DOTNET CORE AZURE
FINAL PROJECT

Create a **Web API Project** to store Product Information. Use Entity Framework to store the product information in the database. The user should be able to perform all the CRUD Operations. Configure **GET, POST, PUT and DELETE**.

The Product Entity should have the following properties:

- Product ID
- ProductName
- Price
- Brand
- Manufacture Date
- Expiration Date

UseDataAnnotations:

- Mark the Primary Key
- Make ProductName Mandatory
- Make Price a Number

Create a jQuery and AJAX Client to consume the Web API and show the result.

Azure Hosting:

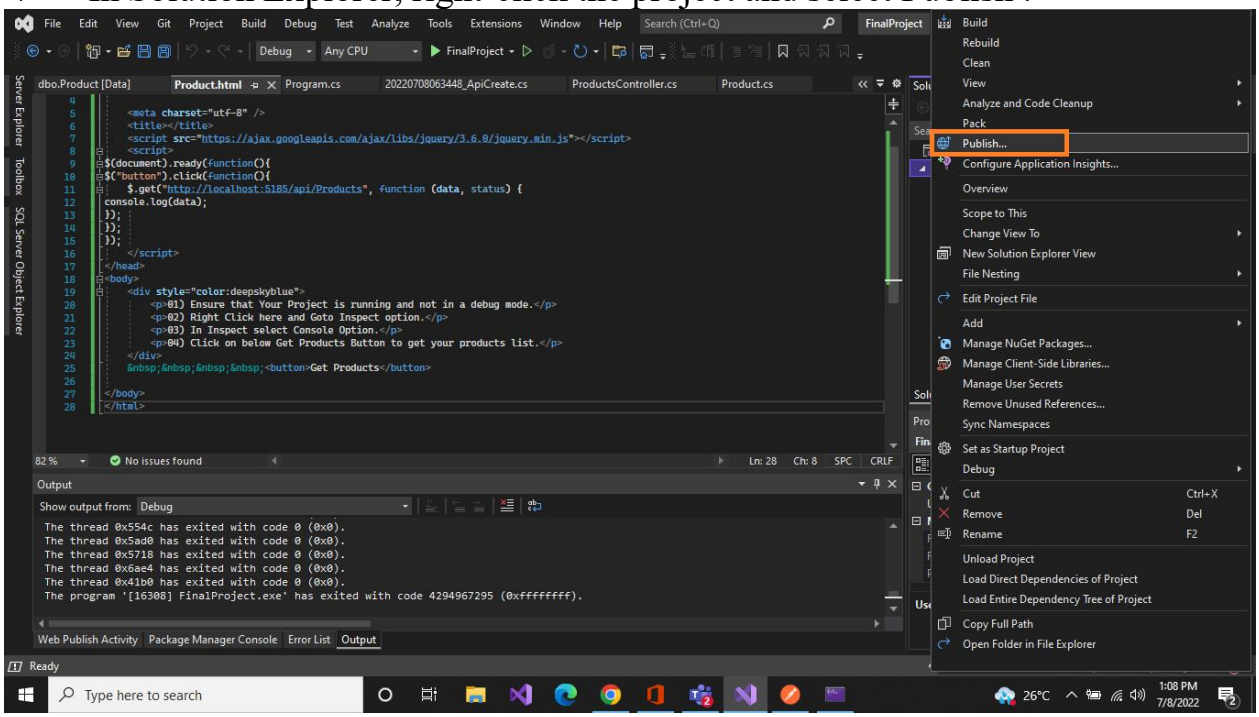
- Host the web api in azure and consume the same using jQuery Client.
- Configure Scale out by adding rules for custom scaling
- Configure Deployment slots for staging and production
- Configure Application Insights for the project
- Configure Swagger for the api
- Work with Log Analytics with the sample logs available

CREATING WEB API PROJECT : -

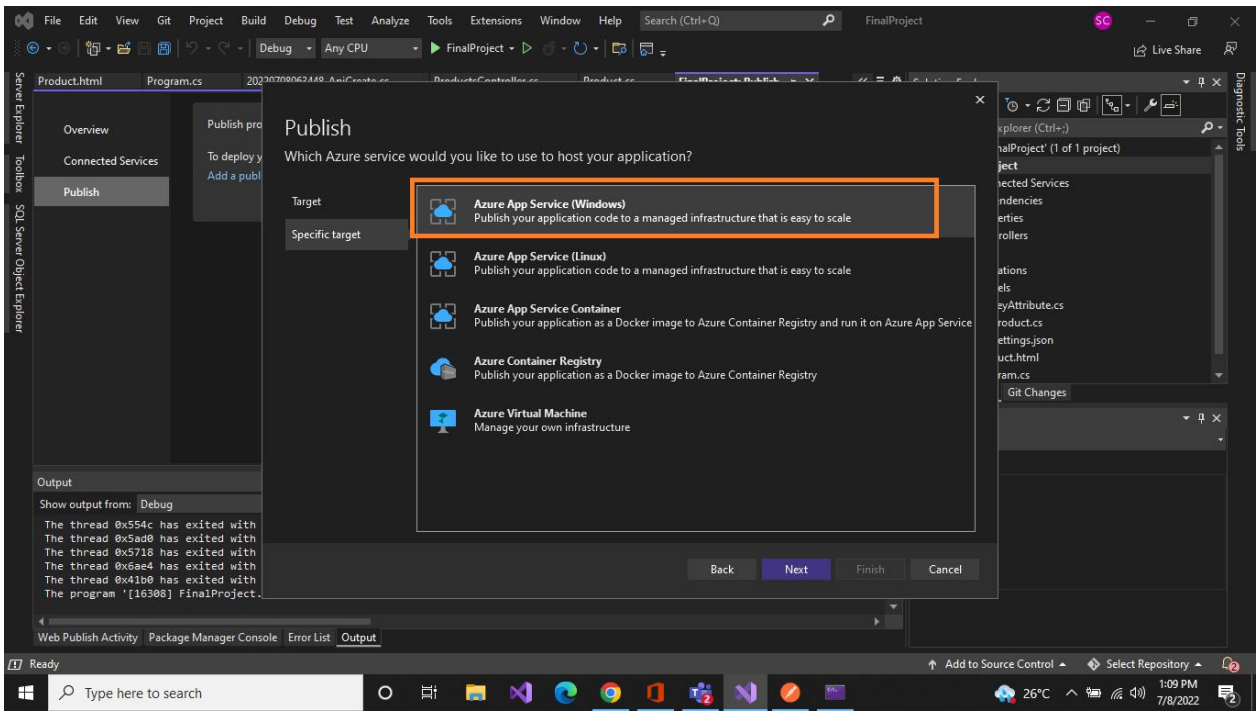
1. Start with a new project on visual studo, with project type as **ASP.NET Core Web API**.
2. Create a new **Folder** called **Models** with class **Products** with the given specifications, added all the **validation attributes** on to the fields of the Product class.
3. Added required **entity framework** libraries using **nuget package** manager.
4. Performed data base **migration** operations on PS console.
5. Created a **controller** by right clicking on Controller and selecting the option Conroller, **create a controller** with the created model class by using **scaffolidng** feature of visual studio.
6. **Tested** the **API** with the default end point weather forecast by publishing it into the **Azure App Services**.
7. The **jQuery AJAX** calls (REST Client) or **Index.html** page is served on the same host, in the folder on Server at the path **Static Files/index.html**. The html page is designed to perform all the **CRUD operations** on the Created WEB API.

1. Host the web api in azure and consume the same using jQuery Client.

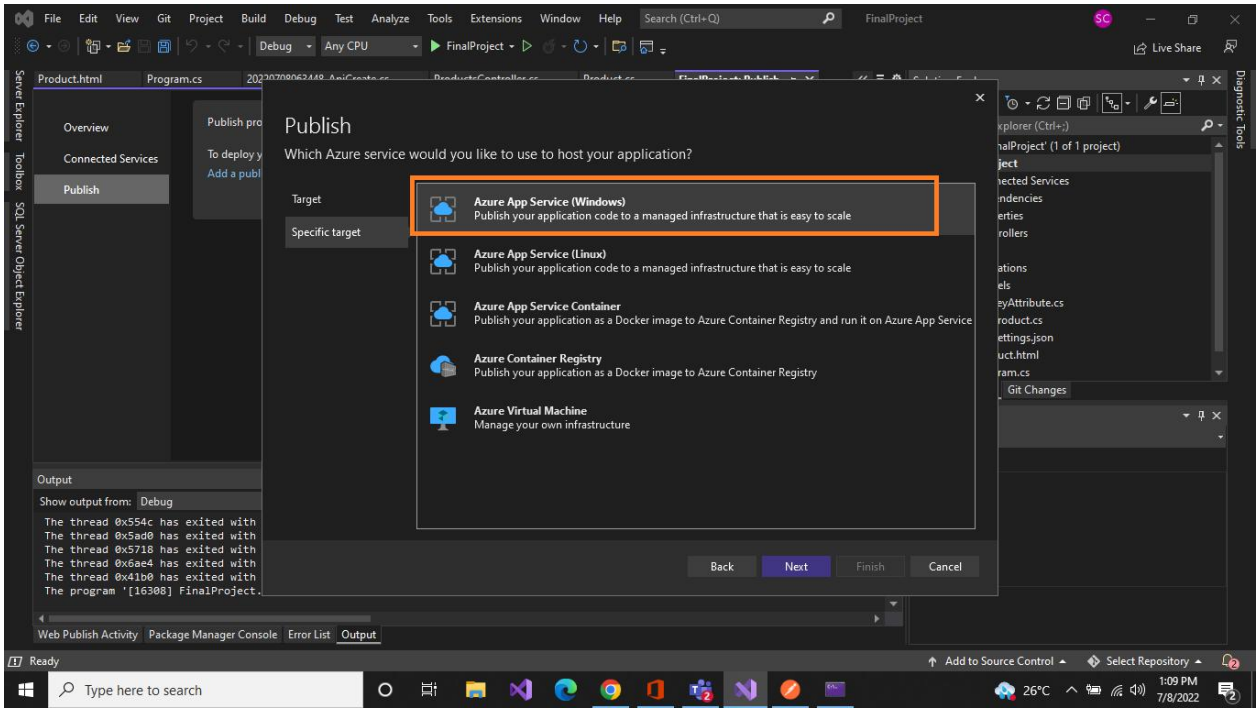
✧ In Solution Explorer, right-click the project and select Publish .



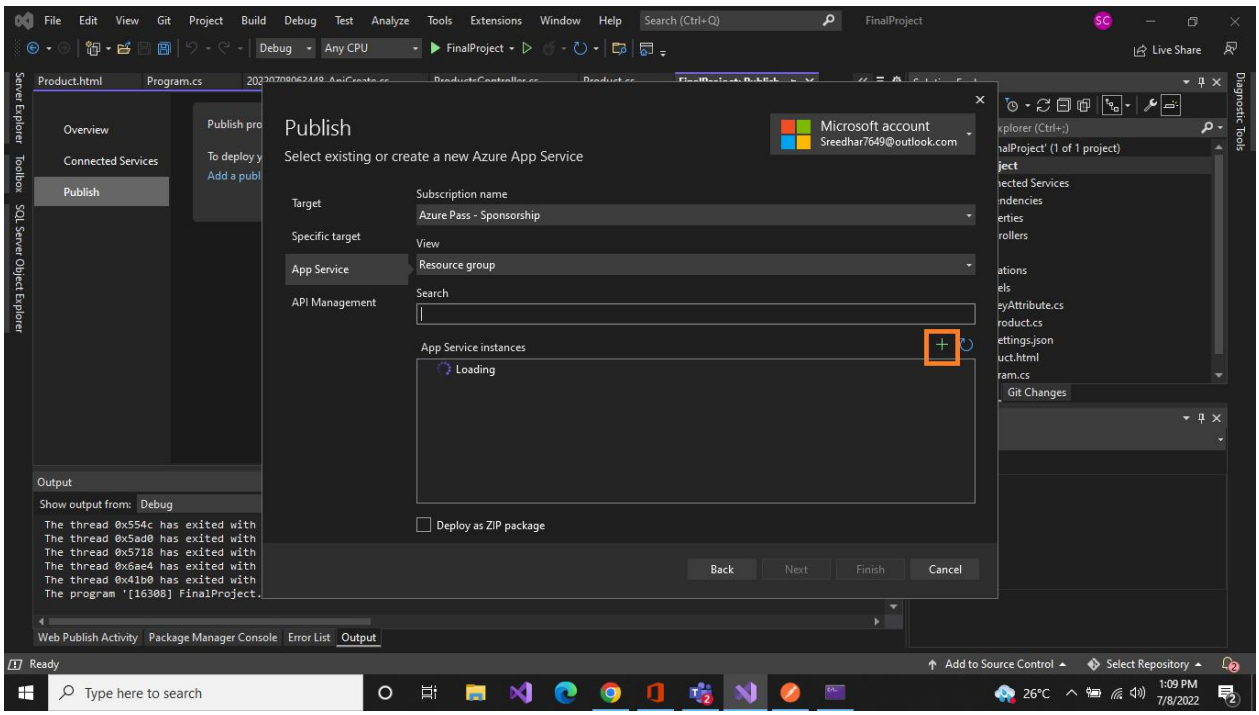
✧ In the Publish dialog, select Azure and select the Next button.



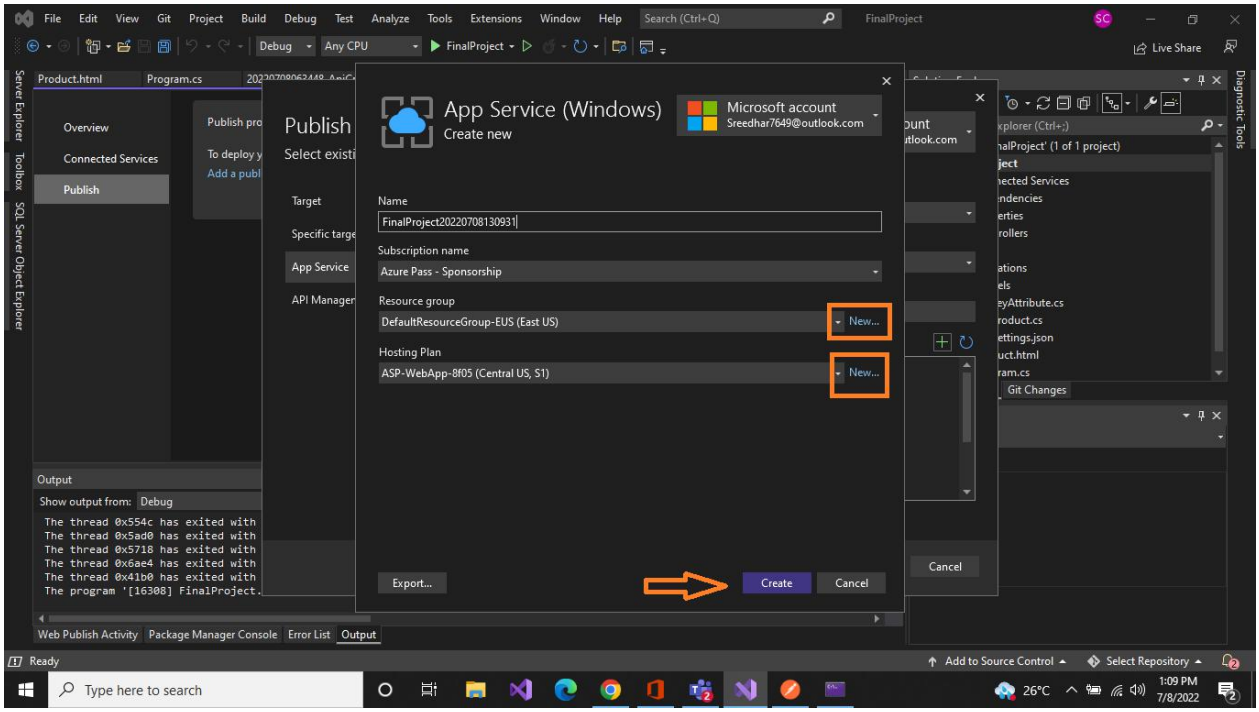
✧ Select Azure App Service (Windows) and select the Next button .



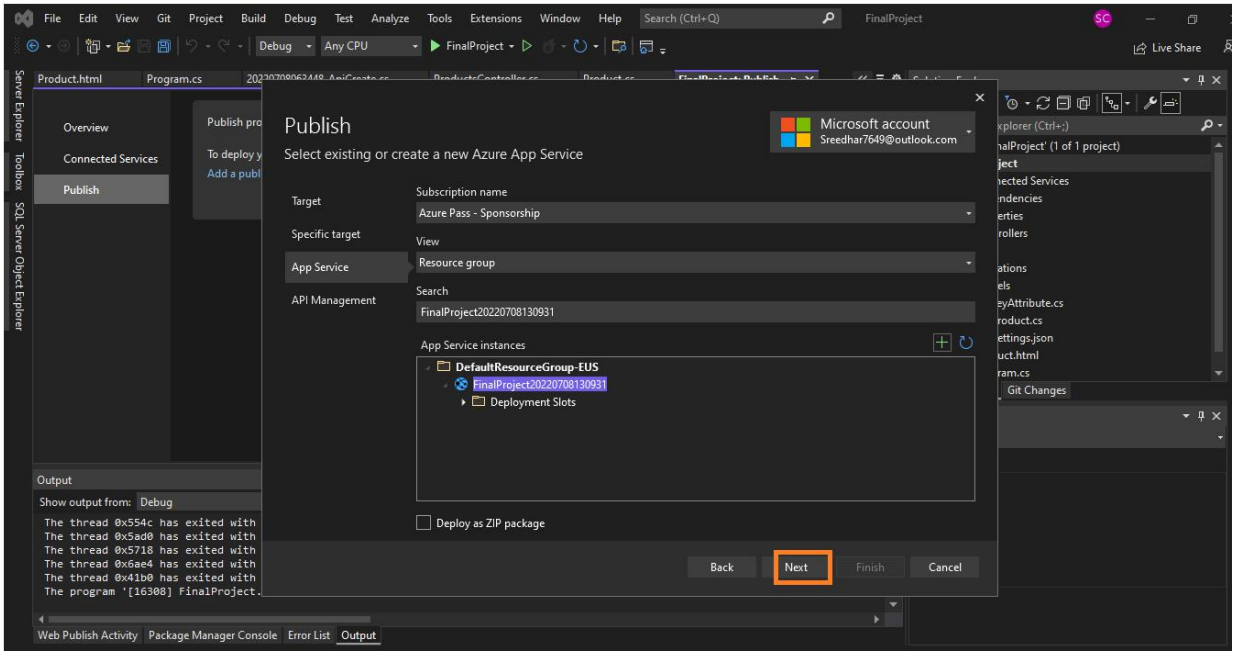
✧ Select Create a new Azure App Service.



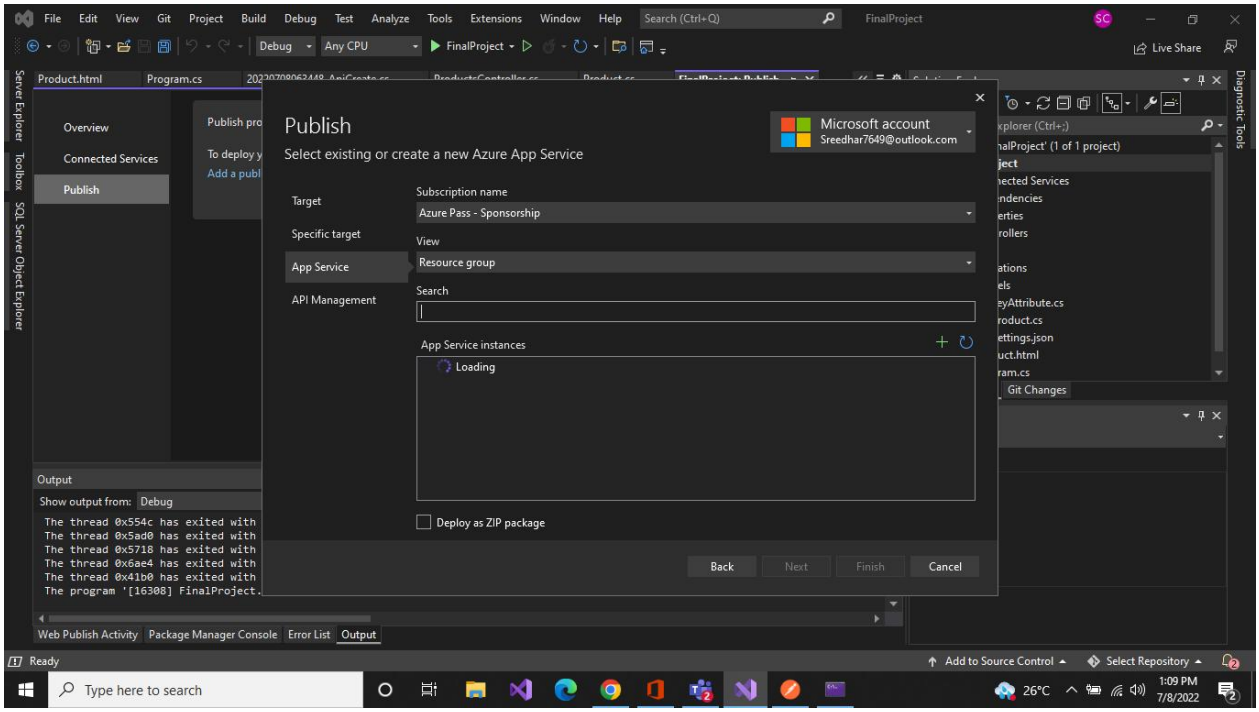
✧ The Create App Service dialog appears. The App Name, Resource Group, and App Service Plan entry fields are populated. You can keep these names or changethem. Select the Create button.



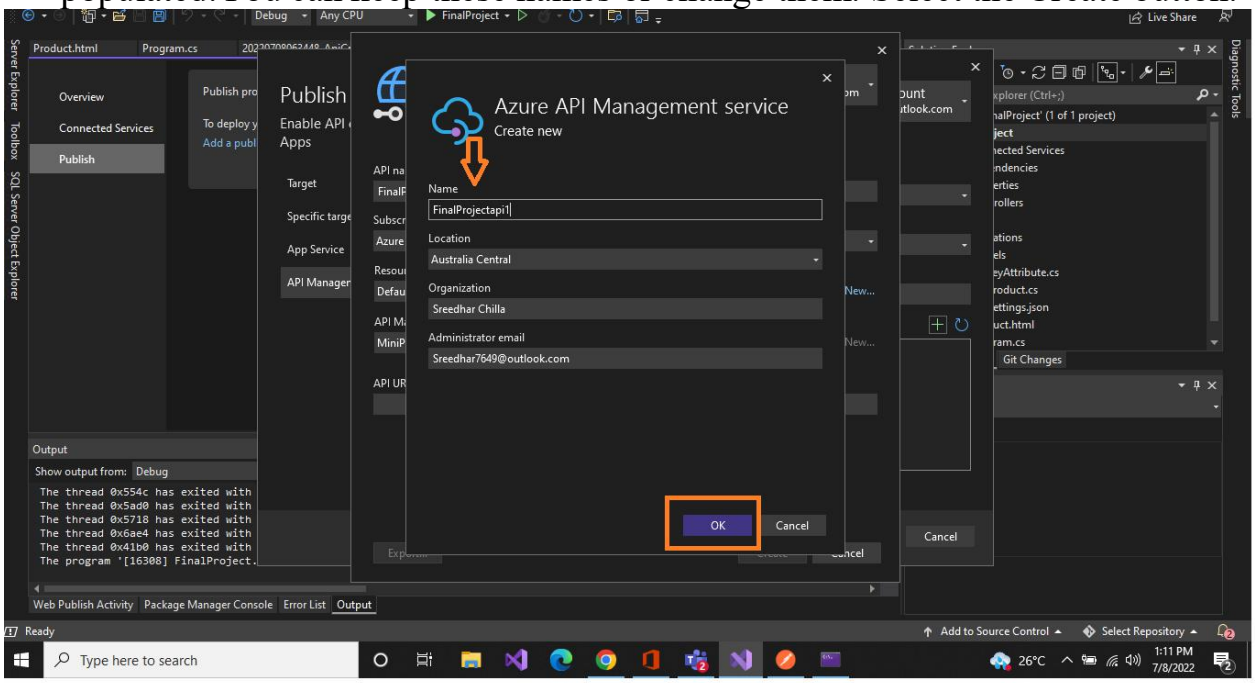
✧ After creation is completed, the dialog is automatically closed and the Publish dialog gets focus again. The instance that was created is automatically selected.



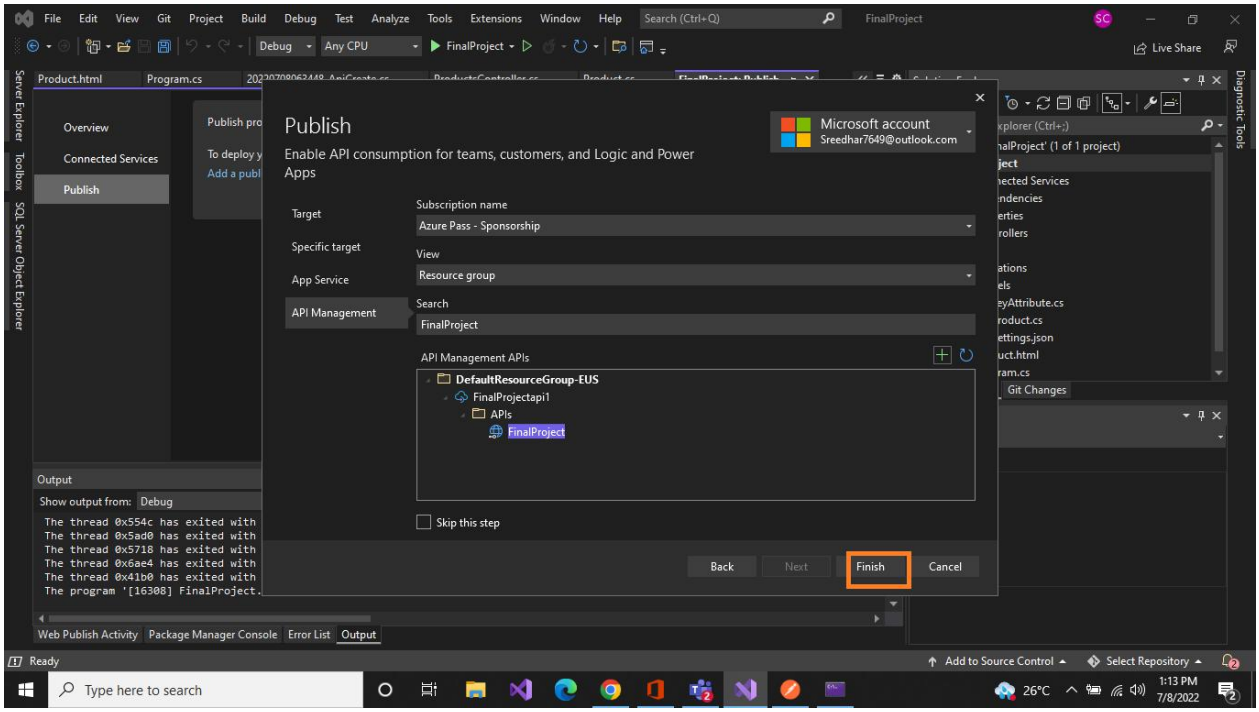
✧ The dialog now shows the Azure API Management service to create.



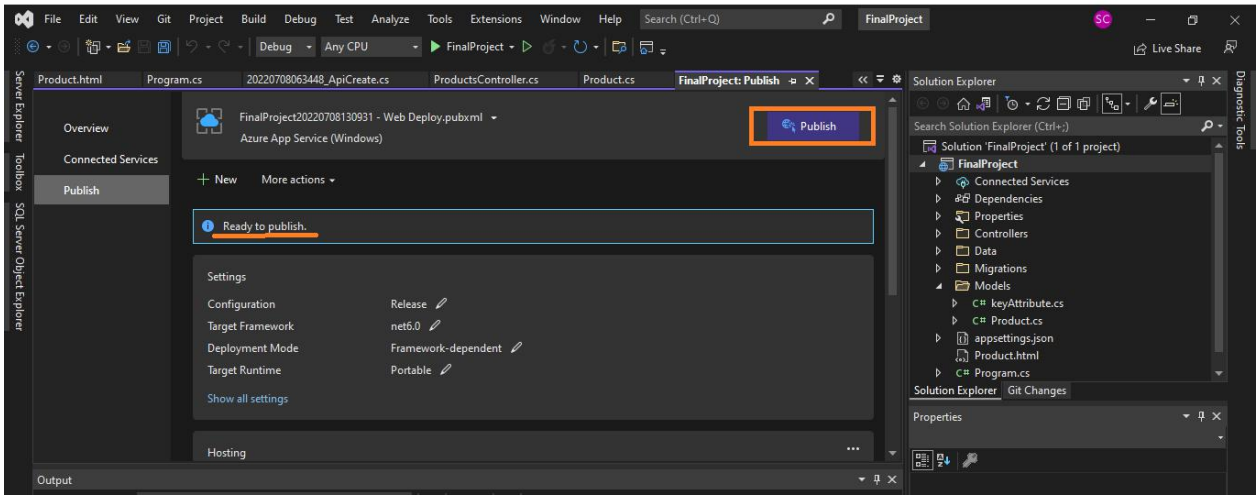
✧ The Create the Azure API Management service dialog appears. The App Name, Resource Group, and API Management service entry fields are populated.You can keep these names or change them. Select the Create button.



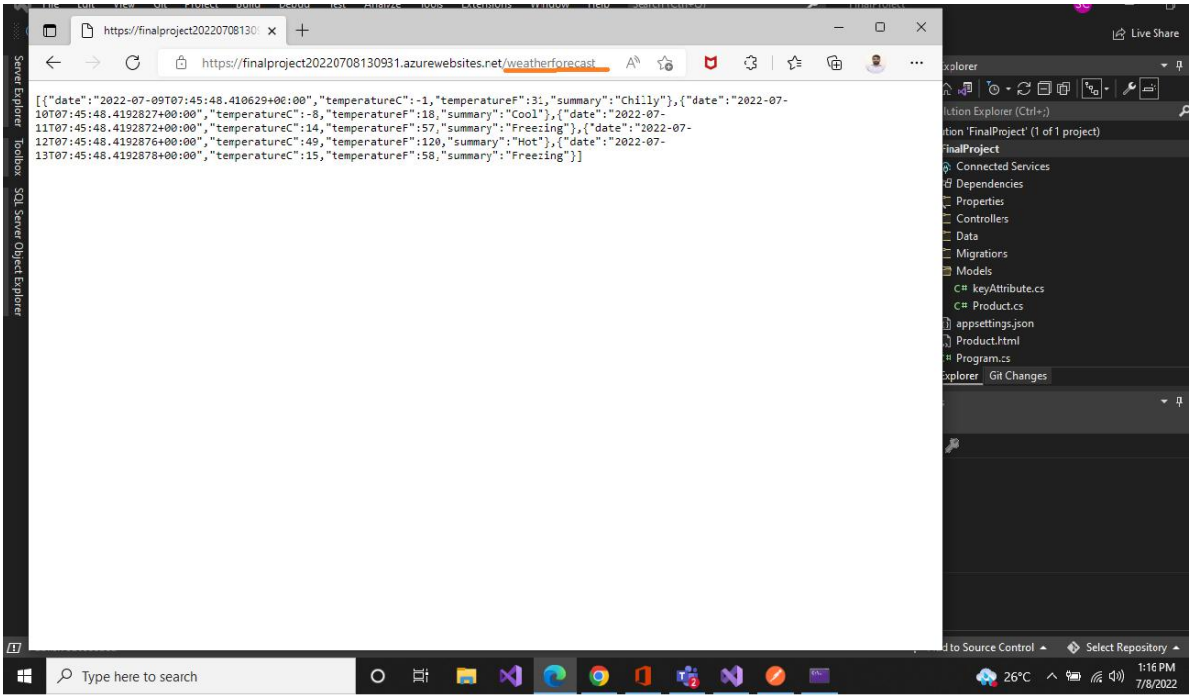
✧ After creation is completed, the dialog is automatically closed and the Publish dialog gets focus again. The instance that was created is automatically selected.



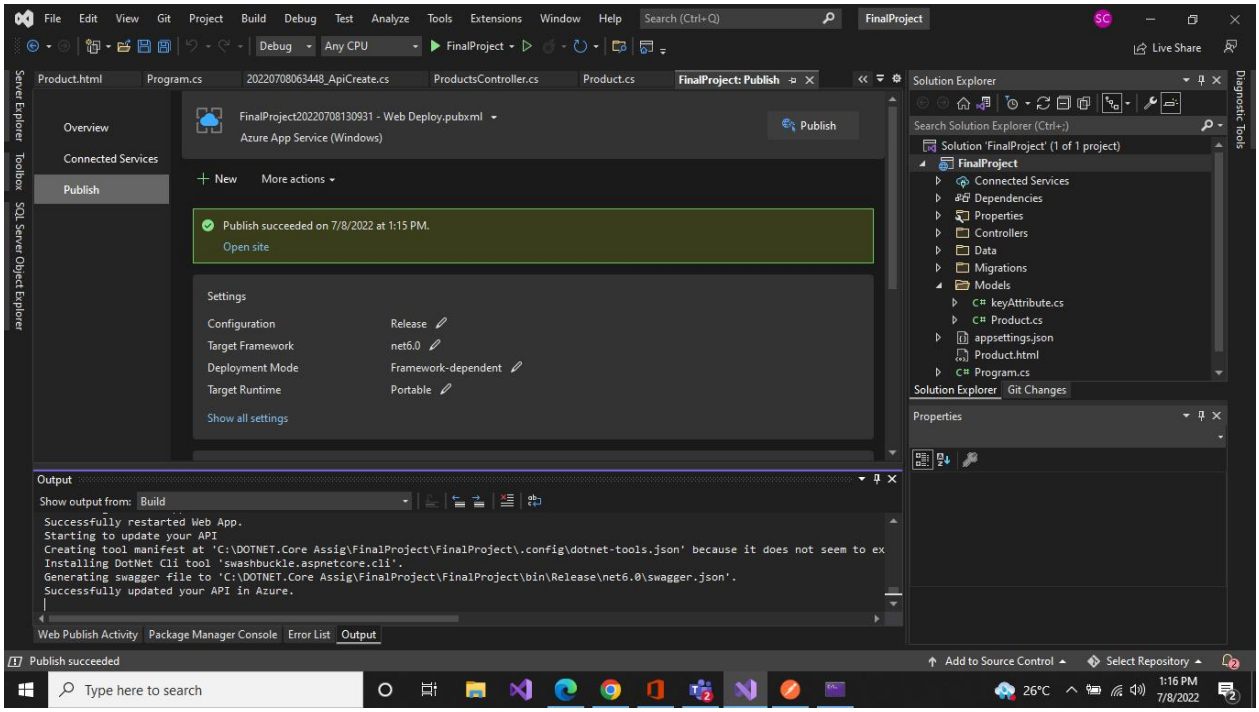
✧ The dialog closes and a summary screen appears with information about the publish. Select the Publish button.



- ✧ Select the Publish button after that it will Publishing to Azure App Service(Window) and checking your application will run Successfully..
- ✧ The web API will publish to both Azure App Service and Azure API Management. A new browser window will appear and show the API running inAzure App Service. You can close that window.

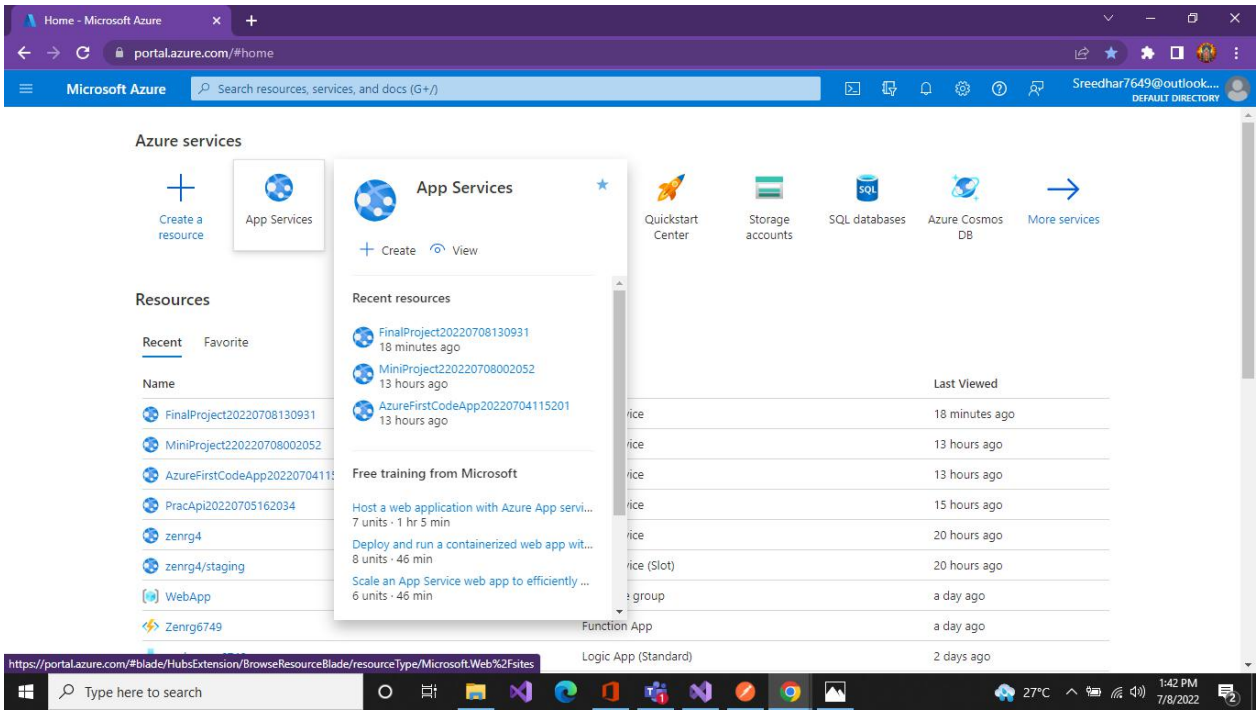


✧ Select the Publish button on site to open.

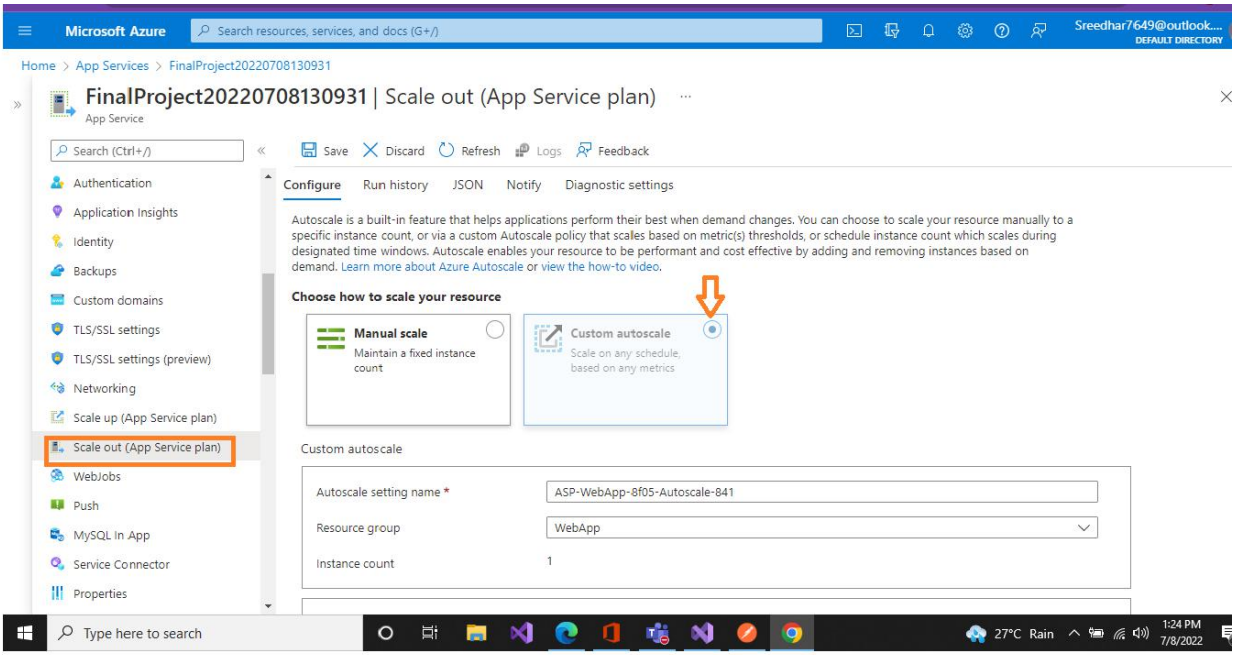


2. Configure Scale out by adding rules for custom scaling

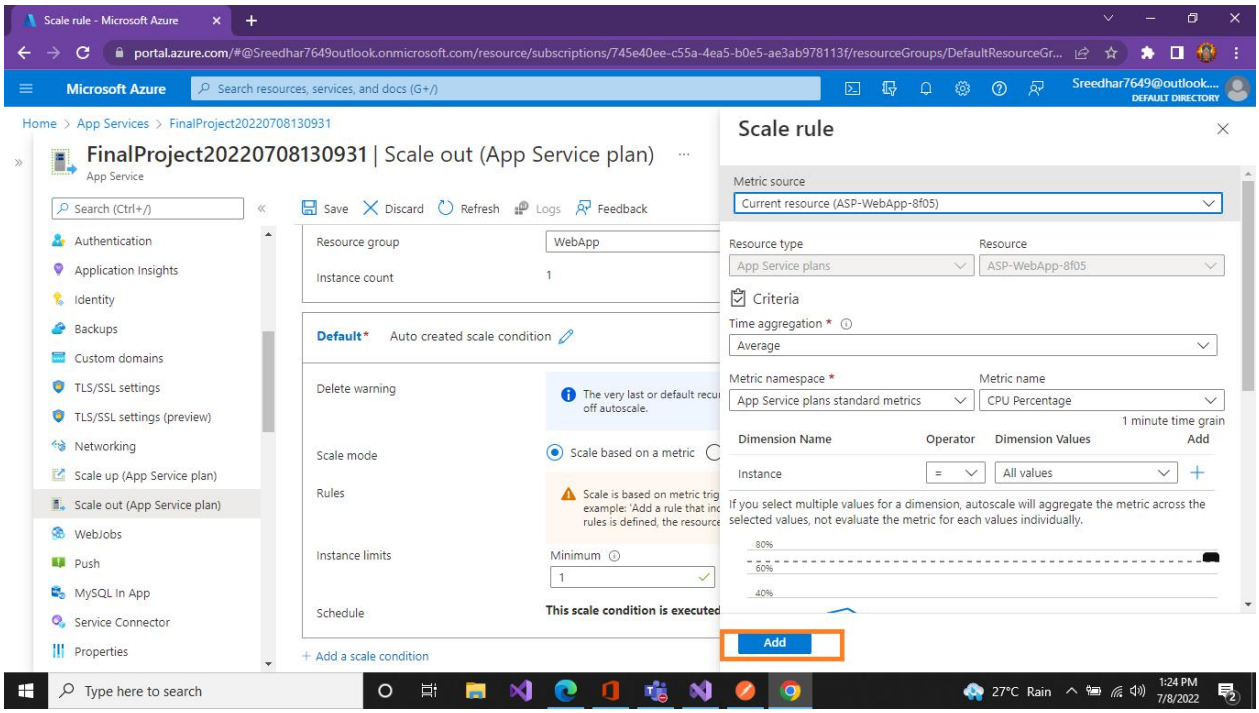
✧ Switch back to the Azure API Management instance in the Azure portal. Refresh the browser window. Select the API you created in the preceding steps.It's now populated and you can explore around.



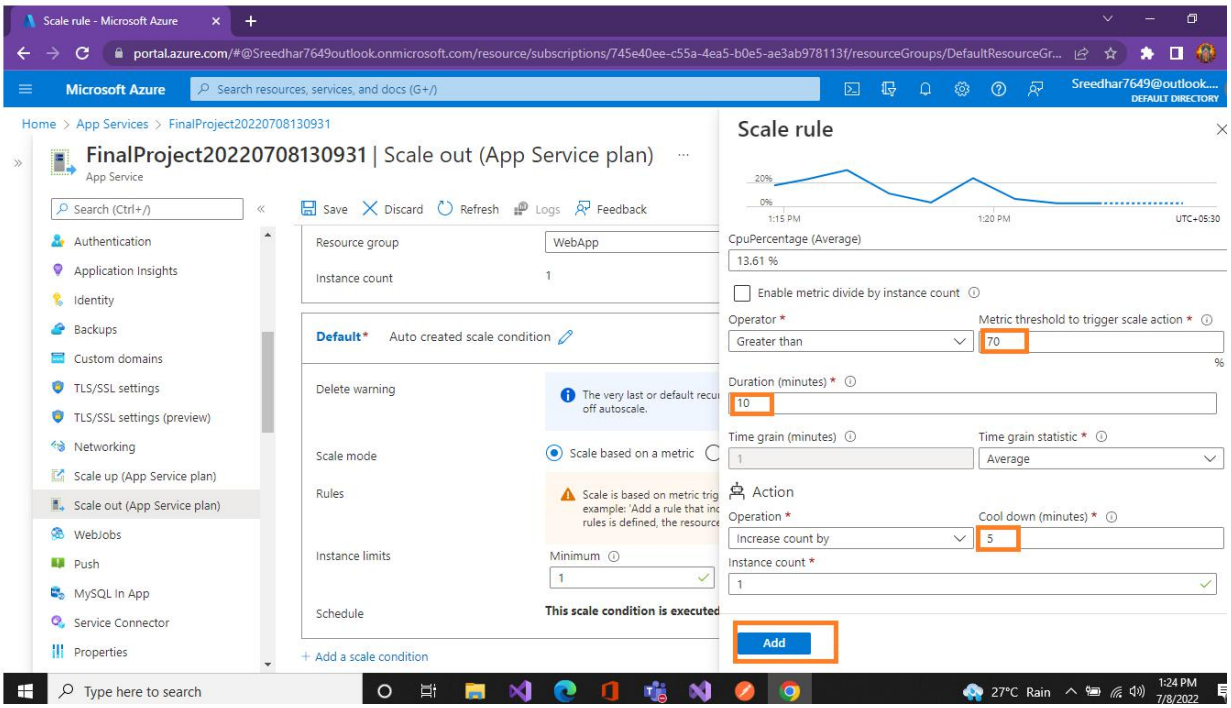
✧ Search and select Auto scale in the search bar and Select Custom Auto scale In the Rules section of the default scale condition, select Add a rule. From the Metric source dropdown, select current resource. From Resource Type, select Application Insights.



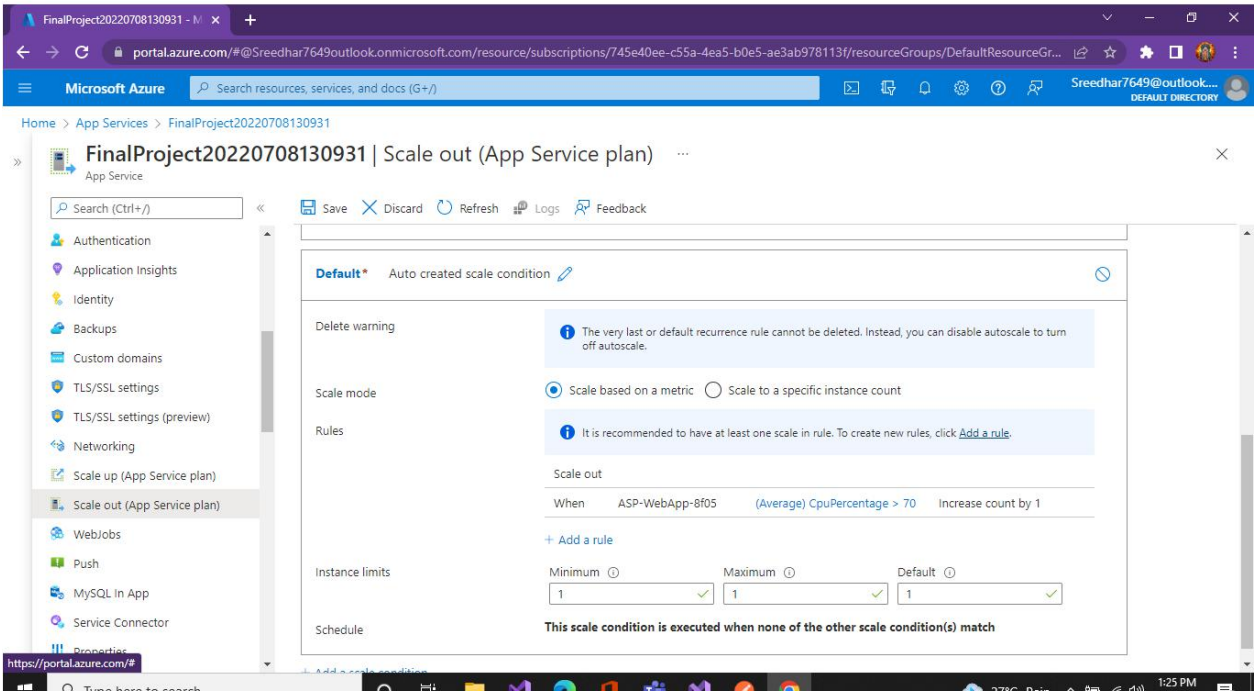
✧ From the Resource dropdown, select your App services plan standard metrics. Select a Metric name to CPU Percentage and Select Enable metric divide by instance count so that the number of sessions per instance is measured.From theOperator dropdown, select Greater than.



✧ Enter the Metric threshold to trigger the scale action, for example, 70 and Under Actions, set the Operation to Increase count and set the Instance count to 1and Cool down by 5minutes and then click Add. Set the maximum number of instances that can be spun up in the Maximum field of the Instance limits sectionand Select Add.

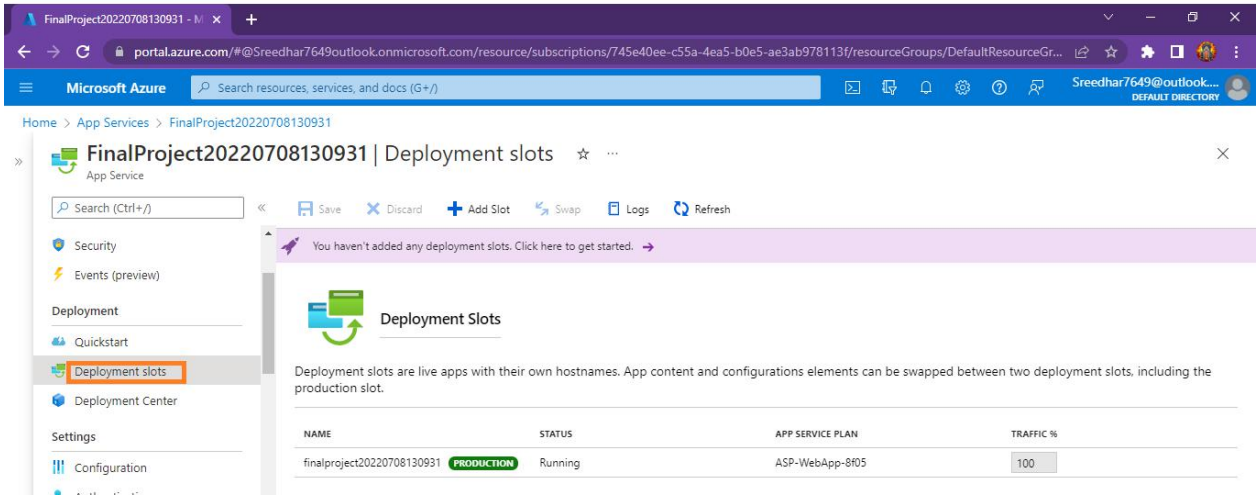


✧ After adding the scale it show Rules.

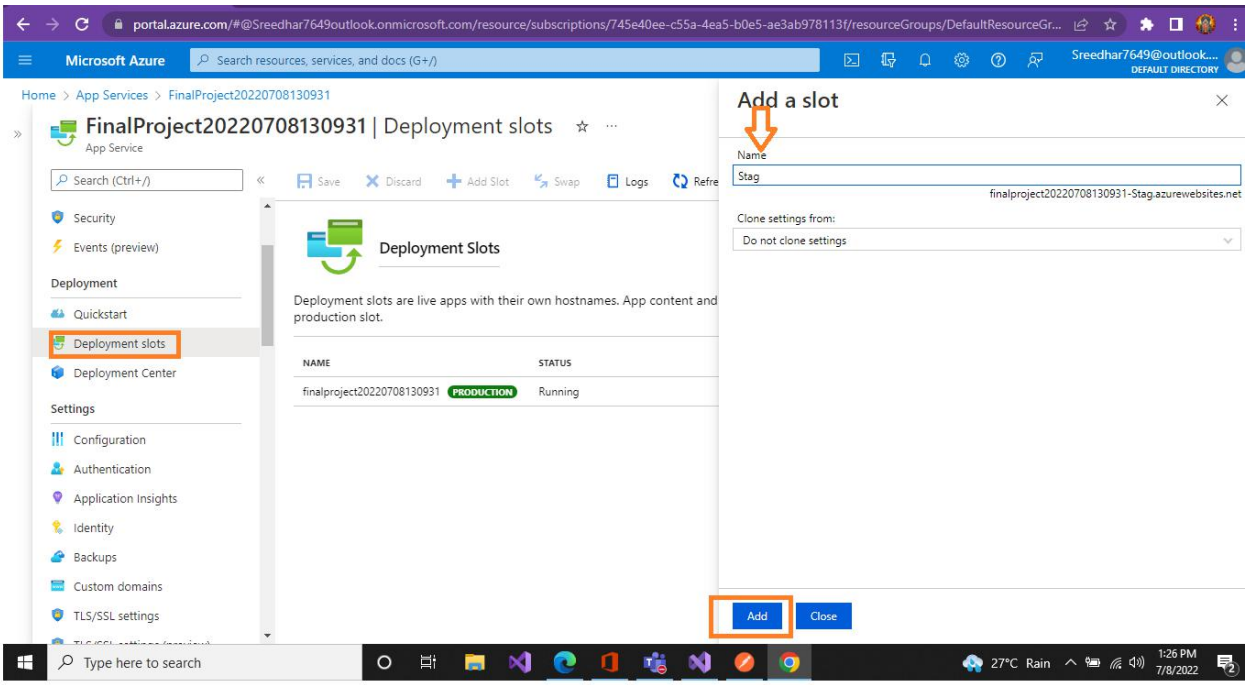


3. Configure Deployment slots for staging and production

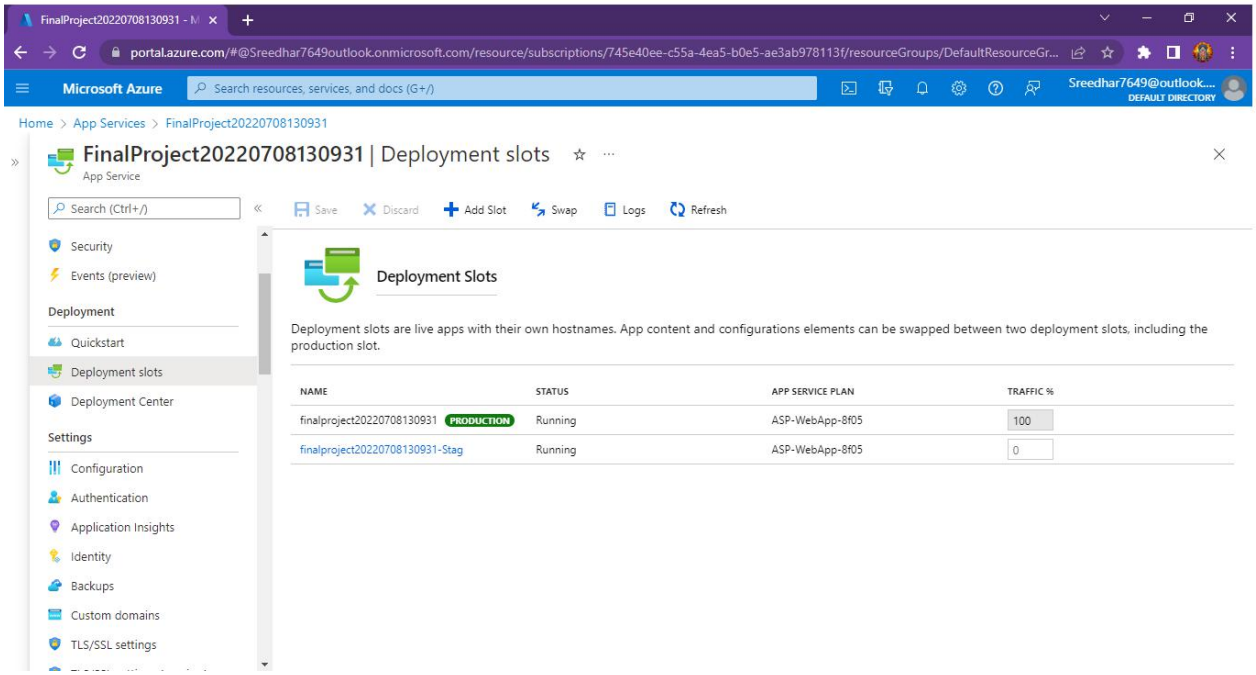
Azure Functions deployment slots allow your function app to run different instances called "slots". Slots are different environments exposed via a publicly available endpoint. One app instance is always mapped to the production slot, and you can swap instances assigned to a slot on demand. Function apps running under the Apps Service plan may have multiple slots, while under the Consumption plan only one slot is allowed.



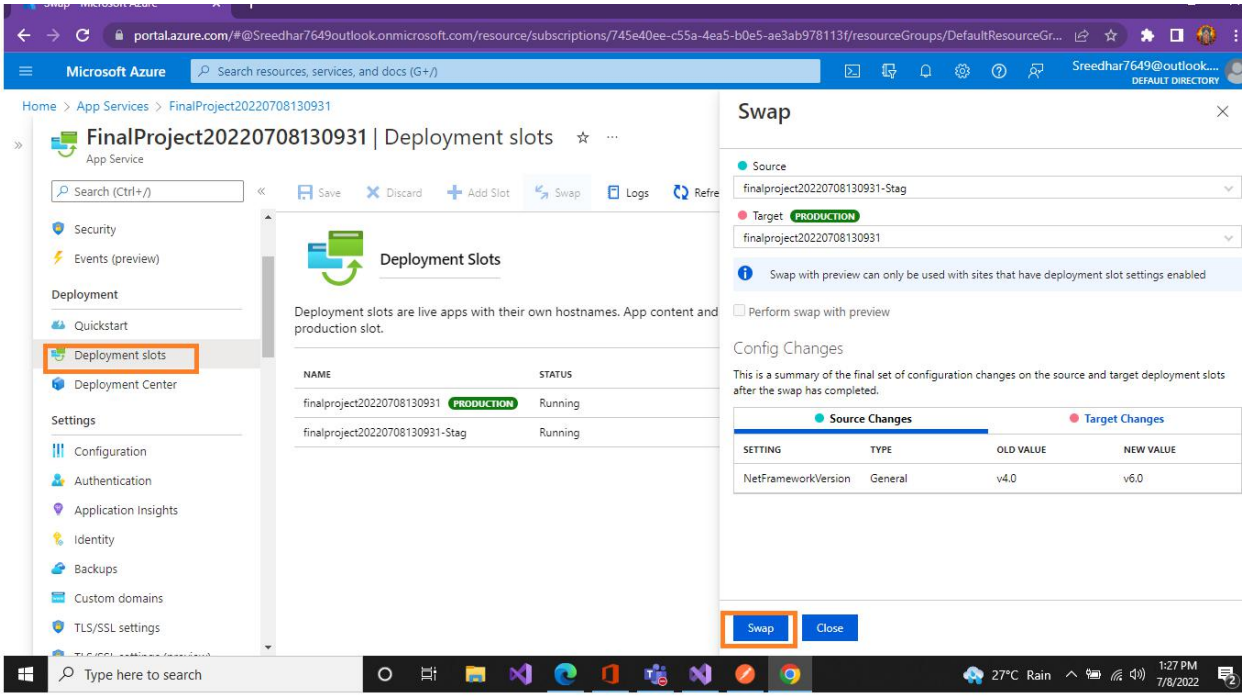
✧ Add the Slot and Name as Staging and then Add the Slot .



✧ After Add Slot Successful both the Production and Staging is Displayed .



✧ Select Deployment slots, and then select Swap and Verify the configuration settings for your swap and select Swap.

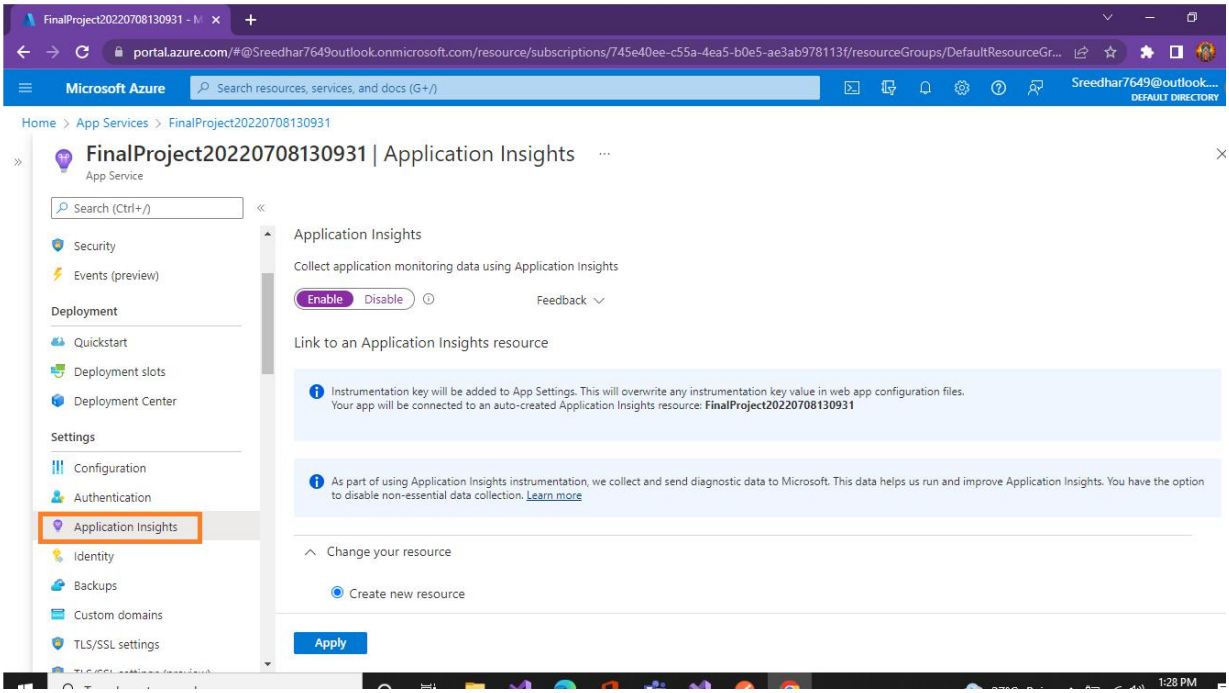


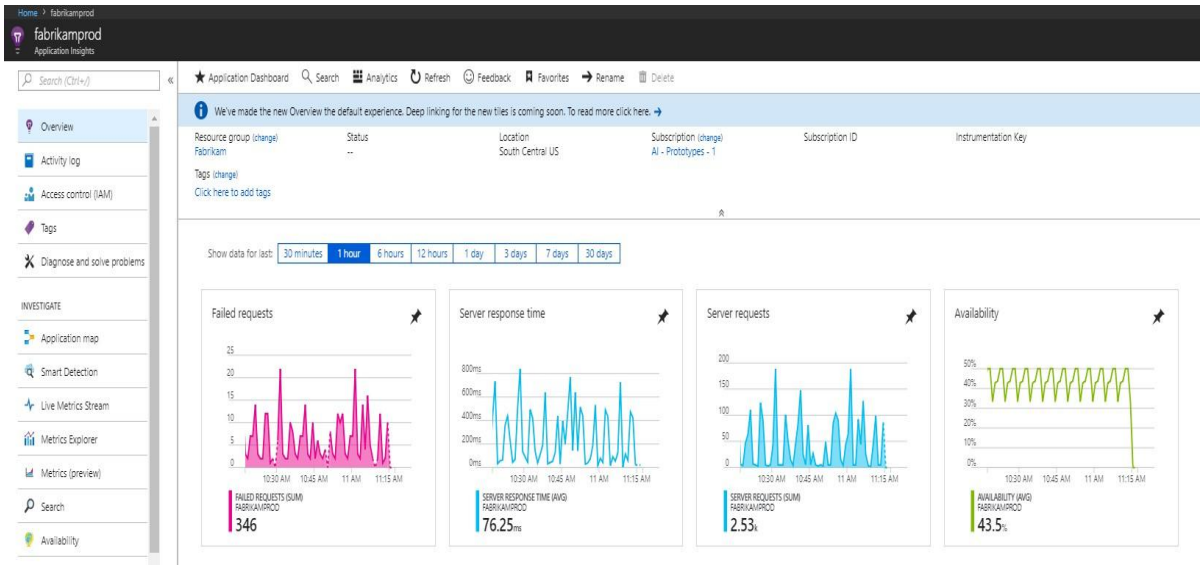
4. Configure Application Insights for the project

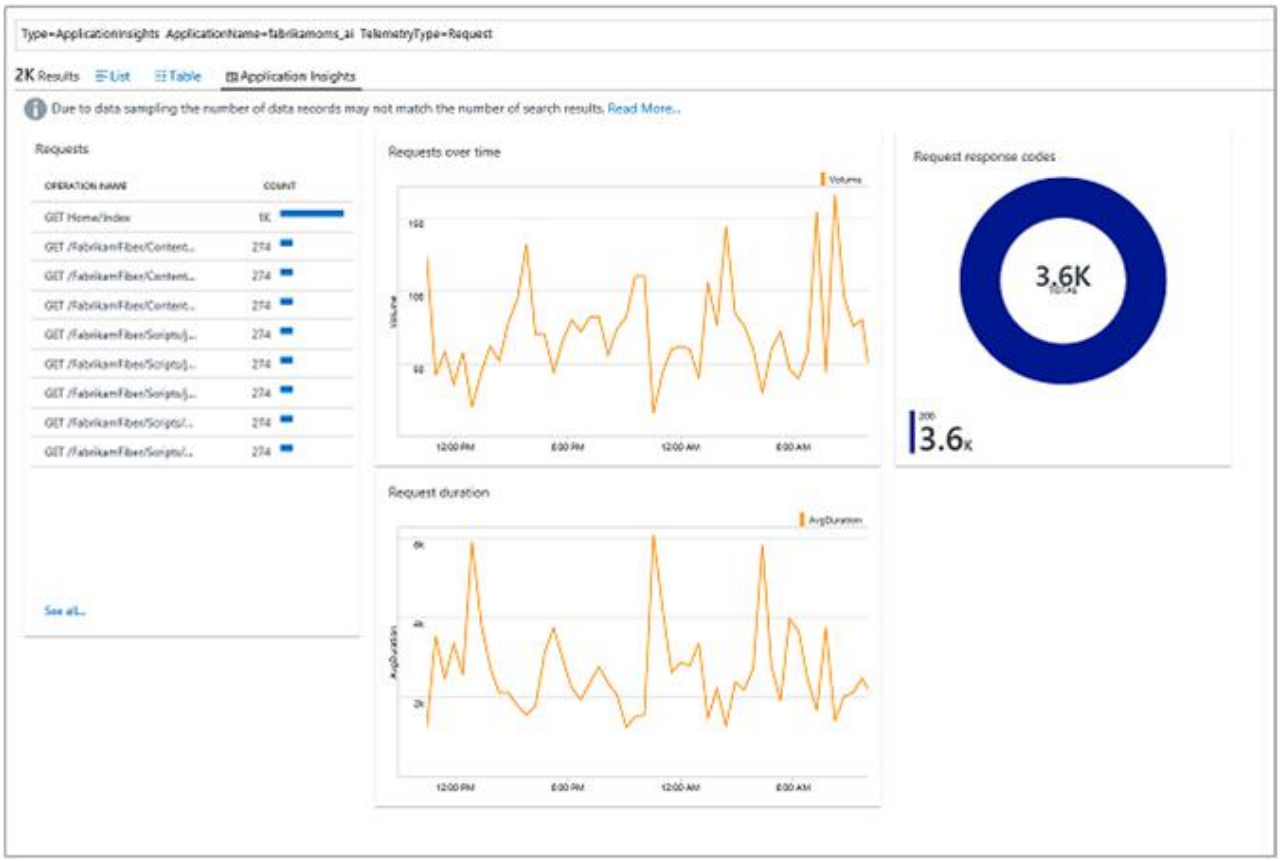
Application Insights is a feature of Azure Monitor that provides extensible application performance management (APM) and monitoring for live web apps. Developers and DevOps professionals can use Application Insights to.

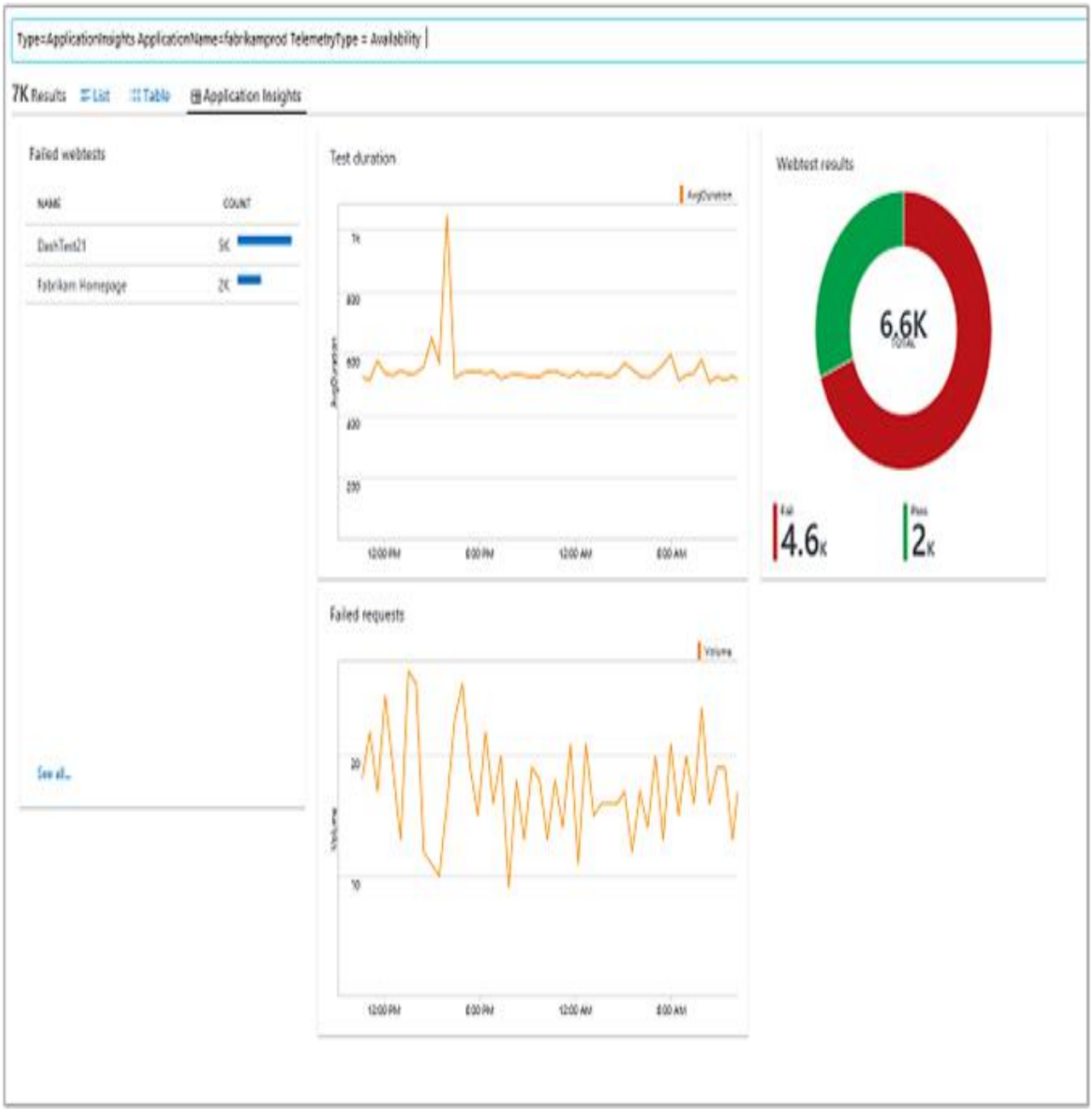
- Automatically detect performance anomalies.
- Help diagnose issues by using powerful analytics tools.
- See what users actually do with apps.
- Help continuously improve app performance and usability.

✧ Select the Application Insights




Home › fabrikamprod
fabrikamprod
Application Insights
Search (Ctrl+/)
Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
INVESTIGATE
Application map
Smart Detection
Live Metrics Stream
Metrics Explorer
Metrics (preview)
Search
Availability
Application Dashboard
Search
Analytics
Refresh
Feedback
Favorites
Rename
Delete
We've made the new Overview the default experience. Deep linking for the new tiles is coming soon. To read more click here.
Resource group (change)
Fabrikam
Status
--
Location
South Central US
Subscription (change)
AI - Prototypes - 1
Subscription ID
Instrumentation Key
Tags (change)
Click here to add tags
Show data for last:
30 minutes
1 hour
6 hours
12 hours
1 day
3 days
7 days
30 days
Failed requests
FAILED REQUESTS (SUM)
FABRIKAMPROD
346
Server response time
SERVER RESPONSE TIME (AVG)
FABRIKAMPROD
76.25ms
Server requests
SERVER REQUESTS (SUM)
FABRIKAMPROD
2.53k
Availability
AVAILABILITY (AVG)
FABRIKAMPROD
43.5%


Type=ApplicationInsights ApplicationName=fabrikamoms_ai TelemetryType=Request
2K Results
List
Table
Application Insights
Due to data sampling the number of data records may not match the number of search results. Read More...
Requests
OPERATION NAME
COUNT
GET Home/Index
1K
GET /FabrikamFiber/Content...
274
GET /FabrikamFiber/Content...
274
GET /FabrikamFiber/Content...
274
GET /FabrikamFiber/Scripts/j...
274
GET /FabrikamFiber/Scripts/j...
274
GET /FabrikamFiber/Scripts/j...
274
GET /FabrikamFiber/Scripts/...
274
GET /FabrikamFiber/Scripts/...
274
See all...
Requests over time
Volume
12:00 PM
6:00 PM
12:00 AM
6:00 AM
Request response codes
3.6K
200
3.6k
Request duration
AvgDuration
12:00 PM
6:00 PM
12:00 AM
6:00 AM

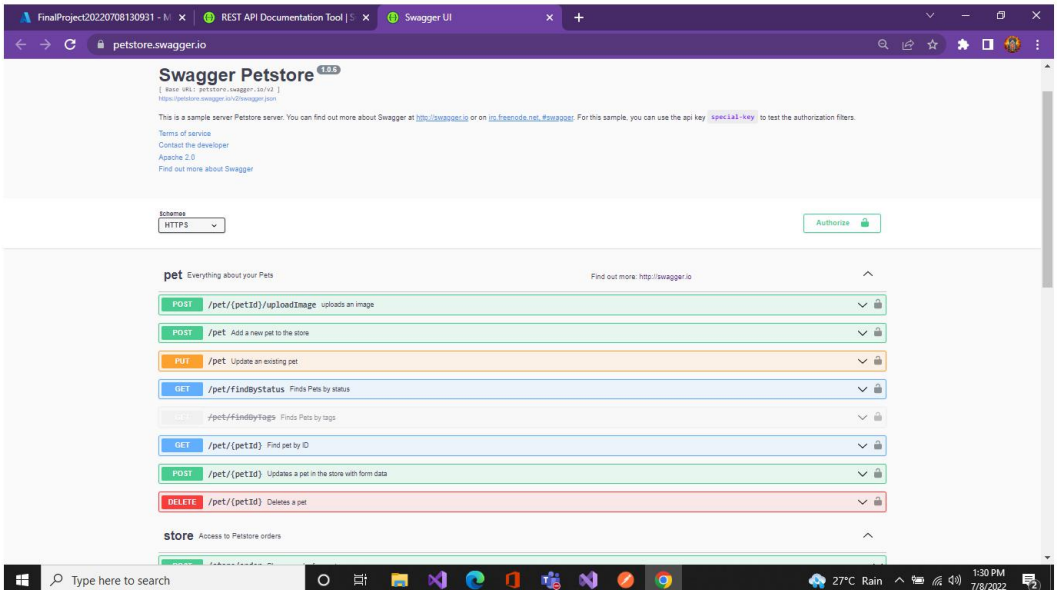

Type=ApplicationInsights ApplicationName=fabrikamprod TelemetryType = Availability
7K Results
List
Table
Application Insights
Failed webtests
NAME
COUNT
DavisTest21
5K
Fabrikam Homepage
2K
See all...
Test duration
AvgDuration
1K
800
600
400
200
12:00 PM
8:00 PM
12:00 AM
8:00 AM
Webtest results
6.6K
TOTAL
Fail
4.6K
Pass
2K
Failed requests
Volume
20
10
12:00 PM
8:00 PM
12:00 AM
8:00 AM

5. Configure Swagger for the API

Swagger UI allows anyone be it your development team or your end consumers to visualize and interact with the API's resources without having any of the implementation logic in place. It's automatically generated from your Open API (formerly known as Swagger) Specification, with the visual documentation making it easy for back end implementation and client side consumption.

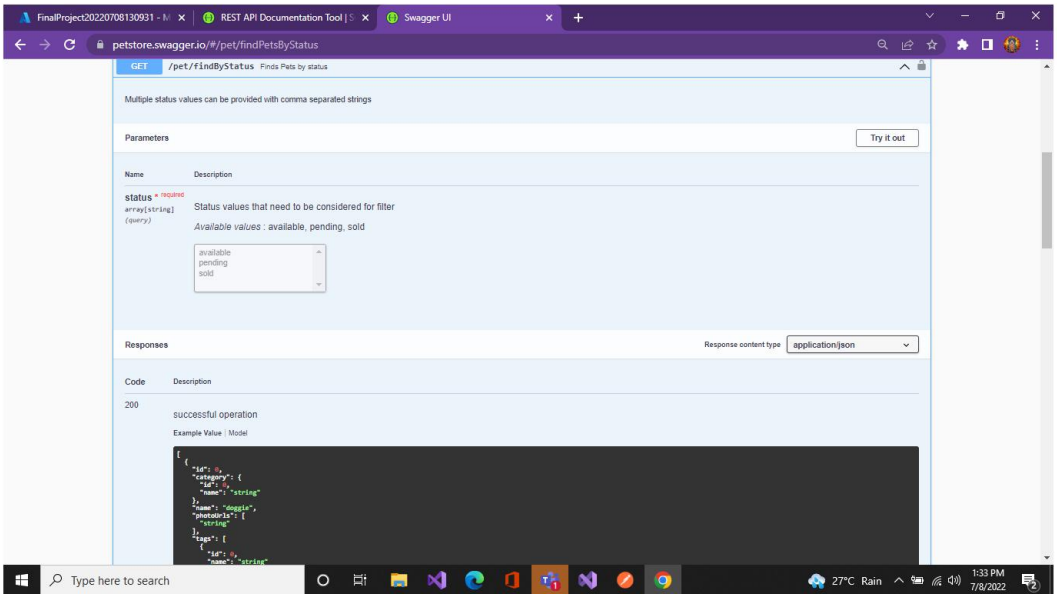
Advantages :-

- Dependency Free - The UI works in any development environment, be it locally or in the web
- Human Friendly - Allow end developers to effortlessly interact and try out every single operation your API exposes for easy consumption
- Easy to Navigate - Quickly find and work with resources and endpoints with neatly categorized documentation
- All Browser Support - Cater to every possible scenario with Swagger UI working in all major browsers.
- Fully Customizable - Style and tweak your Swagger UI the way you want with full source code access.
- Complete OAS Support - Visualize APIs defined in Swagger 2.0 or OAS 3.0



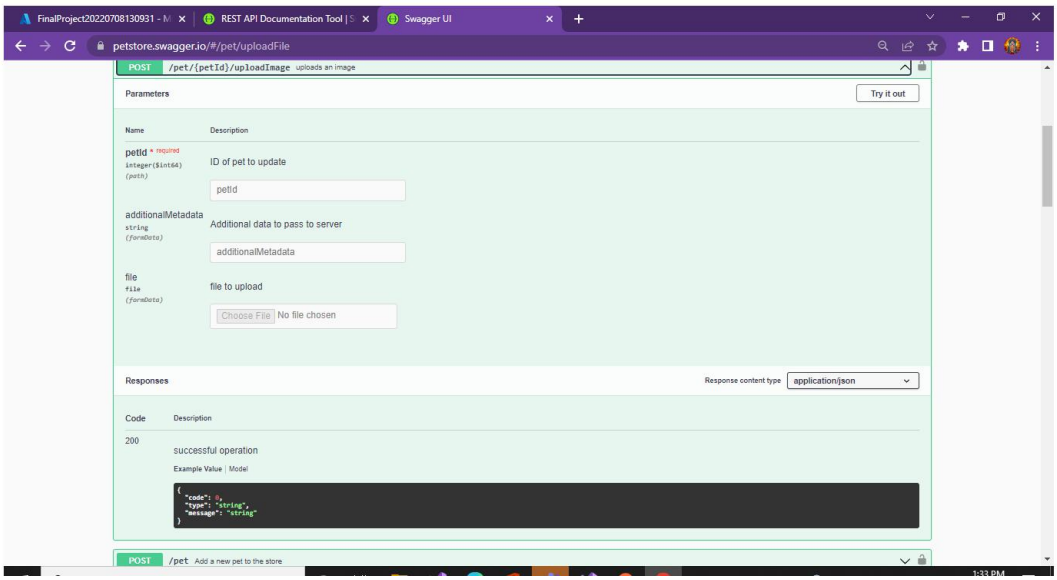
This image is showing swagger documentation for the created Products Web API project.

GET:-



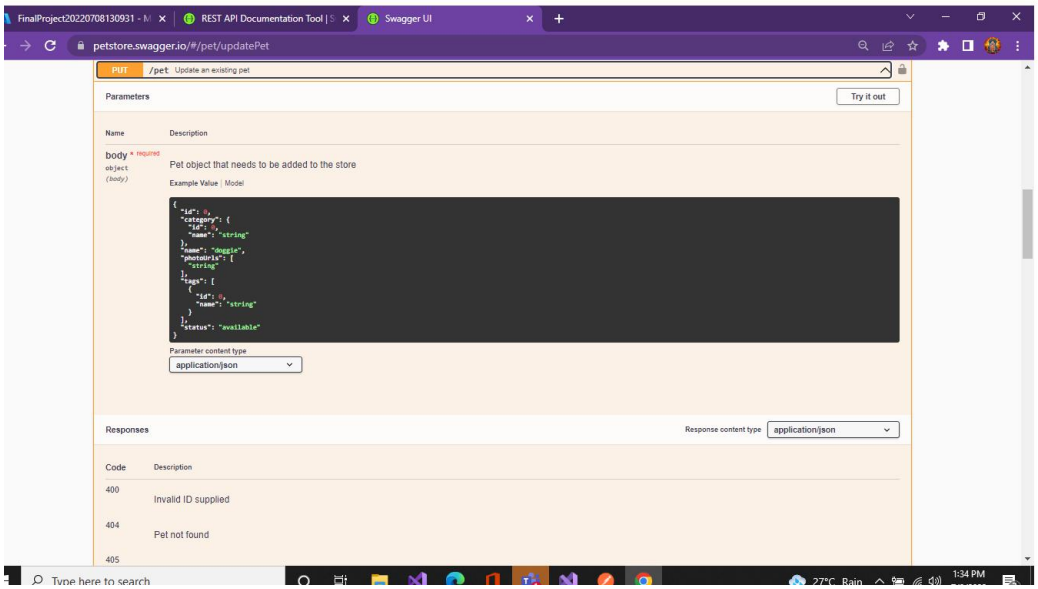
This Image showing the get call for a product with specified product id passed as a path variable.

POST:-



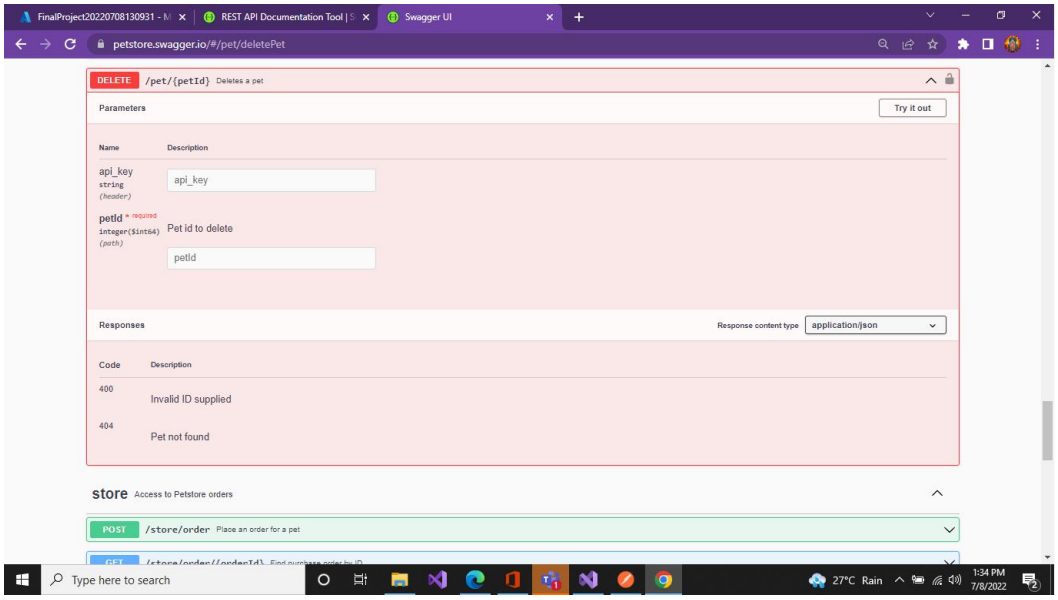
This Image showing the post call for creating a new product.

PUT:-



This Image showing the put call for updating an existing product with request body (fields to be updated) and path variable as product id.

DELETE:-

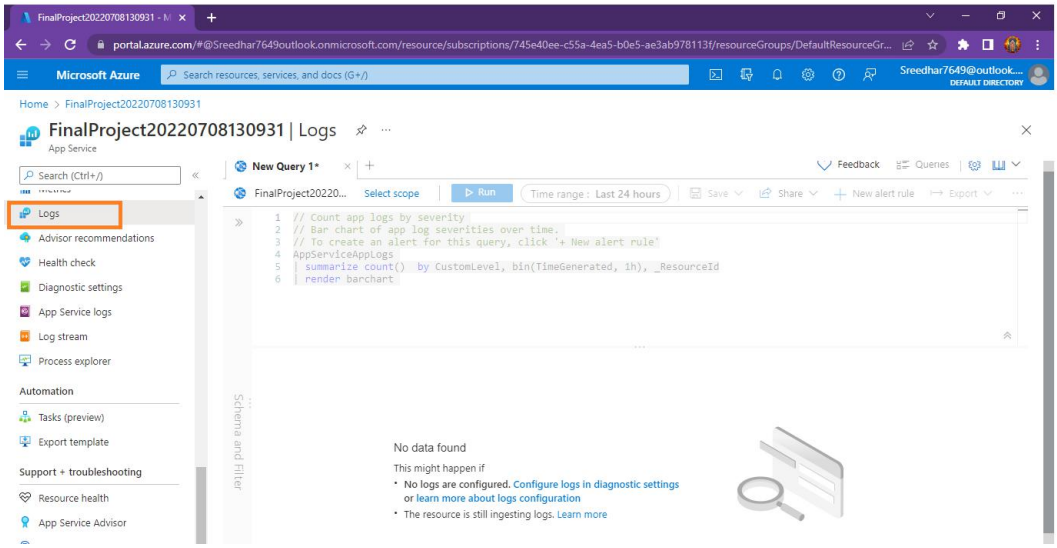


This Image showing the delete call for deleting an existing product with the specified id passed as a path variable.

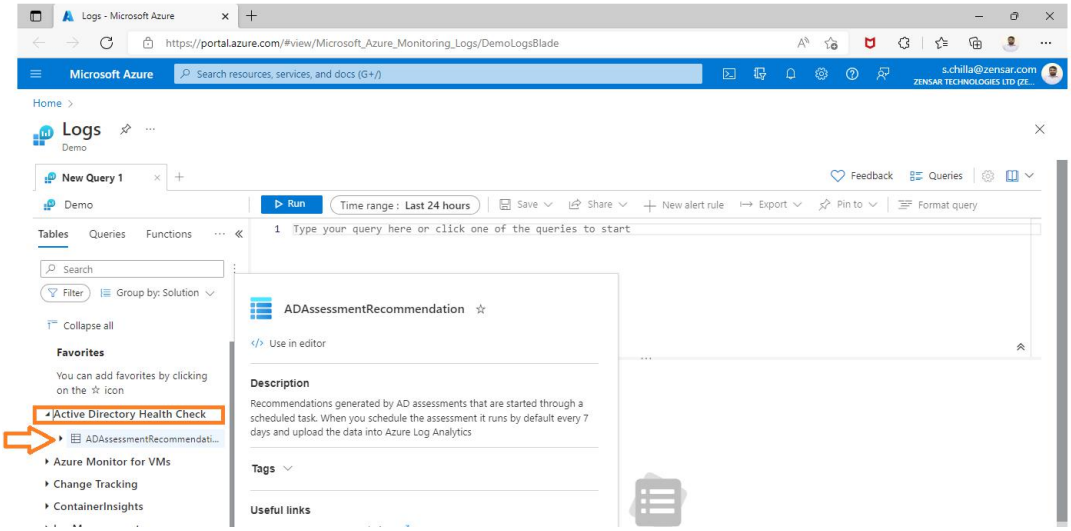
6. Work with Log Analytics with the sample logs available

Log Analytics is a tool in the Azure portal to edit and run log queries from data collected by Azure Monitor logs and interactively analyze their results. You can use Log Analytics queries to retrieve records that match particular criteria, identify trends, analyze patterns, and provide various insights into your data.

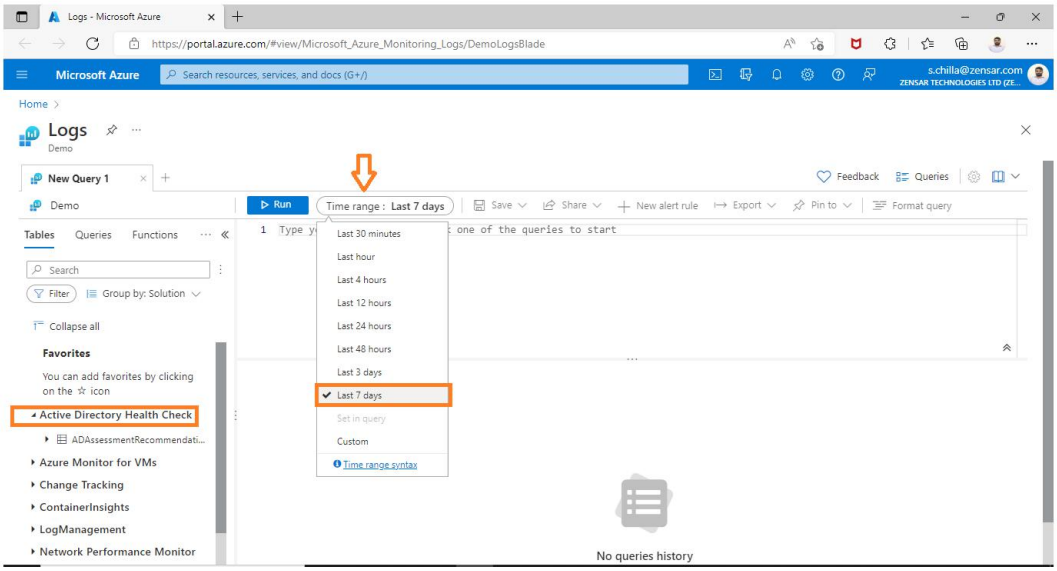
✧ Select the Logs in Azure Portal.



✧ Select Logs from the Azure Monitor menu . This step sets the initial scope to a Log Analytics workspace so that your query selects from all data in that workspace .



✧ All queries return records generated within a set time range. By default, the query returns records generated in the last 24 hours. You can set a different timerange by using the where operator in the query. You can also use the Time range dropdown list at the top of the screen. Change the time range of the query by selecting Last 12 hours from the Time range dropdown. Select Run to return theresults.



✧ This is the simplest query that we can write. It just returns all the records in a table. Run it by selecting the Run button or by selecting Shift+Enter with the cursor positioned anywhere in the query text and Select Run to return the results.