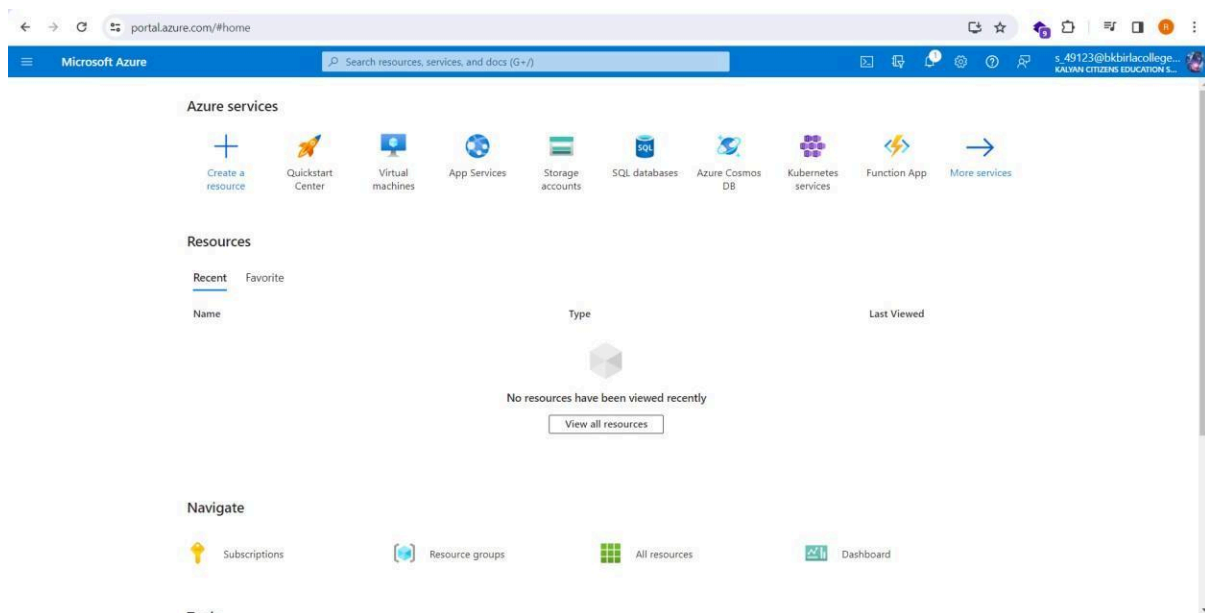


Practical 1

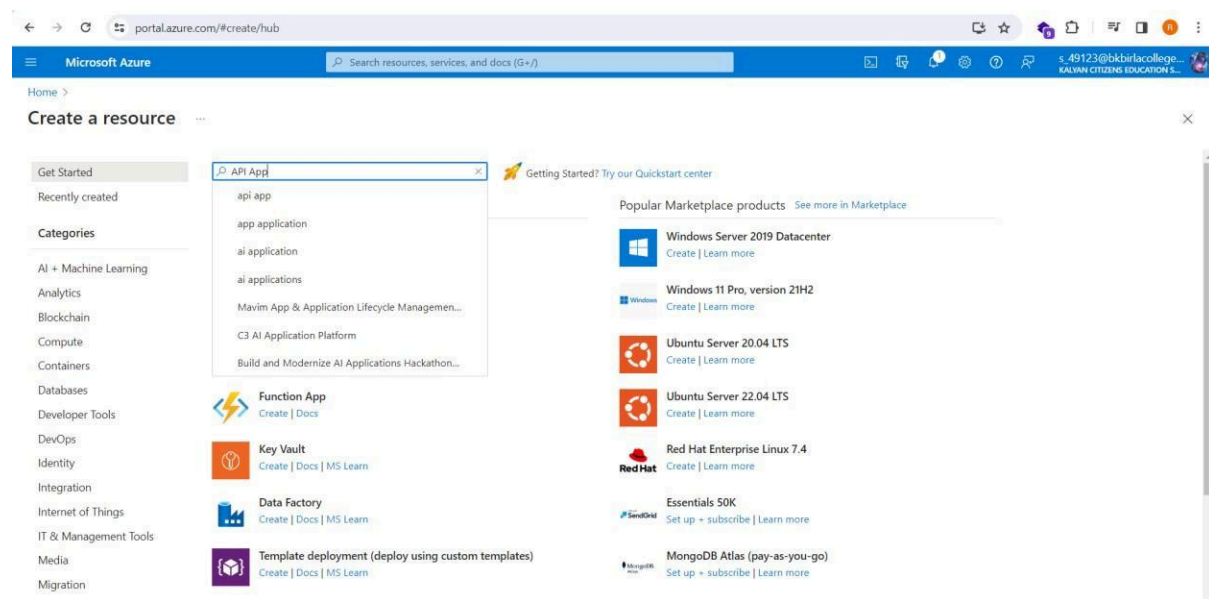
Create a simple Azure API Application

Theory: Azure API Applications provide a streamlined approach to building and deploying APIs on the Azure cloud platform. These applications leverage Azure's robust infrastructure to handle API requests and responses efficiently. With Azure API Applications, developers can easily create RESTful APIs using popular frameworks like ASP.NET Core and Node.js, while also taking advantage of Azure's scalable and reliable architecture. This enables developers to focus more on building the core functionality of their APIs without worrying about infrastructure management. Additionally, Azure API Applications offer features such as automatic scaling, authentication, and monitoring, ensuring that APIs remain performant and secure. Overall, Azure API Applications simplify the process of creating and managing APIs, empowering developers to deliver high-quality services to their users.

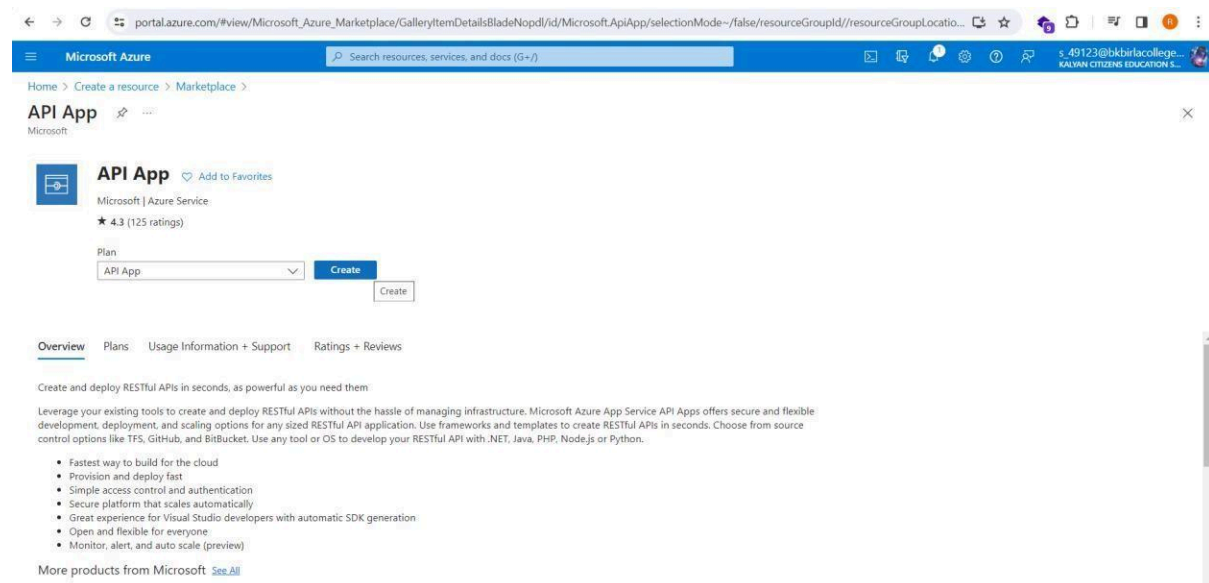
Step1: Go to Azure portal i.e. <https://www.portal.azure.com/> and login with your Azure Account.



Step2: Now go to “Create a resource” and search for “API App”.



Step3: Now click on create and create a new API App.



Step4: Now give your API app a name and click on “review + create”.

The screenshot shows the 'Create API App' wizard in the Microsoft Azure portal, specifically the 'Review + create' step. The breadcrumb navigation at the top reads 'Home > Create a resource > Marketplace > API App >'. The page title is 'Create API App'. Below the title, there are tabs for 'Basics', 'Database', 'Deployment', 'Networking', 'Monitoring', 'Tags', and 'Review + create'. The 'Basics' tab is active. The page content includes a description of Azure App Service API Apps, followed by 'Project Details' where a subscription and resource group are selected. The 'Instance Details' section contains fields for Name, Publish method, Runtime stack, and Operating System. At the bottom, there are buttons for 'Review + create', '< Previous', and 'Next: Database >'. The 'Review + create' button is highlighted in blue.

Microsoft Azure

Search resources, services, and docs (G+)

Home > Create a resource > Marketplace > API App >

Create API App

Basics Database Deployment Networking Monitoring Tags Review + create

Microsoft Azure App Service API Apps offers secure and flexible development, deployment, and scaling options for any sized RESTful API application. Use frameworks and templates to create RESTful APIs in seconds. Choose from source control options like TFS, GitHub, and BitBucket. Use any tool or OS to develop your RESTful API with .NET, Java, PHP, Node.js or Python. [Learn more](#)

Project Details

Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * Azure for Students

Resource Group * (New) practicalapiapp_group

[Create new](#)

Instance Details

Name * practicalapiapp

Publish * ☒ Code ☐ Docker Container ☐ Static Web App

Runtime stack * Select a runtime stack

Operating System ☒ Linux ☐ Windows

[Review + create](#) < Previous Next: Database >

Step5: Now click on create.

The screenshot shows the 'Create API App' wizard in the Microsoft Azure portal, specifically the 'Review + create' step. The breadcrumb navigation at the top reads 'Home > Create a resource > Marketplace > API App >'. The page title is 'Create API App'. Below the title, there are tabs for 'Basics', 'Database', 'Deployment', 'Networking', 'Monitoring', 'Tags', and 'Review + create'. The 'Review + create' tab is active. The 'Summary' section shows the API App icon, name, and 'Free sku' with an estimated price of 'Free'. A warning message states: 'Basic authentication for this app is currently disabled and may impact deployments. Click to learn more.' The 'Details' section lists the subscription, resource group, name, publish method, and runtime stack. The 'App Service Plan (New)' section shows the name and operating system. At the bottom, there are buttons for 'Create', '< Previous', 'Next >', and 'Download a template for automation'. The 'Create' button is highlighted in blue.

Microsoft Azure

Search resources, services, and docs (G+)

Home > Create a resource > Marketplace > API App >

Create API App

Basics Database Deployment Networking Monitoring Tags Review + create

Summary

API App by Microsoft

Free sku
Estimated price - Free

Basic authentication for this app is currently disabled and may impact deployments. Click to learn more.

Details

Subscription c3b5702f-0d45-4157-9746-27c96ed105e8

Resource Group practicalapiapp_group

Name practicalapiapp

Publish Code

Runtime stack .NET 8 (LTS)

App Service Plan (New)

Name ASP-practicalapiappgroup-9d63

Operating System Windows

[Create](#) < Previous Next > [Download a template for automation](#)

Step6: Now you can see your app deployment is in progress.

The screenshot shows the Microsoft Azure portal interface. The main heading is "Microsoft.Web-WebApp-Portal-35a64da3-8d85 | Overview". The left sidebar shows the navigation menu with "Overview" selected. The main content area displays "Deployment is in progress" with a status of "Running". It includes deployment details such as the deployment name, subscription, resource group, start time, and correlation ID. A table lists the resources being deployed, including "ASP-practicalapiapp" and "newWorkspaceTemplate". The right sidebar shows a "Notifications" panel with a message about the deployment progress.

Wait till the deployment is completed.

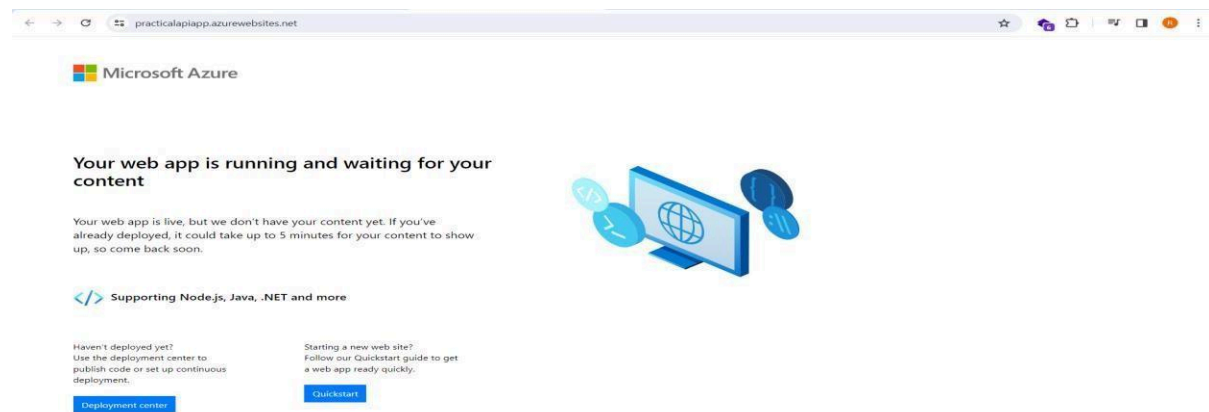
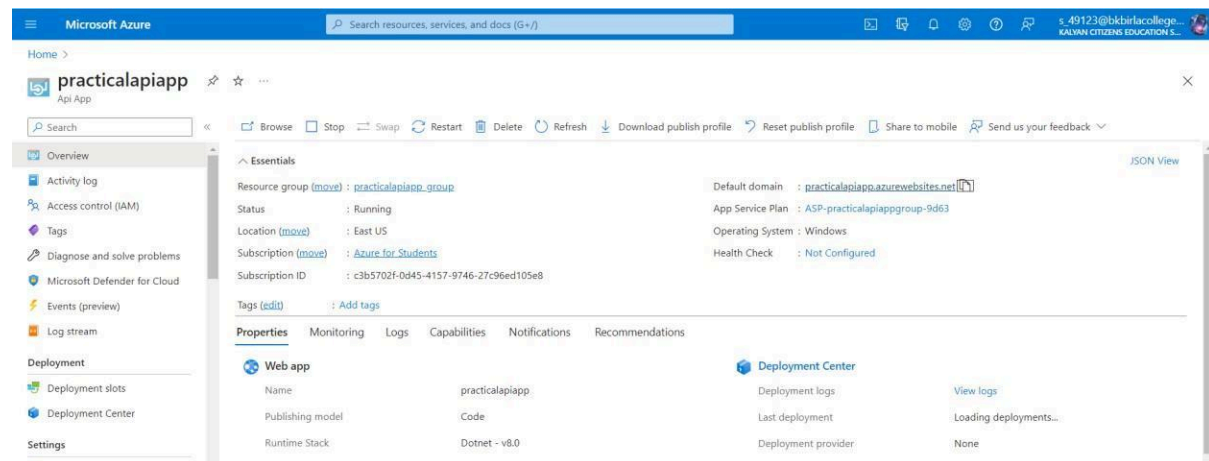
Step7: Once the deployment is completed click on “Go to resources”.

The screenshot shows the Microsoft Azure portal home page. The main heading is "Azure services". The left sidebar shows the navigation menu with "Overview" selected. The main content area displays a grid of Azure services including "Create a resource", "Quickstart Center", "Virtual machines", "App Services", "Storage accounts", "SQL databases", "Azure Cosmos DB", "Kubernetes services", and "Function App". The right sidebar shows a "Notifications" panel with a message about the deployment success.

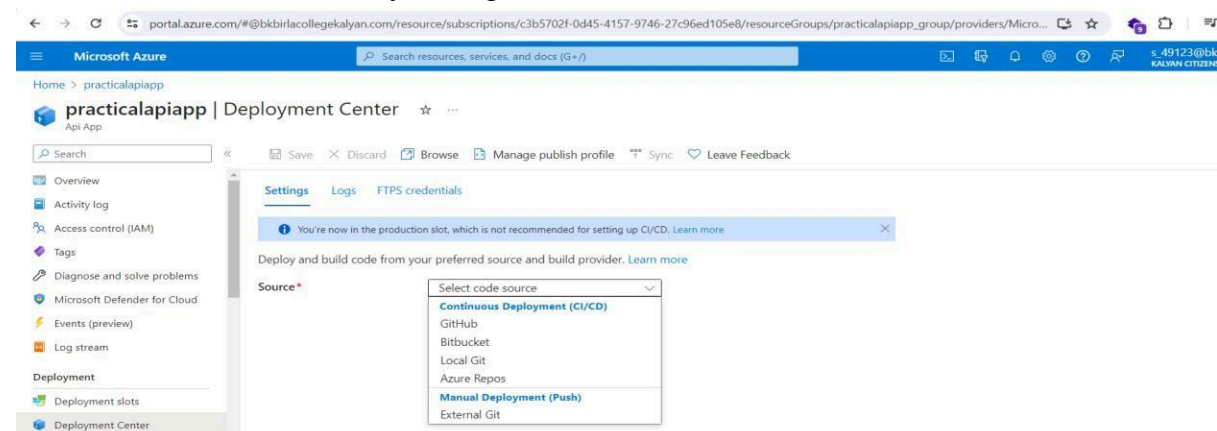
Now you will see a window like this:

The screenshot shows the Microsoft Azure portal interface for the "practicalapiapp" web application. The main heading is "practicalapiapp | Overview". The left sidebar shows the navigation menu with "Overview" selected. The main content area displays the "Overview" tab with essential information such as the resource group, status, location, subscription, and tags. It also shows the "Properties" tab with details about the web app, including the name, publishing model, runtime stack, and domains. The right sidebar shows a "Notifications" panel with a message about the deployment success.

Step8: Now you can click on your default domain and it will take you to your web applications default webpage.



Step9: Now click on deployment center and now on this tab you can add your code source via GitHub or from any other platform.



Step10: Now click on Configuration and here you can configure your application.

The screenshot displays the Microsoft Azure portal interface. The top navigation bar shows the user's profile and the search bar. The left sidebar contains the navigation menu with categories like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Microsoft Defender for Cloud, Events (preview), Log stream, Deployment, and Settings. The 'Configuration' page is selected, showing tabs for General settings, Default documents, Path mappings, and Error pages (preview). The 'General settings' tab is active, displaying 'Stack settings' and 'Platform settings'. The 'Stack settings' section includes 'Stack' (set to .NET) and '.NET version' (set to .NET 8 (LTS)). The 'Platform settings' section includes 'Platform' (set to 32 Bit), 'Managed pipeline version' (set to Integrated), and two toggle switches for 'SCM Basic Auth Publishi...' and 'FTP Basic Auth Publishi...', both of which are currently turned off. A link to 'Learn more' is provided at the bottom of the page.

Microsoft Azure

portal.azure.com/#@bkbirlacollegekalyan.com/resource/subscriptions/c3b5702f-0d45-4157-9746-27c96ed105e8/resourceGroups/practicalapiapp_group/providers/Micro...

practicalapiapp | Configuration

Search resources, services, and docs (G+)

Home > practicalapiapp

practicalapiapp | Configuration

Search

Refresh Save Discard Leave Feedback

Click here to upgrade to a higher SKU and enable additional features.

View and edit your application settings and connection strings from Environment variables. Click here to go to Environment Variables menu

General settings Default documents Path mappings Error pages (preview)

Stack settings

Stack .NET

.NET version .NET 8 (LTS)

Platform settings

Platform 32 Bit

Managed pipeline version Integrated

SCM Basic Auth Publishi... ☐ On ☒ Off

FTP Basic Auth Publishi... ☐ On ☒ Off

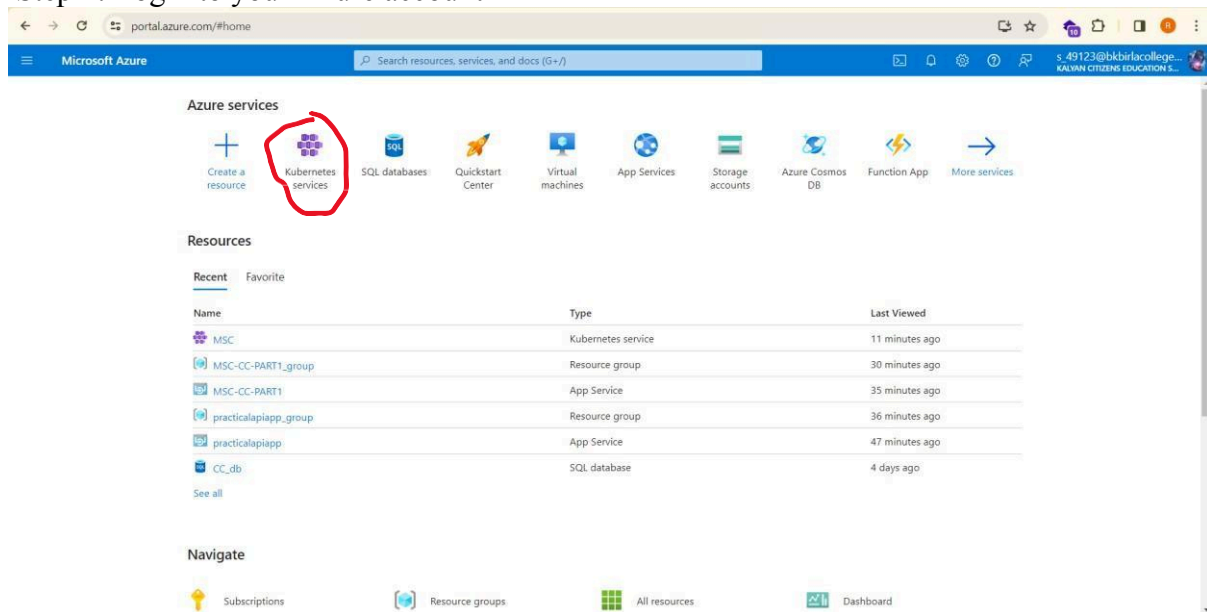
Disable basic authentication for FTP and SCM access. [Learn more](#)

Practical 2

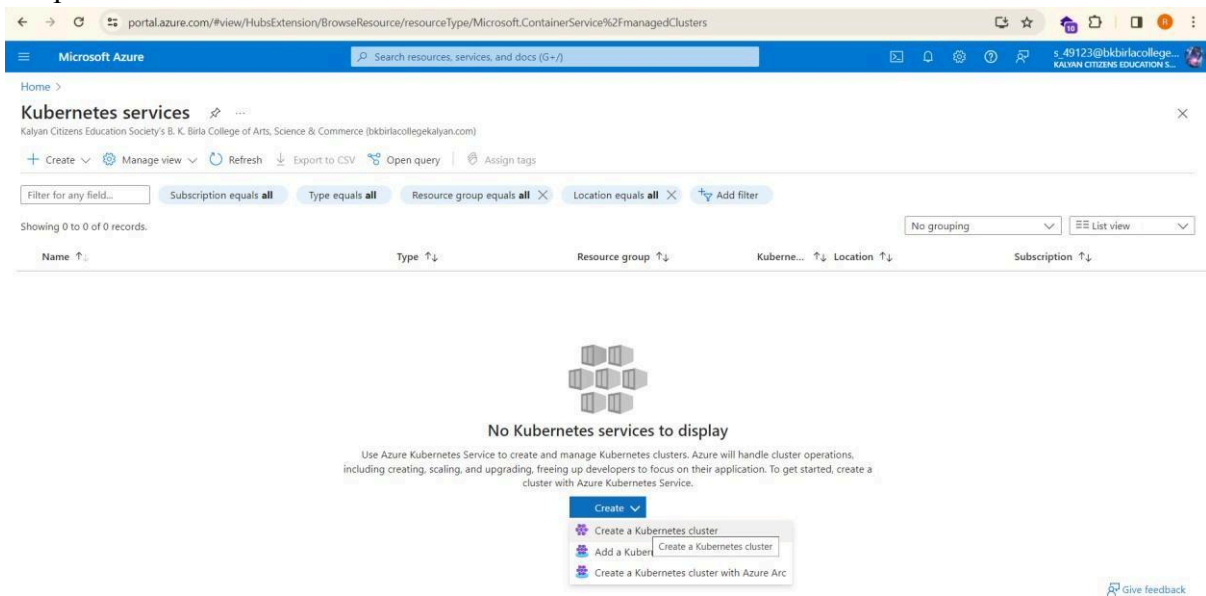
Create an Azure Kubernetes Cluster and integrate it with our Azure API App.

Theory: An Azure Kubernetes Cluster (AKS) is a managed container orchestration service provided by Microsoft Azure, allowing users to deploy, manage, and scale containerized applications using Kubernetes. With AKS, users can abstract away the complexities of managing Kubernetes infrastructure and focus on deploying and running their applications efficiently. It automates tasks such as cluster setup, scaling, and maintenance, providing built-in monitoring, logging, and security features. AKS integrates seamlessly with other Azure services, enabling users to leverage functionalities such as Azure Active Directory for authentication, Azure Monitor for monitoring, and Azure DevOps for continuous integration and deployment pipelines. This managed service empowers developers and operators to streamline the deployment and management of containerized applications, fostering agility, scalability, and reliability in cloud-native development workflows.

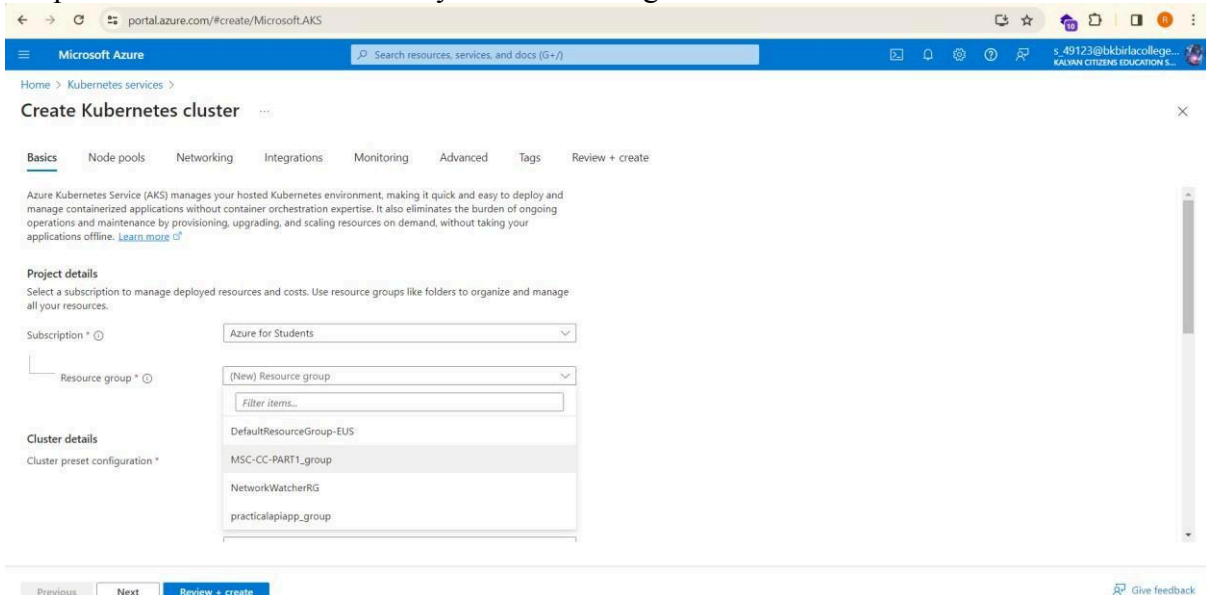
Step 1: Login to your Azure account



Step 2: Click on Kubernetes Services and then click on “Create a Kubernetes Cluster”



Step 3: Now select the Resource you want to integrate it with.



For e.g. here we have selected MSC-CC-PART1_group

Step 4: Now give you Kubernetes Cluster a name and Click on “Review + Create”

portal.azure.com/#create/Microsoft.AKS

Microsoft Azure

Home > Kubernetes services >

Create Kubernetes cluster

Basics Node pools Networking Integrations Monitoring Advanced Tags Review + create

Compare presets

Kubernetes cluster name * MSC

Region * (US) East US

Availability zones None

AKS pricing tier Free

Kubernetes version * 1.28.5 (default)

Automatic upgrade Enabled with patch (recommended)

Automatic upgrade scheduler Every week on Sunday (recommended)

Start on: Wed Apr 10 2024 00:00 +00:00 (Coordinated Universal Time)

Edit schedule

Node security channel type Node Image

Security channel scheduler Every week on Sunday (recommended)

Previous Next **Review + create** Give feedback

For e.g. here we have given it the name MSC.

portal.azure.com/#create/Microsoft.AKS

Microsoft Azure

Home > Kubernetes services >

Create Kubernetes cluster

Basics Node pools Networking Integrations Monitoring Advanced Tags **Review + create**

Validation in progress

Previous Next Create Give feedback

Step 5: Now click on Create.

portal.azure.com/#create/Microsoft.AKS

Microsoft Azure

Home > Kubernetes services >

Create Kubernetes cluster

Basics Node pools Networking Integrations Monitoring Advanced Tags **Review + create**

View automation template

Basics

Subscription	Azure for Students
Resource group	MSC-CC-PART1_group
Region	East US
Kubernetes cluster name	MSC
Kubernetes version	1.28.5
Automatic upgrade	patch
Automatic upgrade scheduler	Every week on Sunday (recommended)
Node security channel type	NodeImage
Security channel scheduler	Every week on Sunday (recommended)

Node pools

Node pools	1
Enable virtual nodes	Disabled

Previous Next **Create** Give feedback

Step 6: Now wait till the deployment finishes.

The screenshot shows the Azure portal interface for a deployment named 'microsoft.aks-1712672334684'. The main heading is 'Deployment is in progress'. Below this, there is a table for 'Deployment details' with columns 'Resource', 'Type', and 'Status', which currently shows 'No results.' To the right, a 'Notifications' panel displays a message: 'Deployment in progress... Running X Deployment to resource group 'MSC-CC-PART1_group' is in progress. a few seconds ago'. The left sidebar contains navigation links for Overview, Inputs, Outputs, and Template.

The screenshot shows the Azure portal interface after the deployment has completed. The main heading is 'Your deployment is complete'. It provides details such as 'Deployment name: micro...', 'Subscription: Azure for Stu...', 'Resource group: MSC-CC...', 'Start time: 4/9/2024, 7:50:05 PM', and 'Correlation ID: 3e370d92-514d-4b7f'. A 'Next steps' section includes a 'Go to resource' button. To the right, the 'Notifications' panel shows a green checkmark and the message: 'Deployment succeeded X Deployment 'microsoft.aks-1712672334684' to resource group 'MSC-CC-PART1_group' was successful. Go to resource Pin to dashboard a few seconds ago'. The left sidebar remains the same.

Step 7: Now you can see that our Azure API App is successfully connected with our Azure Kubernetes Clusters.

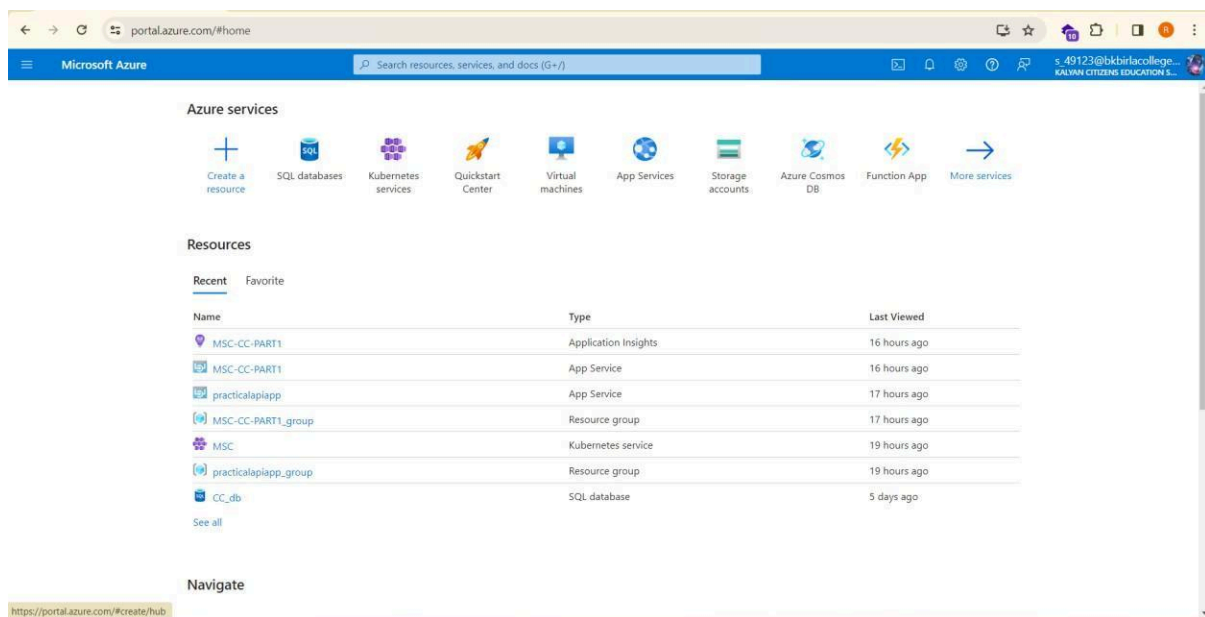
The screenshot shows the Azure portal interface for a Kubernetes service named 'MSC'. The main heading is 'Overview'. It displays various details including 'Resource group: MSC-CC-PART1_group', 'Status: Succeeded (Running)', 'Subscription: Azure for Students', 'Location: East US', and 'Subscription ID: c3b5702f-0d45-4157-9746-27c96ed105e8'. A table lists 'Kubernetes version: 1.28.5', 'API server address: msc-dns-g7rvfxp.hcp.eastus.azmk8s.io', 'Network type (plugin): Azure CNI', and 'Node pools: 1 node pool'. Below this, there are tabs for 'Get started', 'Properties', 'Monitoring', 'Capabilities (5)', 'Recommendations (0)', and 'Tutorials'. The 'Properties' tab is active, showing sections for 'Kubernetes services', 'Node pools', and 'Networking'. The 'Node pools' section shows 'Node pools: 1 node pool', 'Kubernetes versions: 1.28.5', and 'Node sizes: Standard_DS2_v2'. The 'Networking' section shows 'API server address: msc-dns-g7rvfxp.hcp.eastus.azmk8s.io', 'Network type (plugin): Azure CNI', 'Pod CIDR: -', 'Service CIDR: 10.0.0.0/16', 'DNS service IP: 10.0.0.10', 'Docker bridge CIDR: -', 'Network Policy: None', 'Load balancer: Standard', and 'UTTD configuration location: Not available'.

Practical 3

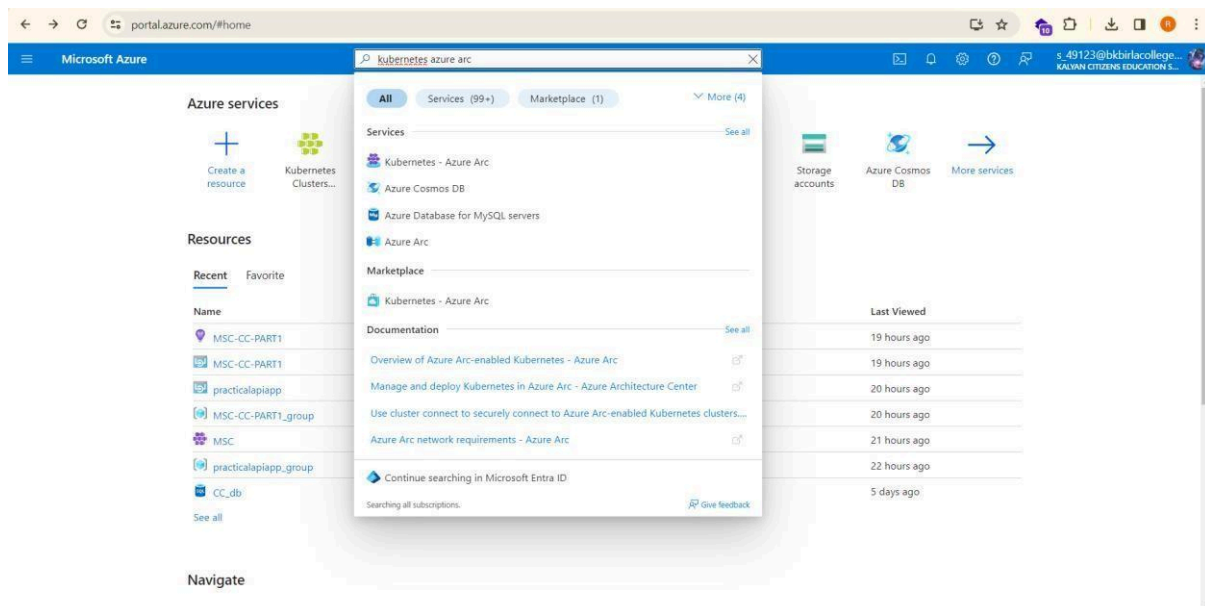
Create and add an Azure Kubernetes Cluster with Azure Arc

Theory: To create and add an Azure Kubernetes Cluster (AKS) with Azure Arc, you would first create an AKS cluster using the Azure portal, Azure CLI, or Azure Resource Manager templates. Then, you would enable Azure Arc for Kubernetes on the AKS cluster by navigating to the Azure Arc-enabled Kubernetes service in the Azure portal and selecting "Add Azure Kubernetes Cluster." Follow the prompts to specify the subscription, resource group, and other details, and Azure Arc will automatically onboard the AKS cluster, allowing you to manage and monitor it alongside your other Arc-enabled resources from a centralized Azure management interface.

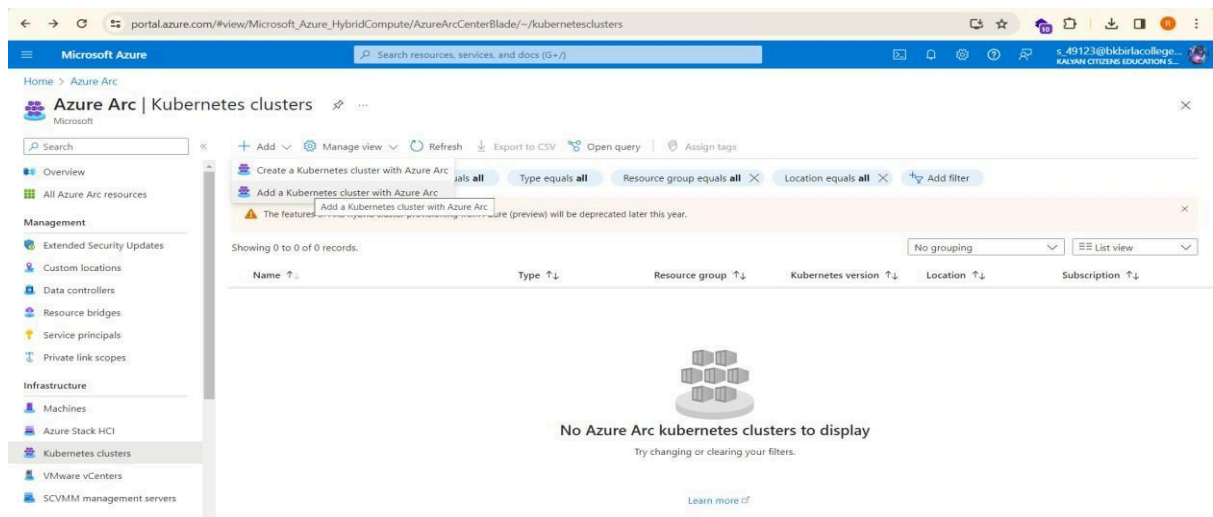
Step 1: Login to your Azure account on Azure Portal.



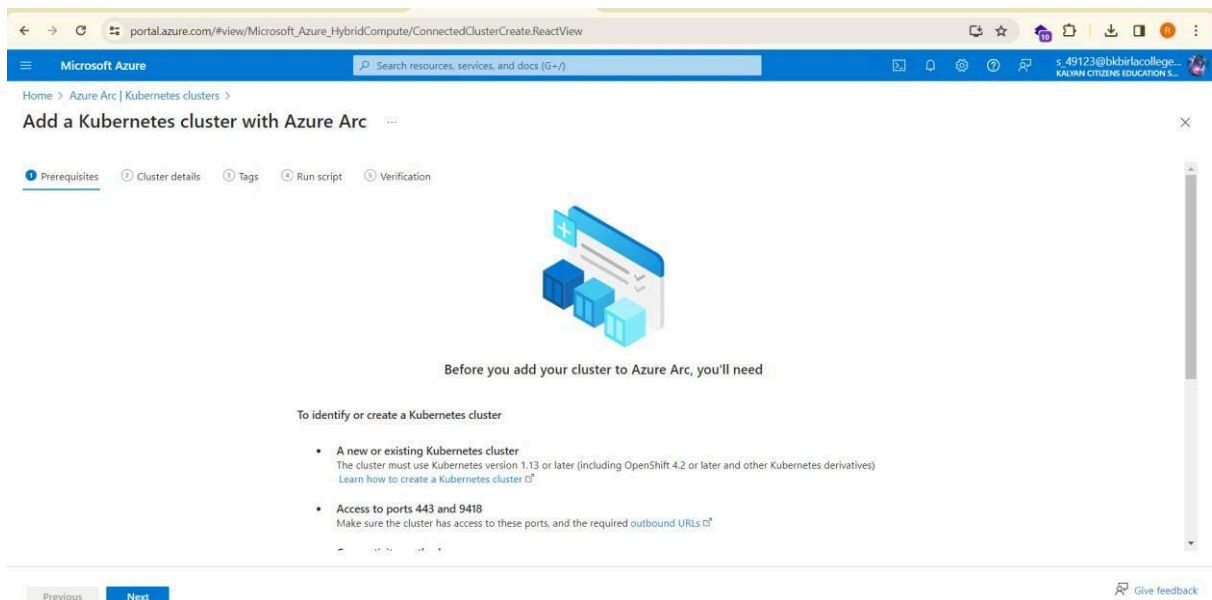
Step 2: click on search and search for Kubernetes Azure Arc and click on it.



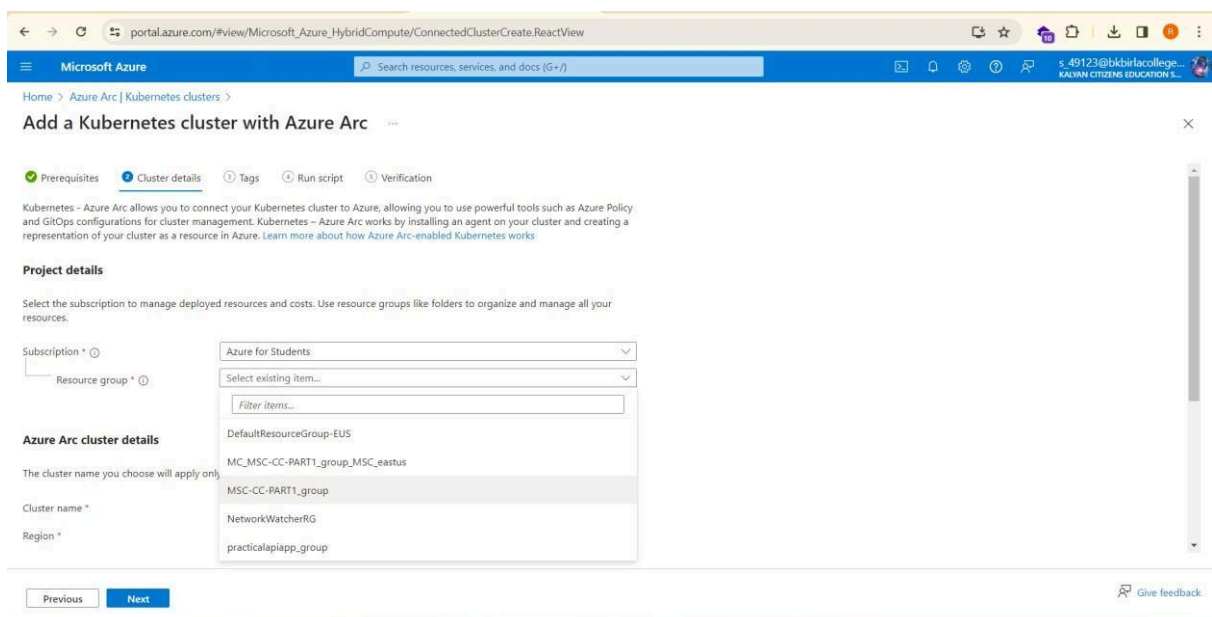
Step 3: Now click on Add a “Kubernetes Cluster with Azure Arc”



Step 4: Now click on “next.”



Step 5: Now select a Resource group for your Kubernetes cluster.



For e.g. here we have selected MSC-CC-PART1_group

Step 6: Now give your Kubernetes Cluster a name and click on next.

portal.azure.com/#view/Microsoft_Azure_HybridCompute/ConnectedClusterCreate.ReactView

Microsoft Azure Search resources, services, and docs (G+)

Home > Azure Arc | Kubernetes clusters >

Add a Kubernetes cluster with Azure Arc

Azure Arc cluster details

The cluster name you choose will apply only to Azure. It will not affect your cluster settings outside Azure.

Cluster name * MSC_CC_1

Region * (US) East US

Network connectivity

Choose the type of connection you want to use to connect your cluster to Azure. [Learn more](#)

Connectivity method *

- ☒ Public endpoint
The cluster can directly access the internet.
- ☐ Proxy server
The cluster uses a proxy cluster to access the internet.
- ☐ Private endpoint (preview)
Connect the cluster to Azure using a private endpoint in an Azure virtual network. Requires Express Route or a VPN connection.

Previous Next Give feedback

For e.g. here we have given the name “MSC_CC_1”

Step 7: Now here you can add physical location tag, or you can directly move to the next step by clicking on next.

portal.azure.com/#view/Microsoft_Azure_HybridCompute/ConnectedClusterCreate.ReactView

Microsoft Azure Search resources, services, and docs (G+)

Home > Azure Arc | Kubernetes clusters >

Add a Kubernetes cluster with Azure Arc

Prerequisites Cluster details Tags Run script Verification

To manage and create custom views of your resources, assign tags. [Learn more about tags](#)

Physical location tags

Start with these options for physical location types, change them to suit your needs, or create your own. If you leave the value field blank for these options, the tags will not be created.

Name	Value	Resource
Datacenter		Kubernetes - Azure Arc
City		Kubernetes - Azure Arc
StateOrDistrict		Kubernetes - Azure Arc
CountryOrRegion		Kubernetes - Azure Arc
		Kubernetes - Azure Arc

Custom tags

Previous Next Give feedback

Step 8: Here you can see that the bash script is generated successfully.

portal.azure.com/#view/Microsoft_Azure_HybridCompute/ConnectedClusterCreate.ReactView

Microsoft Azure Search resources, services, and docs (G+)

Home > Azure Arc | Kubernetes clusters >

Add a Kubernetes cluster with Azure Arc

Prerequisites Cluster details Tags Run script Verification

1. Download or copy the following script

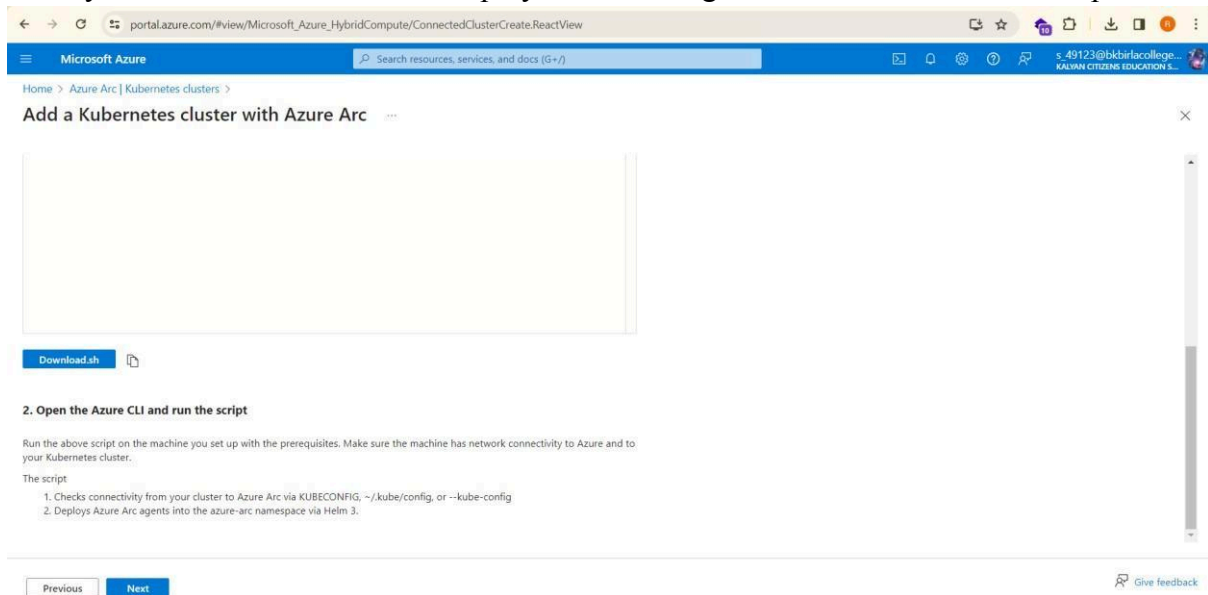
Script type *

- ☒ Bash
- ☐ PowerShell

```
1 # This script creates an Azure Arc resource to connect a Kubernetes cluster to
2 # Azure
3 # Documentation: https://aka.ms/AzureArcK8sDocs
4 # Log into Azure
5 az login --use-device-code
6 # Set Azure subscription
7 az account set --subscription "c3b5702f-0d45-4157-9746-27c96ed105e8"
8 # Create connected cluster
9 az connectedk8s connect --name "MSC_CC_1" --resource-group "MSC-CC-PART1_group"
10 --location "eastus" --correlation-id "c18ab9db-685e-48e7-ab55-12588447b0ed"
11 --tags "Datacenter City StateOrDistrict CountryOrRegion"
```

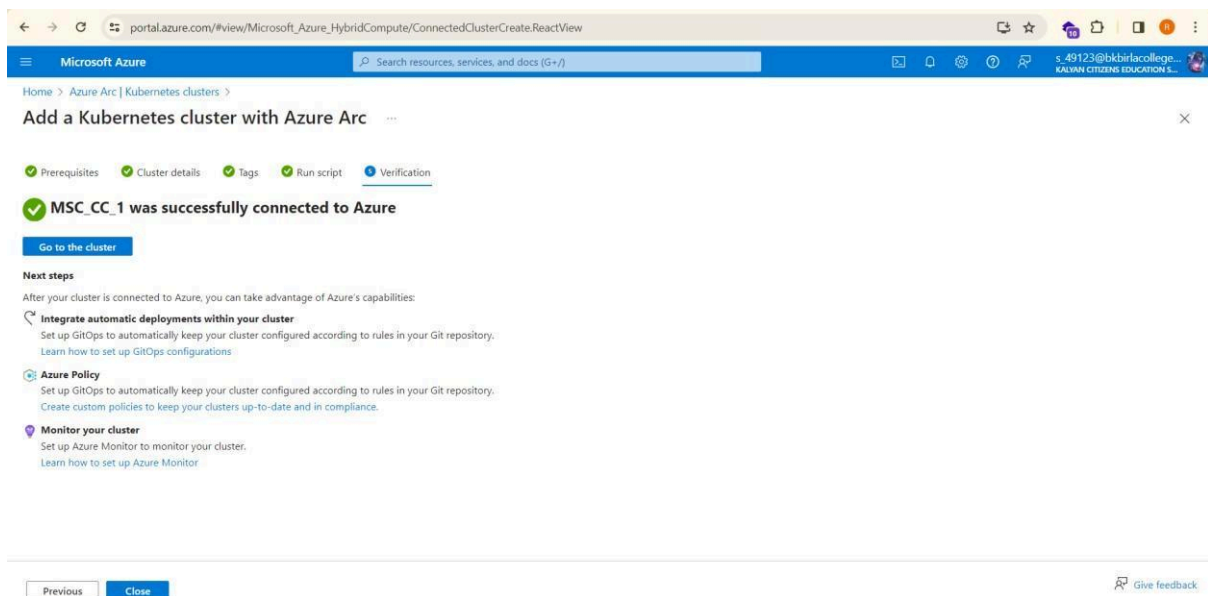
Previous Next Give feedback

Here you can even download and deploy Azure Arc agents into the azure-arc namespace.

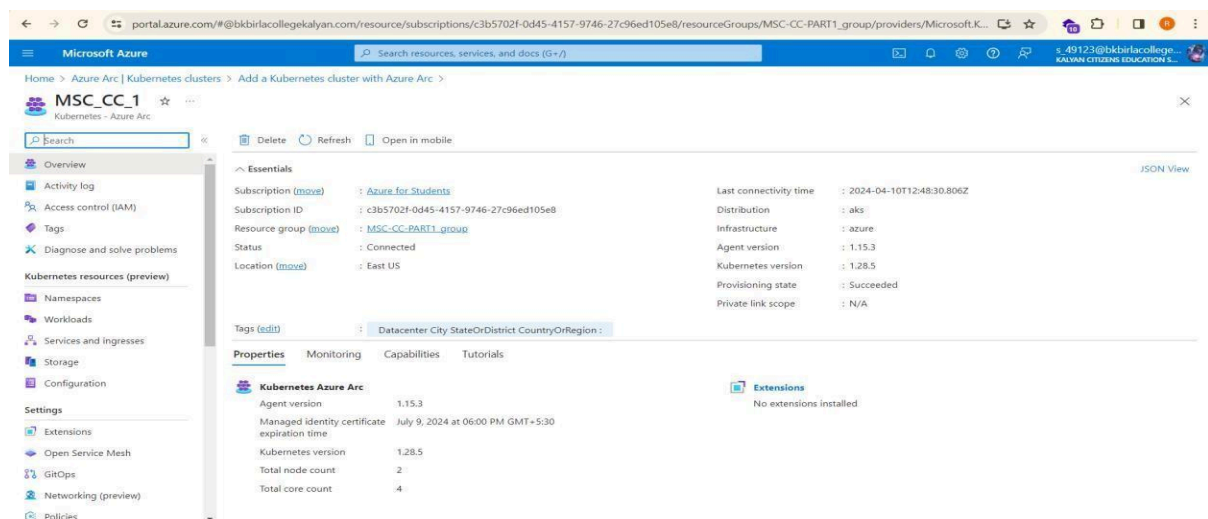


And now click on Next

Step 9: Now you can see that Kubernetes Cluster is successfully connected to Azure.



Now click on cluster and you will be able to see your Azure Kubernetes Cluster.

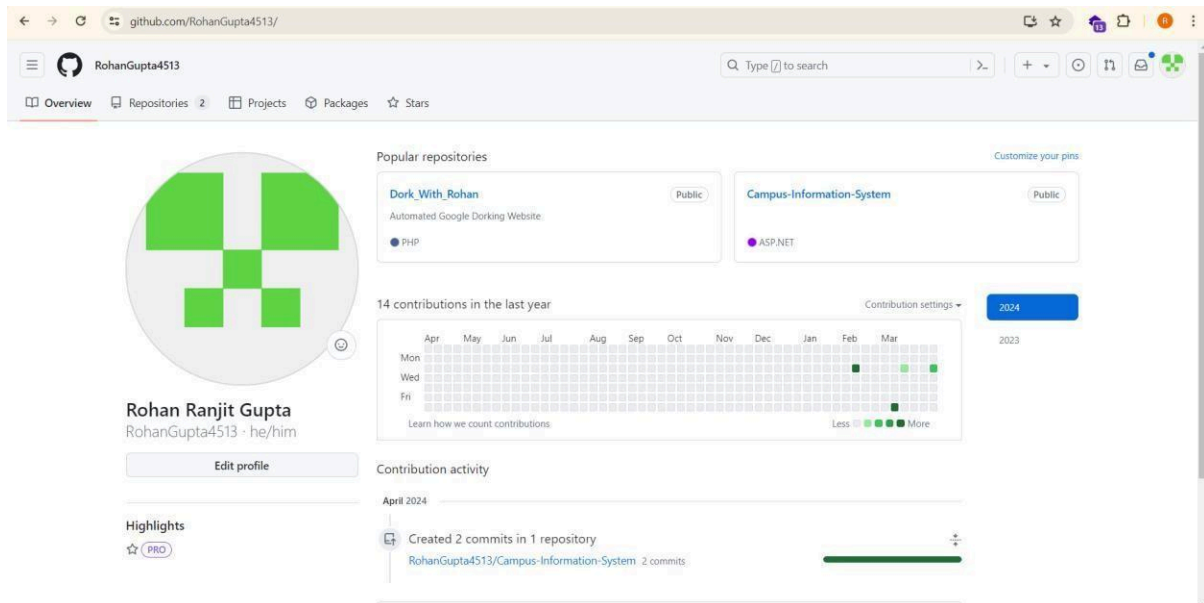


Practical 4

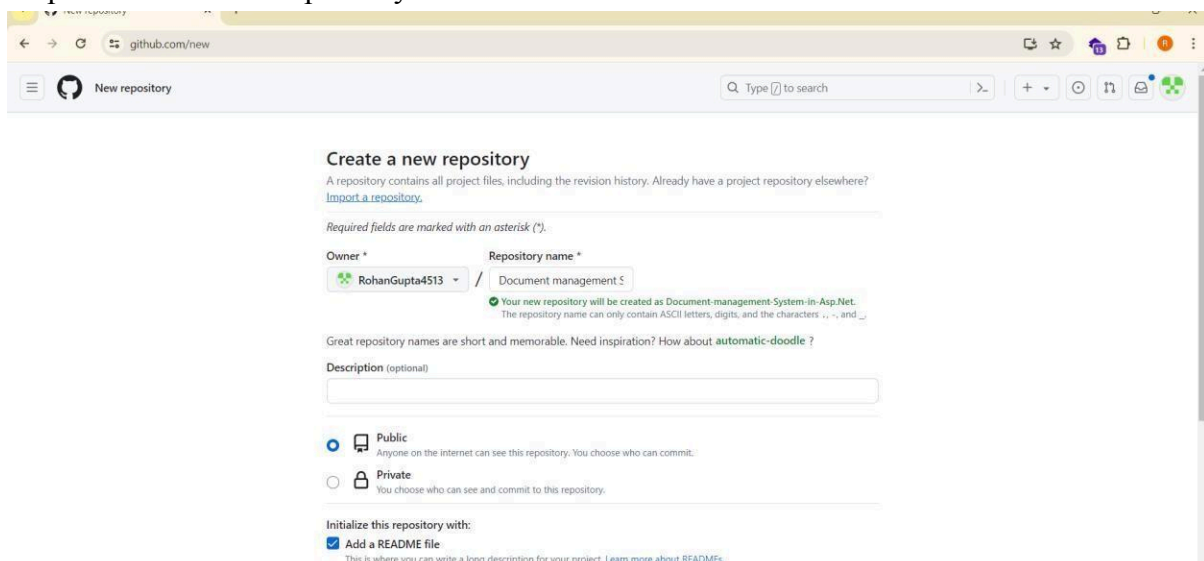
Create a GitHub Repository for our .NET DevOps Web Application

Theory: To create a GitHub repository for your .NET DevOps web application, navigate to GitHub and sign in. Click on the "+" icon in the top-right corner and select "New repository." Provide a name, description, and choose visibility settings for your repository. Optionally, initialize the repository with a README file. If your application already exists locally, follow the instructions to push an existing repository from the command line. Otherwise, clone the repository to your local machine, add your .NET project files, commit the changes, and push them to GitHub. This repository will serve as a centralized location for collaboration, version control, and automated DevOps workflows for your .NET web application.

Step 1: Login to your GitHub account.

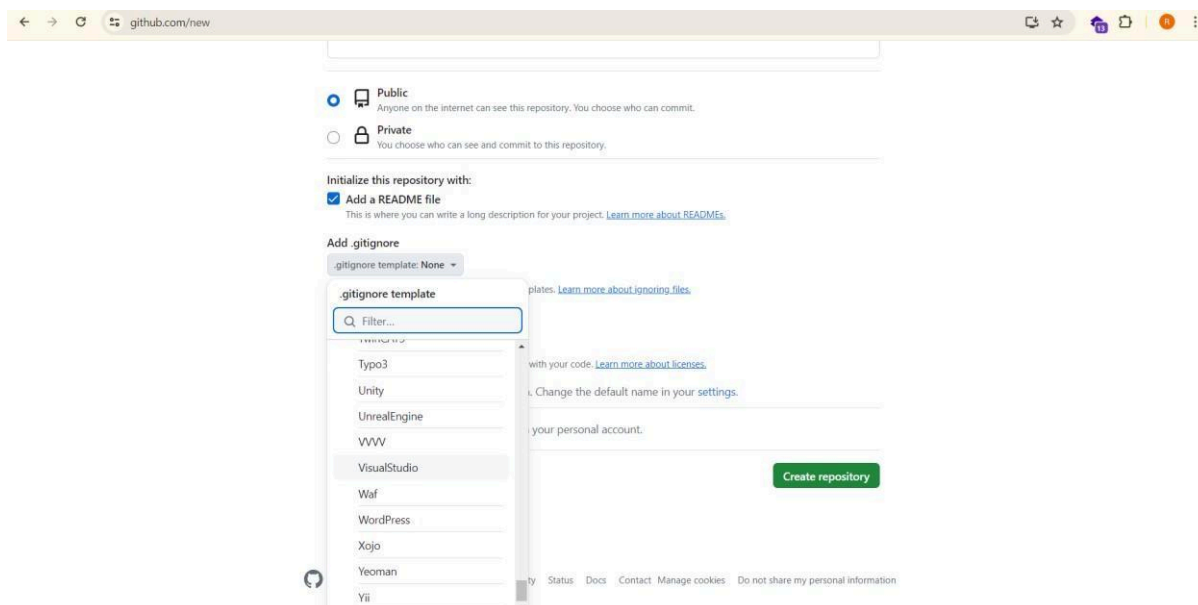


Step 2: Create a new repository.



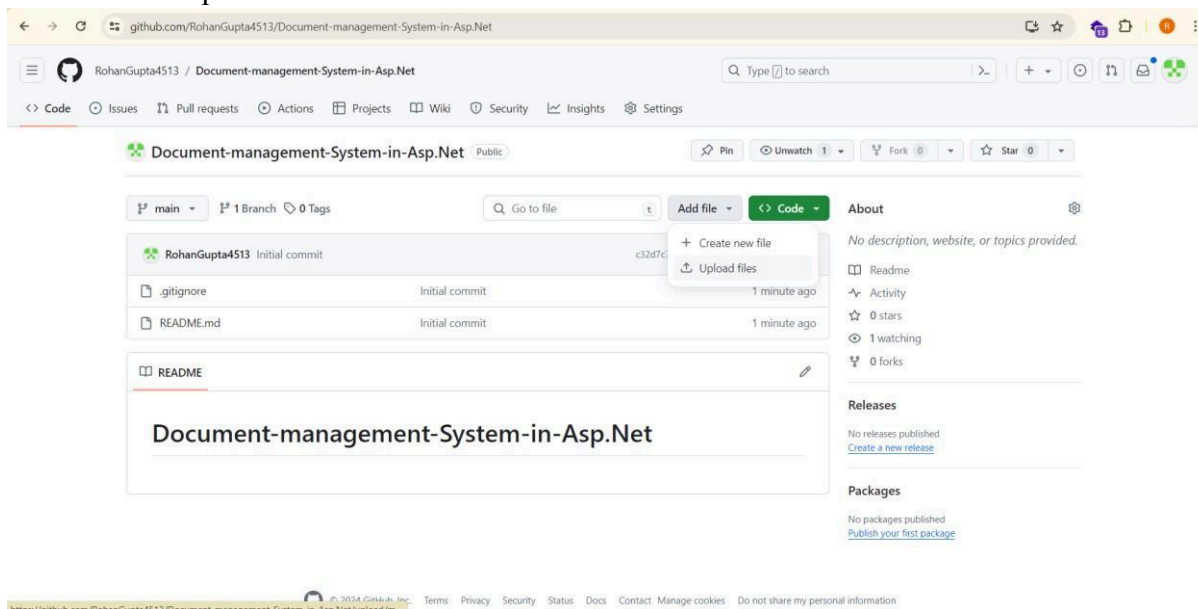
And give your repository a name. Here we have given the name
“_Project_E-Blogging-asp-net-project”

Step 3: Now add a README file and add “. gitignore” and select “VisualStudio”

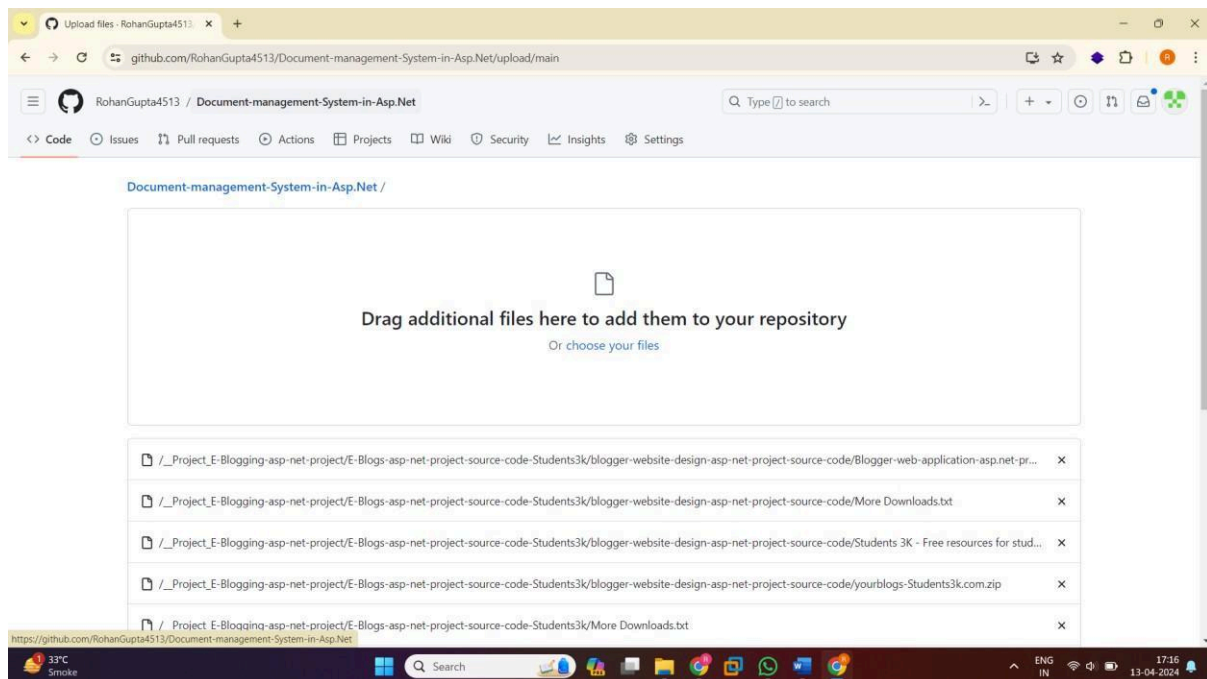


And then click on “Create repository”

Step 4: After successfully creating the repository click on “Add file”
and then on “Upload File”

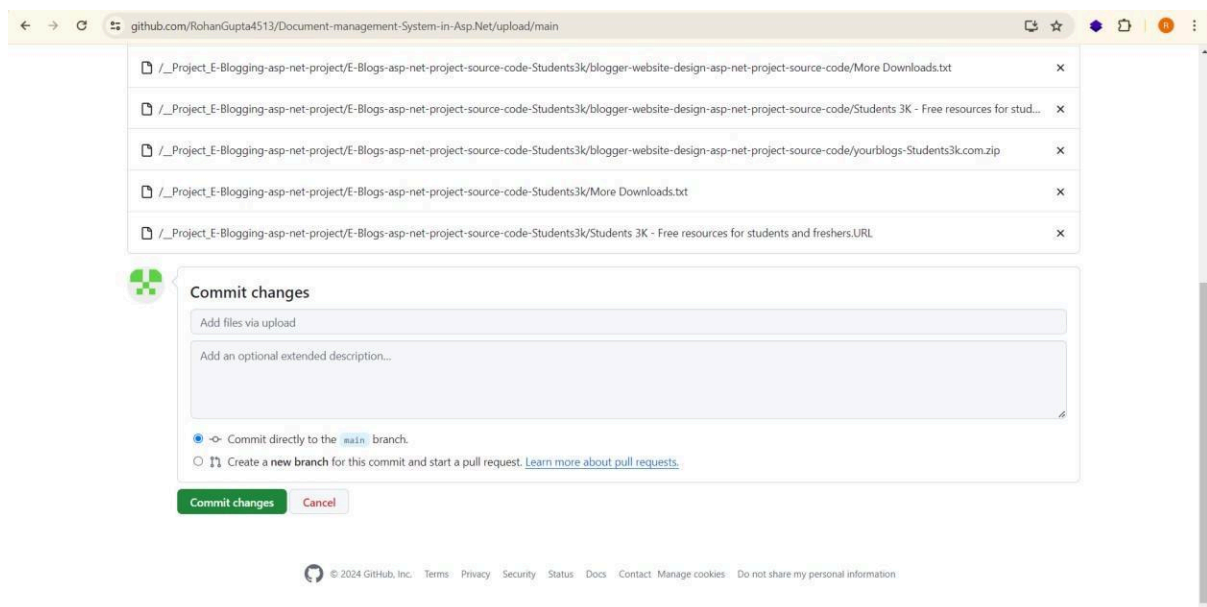


Step 5: Now upload your file here or just simply drag and drop your folder to upload it.



Here we are going to take the project named as “_Project_E-Blogging-asp-net-project”

Step 6: Once the project is uploaded click on “Commit Changes”



Step 7: Here we have successfully uploaded our project.

The screenshot shows a GitHub repository page for the project '_Project_E-Blogging-asp-net-project' by user 'RohanGupta4513'. The repository is public and has 1 branch (main) and 0 tags. The commit history shows three commits: an initial commit for '.gitignore' 1 hour ago, an upload of '_Project_E-Blogging-asp-net-project/E-Blogs...' 7 minutes ago, and an update to 'README.md' 3 minutes ago. The README file is currently empty. The right sidebar shows the repository has 0 stars, 1 watcher, and 0 forks. It also lists sections for Releases, Packages, and Deployments, with the latest deployment 'github-pages' made 2 minutes ago.

github.com/RohanGupta4513/_Project_E-Blogging-asp-net-project

RohanGupta4513 / _Project_E-Blogging-asp-net-project

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

_Project_E-Blogging-asp-net-project Public

main 1 Branch 0 Tags

Go to file Add file Code

RohanGupta4513 Update README.md a244e1e · 3 minutes ago 3 Commits

- _Project_E-Blogging-asp-net-project/E-Blogs... Add files via upload 7 minutes ago
- .gitignore Initial commit 1 hour ago
- README.md Update README.md 3 minutes ago

README

_Project_E-Blogging-asp-net-project

About

No description, website, or topics provided.

Readme Activity 0 stars 1 watching 0 forks

Releases

No releases published
[Create a new release](#)

Packages

No packages published
[Publish your first package](#)

Deployments 1

github-pages 2 minutes ago

https://github.com/RohanGupta4513/_Project_E-Blogging-asp-net-project

Practical 5

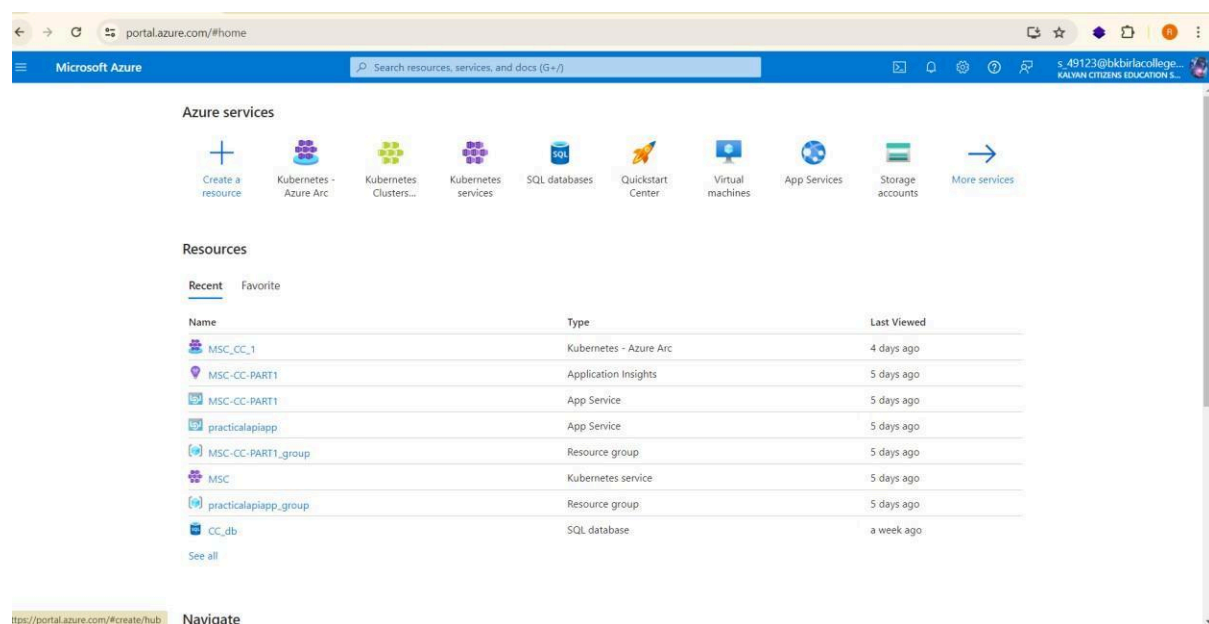
Deploy the GitHub Repository in your Azure API App.

Theory: Set up your GitHub repository: Begin by creating a GitHub repository that houses your API code. If you've already developed your API locally, initialize a Git repository in your project directory and link it to your GitHub account. You can do this by running `git init` to initialize a new Git repository and then `git remote add origin <repository_URL>` to connect it to your GitHub repository.

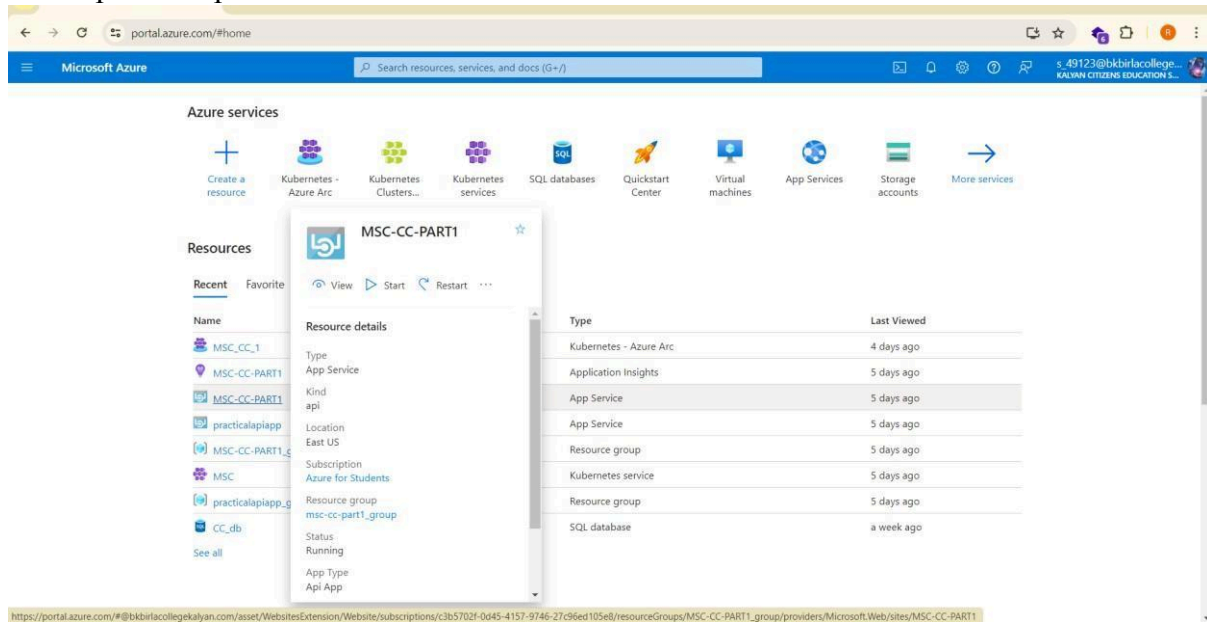
If you're starting fresh, create a new repository on GitHub. Give it a meaningful name and description, and consider adding a README file to provide information about your API. You can also add any necessary files and directories for your API code.

Ensure your repository is properly structured and organized, with clear documentation and a license if applicable. This foundational step ensures that your codebase is ready for collaboration and deployment.

Step 1: Login to Azure portal with your Azure account.

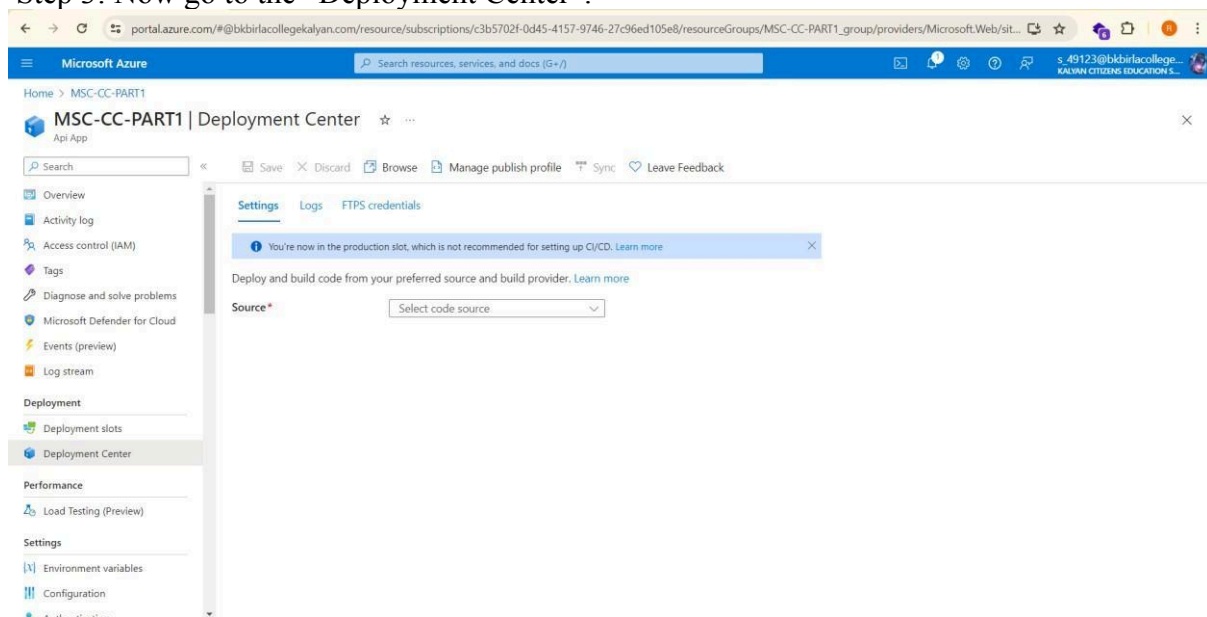


Step 2: Open your API App in which you want to deploy your project which we have created in the previous practical.

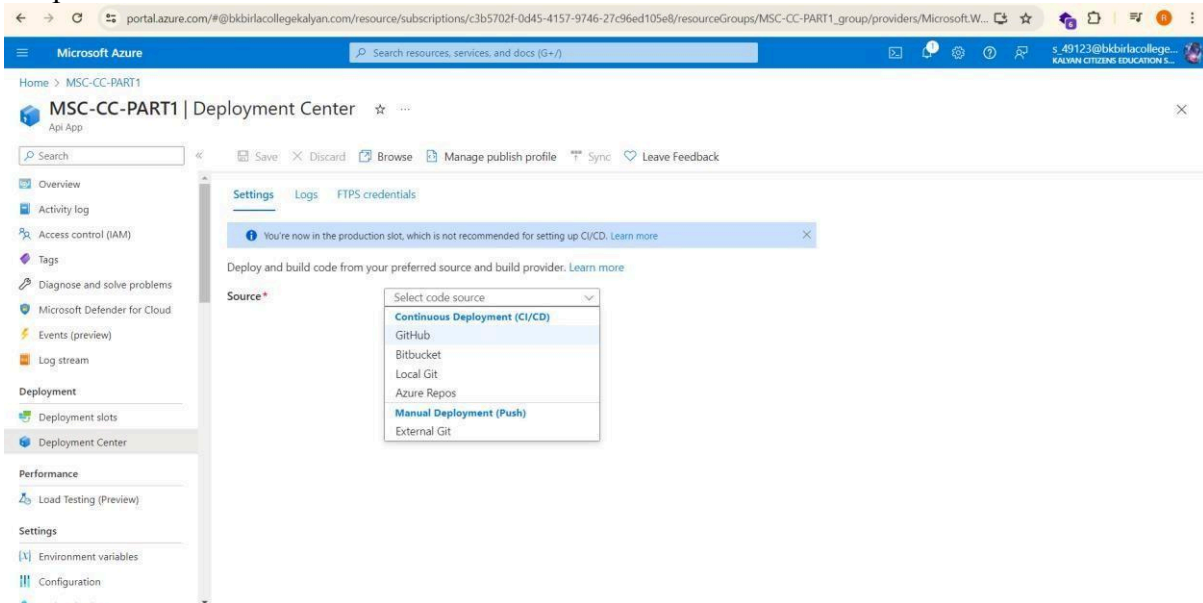


Here we have selected “MSC-CC-PART1” as our API App in which we will be deploying our projects.

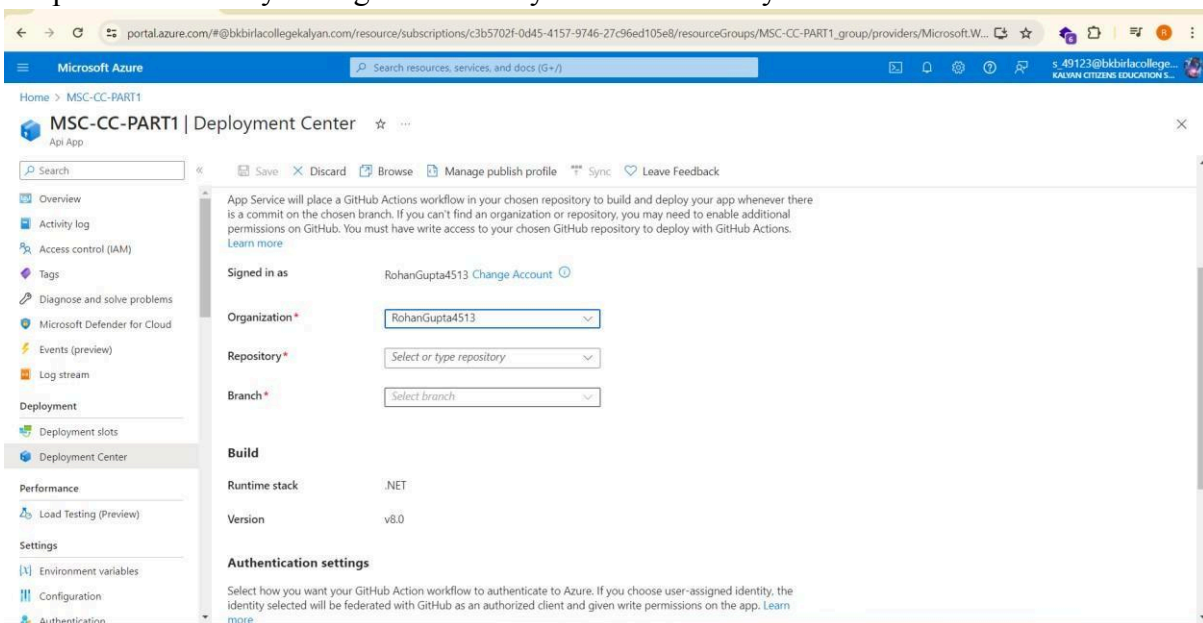
Step 3: Now go to the “Deployment Center”.



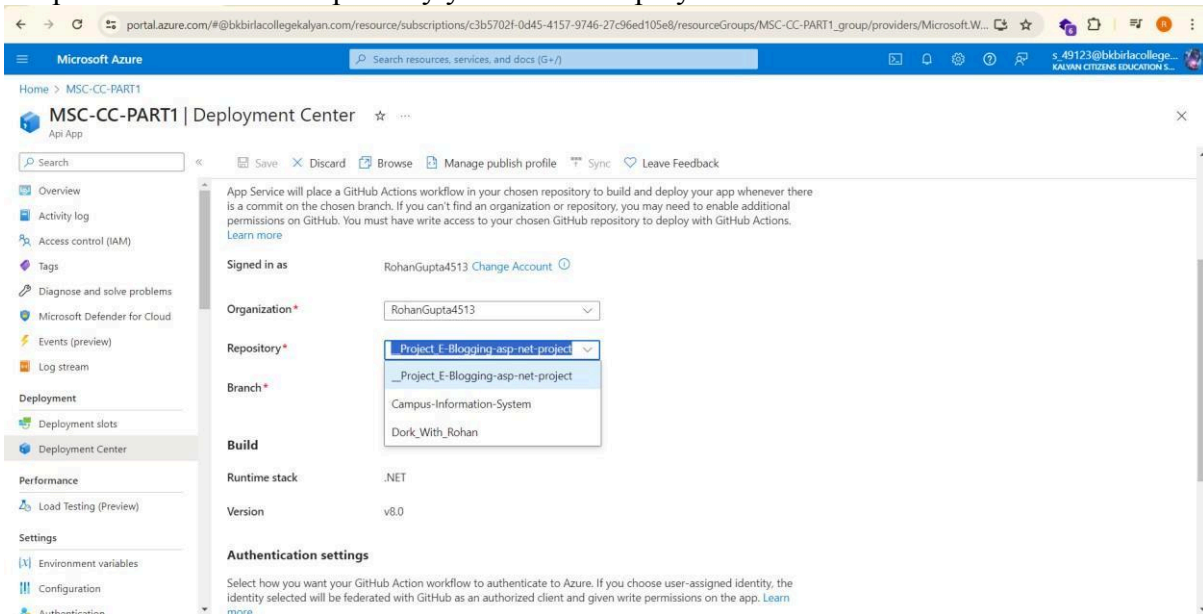
Step 4: Now select the Source as GitHub.



Step 5: Now select your organization as your username of your GitHub account.



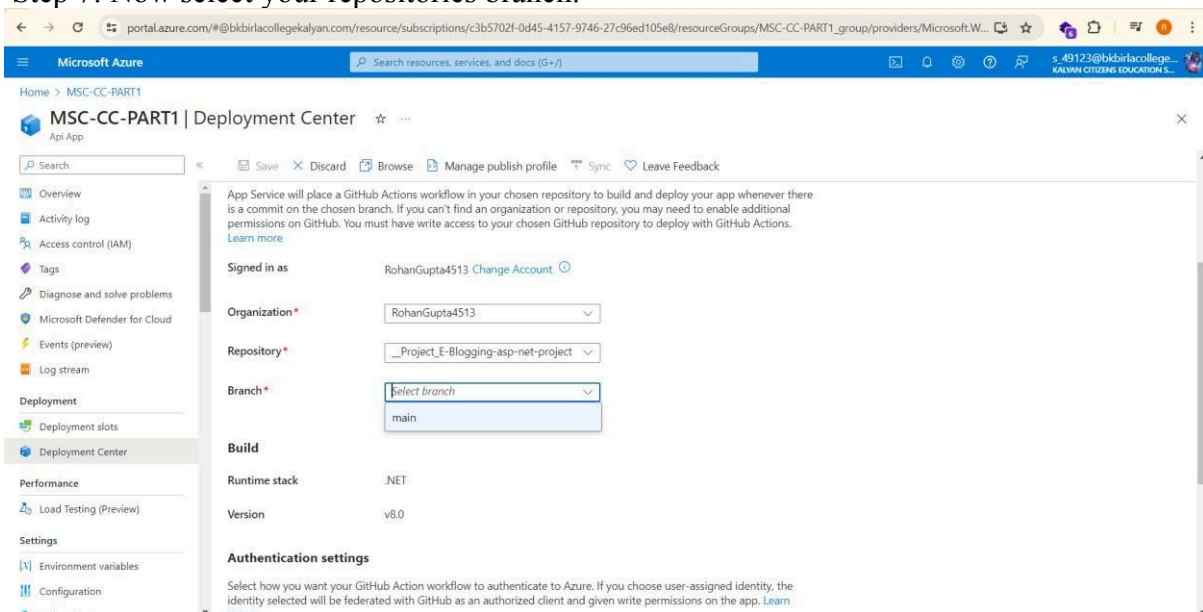
Step 6: Now select the repository you want to deploy.



Here we have selected the repository named “asp-net-project”.

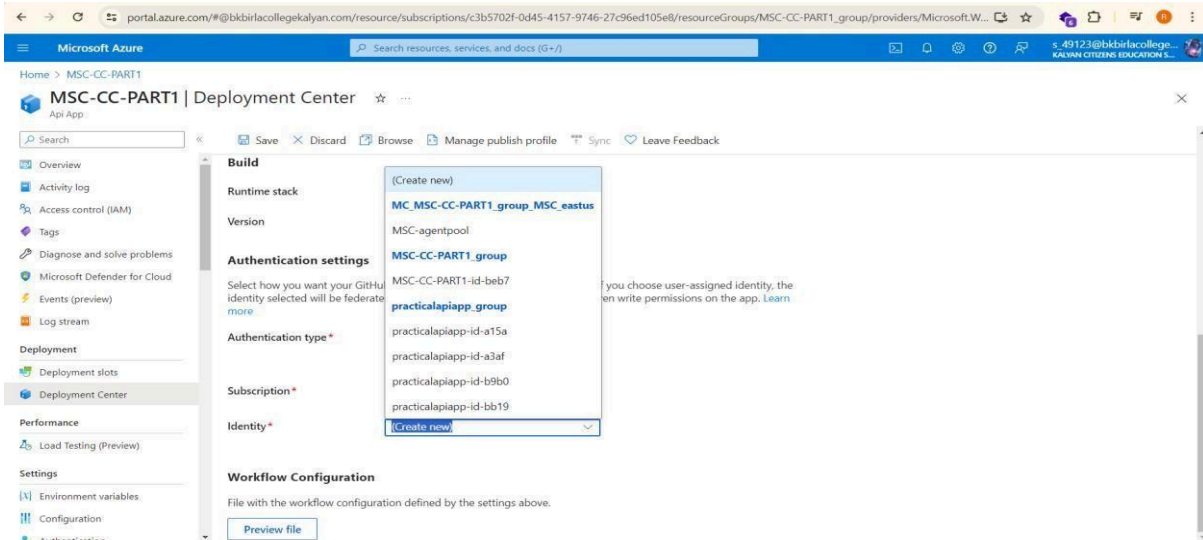
Project_E-Blogging-

Step 7: Now select your repositories branch.



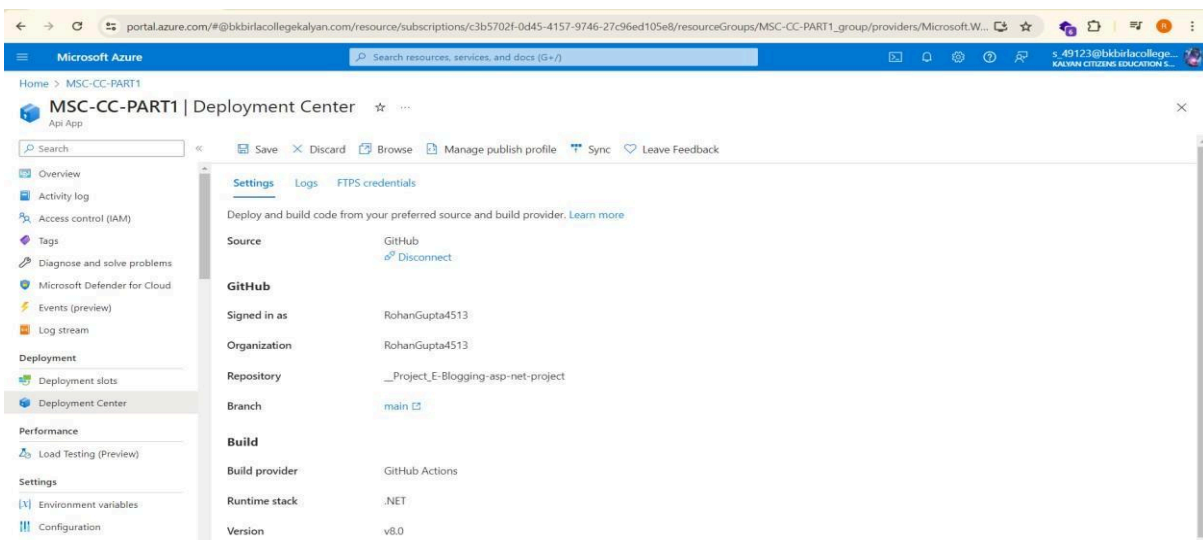
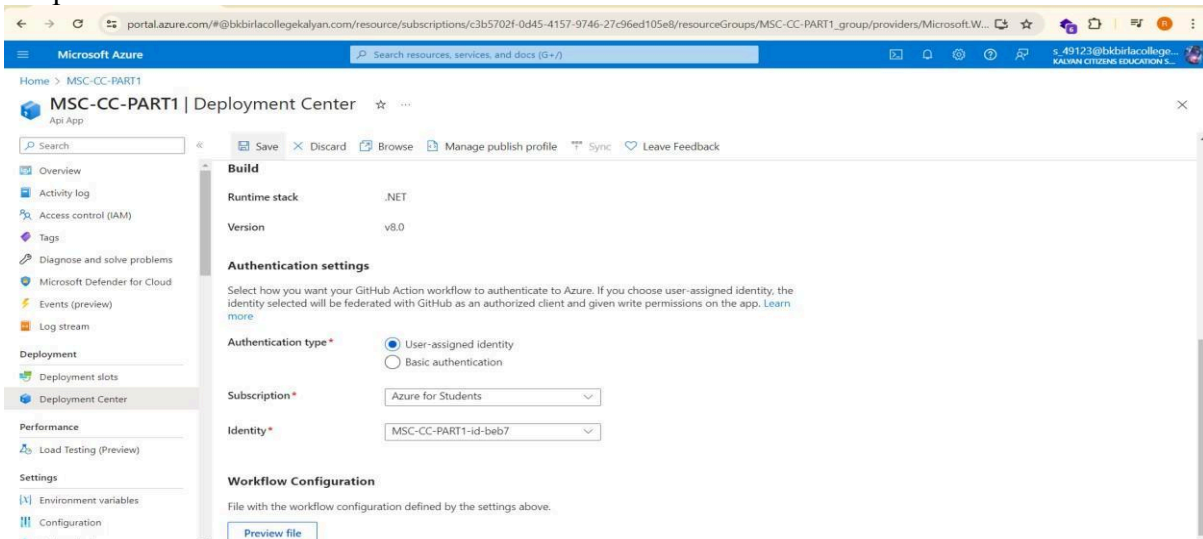
Here we have selected the branch as “main”.

Step 8: Now select the identity of your project or create a new identity of your project.



Here we have selected the identity as “MSC-CC-PART1-id-beb7”

Step 9: Now click on save.

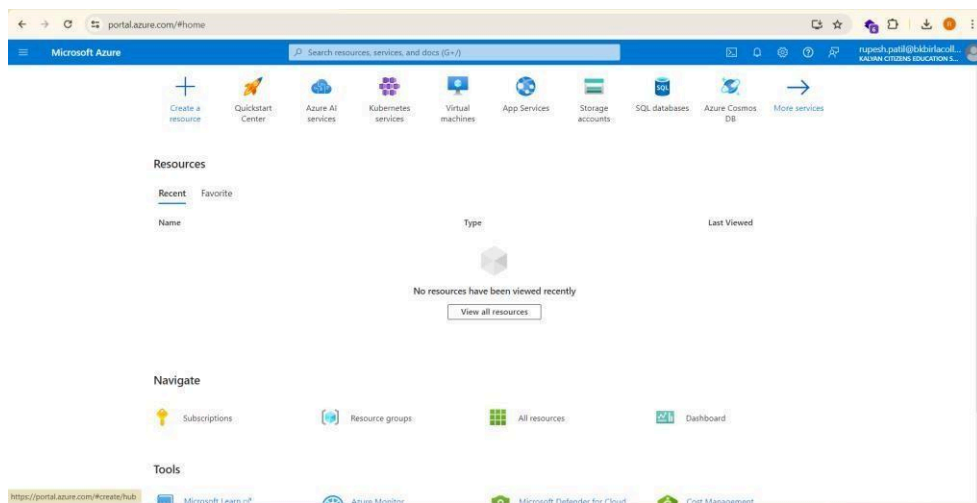


Practical 6

Implement API Authentication Mechanism in your Azure API App

Theory: To implement API authentication in your Azure API App, first, navigate to the Azure Portal and select your API App. Under the "Settings" section, configure the "Authentication / Authorization" option by turning on the "App Service Authentication" switch. Choose the authentication provider you prefer, such as Azure Active Directory, Facebook, Google, or Twitter. For Azure Active Directory, register your application in the Azure AD to obtain the Client ID and Tenant ID, then configure these in the authentication settings. Set the API to require authentication by selecting the appropriate action under "Action to take when the request is not authenticated." This ensures that every request to your API must include a valid token issued by your chosen provider, effectively securing your API endpoints.

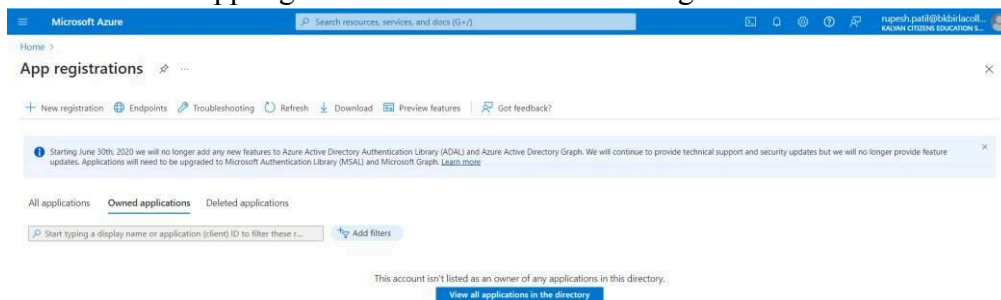
Step 1: Sign in to the Azure portal, navigate to Azure Active Directory in the Azure portal.



Step 2:

1. Register a new application:

Go to App registrations and click on New registration.



Enter a name for your application (e.g., "MyAPIApp"). Choose the supported account types (Single tenant or Multitenant).

Microsoft Azure

Home > App registrations > Register an application

The user-facing display name for this application (this can be changed later):

MyAPIApp

Supported account types

Who can use this application or access this API?

- ☒ Accounts in this organizational directory only (Kalyan Citizens Education Society's B. K. Birla College of Arts, Science & Commerce only - Single tenant)
- ☐ Accounts in any organizational directory (Any Microsoft Entra ID tenant - Multitenant)
- ☐ Accounts in any organizational directory (Any Microsoft Entra ID tenant - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)
- ☐ Personal Microsoft accounts only

Help me choose...

Redirect URI (optional)

We'll return the authentication response to this URI after successfully authenticating the user. Providing this now is optional and it can be changed later, but a value is required for most authentication scenarios.

Select a platform: Web

e.g. https://example.com/auth

By proceeding, you agree to the Microsoft Platform Policies

Register

Add a Redirect URI if needed (for testing purposes, you can use http://localhost).

Redirect URI (optional)

We'll return the authentication response to this URI after successfully authenticating the user. Providing this now is optional and it can be changed later, but a value is required for most authentication scenarios.

Web

http://localhost

Register an app you're working on here. Integrate gallery apps and other apps from outside your organization by adding from [Enterprise applications](#).

2. Configure API permissions:

Microsoft Azure

Home > App registrations > MyAPIApp

MyAPIApp | API permissions

Search

Overview

Quickstart

Integration assistant

Manage

- Branding & properties
- Authentication
- Certificates & secrets
- Token configuration
- API permissions
- Expose an API
- App roles
- Owners
- Roles and administrators
- Manifest

Support + Troubleshooting

Troubleshooting

Granting tenant-wide consent may revoke permissions that have already been granted tenant-wide for that application. Permissions that users have already granted on their own behalf aren't affected. [Learn more](#)

The "Admin consent required" column shows the default value for an organization. However, user consent can be customized per permission, user, or app. This column may not reflect the value in your organization, or in organizations where this app will be used. [Learn more](#)

Configured permissions

Applications are authorized to call APIs when they are granted permissions by users/admins as part of the consent process. The list of configured permissions should include all the permissions the application needs. [Learn more about permissions and consent](#)

+ Add a permission

Grant admin consent for Kalyan Citizens Education Society's B. K. Birla College of Arts, Science & Commerce

API / Permissions name	Type	Description	Admin consent required	Status
Microsoft Graph (1)				
User.Read	Delegated	Sign in and read user profile	No	...

To view and manage consented permissions for individual apps, as well as your tenant's consent settings, try [Enterprise applications](#).

Click on Add permission and select My APIs. Select your API and choose the required permissions.

Microsoft Azure

Home > App registrations > MyAPIApp

MyAPIApp | API permissions

Search resources, services, and docs (G+)

Granting tenant-wide consent may revoke permissions that have all the permissions the application needs. [Learn more](#)

The "Admin consent required" column shows the default value for all organizations where this app will be used. [Learn more](#)

Configured permissions

Applications are authorized to call APIs when they are granted permission. All the permissions the application needs. [Learn more about permissions](#)

+ Add a permission ✓ Grant admin consent for Kalyan Citizen

API / Permissions name	Type	Description
Microsoft Graph (1)		
User.Read	Delegated	Sign in and read

To view and manage consented permissions for individual apps, as well as

Request API permissions

Select an API

Microsoft APIs APIs my organization uses My APIs

Applications that expose permissions are shown below

Name	Application (client) ID
No results.	

3. Generate a client secret:

Microsoft Azure

Home > App registrations > MyAPIApp

MyAPIApp | Certificates & secrets

Search resources, services, and docs (G+)

Credentials enable confidential applications to identify themselves to the authentication service when receiving tokens at a web addressable location (using an HTTPS scheme). For a higher level of assurance, we recommend using a certificate (instead of a client secret) as a credential.

Application registration certificates, secrets and federated credentials can be found in the tabs below.

Certificates (0) Client secrets (0) Federated credentials (0)

A secret string that the application uses to prove its identity when requesting a token. Also can be referred to as application password.

+ New client secret

Description	Expires	Value	Secret ID
No client secrets have been created for this application.			

https://portal.azure.com/#view/Microsoft_AAD_RegisteredApps/ApplicationMenuBlade~/~/Credentials/applid/ed300fb9-e461-4733-af3a-ed123187b608/objectid/13a641ee-490a-4002-b5a6-6994e6991048/IsMSAApp-/false/defaultBlade/Overview/appSignInAudience/AzureADMultipl...

Click on New client secret, provide a description, and set an expiration period.

Add a client secret

Description

Secret

Expires

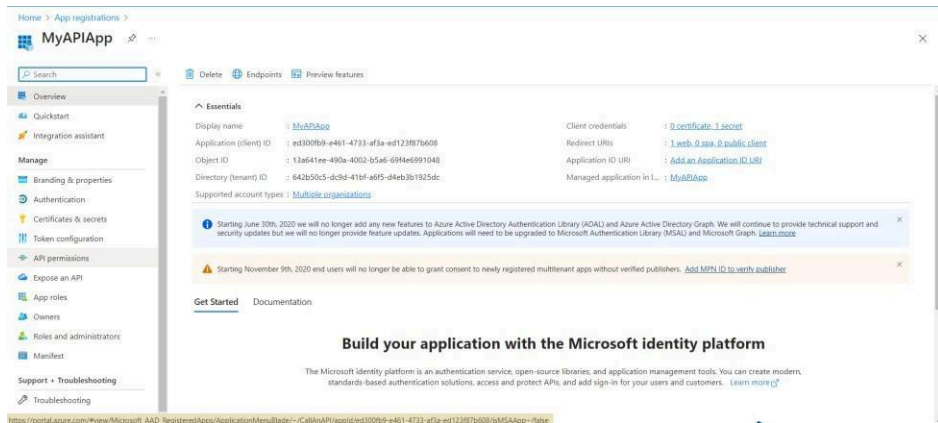
Recommended: 180 days (6 months)

Save the value of the client secret, as it will not be displayed again.

+ New client secret			
Description	Expires	Value ⓘ	Secret ID
Secret	11/29/2024	4468Q~OiznZo4ZmG5S16gQvHmLpLbL...	759166a0-366d-400a-afa2-d5a07e326e59 ⓘ

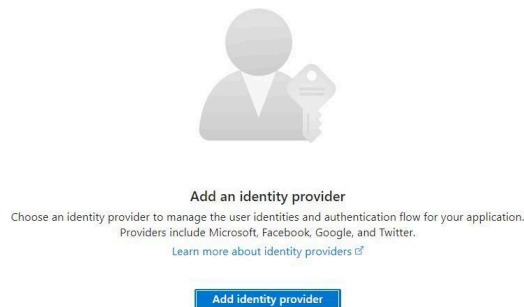
4. Configure Your API App to Use Azure AD Authentication

4.1 Open your API App in the Azure portal.

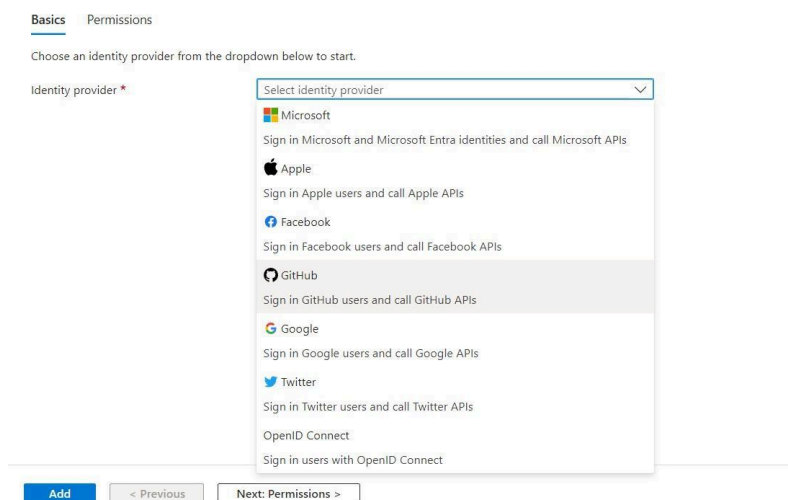


4.2 Navigate to Authentication/Authorization settings:

- Click on Add identity provider.



- Now select GitHub as an Identity Provider.



4.3 Configure Azure Active Directory settings:

- Use the Client ID and Client Secret from the Azure AD app registration.

Add an identity provider ...

Identity provider *

GitHub

App registration

An app registration associates your identity provider with your app. Enter the app registration information here, or go to your provider to create a new one. [Learn more](#)

Client ID *

123

Client secret *

Secret

App Service authentication settings

Requiring authentication ensures that requests to your app include information about the caller, but your app may still need to make additional authorization decisions to control access. If unauthenticated requests are allowed, any client can call the app and your code will need to handle both authentication and authorization. [Learn more](#)

Restrict access *

☒ Require authentication

☐ Allow unauthenticated access

Unauthenticated requests *

☐ HTTP 302 Found redirect: recommended for websites

☐ HTTP 401 Unauthorized: recommended for APIs

☒ HTTP 403 Forbidden

☐ HTTP 404 Not found

Token store

☒

Add

< Previous

Next: Scopes >

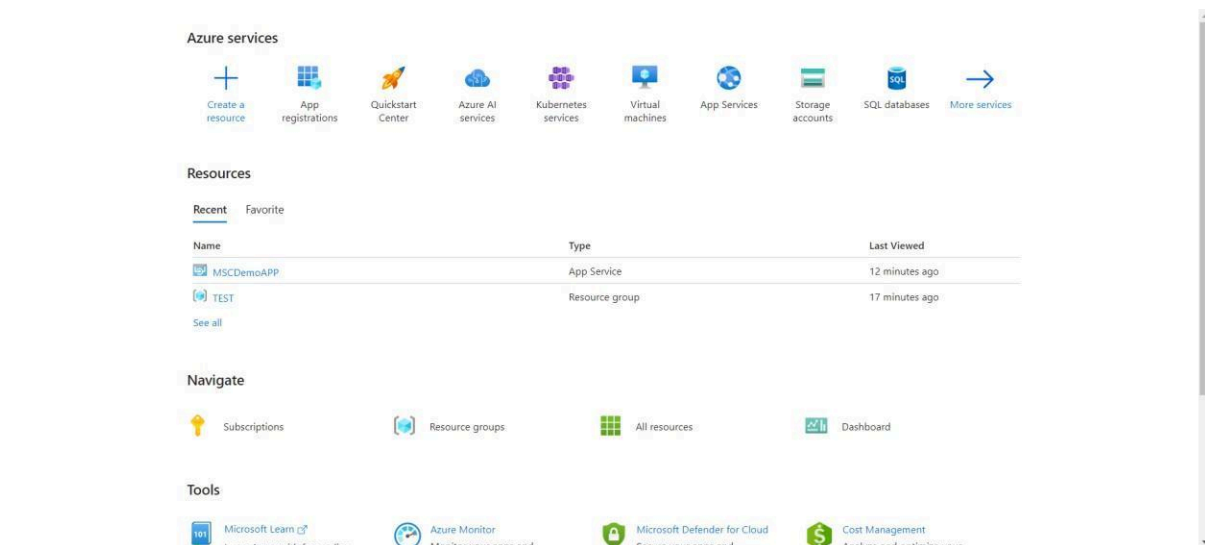
- o After clicking on Add your API App will be secured and API Authentication mechanism will be completed successfully.

Practical 7

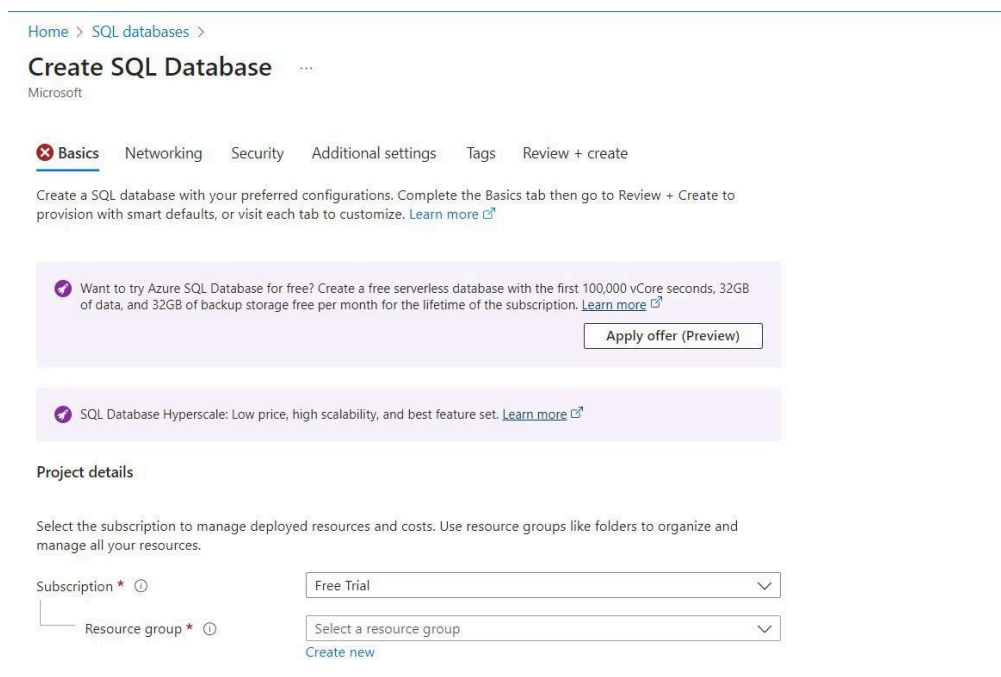
Create a SQL Database and integrate it with your Azure API App

Theory: To create a SQL Database and integrate it with your Azure API App, start by logging into the Azure Portal and navigating to "SQL Databases." Click "Add" to create a new database, providing the necessary details such as database name, server, and pricing tier. Once the database is created, go to your API App in the Azure Portal, and under "Settings," select "Configuration" to add a new connection string. Input the connection details from your SQL Database, including the server name, database name, and authentication credentials. In your API App code, use this connection string to connect to the database, enabling your API endpoints to interact with the SQL Database for data operations. This setup ensures your API can securely access and manage data stored in the SQL Database.

Step 1: Login to your Azure Account in the Azure Portal.



Step 2: Click on SQL Database and then click on Create SQL Database.



Step 3: Fill in all the required details and then click on Review + Create.

Home >

Create SQL Database

Microsoft
manage all your resources.

Subscription *

Resource group *
[Create new](#)

Estimate of resources used / month

COMPUTE COST / VCORE SECOND : 0.000000 INR

1 There will be no charges for usage within the free limits. The database will be paused automatically when the free limits are reached.
2 Serverless databases are billed in vCore seconds based on a combination of CPU and memory utilization. [Learn more about serverless billing](#)

Database details

Enter required settings for this database, including picking a logical server and configuring the compute and storage resources:

Database name *

Server *
[Create new](#)

Compute + storage * **General Purpose - Serverless**
Standard-series (Gen5), 2 vCores, 32 GB storage, zone redundant disabled
[Configure database](#)

Behavior when free offer limit reached

Behavior when free offer limit reached: ☒ Auto-pause the database until next month
When free offer limit is reached, the database will not be accessible until...

[Review + create](#) [Next: Networking >](#)

Step 4: Once your SQL Database is deployed, it will integrate with your API App since you have added the resource group which is linked with your API App.

Home >

MSC-CC-PART1_group

Resource group

Search

+ Create Manage view Delete resource group Refresh Export to CSV Open query Assign tags Move Delete Export template

Overview

- Activity log
- Access control (IAM)
- Tags
- Resource visualizer
- Events
- Settings
 - Deployments
 - Security
 - Deployment stacks
 - Policies
 - Properties
 - Locks
- Cost Management
- Monitoring
- Automation
 - Funnot template

Essentials

Resources Recommendations (1)

Filter for any field... Type equals all Location equals all Add filter

Showing 1 to 8 of 8 records. Show hidden types

Name	Type	Location
cc-db	SQL server	Central India
CC_db (cc-db/CC_db)	SQL database	Central India
Failure Anomalies - MSC-CC-PART1	Smart detector alert rule	Global
MSC	Kubernetes service	East US
MSC-CC-PART1	App Service	East US
MSC-CC-PART1	Application Insights	East US
MSC-CC-PART1-id-beb7	Managed Identity	East US
MSC_CC_1	Kubernetes - Azure Arc	East US

< Previous Page 1 of 1 Next >

[Give feedback](#)

Practical 8

Implement Monitoring and Logging in your Azure API App

Theory: To implement monitoring and logging in your Azure API App, go to the Azure Portal and select your API App. Under the "Monitoring" section, enable "Application Insights" by selecting it and clicking "Enable." Application Insights provides powerful monitoring and diagnostic capabilities, including request rates, response times, and failure rates. Once enabled, configure it by

adding the Application Insights SDK to your API App code if it's not already integrated.

This setup allows you to capture detailed telemetry data.

Additionally, under "Logs," configure "Log Analytics" to centralize logs from your API App. Define diagnostic settings to capture specific logs and metrics, and set up alerts for critical conditions. These tools together offer

comprehensive insights into your API App's performance, usage patterns, and potential issues, facilitating proactive maintenance and troubleshooting.

Step 1: Navigate to your API App in the Azure Portal.

Azure services



Resources

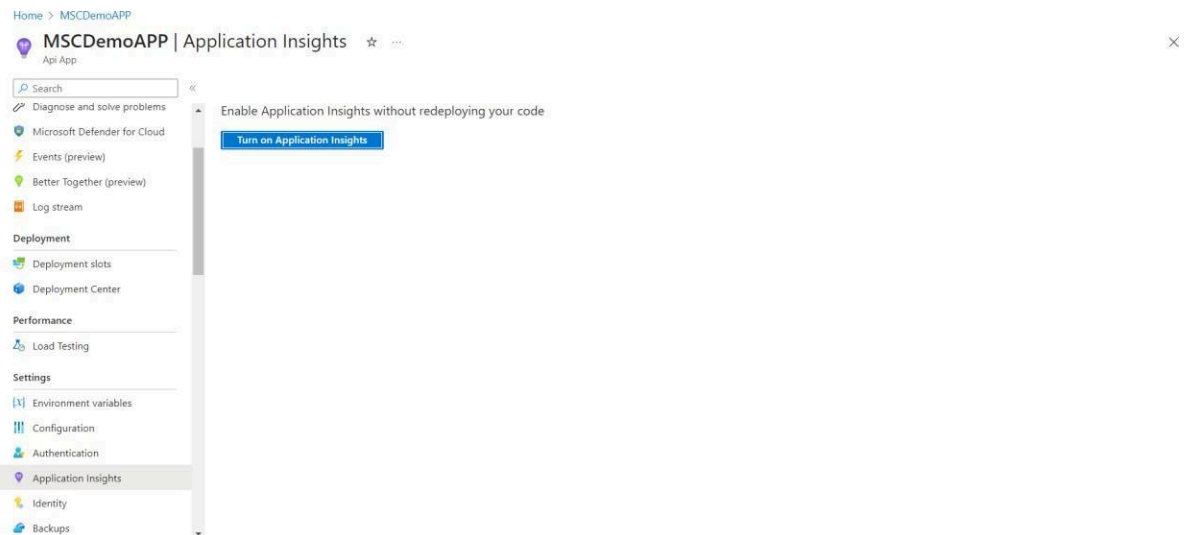
Recent Favorite

Name	Type	Last Viewed
 MSCDemoAPP	App Service	a few seconds ago
 test	SQL database	18 minutes ago
 test-api	SQL server	19 minutes ago
 ASP-TEST-9b13	App Service plan	3 hours ago
 TEST	Resource group	3 hours ago

See all

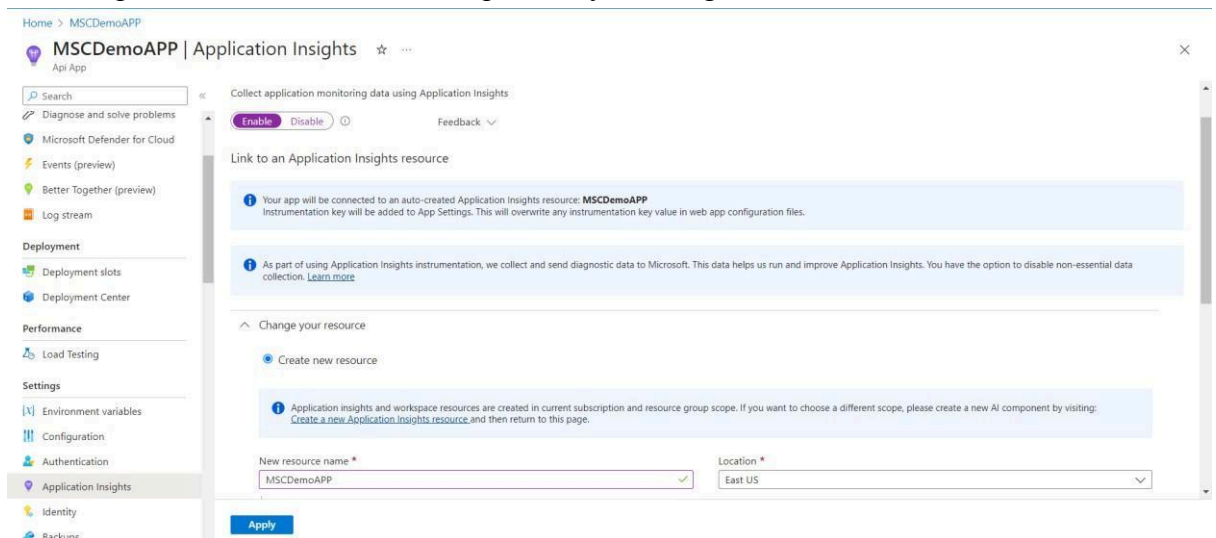
Step 2: Go to Application Insights:

- In the left-hand menu, select Application Insights.
- If Application Insights is not already set up, click Turn on Application Insights and follow the prompts to create a new Application Insights resource or select an existing one.



Step 3: Configure Application Insights:

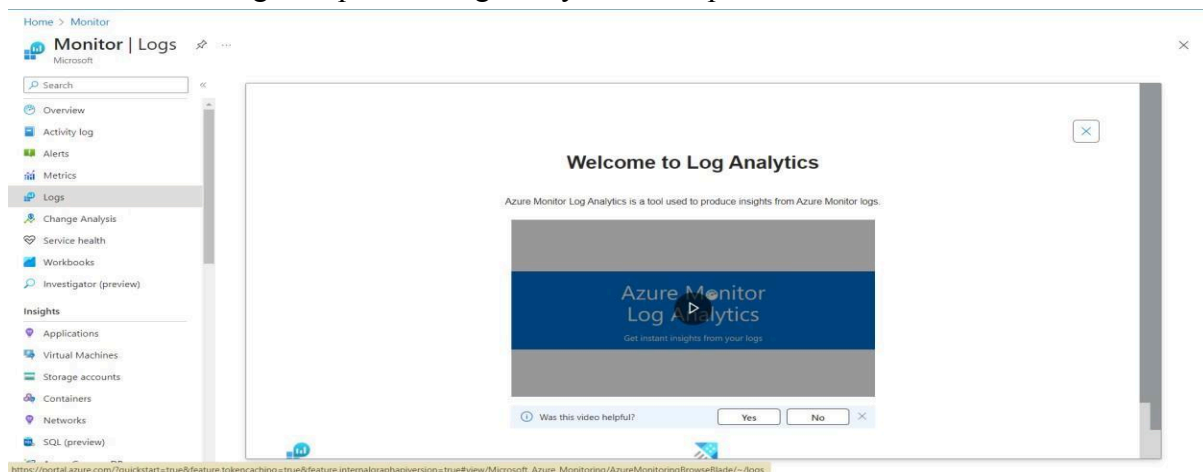
- Once enabled, Application Insights will automatically start collecting data like request rates, response times, failure rates, dependency tracking, and more.



Step 4: Configure Log Analytics

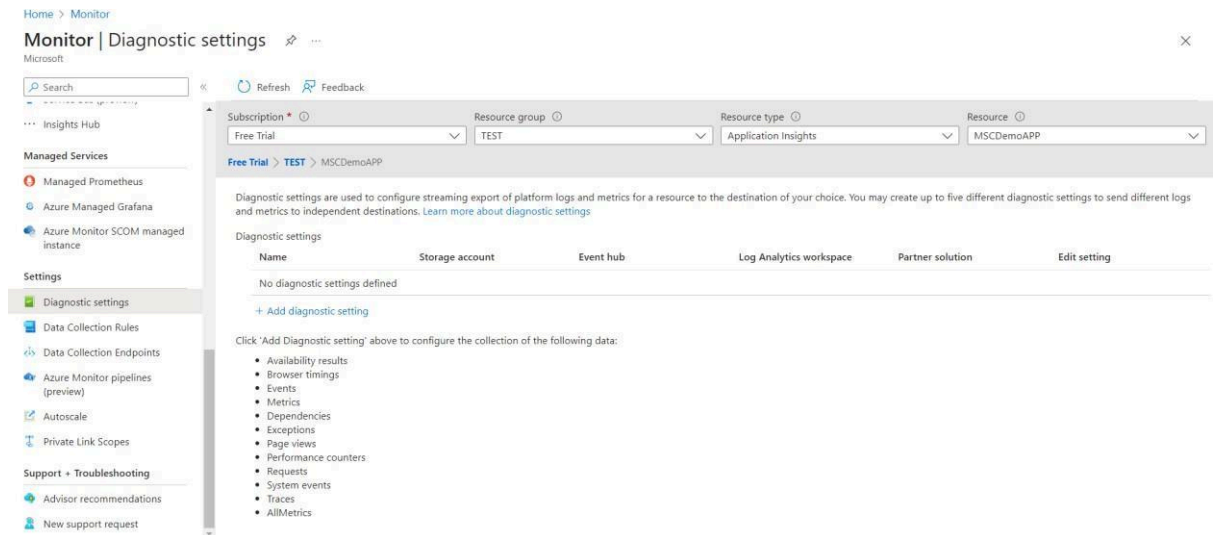
1. Navigate to Azure Monitor:

- In the Azure portal, search for and select Monitor.
- Click on Logs to open the Log Analytics workspace.



Step 5: Link your API App to a Log Analytics workspace:

- If you don't have a workspace, create a new one.
- Navigate back to your API App and go to Diagnostics settings.
- Click Add diagnostic setting, choose the logs and metrics you want to collect, and send them to your Log Analytics workspace.



Step 6: Configure diagnostics logs:

- Select allLogs and any other logs you need.
- Choose the destination (Log Analytics workspace, Storage account, Event Hub).

