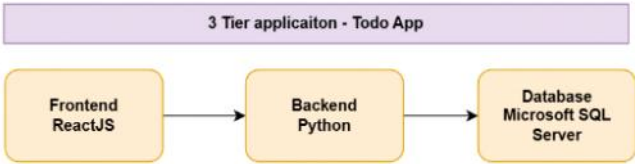


Todo 3 Tier Monolithic-Application Deployment on Azure Cloud

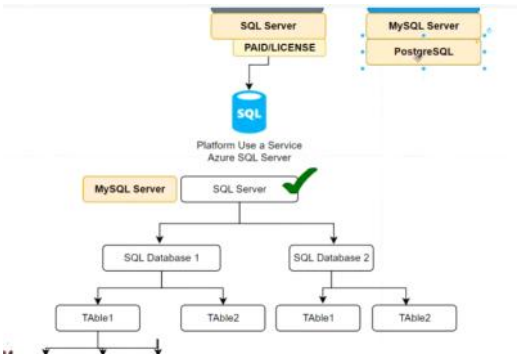
01 June 2025 10:00

Create The Resource Group In which we are going to Deploy our 3 Tier Application.



1. Database Set Up.

We will set up SQL Database using **Azure** in the Resource Group that we can as Azure SQL database to insert the coming Load from the Application UI. (To avoid whole setup that we are required to setup SQL Database on the local machine and Management, Backup, License.)



Create SQL Database Server.

Microsoft Azure

Search

[Home](#) > [SQL servers](#) >

Create SQL Database Server

Microsoft

⚠

Changing Basic options may reset selections you have made. Review all options prior to creating the resource.

Subscription *

Azure subscription 1

Resource group *

rg-application-deployment

[Create new](#)

Server details

Enter required settings for this server, including providing a name and location.

Server name *

application-database-server

.database.windows.net

Location *

(US) Central US

Authentication

ℹ

Azure Active Directory (Azure AD) is now Microsoft Entra ID. [Learn more](#)

Select your preferred authentication methods for accessing this server. Create a server admin login and password to access your server with SQL authentication, select only Microsoft Entra authentication [Learn more](#) using an existing Microsoft Entra user, group, or application as Microsoft Entra admin [Learn more](#), or select both SQL and Microsoft Entra authentication.

Authentication method

☐ Use Microsoft Entra-only authentication

☐ Use both SQL and Microsoft Entra authentication

☒ Use SQL authentication

Server admin login *

database-mainuser

Password *

Review + create

Next : Networking >

[Home](#) > [SQL servers](#) >

Create SQL Database Server

Microsoft

Basics

Networking

Security

Additional settings

Tags

Review + create

Configure networking access for your server.

Firewall rules

Allow Azure services and resources to access this server

Yes

No

Create SQL Database.

Create SQL Database

Microsoft

⚠ Changing Basic options may reset selections you have made. Review all options prior to creating the resource.

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

Subscription

Resource group

Database details

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

Database name

Server

Compute + storage

Behavior when free offer limit reached

- Behavior when free offer limit reached ☒ Auto-pause the database until next month
When free offer limit is reached, the database will not be accessible until the beginning of next calendar month when free amount is renewed. There will be no additional charges.
- ☐ Continue using database for additional charges
Database continues to be accessible after free offer limit is reached. Additional usage beyond the free offer amount for that month will be charged at general purpose serverless tier rates. The free amount will be renewed at the beginning of the next calendar month.

Validating...

Next : Networking >

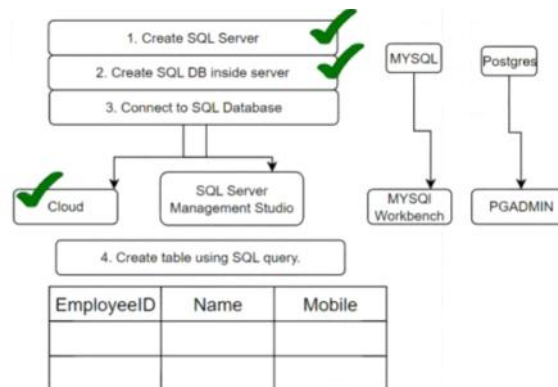
Overage billing ¹	Disabled
ESTIMATED STORAGE COST / MONTH	0.00 USD
COMPUTE COST / VCORE SECOND ²	0.000000 USD

¹ There will be no charges for usage within the free limits. The database will be paused automatically when the free limits are reached.

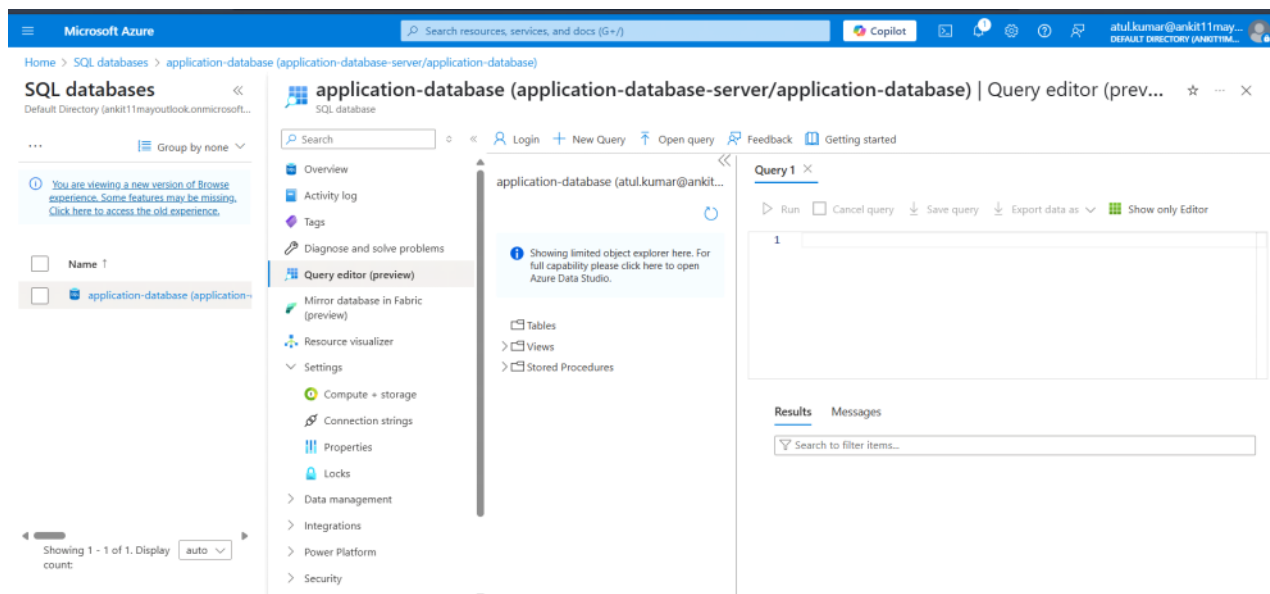
² Serverless databases are billed in vCore seconds based on a combination of CPU and memory utilization. [Learn more about serverless billing](#)

SQL Database is created.

You Can Access the Database from Cloud and MySQL Workbench, PGADMIN(Postgres Admin), SSMS(SQL Server Management Studio).

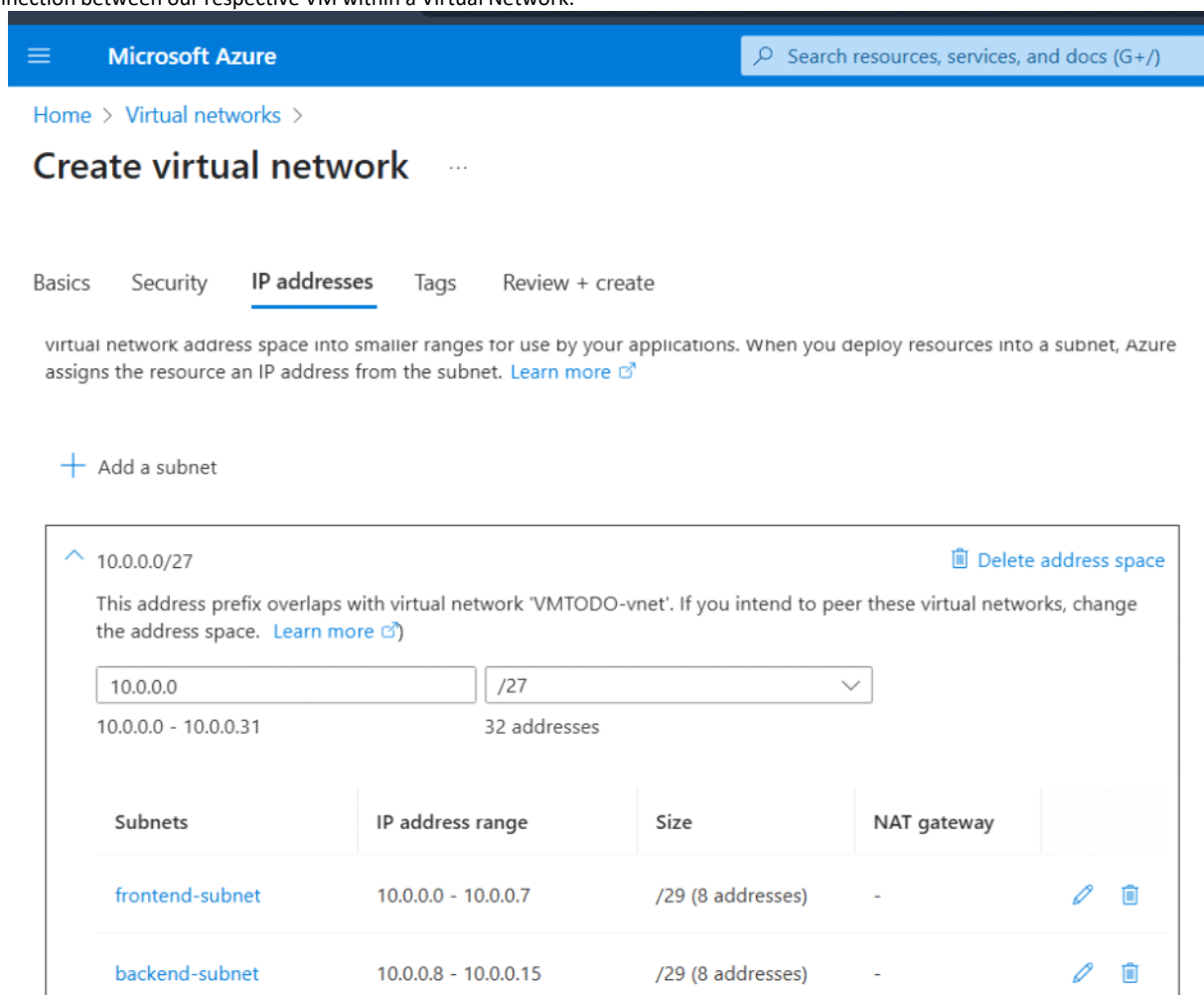


When you are accessing the Azure SQL Database from Cloud.
Then Allow IP to the AllowList to access the database and for querying it.
SQL Database is now Online and ready for use.



Now To Set Up Backend Application and Frontend Application.

Create Virtual Network and Define the Subnet For both Frontend and Backend within the same virtual Network to establish a smooth connection between our respective VM within a Virtual Network.



2. Backend Set Up

Follow the Readme.md to Check for the specific image need to be pull to set up infrastructure for Backend.

Prerequisites

Before getting started, make sure you have the following prerequisites installed on your system:

- `source_image_reference = { publisher = "Canonical" offer = "0001-com-ubuntu-server-focal" sku = "20_04-lts" version = "latest" }`
- Python
- pip

Create Backend VM

Microsoft Azure

Search resources, services, and docs

Home > Compute infrastructure | Virtual machines >

Create a virtual machine

Help me create a low cost VM

Help me create a VM optimized for high availability

Help me choose the right VM size for my workload

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

Subscription *

Azure subscription 1

Resource group *

rg-application-deployment

Create new

Instance details

Virtual machine name *

Backend-VM

Region *

(US) Central US

Availability options

No infrastructure redundancy required

Security type

Trusted launch virtual machines

Configure security features

Image *

Ubuntu Server 20.04 LTS - x64 Gen2 (free services eligible)

See all images | Configure VM generation

VM architecture

Arm64

☒ x64

Run with Azure Spot discount

☐

You are in the free trial period. Costs associated with this VM can be covered by any remaining credits on your subscription. [Learn more](#)

Size *

Standard_D2s_v3 - 2 vcpus, 8 GiB memory (\$80.30/month)

See all sizes

Administrator account

Authentication type

SSH public key

☒ Password

Username *

backendvm

Password *

Confirm password *

Inbound port rules

Select which virtual machine network ports are accessible from the public internet. You can specify more limited or granular network access on the Networking tab.

Public inbound ports *

None

☒ Allow selected ports

Select inbound ports *

SSH (22)

This will allow all IP addresses to access your virtual machine. This is only recommended for testing. Use the Advanced controls in the Networking tab to create rules to limit inbound traffic to known IP addresses.

Azure Page 5

Home > Compute infrastructure | Virtual machines >

Create a virtual machine

Help me create a low cost VM | Help me create a VM optimized for high availability | Help me choose the right VM size for my workload

Basics | Disks | **Networking** | Management | Monitoring | Advanced | Tags | Review + create

Define network connectivity for your virtual machine by configuring network interface card (NIC) settings. You can control ports, inbound and outbound connectivity with security group rules, or place behind an existing load balancing solution. [Learn more](#)

Network interface

When creating a virtual machine, a network interface will be created for you.

Virtual network * [Create new](#)

Subnet * [Manage subnet configuration](#)

Public IP [Create new](#)

NIC network security group ☐ None ☒ Basic ☐ Advanced

Public inbound ports * ☐ None ☒ Allow selected ports

Select inbound ports *

⚠ This will allow all IP addresses to access your virtual machine. This is only recommended for testing. Use the Advanced controls in the Networking tab to create rules to limit inbound traffic to known IP addresses.

Delete public IP and NIC when VM is deleted ☐

Backend VM Created:

The screenshot shows the Azure portal interface for a virtual machine named 'Backend-VM'. The left sidebar contains navigation options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Resource visualizer, Connect, Bastion, Networking, Network settings, Load balancing, Application security groups, Network manager, Settings, and Availability + scale. The main content area displays the 'Overview' tab for the virtual machine. Key details include: Resource group (rg-application-deployment), Status (Running), Location (Central US), Subscription (Azure subscription 1), Subscription ID (18b5931-6bb5-431b-a964-40d9c2211486), Operating system (Linux (Ubuntu 20.04)), Size (Standard D2s v3 (2 vcpus, 8 GB memory)), Public IP address (52.238.212.112), Virtual network/subnet (application-vnet/backend-subnet), DNS name (Not configured), Health state (OK), and Time created (6/1/2025, 6:36 AM UTC). Below this, the 'Properties' tab shows details for the virtual machine, including Computer name (Backend-VM), Operating system (Linux (Ubuntu 20.04)), VM generation (V2), VM architecture (x64), and Agent status (Ready). The 'Networking' tab shows the Public IP address (52.238.212.112), Private IP address (10.0.0.12), and Virtual network/subnet (application-vnet/backend-subnet).

Connect to Virtual Machine using SSH

```
backendvm@Backend-VM: ~
Microsoft Windows [Version 10.0.26100.4061]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ATUL KUMAR>ssh backendvm@52.238.212.112
The authenticity of host '52.238.212.112 (52.238.212.112)' can't be established.
ED25519 key fingerprint is SHA256:/KXGmEgPkB+Js7cjHlNyE3TZATbd4FupzhEMAwTz75M.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '52.238.212.112' (ED25519) to the list of known hosts.
backendvm@52.238.212.112's password:
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.15.0-1089-azure x86_64)
```

Follow the requirement for backend application
Install PIP, Python

sudo apt install pip (It comes with Python3)

```
backendvm@Backend-VM:~$ python3 --version
Python 3.8.10
backendvm@Backend-VM:~$ pip --version
pip 20.0.2 from /usr/lib/python3/dist-packages/pip (python 3.8)
backendvm@Backend-VM:~$
```

Clone the repository.

Step 1: Clone the Repository

Clone the application's source code from your version control system or download it as a zip archive and extract it to your local machine.

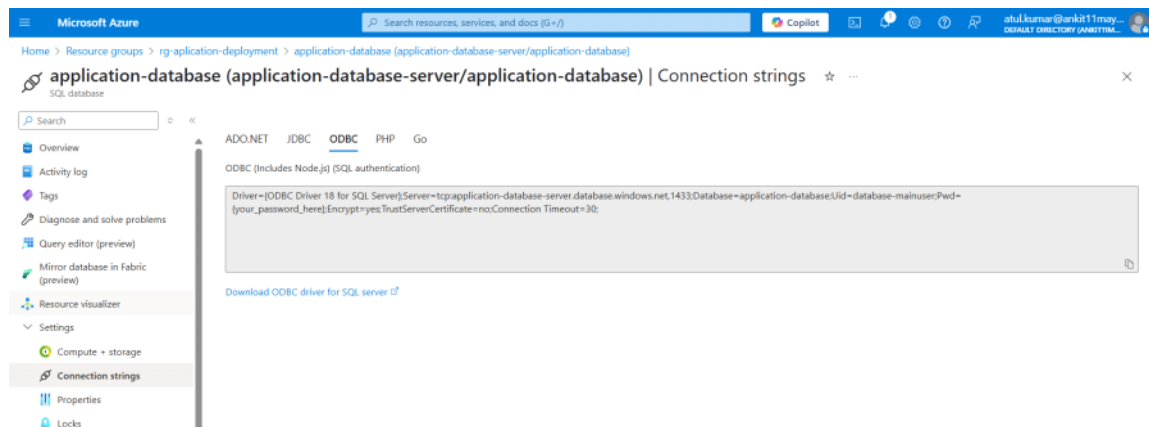
```
git clone <repository_url>
cd <repository_directory>
```

Make the required changes in the specified Files.

Step 2: Update Connection String

Edit the `app.py` file to update the `connection_string` variable with the appropriate connection details for your SQL Server database. Update ODBC Version to 17

Retrieve the connection string URL from the created application-database and make changes in the Backend code according to the Mentioned by developer in the Readme.md file(app.py)



Driver={ODBC Driver 17 for SQL Server};Server=tcp:application-database-server.database.windows.net,1433;Database=application-database;Uid=database-mainuser;Pwd=Adminuser@123;Encrypt=yes;TrustServerCertificate=no;Connection Timeout=30;

```
backendvm@Backend-VM:~$ nano app.py
backendvm@Backend-VM:~$ cat app.py
import pyodbc
import uvicorn
import os
from fastapi import FastAPI
from pydantic import BaseModel
from fastapi.middleware.cors import CORSMiddleware
from dotenv import load_dotenv

# Load environment variables from .env file
load_dotenv()

connection_string = "Driver={ODBC Driver 17 for SQL Server};Server=tcp:application-database-server.database.windows.net,1433;Database=application-database;U
id=database-mainuser;Pwd=Adminuser@123;Encrypt=yes;TrustServerCertificate=no;Connection Timeout=30;"

app = FastAPI()

# Configure CORSMiddleware to allow all origins (disable CORS for development)
app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"], # This allows all origins (use '*' for development only)
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

# Define the Task model
class Task(BaseModel):
    title: str
    description: str

# Create a table for tasks (You can run this once outside of the app)
@app.get("/api")
def create_tasks_table():
    try:
        conn = pyodbc.connect(connection_string)
        cursor = conn.cursor()
        cursor.execute("""
            CREATE TABLE Tasks (
                ID int NOT NULL PRIMARY KEY IDENTITY,

```

Note: We don't hard code the Connection string in the source code. We should keep it in Azure Key Vault and access from there.

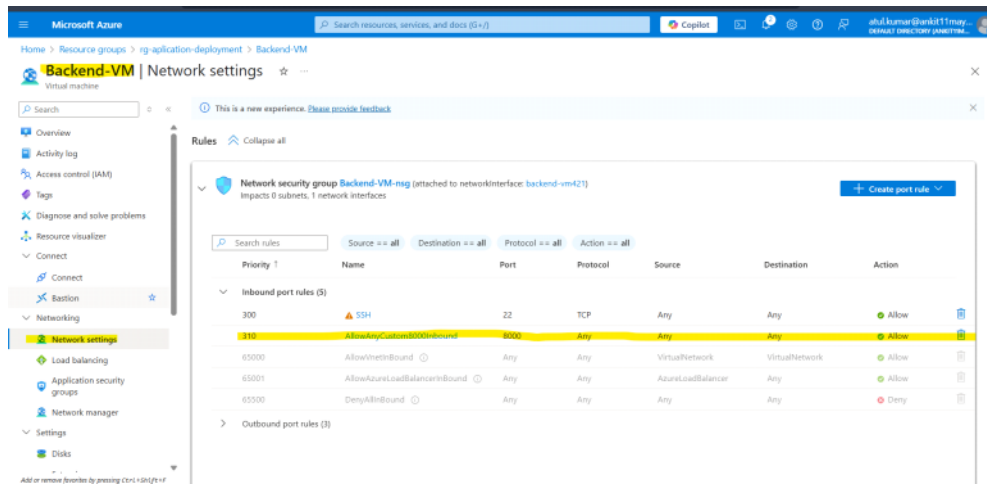
Run The Application by running the code lines one by one. It will be running on Port specified.

Step 3: Run Below Commands to make the application running

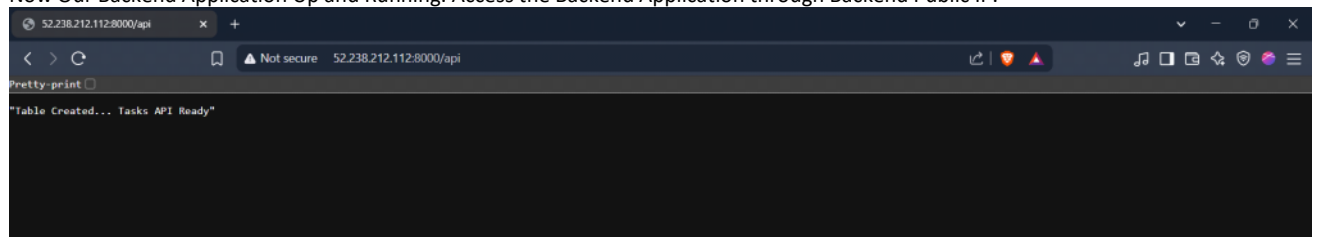
To Run the Application, open a terminal, navigate to the project directory, and run the following command:

```
sudo su
apt-get update && apt-get install -y unixodbc unixodbc-dev
curl https://packages.microsoft.com/keys/microsoft.asc | apt-key add -
curl https://packages.microsoft.com/config/debian/10/prod.list > /etc/apt/sources.list.d/mssql-release.list
apt-get update
ACCEPT_EULA=Y apt-get install -y msodbcsql17
pip install -r requirements.txt
uvicorn app:app --host 0.0.0.0 --port 8000
```

Set Inbound Rule for the Port on which our VM is running.



Now Our Backend Application Up and Running. Access the Backend Application through Backend Public IP.

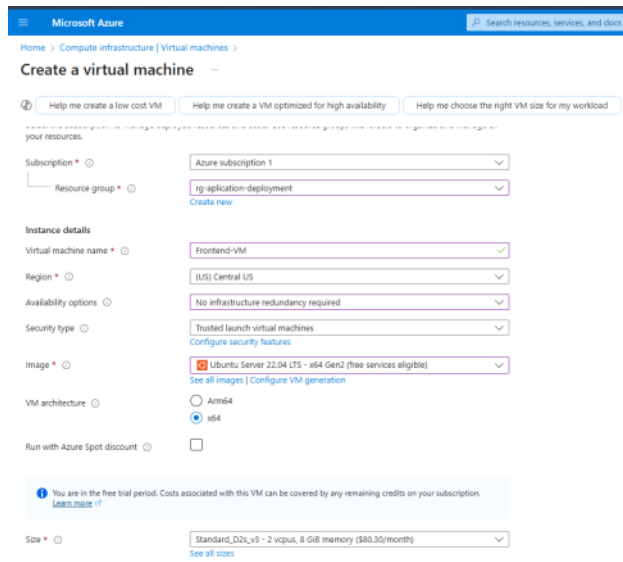


To Troubleshoot Backend Based Application you can use Backend VM connection console and see where it is failing and which all request requests is success and failure.

3. Frontend Set Up

Start Following the Frontend Application Readme.md File in the repository and create VM if there is any specific image Infrastructure requirements.

Create Frontend VM



Enabled Port No. 80 to access Nginx Server and by keep the by default ssh Port No. 22 enabled.

Microsoft Azure

Home > Compute infrastructure | Virtual machines >

Create a virtual machine

Help me create a low cost VM Help me create a VM optimized for high availability Help me choose the right VM size for my workload

Basics Disks **Networking** Management Monitoring Advanced Tags Review + create

Define network connectivity for your virtual machine by configuring network interface card (NIC) settings. You can control ports, inbound and outbound connectivity with security group rules, or place behind an existing load balancing solution. [Learn more](#)

Network interface

When creating a virtual machine, a network interface will be created for you.

Virtual network * [Create new](#)

Subnet * [Manage subnet configuration](#)

Public IP [Create new](#)

NIC network security group ☐ None ☒ Basic ☐ Advanced

Public inbound ports * ☐ None ☒ Allow selected ports

Select inbound ports *

⚠ This will allow all IP addresses to access your virtual machine. This is only recommended for testing. Use the Advanced controls in the Networking tab to create rules to limit inbound traffic to known IP addresses.

Delete public IP and NIC when VM is deleted ☐

Frontend VM Created.

Microsoft Azure

Home > Resource groups > rg-application-deployment >

Frontend-VM

Virtual machine

Search

Help me copy this VM in any region

Connect Start Restart Stop Hibernate Capture Delete Refresh Open in mobile Feedback CLI / PS

Essentials

Resource group (move) : rg-application-deployment

Status : Running

Location : Central US

Subscription (move) : Azure subscription_1

Subscription ID : 18b05931-6bb5-431b-a964-40d9c2211486

Operating system : Linux (ubuntu 22.04)

Size : Standard D2s v3 (2 vcpus, 8 GiB memory)

Public IP address : 74.249.144.105

Virtual network/subnet : application-vnet/frontend-subnet

DNS name : Not configured

Health state : -

Time created : 6/1/2025, 6:51 AM UTC

Tags (edit) : Name : Frontend VM

Properties Monitoring Capabilities (7) Recommendations Tutorials

Virtual machine

Computer name : Frontend-VM

Operating system : Linux (ubuntu 22.04)

VM generation : V2

VM architecture : x64

Export status : Ready

Networking

Public IP address : 74.249.144.105 (Network interface frontend-vm917)

Public IP address (IPv6) : -

Private IP address : 10.0.0.4

Private IP address (IPv6) : -

Virtual network/subnet : application-vnet/frontend-subnet

Now connect to the Frontend VM using SSH.

```
frontendvm@Frontend-VM: ~
Microsoft Windows [Version 10.0.26100.4061]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ATUL KUMAR>ssh frontendvm@74.249.144.105
frontendvm@74.249.144.105's password:
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.8.0-1029-azure x86_64)
```

Start Following the Readme.md File in Frontend Application source code.

Installation

1. Install Node.js and NPM on Ubuntu:

- Make sure you have Node.js 16.x and NPM installed on your machine. If not, you can install them using the following commands:

```
curl -s https://deb.nodesource.com/setup_16.x | sudo bash
sudo apt install nodejs -y
```

Clone the Frontend Application Repository and change to the application repository.
Make the required changes in the specified Files.

Configuration ⚙️

2. Update Backend URL:

- Open the `src/ToDoApp.js` file.
- Locate the variable storing the backend URL and update it with the appropriate value. (* See Below for Privately Configuration)

```
frontendvm@Frontend-VM: ~$ cd ReactToDoUIMonolith/
frontendvm@Frontend-VM:~/ReactToDoUIMonolith$ ls
README.md package-lock.json package.json public src
frontendvm@Frontend-VM:~/ReactToDoUIMonolith$ cd src
frontendvm@Frontend-VM:~/ReactToDoUIMonolith/src$ ls
App.css App.js App.test.js ToDoApp.js index.css index.js logo.svg reportWebVitals.js setupTests.js theme.js
frontendvm@Frontend-VM:~/ReactToDoUIMonolith/src$ nano ToDoApp.js
frontendvm@Frontend-VM:~/ReactToDoUIMonolith/src$ cat ToDoApp.js
import React, { useState, useEffect } from 'react';
import axios from 'axios';
import { Button, TextField, Container, Typography, Grid, Card, CardContent, IconButton } from '@mui/material';
import { Delete } from '@mui/icons-material';
import { Box } from '@mui/material';

const API_BASE_URL = 'http://52.238.212.112:8000/api';
const backgroundImage = process.env.PUBLIC_URL + '/background.jpg';

function ToDoApp() {
  const [tasks, setTasks] = useState([]);
  const [newTask, setNewTask] = useState({ title: '', description: '' });

  const fetchTasks = async () => {
    try {
      const response = await axios.get(`${API_BASE_URL}/tasks`);
      setTasks(response.data);
    } catch (error) {
      console.error('Error fetching tasks', error);
    }
  };

  const createTask = async () => {
    try {
      await axios.post(`${API_BASE_URL}/tasks`, newTask);
      fetchTasks();
      setNewTask({ title: '', description: '' });
    } catch (error) {
      console.error('Error creating task', error);
    }
  };
}
```

Before installing the required node module mentioned in package.json file in React based Application.

```
frontendvm@Frontend-VM:~/ReactToDoUIMonolith$ ls
README.md package-lock.json package.json public src
```

3. Install Dependencies:

- Run the following command to install project dependencies:

```
npm install
```

After npm install all the required node modules and there in the node module folder.

```
frontendvm@Frontend-VM:~/ReactToDoUIMonolith$ ls
README.md node_modules package-lock.json package.json public src
```

After npm run build our build artifacts is ready and ready to deploy/copy and paste in the /var/www/html on nginx server.

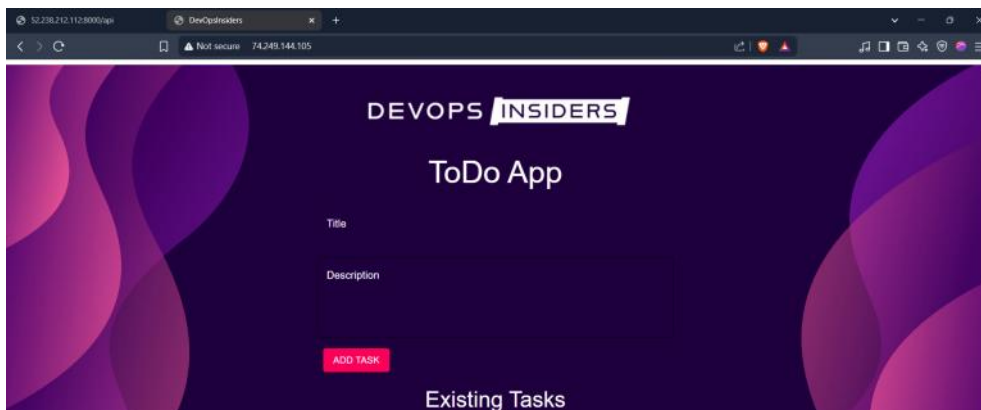
```
frontendvm@Frontend-VM:~/ReactToDoUIMonolith$ ls
README.md build node_modules package-lock.json package.json public src
frontendvm@Frontend-VM:~/ReactToDoUIMonolith$ cd build
frontendvm@Frontend-VM:~/ReactToDoUIMonolith/build$ ls
asset-manifest.json background.jpg devopsinsiderslogo.png favicon.ico index.html logo192.png logo512.png manifest.json robots.txt static
```

Now Install the Nginx server using -> `sudo apt install nginx`

Copy the build folder files to nginx /var/www/html folder.

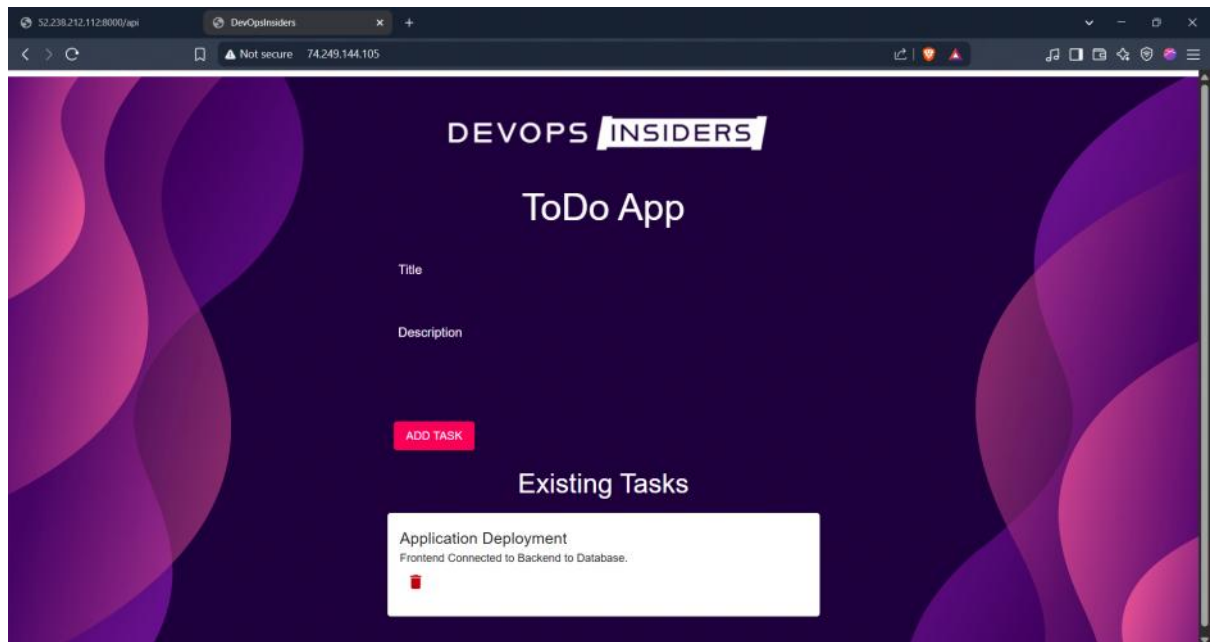
```
frontendvm@Frontend-VM:~/ReactToDoUIMonolith/build$ sudo cp -r * /var/www/html
frontendvm@Frontend-VM:~/ReactToDoUIMonolith/build$ cd /var/www/html
frontendvm@Frontend-VM:/var/www/html$ ls
asset-manifest.json devopsinsiderslogo.png index.html logo192.png manifest.json static
background.jpg favicon.ico index.nginx-debian.html logo512.png robots.txt
```

Now access the Frontend Application using the Public IP.

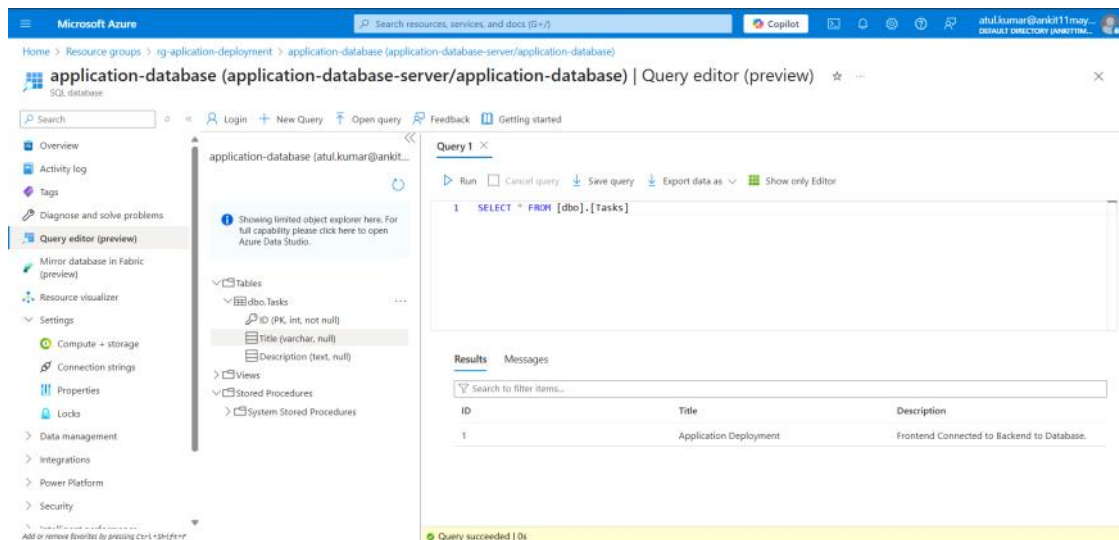


Now when you put the data in the application from Frontend it will be saved to Database.

To Troubleshoot Frontend Based Application you can use Browser Networking section and see where it is failing to send the request and getting the response.



You Can check the Data is now inserted into the Azure SQL Database and can see using the query by accessing the database in cloud using query editor.



Note:

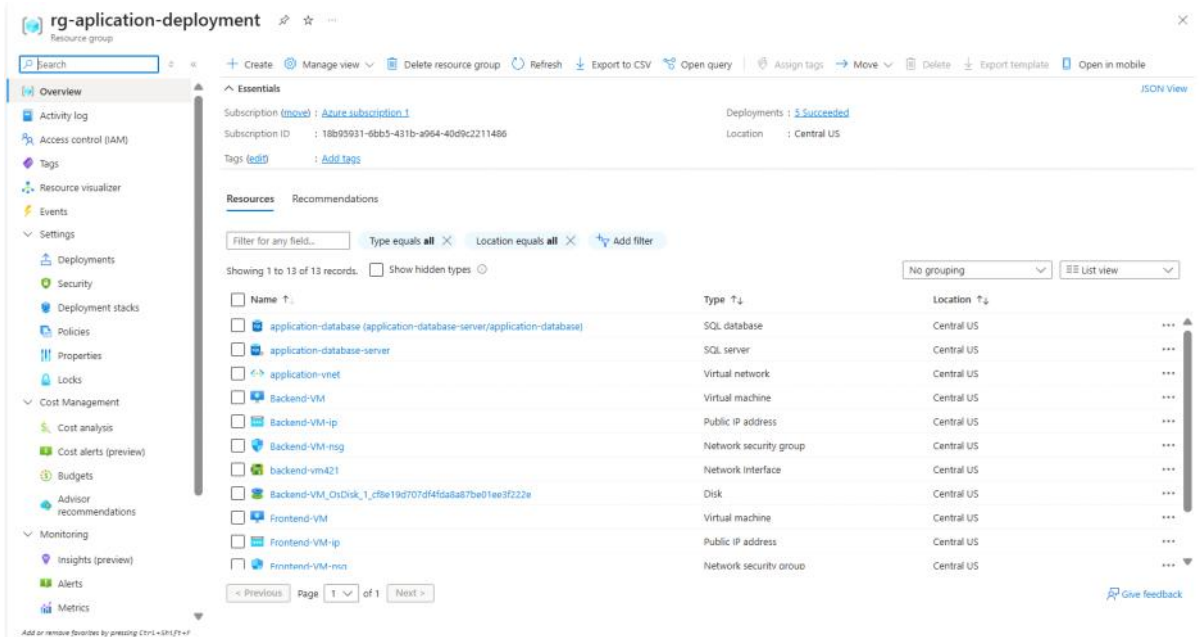
- Keep the status of All-Frontend VM, Backend VM and Azure SQL Database showing running/online to avoid any disconnection.
- We can setup the whole Frontend Application build folder in the local and can use SCP to securely copy the code from the local to the virtual machine home folder

```
scp -r myfolder username@vm_ip:/home/application
```

and then can copy the all the files recursively from the folder using the cp command and past into /var/www/html to access the application.

```
sudo cp -r * /var/www/html
```

- Whole Set up in Resource Group Will Look like this.



- We need to run specific build commands depending on the type of application and the technology it's built with.

Source Code	Dependencies	Build Tool	Artifacts Type	Server Type (deployment)
Angular/React JS	npm install/ package.json	tool = npm npm run build	.html, .css, .js	nginx
Java	pom.xml	tool = maven mvn clean	.jar, .war	tomcat
dotnet	csproj	dotnet build and publish, msbuild	.dll	iis, dotnet runtime
Python	requirements.txt	pip install	No Artifacts	Uvicorn