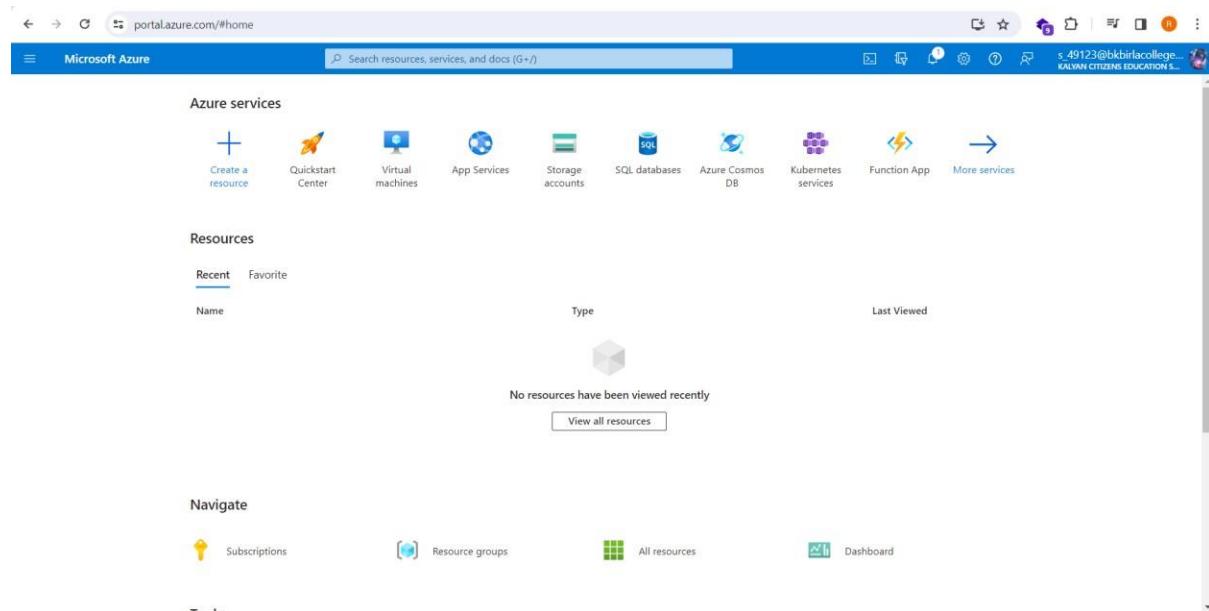


## Practical 1

### 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”.

Get Started

Recently created

Categories

- AI + Machine Learning
- Analytics
- Blockchain
- Compute
- Containers
- Databases
- Developer Tools
- DevOps
- Identity
- Integration
- Internet of Things
- IT & Management Tools
- Media
- Migration

Popular Marketplace products See more in Marketplace

- Windows Server 2019 Datacenter Create | Learn more
- Windows 11 Pro, version 21H2 Create | Learn more
- Ubuntu Server 20.04 LTS Create | Learn more
- Ubuntu Server 22.04 LTS Create | Learn more
- Red Hat Enterprise Linux 7.4 Create | Learn more
- Essentials 50K Set up + subscribe | Learn more
- MongoDB Atlas (pay-as-you-go) Set up + subscribe | Learn more

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

Microsoft Azure

Home > Create a resource > Marketplace >

API App

Microsoft | Azure Service

★ 4.3 (125 ratings)

Plan

API App Create

Create

Overview Plans Usage Information + Support Ratings + Reviews

Create and deploy RESTful APIs in seconds, as powerful as you need them

Leverage your existing tools to create and deploy RESTful APIs without the hassle of managing infrastructure. 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.

- Fastest way to build for the cloud
- Provision and deploy fast
- Simple access control and authentication
- Secure platform that scales automatically
- Great experience for Visual Studio developers with automatic SDK generation
- Open and flexible for everyone
- Monitor, alert, and auto scale (preview)

More products from Microsoft [See All](#)

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

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

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.azurewebsites.net

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.

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

Microsoft Azure

Search resources, services, and docs (G+)

Home > Create a resource > Marketplace > API App > Create API App

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

Compute 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 with the URL <https://portal.azure.com/#view/HubsExtension/DeploymentDetailsBlade/~/overview/id/%2Fsubscriptions%2Fc3b5702f-0d45-4157-9746-27c96ed105e8%2FresourceGroups%2Fpracticalapiapp%2BDeployment%2B35a64da3-8d85>. The page title is "Microsoft.Web-WebApp-Portal-35a64da3-8d85 | Overview". The main content area displays a deployment status message: "Deployment is in progress". Below this, it shows deployment details: Deployment name: Microsoft.Web..., Start time: 3/14/2024, 5:52:5..., Subscription: Azure for Students, Correlation ID: d191123f-0070-0000-0000-000000000000, Resource group: practicalapiapp. It lists two resources: "ASP-practicalapiapp" (Microsoft.Web/server...) and "newWorkspaceTemplate" (Microsoft.Resources/deployments). Both are marked as "OK". A "Deployment details" table shows the same information. On the right side, there are notifications for "Deployment in progress..." and "Microsoft Defender for Cloud". There are also links for "Free Microsoft tutorials", "Work with an expert", and "Give feedback".

Wait till the deployment is completed.

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

The screenshot shows the Microsoft Azure portal with the URL <https://portal.azure.com/#home>. The main content area shows the "Azure services" section with icons for Create a resource, Quickstart Center, Virtual machines, App Services, Storage accounts, SQL databases, Azure Cosmos DB, Kubernetes services, and Function App. Below this is the "Resources" section, which lists recent resources: "practicalapiapp" (App Service) and "practicalapiapp\_group" (Resource group), both last viewed 13 minutes ago. On the right side, there is a "Notifications" panel. It shows a "Pinned to dashboard" message from 10 minutes ago: "Successfully pinned to new dashboard 'My Project'". Below it is a "Deployment succeeded" message from 10 minutes ago: "Deployment 'Microsoft.Web-WebApp-Portal-35a64da3-8d85' to resource group 'practicalapiapp\_group' was successful." It includes buttons for "Go to resource" and "Pin to dashboard".

Now you will see a window like this:

The screenshot shows the Microsoft Azure portal with the URL <https://portal.azure.com/#blade/HubsExtension/DeploymentDetailsBlade/~/overview/id/%2Fsubscriptions%2Fc3b5702f-0d45-4157-9746-27c96ed105e8%2FresourceGroups%2Fpracticalapiapp%2BDeployment%2B35a64da3-8d85>. The page title is "practicalapiapp". The left sidebar shows navigation options: Home, Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Microsoft Defender for Cloud, Events (preview), Log stream, Deployment slots, Deployment Center, Environment variables, Configuration, Authentication, Application Insights, and Identity. The main content area shows the "Essentials" section with details: Resource group (move) : practicalapiapp\_group, Status : Running, Location (move) : East US, Subscription (move) : Azure for Students, Subscription ID : c3b5702f-0d45-4157-9746-27c96ed105e8. It also shows "Properties" (Name: practicalapiapp, Publishing model: Code, Runtime Stack: Dotnet - v8.0), "Deployment Center" (Deployment logs, Last deployment, Deployment provider), "Domains" (Default domain: practicalapiapp.azurewebsites.net, Custom domain: Add custom domain), and "Application Insights" (Name: practicalapiapp, Region: East US). A "JSON View" button is located at the top right.

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 addyour code source via GitHub or from any other platform.

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

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 Publish...  On  Off

FTP Basic Auth Publish...  On  Off

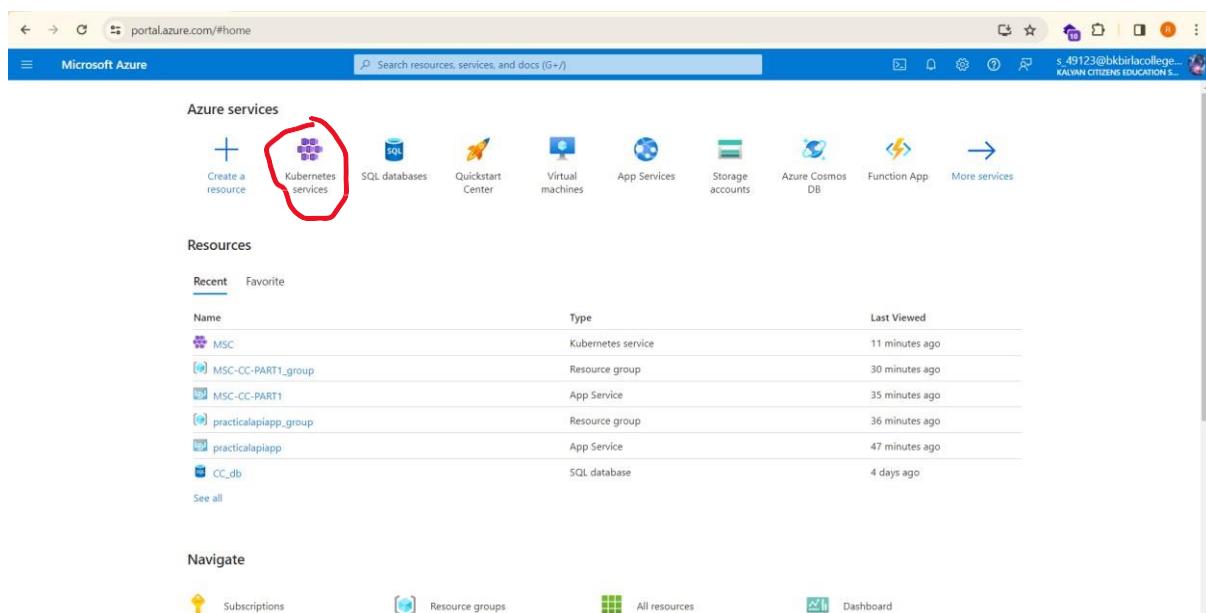
Disable basic authentication for FTP and SCM access. [Learn more](#)

## Practical 2

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

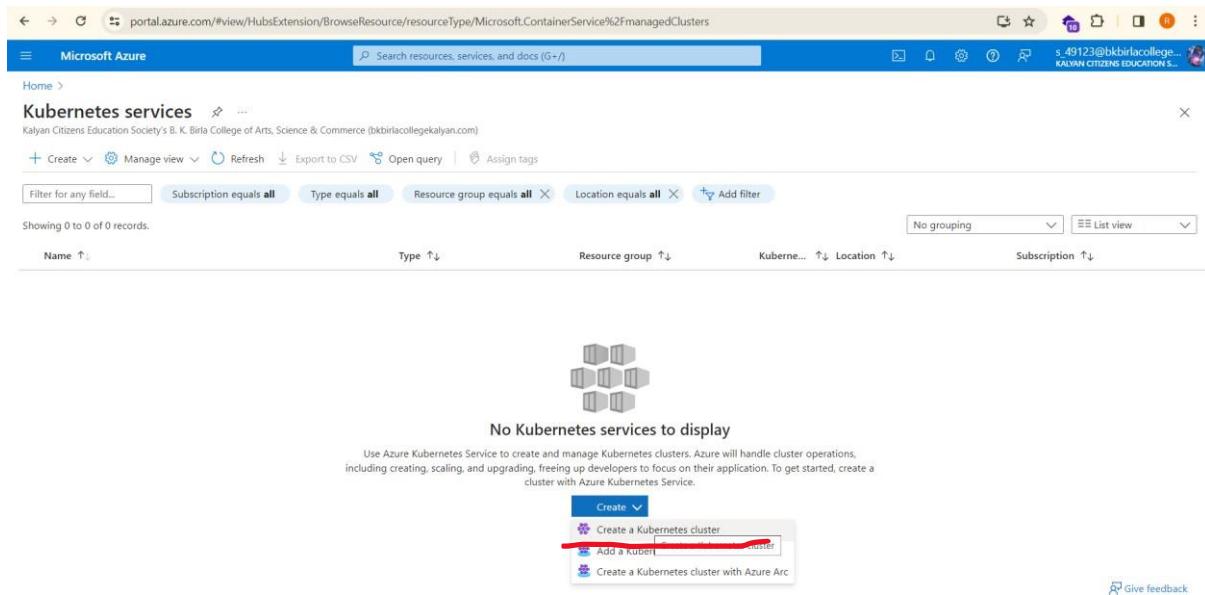
**Theory:** An Azure Kubernetes Cluster (AKS) is a managed container orchestration service provided by Microsoft Azure, allowing users to deploy, manage, and scale containerized applications using Kubernetes. With AKS, users can abstract away the complexities of managing Kubernetes infrastructure and 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



The screenshot shows the Microsoft Azure portal interface. At the top, there's a navigation bar with icons for back, forward, search, and user information. Below the bar, the title 'Microsoft Azure' is visible. The main content area is titled 'Azure services' and features a grid of service icons. One icon, 'Kubernetes services', is circled in red. Other icons include 'Create a resource', 'SQL databases', 'Quickstart Center', 'Virtual machines', 'App Services', 'Storage accounts', 'Azure Cosmos DB', 'Function App', and 'More services'. Below this grid is a section titled 'Resources' with tabs for 'Recent' and 'Favorite'. Under 'Recent', there's a table listing resources: 'MSC' (Kubernetes service, last viewed 11 minutes ago), 'MSC-CC-PART1\_group' (Resource group, 30 minutes ago), 'MSC-CC-PART1' (App Service, 35 minutes ago), 'practicalapiapp\_group' (Resource group, 36 minutes ago), 'practicalapiapp' (App Service, 47 minutes ago), and 'CC\_db' (SQL database, 4 days ago). At the bottom, there's a 'Navigate' section with links for 'Subscriptions', 'Resource groups', 'All resources', and 'Dashboard'.

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



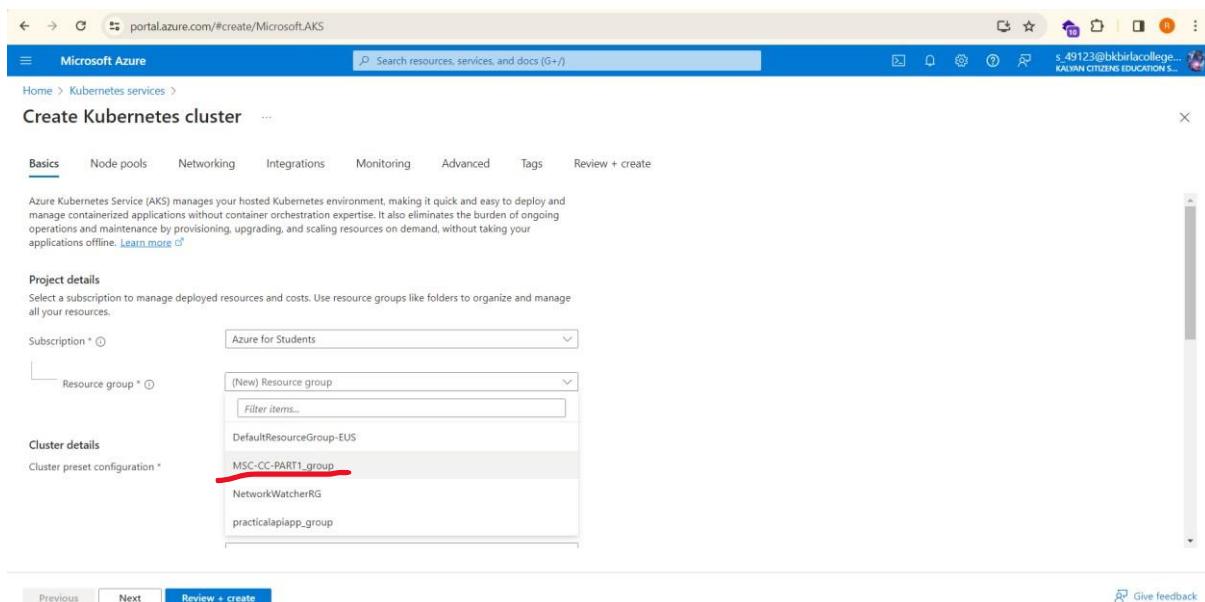
No Kubernetes services to display

Use Azure Kubernetes Service to create and manage Kubernetes clusters. Azure will handle cluster operations, including creating, scaling, and upgrading, freeing up developers to focus on their application. To get started, create a cluster with Azure Kubernetes Service.

Create

- Create a Kubernetes cluster
- Add a Kubernetes cluster
- Create a Kubernetes cluster with Azure Arc

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



Create Kubernetes cluster

Basics Node pools Networking Integrations Monitoring Advanced Tags Review + create

Azure Kubernetes Service (AKS) manages your hosted Kubernetes environment, making it quick and easy to deploy and manage containerized applications without container orchestration expertise. It also eliminates the burden of ongoing operations and maintenance by provisioning, upgrading, and scaling resources on demand, without taking your applications offline. [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 \*

Resource group \*

**Cluster details**  
Cluster preset configuration \*

MSC-CC-PART1\_group (highlighted)

NetworkWatcherRG  
practicalapiapp\_group

Previous Next Review + create

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”

Kubernetes cluster name \*

Region \*

AKS pricing tier

Kubernetes version

Automatic upgrade

Start on

Node security channel type

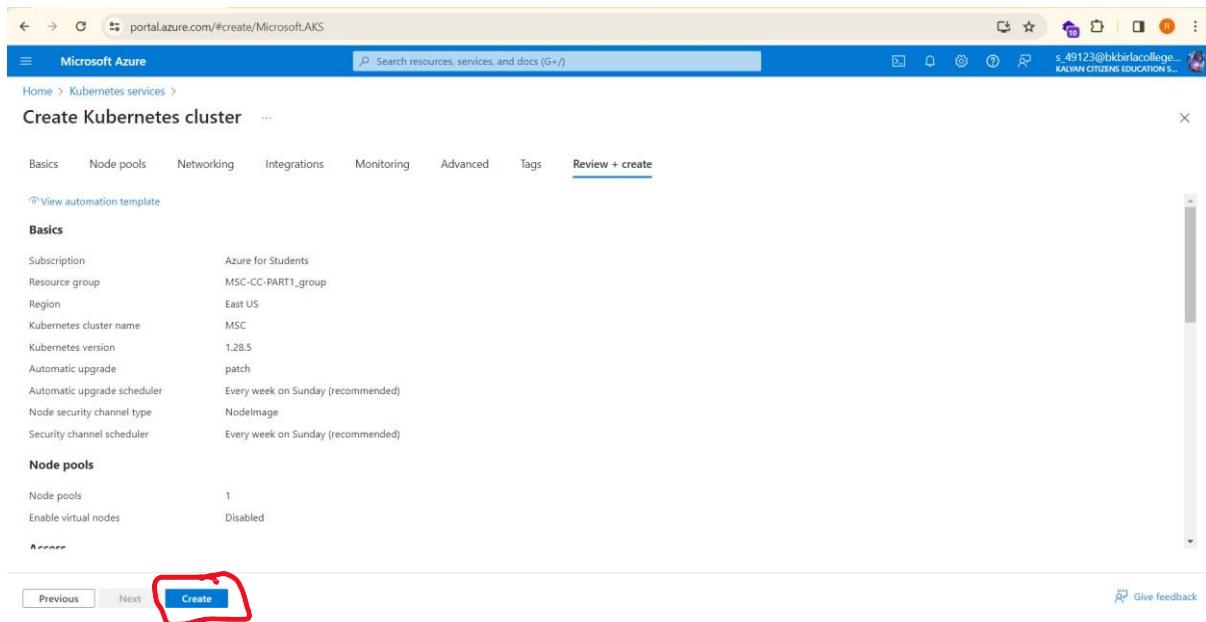
Review + create

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

Validation in progress

Create

Step 5: Now click on Create.



The screenshot shows the 'Create Kubernetes cluster' wizard in the Microsoft Azure portal. The current step is 'Review + create'. The configuration details are as follows:

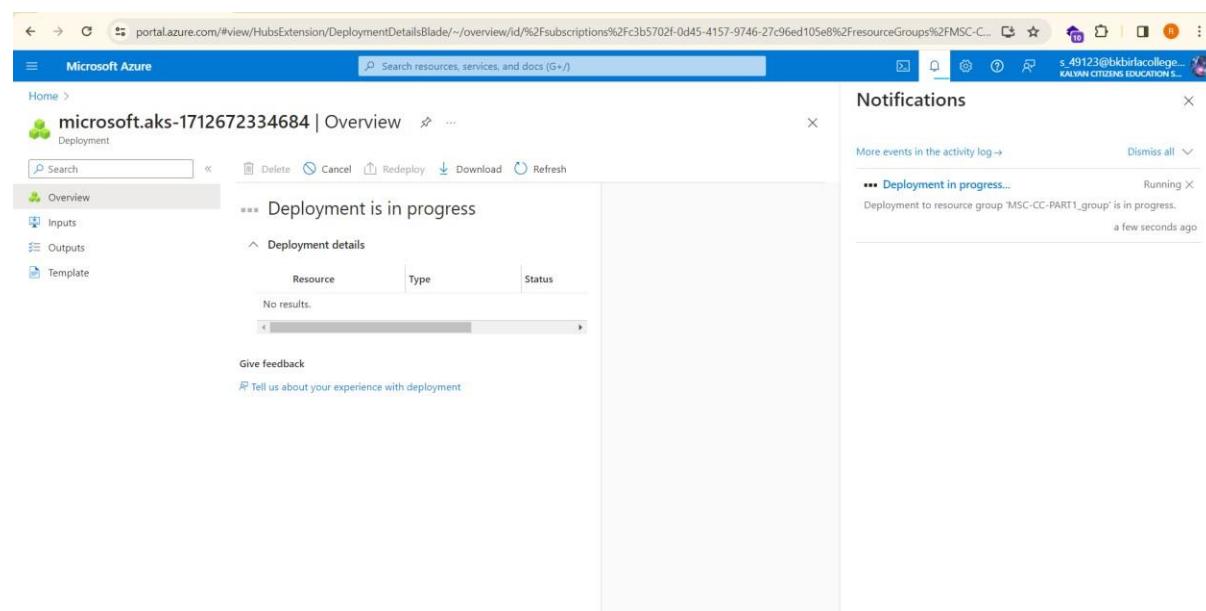
Setting	Value
Subscription	Azure for Students
Resource group	MSC-CC-PART1_group
Region	East US
Kubernetes cluster name	MSC
Kubernetes version	1.28.5
Automatic upgrade	patch
Automatic upgrade scheduler	Every week on Sunday (recommended)
Node security channel type	NodelImage
Security channel scheduler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

**Node pools**

Setting	Value
Node pools	1
Enable virtual nodes	Disabled

At the bottom, there are 'Previous' and 'Next' buttons, and a prominent blue 'Create' button which is circled in red.

Step 6: Now wait till the deployment finishes.



The screenshot shows the 'Deployment Details' blade for a Kubernetes cluster named 'microsoft.aks-1712672334684'. The main area displays the message '\*\*\* Deployment is in progress'. On the right, the 'Notifications' sidebar shows a single entry: 'Deployment in progress...' with a status of 'Running' and a note that it is 'Deployment to resource group 'MSC-CC-PART1\_group'' and was 'a few seconds ago'.

The screenshot shows the Microsoft Azure portal with the URL [https://portal.azure.com/#view/HubsExtension/DeploymentDetailsBlade/~/overview/id/%2Fsubscriptions%2Fc3b5702f-0d45-4157-9746-27c96ed105e8%2FresourceGroups%2FMSCC-Part1\\_group](https://portal.azure.com/#view/HubsExtension/DeploymentDetailsBlade/~/overview/id/%2Fsubscriptions%2Fc3b5702f-0d45-4157-9746-27c96ed105e8%2FresourceGroups%2FMSCC-Part1_group). The main content area displays the 'Overview' tab for a deployment named 'microsoft.aks-1712672334684'. The status is 'Your deployment is complete'. Deployment details include a name, start time (4/9/2024, 7:50:05 PM), subscription (Azure for Students), and resource group (MSCC-Part1\_group). Below this, there are sections for 'Deployment details' and 'Next steps', with a 'Go to resource' button. To the right, a 'Notifications' sidebar shows a single successful deployment message: 'Deployment succeeded' for 'microsoft.aks-1712672334684' to resource group 'MSCC-Part1\_group'. The message was sent 'a few seconds ago'.

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

The screenshot shows the Microsoft Azure portal with the URL [https://portal.azure.com/#@blkbirlacollegekalyan/resource/subscriptions/c3b5702f-0d45-4157-9746-27c96ed105e8/resourcegroups/MSCC-PART1\\_group/providers/Microsoft.ContainerService/managedClusters/MSCC-Part1\\_group](https://portal.azure.com/#@blkbirlacollegekalyan/resource/subscriptions/c3b5702f-0d45-4157-9746-27c96ed105e8/resourcegroups/MSCC-PART1_group/providers/Microsoft.ContainerService/managedClusters/MSCC-Part1_group). The main content area displays the 'Properties' tab for a Kubernetes service named 'MSC'. The 'Essentials' section shows the resource group as 'MSCC-PART1\_group' (highlighted with a red box), status as 'Succeeded (running)', subscription as 'Azure for Students', location as 'East US', and subscription ID as 'c3b5702f-0d45-4157-9746-27c96ed105e8'. Other properties listed include Kubernetes version (1.28.5), API server address (msc-dns-g7rvfkp.hcp.eastus.azmk8s.io), network type (Azure CNI), node pools (1 node pool), and tags (Add tags). The 'Networking' section lists the API server address, network type (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 LTTD (Not enabled).

## 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 KubernetesCluster." 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.

The screenshot shows the Microsoft Azure portal homepage. At the top, there's a search bar and a navigation bar with icons for account management and help. Below the header, the 'Azure services' section features a 'Create a resource' button and links to various services: SQL databases, Kubernetes services, Quickstart Center, Virtual machines, App Services, Storage accounts, Azure Cosmos DB, Function App, and More services. The 'Resources' section displays a list of recent resources, each with a thumbnail icon, name, type, and last viewed time. The resources listed are: MSC-CC-PART1 (Application Insights, 16 hours ago), MSC-CC-PART1 (App Service, 16 hours ago), practicalapiapp (App Service, 17 hours ago), MSC-CC-PART1\_group (Resource group, 17 hours ago), MSC (Kubernetes service, 19 hours ago), practicalapiapp\_group (Resource group, 19 hours ago), and CC\_db (SQL database, 5 days ago). Below this list is a 'See all' link. At the bottom, there's a 'Navigate' section with a link to 'https://portal.azure.com/#create/hub'.

Name	Type	Last Viewed
MSC-CC-PART1	Application Insights	16 hours ago
MSC-CC-PART1	App Service	16 hours ago
practicalapiapp	App Service	17 hours ago
MSC-CC-PART1_group	Resource group	17 hours ago
MSC	Kubernetes service	19 hours ago
practicalapiapp_group	Resource group	19 hours ago
CC_db	SQL database	5 days ago

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

The screenshot shows the Microsoft Azure portal homepage. In the top right corner, there is a search bar with the text "kubernetes azure arc". Below the search bar, there are two tabs: "All" (which is highlighted with a red box) and "Marketplace (1)". The main content area displays a list of services under the heading "Services". The first item in the list is "Kubernetes - Azure Arc", which is also highlighted with a red box. Other items listed include "Azure Cosmos DB", "Azure Database for MySQL servers", "Azure Arc", and "Kubernetes Clusters...". To the right of the search results, there are sections for "Storage accounts", "Azure Cosmos DB", and "More services". Below the search results, there is a "Last Viewed" section showing a list of resources last viewed between 5 days ago and 19 hours ago.

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

The screenshot shows the "Azure Arc | Kubernetes clusters" blade in the Microsoft Azure portal. On the left, there is a navigation menu with sections like "Overview", "All Azure Arc resources", "Management", "Infrastructure", and "Kubernetes clusters" (which is currently selected). The main content area has a search bar at the top. Below the search bar, there is a button labeled "Add a Kubernetes cluster with Azure Arc" with a red box around it. The blade also includes filter options for "Name", "Type", "Resource group", "Location", and "Subscription". At the bottom, there is a message stating "No Azure Arc kubernetes clusters to display" and a link to "Learn more".

## Step 4: Now click on “next.”

The screenshot shows the Microsoft Azure portal with the URL [portal.azure.com/#view/Microsoft\\_Azure\\_HybridCompute/ConnectedClusterCreate.ReactView](https://portal.azure.com/#view/Microsoft_Azure_HybridCompute/ConnectedClusterCreate.ReactView). The page title is "Add a Kubernetes cluster with Azure Arc". The top navigation bar includes "Microsoft Azure", a search bar, and user information "s.49123@bkbirlacollege... KALYAN CITIZENS EDUCATION S...". The main content area has tabs: "Prerequisites" (selected), "Cluster details", "Tags", "Run script", and "Verification". A large icon of a server with a plus sign is displayed. Below it, a section titled "Before you add your cluster to Azure Arc, you'll need" lists requirements:

- A new or existing Kubernetes cluster: The cluster must use Kubernetes version 1.13 or later (including OpenShift 4.2 or later and other Kubernetes derivatives). [Learn how to create a Kubernetes cluster](#).
- Access to ports 443 and 9418: Make sure the cluster has access to these ports, and the required outbound URLs.

At the bottom left, there are "Previous" and "Next" buttons. The "Next" button is highlighted with a red box.

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

The screenshot shows the Microsoft Azure portal with the same URL and title as the previous step. The "Cluster details" tab is selected. The "Project details" section contains a note about Azure Arc allowing connection to Kubernetes clusters. The "Azure Arc cluster details" section includes fields for "Cluster name" (set to "MSC-CC-PART1\_group") and "Region" (set to "NetworkWatcherRG"). The "Resource group" dropdown is open, showing options like "DefaultResourceGroup-EUS", "MC\_MSC-CC-PART1\_group\_MSC\_eastus", and "MSC-CC-PART1\_group". The "MSC-CC-PART1\_group" option is highlighted with a red box. At the bottom left, there are "Previous" and "Next" buttons. The "Next" button is highlighted with a red box.

For e.g. here we have selected MSC-CC-PART1\_group

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

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 \*

**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.

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	<input type="text"/>	Kubernetes - Azure Arc
City	<input type="text"/>	Kubernetes - Azure Arc
StateOrDistrict	<input type="text"/>	Kubernetes - Azure Arc
CountryOrRegion	<input type="text"/>	Kubernetes - Azure Arc
	<input type="text"/>	Kubernetes - Azure Arc

**Custom tags**

[Previous](#) [Next](#) [Give feedback](#)

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

The screenshot shows the Azure portal interface for creating a Connected Cluster. The top navigation bar includes 'Microsoft Azure', a search bar, and user information. Below it, the breadcrumb navigation shows 'Home > Azure Arc | Kubernetes clusters > Add a Kubernetes cluster with Azure Arc ...'. The main content area has tabs for 'Prerequisites' (green checkmark), 'Cluster details' (green checkmark), 'Tags' (green checkmark), 'Run script' (blue circle), and 'Verification' (grey). The 'Run script' tab is selected. A sub-section titled '1. Download or copy the following script' is displayed, with 'Script type \*' set to 'Bash' (radio button selected). The script content is a multi-line command to create an Azure Arc resource:

```
1 # This script creates an Azure Arc resource to connect a Kubernetes cluster to
2 # Documentation: https://aka.ms/AzureArcK8sDocs
3
4 # Log into Azure
5 az login --use-device-code
6
7 # Set Azure subscription
8 az account set --subscription "e3b5702f-0d45-4157-9746-27c96ed105e8"
9
10 # Create connected cluster
11 az connectedk8s connect --name "MSC_CC_1" --resource-group "MSC-CC-PART1_group"
--location "eastus" --correlation-id "c18a09d0-685e-48e7-ab55-12588447b0ed"
--tags "datacenter City StateOrDistrict CountryOrRegion"
```

At the bottom of the script area are 'Previous' and 'Next' buttons, and a 'Give feedback' link.

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

The screenshot shows the same Azure portal interface as the previous step, but the script content is now partially obscured by a red redaction mark. The 'Run script' tab is still selected. The 'Download.sh' button is visible at the bottom left of the script area. At the bottom of the page, there is a section titled '2. Open the Azure CLI and run the script' with instructions and a numbered list:

Run the above script on the machine you set up with the prerequisites. Make sure the machine has network connectivity to Azure and to your Kubernetes cluster.

The script

1. Checks connectivity from your cluster to Azure Arc via KUBECONFIG, ~./kube/config, or --kube-config
2. Deploys Azure Arc agents into the azure-arc namespace via Helm

At the bottom of the page are 'Previous' and 'Next' buttons, and a 'Give feedback' link.

And now click on Next

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

The screenshot shows the Microsoft Azure portal interface. At the top, the URL is [portal.azure.com/#view/Microsoft\\_Azure\\_HybridCompute/ConnectedClusterCreate.ReactView](https://portal.azure.com/#view/Microsoft_Azure_HybridCompute/ConnectedClusterCreate.ReactView). The page title is "Add a Kubernetes cluster with Azure Arc". Below the title, there are tabs: "Prerequisites" (highlighted with a red underline), "Cluster details", "Tags", "Run script", and "Verification". A prominent green checkmark icon indicates that "MSC\_CC\_1 was successfully connected to Azure". Below this, there is a button labeled "Go to the cluster". Under the heading "Next steps", there are three sections: "Integrate automatic deployments within your cluster", "Azure Policy", and "Monitor your cluster". Each section has a brief description and a "Learn how to set up" link. At the bottom of the page are "Previous" and "Close" buttons, and a "Give feedback" link.

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

The screenshot shows the Microsoft Azure portal interface, specifically the "Overview" page for a Kubernetes cluster named "MSC\_CC\_1". The left sidebar includes navigation links for Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Namespaces, Workloads, Services and ingresses, Storage, Configuration, Extensions, Open Service Mesh, GitOps, Networking (preview), and Policies. The main content area displays the cluster's configuration under the "Essentials" tab. Key details include:

Setting	Value
Subscription	Azure for Students
Subscription ID	c3b5702f-0d45-4157-9746-27c96ed105e8
Resource group	MSC-CC-PART1_group
Status	Connected
Location	East US
Last connectivity time	2024-04-10T12:48:30.806Z
Distribution	aks
Infrastructure	azure
Agent version	1.15.3
Kubernetes version	1.28.5
Provisioning state	Succeeded
Private link scope	N/A

Below the essentials, there are tabs for "Tags (edit)", "Properties", "Monitoring", "Capabilities", and "Tutorials". The "Properties" tab is selected, showing the "Kubernetes Azure Arc" section with the following data:

Setting	Value
Agent version	1.15.3
Managed identity certificate expiration time	July 9, 2024 at 06:00 PM GMT+5:30
Kubernetes version	1.28.5
Total node count	2
Total core count	4

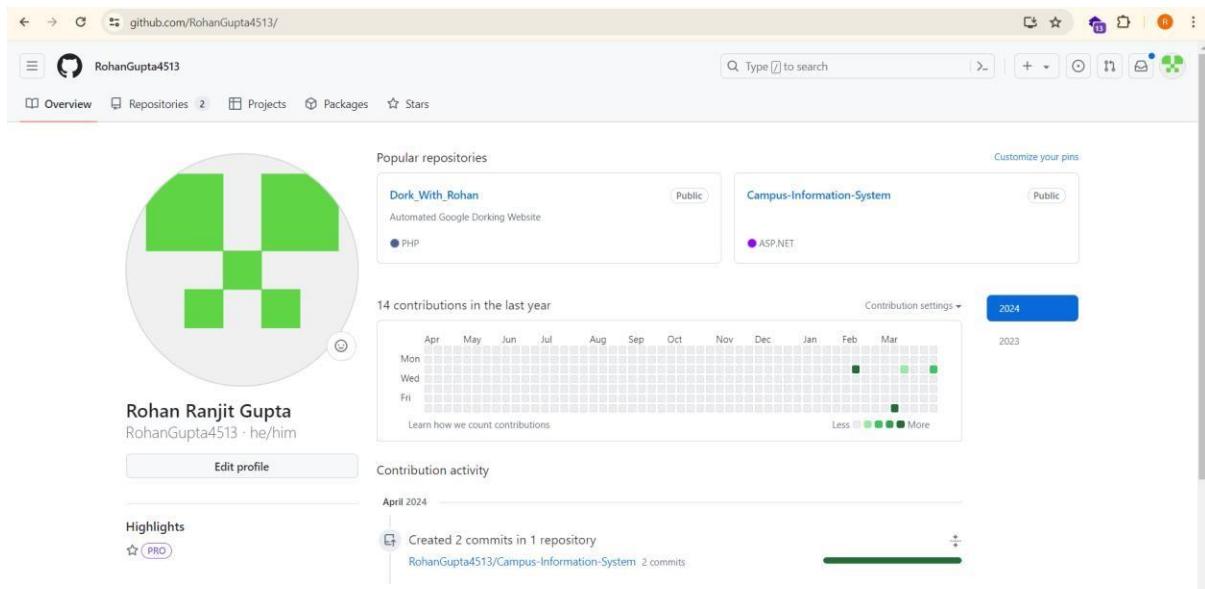
On the right side, there is an "Extensions" section which states "No extensions installed".

## 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.

The screenshot shows the GitHub 'Create a new repository' interface. At the top, it says 'Create a new repository'. Below that, there's a note: 'A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.' A note below says 'Required fields are marked with an asterisk (\*).'. The 'Owner' field is set to 'RohanGupta4513'. The 'Repository name' field contains 'Document management S'. A note next to it says 'Your new repository will be created as Document-management-System-in-Asp.Net.' Below that, it says 'The repository name can only contain ASCII letters, digits, and the characters ., -, and \_.' A note below says 'Great repository names are short and memorable. Need inspiration? How about automatic-doodle ?'. The 'Description (optional)' field is empty. Under 'Visibility', the 'Public' option is selected, with the note 'Anyone on the internet can see this repository. You choose who can commit.'. The 'Private' option is also available. Under 'Initialize this repository with:', the 'Add a README file' checkbox is checked. A note below says 'This is where you can write a long description for your project. Learn more about READMEs.'

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”

The screenshot shows the same GitHub 'Create a new repository' interface as before. The 'Visibility' section is identical. Under 'Initialize this repository with:', the 'Add a README file' checkbox is checked. In the 'Add .gitignore' section, a dropdown menu is open, showing a list of templates: 'None', 'Typo3', 'Unity', 'UnrealEngine', 'VVV', 'VisualStudio' (which is highlighted), 'Waf', 'WordPress', 'Xojo', 'Yeoman', and 'Yii'. A note next to the dropdown says '.gitignore template: None'. A note below the dropdown says 'with your code. Learn more about ignoring files.' Another note below says 'Change the default name in your settings.' A note at the bottom right says 'Create repository'.

And then click on “Create repository”

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

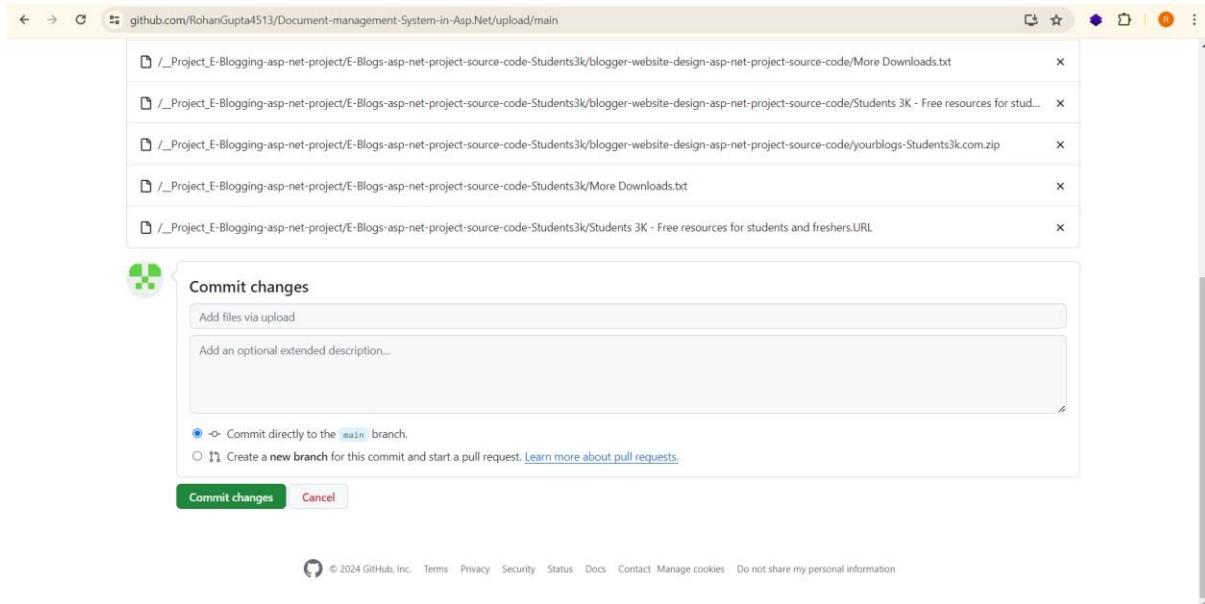
The screenshot shows a GitHub repository page for 'Document-management-System-in-Asp.Net'. The repository was created by 'RohanGupta4513' and has one branch ('main') and one commit ('Initial commit'). The commit was made by 'RohanGupta4513' and is titled 'Initial commit'. The commit message is 'Initial commit'. The commit was made 1 minute ago. The commit hash is 'c32d7c'. A dropdown menu is open at the top right, showing options to 'Create new file' or 'Upload files'. To the right of the commit list, there is an 'About' section with a note: 'No description, website, or topics provided.' It also lists 'Readme', 'Activity', '0 stars', '1 watching', and '0 forks'. Below the commit list, there is a 'README' section with the text 'Document-management-System-in-Asp.Net'. On the right side of the page, there are sections for 'Releases' (no releases published) and 'Packages' (no packages published). At the bottom left, there is a link to the repository's URL: 'https://github.com/RohanGupta4513/Document-management-System-in-Asp.Net/main'. The bottom right corner shows the system tray with icons for battery, signal, and date/time.

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

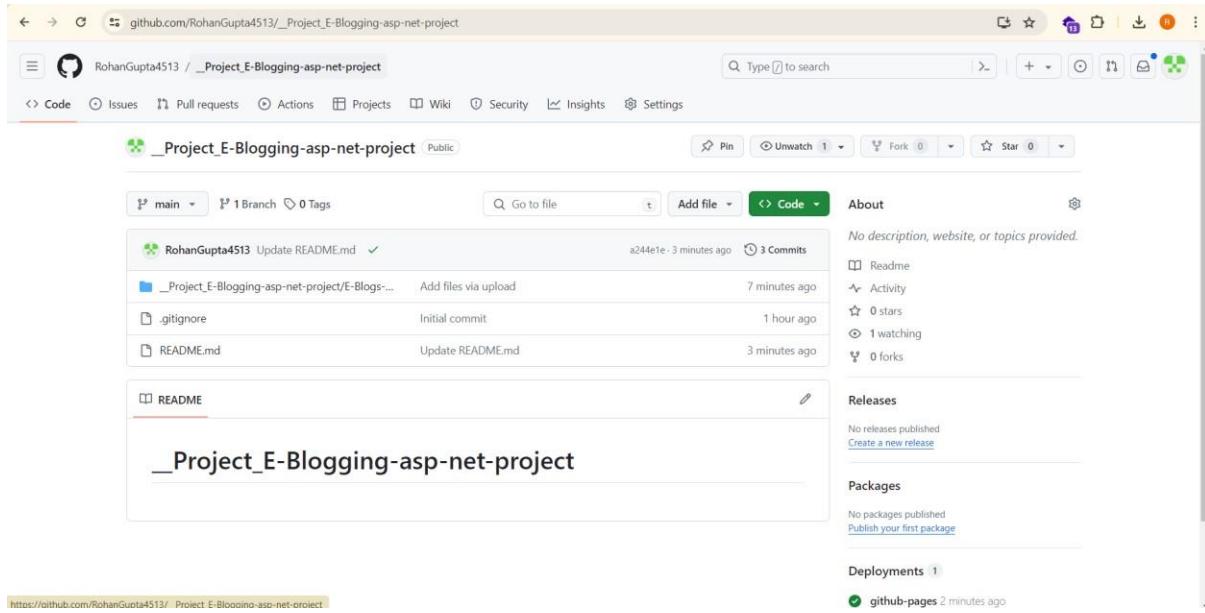
The screenshot shows the 'Upload files' interface for the 'Document-management-System-in-Asp.Net' repository. The URL in the address bar is 'https://github.com/RohanGupta4513/Document-management-System-in-Asp.Net/upload/main'. The interface has a large central area with a placeholder text 'Drag additional files here to add them to your repository' and a smaller text 'Or choose your files'. Below this area, a list of files is shown, each with a delete icon (an 'X'): '\_Project\_E-Blogging-asp-net-project/E-Blogs-asp-net-project-source-code-Students3k/blogger-website-design-asp-net-project-source-code/Blogger-web-application-asp.net.pr...', '\_Project\_E-Blogging-asp-net-project/E-Blogs-asp-net-project-source-code-Students3k/blogger-website-design-asp-net-project-source-code/More Downloads.txt', '\_Project\_E-Blogging-asp-net-project/E-Blogs-asp-net-project-source-code-Students3k/blogger-website-design-asp-net-project-source-code/Students 3K - Free resources for stud...', '\_Project\_E-Blogging-asp-net-project/E-Blogs-asp-net-project-source-code-Students3k/blogger-website-design-asp-net-project-source-code/yourblogs-Students3k.com.zip', and '/ Project E-Blogging-asp-net-project/E-Blogs-asp-net-project-source-code-Students3k/More Downloads.txt'. At the bottom of the screen, the taskbar shows various pinned icons and the system tray with icons for battery, signal, and date/time.

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.



## Practical 5

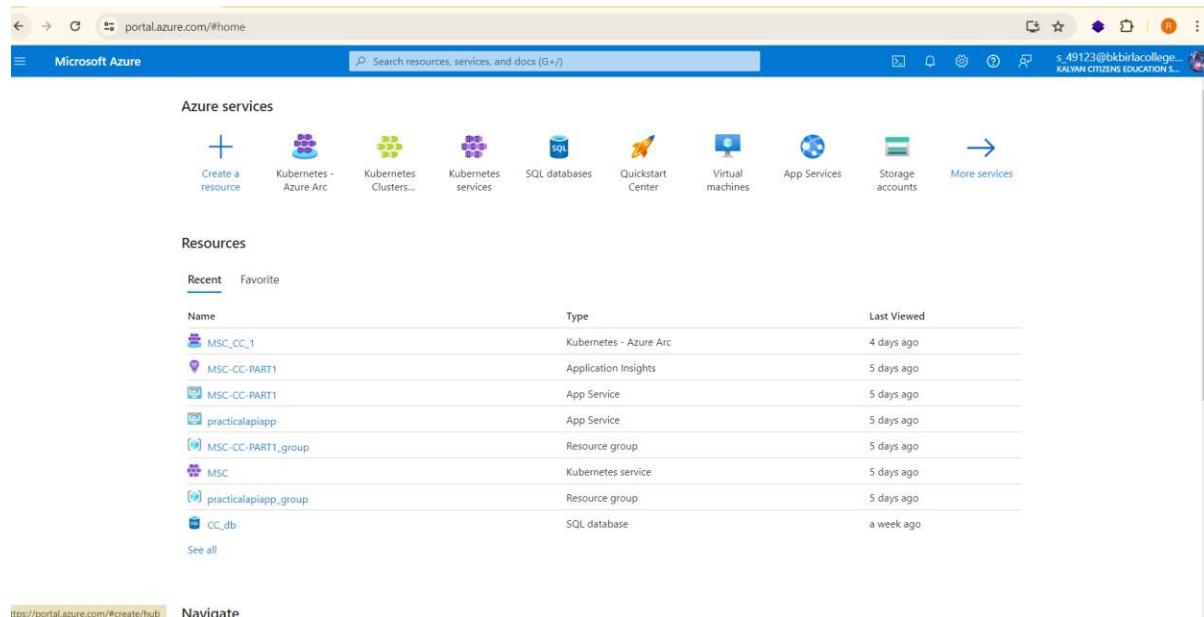
### Deploy the GitHub Repository in your Azure API App.

**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.



The screenshot shows the Microsoft Azure portal interface. At the top, there's a navigation bar with icons for back, forward, refresh, and search, followed by the URL 'portal.azure.com/#home'. Below the search bar is a user profile icon with the email 's.49123@bkbirlacollege... KALYAN CITIZENS EDUCATION S...' and a gear icon. The main content area is titled 'Azure services' and features a grid of icons for various services: 'Create a resource', 'Kubernetes - Azure Arc', 'Kubernetes Clusters...', 'Kubernetes Services', 'SQL databases', 'Quickstart Center', 'Virtual machines', 'App Services', 'Storage accounts', and a 'More services' button. Below this is a section titled 'Resources' with tabs for 'Recent' (selected) and 'Favorite'. It lists recent resources with columns for 'Name', 'Type', and 'Last Viewed'. The listed resources include: 'MSC\_CC\_1' (Kubernetes - Azure Arc, 4 days ago), 'MSC-CC-PART1' (Application Insights, 5 days ago), 'MSC-CC-PART1' (App Service, 5 days ago), 'practicalapiapp' (App Service, 5 days ago), 'MSC-CC-PART1\_group' (Resource group, 5 days ago), 'MSC' (Kubernetes service, 5 days ago), 'practicalapiapp\_group' (Resource group, 5 days ago), and 'CC\_db' (SQL database, a week ago). At the bottom left, there's a link to 'https://portal.azure.com/#create/hub' and a 'Navigate' button.

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

The screenshot shows the Microsoft Azure portal homepage. In the center, there is a detailed view of an API App named "MSC-CC-PART1". The "Resource details" pane on the right provides information about the app, including its type (Kubernetes - Azure Arc), kind (api), location (East US), and status (Running). The "Recent" section of the left sidebar lists several other resources, including "MSC\_CC\_1", "practicalapiapp", "MSC-CC-PART1", and "practicalapiapp\_g".

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”.

The screenshot shows the "Deployment Center" page for the "MSC-CC-PART1" API App. The left sidebar has a "Deployment" section with a "Deployment slots" item highlighted. The main content area displays deployment-related information, including a message about being in the production slot and a "Source\*" dropdown menu. The top navigation bar shows the URL as "portal.azure.com/#@bbkirlacollegekalyan.com/resource/subscriptions/c3b5702f-0d45-4157-9746-27c96ed105e8/resourceGroups/MSC-CC-PART1\_group/providers/Microsoft.Web/sites/MSC-CC-PART1".

## Step 4: Now select the Source as GitHub.

The screenshot shows the Microsoft Azure portal interface for the 'MSC-CC-PART1' deployment center. On the left, there's a navigation sidebar with various options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Microsoft Defender for Cloud, Events (preview), Log stream, Deployment slots, and Deployment Center (which is selected). The main content area has tabs for Settings, Logs, and FTPS credentials. A modal window titled 'Select code source' is open, showing options: Continuous Deployment (CI/CD) (selected), GitHub (highlighted with a red box), Bitbucket, Local Git, Azure Repos, Manual Deployment (Push), and External Git. A message at the top of the modal says, 'You're now in the production slot, which is not recommended for setting up CI/CD. Learn more'.

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

The screenshot shows the Microsoft Azure portal interface for the 'MSC-CC-PART1' deployment center. The left sidebar includes Deployment slots, Deployment Center (selected), Performance, Load Testing (Preview), Settings (selected), Environment variables, Configuration, and Authentication. The main content area displays configuration for GitHub integration. It shows 'Signed in as' as 'RohanGupta4513' with a 'Change Account' link. Under 'Organization\*', 'Repository\*', and 'Branch\*', dropdown menus are shown. In the 'Build' section, 'Runtime stack' is set to '.NET' and 'Version' to 'v8.0'. The 'Authentication settings' section contains a note about federating GitHub identities with Azure. The URL in the browser bar is 'portal.azure.com/#@bkbiracollegekalyan.com/resource/subscriptions/c3b5702f-0d45-4157-9746-27c96ed105e8/resourceGroups/MSC-CC-PART1\_group/providers/Microsoft.Web/sites/MS...'. The user 's.49123@bkbiracollege...' is logged in.

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

MSC-CC-PART1 | Deployment Center

Signed in as RohanGupta4513 Change Account

Organization\* RohanGupta4513

Repository\* Project\_E-Blogging-asp-net-project

Branch\* **Project\_E-Blogging-asp-net-project**

Build

Runtime stack .NET

Version v8.0

Authentication settings

Select how you want your GitHub Action workflow to authenticate to Azure. If you choose user-assigned identity, the identity selected will be federated with GitHub as an authorized client and given write permissions on the app. [Learn more](#)

Here we have selected the repository named “Project\_E-Blogging- asp-net-project”.

Project\_E-Blogging- asp-net-project

## Step 7: Now select your repositories branch.

MSC-CC-PART1 | Deployment Center

Signed in as RohanGupta4513 Change Account

Organization\* RohanGupta4513

Repository\* Project\_E-Blogging-asp-net-project

Branch\* **main**

Build

Runtime stack .NET

Version v8.0

Authentication settings

Select how you want your GitHub Action workflow to authenticate to Azure. If you choose user-assigned identity, the identity selected will be federated with GitHub as an authorized client and given write permissions on the app. [Learn more](#)

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.

The screenshot shows the 'Deployment Center' page for the 'MSC-CC-PART1' project. In the 'Authentication settings' section, the 'Identity' dropdown is open, displaying a list of identities. The identity 'MSC-CC-PART1-id-beb7' is highlighted and selected. Other identities listed are 'practicalapiapp\_group' and several numerical IDs: 'practicalapiapp-id-a15a', 'practicalapiapp-id-a1af', 'practicalapiapp-id-b9b0', and 'practicalapiapp-id-bb19'. The 'Build' section shows the runtime stack as '.NET' and the version as 'v8.0'.

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

Step 9: Now click on save.

The screenshot shows the 'Deployment Center' page for the 'MSC-CC-PART1' project. The 'Save' button at the top left of the form is highlighted with a red box. The rest of the page displays the same configuration as the previous screenshot, including the selected identity 'MSC-CC-PART1-id-beb7' in the 'Identity' dropdown.

Now you can see that your project is successfully deployed and connected in your Azure API App.

Microsoft Azure

MSC-CC-PART1 | Deployment Center

Api App

Search resources, services, and docs [G+]

Home > MSC-CC-PART1

Overview Activity log Access control (IAM) Tags Diagnose and solve problems Microsoft Defender for Cloud Events (preview) Log stream Deployment Deployment slots Deployment Center

Deployment slots Deployment Center

Load Testing (Preview)

Environment variables Configuration

Save Discard Browse Manage publish profile Sync Leave Feedback

Settings Logs FTPS credentials

Deploy and build code from your preferred source and build provider. [Learn more](#)

Source GitHub [Disconnect](#)

Signed in as RohanGupta4513

Organization RohanGupta4513

Repository \_Project\_E-Blogging-asp.net-project

Branch main

Build

Build provider GitHub Actions

Runtime stack .NET

Version v8.0

The screenshot shows the Microsoft Azure Deployment Center settings for the 'MSC-CC-PART1' application. The left sidebar lists various deployment-related sections like Overview, Activity log, and Environment variables. The 'Deployment Center' section is currently selected. The main pane displays the GitHub source configuration, which includes the repository URL '\_Project\_E-Blogging-asp.net-project', which is underlined in red.

## 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.

The screenshot shows the Microsoft Azure portal interface. At the top, there's a navigation bar with links for 'Create a resource', 'Quickstart Center', 'Azure AI services', 'Kubernetes services', 'Virtual machines', 'App Services', 'Storage accounts', 'SQL databases', 'Azure Cosmos DB', and 'More services'. Below the navigation bar is a 'Resources' section with tabs for 'Recent' and 'Favorite'. It displays a message 'No resources have been viewed recently' with a 'View all resources' button. In the 'Navigate' section, there are links for 'Subscriptions', 'Resource groups', 'All resources', and 'Dashboard'. The 'Tools' section includes links for 'Microsoft Learn', 'Azure Monitor', 'Microsoft Defender for Cloud', and 'Cost Management'. The URL in the address bar is 'https://portal.azure.com/#home'.

Step 2:

1. Register a new application:
  - Go to App registrations and click on New registration.

The screenshot shows the 'App registrations' page in the Azure portal. The title bar says 'Home &gt; App registrations'. Below the title, there are buttons for '+ New registration', 'Endpoints', 'Troubleshooting', 'Refresh', 'Download', 'Preview features', and 'Got feedback?'. A note at the top states: 'Starting June 30th, 2020 we will no longer add any new features to Azure Active Directory Authentication Library (ADAL) and Azure Active Directory Graph. We will continue to provide technical support and security updates but we will no longer provide feature updates. Applications will need to be upgraded to Microsoft Authentication Library (MSAL) and Microsoft Graph.' Below this, there are tabs for 'All applications', 'Owned applications' (which is selected), and 'Deleted applications'. A search bar at the bottom allows filtering by 'Start typing a display name or application (client) ID to filter these results...' and a 'Add filters' button. A message at the bottom says 'This account isn't listed as an owner of any applications in this directory.' and a 'View all applications in the directory' button.

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

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 e.g. https://example.com/auth  
By proceeding, you agree to the Microsoft Platform Policies [Learn more](#)

**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:

- After registration, navigate to API permissions.

MyAPIApp | API permissions

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.

API / Permissions name	Type	Description	Admin consent req...	Status
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](#).

[https://portal.azure.com/#view/Microsoft\\_AAD\\_RegisteredApps/ApplicationMenuBlade~/CallAnAPI/appid/a18f7437-8226-4d2a-9f2c-6c8242e3ae7d/objectId/f207adc4-f113-48c4-b021-855451da1f7d/nMSAApp-/false/defaultBlade/Overview/appSignInAudience/AzureADMyOrg...](https://portal.azure.com/#view/Microsoft_AAD_RegisteredApps/ApplicationMenuBlade~/CallAnAPI/appid/a18f7437-8226-4d2a-9f2c-6c8242e3ae7d/objectId/f207adc4-f113-48c4-b021-855451da1f7d/nMSAApp-/false/defaultBlade/Overview/appSignInAudience/AzureADMyOrg...)

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

The screenshot shows the Microsoft Azure portal interface for managing app registrations. On the left, a sidebar lists various management options like Overview, Quickstart, Integration assistant, Manage, API permissions (which is selected), Expose an API, App roles, Owners, Roles and administrators, Manifest, Support + Troubleshooting, and Troubleshooting. The main content area is titled "Request API permissions" under "My APIs". It displays a warning about granting tenant-wide consent and a table of configured permissions for Microsoft Graph. The table has columns for API / Permissions name, Type, and Description, showing one entry for "User.Read" which is Delegated and used for "Sign in and read".

### 3. Generate a client secret:

- Navigate to Certificates & secrets.

The screenshot shows the Microsoft Azure portal interface for managing app registrations. The sidebar shows the "Certificates & secrets" option is selected. The main content area is titled "Certificates & secrets" and displays a table for new client secrets. The table has columns for Description, Expires, Value, and Secret ID. A note at the top states: "A secret string that the application uses to prove its identity when requesting a token. Also can be referred to as application password." Below the table, it says "No client secrets have been created for this application." A URL at the bottom of the page is: [https://portal.azure.com/#view/Microsoft\\_AAD\\_RegisteredApps/ApplicationMenuBlade/-/Credentials/appId/ed300fb9-e461-4733-af3a-ed123f87b608/objectId/13a641ee-490a-4002-b5a6-694e6991048/isMSAApp-/false/defaultBlade/Overview/appSignInAudience/AzureADMulti...](https://portal.azure.com/#view/Microsoft_AAD_RegisteredApps/ApplicationMenuBlade/-/Credentials/appId/ed300fb9-e461-4733-af3a-ed123f87b608/objectId/13a641ee-490a-4002-b5a6-694e6991048/isMSAApp-/false/defaultBlade/Overview/appSignInAudience/AzureADMulti...)

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

The screenshot shows the "Add a client secret" dialog box. It has two input fields: "Description" (with placeholder "Secret") and "Expires" (set to "Recommended: 180 days (6 months)").

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

<a href="#">+ New client secret</a>	Description	Expires	Value ⓘ	Secret ID	<a href="#">View</a>
	Secret	11/29/2024	4468Q~OiznZo4ZmG5Sl6gQvHMIpmLbL...	759166a0-366d-400a-afa2-d5a07e326e59	<a href="#">Edit</a>

## 4. Configure Your API App to Use Azure AD Authentication

### 4.1 Open your API App in the Azure portal.

The screenshot shows the 'MyAPIApp' registration page in the Azure portal. The 'Essentials' section displays the following details:

- Display name: MyAPIApp
- Application (client) ID: ed300fb9-e461-4733-a5a-ed123fb7b608
- Object ID: 123e45ee-49e3-4002-b5ad-ef914e099048
- Directory (tenant) ID: 6426505-dc8d-41ef-adf5-d4eb3b1925dc
- Supported account types: Multiple organizations

Notes at the bottom of the page:

- Starting June 20th, 2020 we will no longer add any new features to Azure Active Directory Authentication Library (ADAL) and Azure Active Directory Graph. We will continue to provide technical support and security updates but we will no longer provide feature updates. Applications will need to be upgraded to Microsoft Authentication Library (MSAL) and Microsoft Graph. [Learn more](#)
- Starting November 9th, 2020 end users will no longer be able to grant consent to newly registered multi-tenant apps without verified publishers. [Add MPN ID to verify publisher](#)

### 4.2 Navigate to Authentication/Authorization settings:

- Click on Add identity provider.



**Add an identity provider**  
Choose an identity provider to manage the user identities and authentication flow for your application.  
Providers include Microsoft, Facebook, Google, and Twitter.  
[Learn more about identity providers](#)

[Add identity provider](#)

- Now select GitHub as an Identity Provider.

The screenshot shows the 'Add identity provider' step in the Azure portal. The 'Identity provider' dropdown menu is open, displaying the following options:

- Microsoft
- Apple
- Facebook
- Github
- Google
- Twitter
- OpenID Connect

The 'GitHub' option is highlighted. At the bottom of the screen, there are navigation buttons: 'Add', '< Previous', and 'Next: Permissions >'.

### 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

Tokens store ...

**Actions** 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.

Home > SQL databases >

### Create SQL Database

Microsoft

**Basics** Networking Security Additional settings Tags Review + create

Create a SQL database with your preferred configurations. Complete the Basics tab then go to Review + Create to provision with smart defaults, or visit each tab to customize. [Learn more](#)

Want to try Azure SQL Database for free? Create a free serverless database with the first 100,000 vCore seconds, 32GB of data, and 32GB of backup storage free per month for the lifetime of the subscription. [Learn more](#)

Apply offer (Preview)

SQL Database Hyperscale: Low price, high scalability, and best feature set. [Learn more](#)

#### Project details

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

Subscription \* ⓘ Free Trial

Resource group \* ⓘ Select a resource group Create new

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

The screenshot shows the 'Create SQL Database' wizard. In the 'Database details' step, the following settings are selected:

- Subscription:** Free Trial
- Resource group:** TEST
- Database name:** test
- Server:** (new) test-api (East US)
- Compute + storage:** General Purpose - Serverless (Standard-series (Gen5), 2 vCores, 32 GB storage, zone redundant disabled)
- Behavior when free offer limit reached:** Auto-pause the database until next month (radio button selected)

At the bottom, there are 'Review + create' and 'Next : Networking >' buttons.

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

The screenshot shows the Azure Resource Group 'MSC-CC-PART1\_group' overview. The 'Essentials' blade displays the following resources:

Name	Type	Location	Actions
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-id-beb7	Application Insights	East US	...
MSC_CC_1	Managed Identity	East US	...
	Kubernetes - Azure Arc	East US	...

At the bottom right, there is a 'Give feedback' link.

## 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.

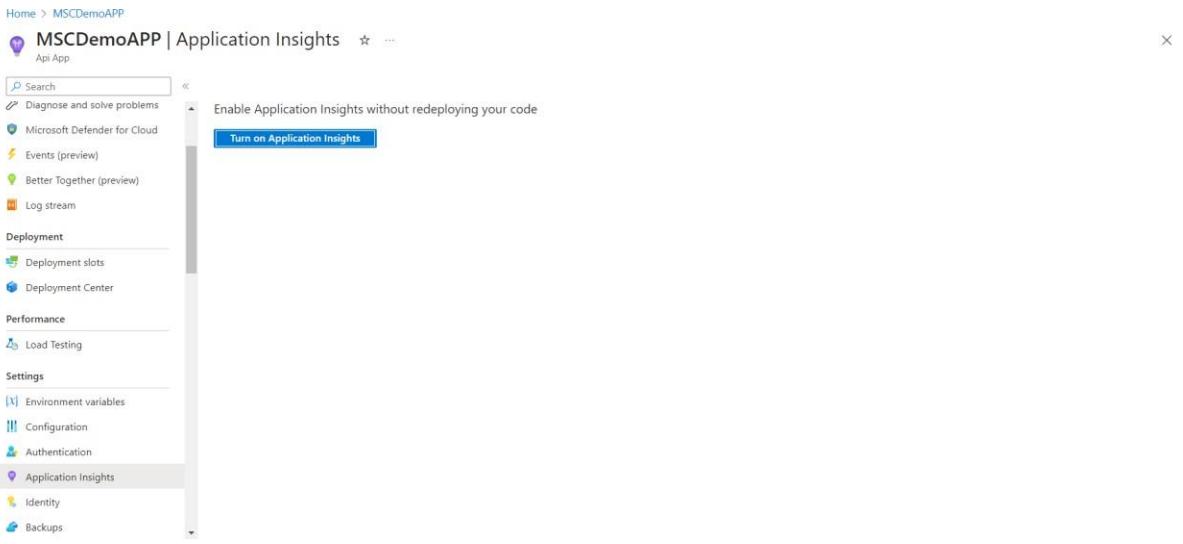
Additionally, under "Logs," configure "Log Analytics" to centralize logs from your API App. Define diagnostic settings to capture specific logs and metrics, and set up alerts for critical conditions. These tools together offer comprehensive insights into your API App's performance, usage patterns, and potential issues, facilitating proactive maintenance and troubleshooting.

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

The screenshot shows the Azure Portal interface. At the top, there's a navigation bar with 'Azure services' and various service icons like Create a resource, App registrations, SQL databases, Quickstart Center, Azure AI services, Kubernetes services, Virtual machines, App Services, Storage accounts, and More services. Below this is the 'Resources' blade. It has tabs for 'Recent' (which is selected) and 'Favorite'. The 'Recent' tab lists resources: MSCDemoAPP (App Service), test (SQL database), test-api (SQL server), ASP-TEST-9b13 (App Service plan), and TEST (Resource group). Each item shows its type, last viewed time, and a small icon. At the bottom of the 'Recent' list is a 'See all' link.

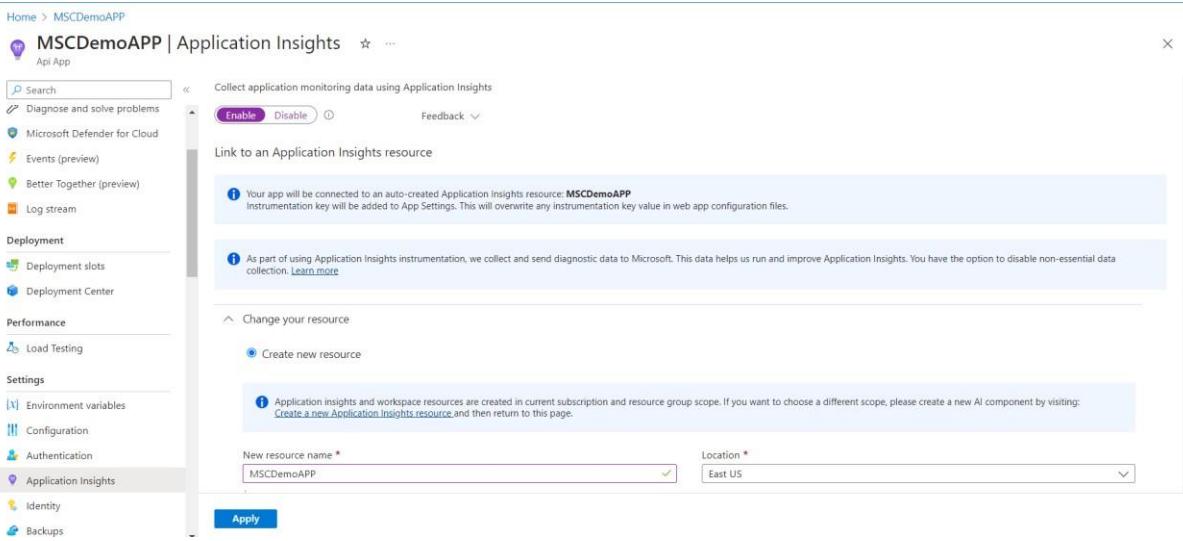
Step 2: Go to Application Insights:

- 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:

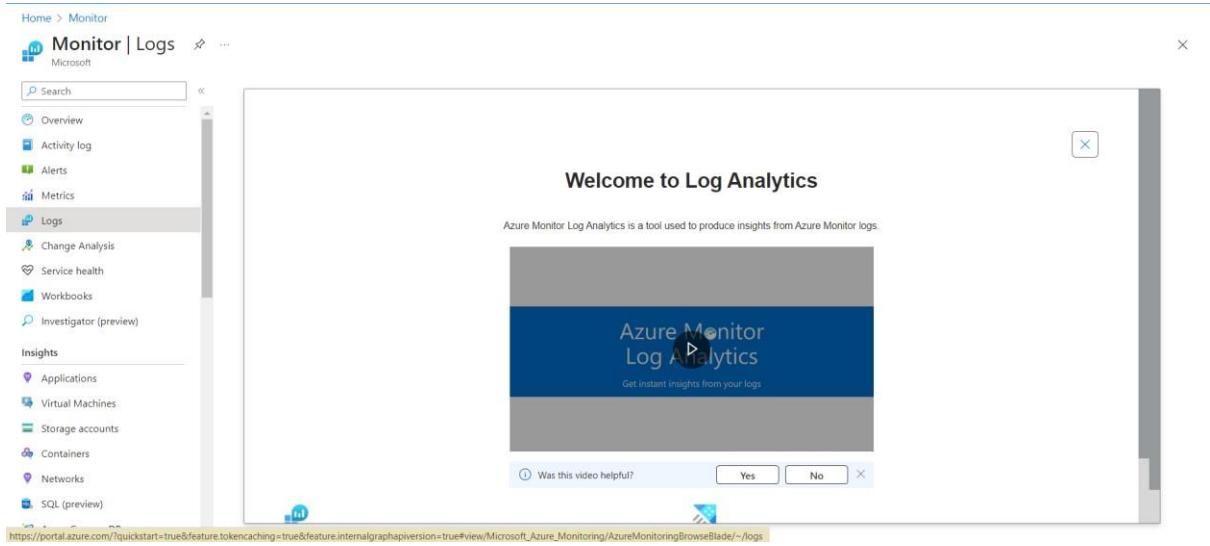
- Once enabled, Application Insights will automatically start collecting datalike request rates, response times, failure rates, dependency tracking, and more.



### 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.

The screenshot shows the 'Diagnostic settings' page for an Application Insights resource named 'MSCDemoAPP'. The top navigation bar includes 'Home > Monitor' and the title 'Monitor | Diagnostic settings'. The left sidebar lists Managed Services (Managed Prometheus, Azure Managed Grafana, Azure Monitor SCOM managed instance) and Settings (Diagnostic settings, Data Collection Rules, Data Collection Endpoints, Azure Monitor pipelines (preview), Autoscale, Private Link Scopes). The main content area shows diagnostic settings for the 'TEST' resource group. It includes filters for Subscription (Free Trial), Resource group (TEST), Resource type (Application Insights), and Resource (MSCDemoAPP). A table lists diagnostic settings, with a note that none are defined. Below the table, a list of data types available for collection is provided, including Availability results, Browser timings, Events, Metrics, Dependencies, Exceptions, Page views, Performance counters, Requests, System events, Traces, and AllMetrics.

## Step 6: Configure diagnostics logs:

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

**Diagnostic setting** ... X[Save](#) [Discard](#) [Delete](#) [Feedback](#)

A diagnostic setting specifies a list of categories of platform logs and/or metrics that you want to collect from a resource, and one or more destinations that you would stream them to. Normal usage charges for the destination will occur. Learn more about the different log categories and contents of those logs.

JSON View

Diagnostic setting name \*

Secret ✓**Logs**Category groups (1) allLogs**Categories** Availability results Browser timings Events Metrics Dependencies Exceptions Page views**Destination details** Send to Log Analytics workspace

## Subscription

Free Trial ✓

## Log Analytics workspace

DefaultWorkspace-a205137a-a473-4859-a0c3-012df65da374-EUS ( eastus ) ✓ Archive to a storage account Stream to an event hub Send to partner solution