DOTNET CORE-AZURE

MINI 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

Use Data Annotations to

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

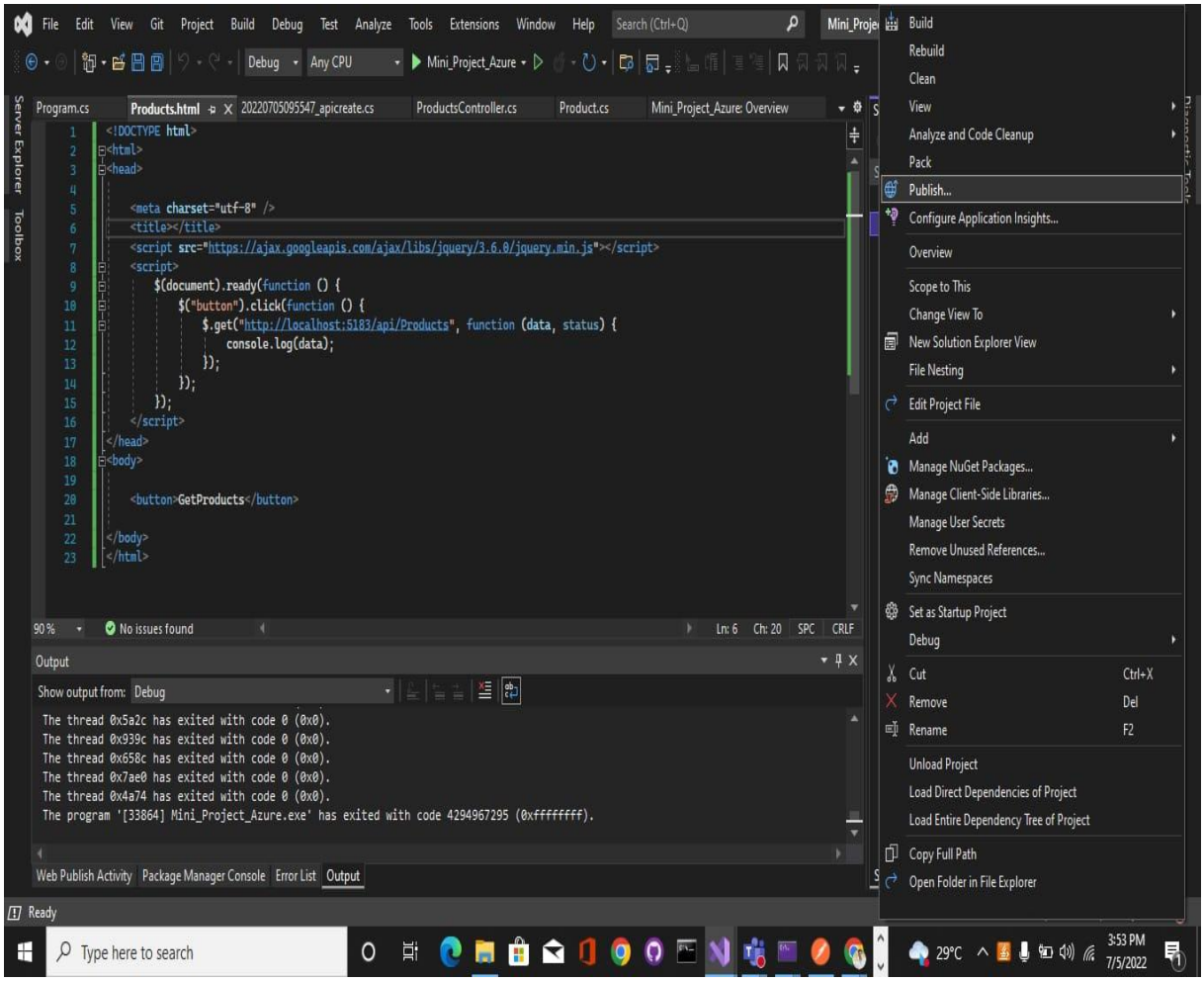
Create a jQuery and AJAX Client to consume the Web API and show the result.

Azure Hosting:

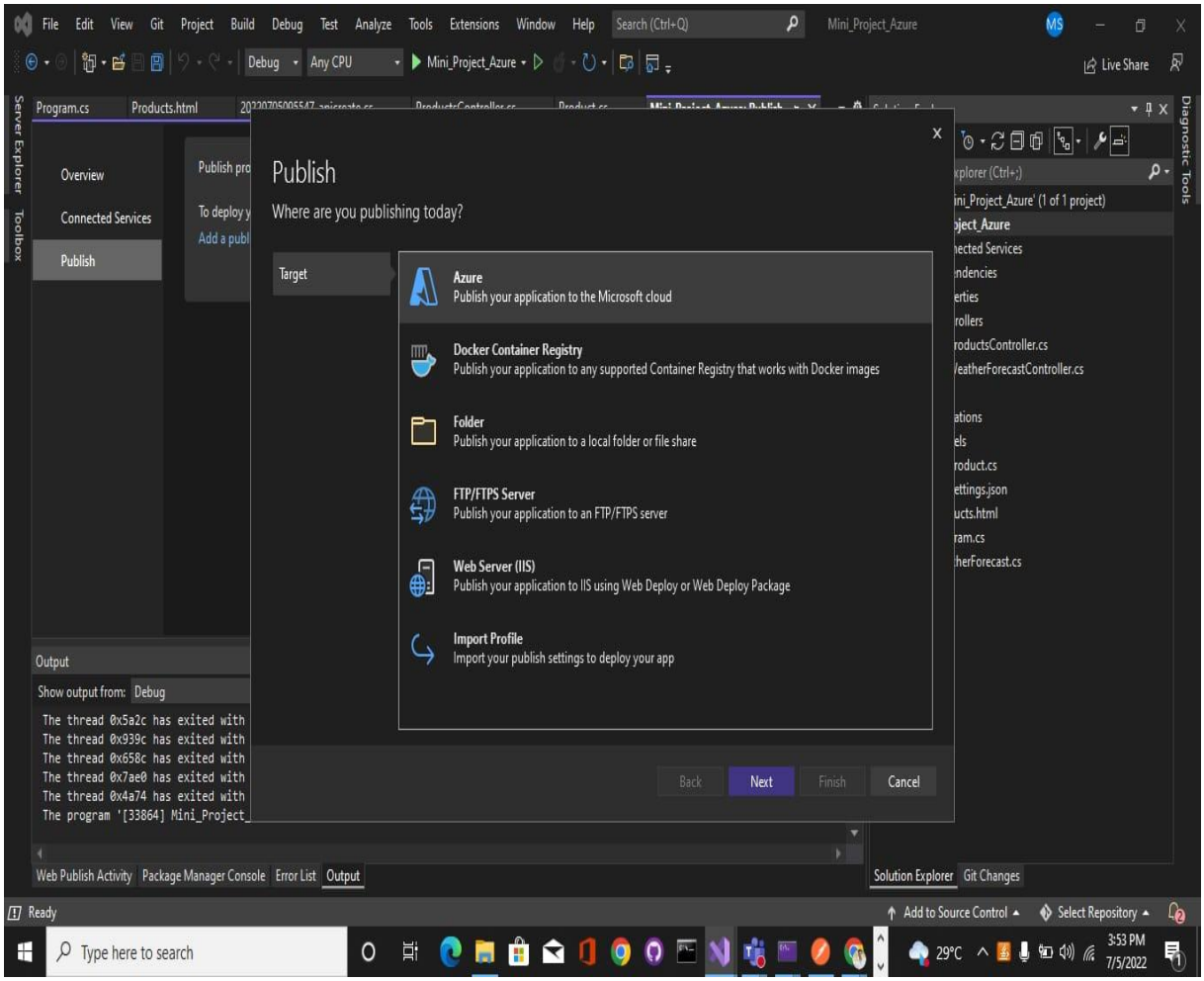
- Host the web api in azure and consume the same using jQuery Client.
- Configure Scale out by adding rules for custom scaling
- Configure Deployment slots for staging and production
- Configure Application Insights for the project
- Configure Swagger for the api
- Work with Log Analytics with the sample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

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

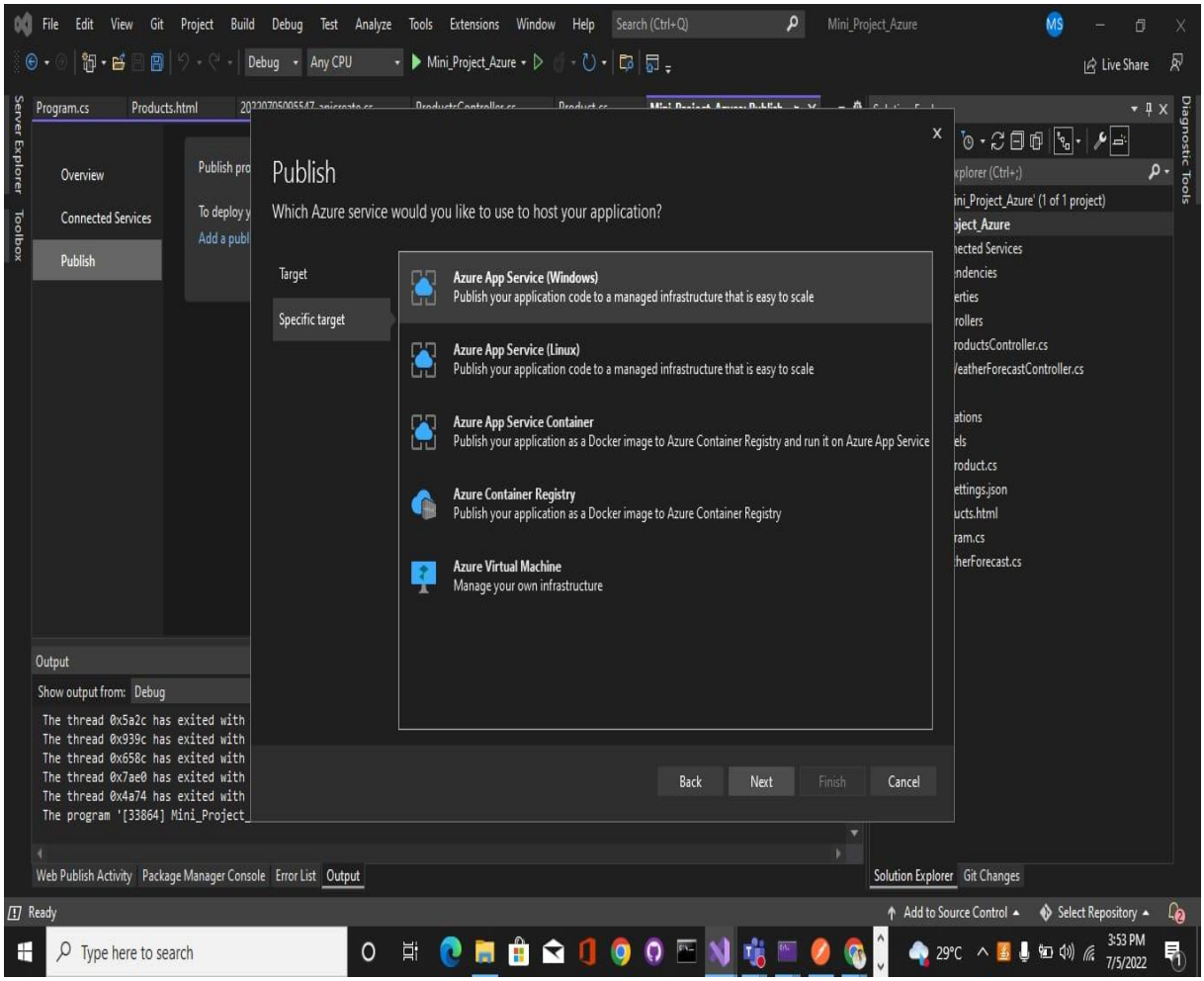
- In Solution Explorer, right-click the project and select Publish .



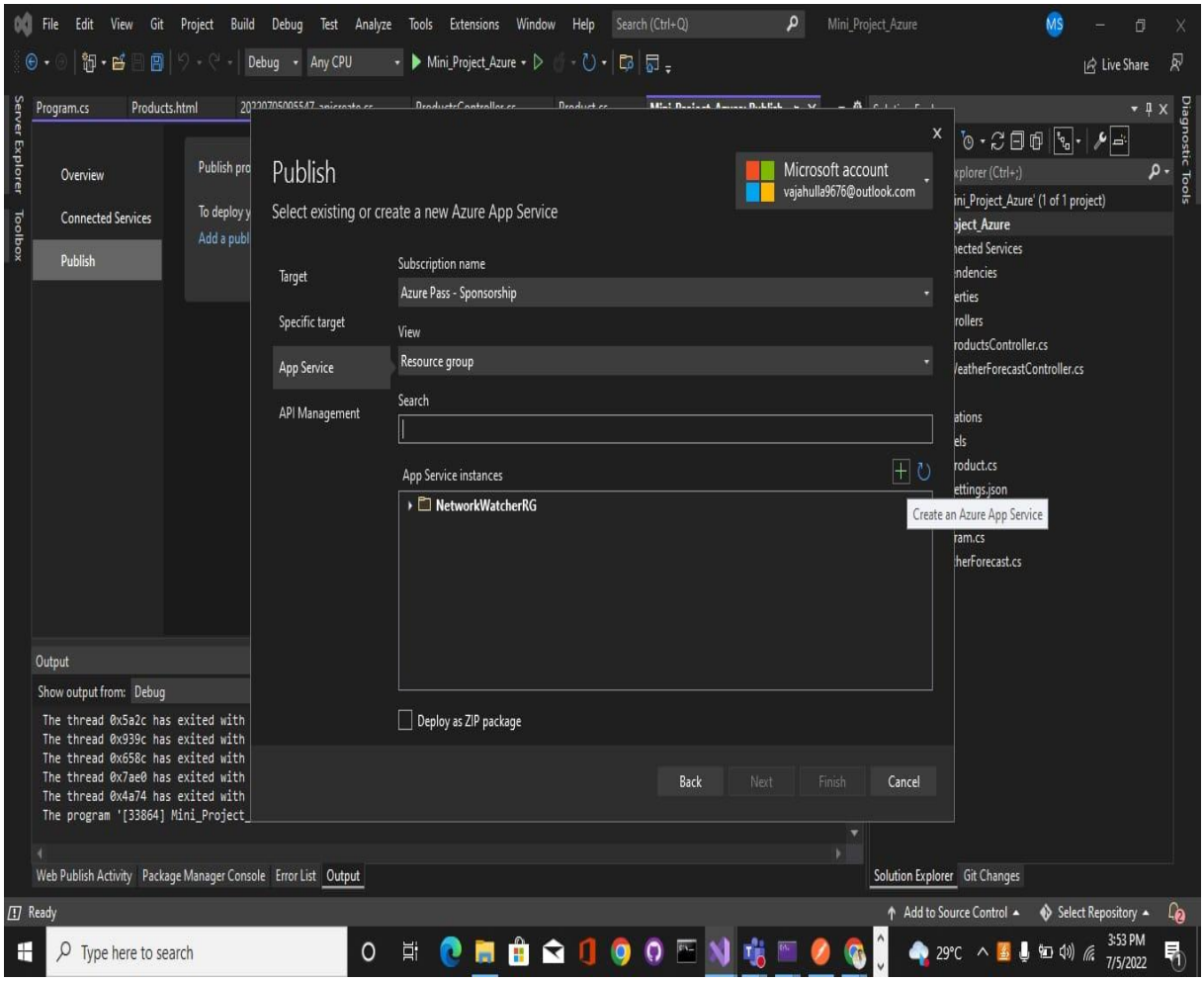
❖In the Publish dialog, select Azure and select the Next button.



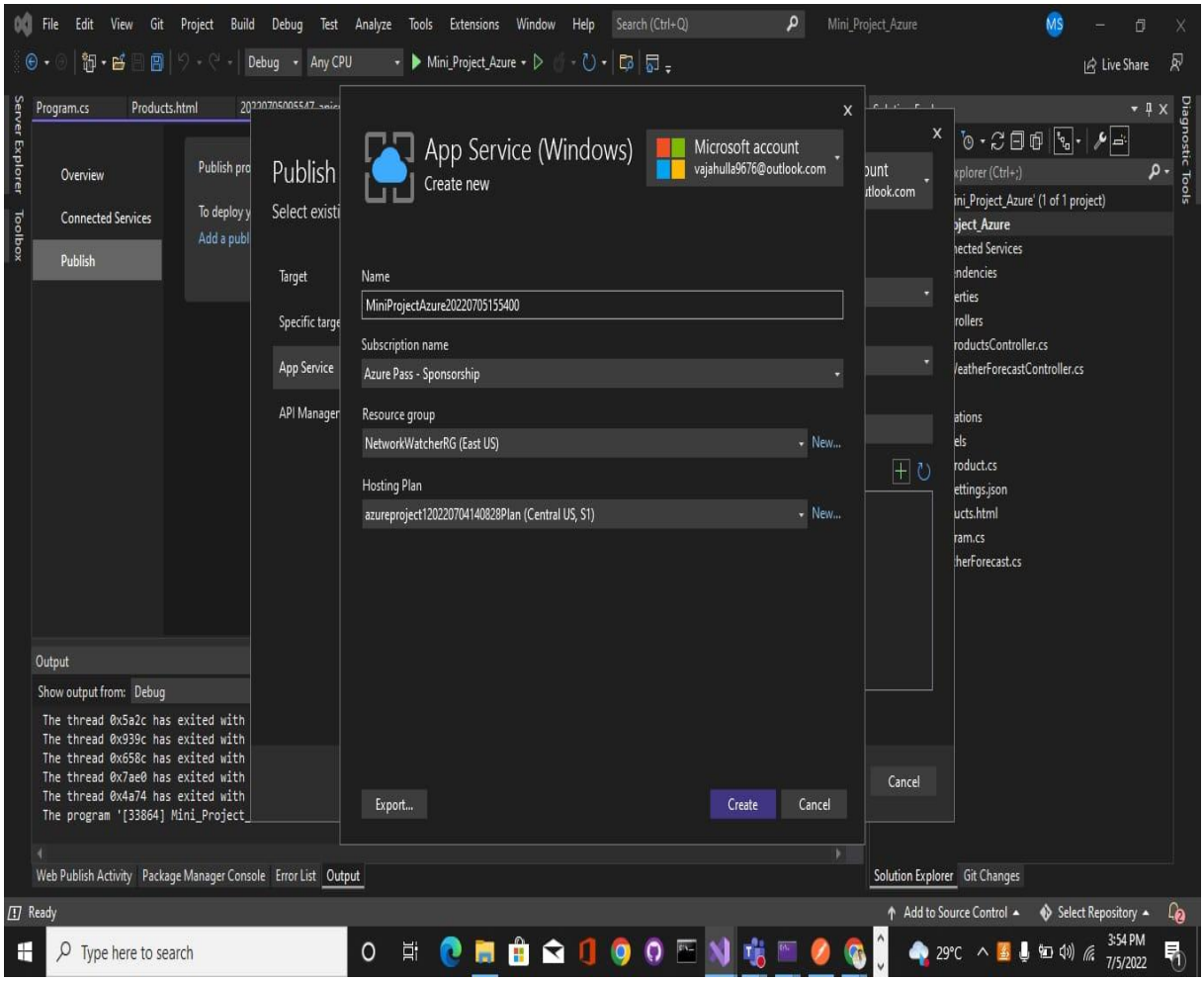
❖Select Azure App Service (Windows) and select the Next button .



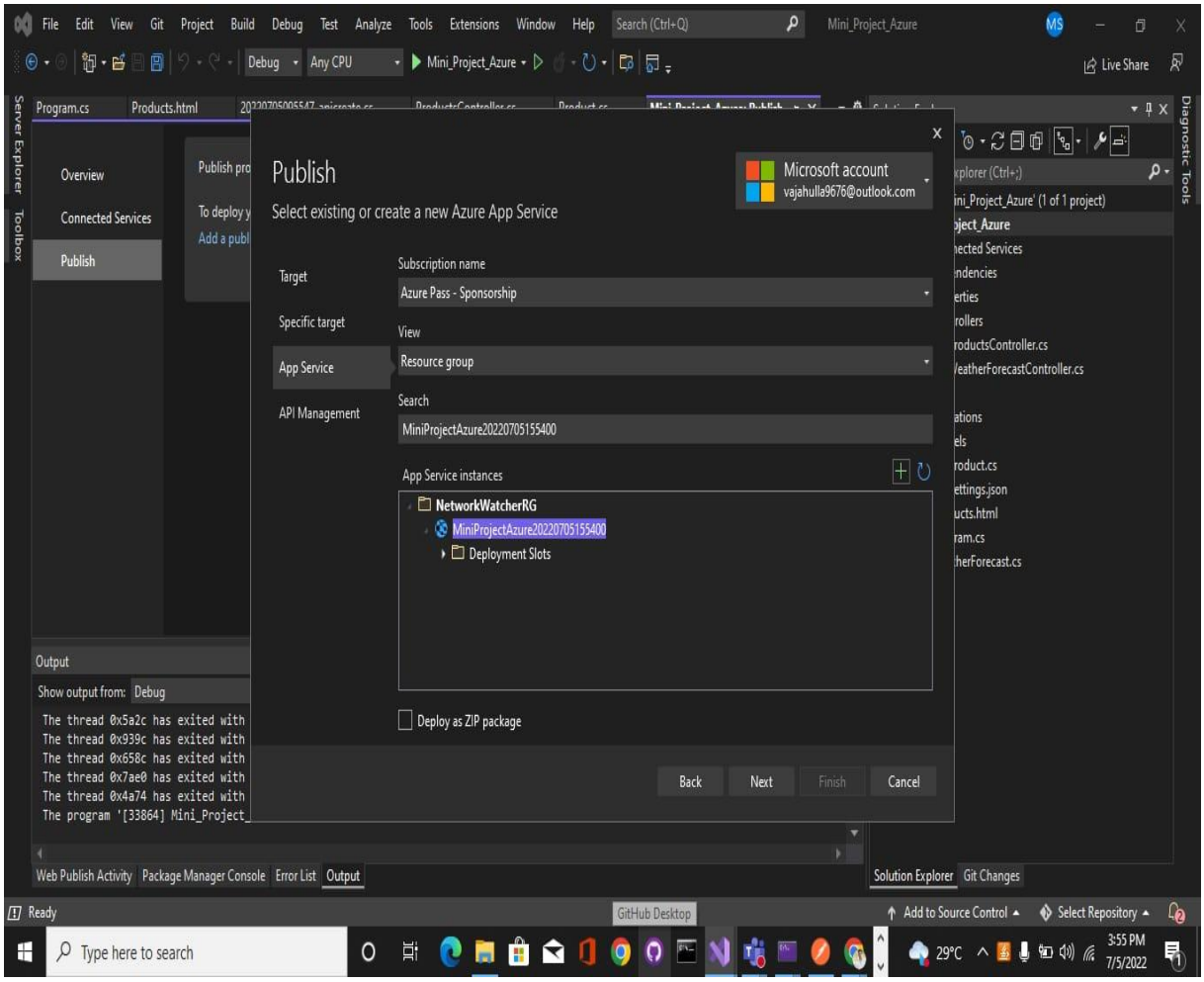
❖Select Create a new Azure App Service.



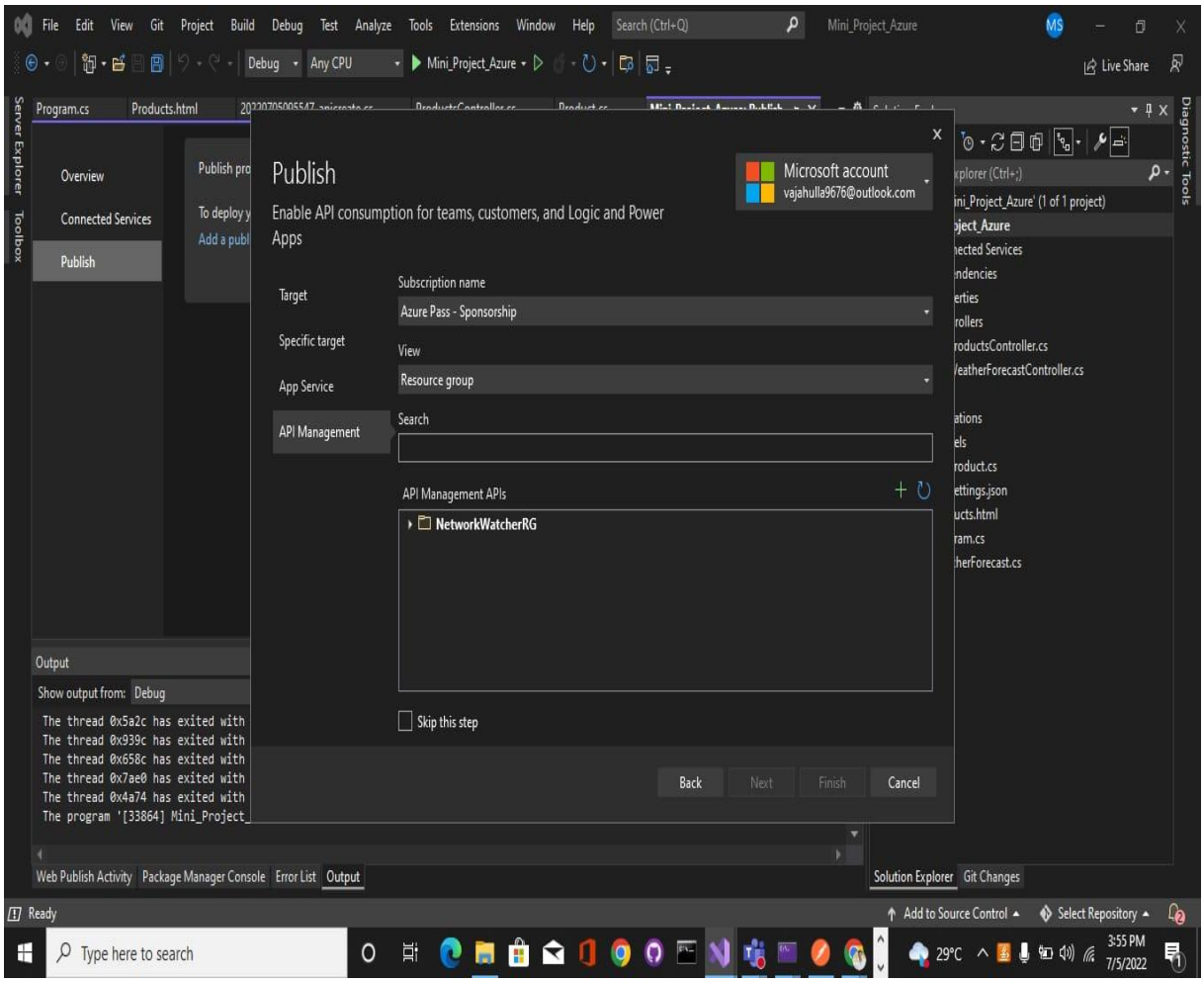
❖The Create App Service dialog appears. The App Name, Resource Group, and App Service Plan entry fields are populated. You can keep these names or change them. Select the Create button.



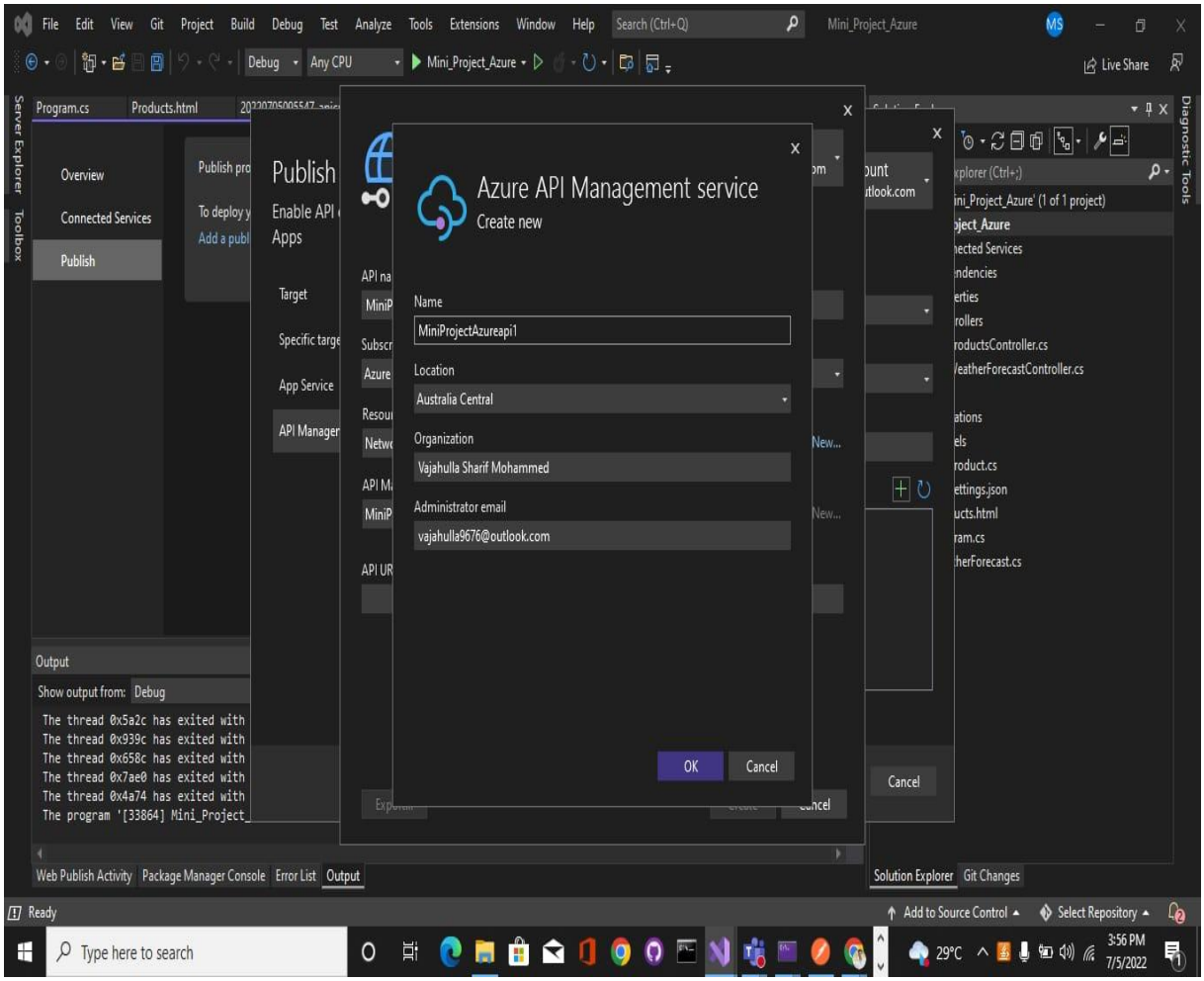
❖After creation is completed, the dialog is automatically closed and the Publish dialog gets focus again. The instance that was created is automatically selected.



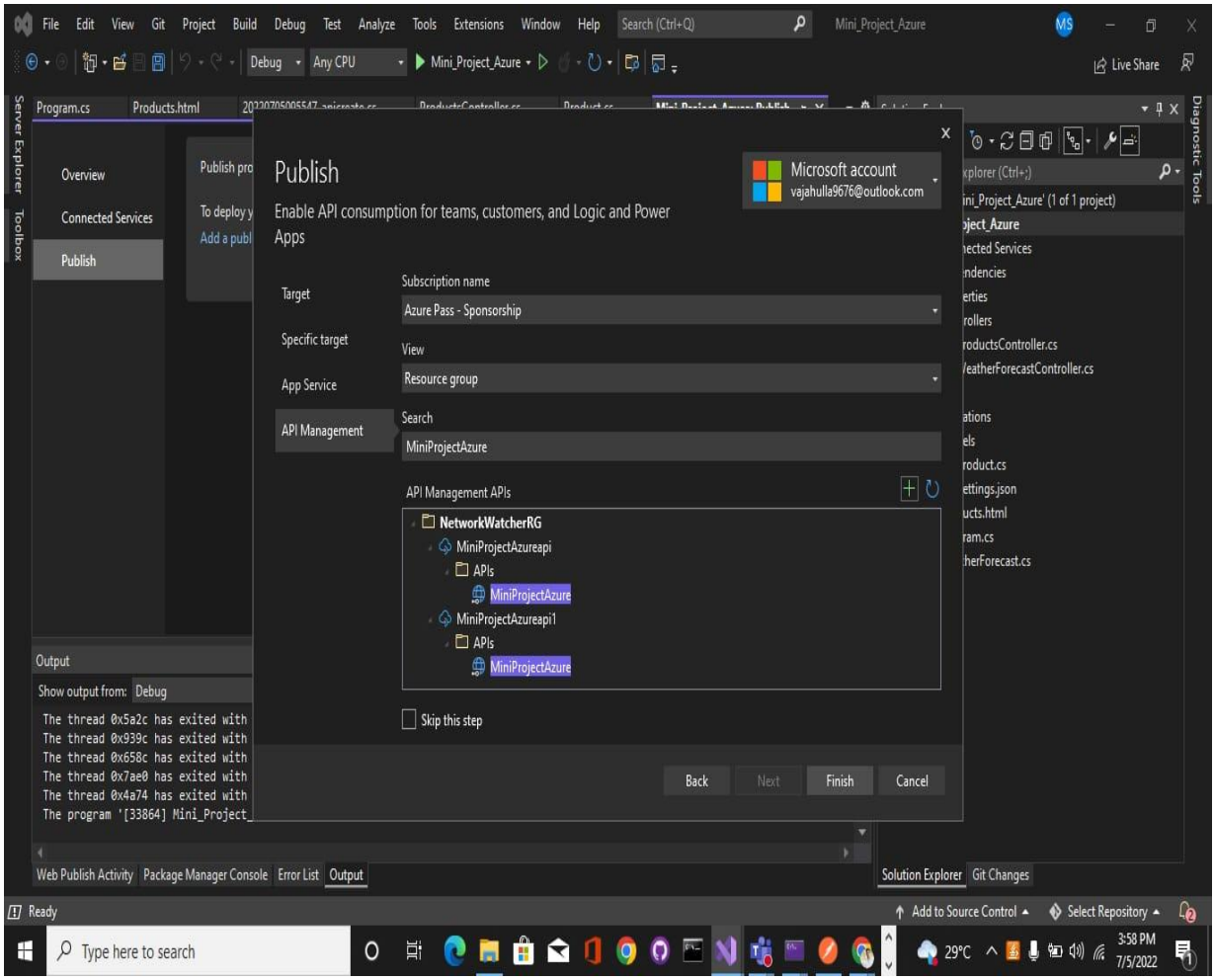
❖The dialog now shows the Azure API Management service to create.



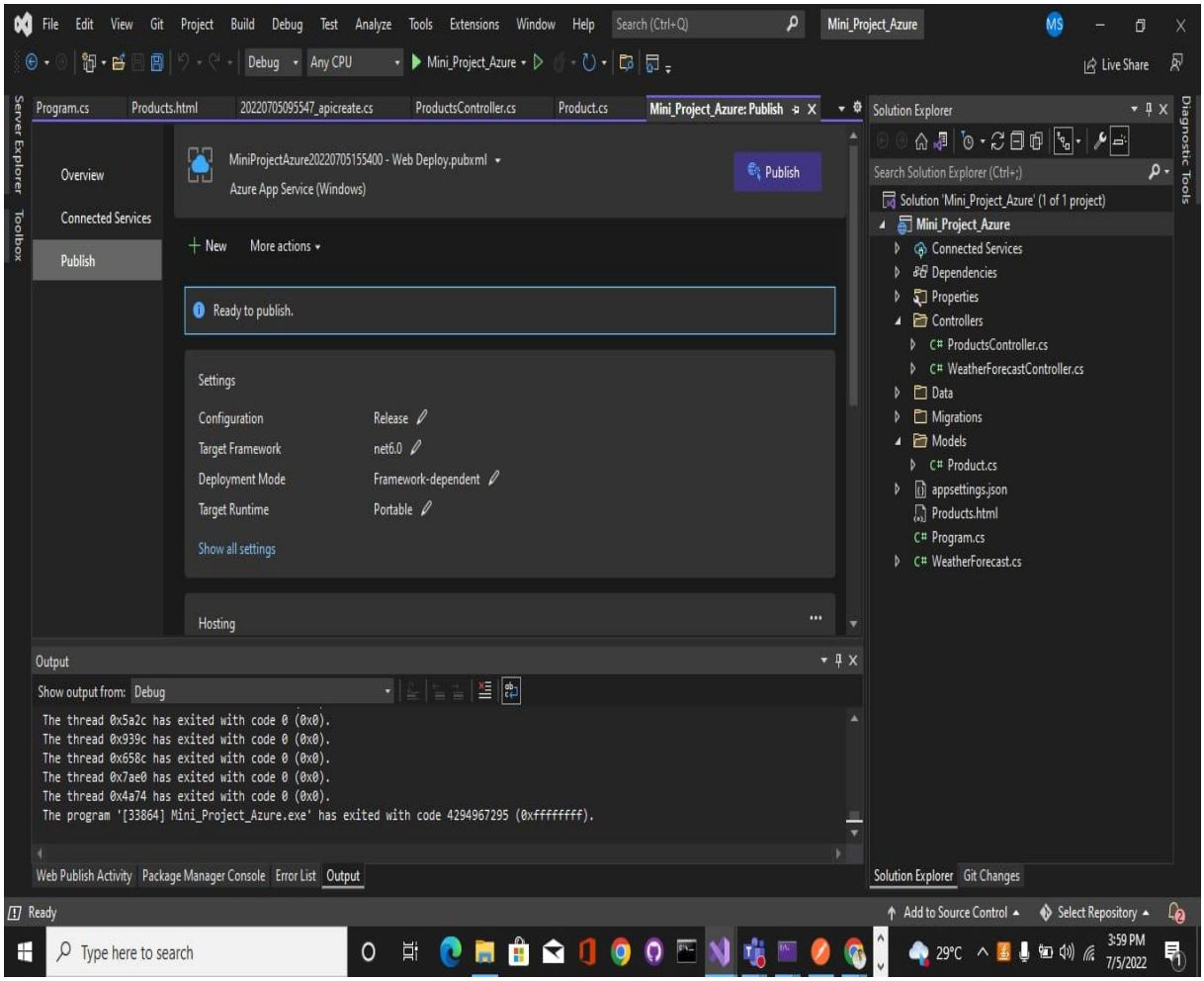
❖The Create the Azure API Management service dialog appears. The App Name, Resource Group, and API Management service entry fields are populated. You can keep these names or change them. Select the Create button.



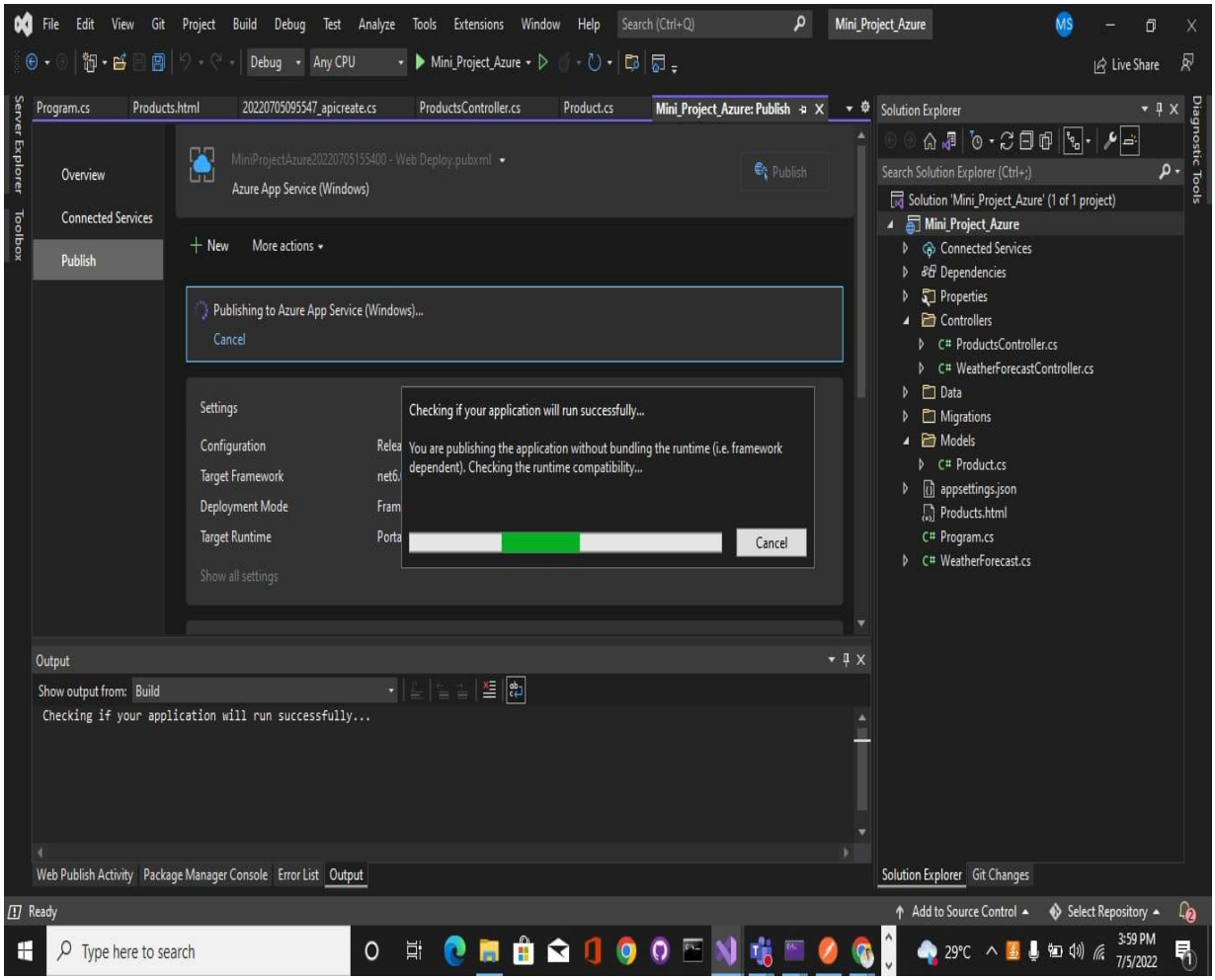
❖After creation is completed, the dialog is automatically closed and the Publish dialog gets focus again. The instance that was created is automatically selected.



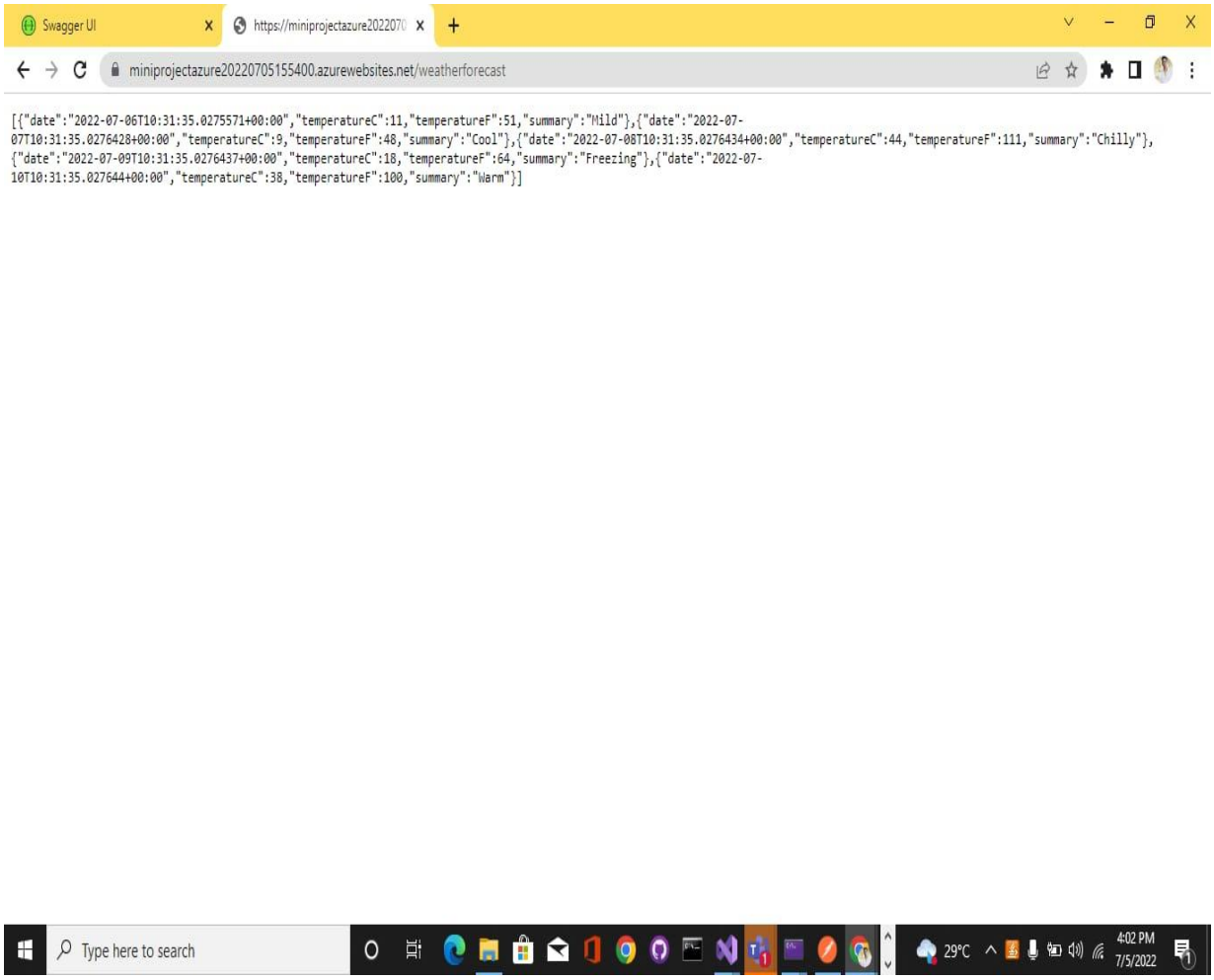
❖The dialog closes and a summary screen appears with information about the publish. Select the Publish button.



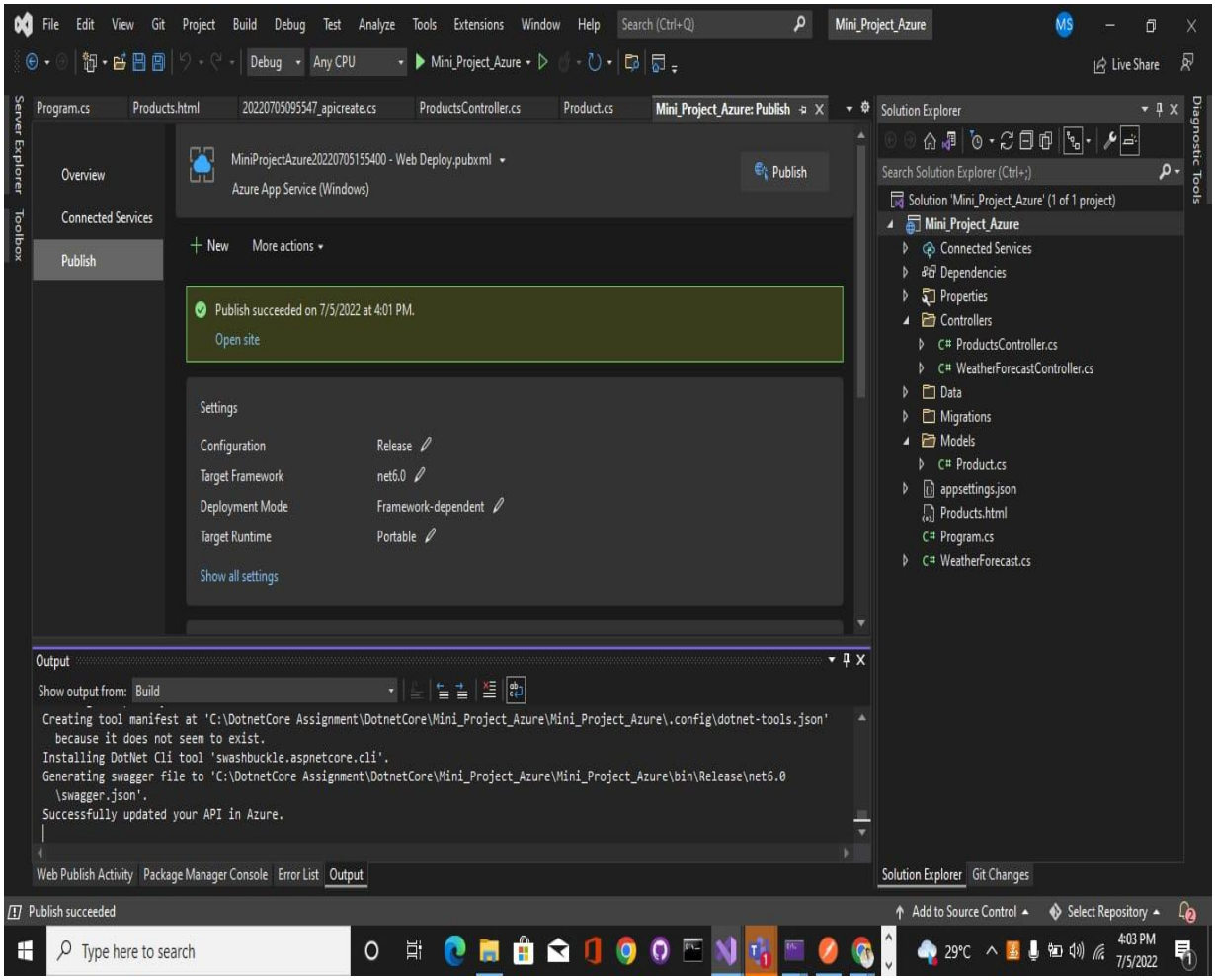
❖Select the Publish button after that it will Publishing to Azure App Service(Window) and checking your application will run Successfully..



❖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 in Azure App Service. You can close that window.

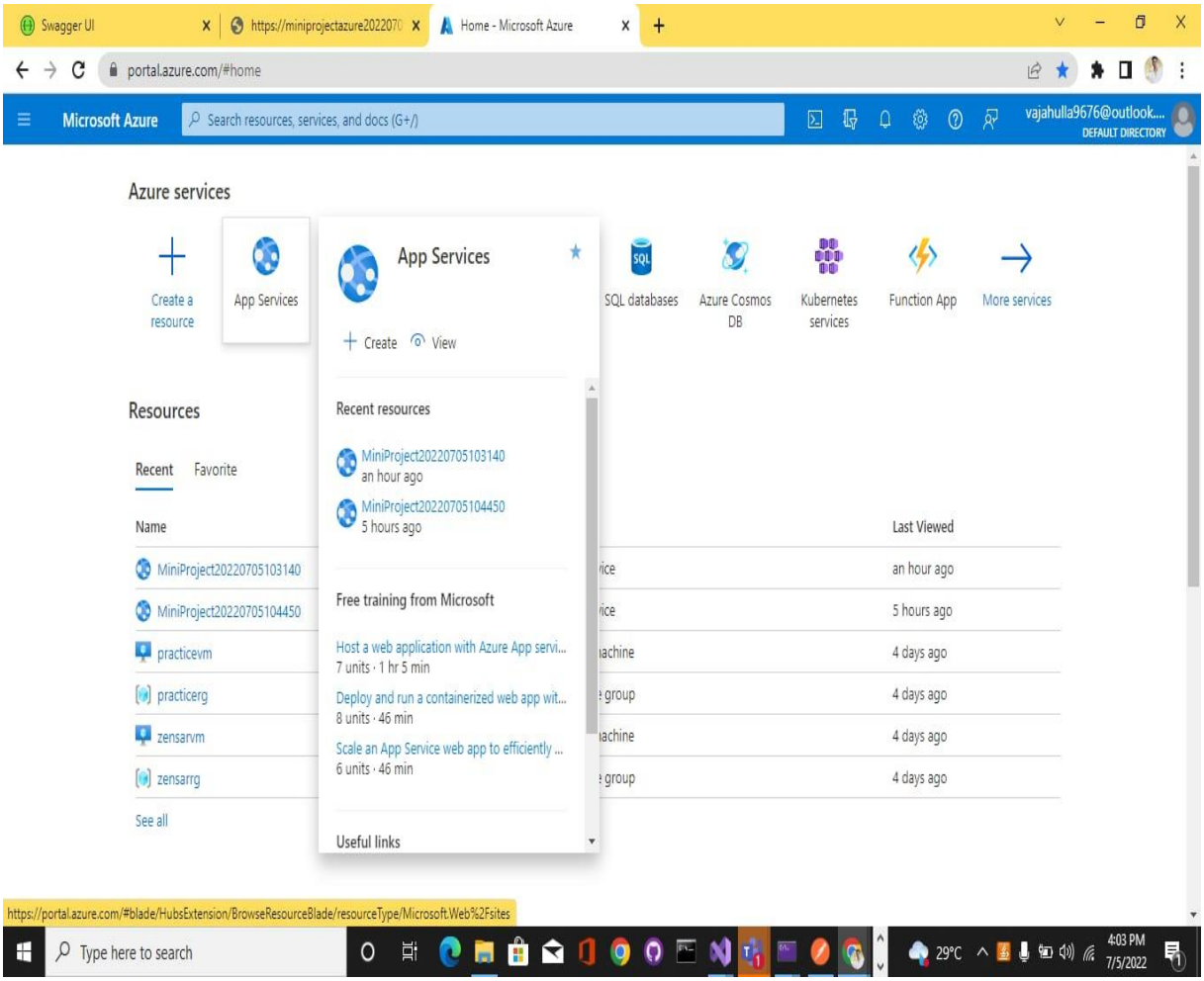


❖Select the Publish button on site to open.

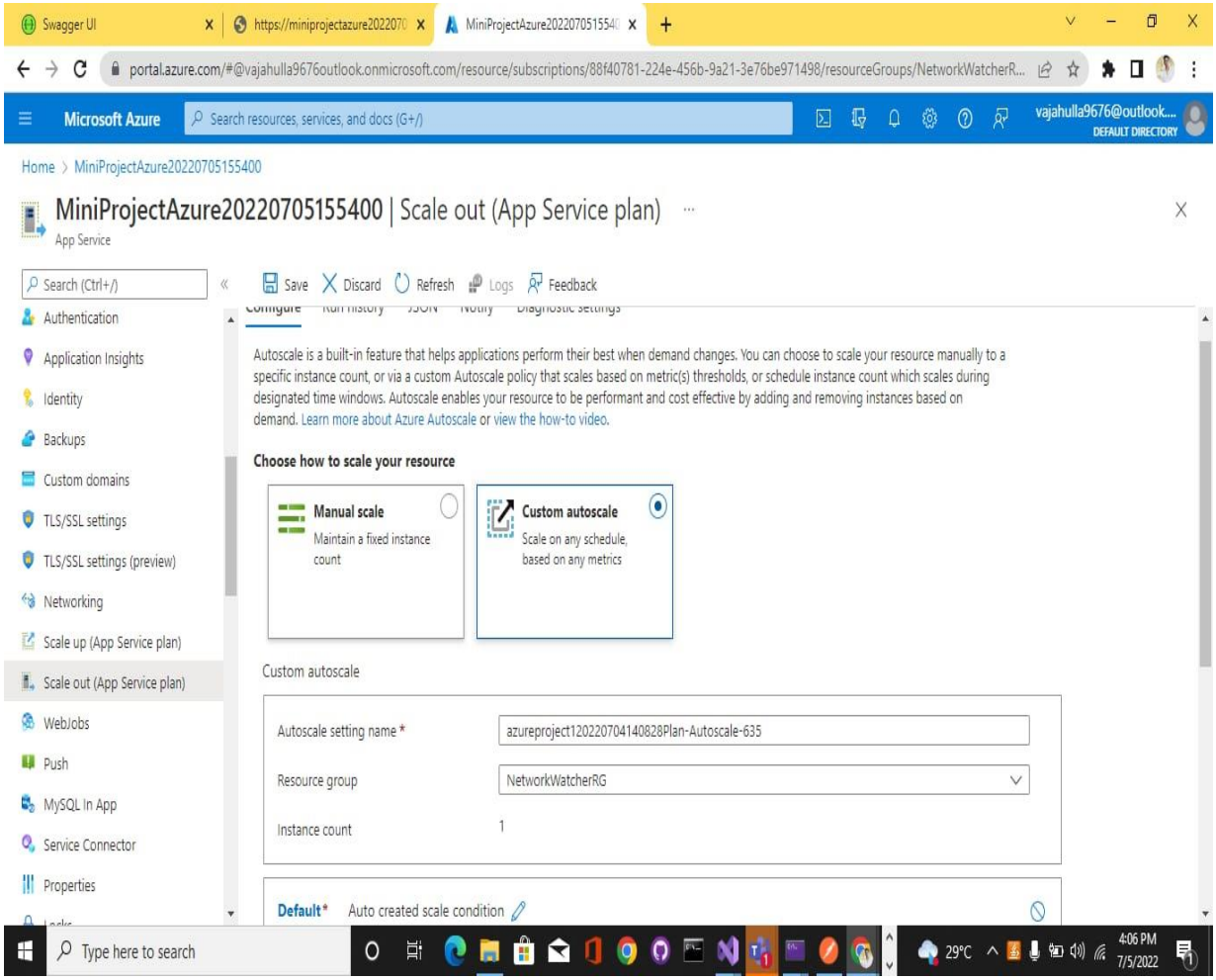


2. Configure Scale out by adding rules for custom scaling

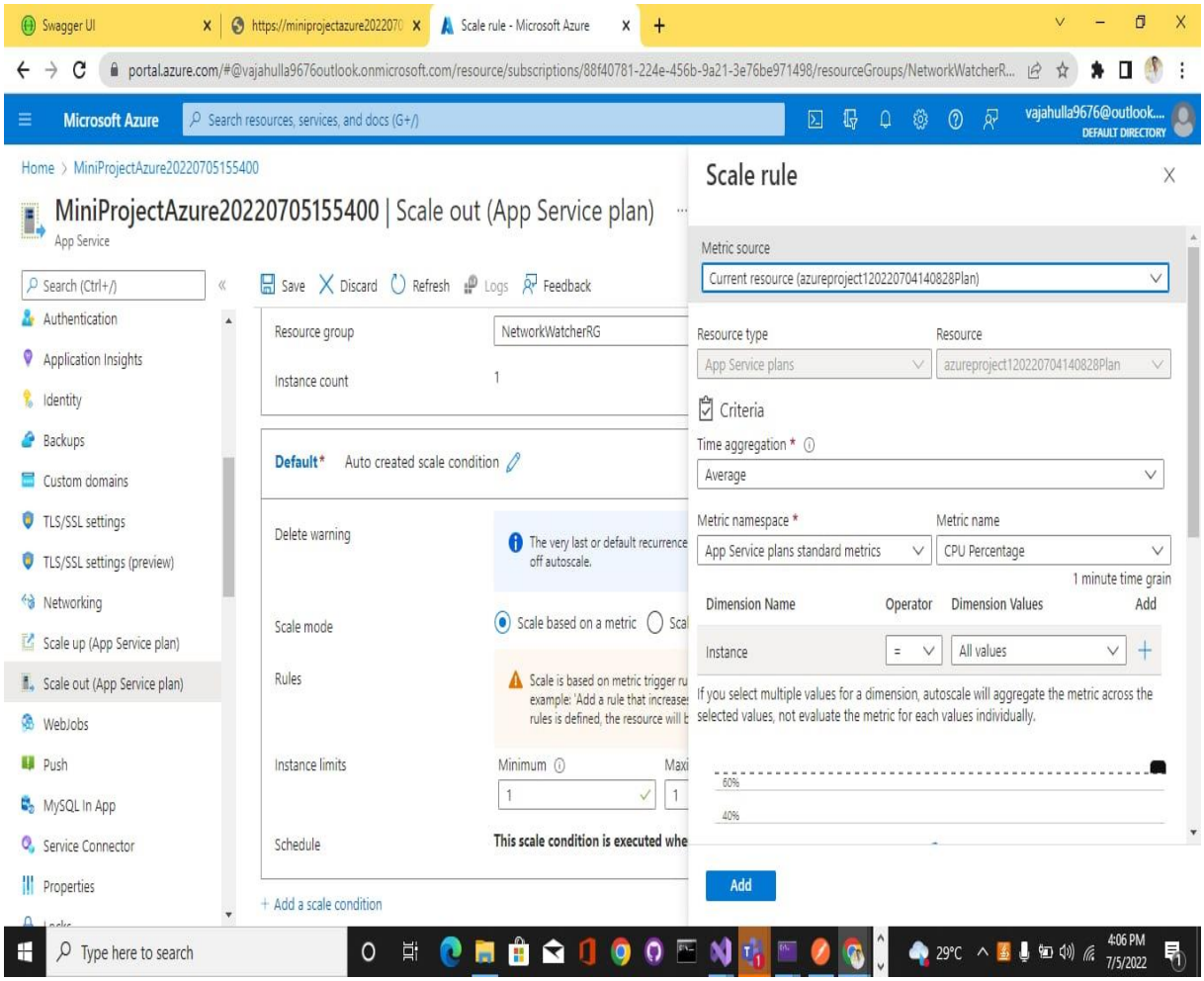
❖ Switch back to the Azure API Management instance in the Azure portal. Refresh the browser window. Select the API you created in the preceding steps. It's now populated and you can explore around.



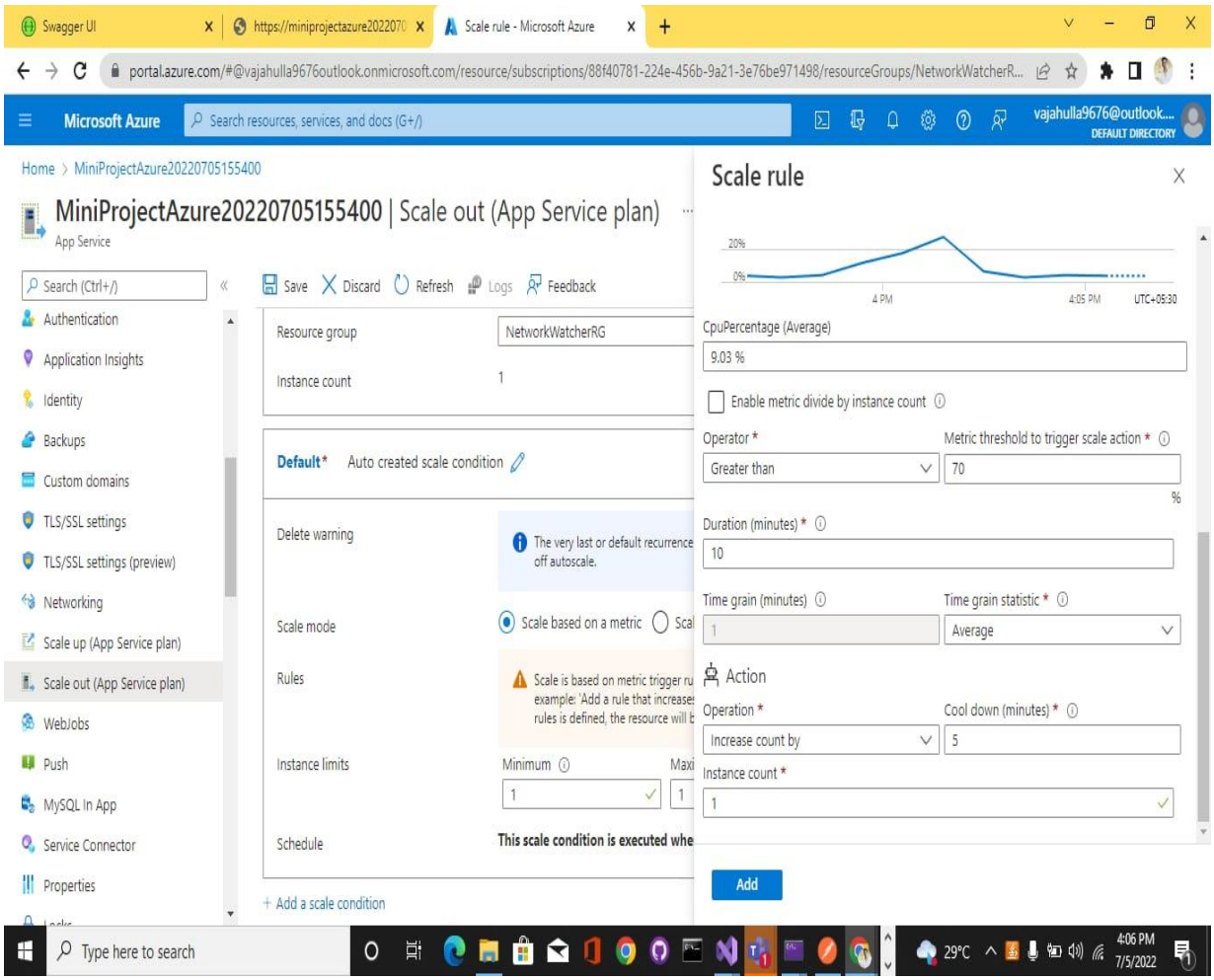
❖Search and select Auto scale in the search bar and Select Custom Auto scale In the Rules section of the default scale condition, select Add a rule. From the Metric source dropdown, select current resource. From Resource Type, select Application Insights.



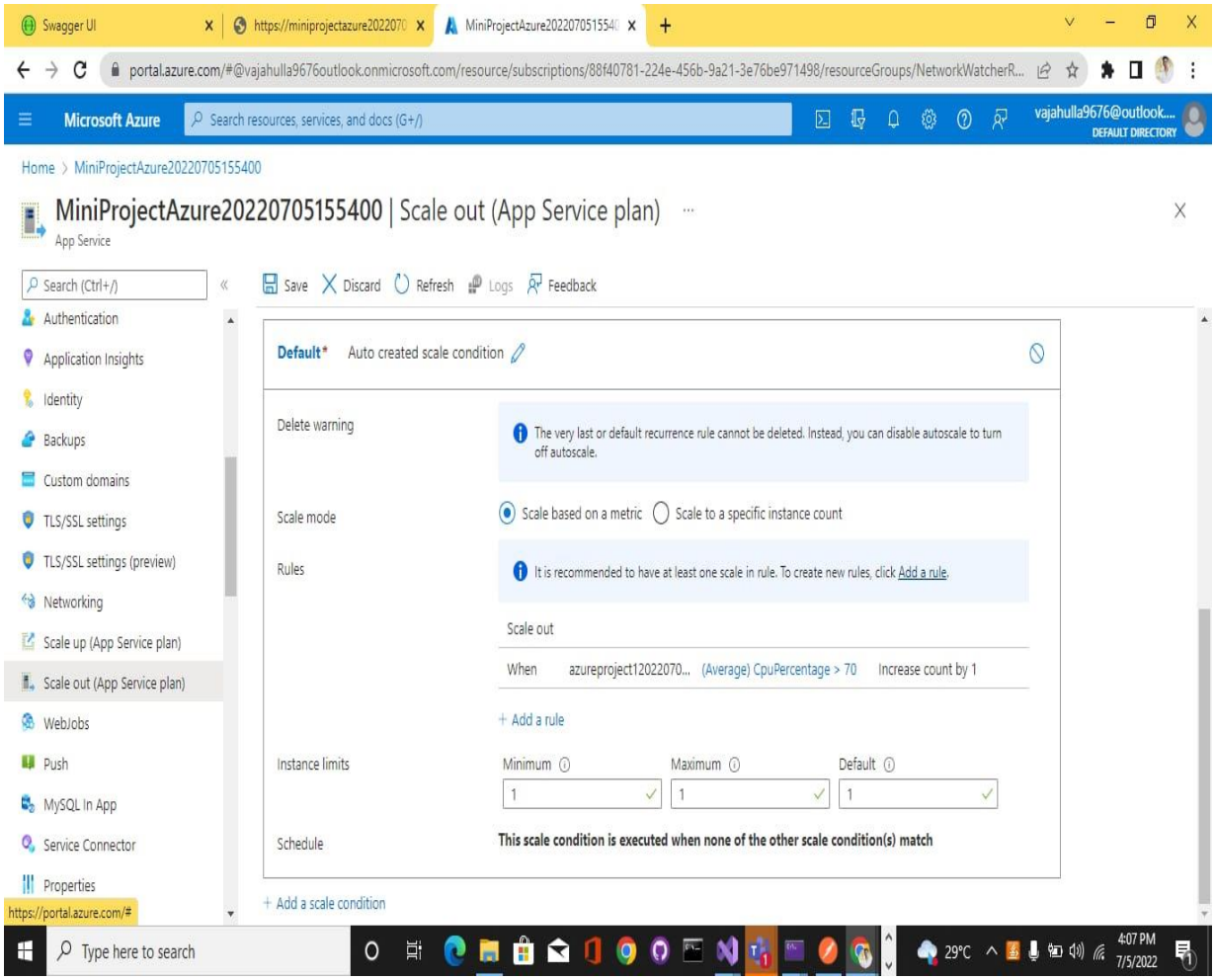
❖From the Resource dropdown, select your App services plan standard metrics. Select a Metric name to CPU Percentage and Select Enable metric divide by instance count so that the number of sessions per instance is measured.From the Operator dropdown, select Greater than.



❖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 1 and 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 section and Select Add.

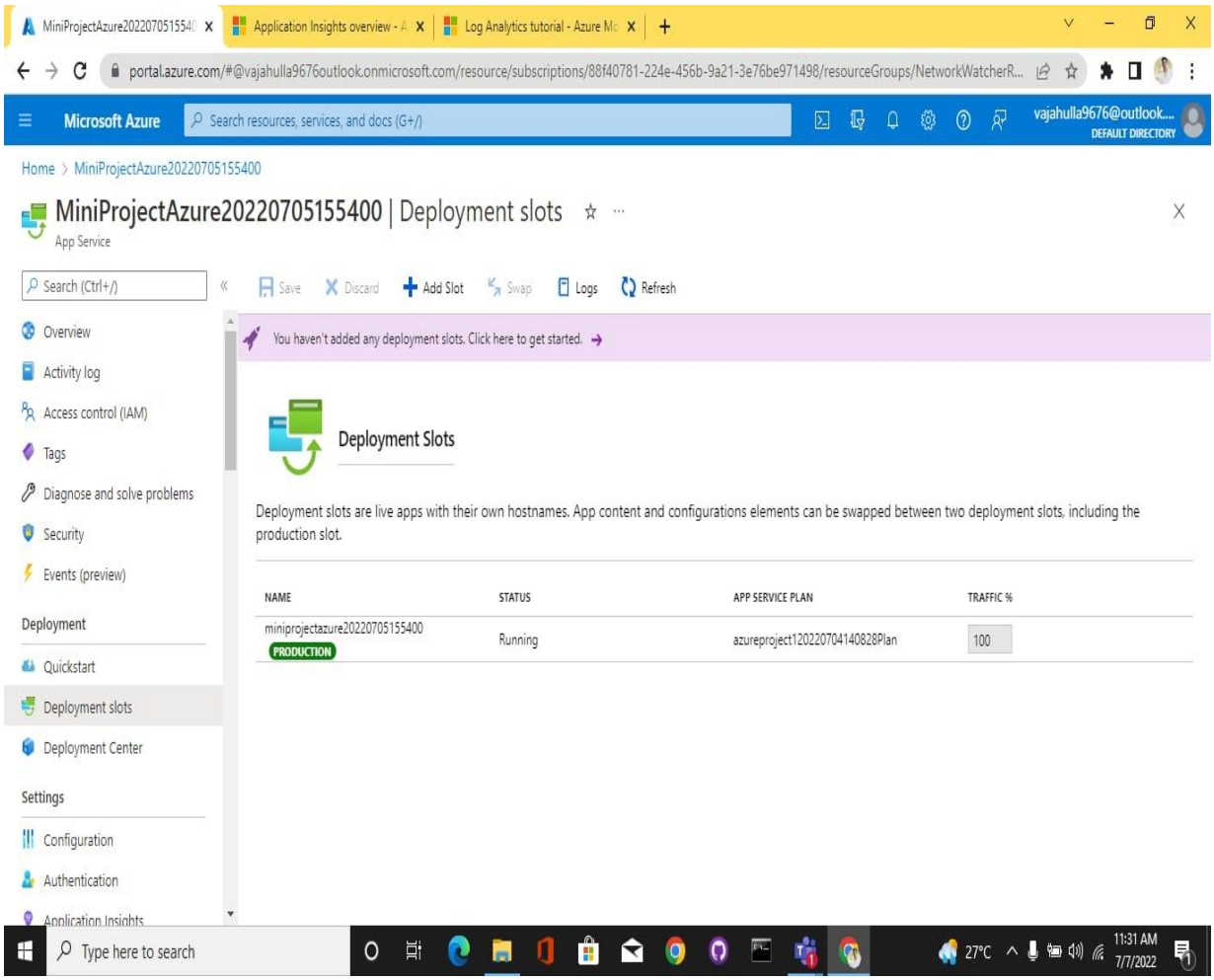


❖After adding the scale it show Rules.

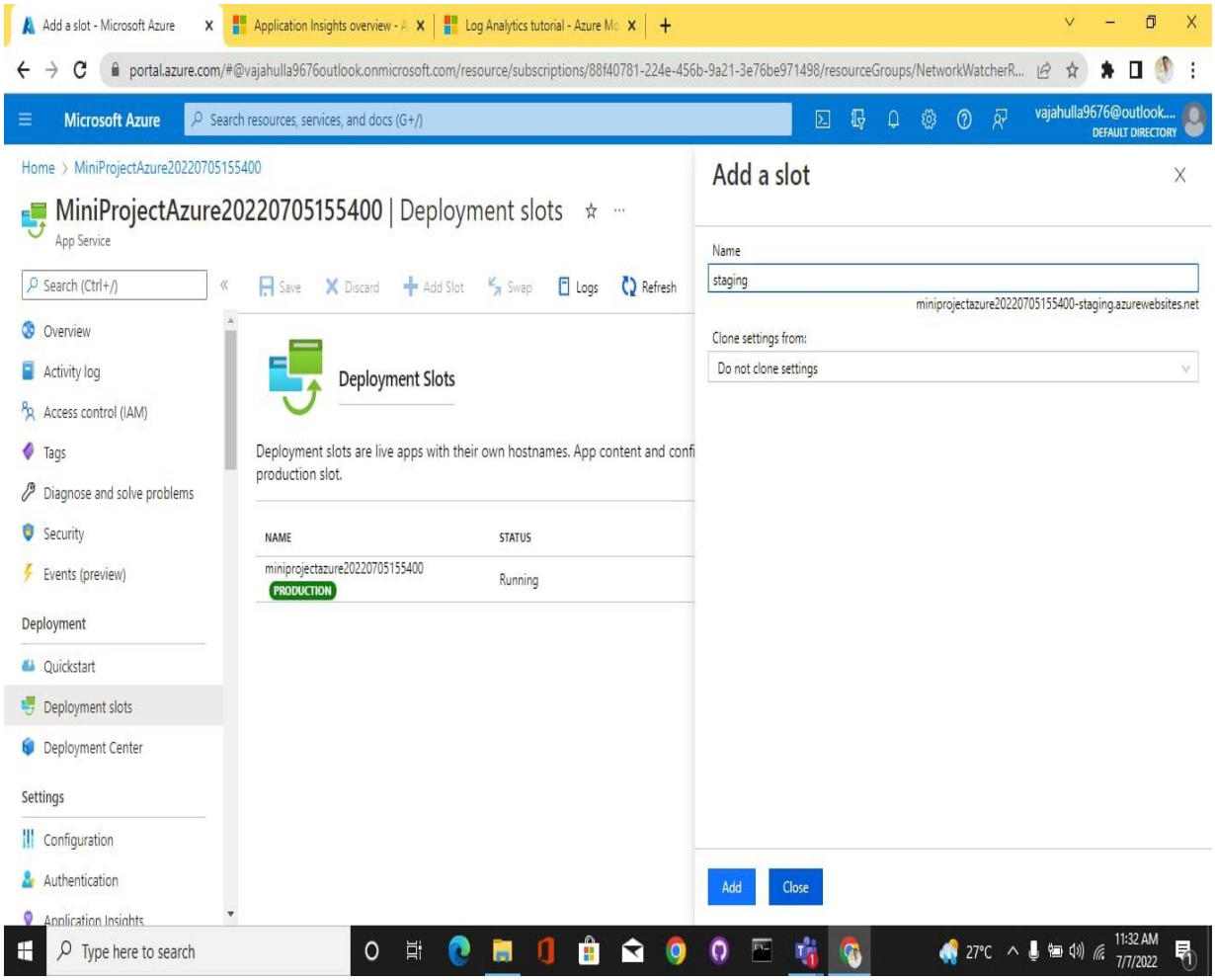


3.Configure Deployment slots for staging and production

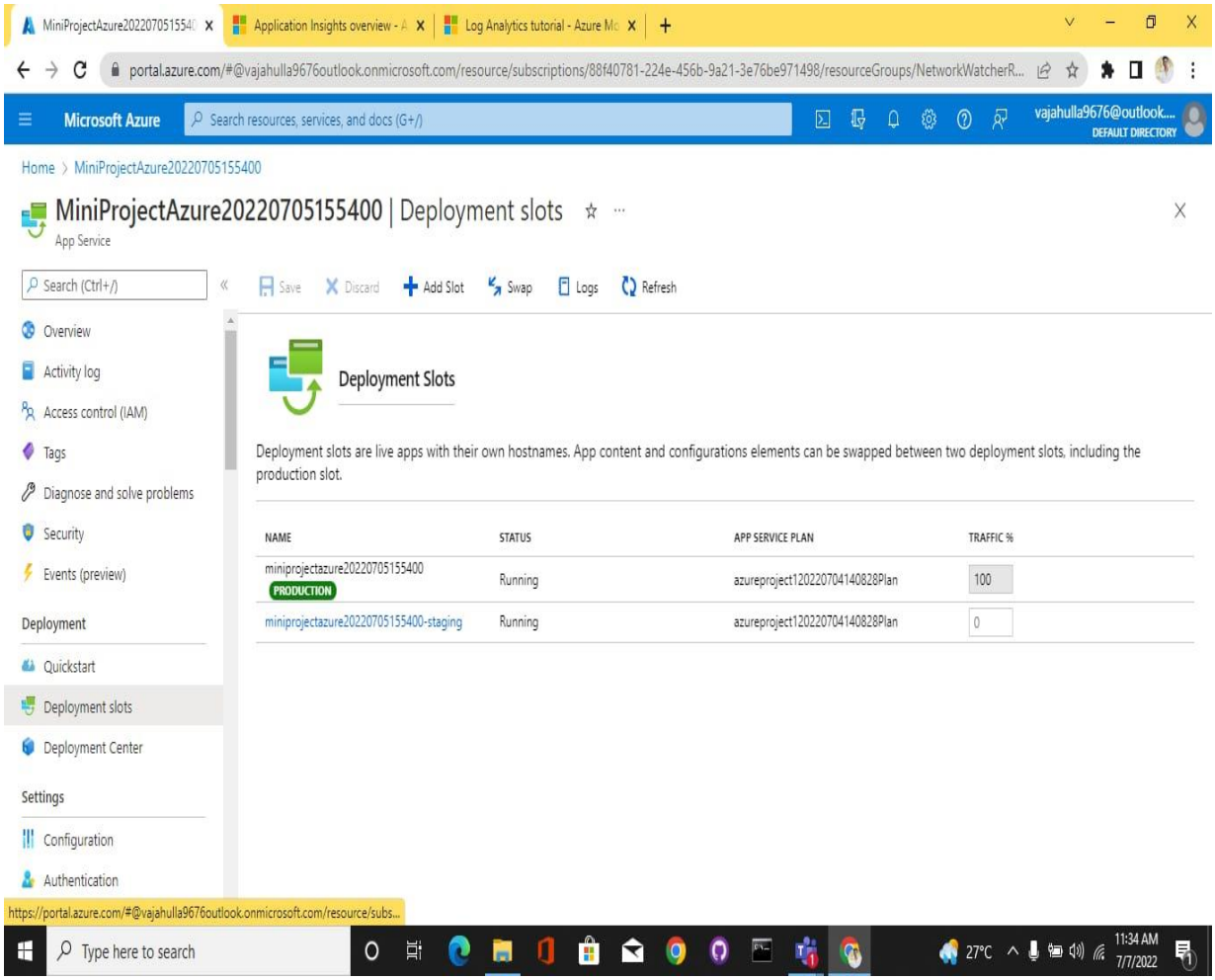
Azure Functions deployment slots allow your function app to run different instances called "slots". Slots are different environments exposed via a publicly available endpoint. One app instance is always mapped to the production slot, and you can swap instances assigned to a slot on demand. Function apps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



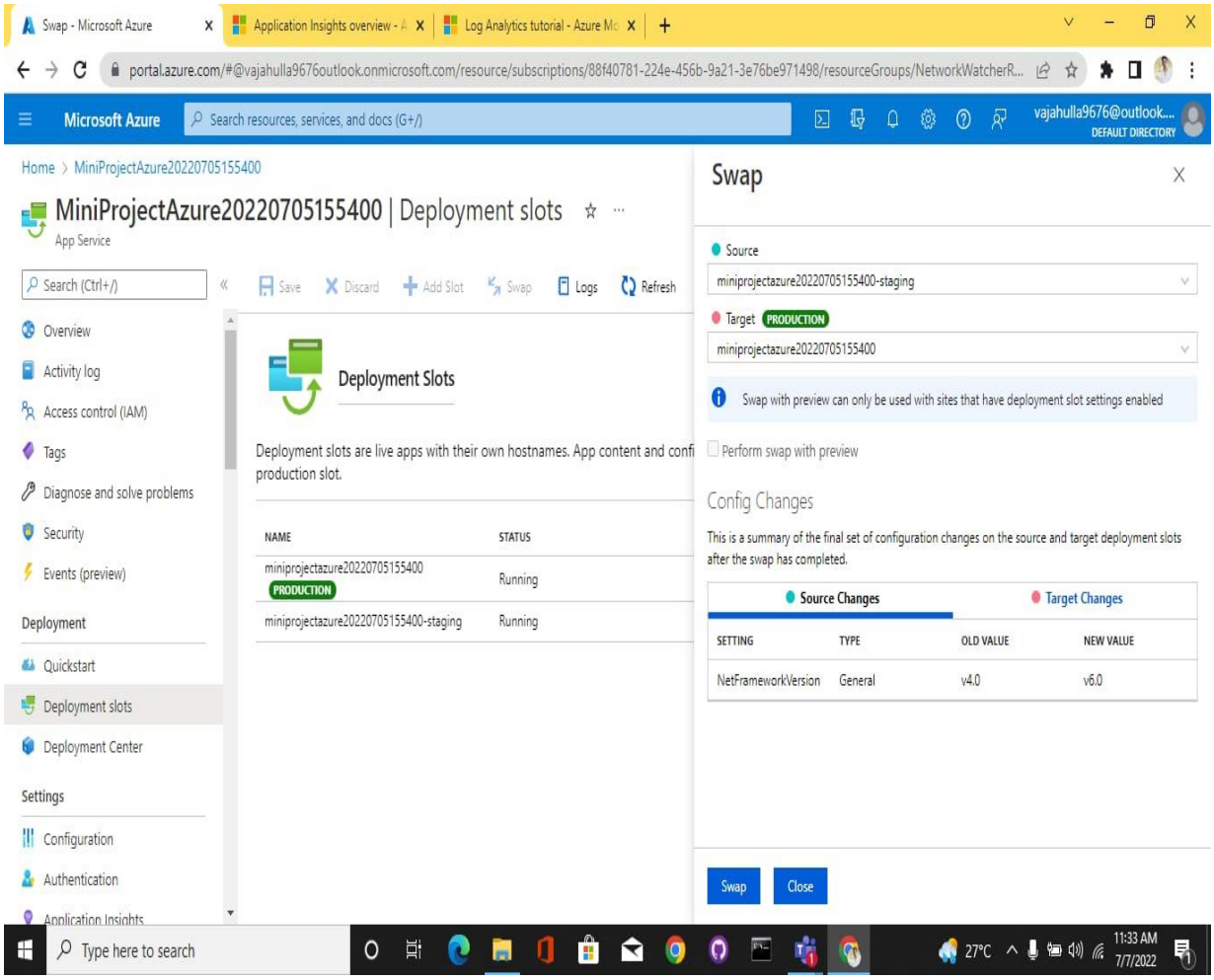
❖Add the Slot and Name as Staging and then Add the Slot .



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

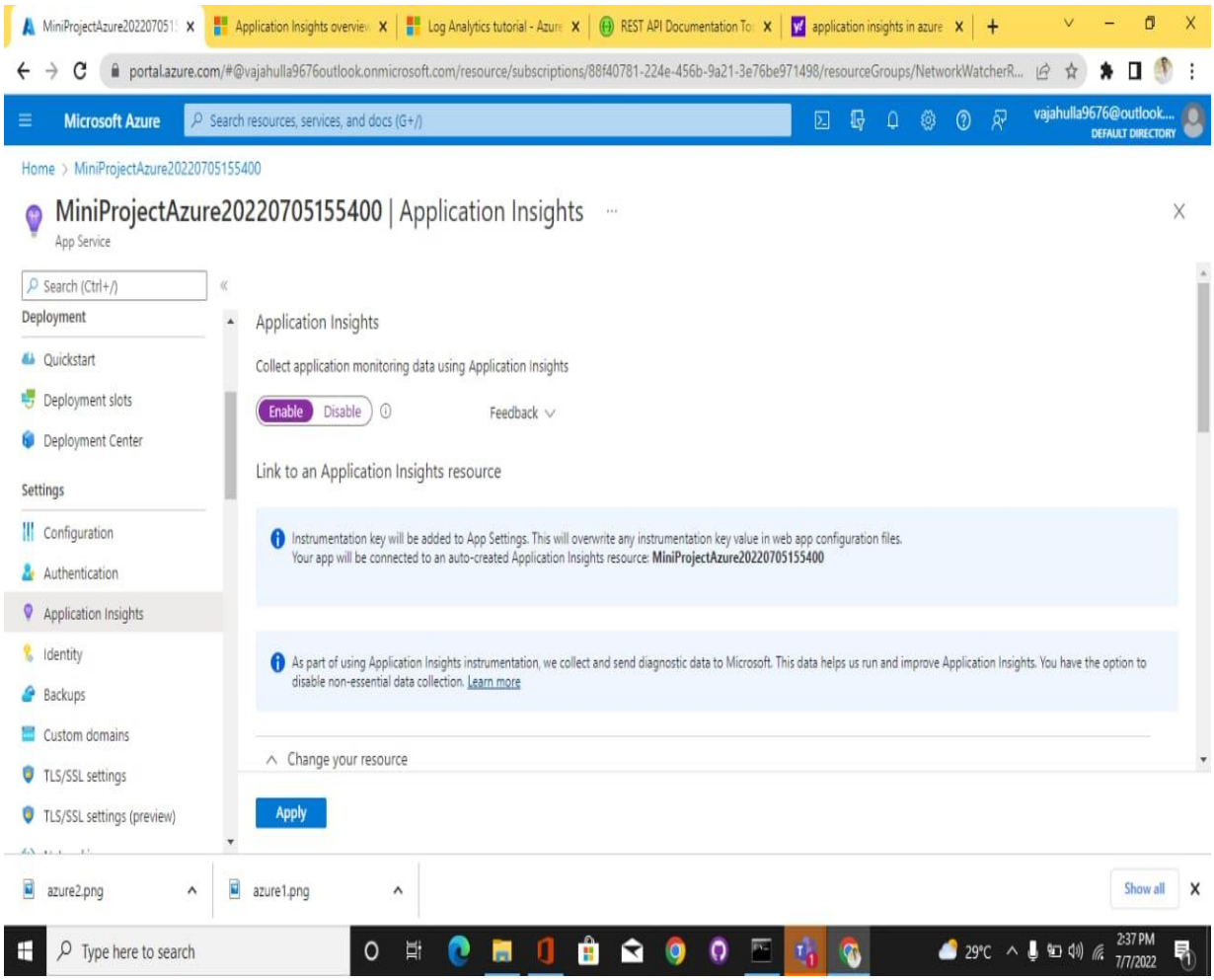


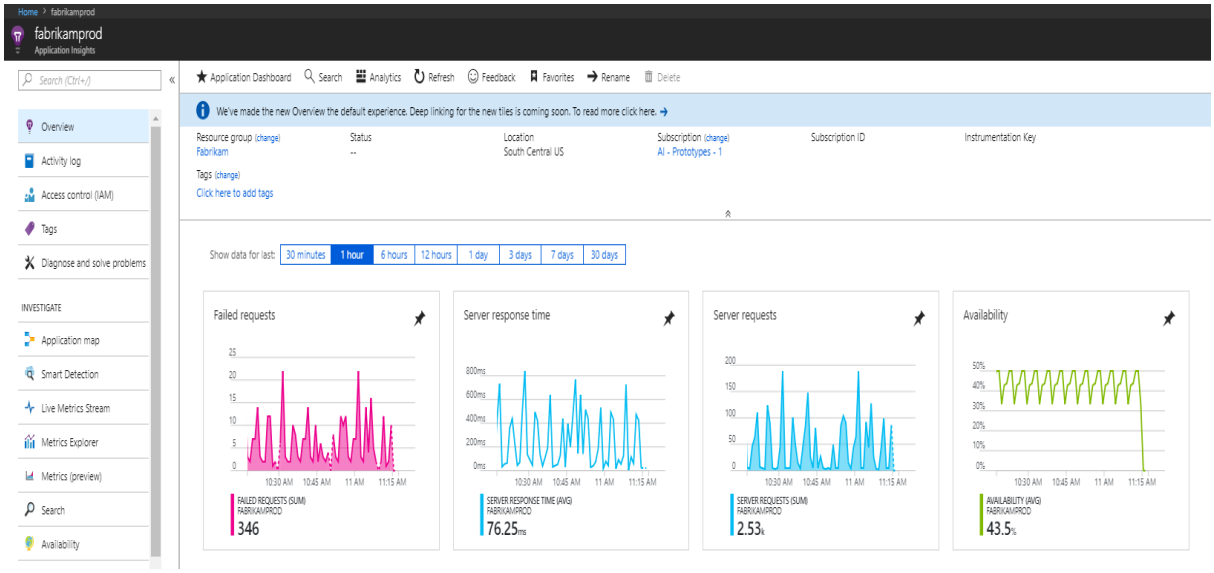
❖Select Deployment slots, and then select Swap and Verify the configuration settings for your swap and select Swap.

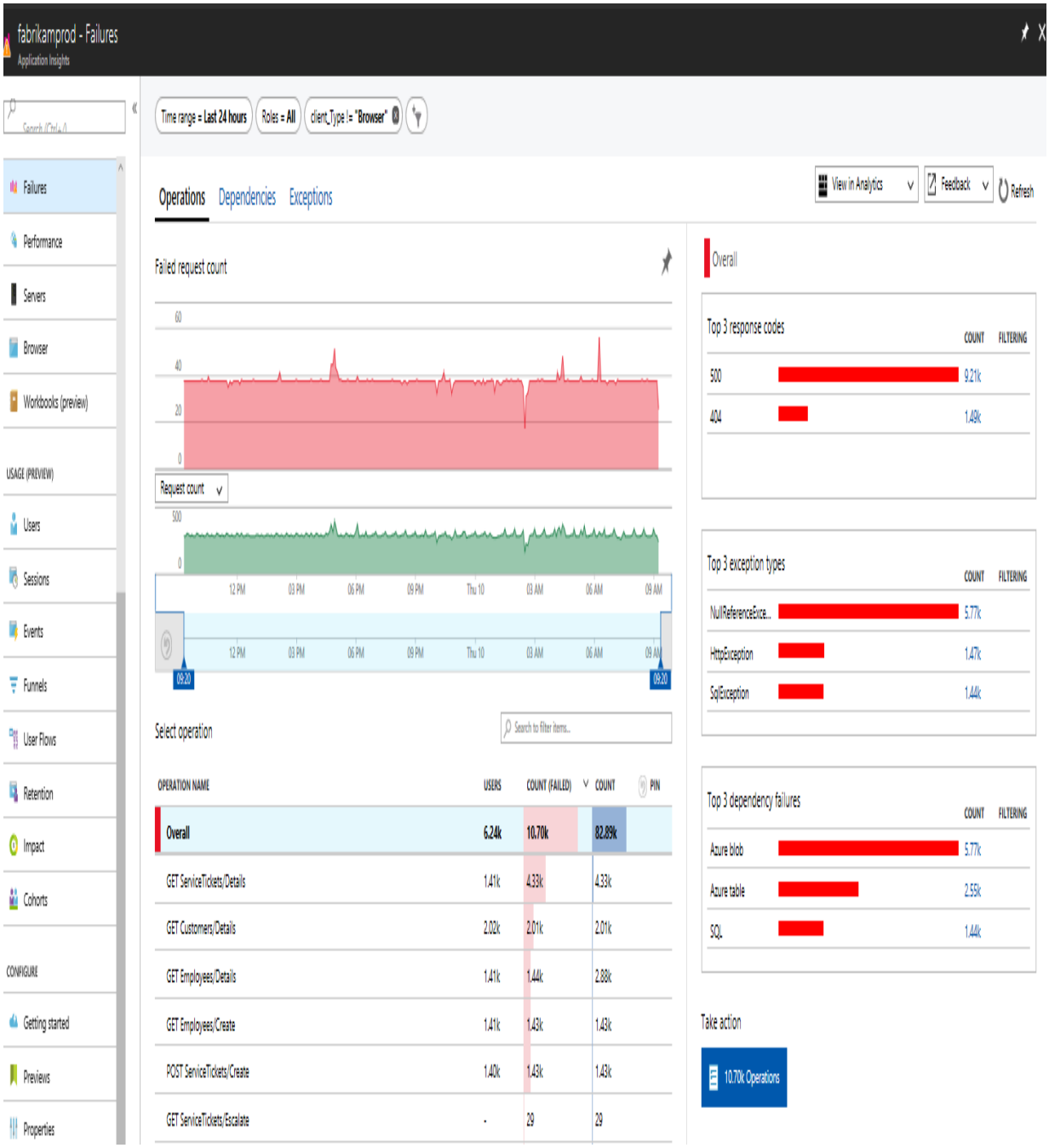


4. Configure Application Insights for the project

❖ Select the Application Insights




Home › fabrikamprod
fabrikamprod
Application Insights
Search (Ctrl+/)
Application Dashboard
Search
Analytics
Refresh
Feedback
Favorites
Rename
Delete
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
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%

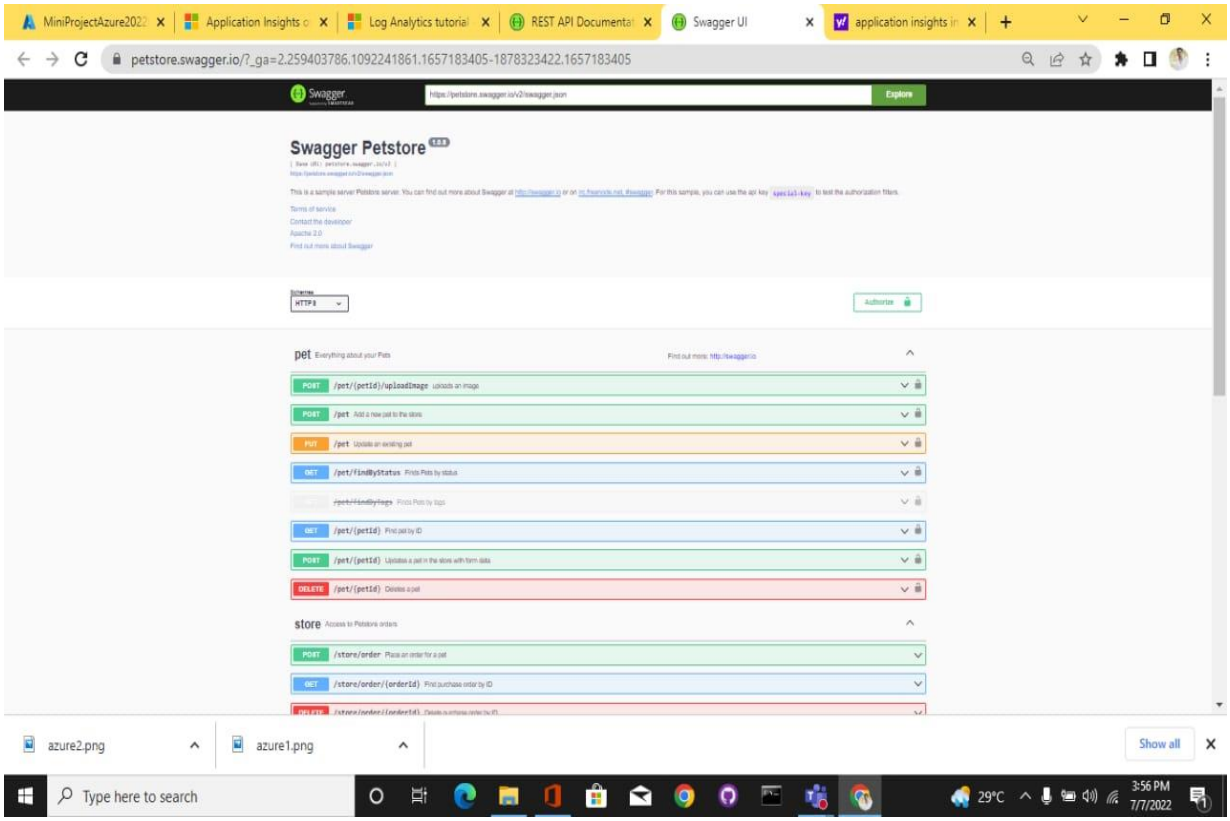

fabrikamprod - Failures
Application Insights
Time range = Last 24 hours
Roles = All
client_Type != "Browser"
Failures
Performance
Servers
Browser
Workbooks (preview)
USAGE (PREVIEW)
Users
Sessions
Events
Funnels
User Flows
Retention
Impact
Cohorts
CONFIGURE
Getting started
Previews
Properties
Operations
Dependencies
Exceptions
View in Analytics
Feedback
Refresh
Failed request count
Request count
Select operation
Search to filter items...
OPERATION NAME
USERS
COUNT (FAILED)
COUNT
PIN
Overall 6.24k 10.70k 82.89k
GET ServiceTickets/Details 1.41k 4.33k 4.33k
GET Customers/Details 2.02k 2.01k 2.01k
GET Employees/Details 1.41k 1.44k 2.88k
GET Employees/Create 1.41k 1.43k 1.43k
POST ServiceTickets/Create 1.40k 1.43k 1.43k
GET ServiceTickets/Escalate - 29 29
Overall
Top 3 response codes
COUNT
FILTERING
500 9.21k
404 1.49k
Top 3 exception types
NullReferenceExce... 5.77k
HttpException 1.47k
SqlException 1.44k
Top 3 dependency failures
Azure blob 5.77k
Azure table 2.55k
SQL 1.44k
Take action
10.70k Operations

5. Configure Swagger for the API

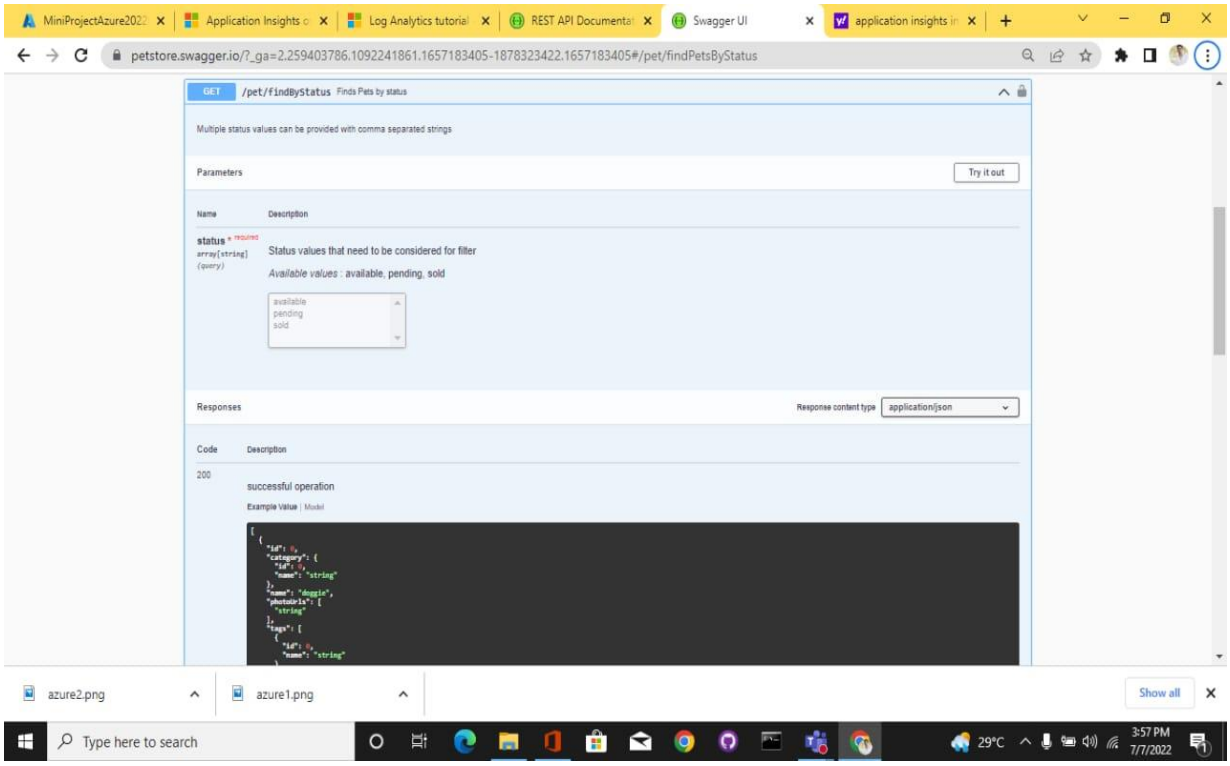
Swagger UI allows anyone be it your development team or your end consumers to visualize and interact with the API's resources without having any of the implementation logic in place. It's automatically generated from your Open API (formerly known as Swagger) Specification, with the visual documentation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

Advantages :

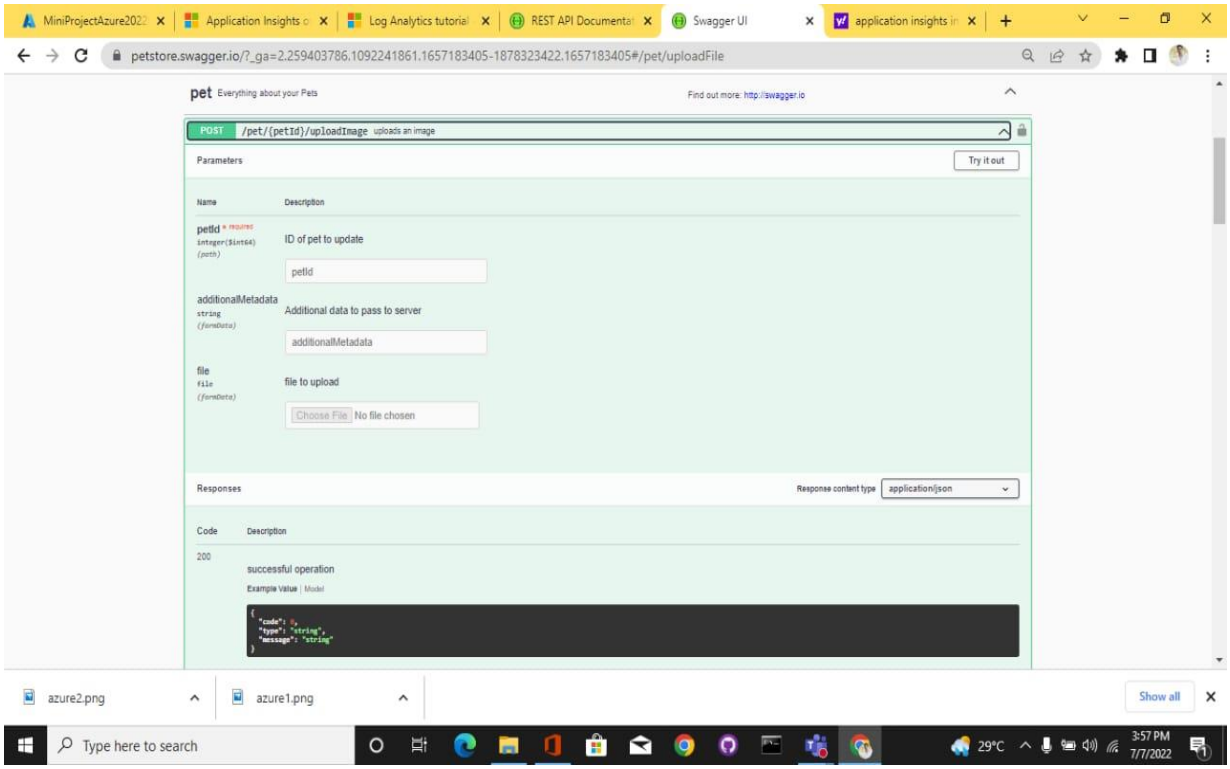
- Dependency Free - The UI works in any development environment, be it locally or in the web
- Human Friendly - Allow end developers to effortlessly interact and try out every single operation your API exposes for easy consumption
- Easy to Navigate - Quickly find and work with resources and endpoints 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



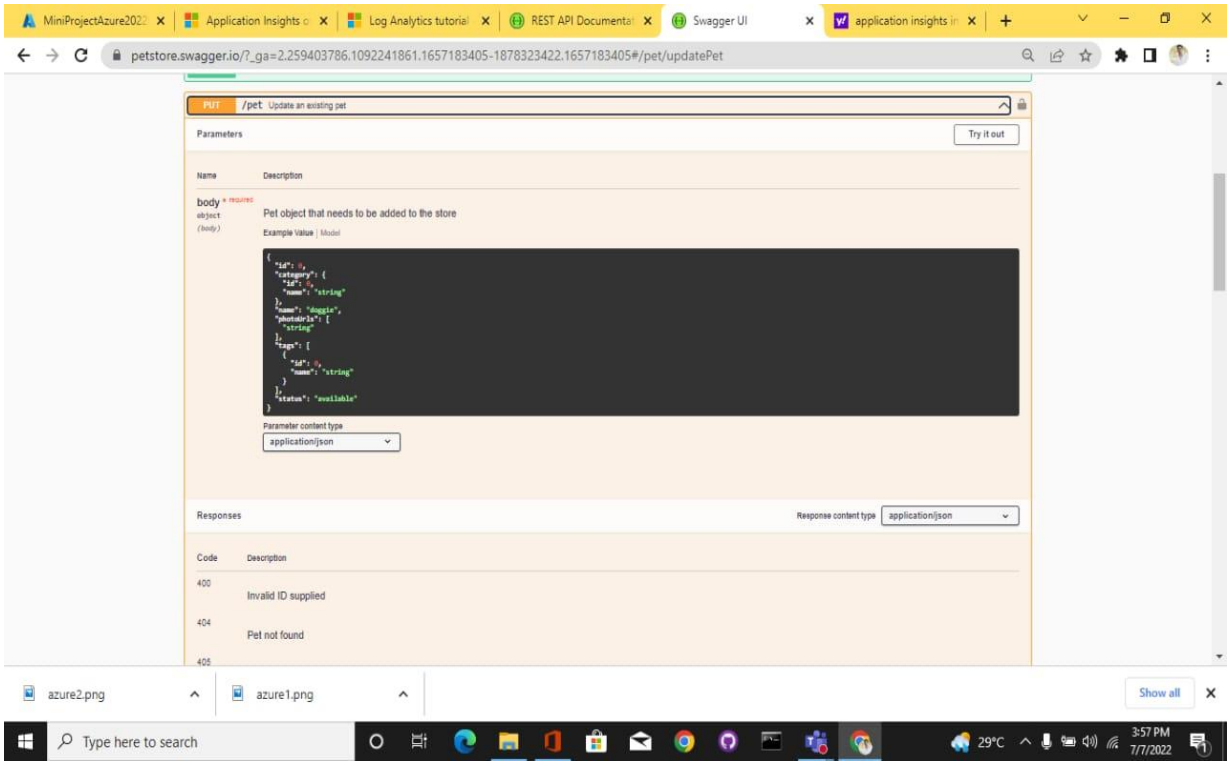
GET


petstore.swagger.io/?_ga=2.259403786.1092241861.1657183405-1878323422.1657183405#/pet/findPetsByStatus
GET /pet/findByStatus Finds Pets by status
Multiple status values can be provided with comma separated strings
Parameters
Try it out
Name
Description
status * required
array[string]
(query)
Status values that need to be considered for filter
Available values : available, pending, sold
available
pending
sold
Responses
Response content type
application/json
Code
Description
200
successful operation
Example Value | Model
azure2.png
azure1.png
Show all
Type here to search
29°C
3:57 PM
7/7/2022

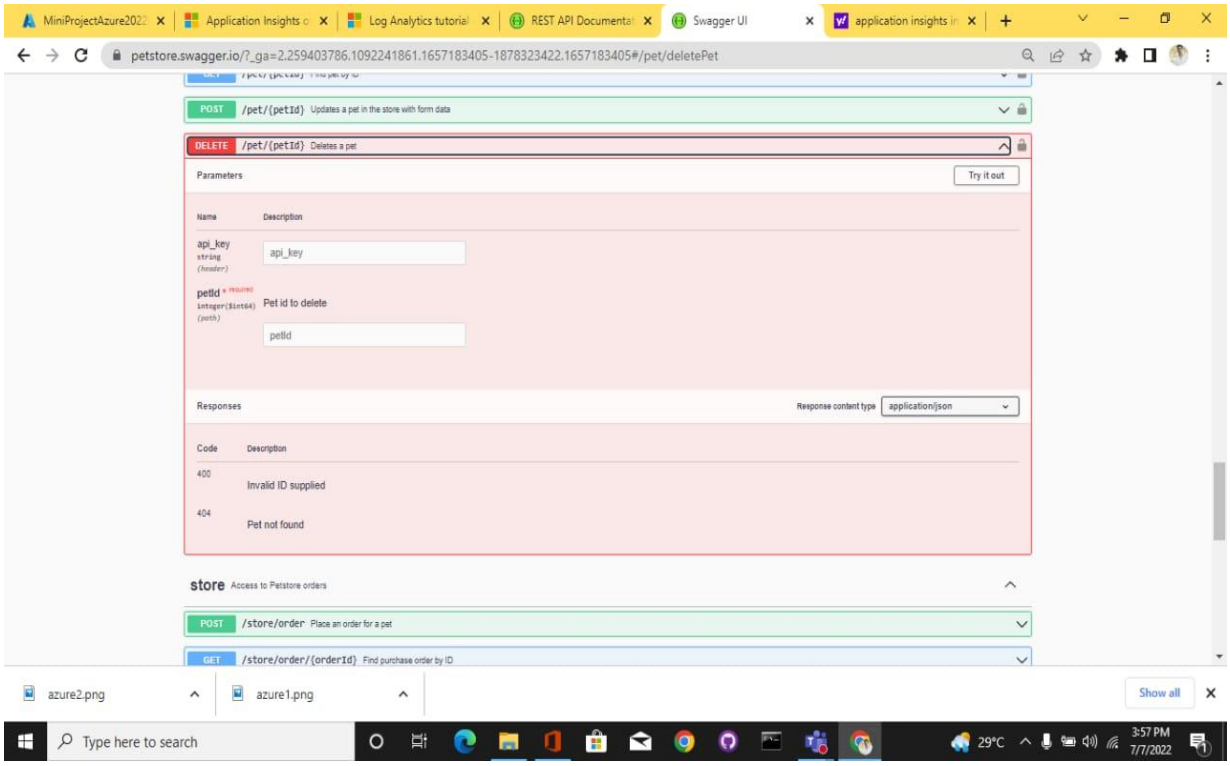
POST


petstore.swagger.io/?_ga=2.259403786.1092241861.1657183405-1878323422.1657183405#/pet/uploadFile
pet Everything about your Pets
Find out more: http://swagger.io
POST /pet/{petId}/uploadImage uploads an image
Parameters
Try it out
Name
Description
petId * required
integer($int64)
(path)
ID of pet to update
petId
additionalMetadata
string
(formData)
Additional data to pass to server
additionalMetadata
file
file
(formData)
file to upload
Choose File
No file chosen
Responses
Response content type
application/json
Code
Description
200
successful operation
Example Value | Model
azure2.png
azure1.png
Show all
Type here to search
29°C
3:57 PM
7/7/2022

PUT


petstore.swagger.io/?_ga=2.259403786.1092241861.1657183405-1878323422.1657183405#/pet/updatePet
PUT /pet Update an existing pet
Parameters
Try it out
Name
Description
body
Pet object that needs to be added to the store
Example Value | Model
Parameter content type
application/json
Responses
Response content type
application/json
Code
Description
400
Invalid ID supplied
404
Pet not found
azure2.png
azure1.png
Show all
Type here to search
29°C
3:57 PM
7/7/2022

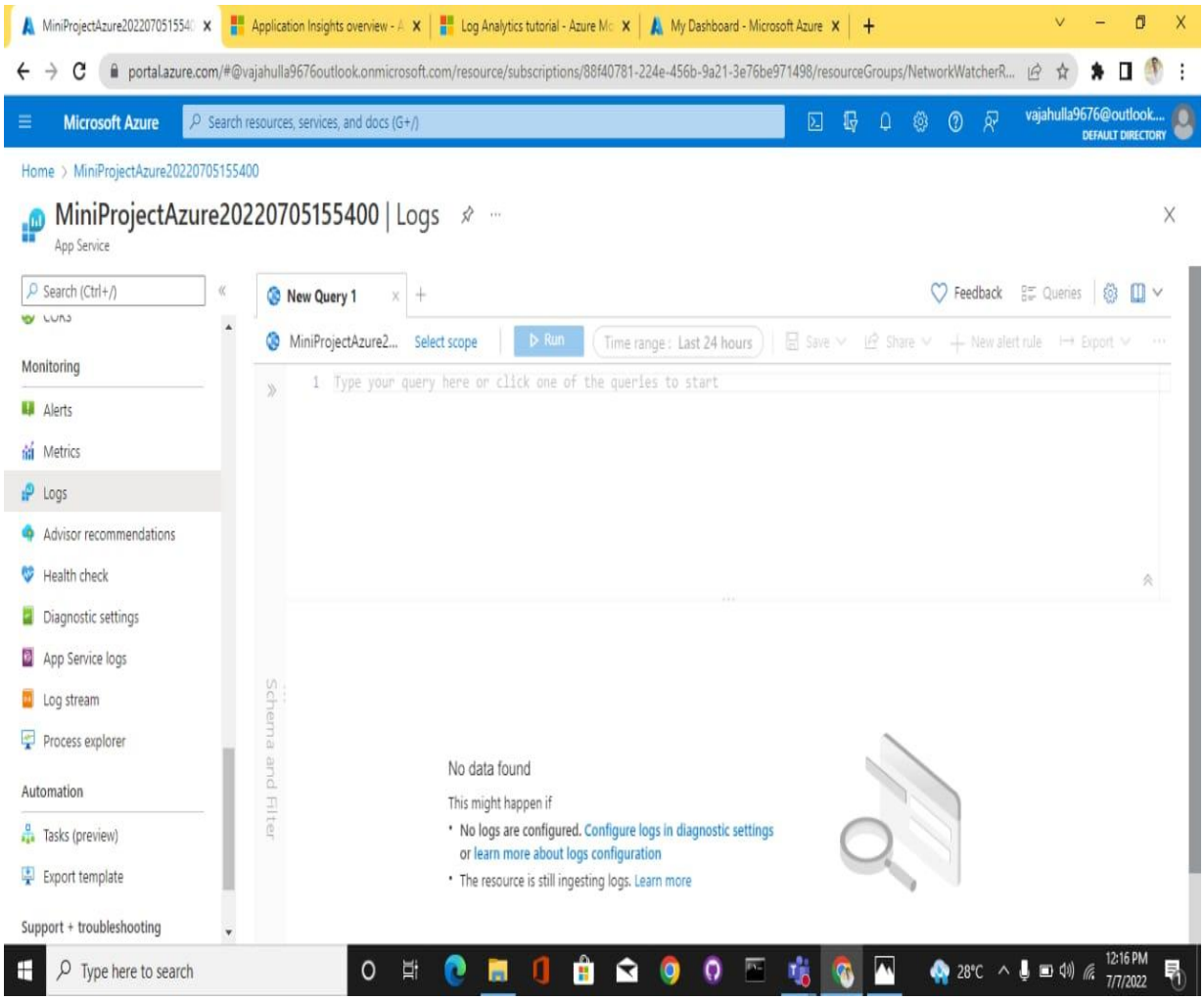
DELETE


petstore.swagger.io/?_ga=2.259403786.1092241861.1657183405-1878323422.1657183405#/pet/deletePet
POST /pet/{petId} Updates a pet in the store with form data
DELETE /pet/{petId} Deletes a pet
Parameters
Try it out
Name
Description
api_key
api_key
petId
Pet id to delete
petId
Responses
Response content type
application/json
Code
Description
400
Invalid ID supplied
404
Pet not found
store Access to Petstore orders
POST /store/order Place an order for a pet
GET /store/order/{orderId} Find purchase order by ID
azure2.png
azure1.png
Show all
Type here to search
29°C
3:57 PM
7/7/2022

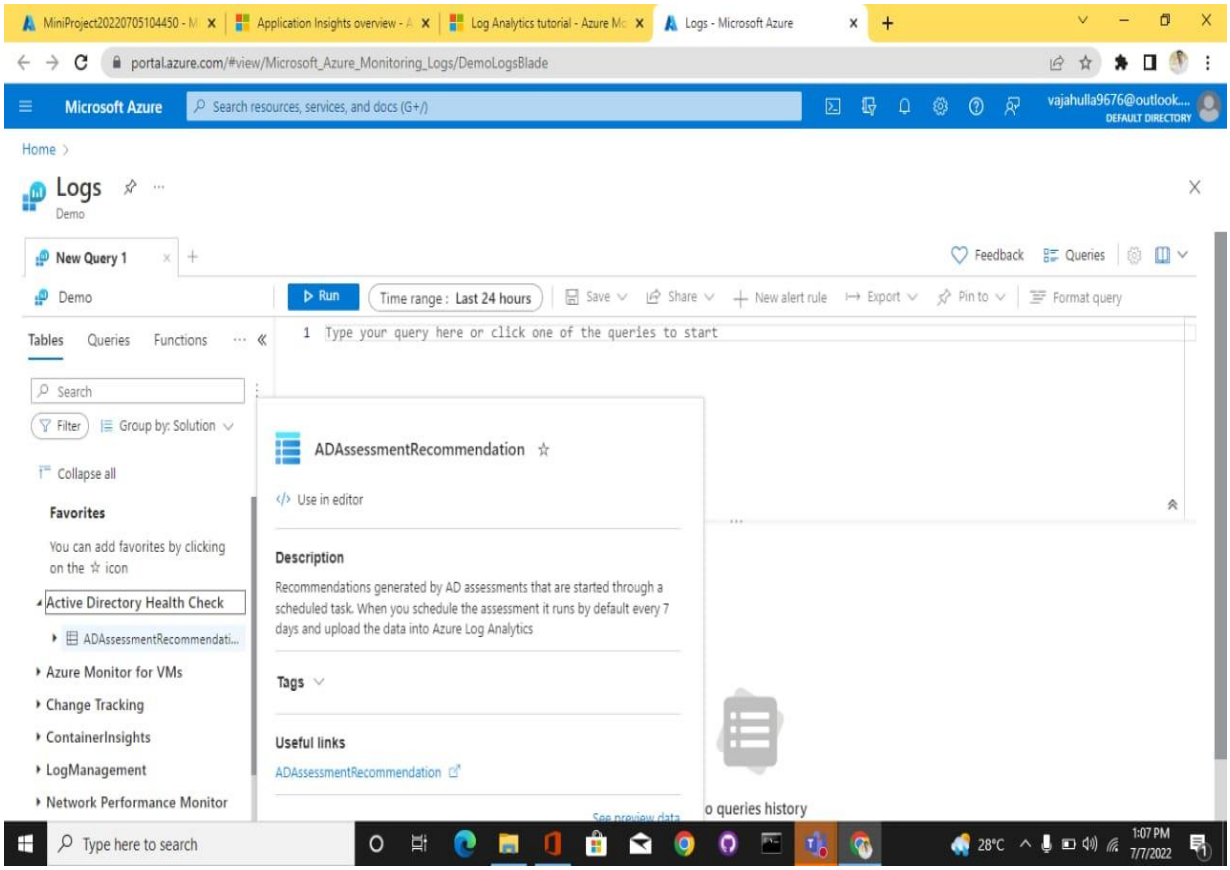
6.Work with Log Analytics with the sample logs available

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.

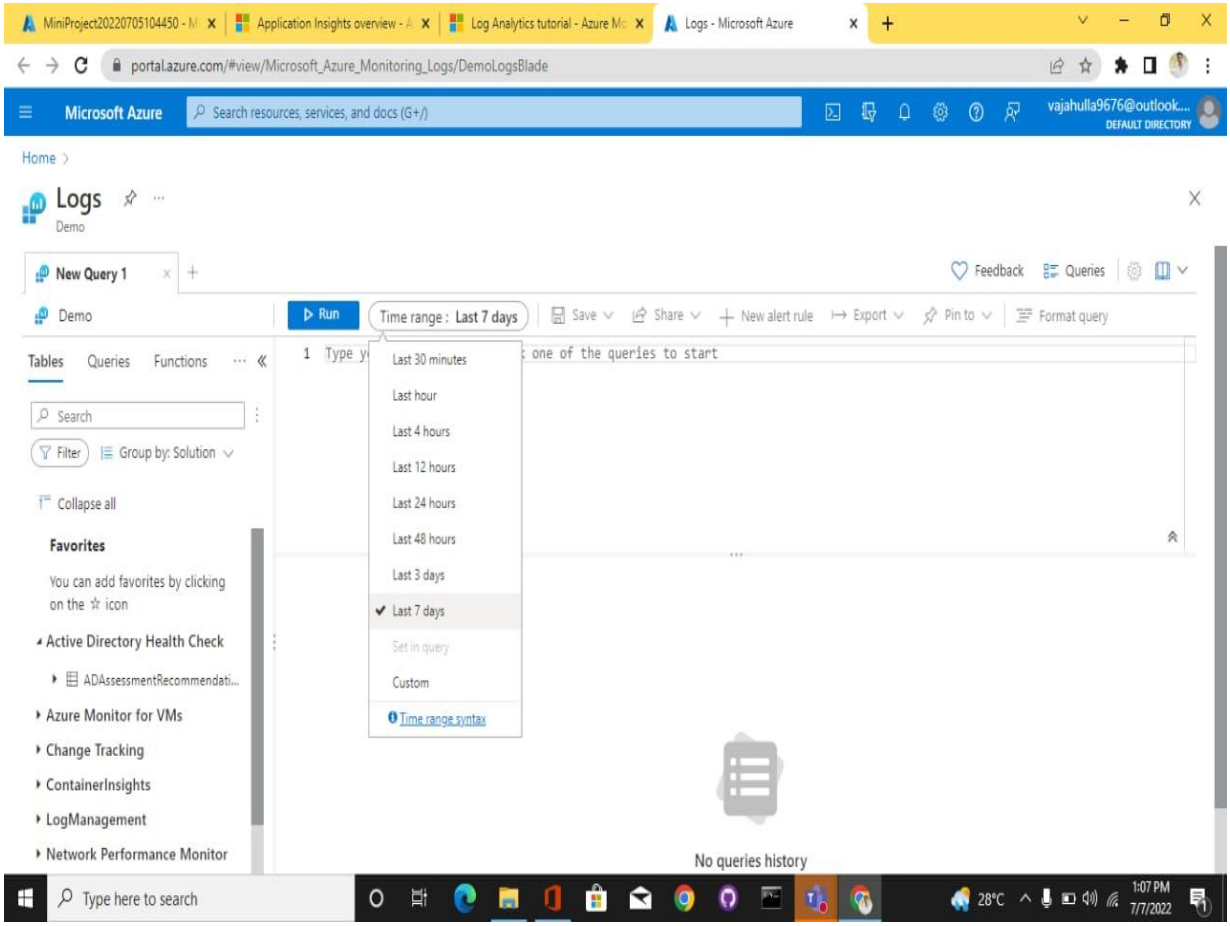
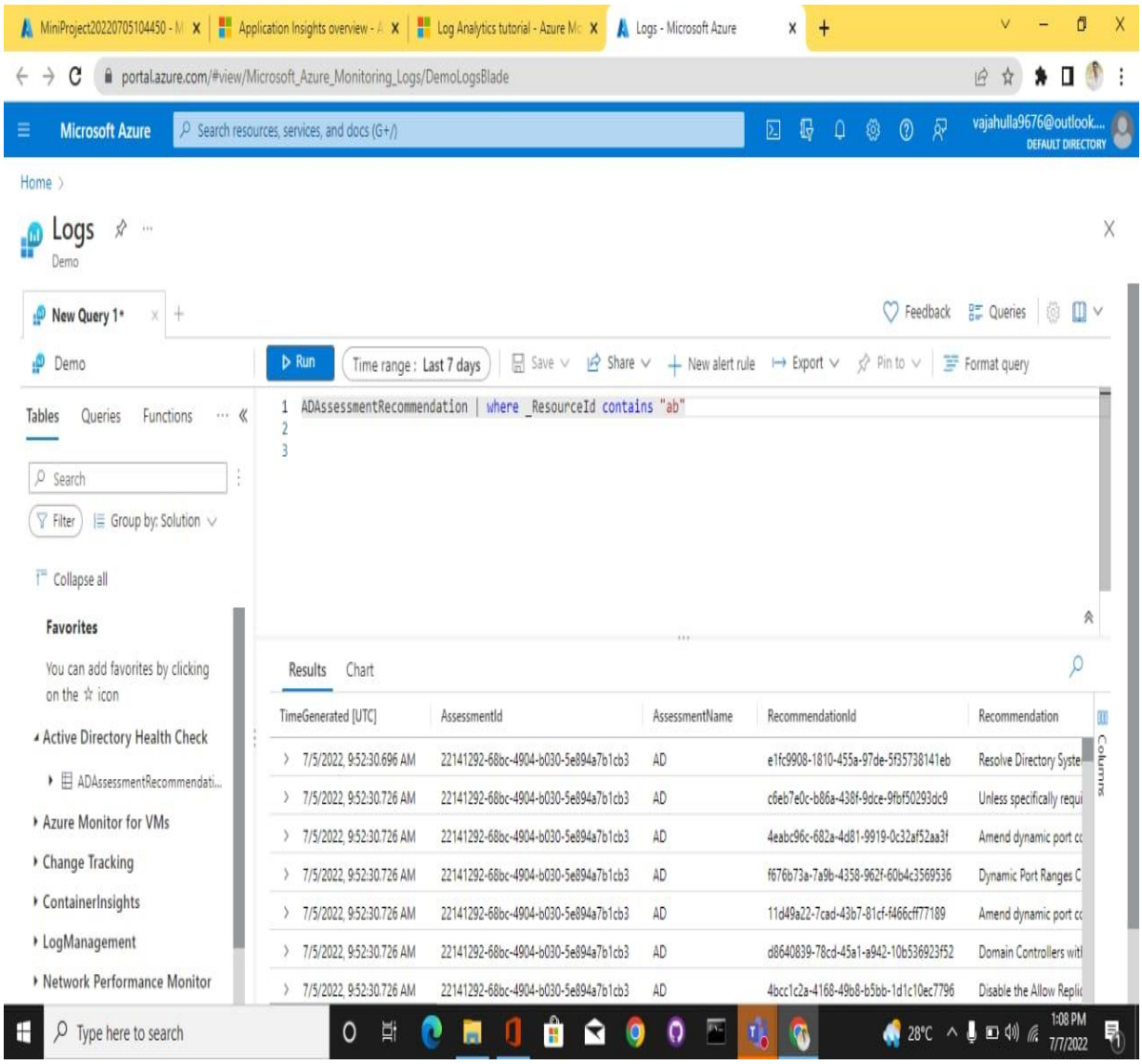
❖ Select the Logs in Azure Portal.



❖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 time range 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 the results.



❖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.