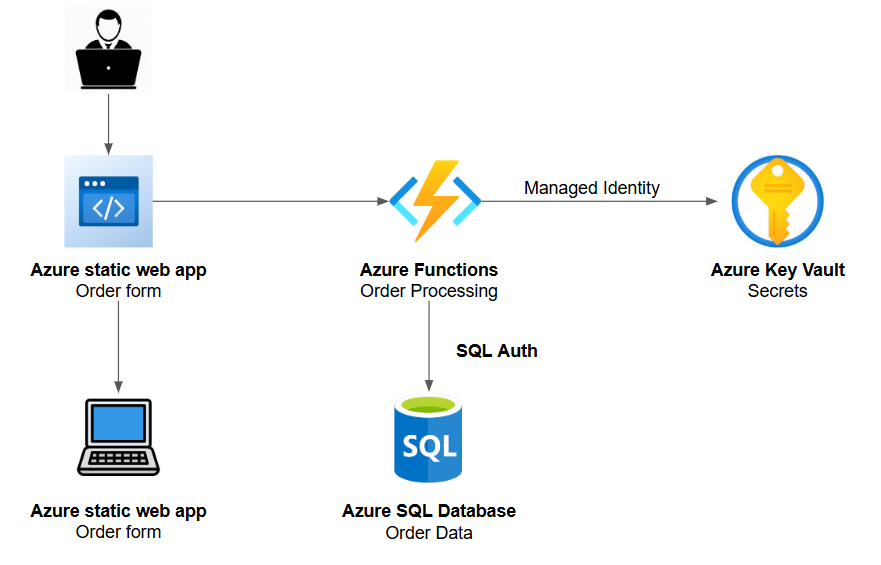
**PROJECT Title - E-commerce Order Processing**

**Objective:** Build a cloud-native order processing system using serverless architecture and secure database connectivity.

**Tech Stack:** Azure Functions (Python), Azure SQL Database, Azure Key Vault, Azure Static Web Apps (HTML/CSS/JS), Azure DevOps/GitHub for CI/CD

**Outcome:** Accept customer orders from a frontend form, securely store them in Azure SQL DB using managed identity and secrets, and provide real-time order placement.

**Architecture Flow:**



1. Frontend – Azure Static Web App

* Technology: HTML + CSS + JS
* Role: Displays a user-friendly Order Form.
* Hosted on: Azure Static Web Apps
* User Action: Customer enters order details and clicks "Place Order".

2. HTTP Request → Azure Function (API)

* The form submission sends a POST request to:
* Data like name, email, product, quantity, etc., is sent to the backend.

3. Backend – Azure Function App (Python)

* Tech Used: Python (Azure Function)
* Role:
  + Receives the order.
  + Validates required fields.
  + Creates a unique tracking code.
  + Retrieves SQL credentials from Azure Key Vault.
  + Inserts order into Azure SQL Database.

4. Azure Key Vault

* Stores:
  + **sqlusr** – SQL username
  + **sqlpasswrd** – SQL password
* Used to securely fetch credentials via Managed Identity.
* Only Azure Function (via Managed Identity) can access secrets.

5. Azure SQL Database

* Stores the order data in a table **Orders**.
* Fields include:
  + **CustomerName, Email, ProductName, TrackingCode,** etc.
* You can query the database to:
  + View customer orders.
  + Track order history.

**Step 1** - Created Azure function on the cloud (EcommerceOP)

**Step 2** - Turned on the identity in azure function

**Step 3** - Created Azure SQL DB and server (ecommerceDB, eautos)

**Step 4** - Created the Azure keyvault (ecommerceKeyV)

**Step 5** - In key vault, assigned access control (Add role assignment(user) - Key Vault Administrator)

**Step 6** - In key vault, assigned access control (Add role assignment(Managed Identity) - Key Vault Administrator)

**Step 7** - In key vault, created the secrets key

Create First Secret - manual\_Name: sqlusr Secret Value: funcusr

Create Second Secret - manual \_ Name: sqlpasswrd \_ Secret Value: yourpasswordhere

**Step 8** - Connected the SQL server through SQL SSMS   
**ecommerceDB**-

SET DATEFORMAT ymd

SET ARITHABORT, ANSI\_PADDING, ANSI\_WARNINGS, CONCAT\_NULL\_YIELDS\_NULL, QUOTED\_IDENTIFIER, ANSI\_NULLS, NOCOUNT ON

SET NUMERIC\_ROUNDABORT, IMPLICIT\_TRANSACTIONS, XACT\_ABORT OFF

GO

CREATE TABLE Orders (

OrderId INT IDENTITY(1,1) PRIMARY KEY,

CustomerName NVARCHAR(100),

Email NVARCHAR(100),

Address NVARCHAR(200),

ProductName NVARCHAR(100),

Quantity INT,

PaymentStatus NVARCHAR(20),

TrackingCode NVARCHAR(50),

OrderDate DATETIME DEFAULT GETDATE()

);

INSERT ecommerceDB.dbo.Orders(CustomerName, Email, Address, ProductName, Quantity, PaymentStatus, TrackingCode) VALUES ('Alice Johnson', 'alice@example.com', '123 Main St, Bangalore', 'Phone', 2, 'Paid', 'ORD-AB123456')

INSERT ecommerceDB.dbo.Orders(CustomerName, Email, Address, ProductName, Quantity, PaymentStatus, TrackingCode) VALUES ('Rahul Verma', 'rahul.verma@example.com', '45 MG Road, Mumbai', 'Laptop', 1, 'Paid', 'ORD-CD789012')

INSERT ecommerceDB.dbo.Orders(CustomerName, Email, Address, ProductName, Quantity, PaymentStatus, TrackingCode) VALUES ('Sara Ali', 'sara.ali@example.com', '89 Koramangala, Bangalore', 'Camera', 3, 'Pending', 'ORD-EF345678')

INSERT ecommerceDB.dbo.Orders(CustomerName, Email, Address, ProductName, Quantity, PaymentStatus, TrackingCode) VALUES ('Ravi Kumar', 'ravi.k@example.com', '12 Lajpat Nagar, Delhi', 'Headphones', 4, 'Paid', 'ORD-GH901234')

INSERT ecommerceDB.dbo.Orders(CustomerName, Email, Address, ProductName, Quantity, PaymentStatus, TrackingCode) VALUES ('Priya Das', 'priya.d@example.com', '76 Park Street, Kolkata', 'Phone', 1, 'Paid', 'ORD-IJ567890')

GO

select \* from Orders

CREATE USER funcusr FOR LOGIN funcusr;

GO

EXEC sp\_addrolemember 'db\_datareader', 'funcusr';

EXEC sp\_addrolemember 'db\_datawriter', 'funcusr';

GO

SELECT r.name AS role\_name

FROM sys.database\_role\_members m

JOIN sys.database\_principals u ON u.principal\_id = m.member\_principal\_id

JOIN sys.database\_principals r ON r.principal\_id = m.role\_principal\_id

WHERE u.name = 'funcusr';

**Master-**

CREATE LOGIN funcusr WITH PASSWORD = 'yourpasswordhere';

GO

SELECT name, type\_desc

FROM sys.sql\_logins

WHERE type\_desc = 'SQL\_LOGIN'

**Step 9** - Created the python Azure function code on VC code

**Step 10** - Created the HTML code

**Step 11** - Run [function.py](http://function.py) and next index.html. Enter the order details, click place order

**Step 12** - Now the data should be reflected on the database. Which means the project is running locally.

**Step 13** - In **Test/Run**, use application/x-www-form-urlencoded as content type, and make sure you pass correct keys like

**name=Customer1&email=customer@gmail.com&address=123 Saraswathipuram, Mysuru&product=Laptop&quantity=1**

**Step 14** - Now deploy the index.html code to github using git commands.

**Step 15** - Created Azure Static Web App and connected it to the GitHub repository hosting index.html. Defined the build location as root.

**Step 16 -** Now click on the static web app url and run the code by filling the required data and place the order.

**Step 17** - Now check the data in SSMS or Query editor(Azure SQL) by giving a query, the data will be reflected.

**CI/CD Section (Deployment Approach)**

* Initially deployed using **GitHub Actions** with Azure Static Web Apps.
* Later migrated the deployment to **Azure DevOps Pipelines** for both frontend and backend.
* Configured **Azure DevOps pipeline YAML** with correct app\_location, api\_location, and used Azure Static Web App build task.
* Enabled **CORS** and **allowed origins** to connect frontend to backend securely.

**NOTE:**

* Used Azure Key Vault to securely store SQL credentials.
* Enabled System-assigned Managed Identity to allow the Function to access secrets without hardcoding credentials.
* Ensured Minimum required SQL roles were granted (db\_datareader, db\_datawriter only).

**Error:**

1. Before deployment

Error: “exception caught attempting to connect to SQL DB”

> Because of the mismatch password between Azure Key vault and SQL server(SSMS)

> Modified and gave the same password in both Azure key vault and SQL Server(SSMS)

2. After deployment

Error: “exception caught attempting to connect to SQL DB”

> Azure functions do not support ODBC Driver 17 for SQL Server**.**

> Change the version of ODBC Driver to 18

> Should give the SQL connection string by going to the Environment variable in the azure function.

3. After deployment

There is a chance of getting message as - “Connection failed to server”

> Goto Azure function, in the left side menu click on API.

> Goto CORS, in Allowed Origins, gives the URL of the static web app and saves it.