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:

- ProductID
- ProductName
- Price
- Brand
- ManufactureDate
- ExpirationDate

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.

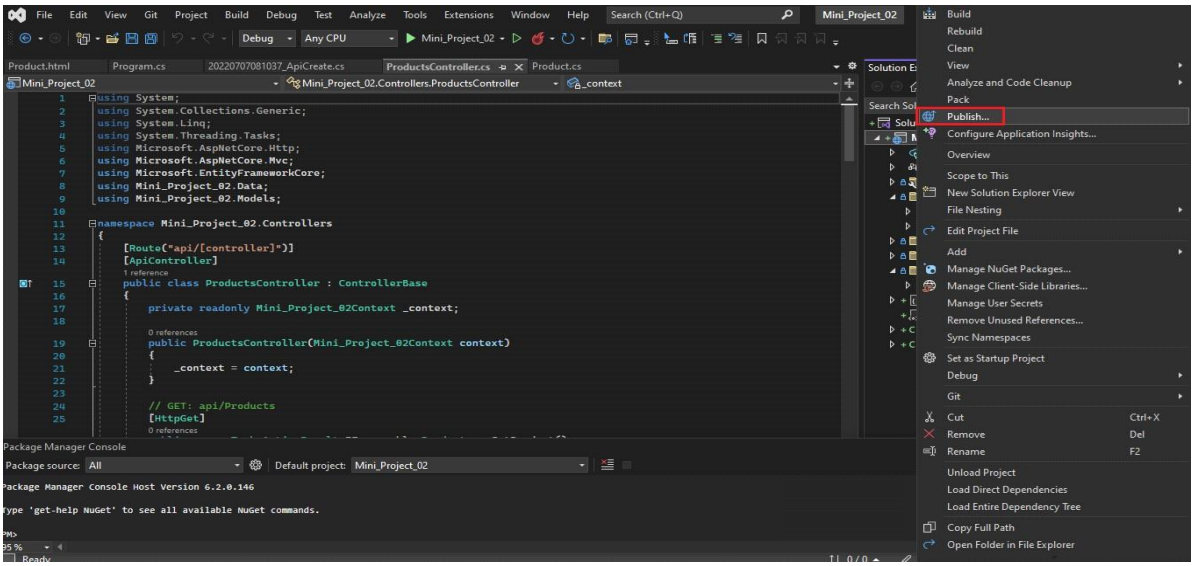
Azure Hosting:

- 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

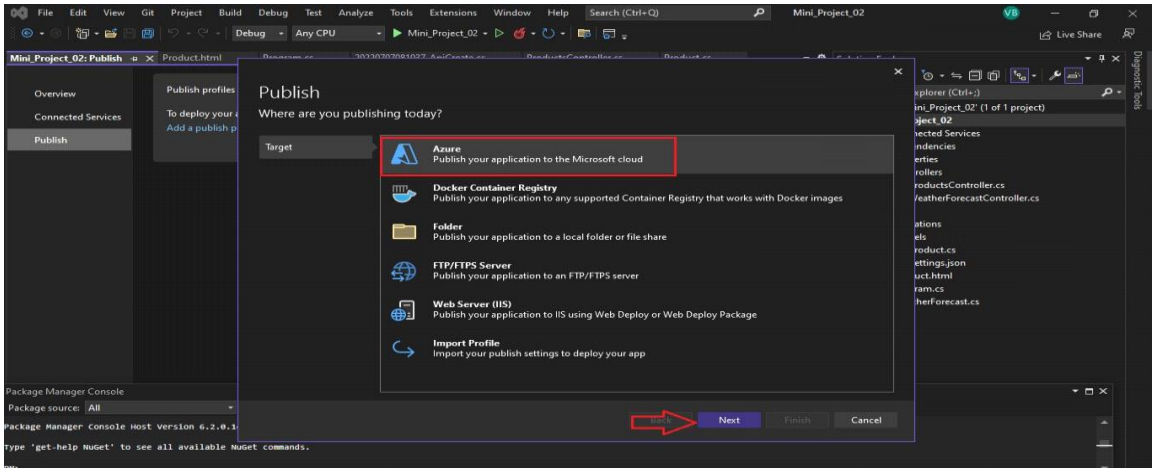
Name :- Vinod Kumar Bantu
Employee No:- 67575

- Host the web API in azure and consume the same using JQuery Client.

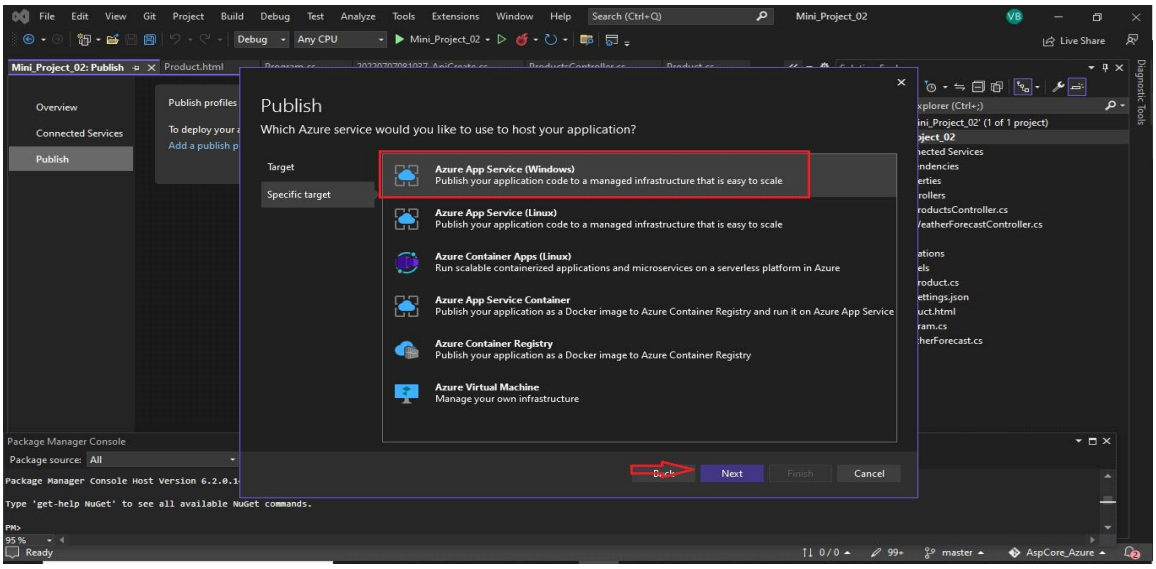
In Solution Explorer, right-click the project and select Publish



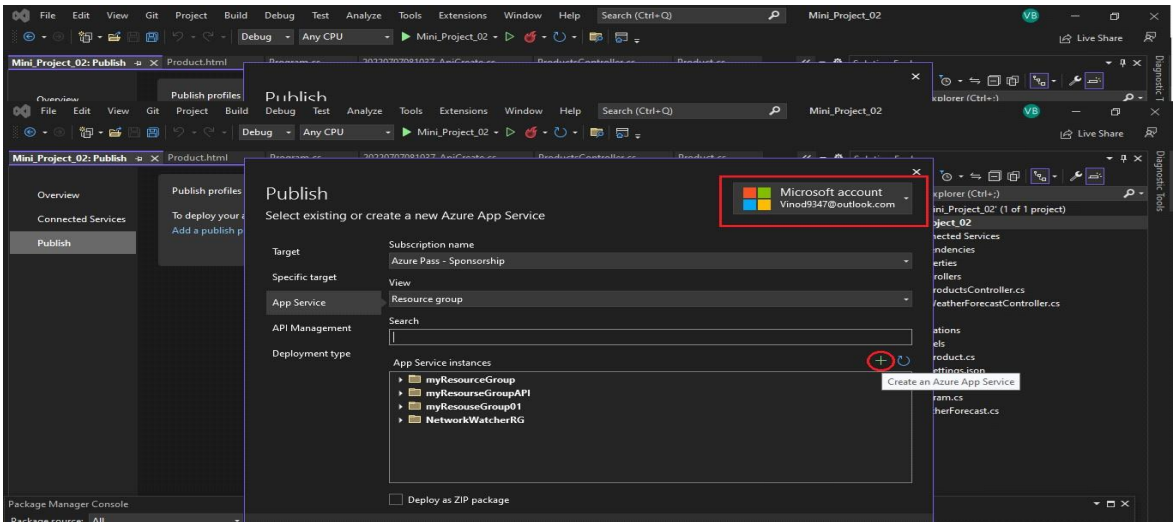
In the Publish dialog, select Azure and select the Next button



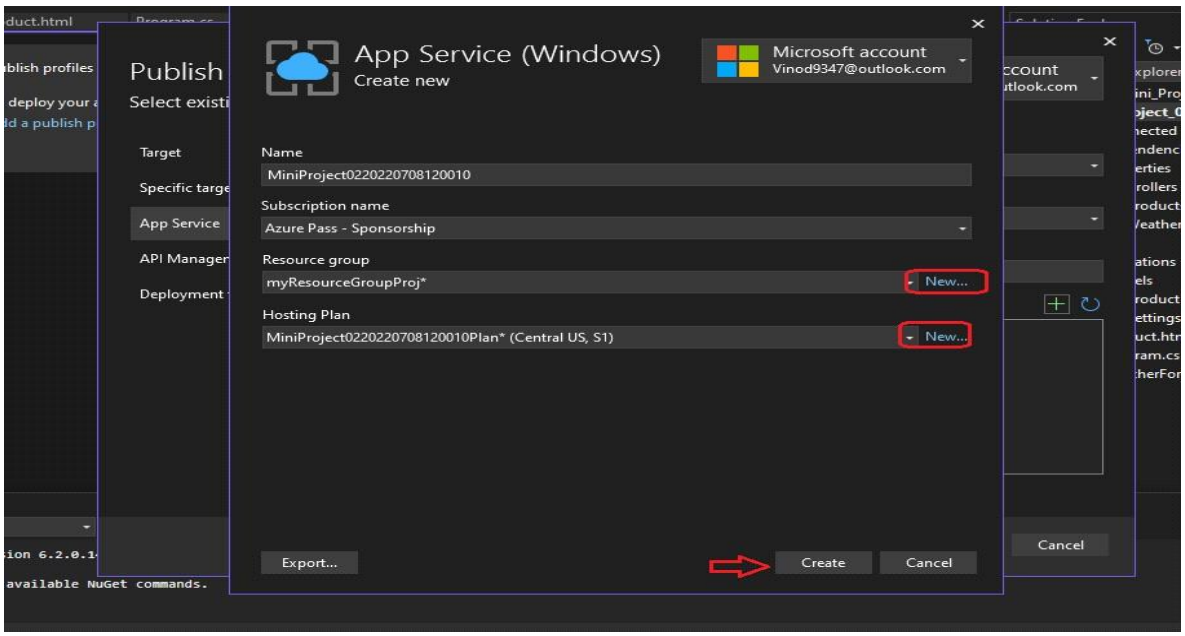
Select Azure App Service (Windows) and select the Next button



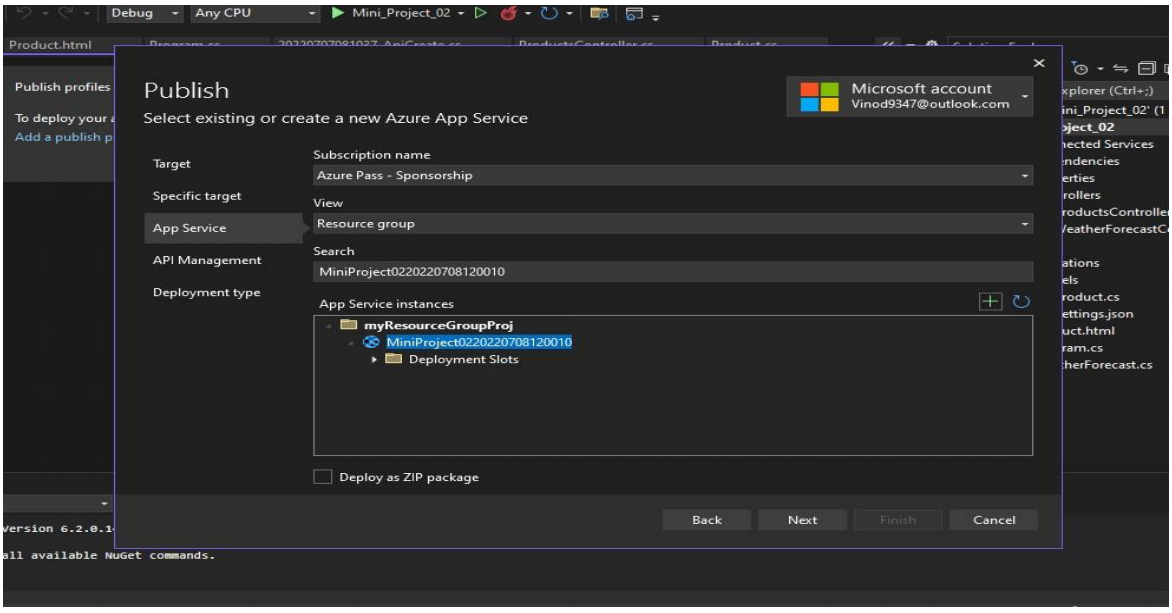
Sign into Microsoft account & Select Create a new Azure App Service.



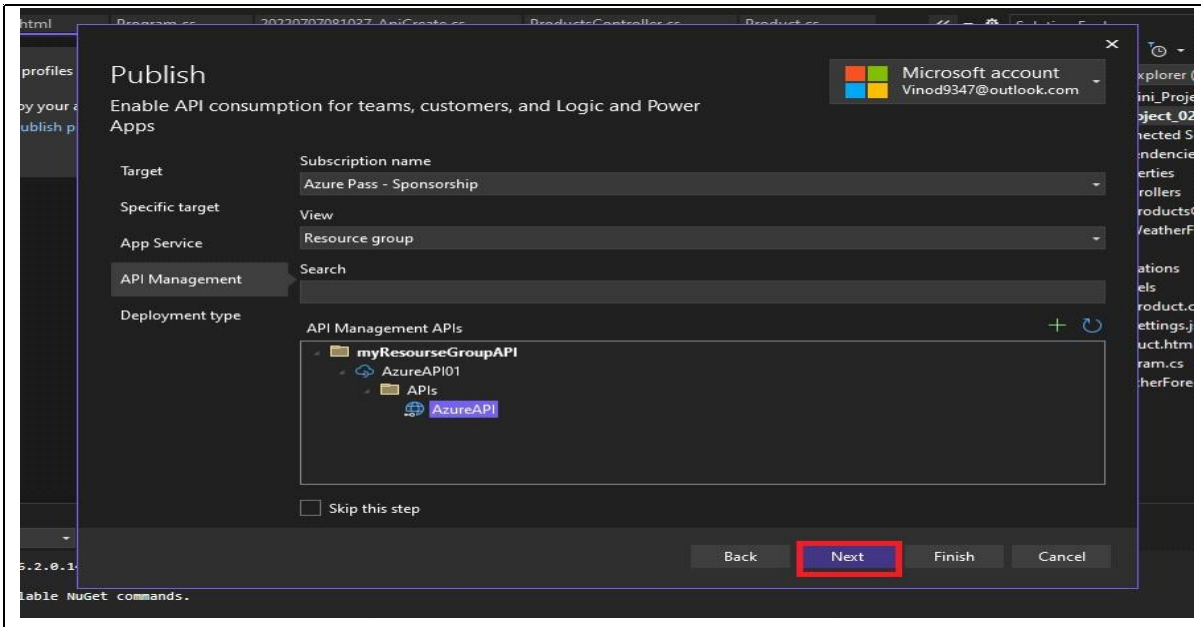
The Create App Service dialog appears. The App Name, Resource Group, and App Service Plan entry fields are populated. You can keep these names or change them. Select the Create button.



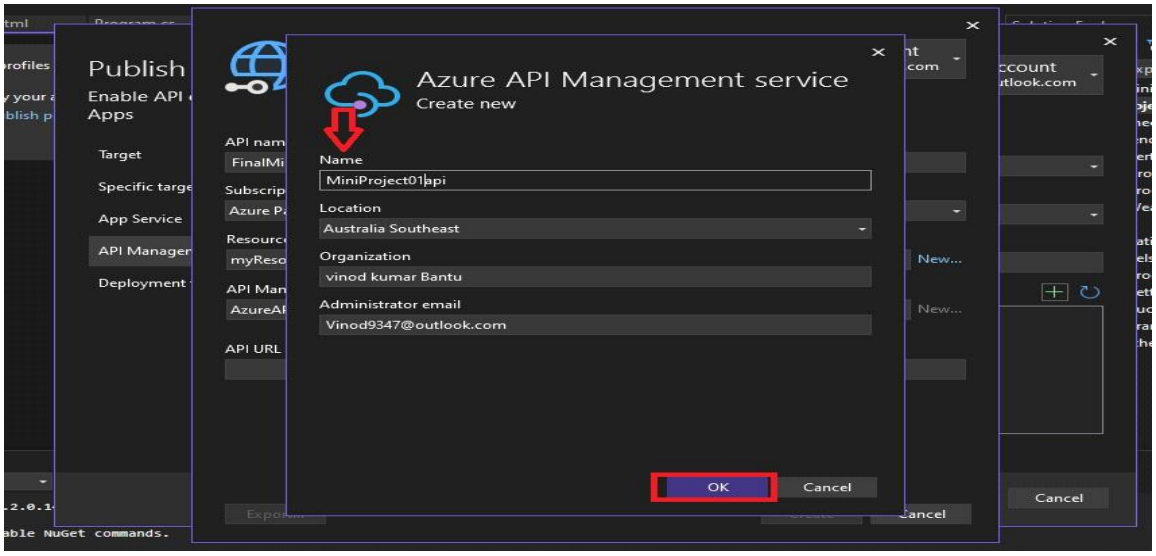
After creation is completed, the dialog is automatically closed and the Publish dialog gets focus again. The instance that was created is automatically selected.



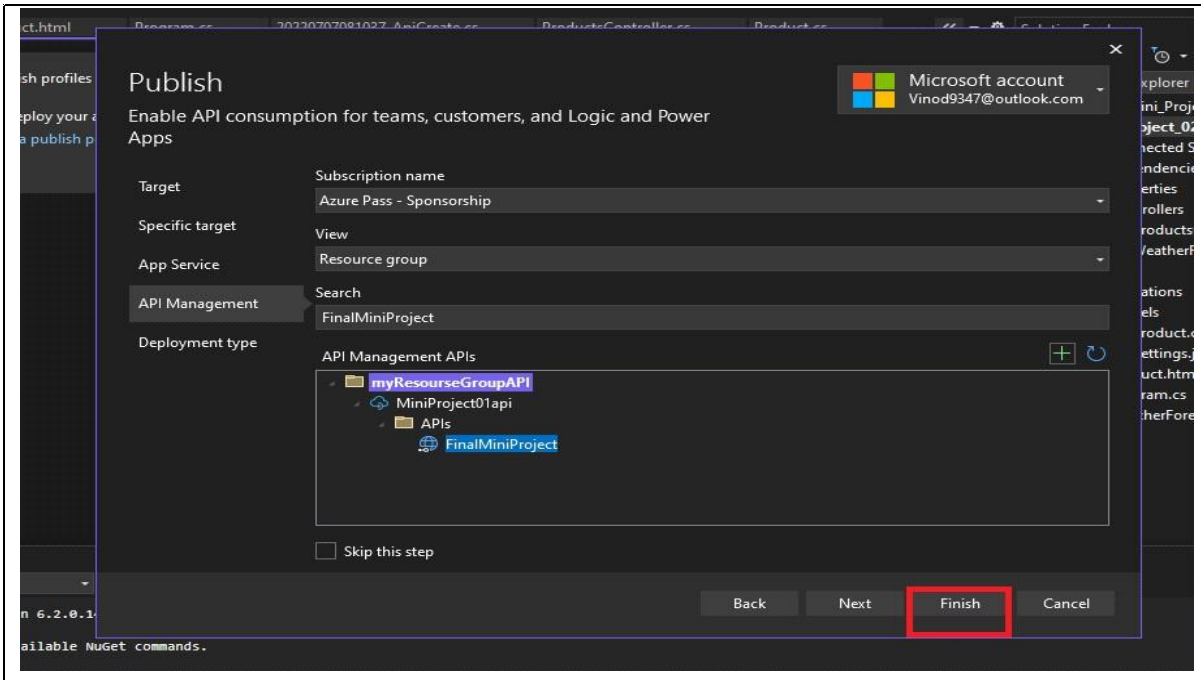
The dialog now shows the Azure API Management service to create.



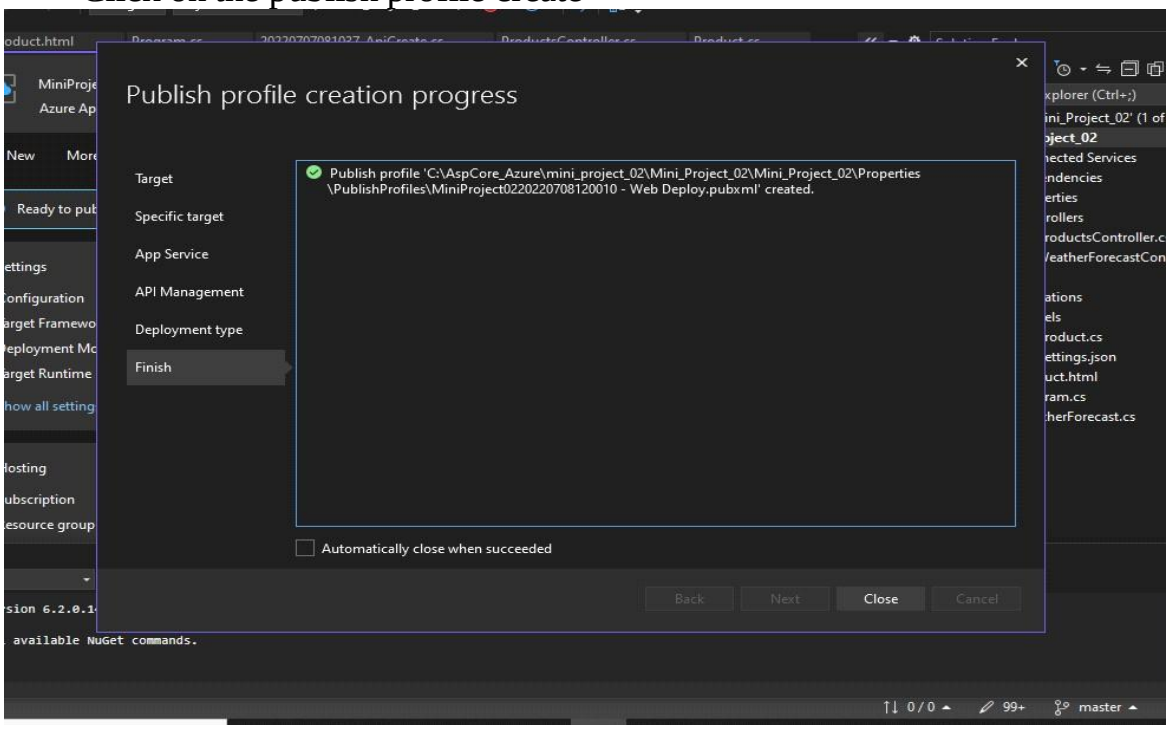
Then Create the Azure API Management service dialog appears. The App Name, Resource Group, and API Management service entry fields are populated. You can keep these names or change them. Select the Create button.



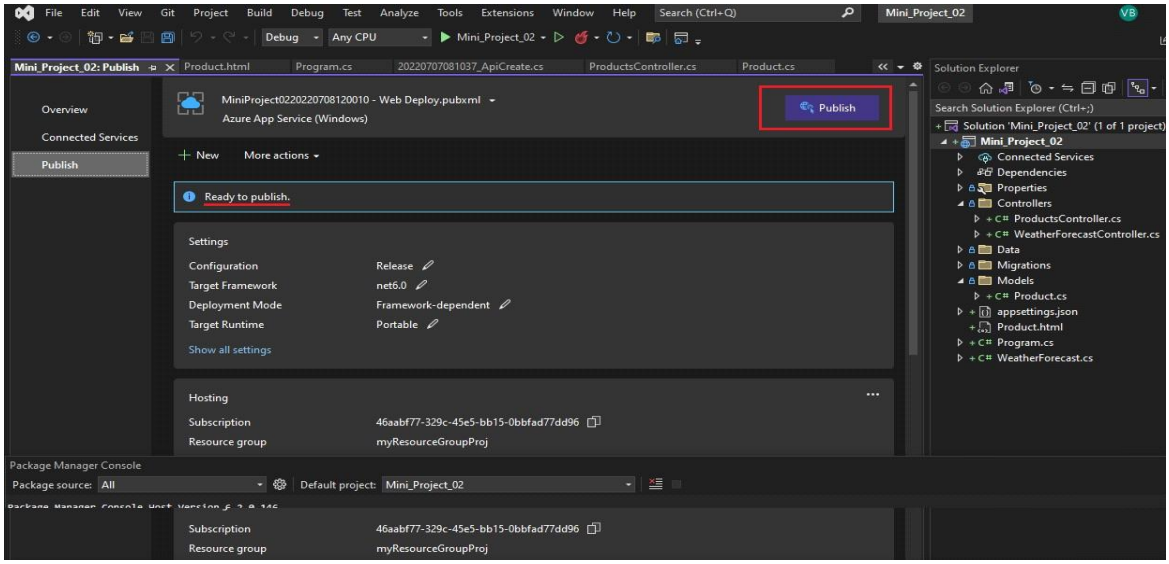
After creation is completed, the dialog is automatically closed and the Publish dialog gets focus again. The instance that was created is automatically selected.



Click on the publish profile create

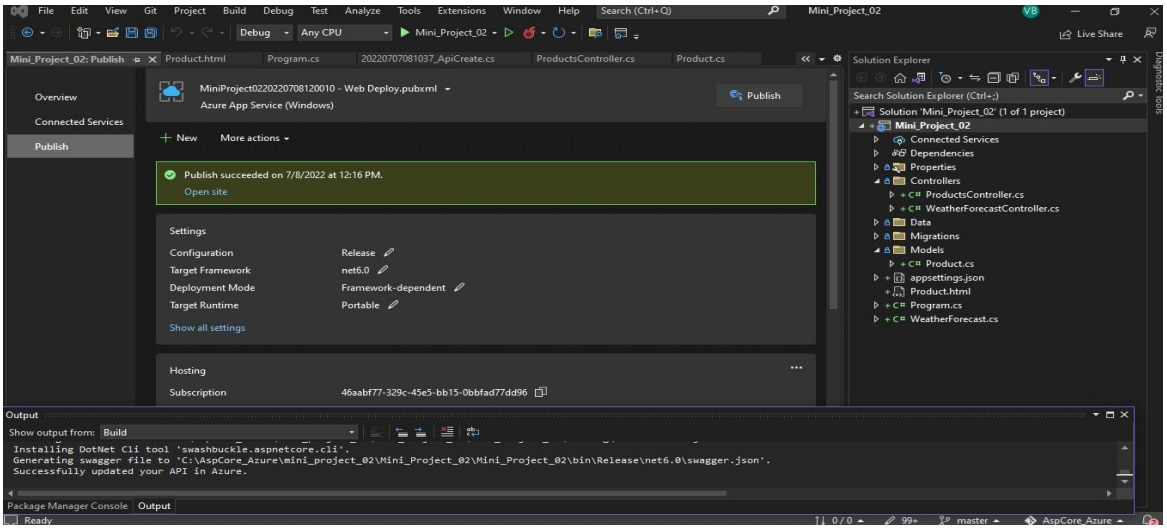


the dialog closes and a summary screen appears with information about thepublish. Click on the Publish button.

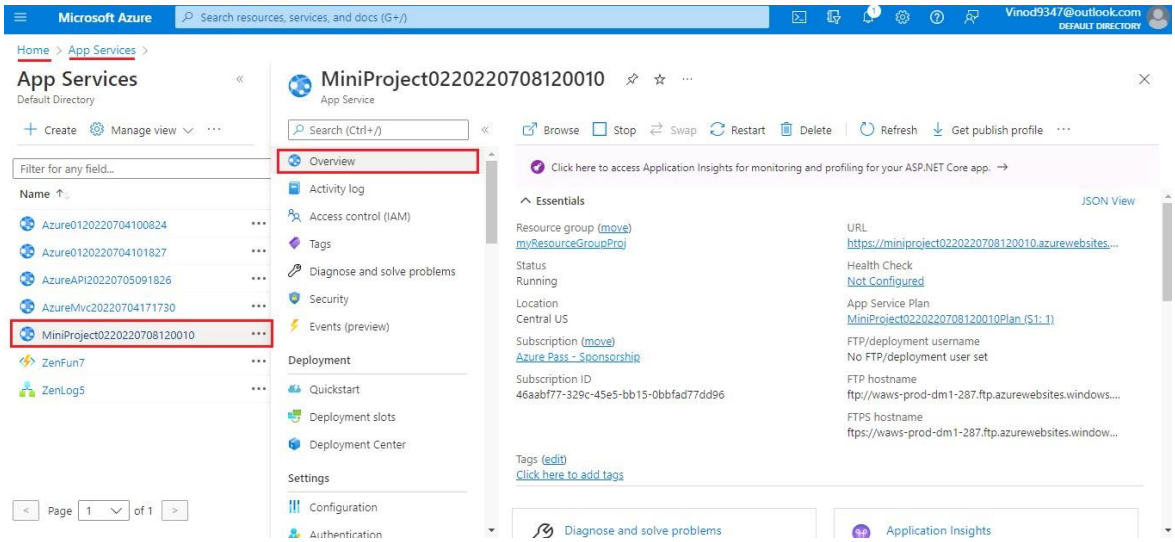


The web API will publish to both Azure App Service and Azure API Management. A new browser window will appear and show the API running in Azure App Service. You can close that window.

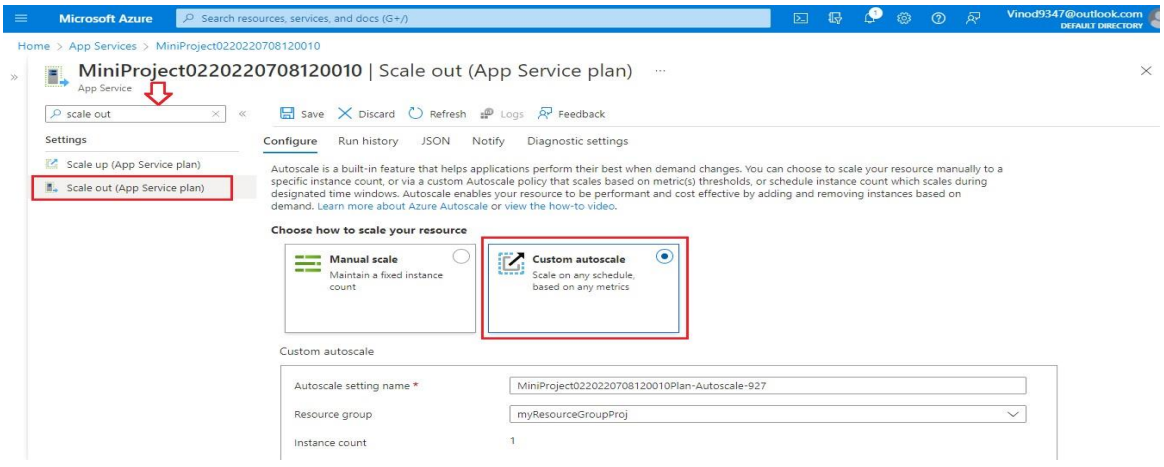
Select the Publish button.



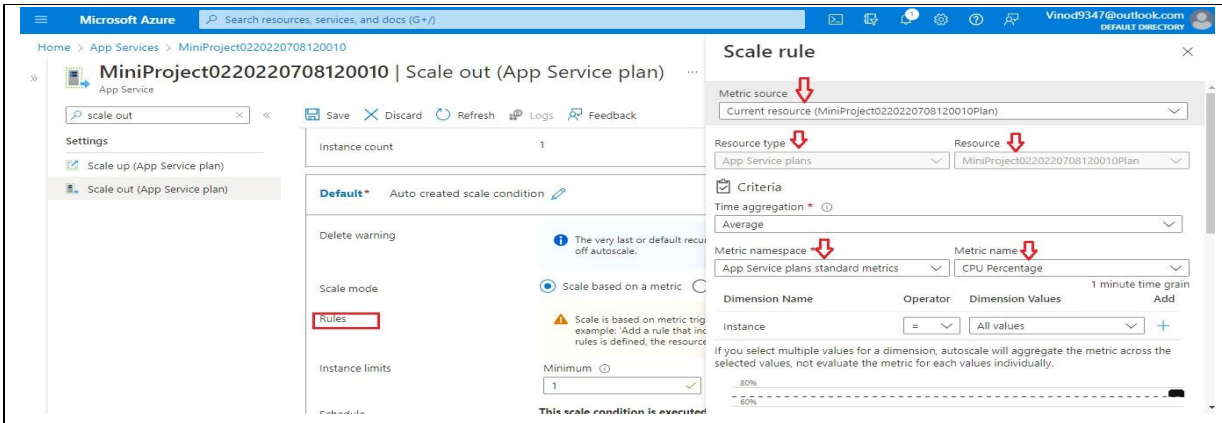
Switch back to the Azure API Management instance in the Azure portal. Refresh the browser window. Select the API you created in the preceding steps. It's now populated and you can explore around.



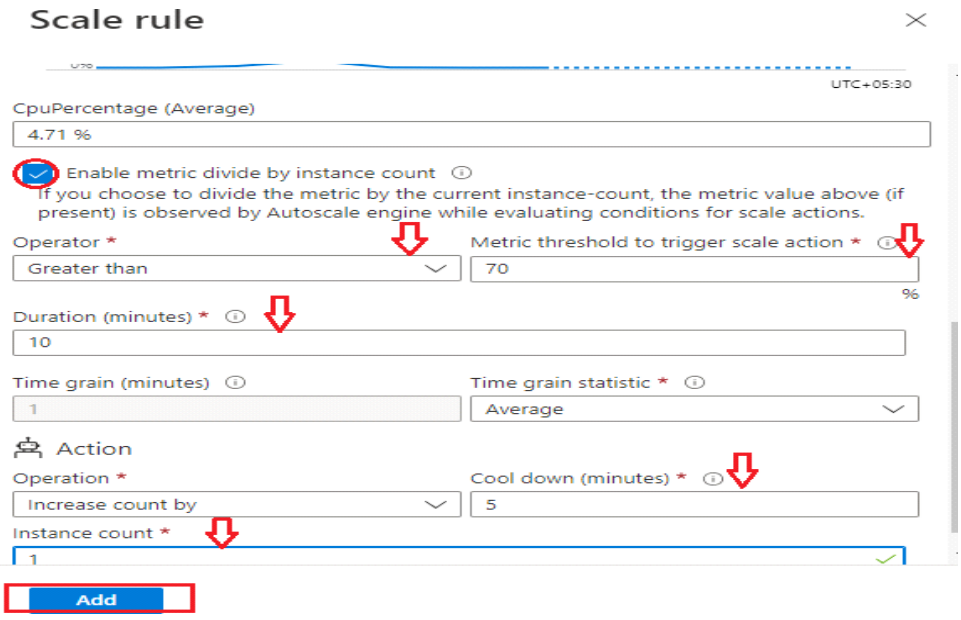
- **Configure Scale out by adding rules for custom scaling**
  - Search and select Scale Out in the search bar
  - Select Custom Autoscale



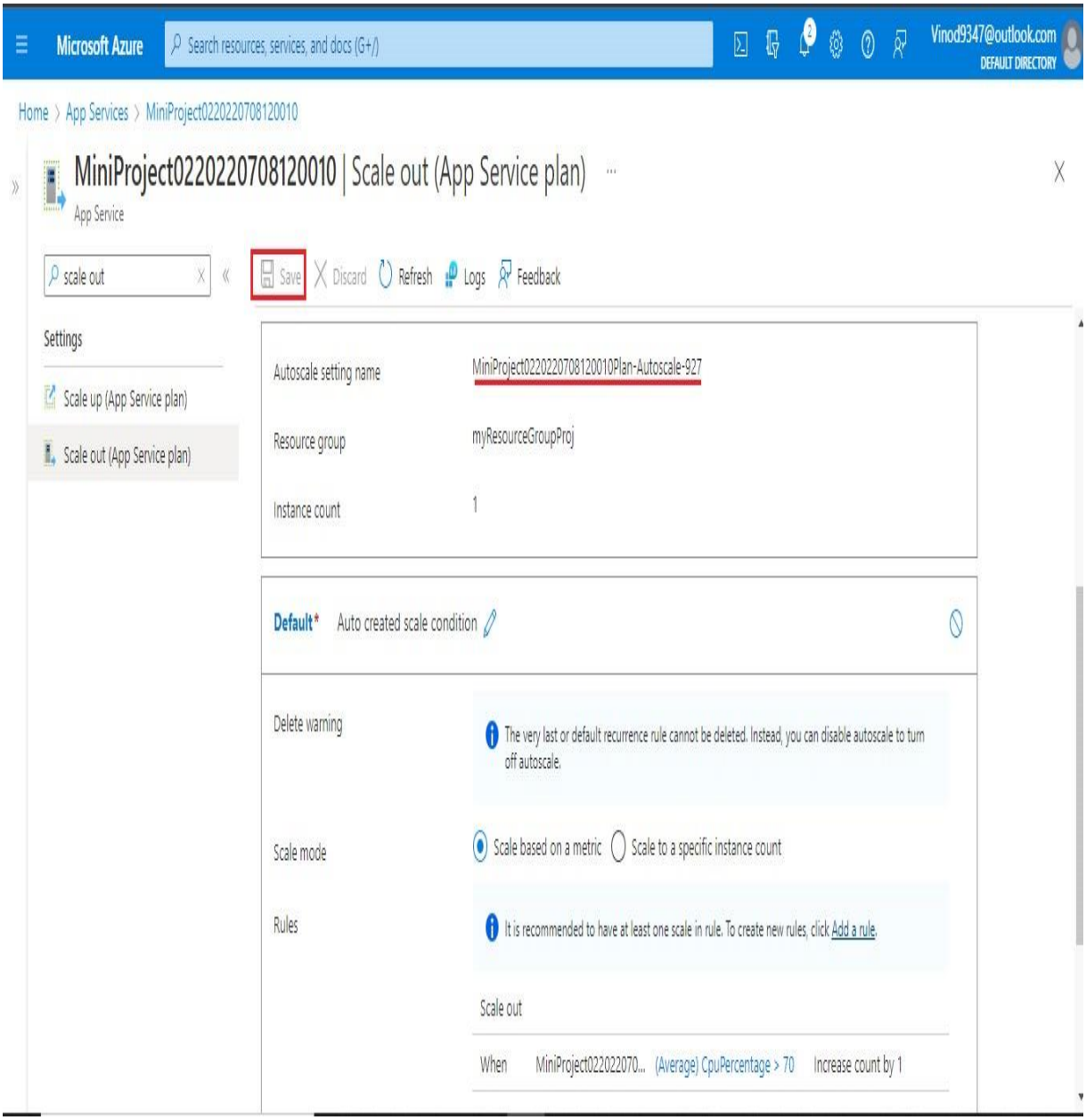
- In the Rules section of the default scale condition, select Add a rule.
- From the Metric source dropdown, select current resource.
- From Resource Type, select Application Insights.
- From the Resource dropdown, select your App services plan standardmetrics.
- Select a Metric name to CPU Percentage.



- Select Enable metric divide by instance count so that the number of sessionsper instance is measured.
- From the Operator dropdown, select Greater than.
- Enter the Metric threshold to trigger the scale action, for example, 70.
- Under Actions, set the Operation to Increase count and set the Instance count to 01 and Cool down by 05 minutes and then click Add.
- Set the maximum number of instances that can be spun up in the Maximumfield of the Instance limits section for example, 01.



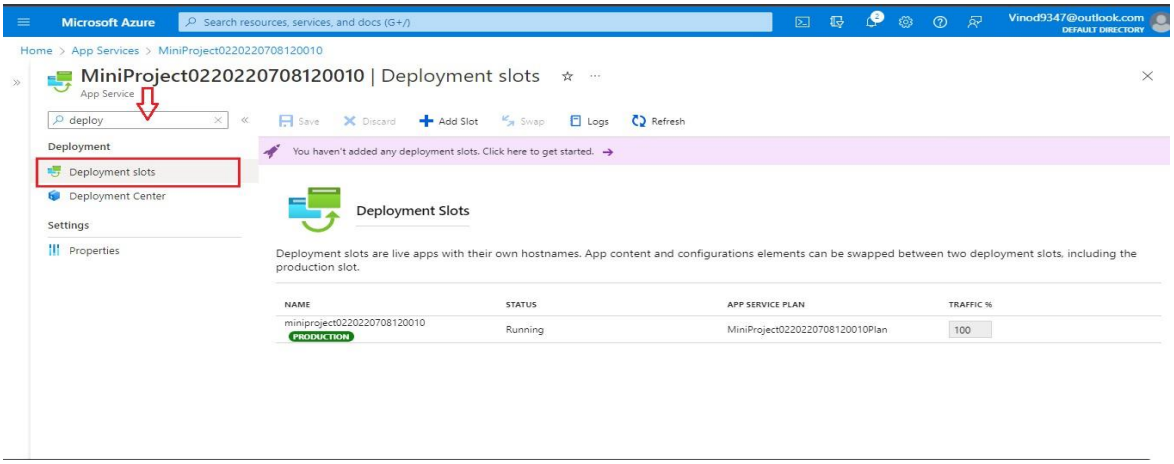
- Select Add & Save.


Microsoft Azure
Search resources, services, and docs (G+/)
Vinod9347@outlook.com
DEFAULT DIRECTORY
Home > App Services > MiniProject0220220708120010
MiniProject0220220708120010 | Scale out (App Service plan)
App Service
scale out
Save
Discard
Refresh
Logs
Feedback
Settings
Scale up (App Service plan)
Scale out (App Service plan)
Autoscale setting name
MiniProject0220220708120010Plan-Autoscale-927
Resource group
myResourceGroupProj
Instance count
1
Default*
Auto created scale condition
Delete warning
The very last or default recurrence rule cannot be deleted. Instead, you can disable autoscale to turn off autoscale.
Scale mode
Scale based on a metric
Scale to a specific instance count
Rules
It is recommended to have at least one scale in rule. To create new rules, click Add a rule.
Scale out
When
MiniProject022022070...
(Average) CpuPercentage > 70
Increase count by 1

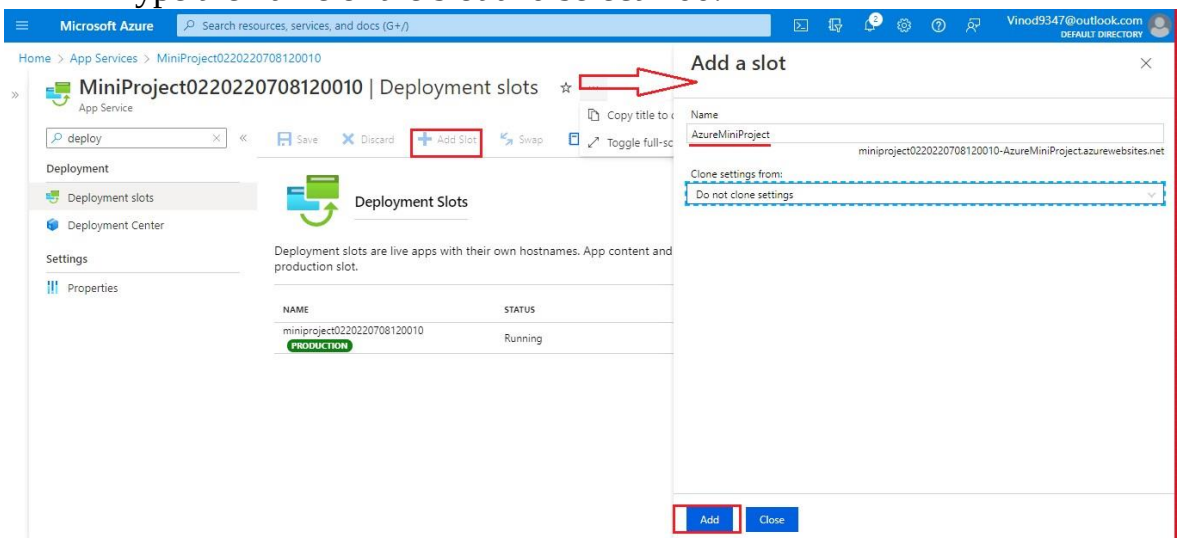
• Configure Deployment slots for staging and production

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.

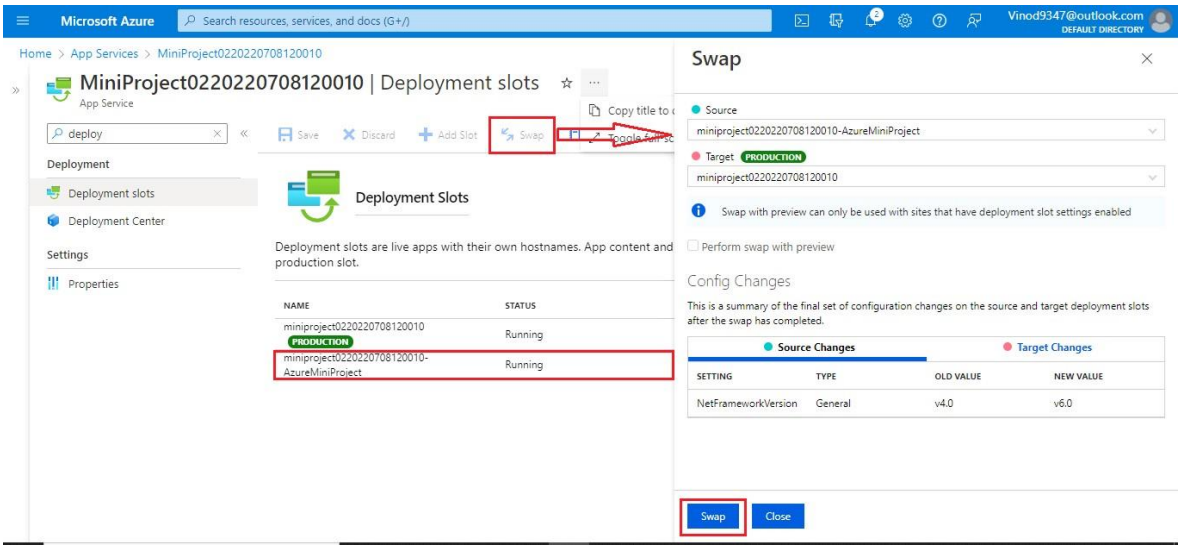
- Navigate to Deployment slots in the function app, and then select the SlotName.



- Select Deployment slots, and then select + Add Slot.
- Type the name of the slot and select Add.



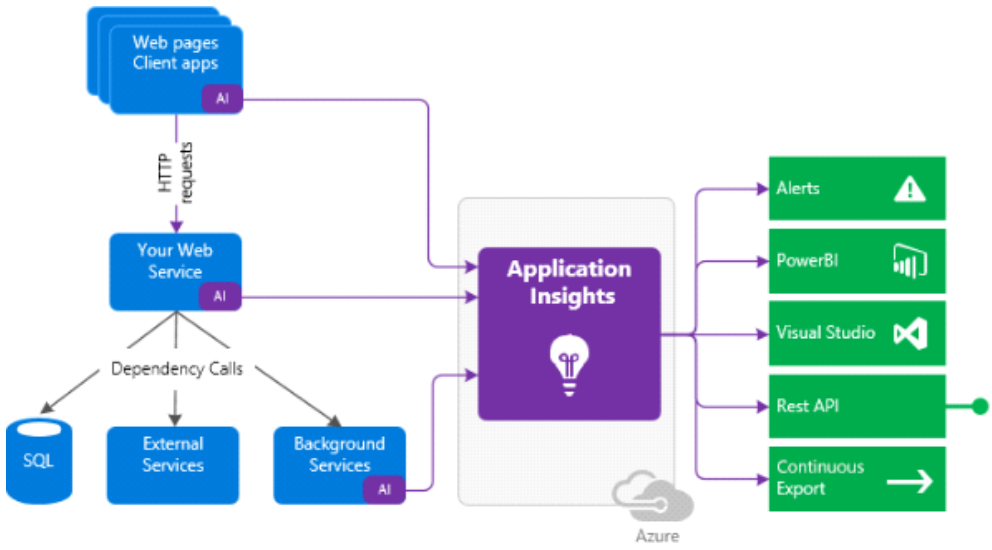
- Select Deployment slots, and then select Swap.
- Verify the configuration settings for your swap and select Swap.



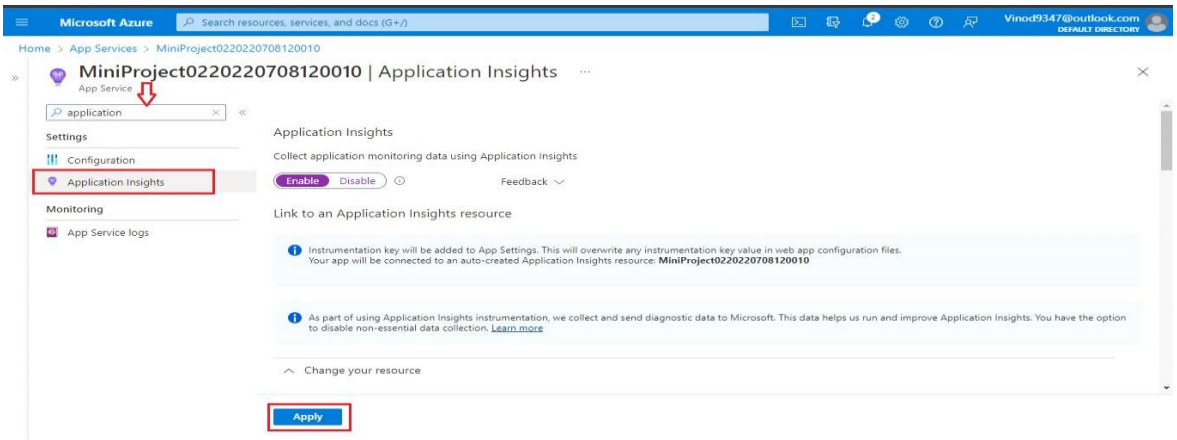
- The operation may take a moment while the swap operation is executing.

• Configure Application Insights for the project

What is Azure Application Insights? Application Performance Management (APM) service for web developers that supports multiple platforms. Application Insights are mainly used to monitor the live web applications, automatically detecting the performance anomalies. The advanced data analytics tools help in diagnosing the application issues raised.

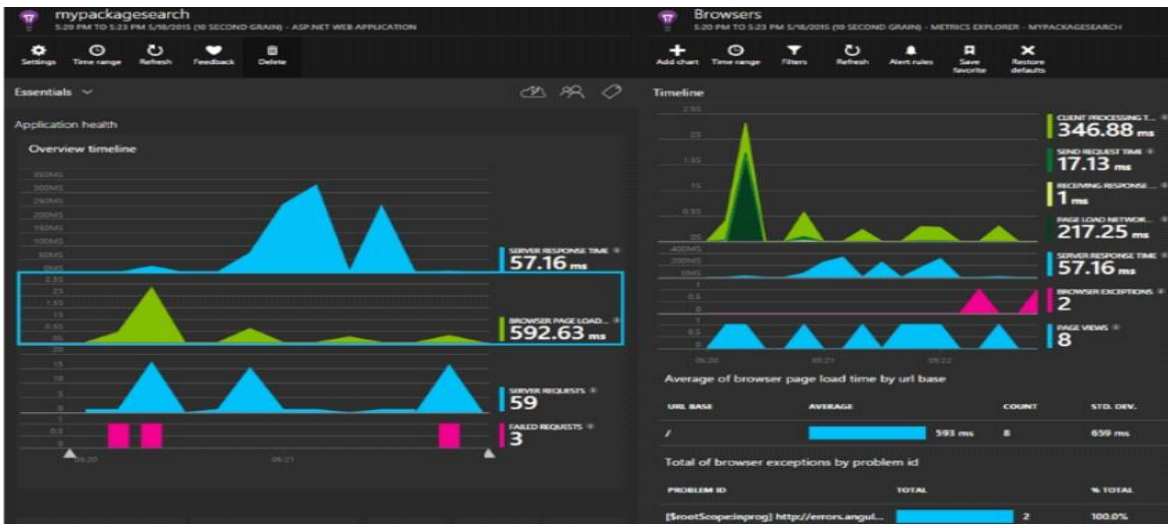


1) Open Azure Portal and search for Application Insights as below.



2) Monitoring non-web apps using Azure Application Insights.

3) Azure Monitoring Tools in Application Insights.



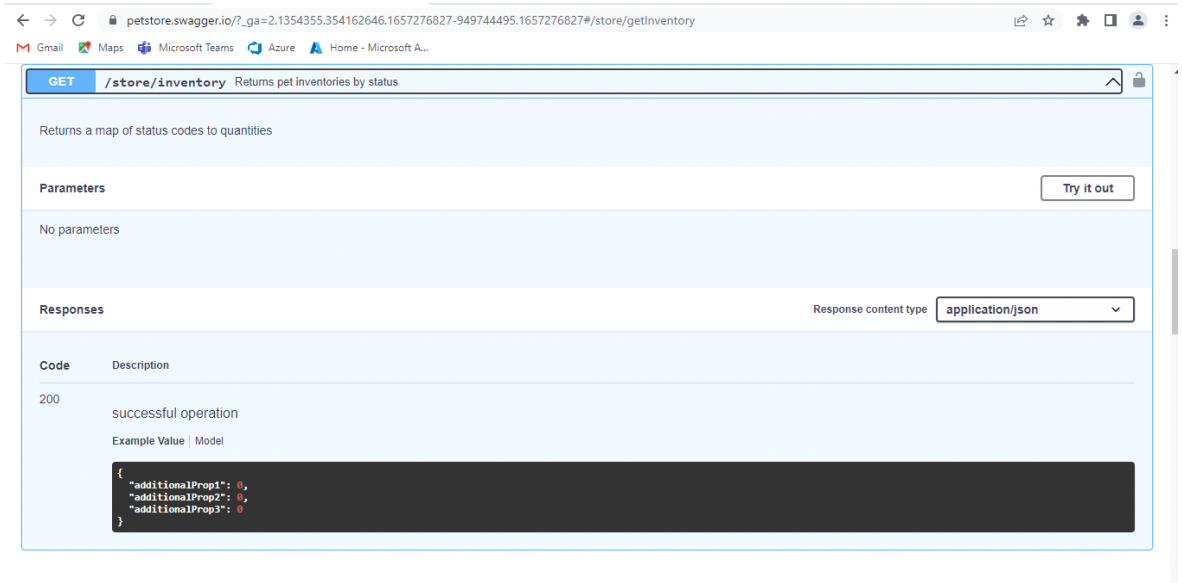
4) Configure Swagger for the API

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.

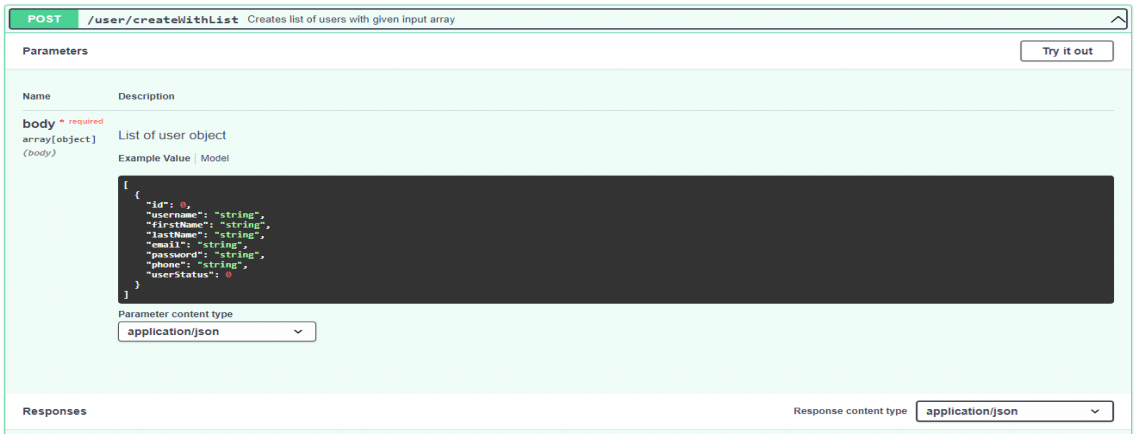
Advantages :

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

GET

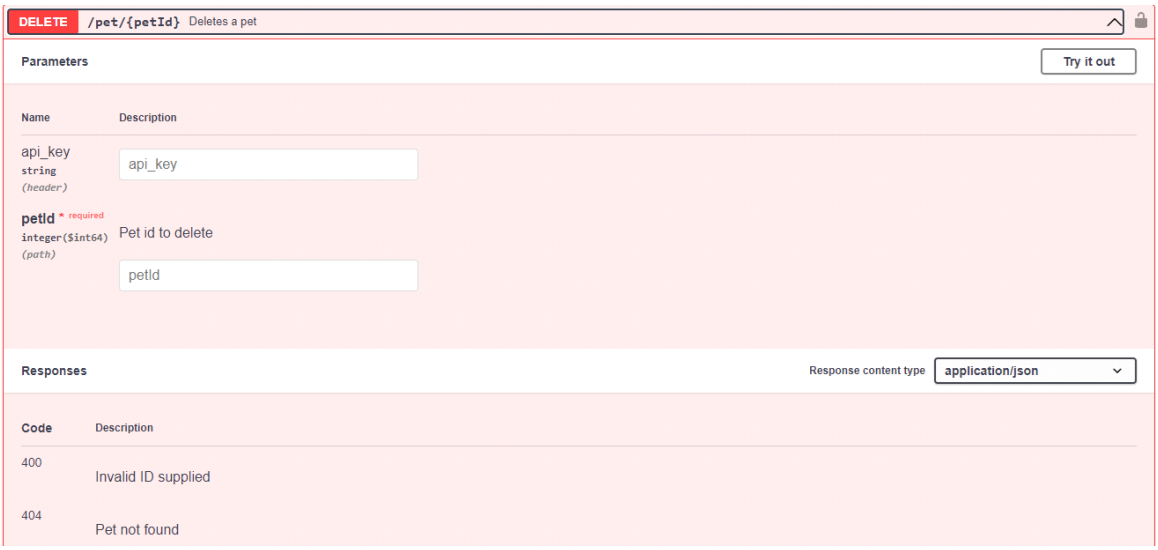


POST



PUT

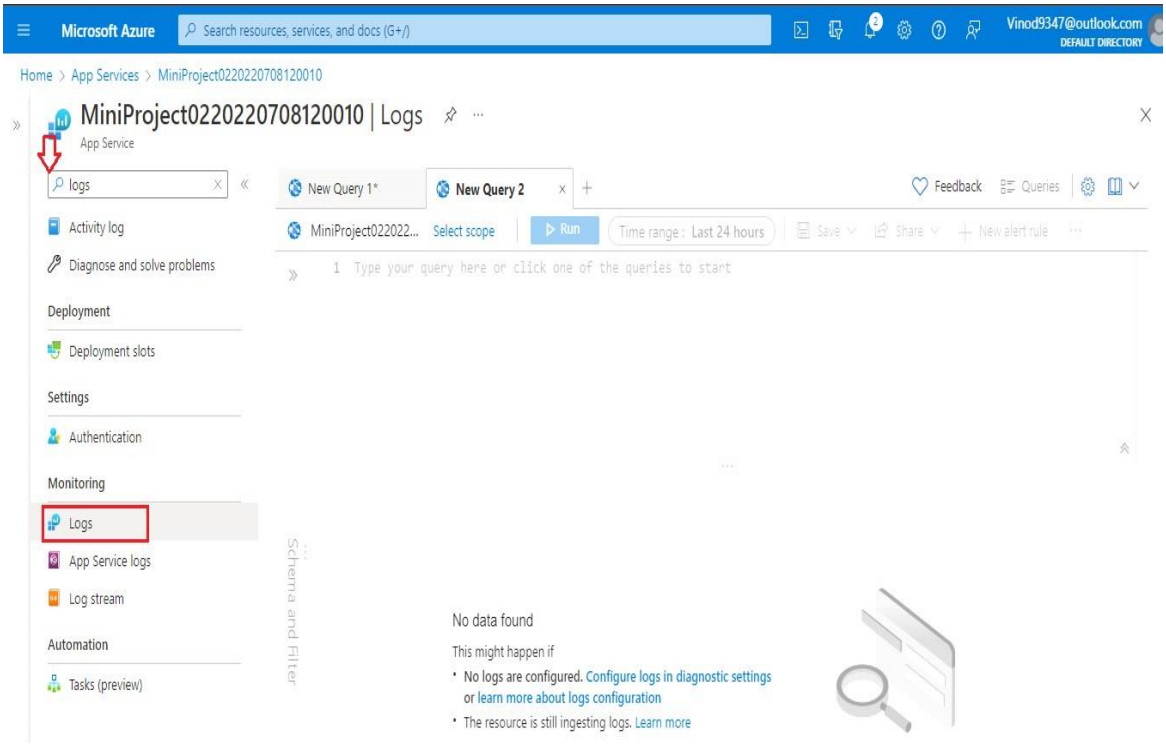
DELETE



- **Work with Log Analytics with the sample logs available**

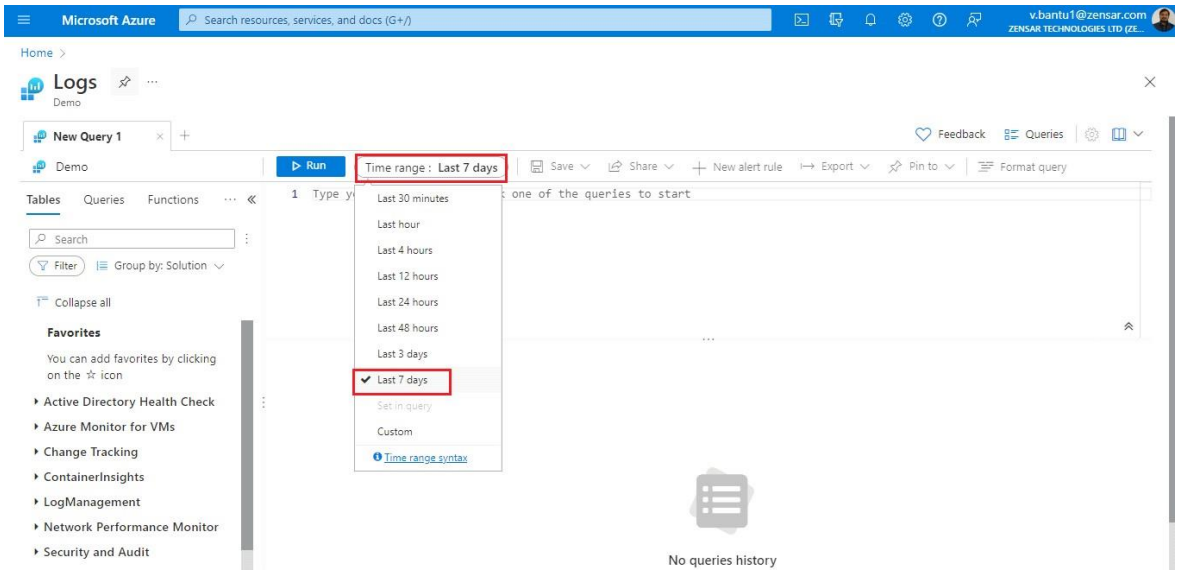
Log Analytics is a tool in the Azure portal to edit and run log queries from data collected by Azure Monitor logs and interactively analyze their results. You can use Log Analytics queries to retrieve records that match particular criteria, identify trends, analyze patterns, and provide various insights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

Select 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

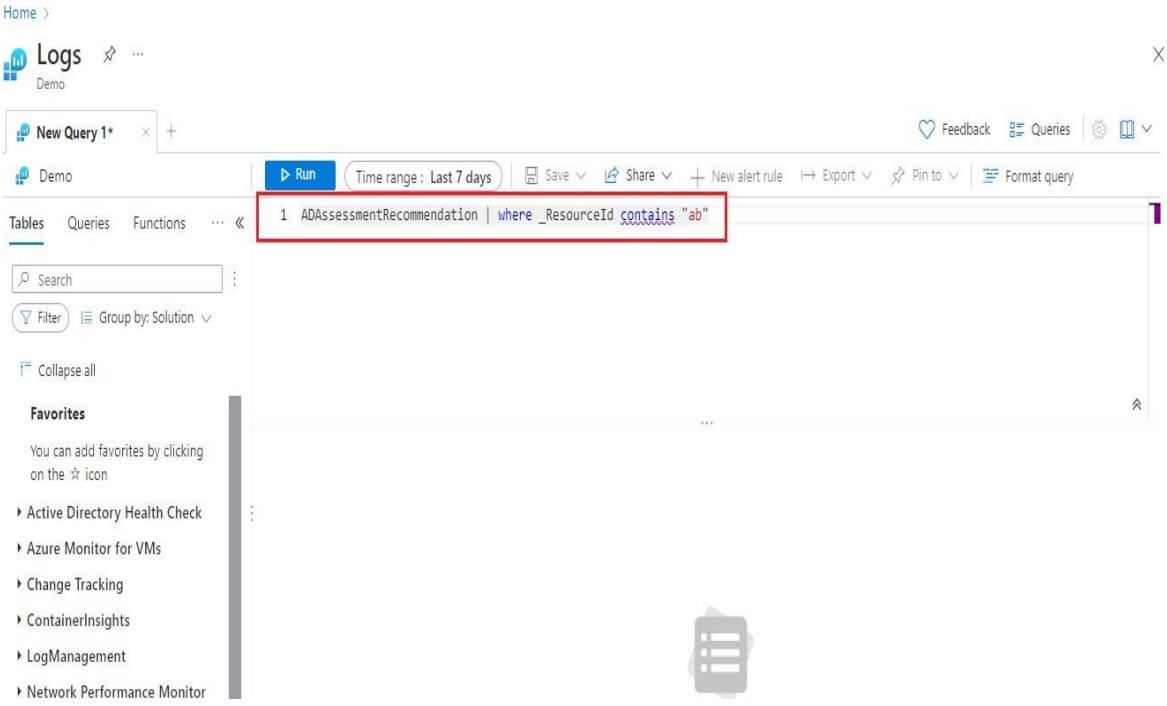


All queries return records generated within a set time range. By default, thequery returns records generated in the last 24 hours.

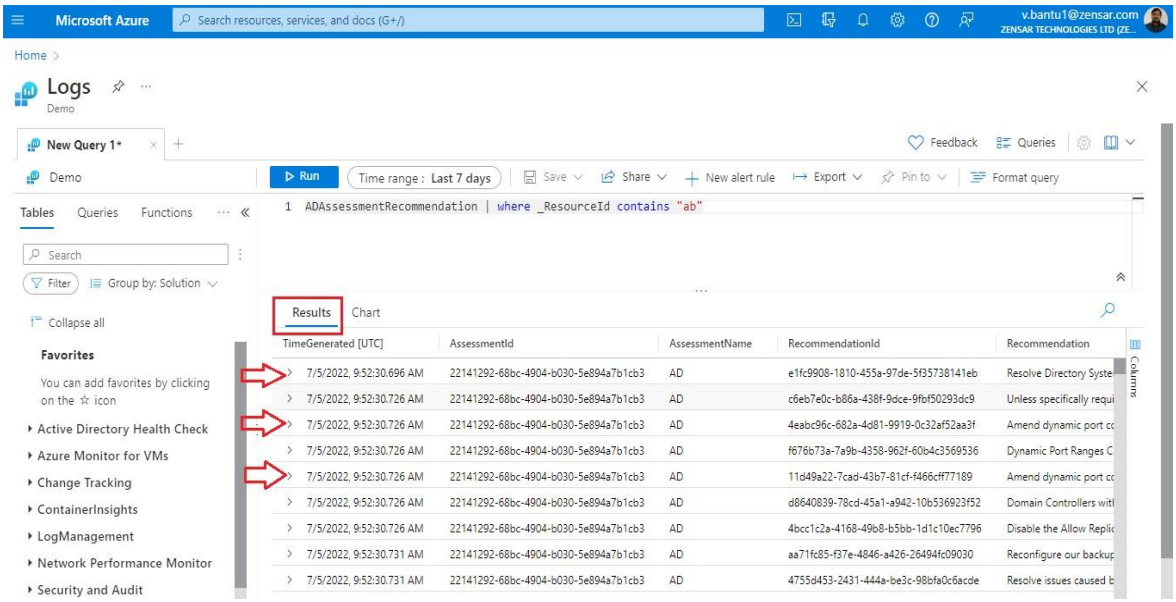
- This is the simplest query that we can write. It just returns all the



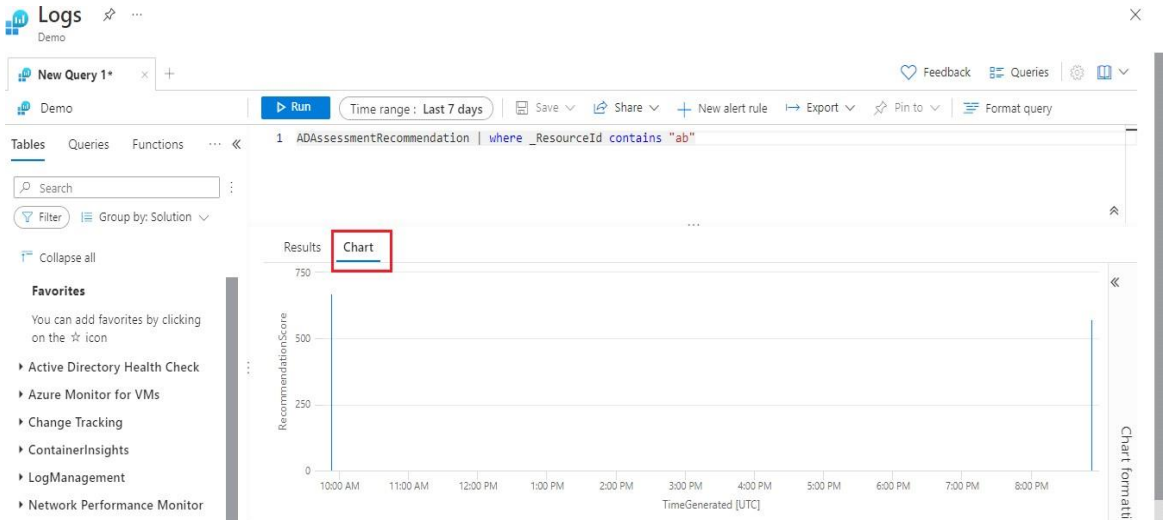
records in a table. Run it by selecting the Run button or by selecting Shift+Enter with thecursor positioned anywhere in the query text.



- Select Run to return the results.



- Charts.



----------THE END----------

Vinod Kumar Bantu -67575