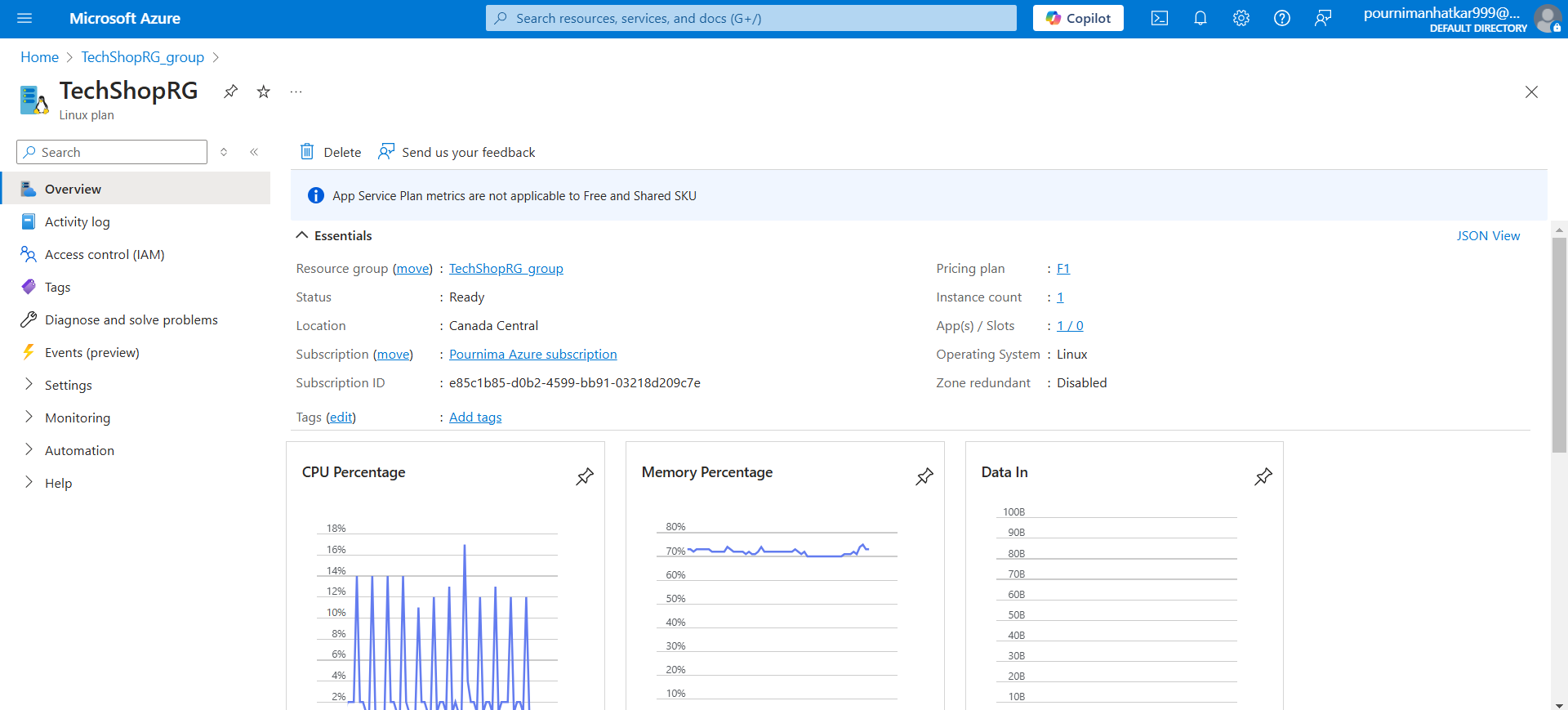
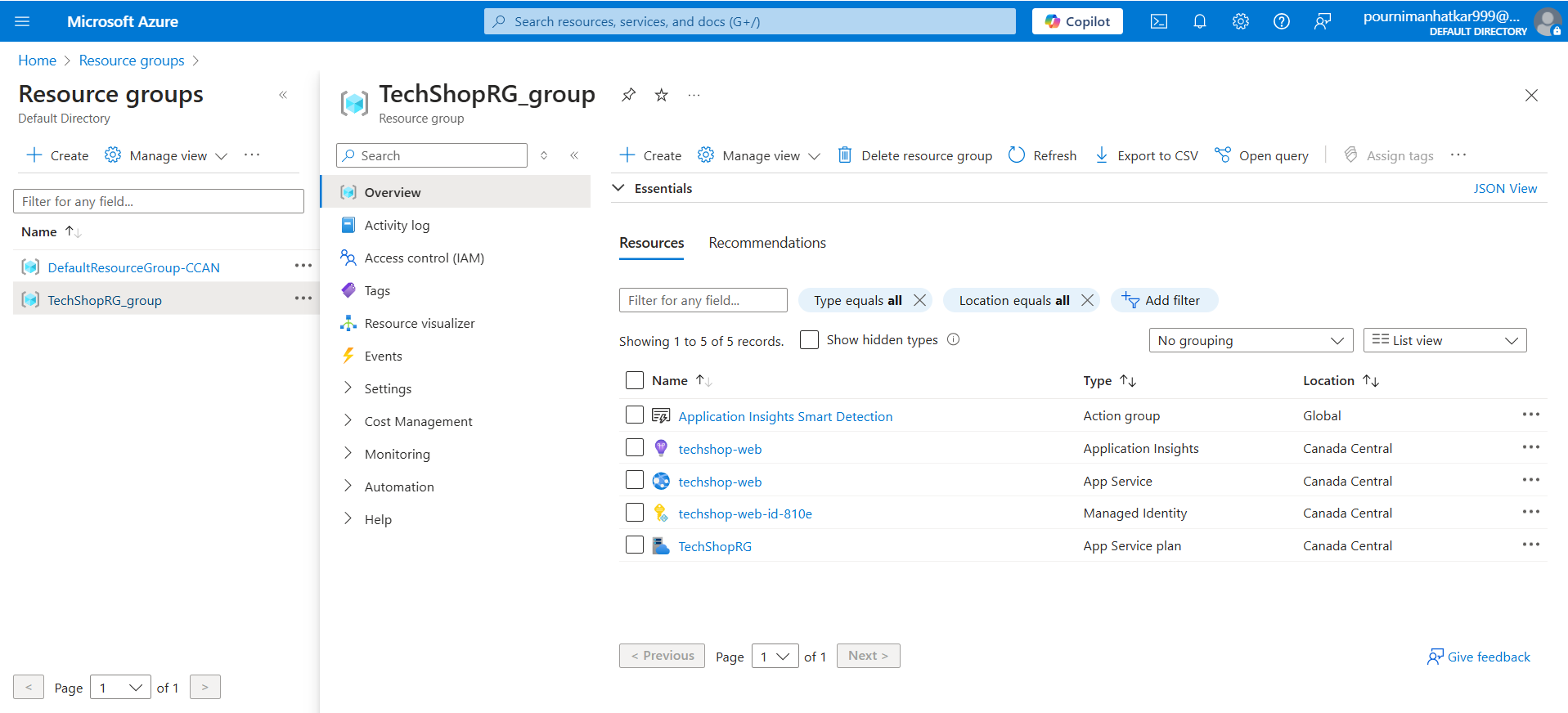
1. Web Application Hosting with Azure App Service

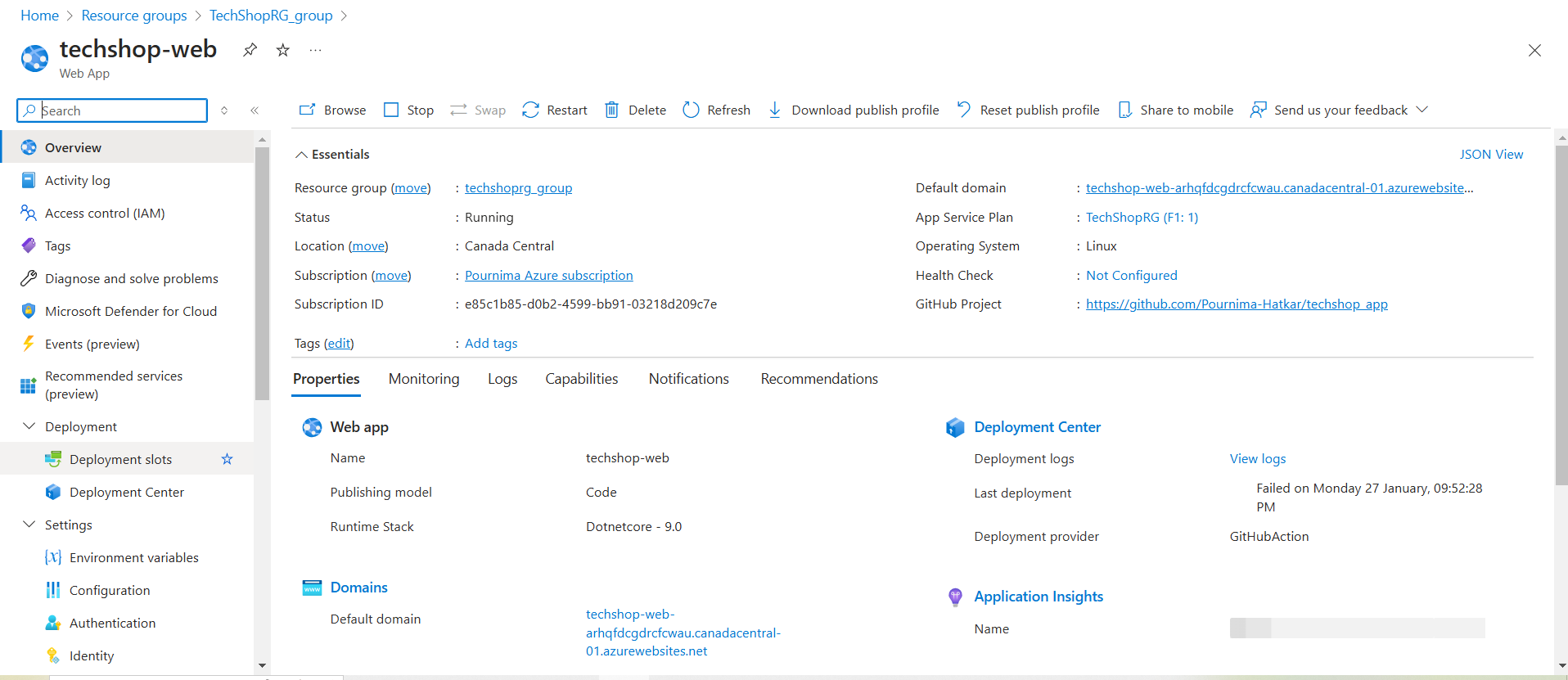
Objective: Host TechShop’s customer-facing website on Azure.

Step 1.1: Create an Azure App Service Plan

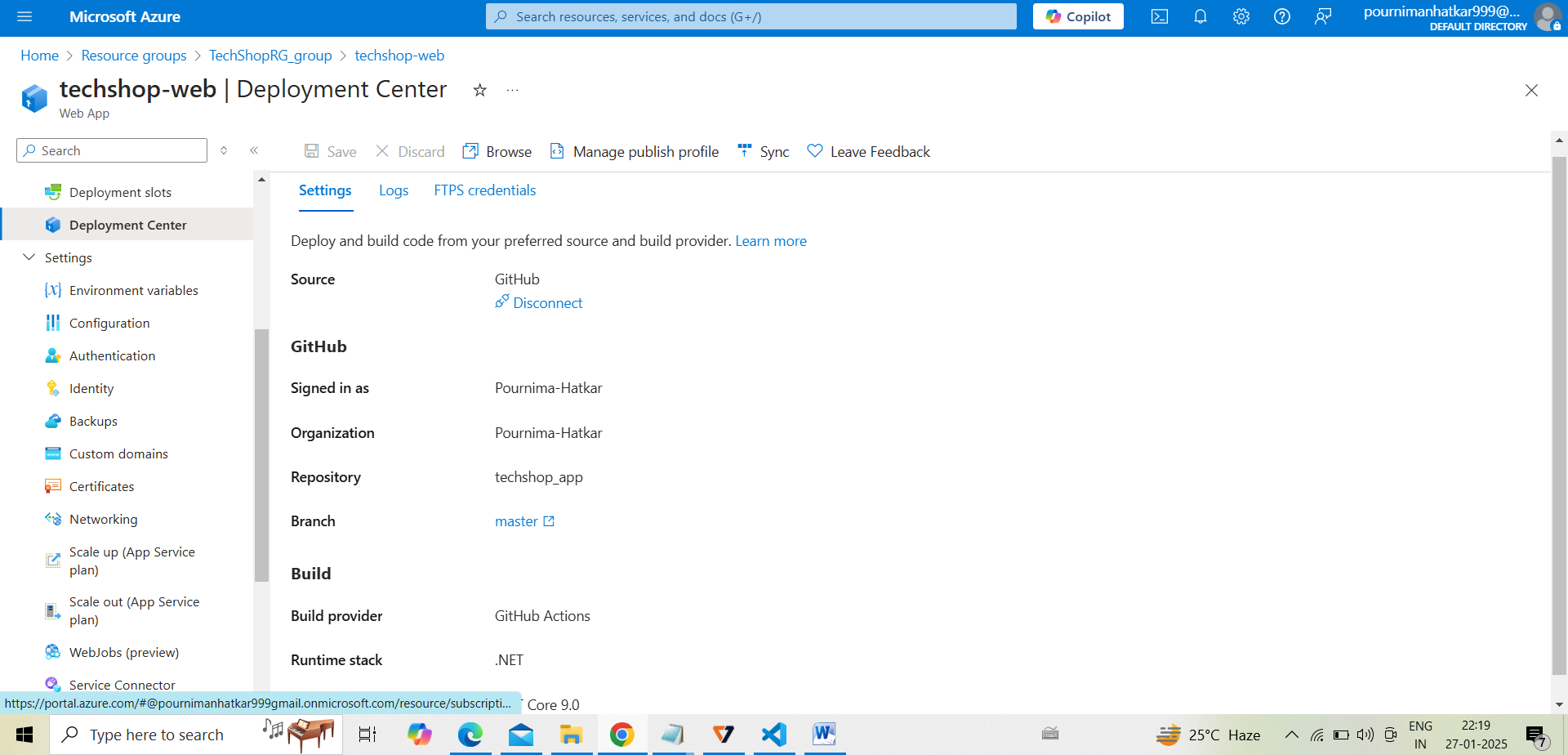


Step 1.2: Create an Azure App Service

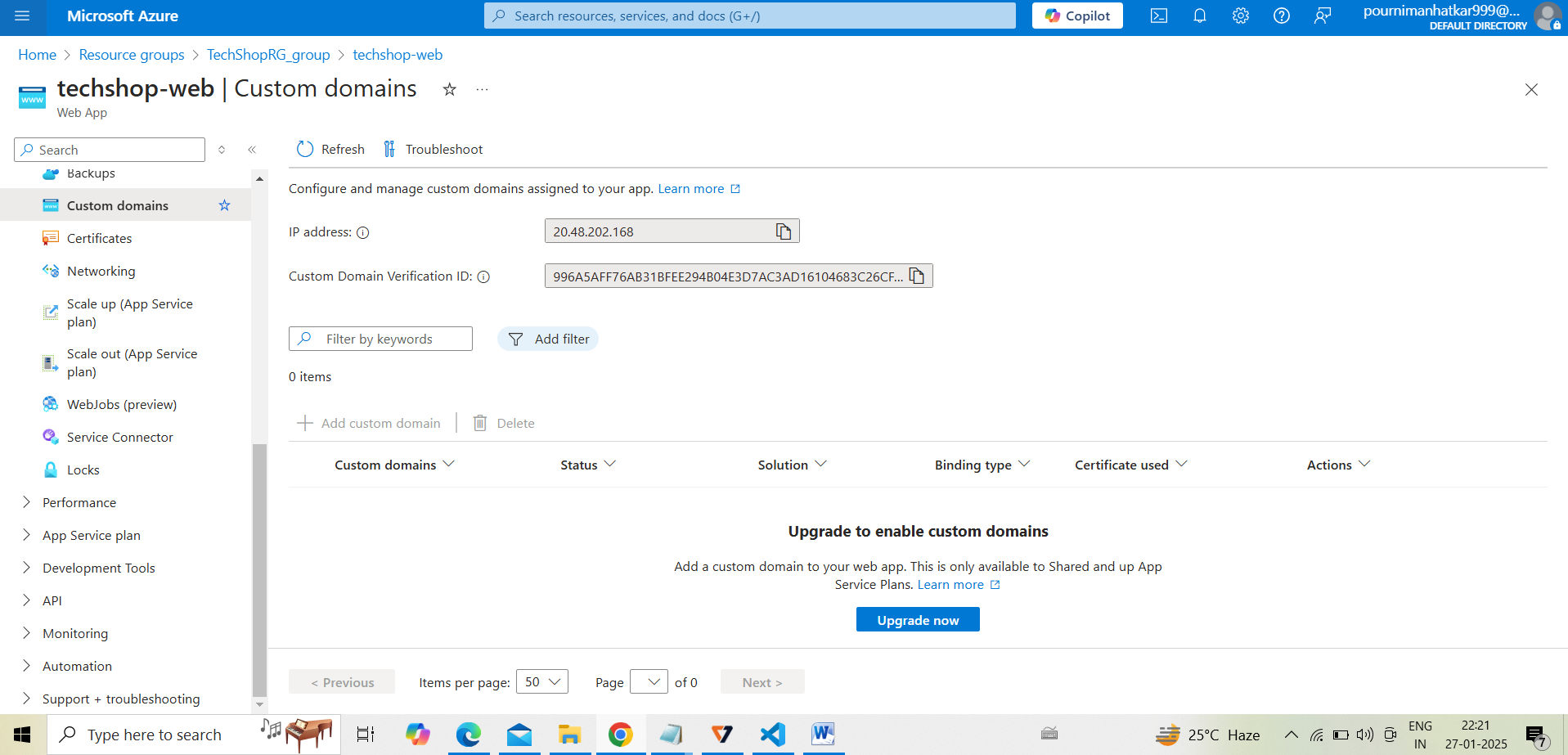


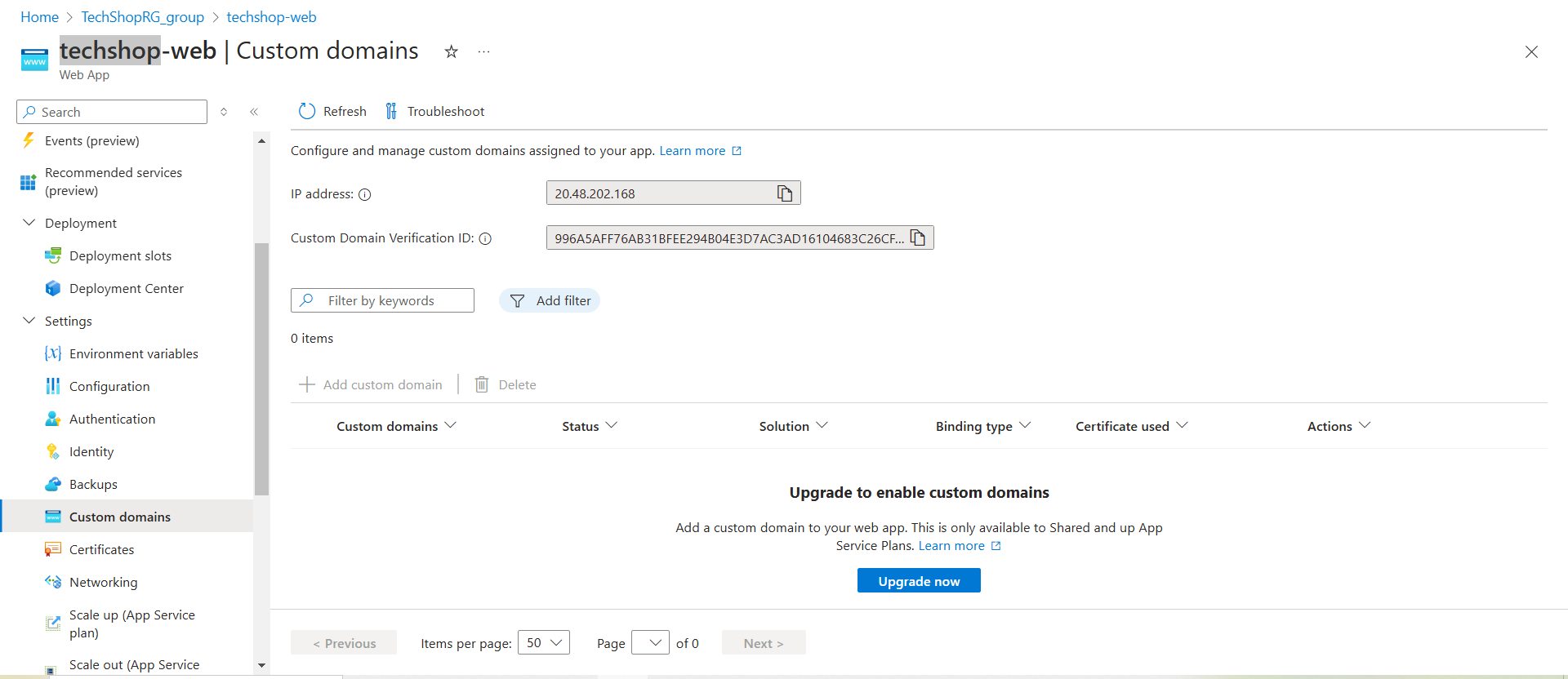


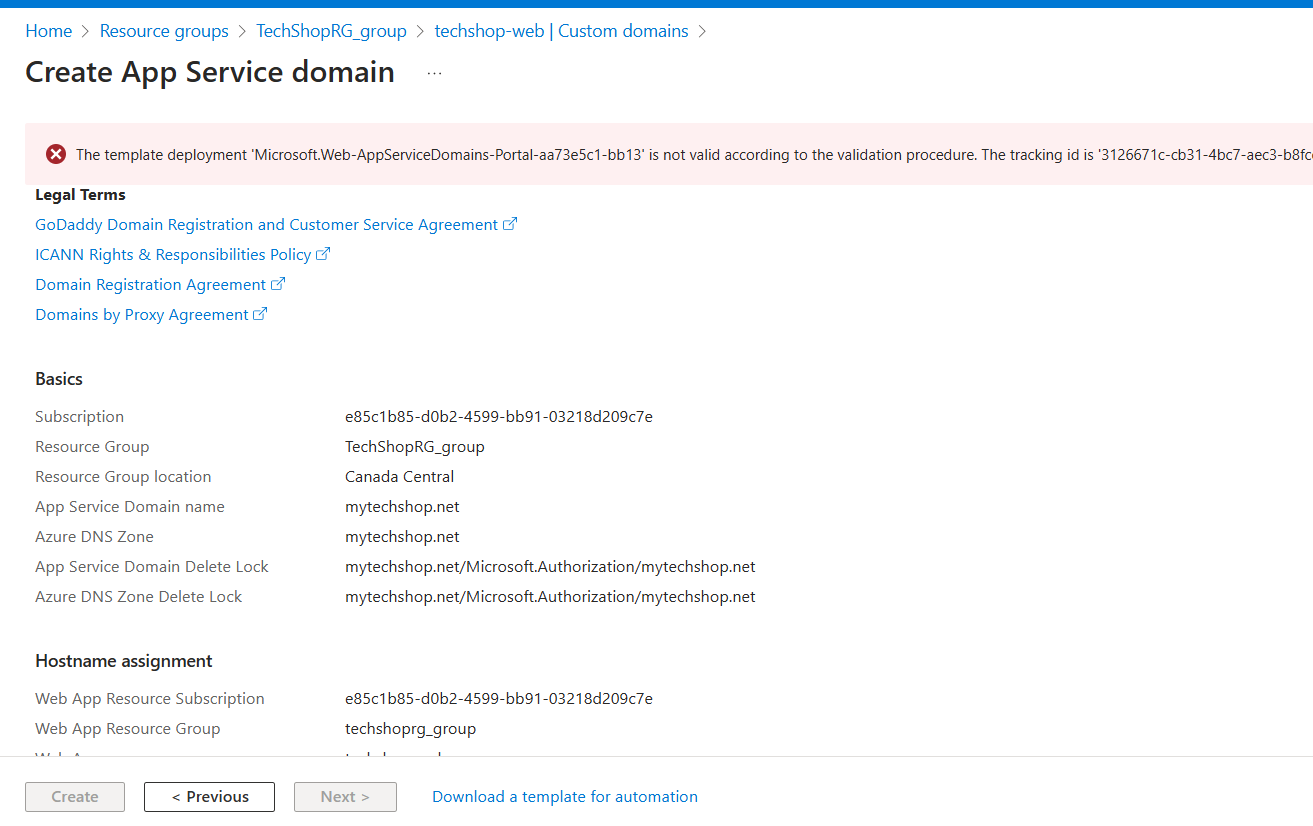
Step 1.3: Deploy the Web Application

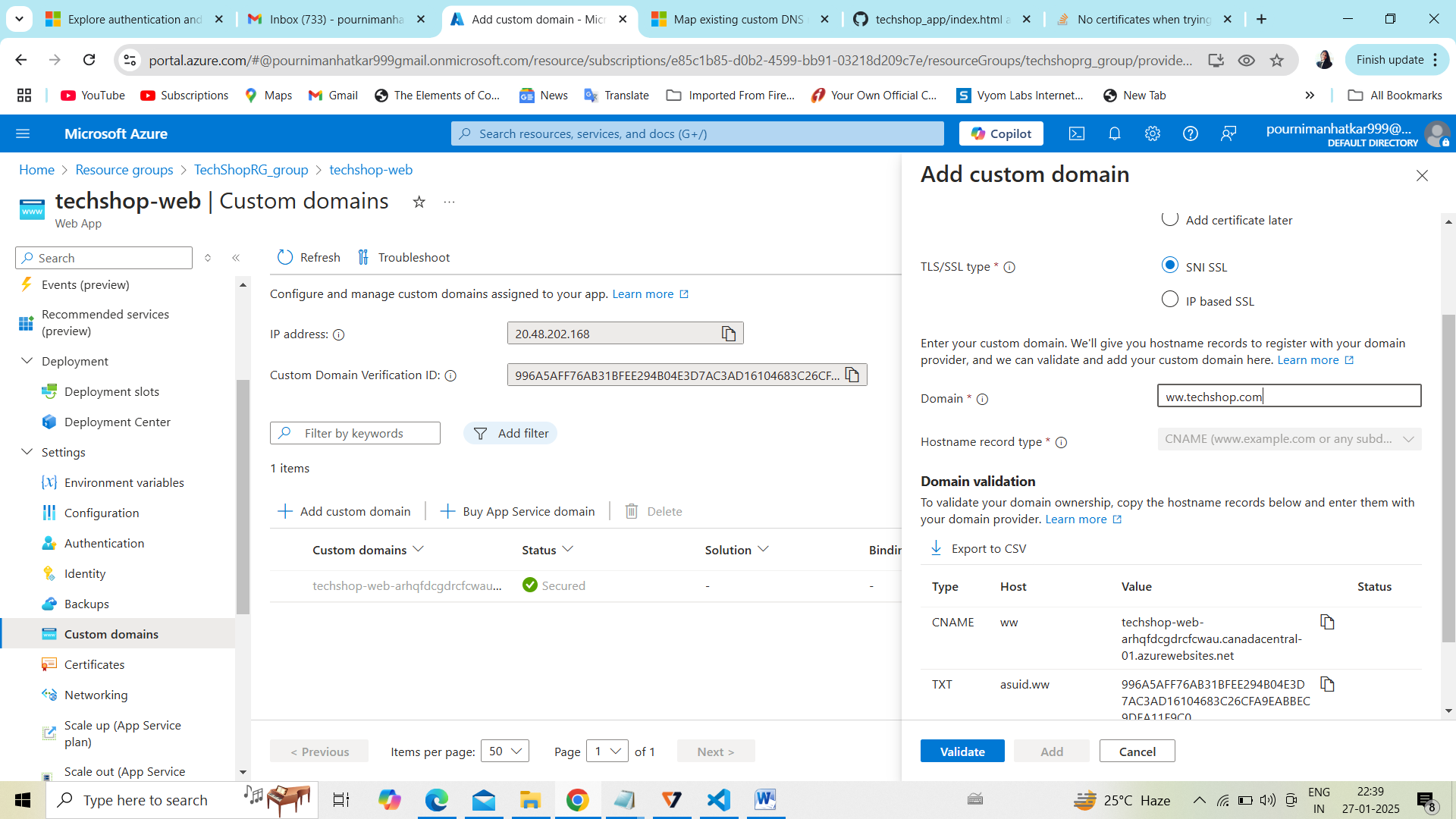


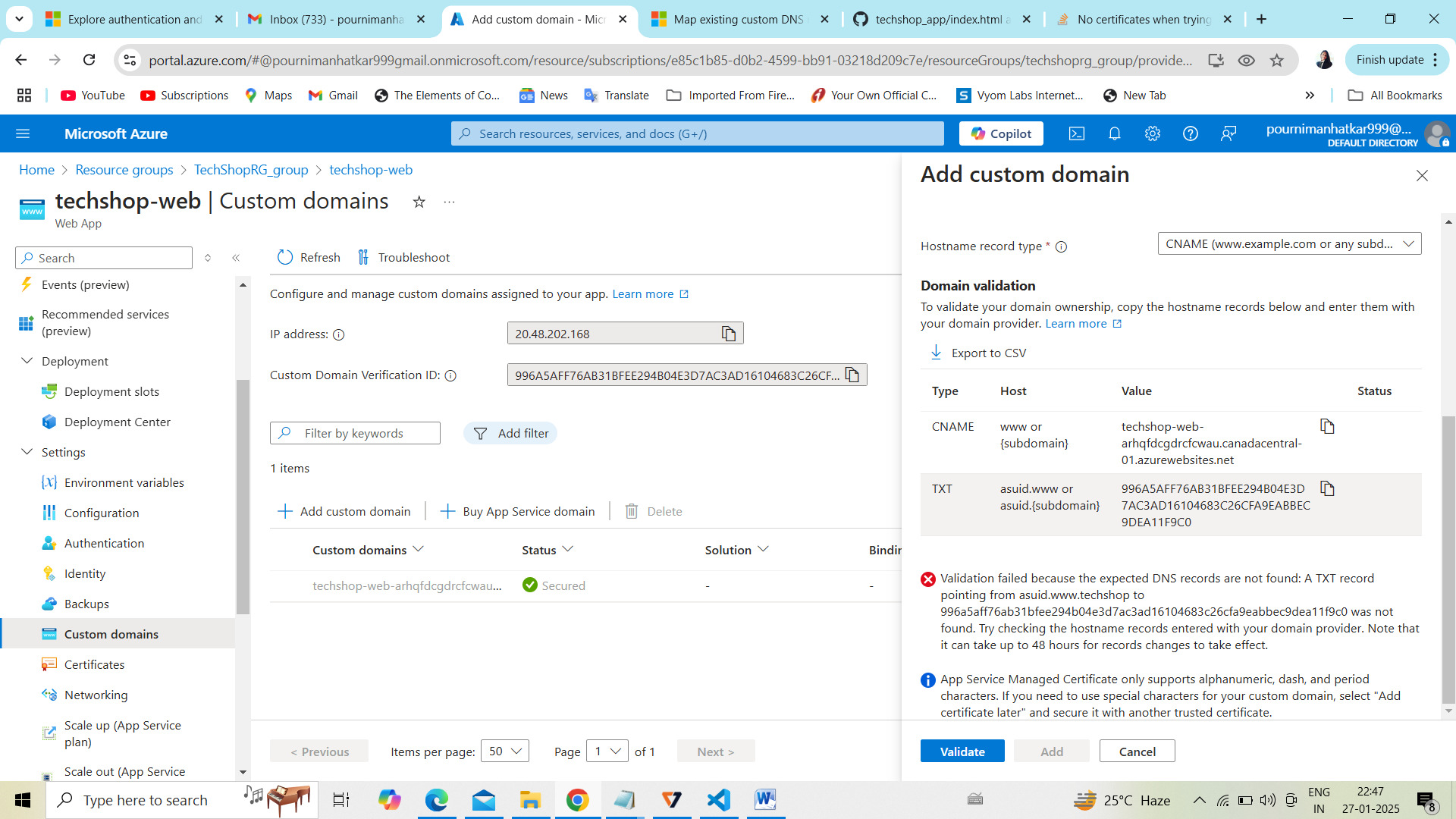
Step 1.4: Configure Custom Domain and SSL

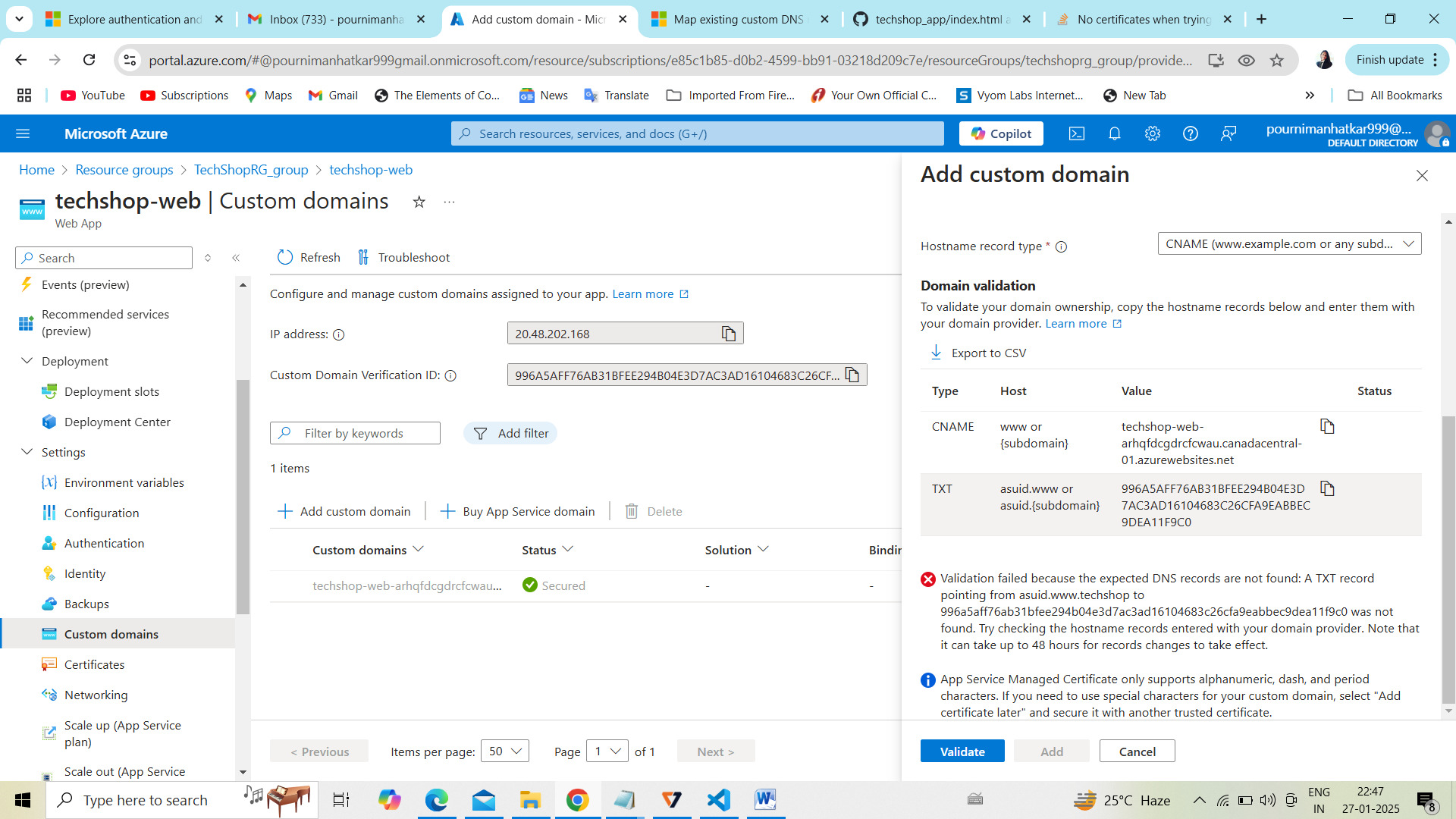




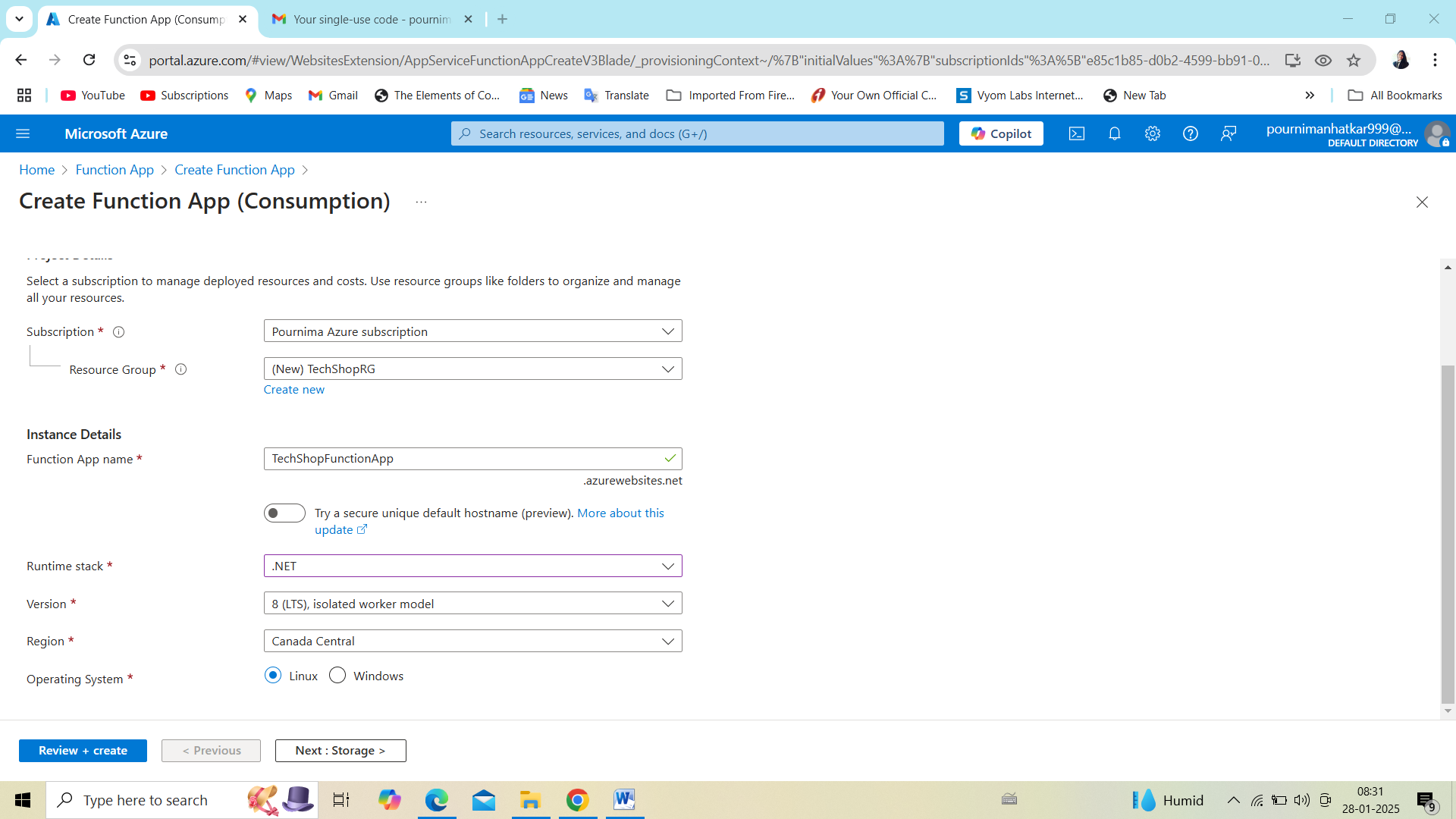


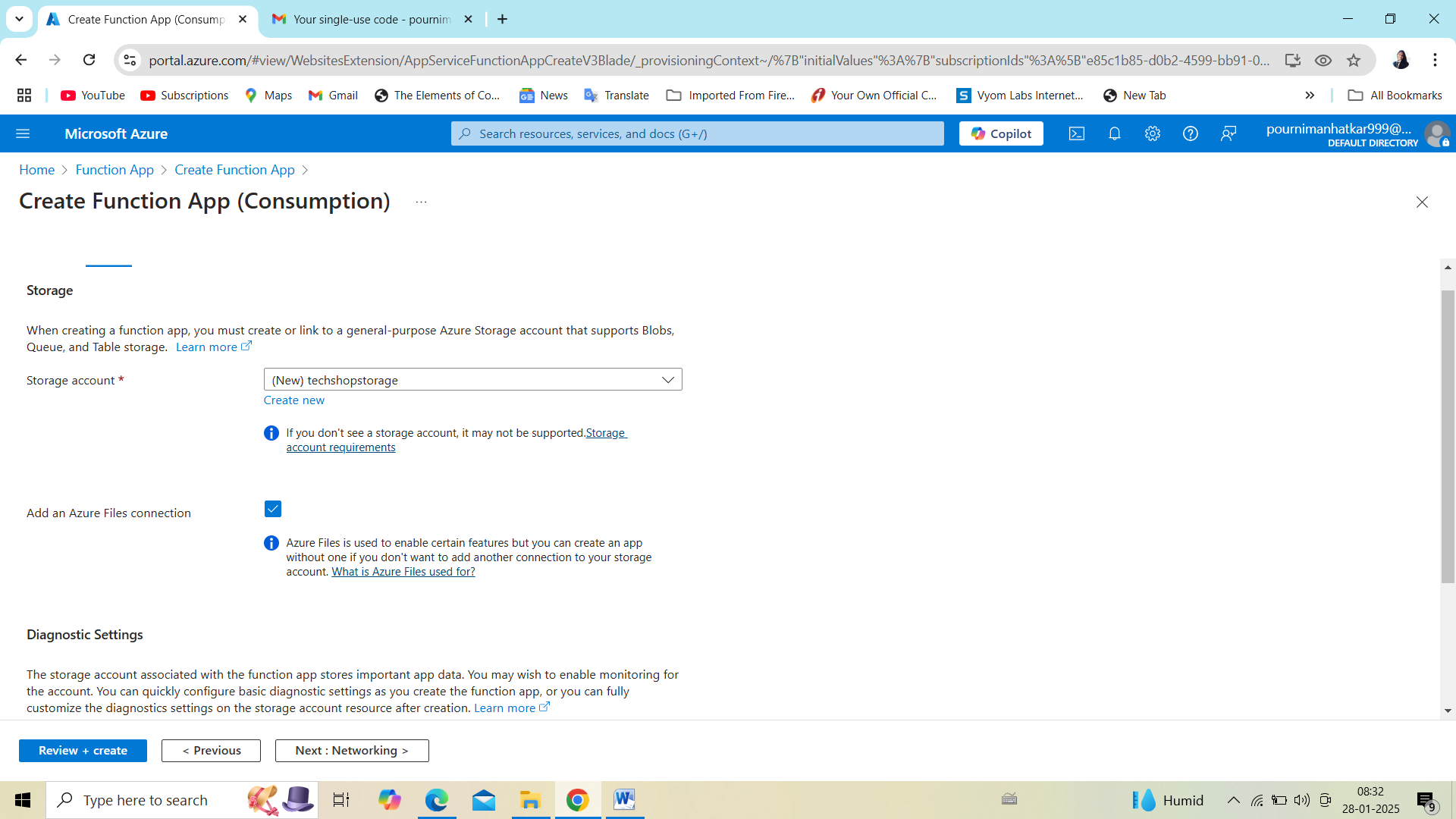






1. Backend API Development with Azure Functions
2. Objective: Develop backend APIs using Azure Functions for managing products, orders, and user authentication.
3. Step 2.1: Create an Azure Function App

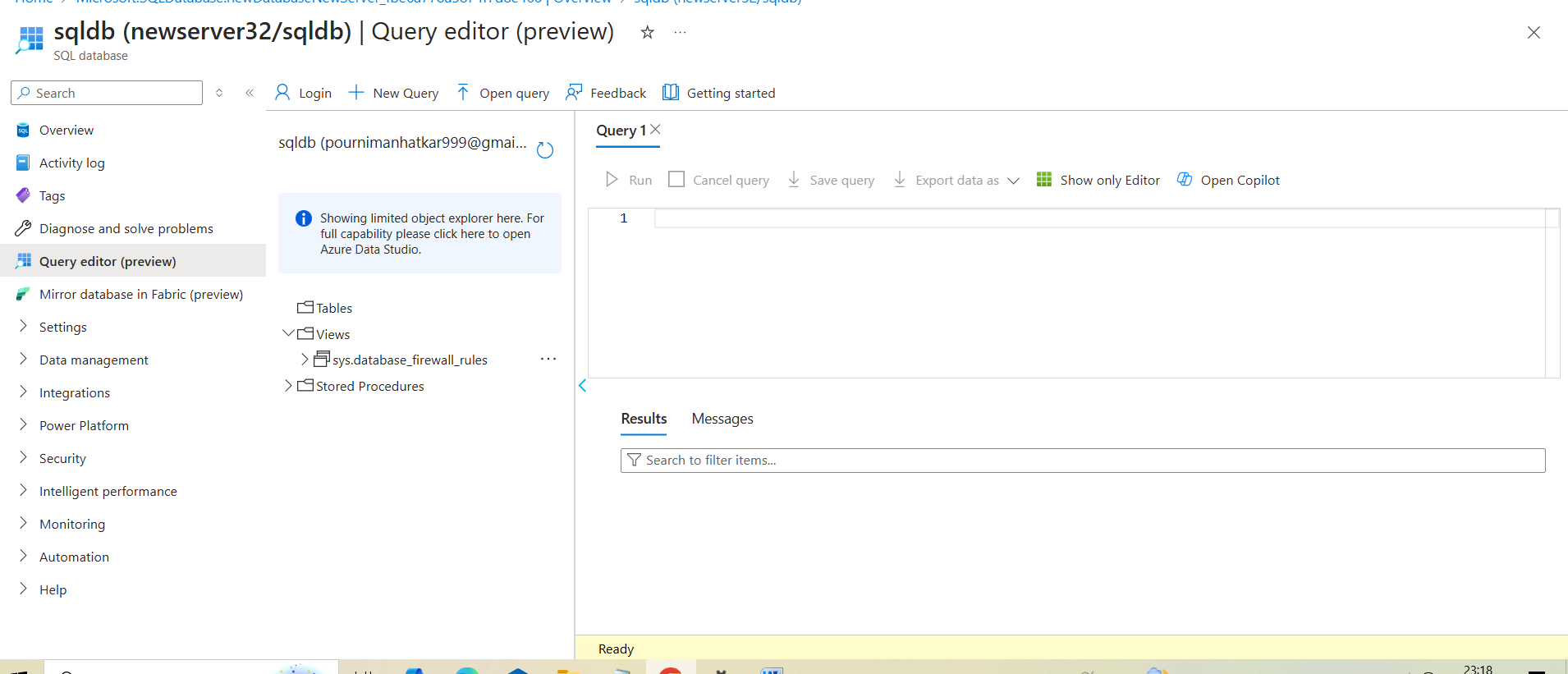




3. Data Storage with Azure SQL Database and Azure Blob Storage

Objective: Set up Azure SQL Database and Blob Storage to store eCommerce data (e.g., products, orders).

Step 3.1: Create Azure SQL Database

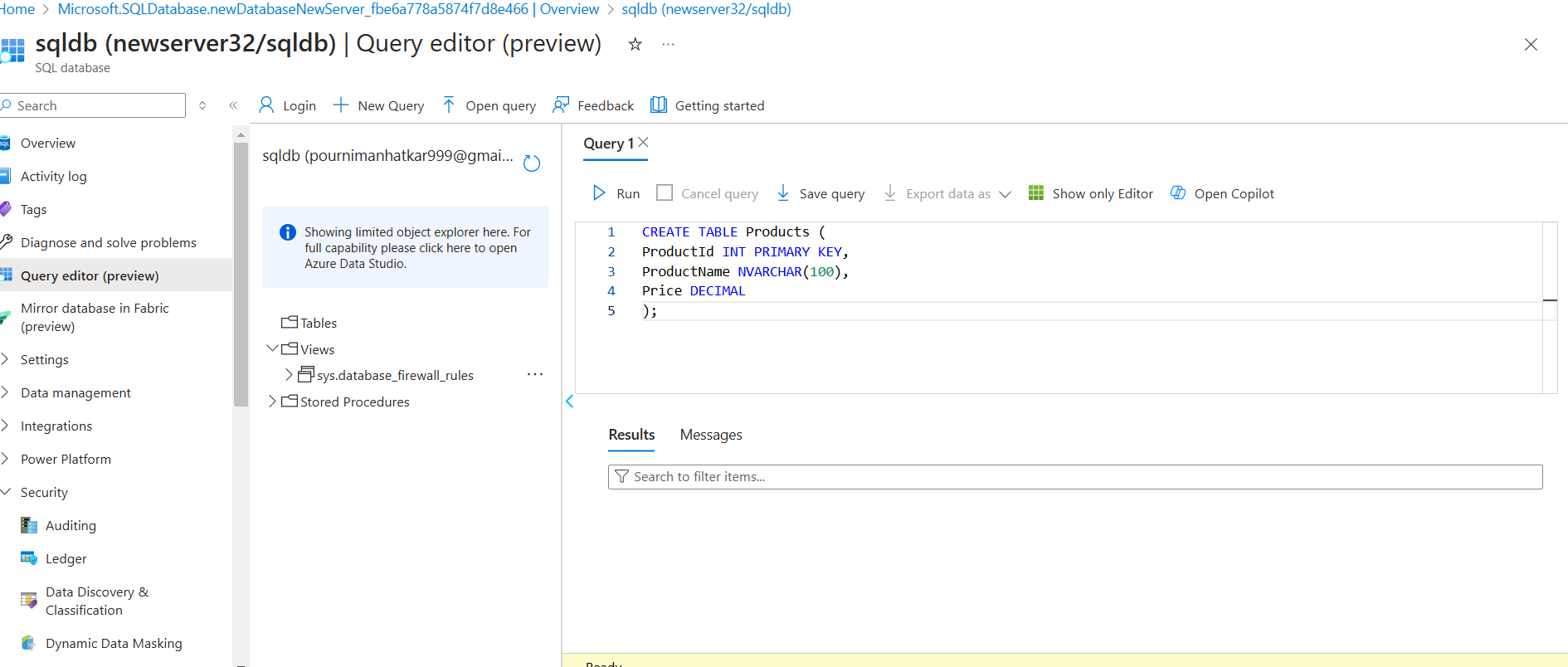


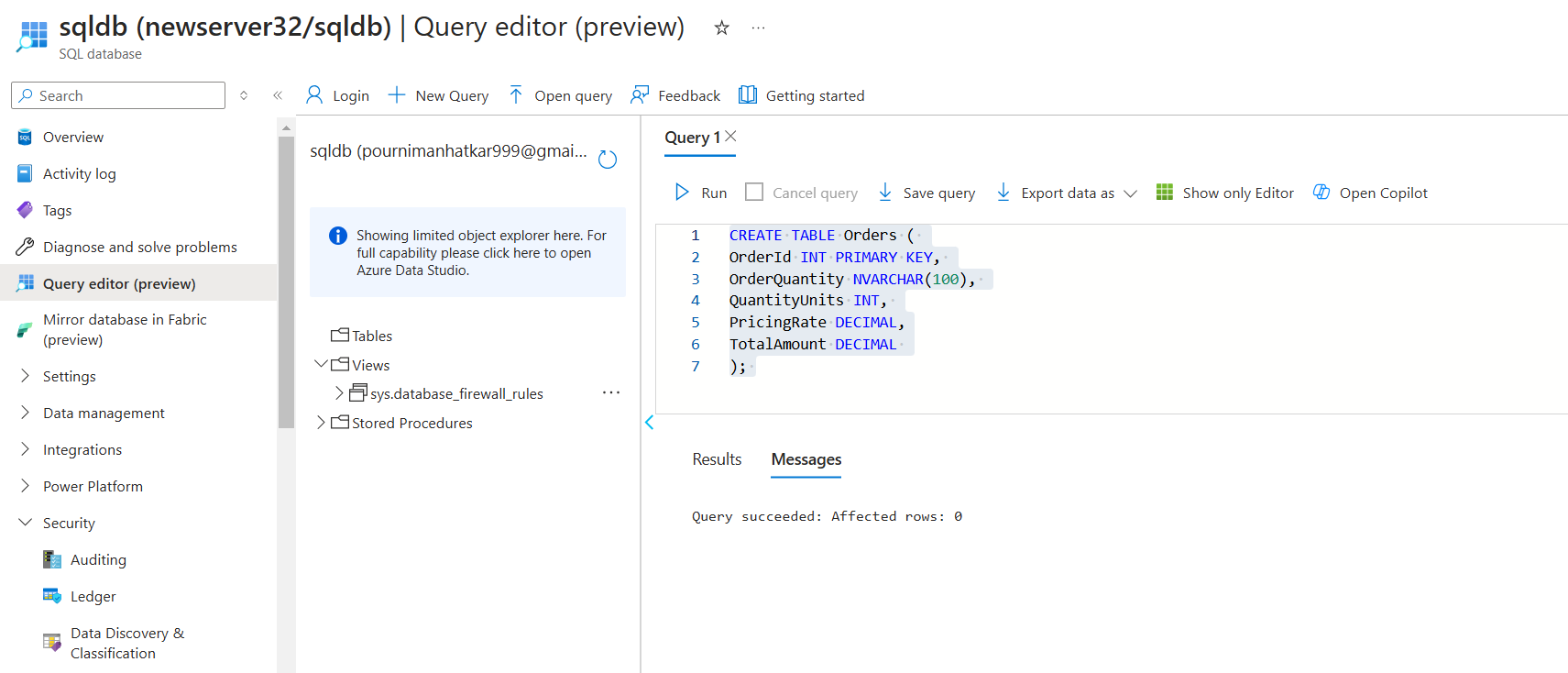
Step 3.2: Set up Tables in SQL Database

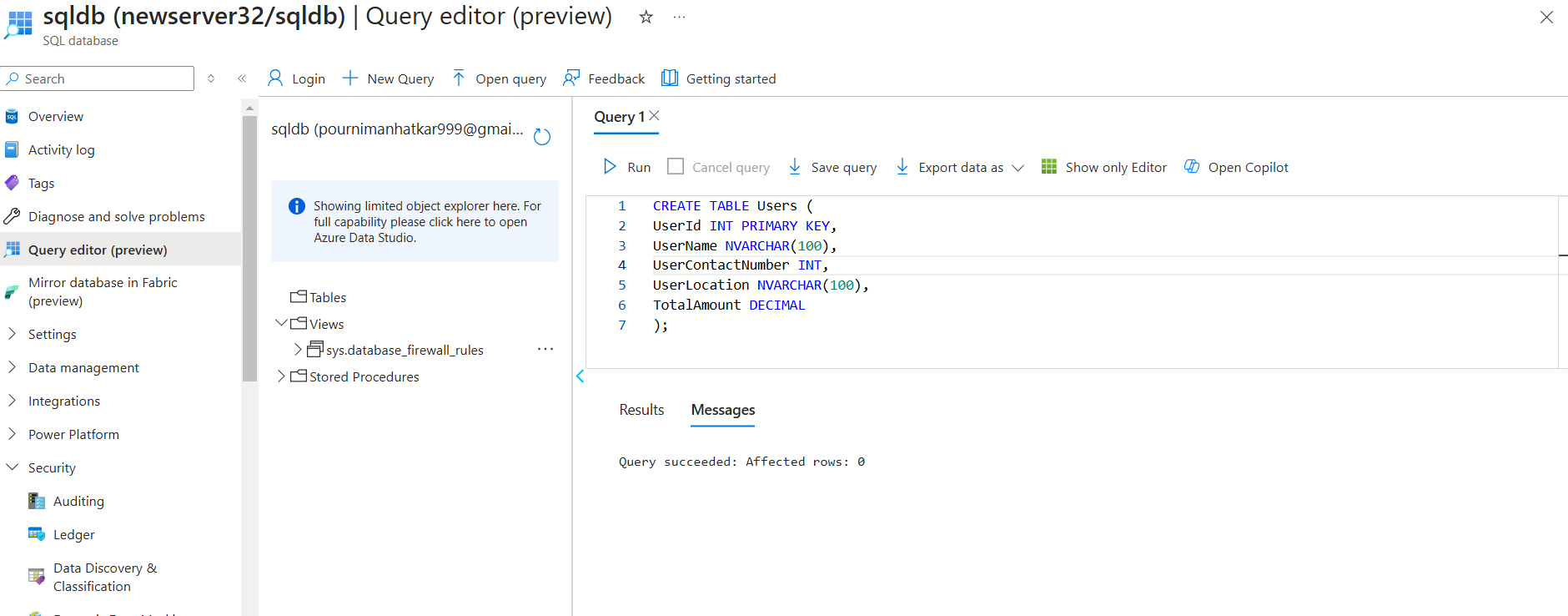
• After the SQL Database is created, go to Query editor.

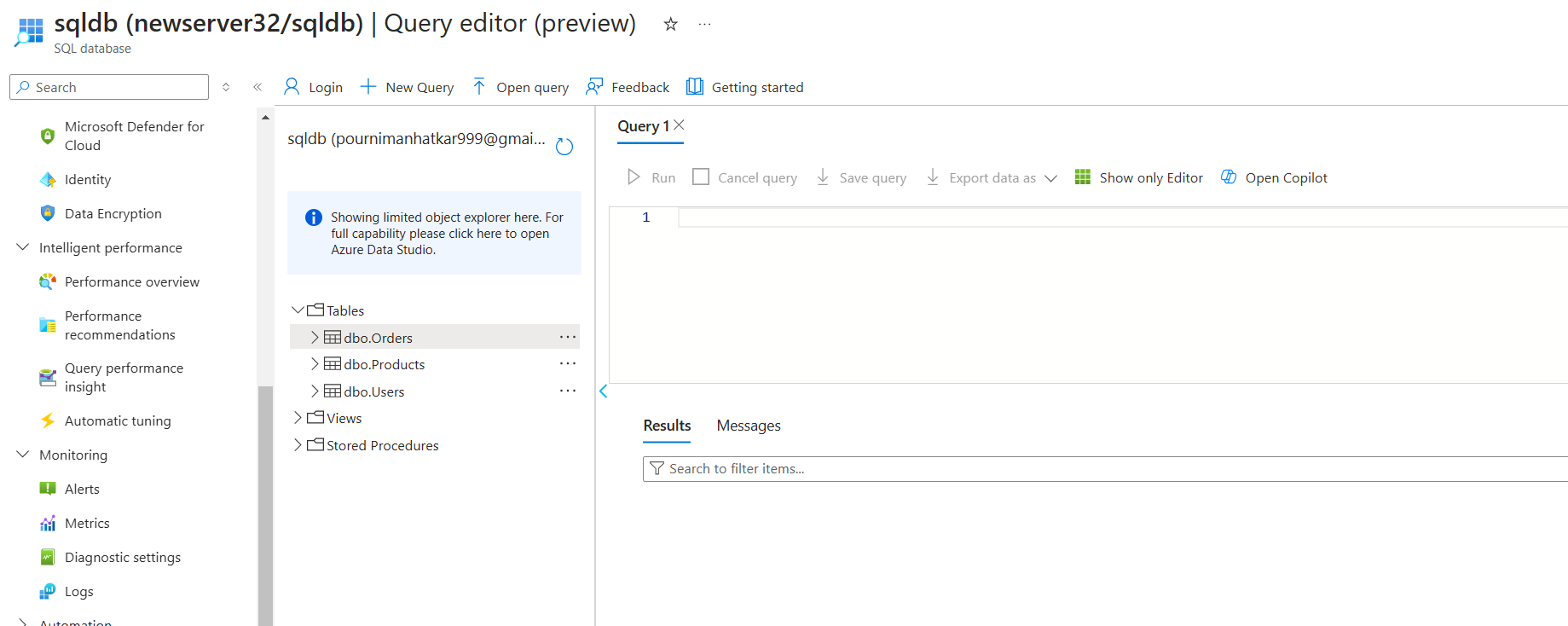
• Create the necessary tables for Products, Orders, Users, etc., using SQL queries.

Example: CREATE TABLE Products ( ProductId INT PRIMARY KEY, ProductName NVARCHAR(100), Price DECIMAL );





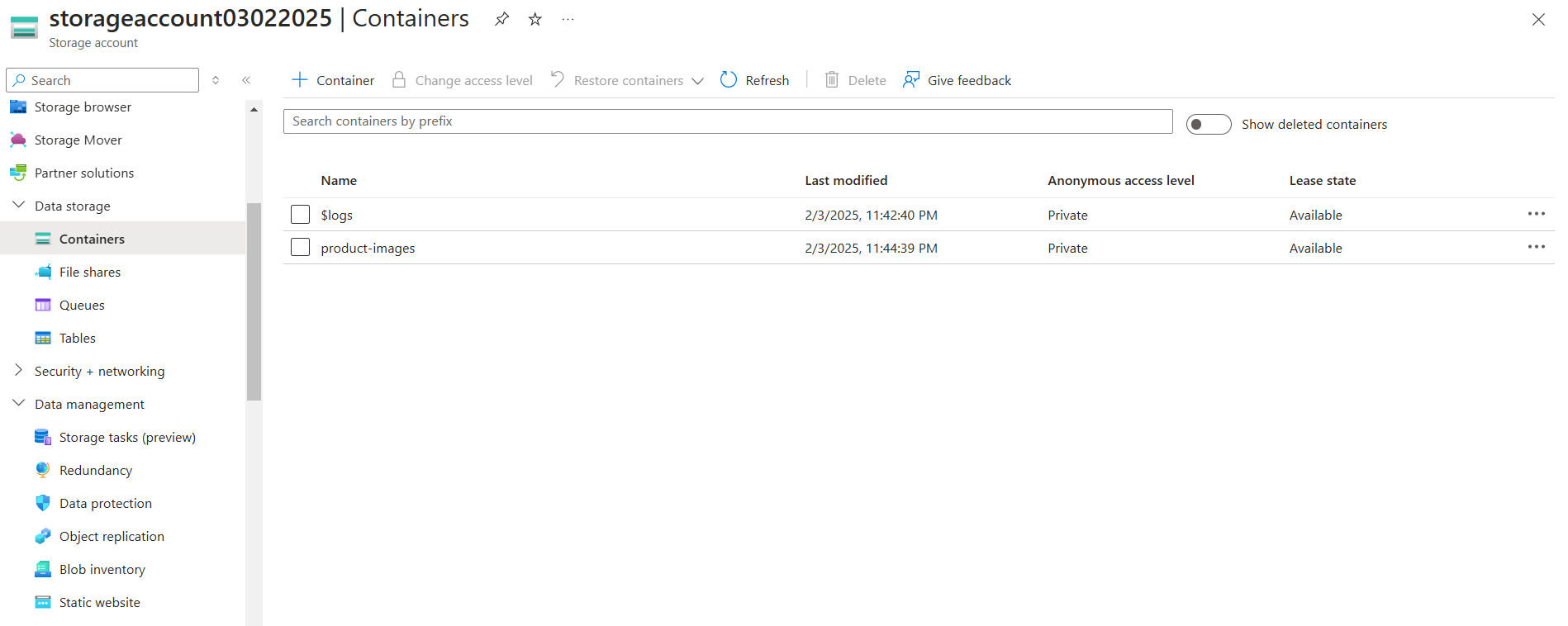


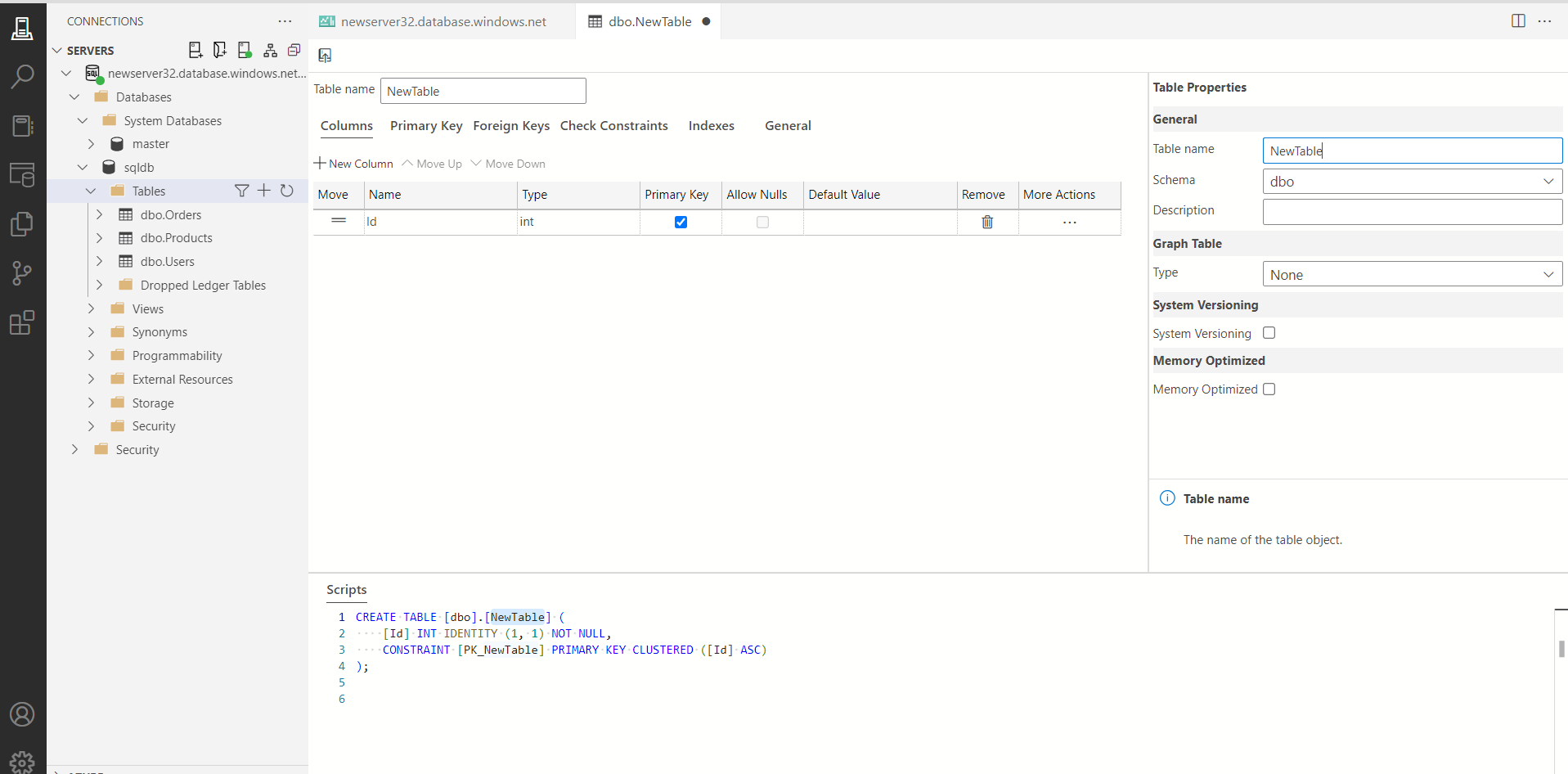


Step 3.3: Create Azure Blob Storage for Images

• In the Azure Portal, search for Storage Accounts and click + Create.

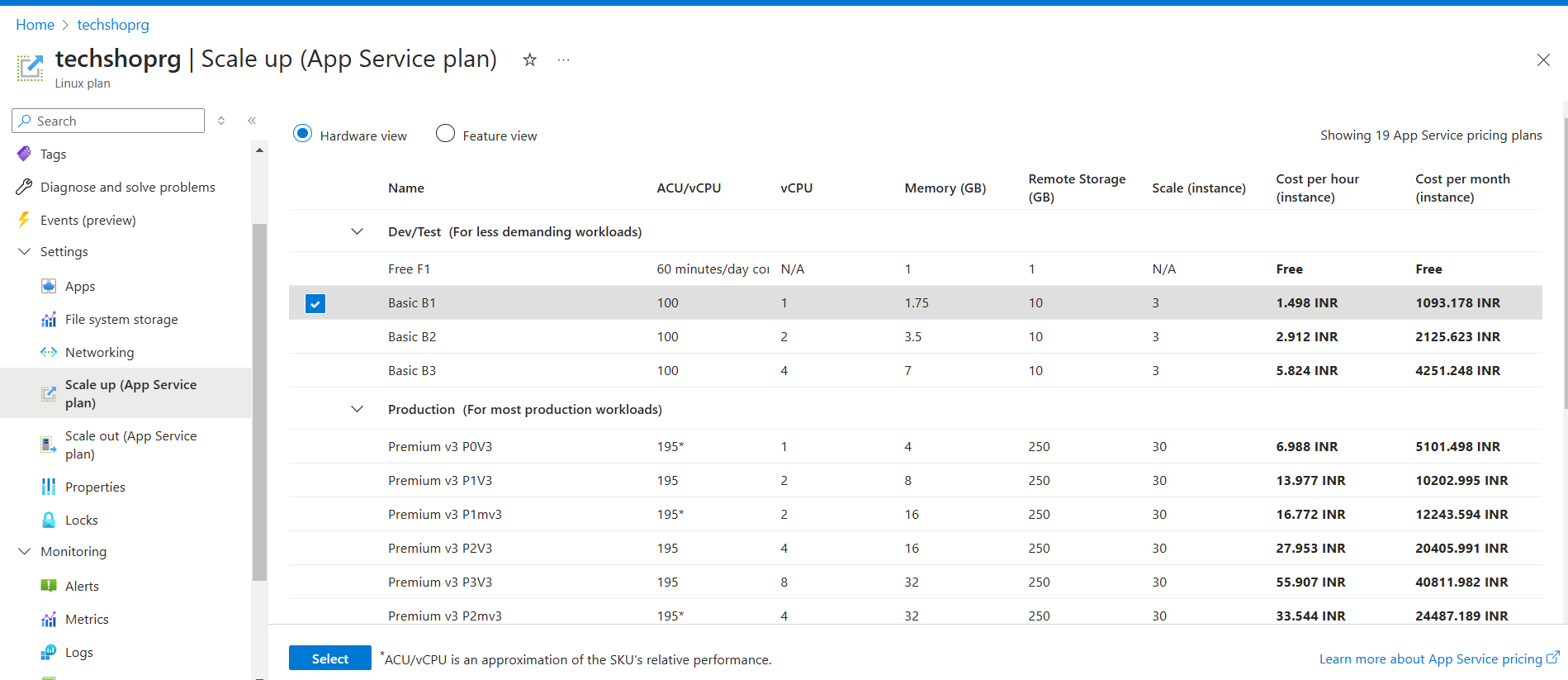
• Select the same resource group TechShopRG and provide a unique storage account name. • After the storage account is created, go to the Containers section and create a container, e.g., product-images.



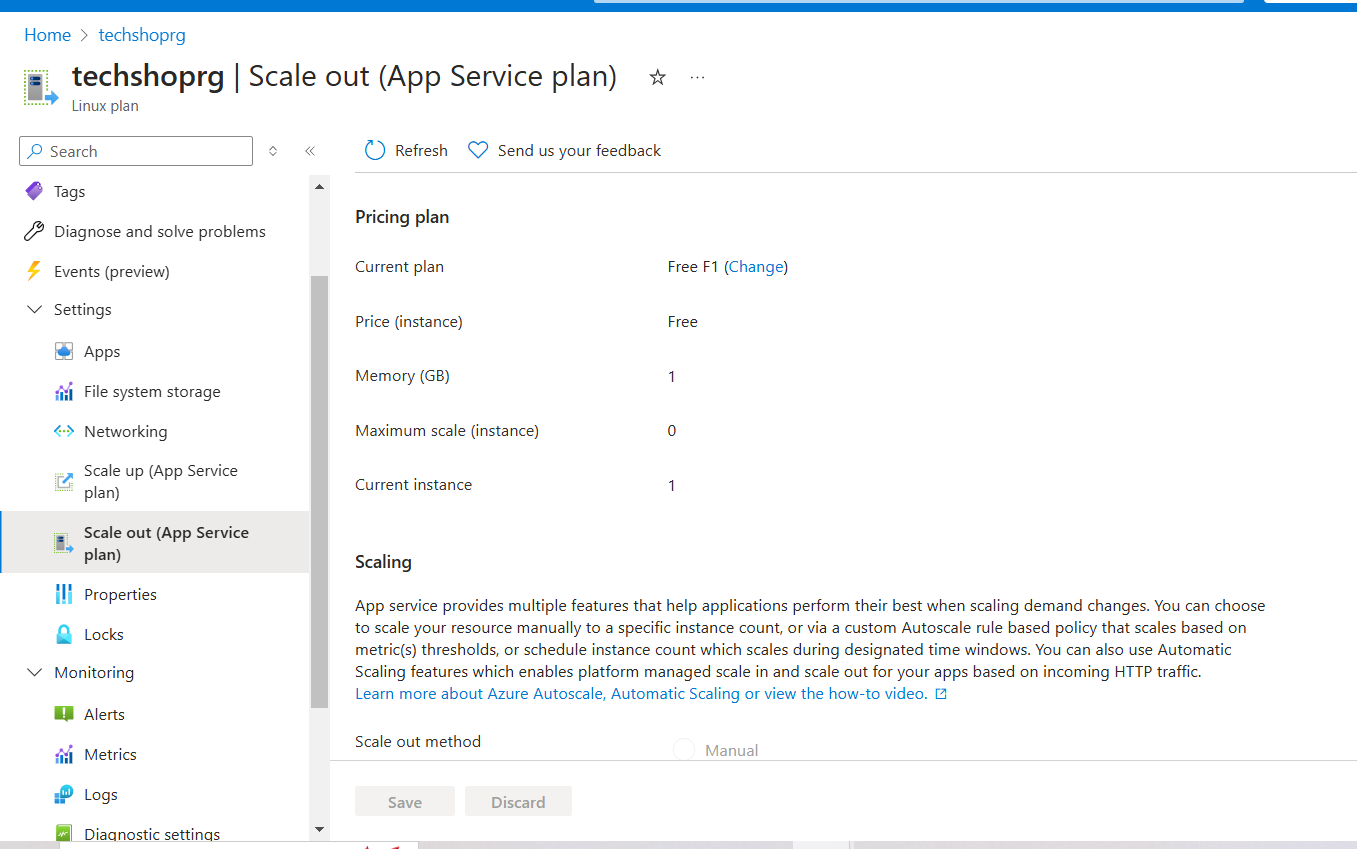


Scalability and Load Balancing

1. Objective: Ensure the application can scale and distribute traffic across multiple instances. Step 4.1: Enable Auto-scaling for Azure App Service
2. • In the App Service settings, go to Scale up (App Service plan) and select the appropriate pricing tier for scaling.



• Go to Scale out (App Service) and enable auto-scaling based on metrics such as CPU usage or memory usage.



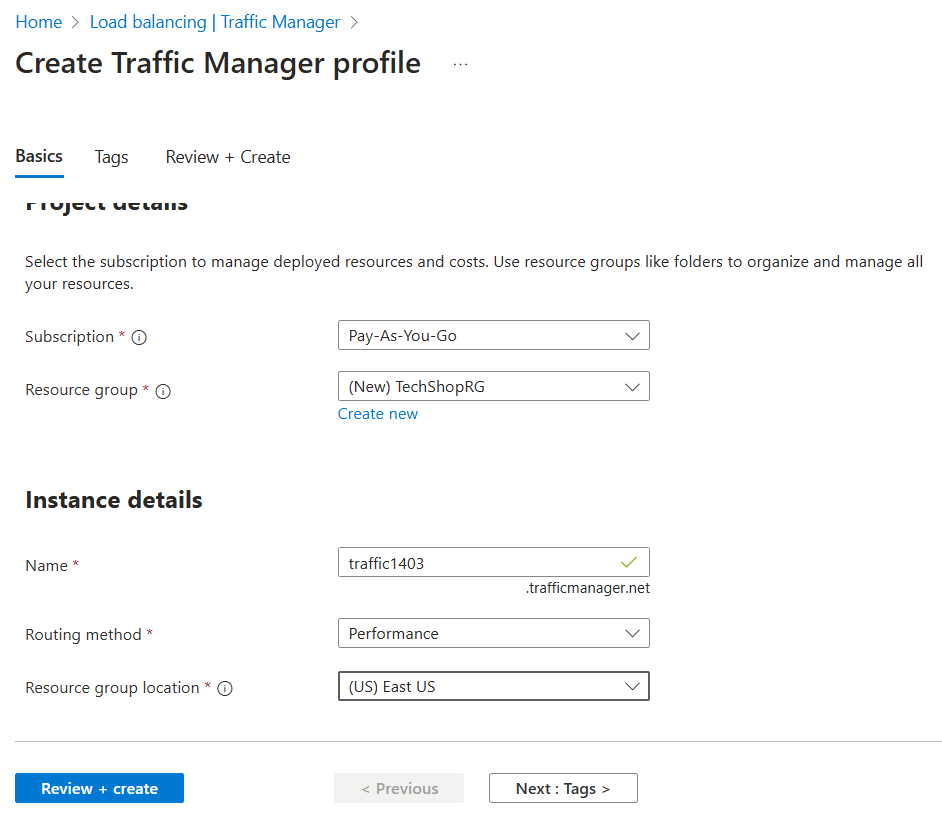
Step 4.2: Configure Azure Traffic Manager

• In the Azure Portal, search for Traffic Manager and click + Create.

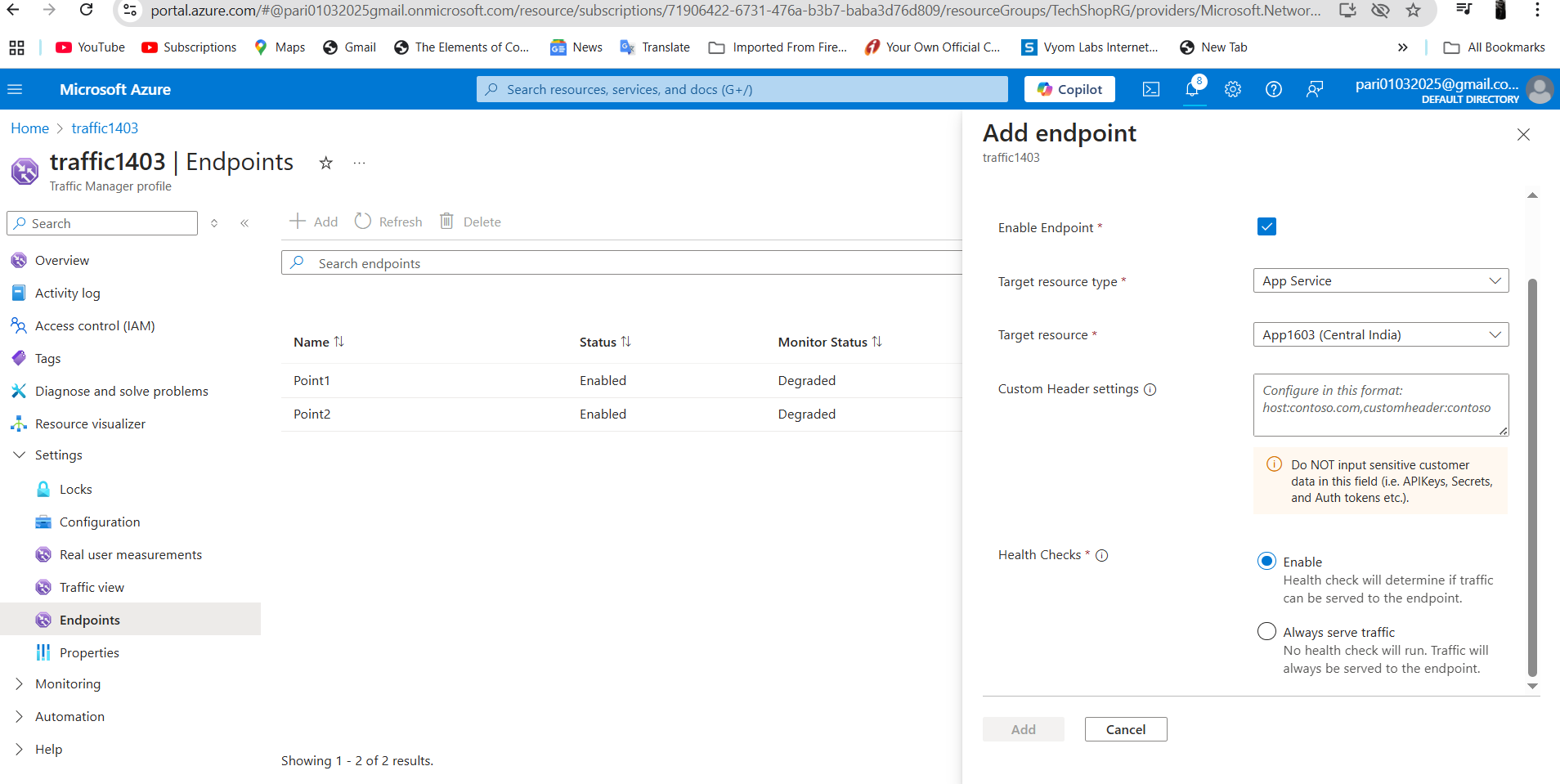
• Choose the routing method (e.g., Performance or Priority).

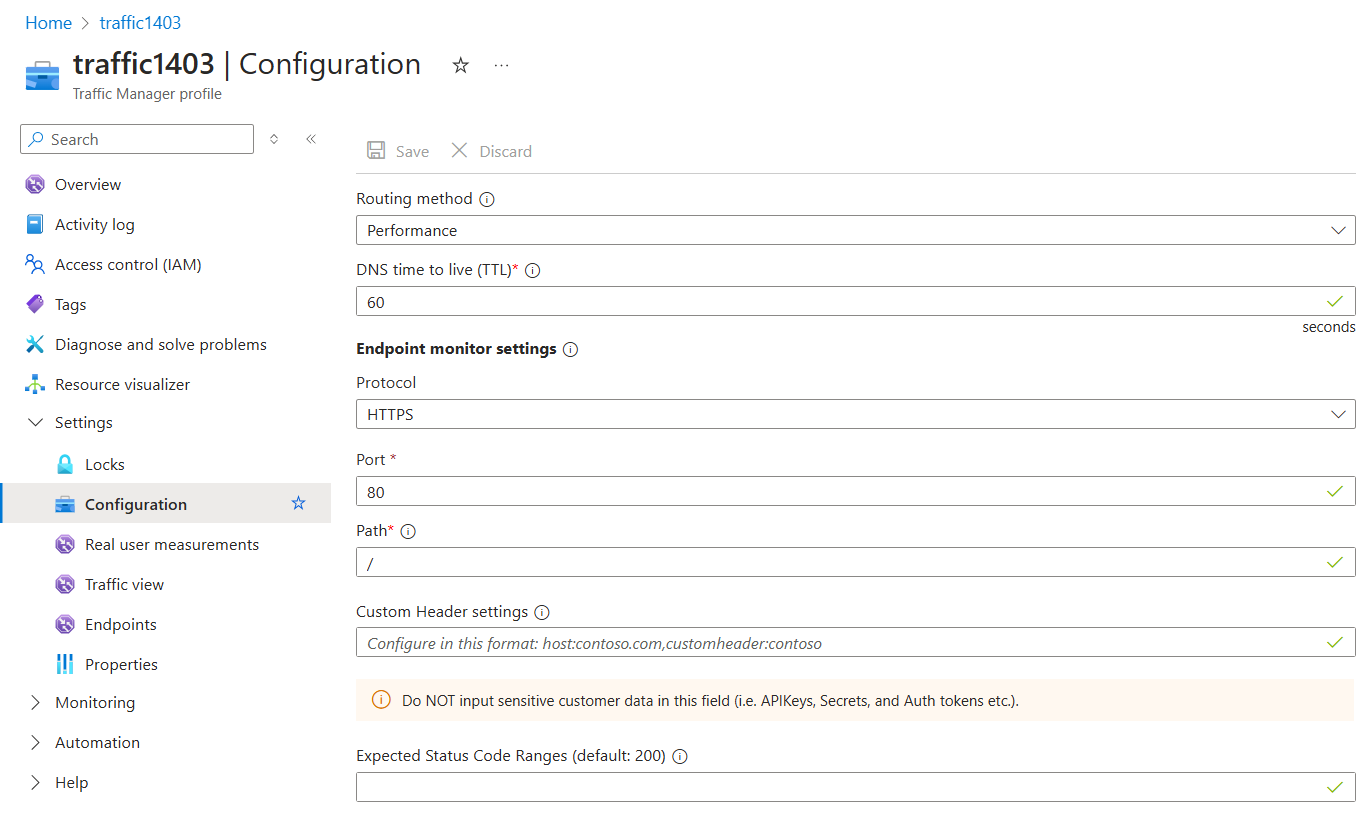
• Add multiple Azure regions (e.g., West US, East US) to distribute the traffic.

• Link the Traffic Manager profile to the App Service instances.



• Link the Traffic Manager profile to the App Service instances.





5. Monitoring and Logging

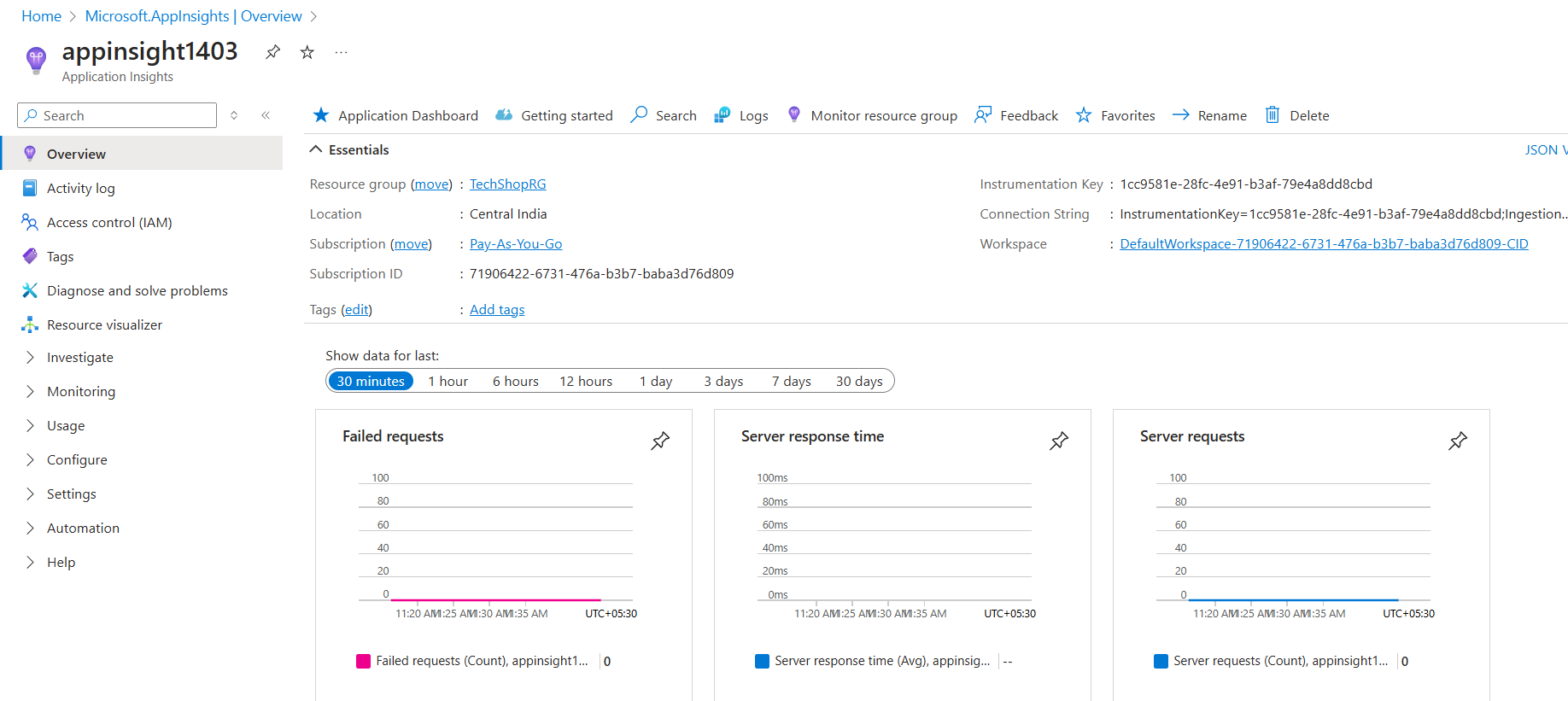
Objective: Set up monitoring and logging for the application.

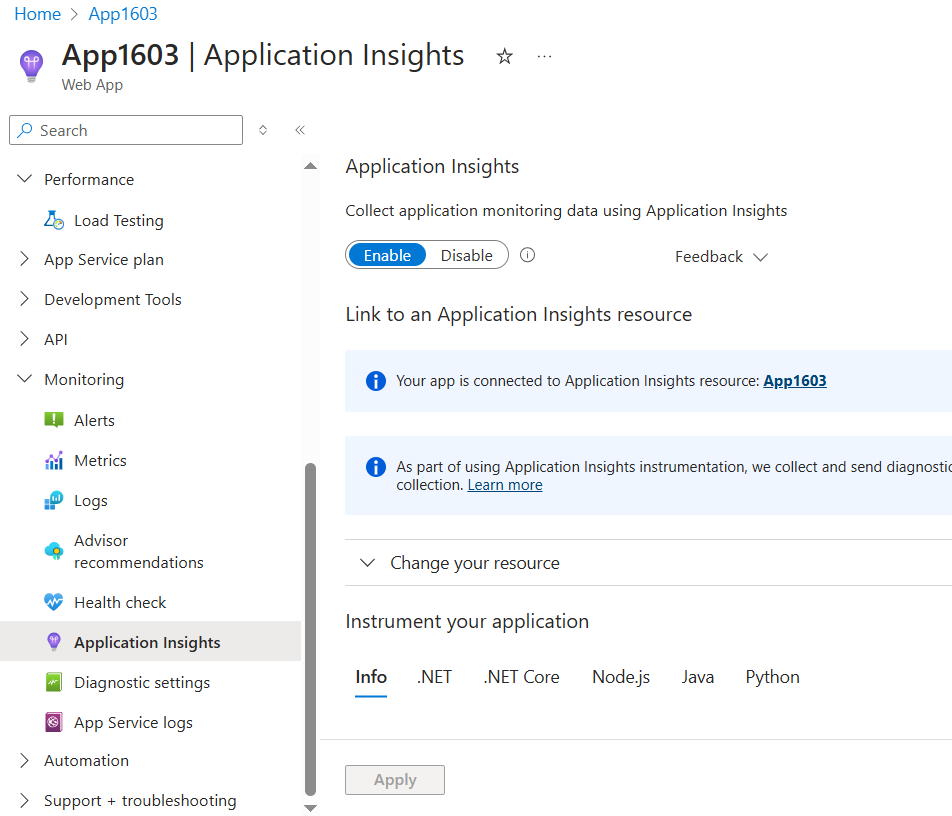
Step 5.1: Enable Application Insights

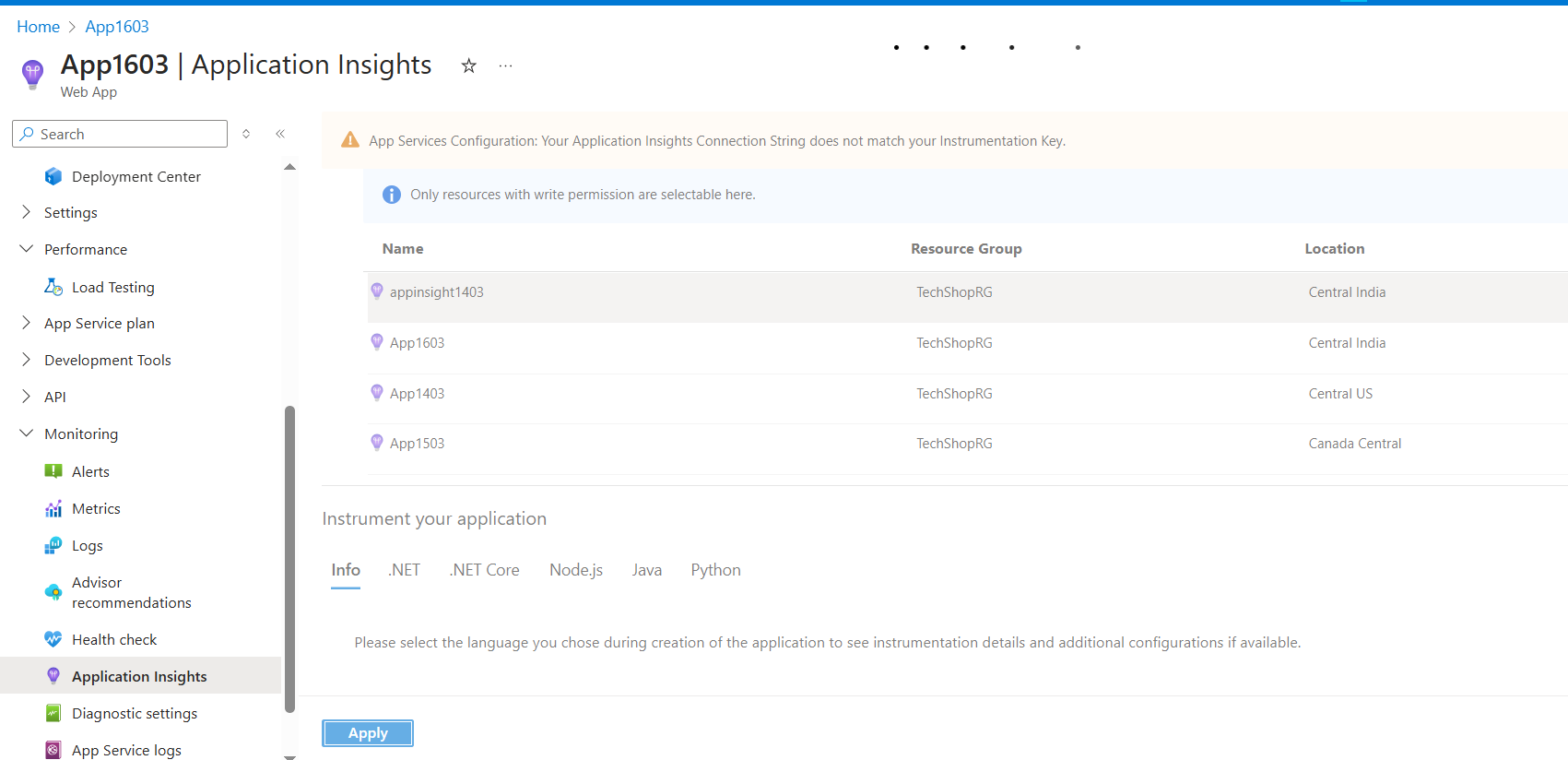
• In the App Service settings, go to Application Insights and click Turn on Application Insights.

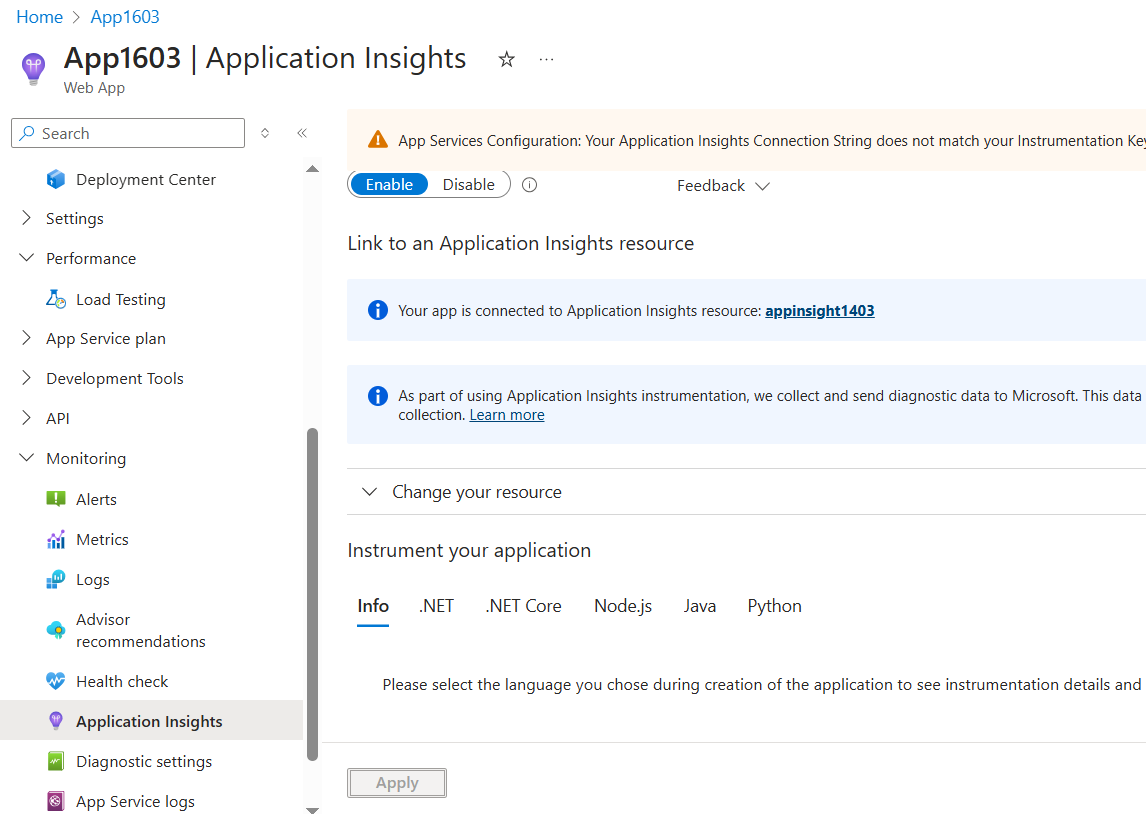
• Choose an existing Application Insights resource or create a new one.

• After enabling it, you can track requests, failures, performance, and more.

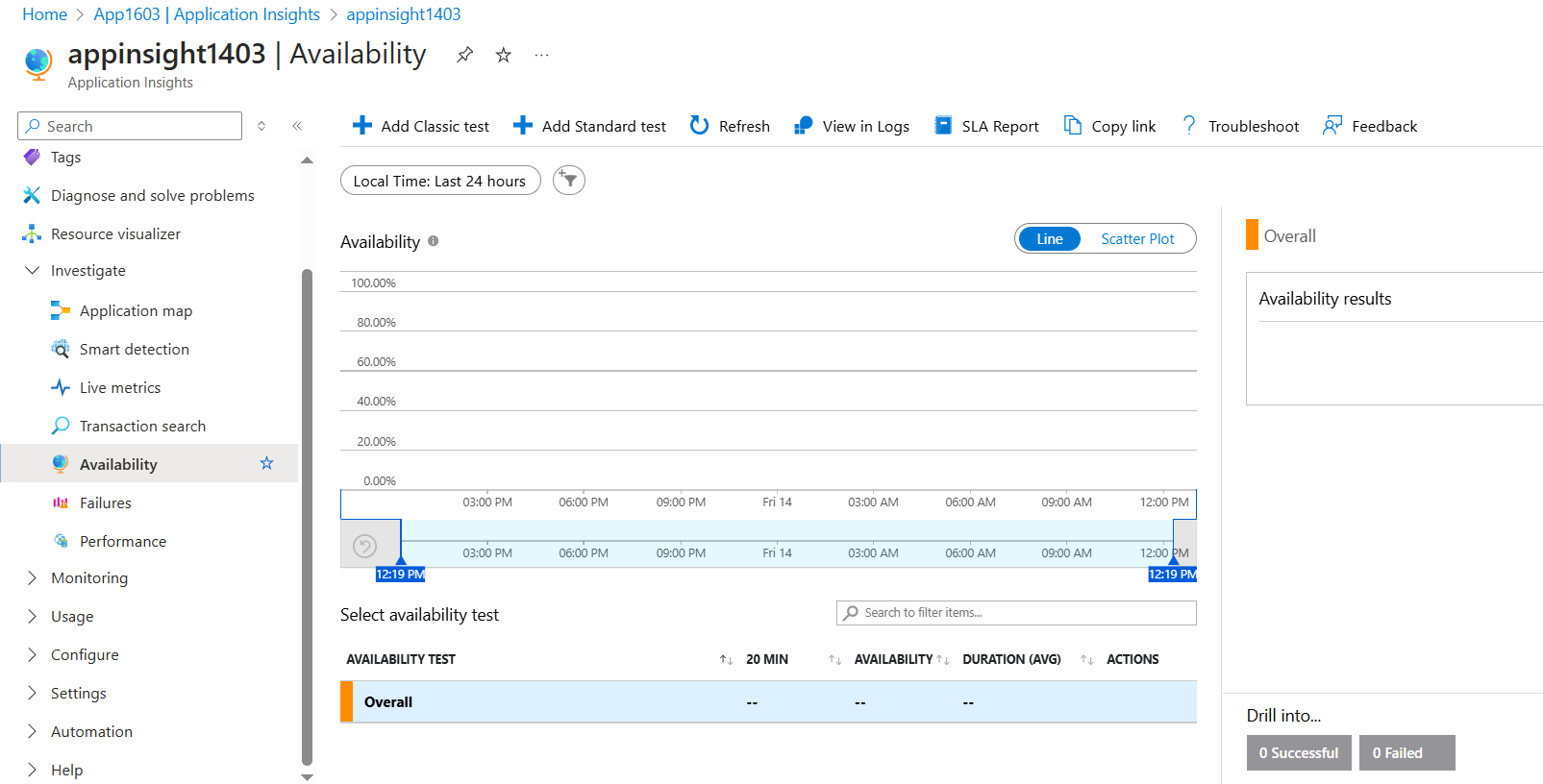










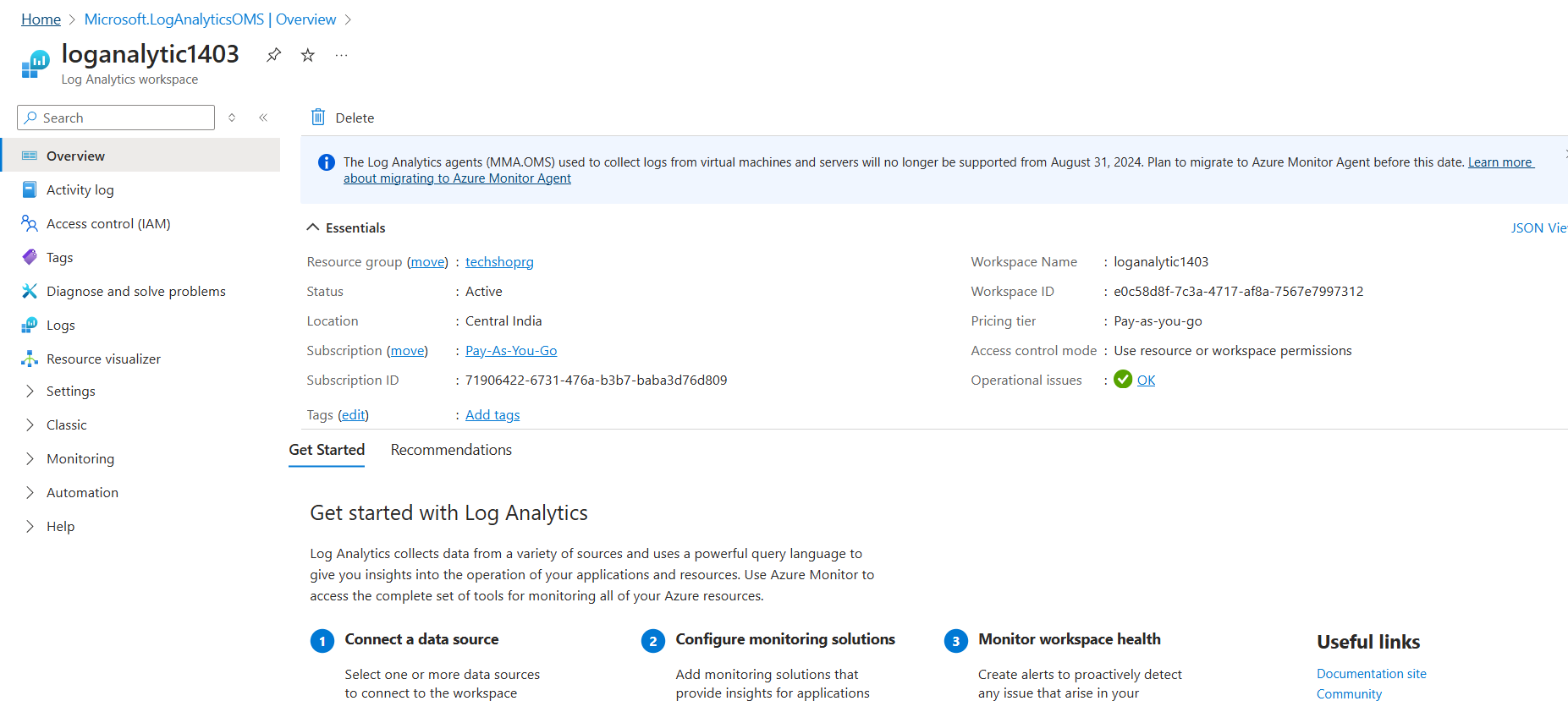


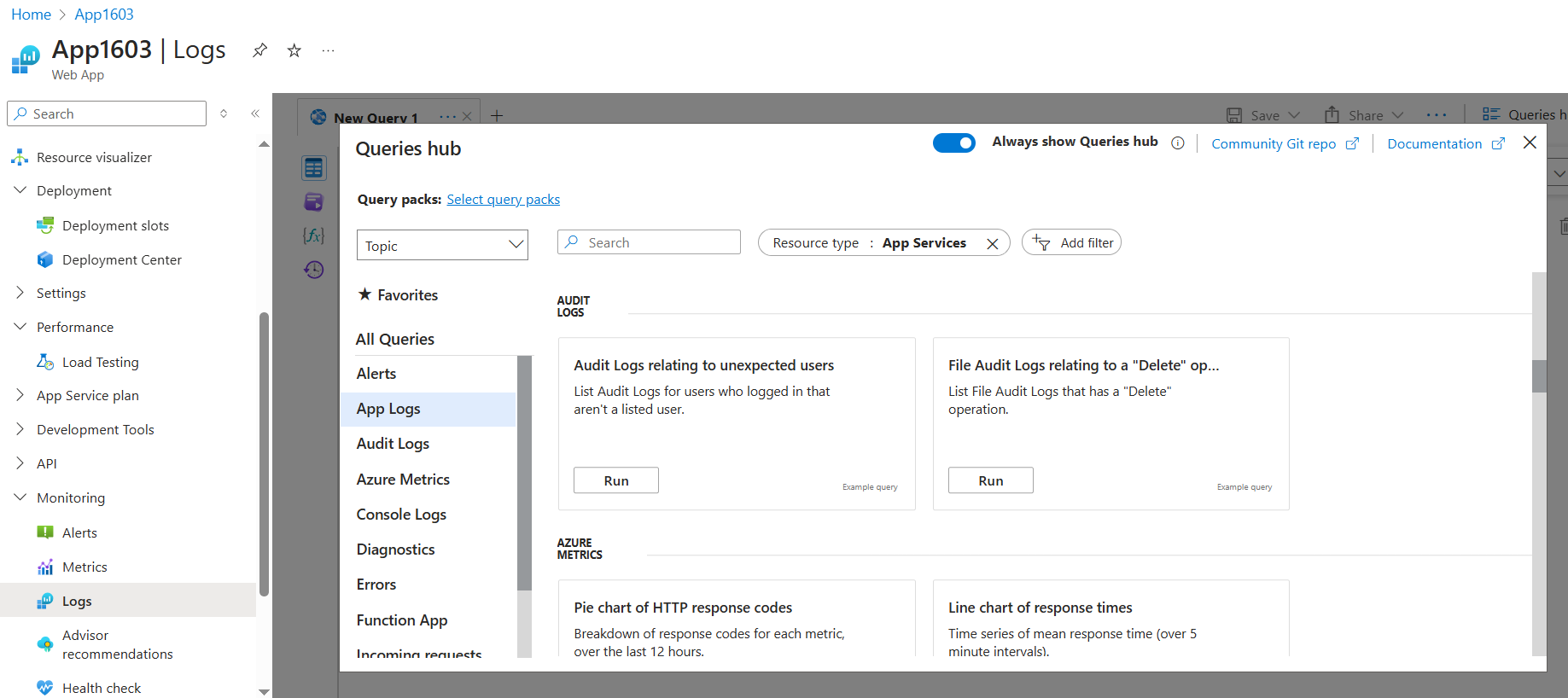
Step 5.2: Set up Log Analytics for Detailed Monitoring

• Go to Log Analytics workspaces and create a new workspace.

• Configure Azure Monitor to send logs from App Services and Azure Functions to this workspace.

• Use Azure Log Analytics to query logs, such as product views, order creations, and user activities.





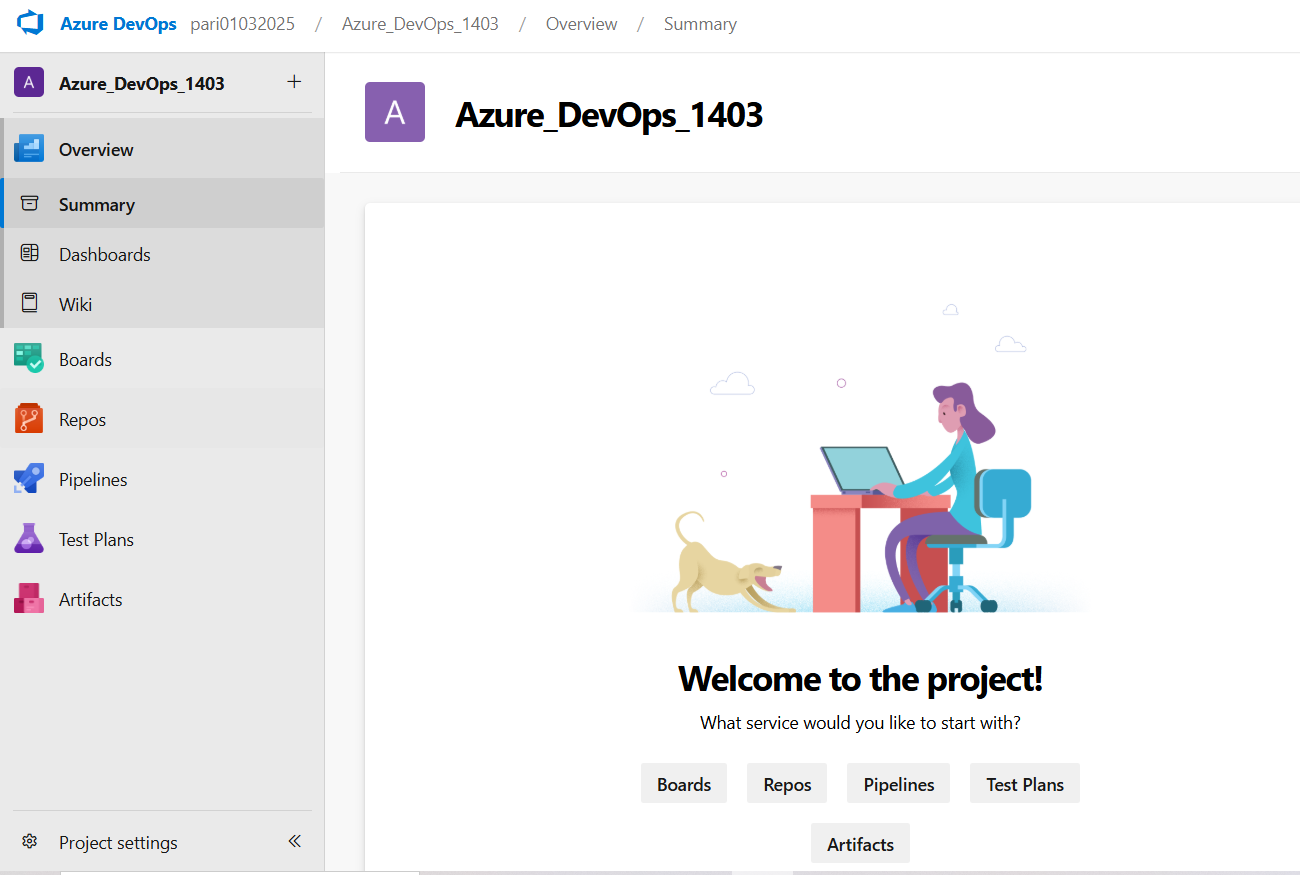
1. Security Configuration
2. Objective: Ensure secure communication and storage of sensitive data.
3. Step 6.1: Implement Azure Active Directory Authentication
4. • In the App Service settings, go to Authentication/Authorization and enable Azure Active Directory (Azure AD).
5. • Configure the authentication provider to use Azure AD for user login.
6. • Optionally, configure OAuth for external logins (e.g., Google, Facebook).

7. CI/CD Pipeline using Azure DevOps

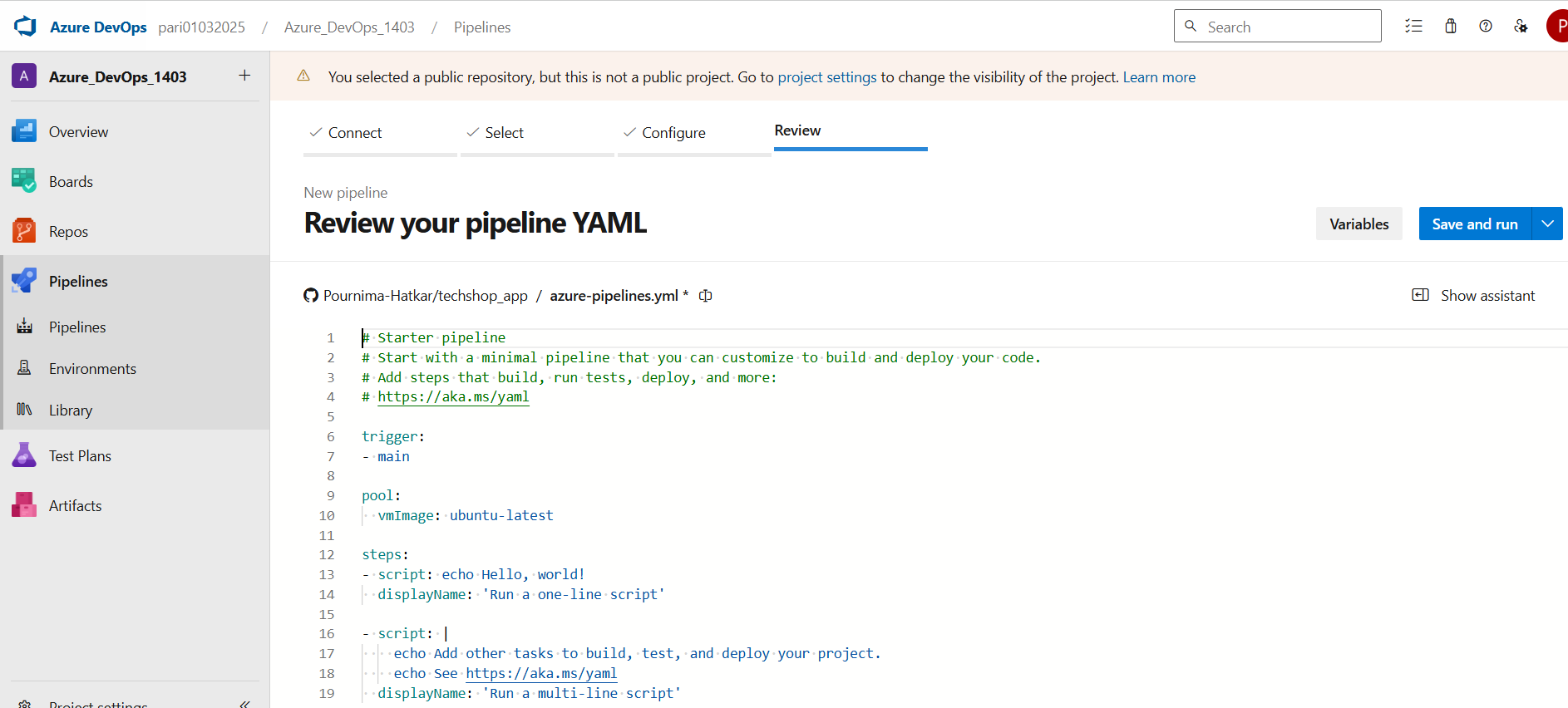
Objective: Automate the build and deployment process.

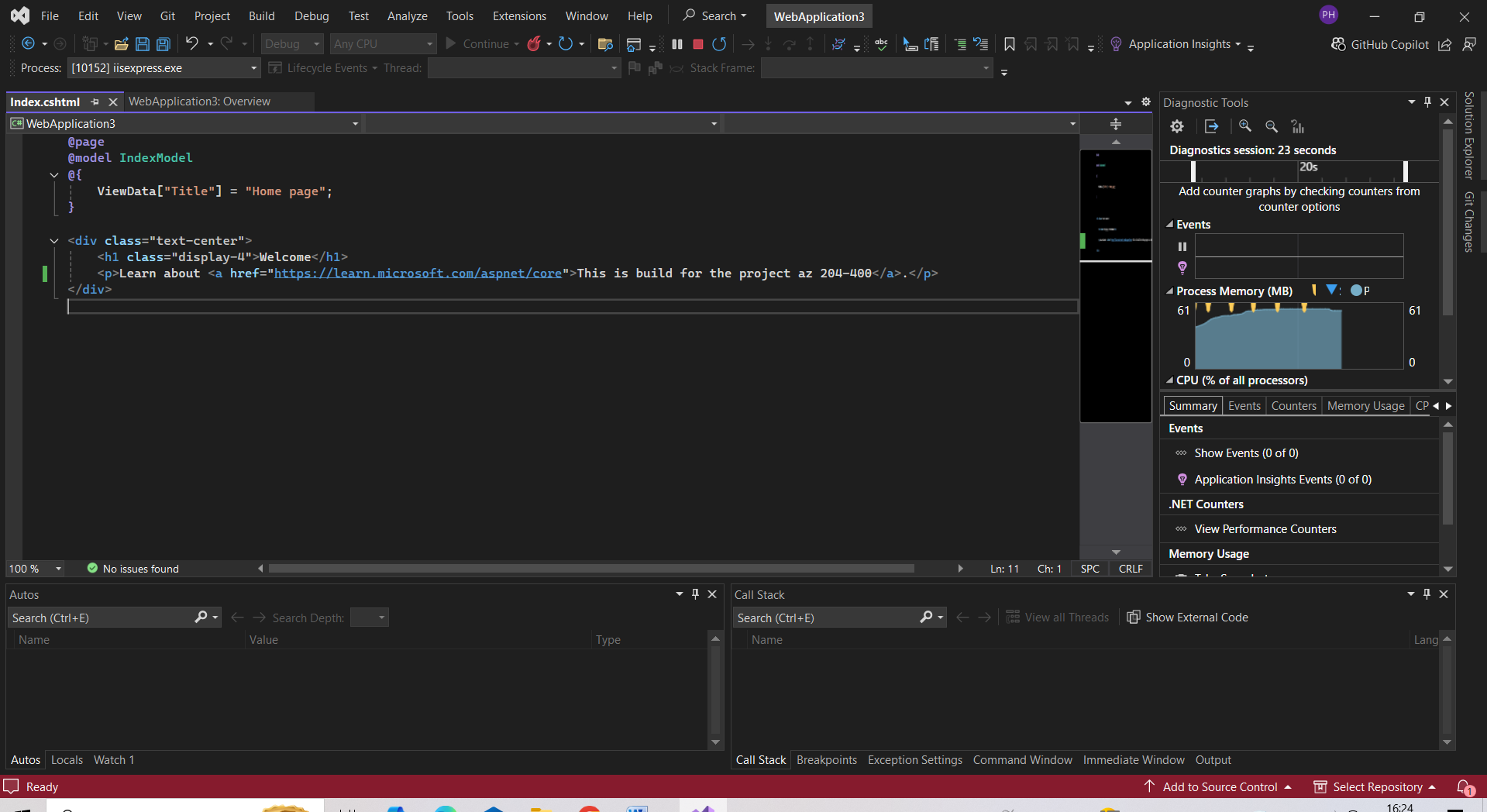
Step 7.1: Set up Azure DevOps Project

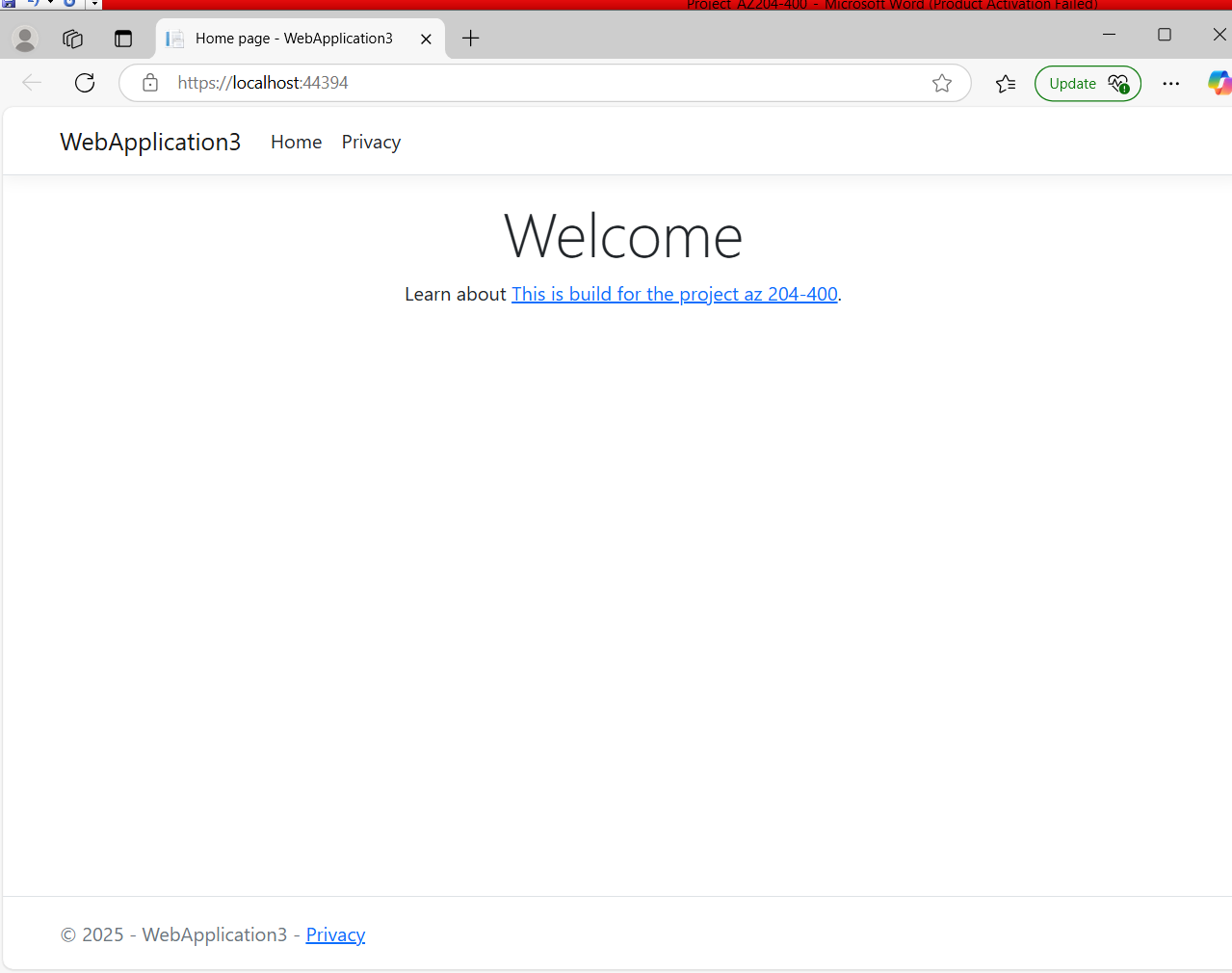
• Create a new Azure DevOps Project.

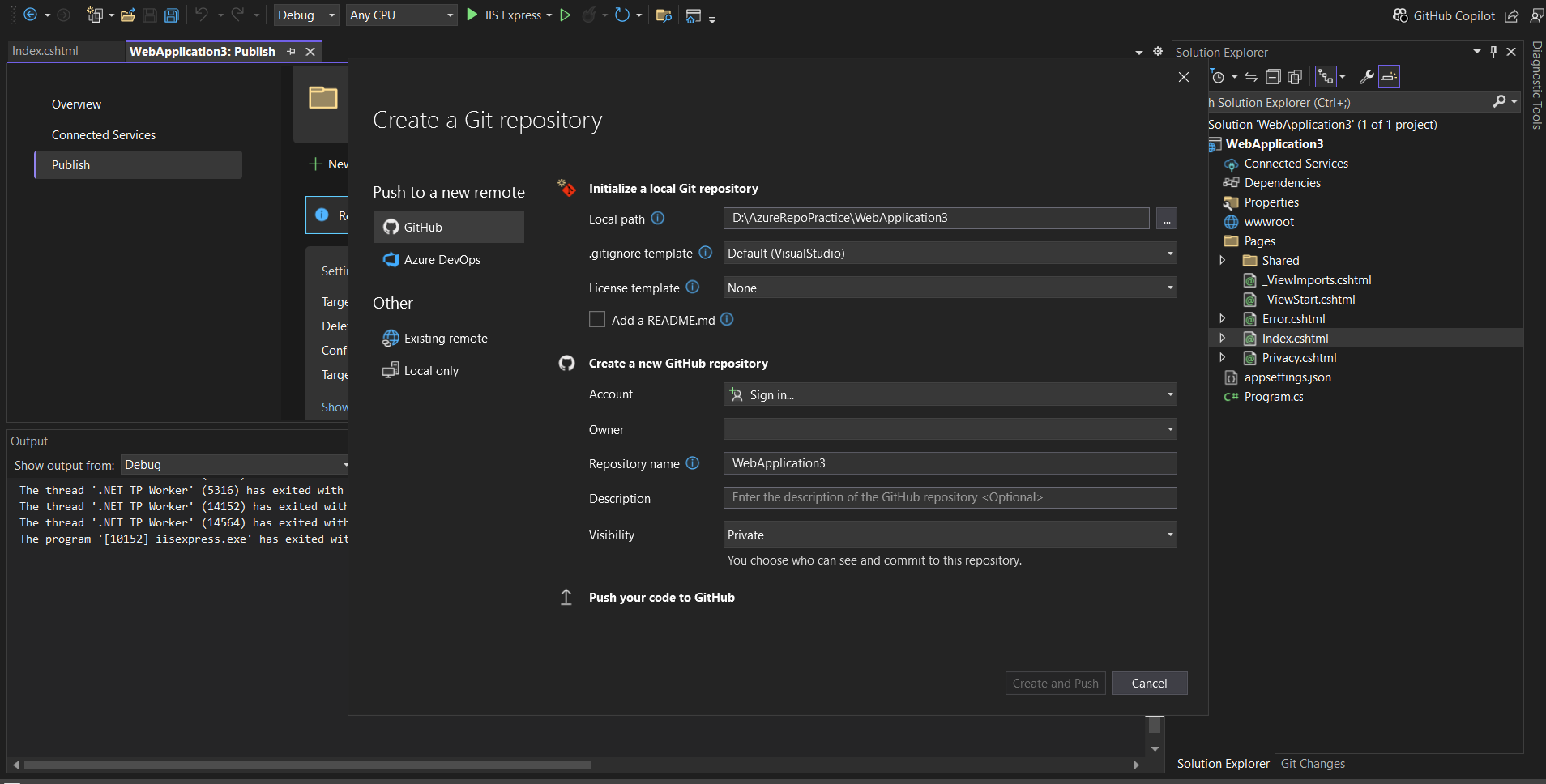


• Link your GitHub or Azure Repos repository to Azure DevOps.







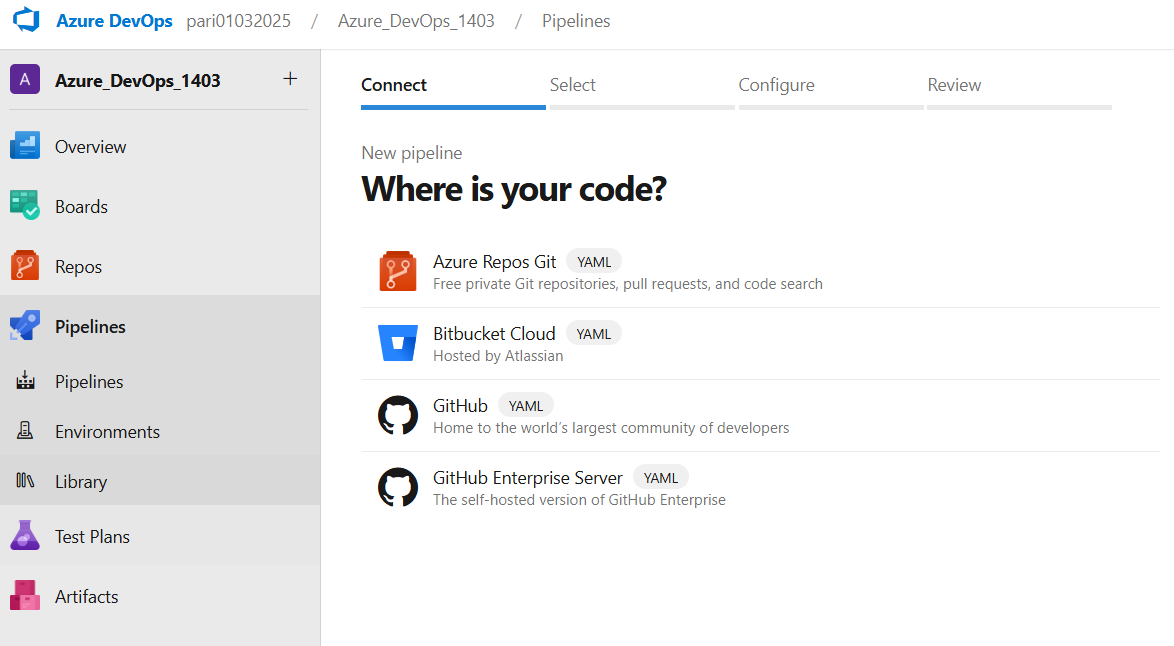


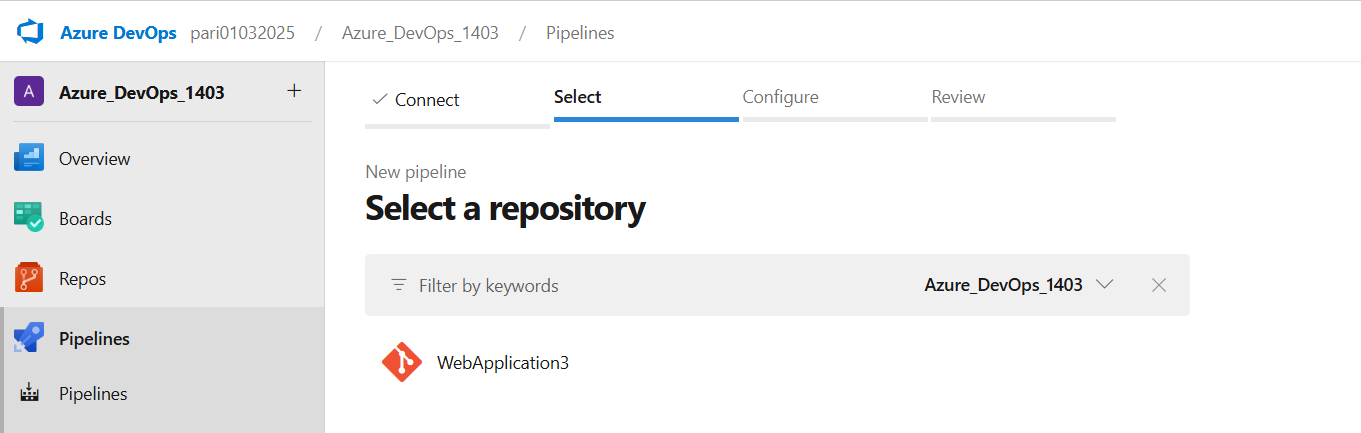
Step 7.2: Create a Build Pipeline

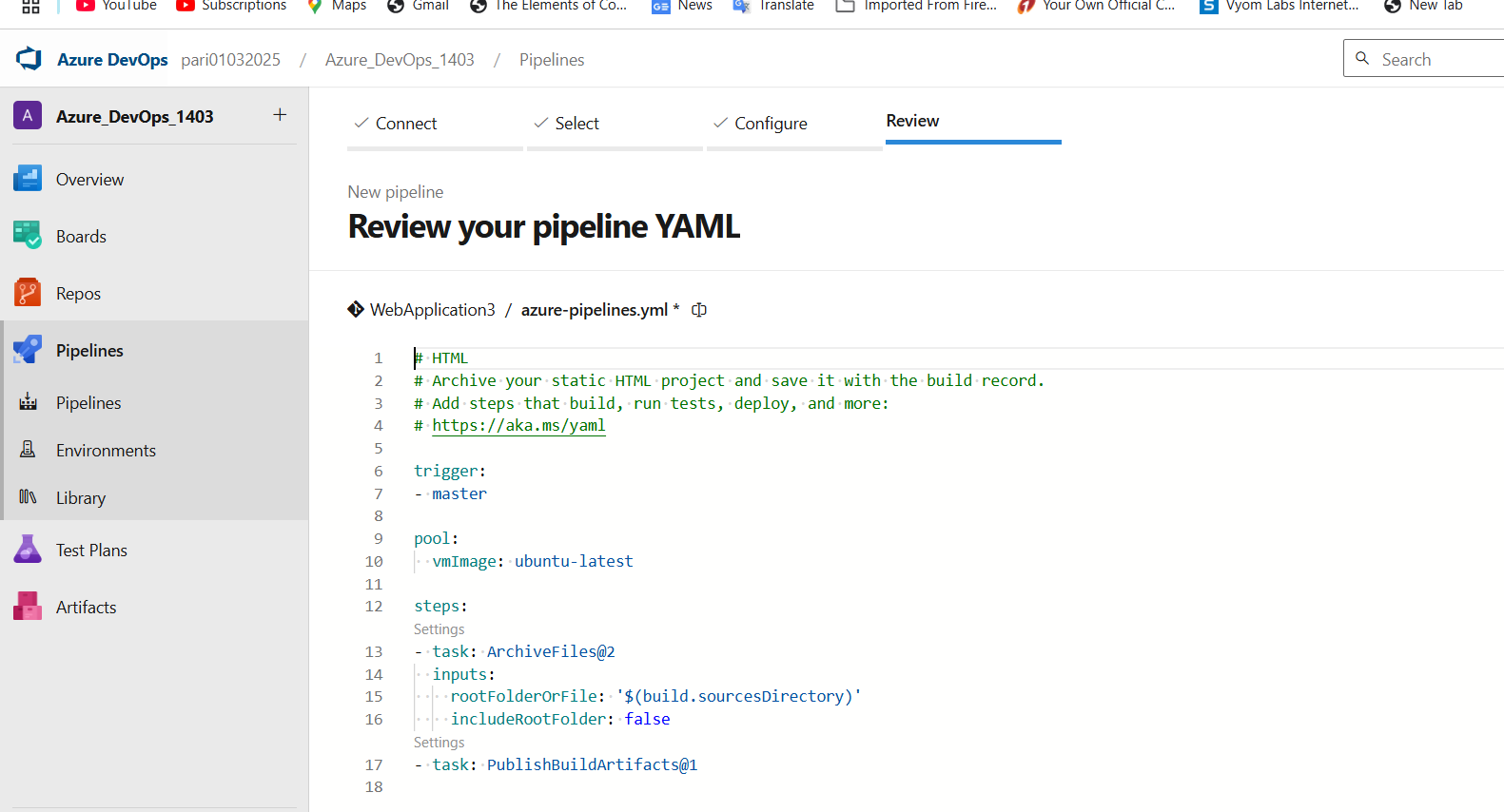
• In Azure DevOps, go to Pipelines > Builds and click New Pipeline.

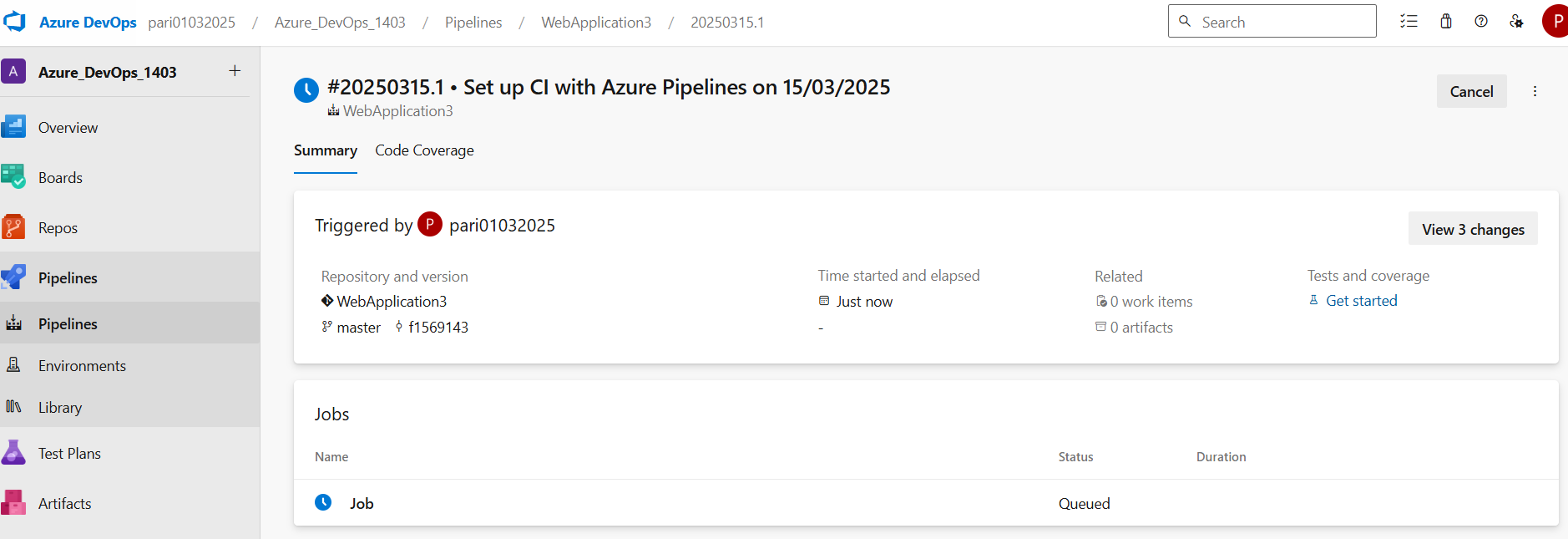
• Choose your repository and select the build template (e.g., Node.js or .NET Core).

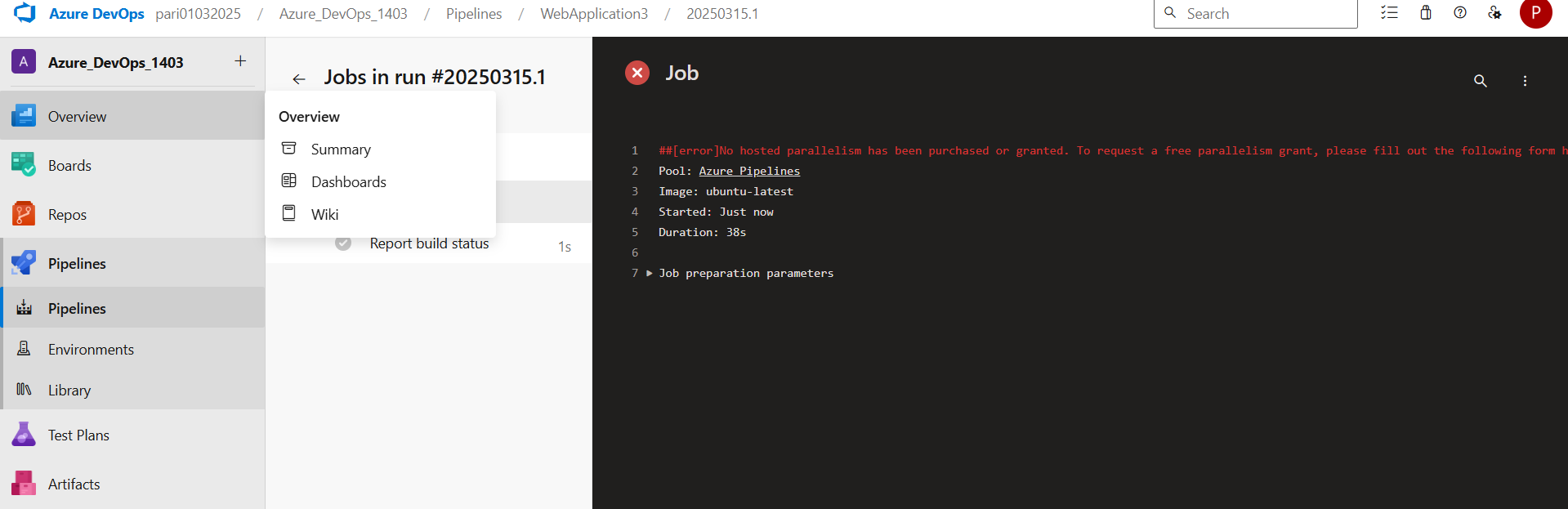
• Configure tasks to build and test the application.











Step 7.3: Create a Release Pipeline

• Go to Pipelines > Releases and create a new release pipeline.

• Add a Stage for deploying to Azure App Service.

For enabling the release pipelines options-

