connect
View connection string to learn how to connect with the application driver you use (.NET, PSQL, Python, JDBC, PHP, Node.js, Ruby, PHP, C++)
View connection strings

Samples
Setup a sample database schema to get started
Samples in Github

Configure your database

MySQL Server

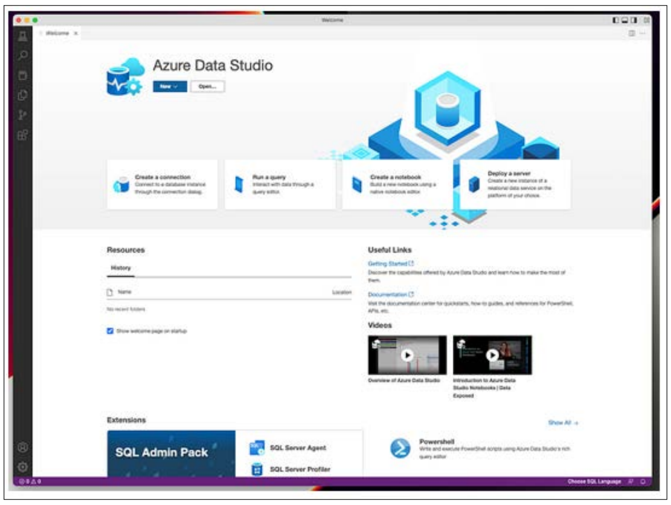
Install Azure Data Studio
Azure Data Studio is a cross-platform tool that can be used to connect databases from several database engines. You can write queries and execute them. There are so many extensions you can install based on your requirement. You can learn about that in the below link.

• What is Azure Data Studio (https://learn.microsoft.com/en-us/azure-data-studio/what-is-azure-data-studio)

Here is the download link you can download the Data Studio based on your OS.

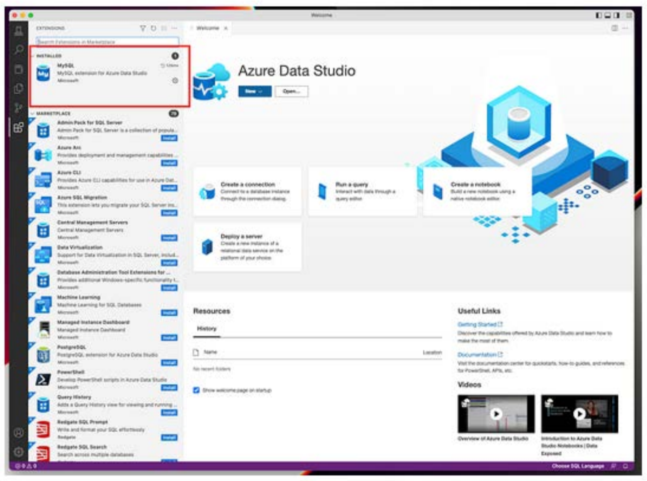
• Download Azure Data Studio (https://learn.microsoft.com/en-us/azure-data-studio/download-azure-data-studio)

It looks like this when it is launched.



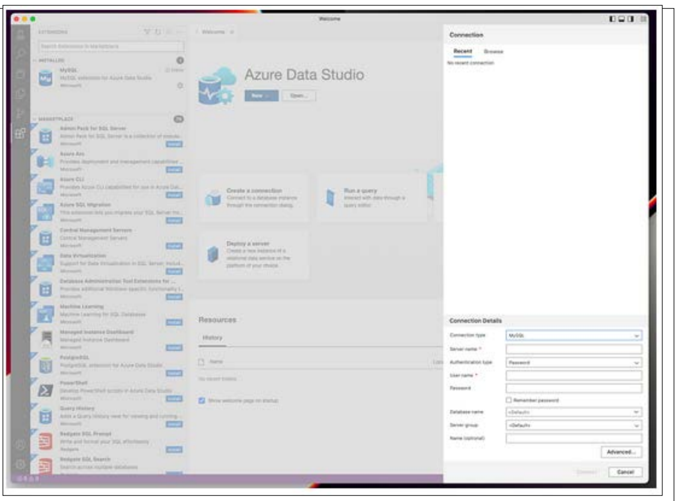
Azure Data Studio

Connect MYSQL through Azure Data Studio
Since we are connecting to the MySQL database, we need to have that extension installed below.



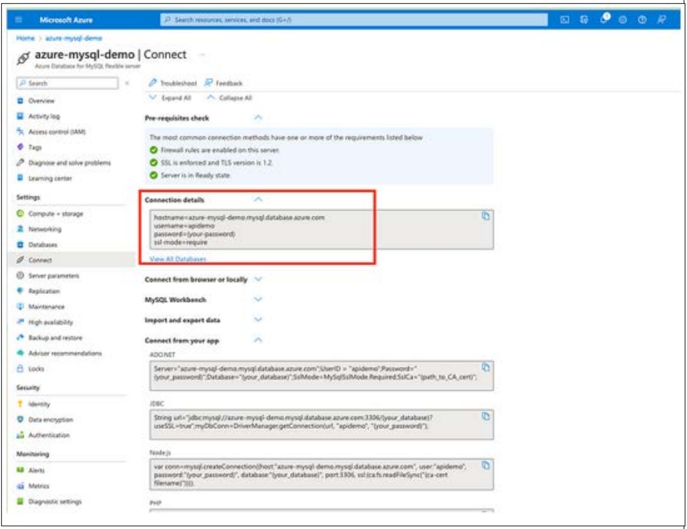
MySQL extension

Once the extension is installed, we can click the new button to connect to the server.



Connecting To the MySQL Database

Let's get the MySQL Connection details from the Azure portal below.



Pre-requisites check

The most common connection methods have one or more of the requirements listed below
- Firewall rules are enabled on this server.
- SSL is enforced and TLS version is 1.2.
- Server is in Ready state.

Connection details

```
hostname=azure-mysql-demo.mysql.database.azure.com
username=apidemo
password={your-password}
ssl-mode=require
```

View All Databases

ADO.NET

```
Server="azure-mysql-demo.mysql.database.azure.com";UserID = "apidemo";Password="
{your_password}";Database="{your_database}";SslMode=MySqlSslMode.Required;SslCa="{path_to_CA_cert}";
```

JDBC

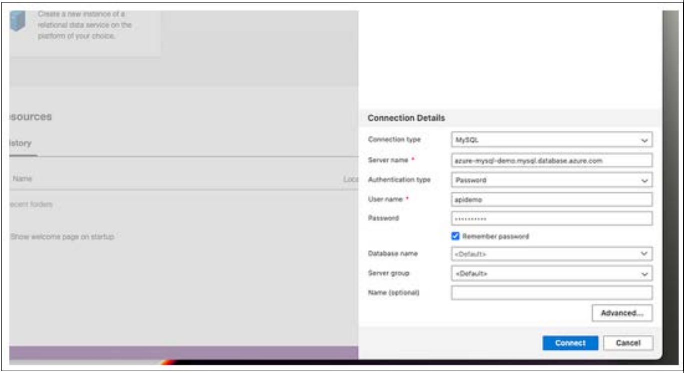
```
String url="jdbc:mysql://azure-mysql-demo.mysql.database.azure.com:3306/{your_database}?
useSSL=true";myDbConn=DriverManager.getConnection(url, "apidemo", "{your_password}");
```

Node.js

```
var conn=mysql.createConnection({host:"azure-mysql-demo.mysql.database.azure.com", user:"apidemo",
password:"{your_password}", database:"{your_database}", port:3306, ssl:{ca:fs.readFileSync("{ca-cert
filename}")}});
```

PHP

MySQL Connection Details

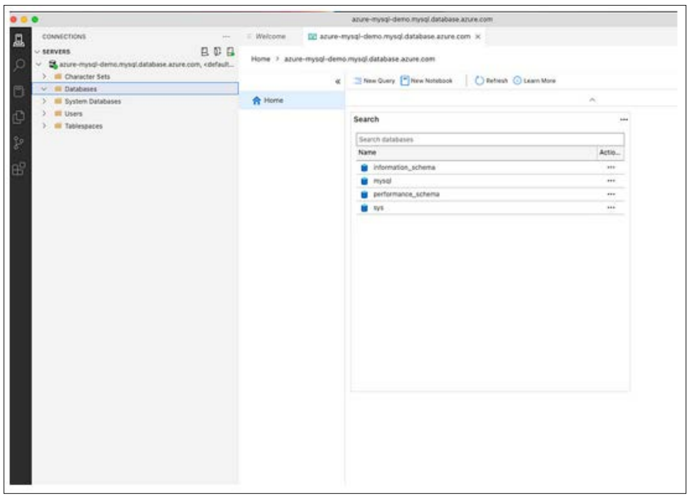


Connection Details

Connection type	MySQL
Server name *	azure-mysql-demo.mysql.database.azure.com
Authentication type	Password
User name *	apidemo
Password	••••••••••
	✓ Remember password
Database name	<Default>
Server group	<Default>
Name (optional)	

SQL Connection

You can see all the details once connected.

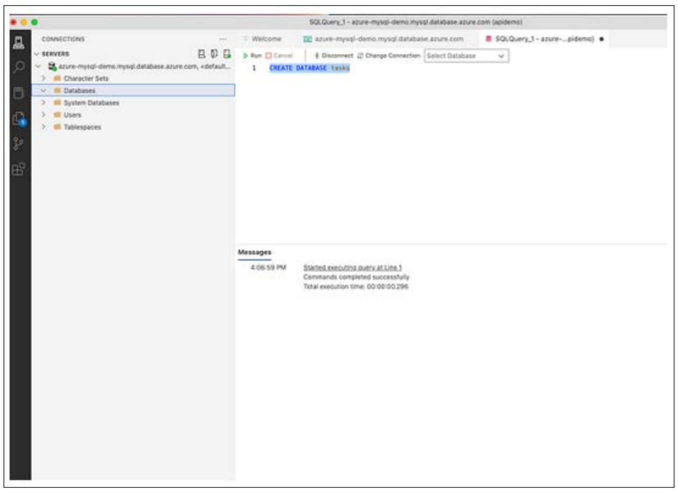


```
information_schema
mysql
performance_schema
sys
```

MySQL Connected

Create a Database and Table
Let's create a Database by clicking on the Query Tool below.

Create Database Tasks

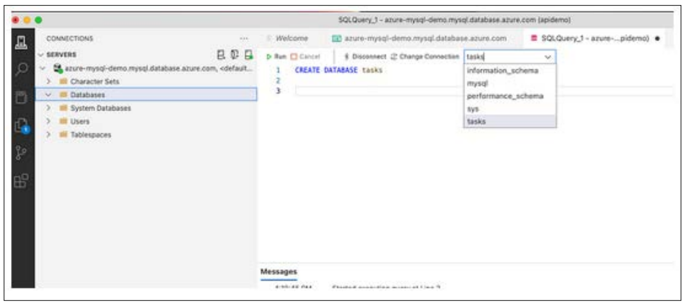


```sql
CREATE DATABASE tasks
```

```
Messages
4:06:59 PM    Started executing query at Line 1
              Commands completed successfully
              Total execution time: 00:00:00.296
```

Database Creation

You can see the Database created in the dropdown below.

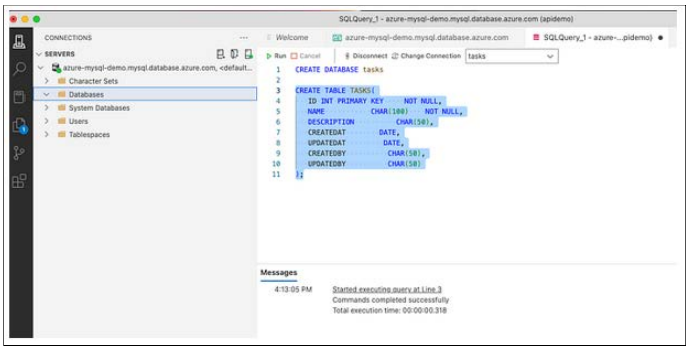


```
tasks
information_schema
mysql
performance_schema
sys
tasks
```

Database Created

Let's run the following query to create the database table.

• Query SQL (https://gist.github.com/bbachi/7aa313f0ec81e0fe73d08c48758b632d#file-database-sql)



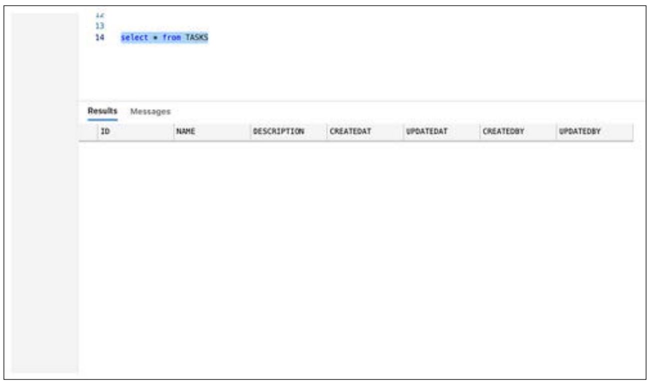
```sql
CREATE DATABASE tasks

CREATE TABLE TASKS(
   ID INT PRIMARY KEY      NOT NULL,
   NAME           CHAR(100)    NOT NULL,
   DESCRIPTION           CHAR(50),
   CREATEDAT         DATE,
   UPDATEDAT          DATE,
   CREATEDBY           CHAR(50),
   UPDATEDBY           CHAR(50)
);
```

```
Messages
4:13:05 PM    Started executing query at Line 3
              Commands completed successfully
              Total execution time: 00:00:00.318
```

Executing the query

Let's run the following query to verify whether the table is created or not.

Select * from tasks

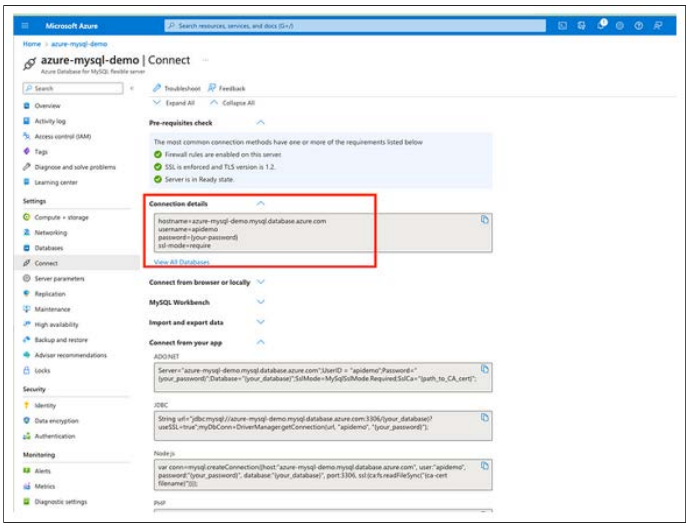


```sql
select * from TASKS
```

ID	NAME	DESCRIPTION	CREATEDAT	UPDATEDAT	CREATEDBY	UPDATEDBY

Table Created

Configure MYSQL in Azure Functions
Let's configure MySQL from our application. The first thing we need to do is to get the connection string or connection details. You can get it from the properties in the Azure Portal.



azure-mysql-demo | Connect

Pre-requisites check

The most common connection methods have one or more of the requirements listed below
- Firewall rules are enabled on this server
- SSL is enforced and TLS version is 1.2.
- Server is in Ready state.

Connection details

```
hostname=azure-mysql-demo.mysql.database.azure.com
username=apidemo
password={your-password}
ssl-mode=require
```

Connect from your app

ADO.NET

```
Server="azure-mysql-demo.mysql.database.azure.com";UserID = "apidemo";Password="{your_password}";Database="{your_database}";SslMode=MySqlSslMode.Required;SslCa="{path_to_CA_cert}";
```

JDBC

```
String url="jdbc:mysql://azure-mysql-demo.mysql.database.azure.com:3306/{your_database}?useSSL=true";myDbConn=DriverManager.getConnection(url, "apidemo", "{your_password}");
```

Node.js

```
var conn=mysql.createConnection({host:"azure-mysql-demo.mysql.database.azure.com", user:"apidemo", password:"{your_password}", database:"{your_database}", port:3306, ssl:{ca:fs.readFileSync("{ca-cert filename}")}});
```

MySQL Connection Details

The next thing is to install mysql2 with the following command.

```
// install client and sequelize
npm install mysql2
npm install sequelize
```

Let's place the connection string and database name in the application properties file as below. You have to URL encode the password if you have any special characters in the password.

You need to define the DB environment variables in the file called local.settings.json as below. Storing configuration in the environment separate from code is based on The Twelve-Factor App methodology.

• local.settings.json
(https://gist.github.com/bbachi/d8e09e95063e7b7f243df1e18743438b#file-local-settings-json)

Let's define the configuration class where it creates a connection with the connection details from the properties. We are using mysql2 to connect with MySQL for all the queries. This client makes it easy for you to interact with MySQL. Sequelize is a promise-based NodeJS ORM tool for many relational databases such as Postgres, MYSQL, etc [1-4].

• db.config.js
(https://gist.github.com/bbachi/2c186a005ef08db1a86170c2cfc52b94#file-db-config-js)

We need to define a model for our collection as below. We need to define the schema for the collection and then you need to pass that schema to the model and export it as a module.

• task.model.js
(https://gist.github.com/bbachi/1baecb0a5bf81bfe78aa7a5dff751f72#file-task-model-js)

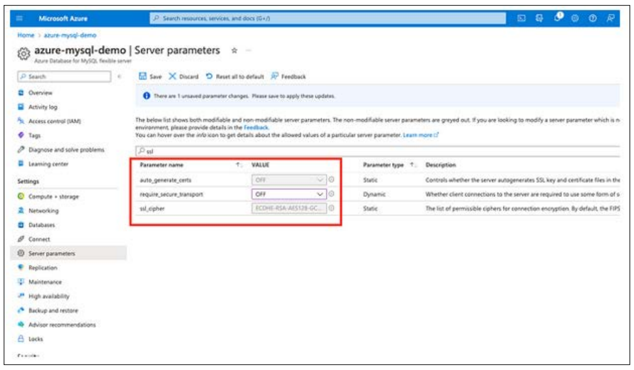
Finally, here is the repository class which uses the above model for all the CRUD operations.

• task.repository.js
(https://gist.github.com/bbachi/e721c43fab8207fbeb94acfdb0b55749#file-task-repository-js)

We have this service file in between the controller and repository for any data manipulation if needed.

• task.service.js
(https://gist.github.com/bbachi/8d11f6d04235d4f43f2e89f7b1aac7ab#file-task-service-js)

Since we are doing this locally. We must disable the SSL for connecting. We should set the parameter require_secure_transport OFF under the section called Server Parameters.



There are 1 unsaved parameter changes. Please save to apply these updates.

The below list shows both modifiable and non-modifiable server parameters. The non-modifiable server parameters are greyed out. If you are looking to modify a server parameter which is n
environment, please provide details in the Feedback.
You can hover over the info icon to get details about the allowed values of a particular server parameter. Learn more

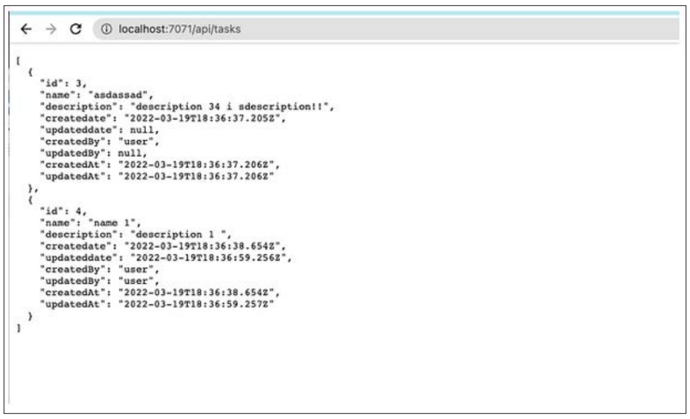
ssl

Parameter name	VALUE	Parameter type	Description
auto_generate_certs	OFF	Static	Controls whether the server autogenerates SSL key and certificate files in the
require_secure_transport	OFF	Dynamic	Whether client connections to the server are required to use some form of s
ssl_cipher	ECDHE-RSA-AES128-GC...	Static	The list of permissible ciphers for connection encryption. By default, the FIPS

Disabling SSL

With all the above files in place, we can hit the following URL.

http://localhost:7071/api/tasks



```
[
  {
    "id": 3,
    "name": "asdassad",
    "description": "description 34 i sdescription!!",
    "createdate": "2022-03-19T18:36:37.205Z",
    "updateddate": null,
    "createdBy": "user",
    "updatedBy": null,
    "createdAt": "2022-03-19T18:36:37.206Z",
    "updatedAt": "2022-03-19T18:36:37.206Z"
  },
  {
    "id": 4,
    "name": "name 1",
    "description": "description 1 ",
    "createdate": "2022-03-19T18:36:38.654Z",
    "updateddate": "2022-03-19T18:36:59.256Z",
    "createdBy": "user",
    "updatedBy": "user",
    "createdAt": "2022-03-19T18:36:38.654Z",
    "updatedAt": "2022-03-19T18:36:59.257Z"
  }
]
```

API GET URL

Summary

- The container that groups functions into a logical unit for easier management, deployment and sharing of resources is called Function app.
- Serverless computing can be thought of as a function as a service (FaaS), or a microservice that is hosted on a cloud platform. Your business logic runs as functions, and you don't have to manually provision or scale infrastructure.
- Azure Functions is a serverless application platform. It allows developers to host business logic that can be executed without provisioning infrastructure.
- You can write Azure functions in several languages such as C#, java, javascript, typescript, Python, etc.
- You need to use this trigger when you want to execute the code in response to a request sent through the HTTP protocol.
- You need to install the Azure Functions extension. Once you install the Azure Functions extension for Visual Studio Code you can create the entire project from VSCode itself.
- Function apps may use one of two types of service plans. The first service plan is the Consumption service plan, and another is the App Service Plan.
- Every function app should be linked with a storage account. When you create a function app, it must be linked to a storage account.
- You need an Azure Functions extension for Visual Studio Code to develop the entire thing locally.
- When you deploy it to the function app you can see it in the Azure portal.
- To deploy locally, you can click on the up-arrow icon in the Azure Functions extension in the VSCode.
- You can monitor logs of the API by accessing the Monitor section of each function on the Azure portal.

Conclusion

In conclusion, Azure Functions emerges as a pivotal solution in the realm of serverless computing, offering a streamlined and efficient approach for developers to deploy and manage their business logic in the cloud. By abstracting away the complexities associated with infrastructure management, Azure Functions enables the execution of code in a variety of programming languages, such as C#, Java, JavaScript, TypeScript, and Python, without the overhead of provisioning and scaling resources. The utilization of Function apps facilitates the grouping of functions into logical units, simplifying management, deployment, and resource sharing.

Furthermore, the platform's flexibility is showcased through its support for multiple triggering options, including HTTP requests, and its accommodation of diverse development environments via extensions for Visual Studio Code. The choice between Consumption and App Service plans offers tailored solutions to meet different operational and financial requirements, underscored by the necessity of linking function apps to a storage account for optimal functionality. Deployment capabilities, both locally and through the Azure portal, coupled with comprehensive monitoring tools, underscore the efficacy and convenience of Azure Functions in the serverless landscape. This exploration of Azure Functions underscores its significance as a robust platform that not only simplifies the development process but also aligns with the evolving needs of modern cloud-based applications. overhead of provisioning and scaling resources. The utilization of Function apps facilitates the grouping of functions into logical units, simplifying management, deployment, and resource sharing.

Furthermore, the platform's flexibility is showcased through its support for multiple triggering options, including HTTP requests, and its accommodation of diverse development environments via extensions for Visual Studio Code. The choice between Consumption and App Service plans offers tailored solutions to meet different operational and financial requirements, underscored by the necessity of linking function apps to a storage account for optimal functionality. Deployment capabilities, both locally and through the Azure portal, coupled with comprehensive monitoring tools, underscore the efficacy and convenience of Azure Functions in the serverless landscape. This exploration of Azure Functions underscores its significance as a robust platform that not only simplifies the development process but also aligns with the evolving needs of modern cloud-based applications.

References

1. Azure Functions Documentation https://azure.microsoft.com/en-us/free/.
2. NodeJS Documentation https://nodejs.org/en.
3. Azure Cloud https://azure.microsoft.com/en-us.
4. MySQL Documentation https://www.mysql.com/.