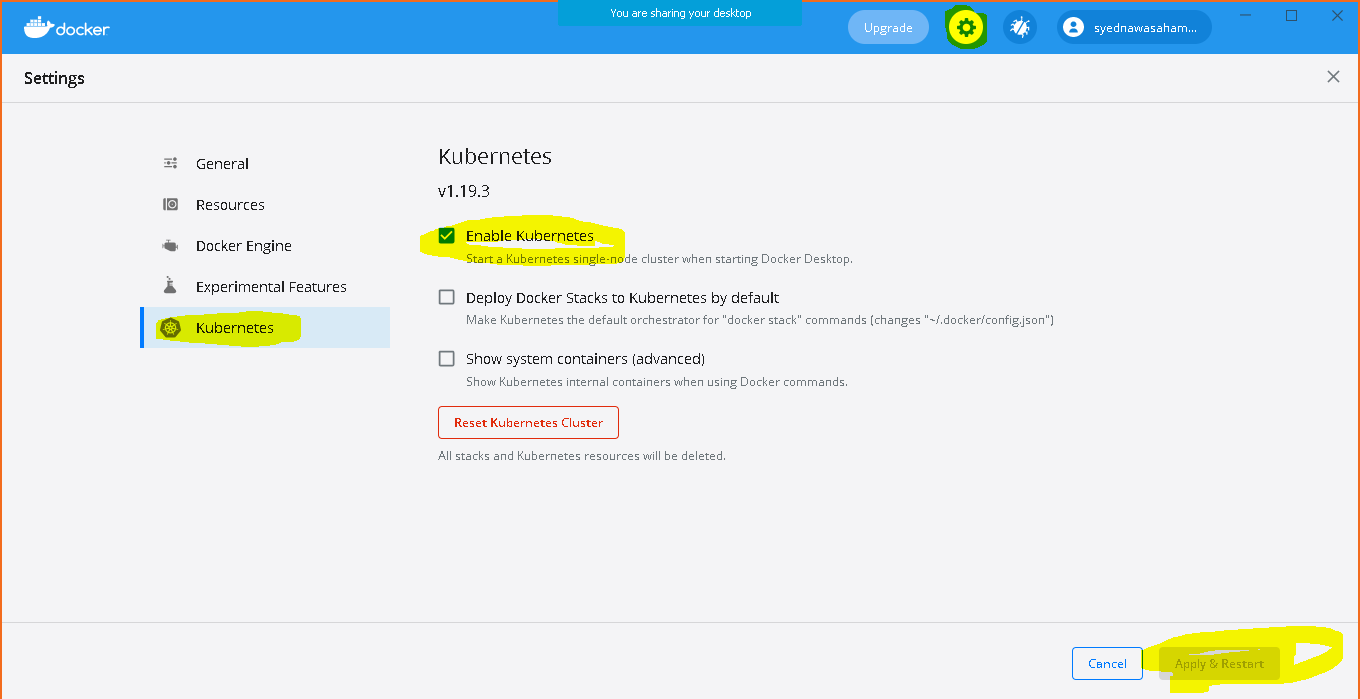
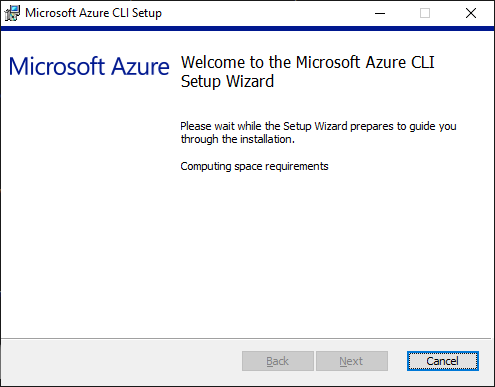
**Deploy a Micro service to Azure-Kubernetes**

Prerequisites

**1. Download, Install and Enable Kubernetes in you Docker Desktop**



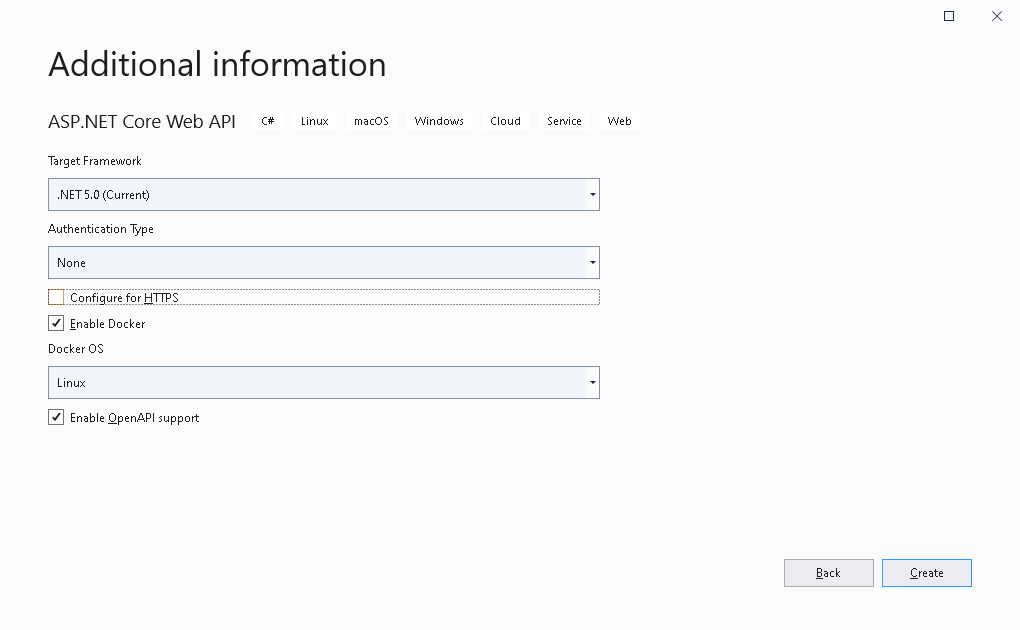
**2. Install Microsoft Azure CLI**



**Chapter 1 – Create a Micro service**

**Stage1 - Create DemoCustMS Web API Project (In Visual Studio)**

Create ASP.NET Core **Web API Project** and Give the Name as “DemoCustMS”, in following screen Enable “Docker”



**Add Models, DBContext and InMemory DB Service**

using System;

using System.ComponentModel.DataAnnotations;

namespace DemoCustMS.Models

{

public class **Customer**

{

[Key]

public int CustomerId { get; set; }

public string Name { get; set; }

public string Address { get; set; }

public DateTime DOB { get; set; }

public string PAN\_Number { get; set; }

}

}

using System.ComponentModel.DataAnnotations.Schema;

namespace DemoCustMS.Models

{

public class **CustomerCreationStatus**

{

[ForeignKey("CustomerId")]

public int CustomerId { get; set; }

public Customer Customer { get; set; }

public string Message { get; set; }

}

}

Add a Class called **Customer Service** to Seed **In-memory DB**

(Create **Services** folder inside the project and add **CustomerService.cs**)

using System;

using System.Collections.Generic;

using System.Linq;

//Added...

**using** DemoCustMS**.Models;**

namespace DemoCustMS.Services

{

public static class CustomerService

{

static List<Customer> Customers { get; }

static CustomerService()

{

Customers = new List<Customer>

{

new Customer {

CustomerId = 1001,

Name = "Karthik",

Address = "Chennai",

DOB = Convert.ToDateTime("10/11/2000"),

PAN\_Number = "BQMPS6580E"

},

new Customer {

CustomerId = 1002,

Name = "Megala",

Address = "Chennai",

DOB = Convert.ToDateTime("10/11/2000"),

PAN\_Number = "BQMPS6580F"

},

new Customer {

CustomerId = 1003,

Name = "Nikhil Reddy",

Address = "Hyderabad",

DOB = Convert.ToDateTime("10/12/2000"),

PAN\_Number = "BQMPS6580G"

},

new Customer {

CustomerId = 1004,

Name = "Sivani Josyula",

Address = "Hyderabad",

DOB = Convert.ToDateTime("10/13/2000"),

PAN\_Number = "BQMPS6580I"

},

};

}

static int nextId = 1005;

public static List<Customer> **GetAll**()

{

return Customers;

}

public static Customer **Get**(int id)

{

return Customers.FirstOrDefault(p => p.CustomerId == id);

}

public static CustomerCreationStatus **Add**(Customer cust)

{

cust.CustomerId = nextId++;

Customers.Add(cust);

return new CustomerCreationStatus

{

CustomerId = cust.CustomerId,

Customer = cust,

Message = "Customer Record Created. Customer Service will connect with AccountService to create Account."

};

}

public static void Delete(int id)

{

var Customer = Get(id);

if (Customer is null)

return;

Customers.Remove(Customer);

}

public static void Update(Customer Customer)

{

var index = Customers.FindIndex(p => p.CustomerId == Customer.CustomerId);

if (index == -1)

return;

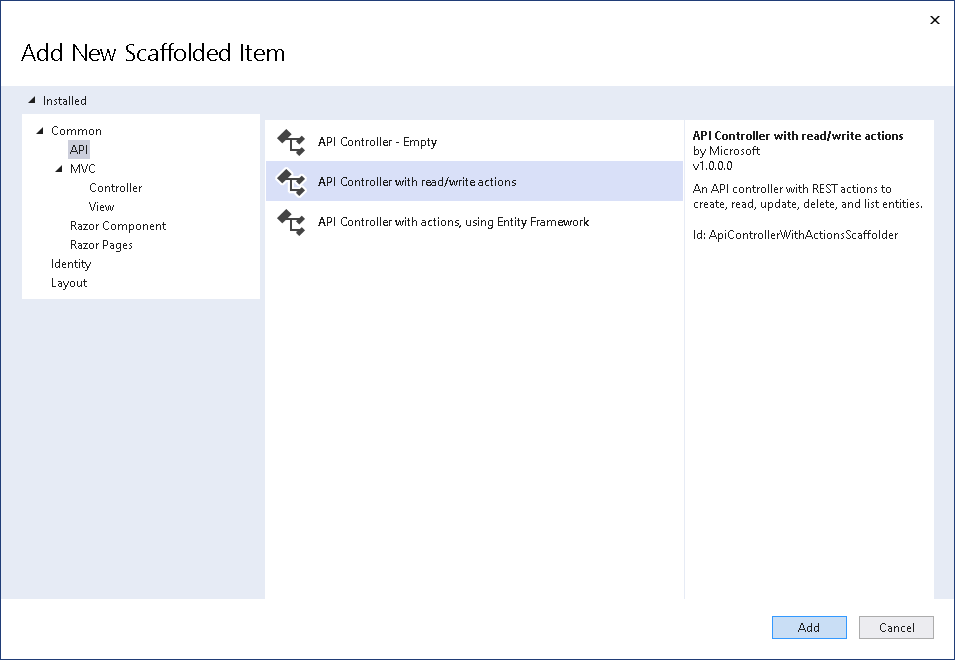
Customers[index] = Customer;

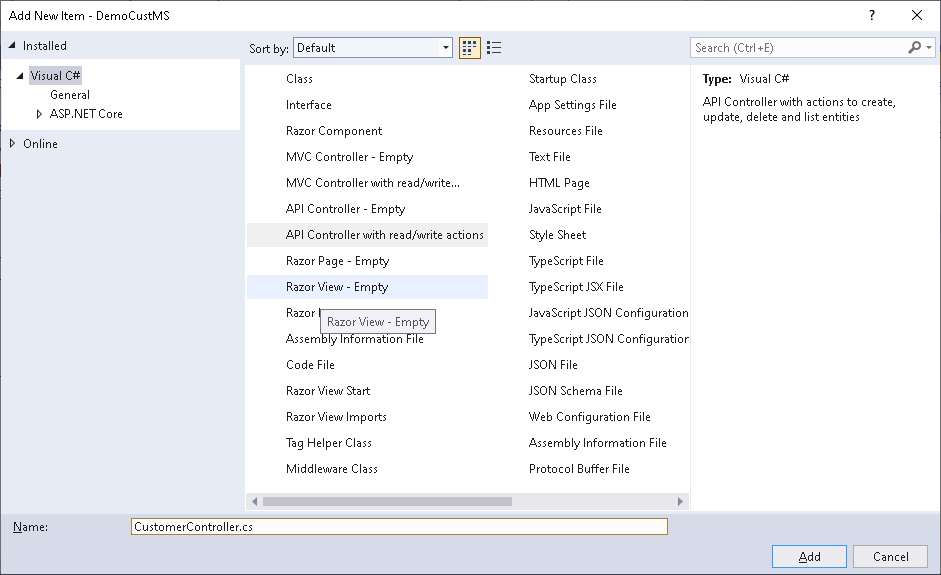
}

}

}

Add Customer Controller (API Controller – with Read / Write actions)





using Microsoft.AspNetCore.Mvc;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

**//Added..**

**using DemoCustMS.Services;**

**using DemoCustMS.Models;**

namespace DemoCustMS.Controllers

{

//[Route("api/[controller]")] - This was Removed and [Route("api")] added to have customized API URLs

[Route("api")]

[ApiController]

public class **CustomerController** : ControllerBase

{

// GET: api/<CustomerController>

[HttpGet]

[Route("Customers")]

public IEnumerable<Customer> **Get()**

{

return CustomerService.GetAll();

}

// GET api/<CustomerController>/5

[HttpGet]

[Route("GetCustomerDetails/{id}")]

public Customer **GetCustomerDetails(int id)**

{

return CustomerService.Get(id);

}

// POST api/<CustomerController>

[HttpPost]

[Route("CreateCustomer")]

public CustomerCreationStatus **CreateCustomer(Customer cust)**

{

return CustomerService.Add(cust);

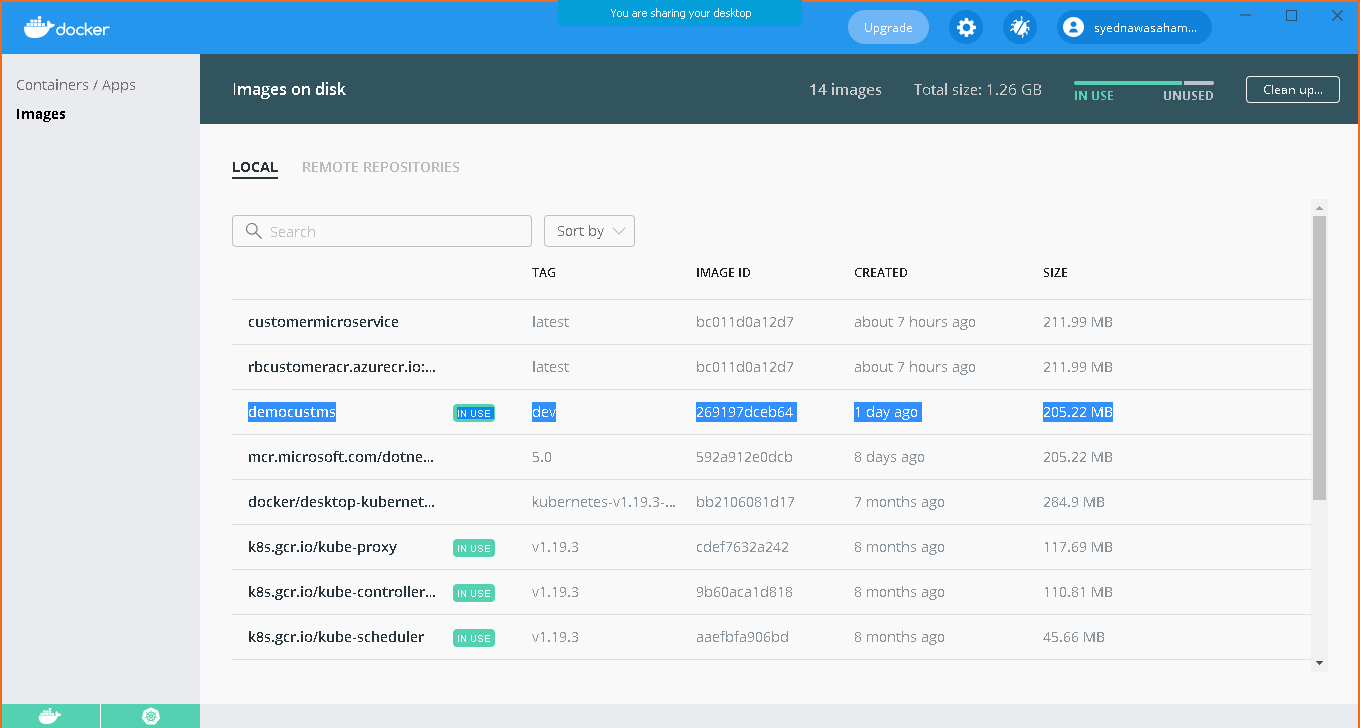
}

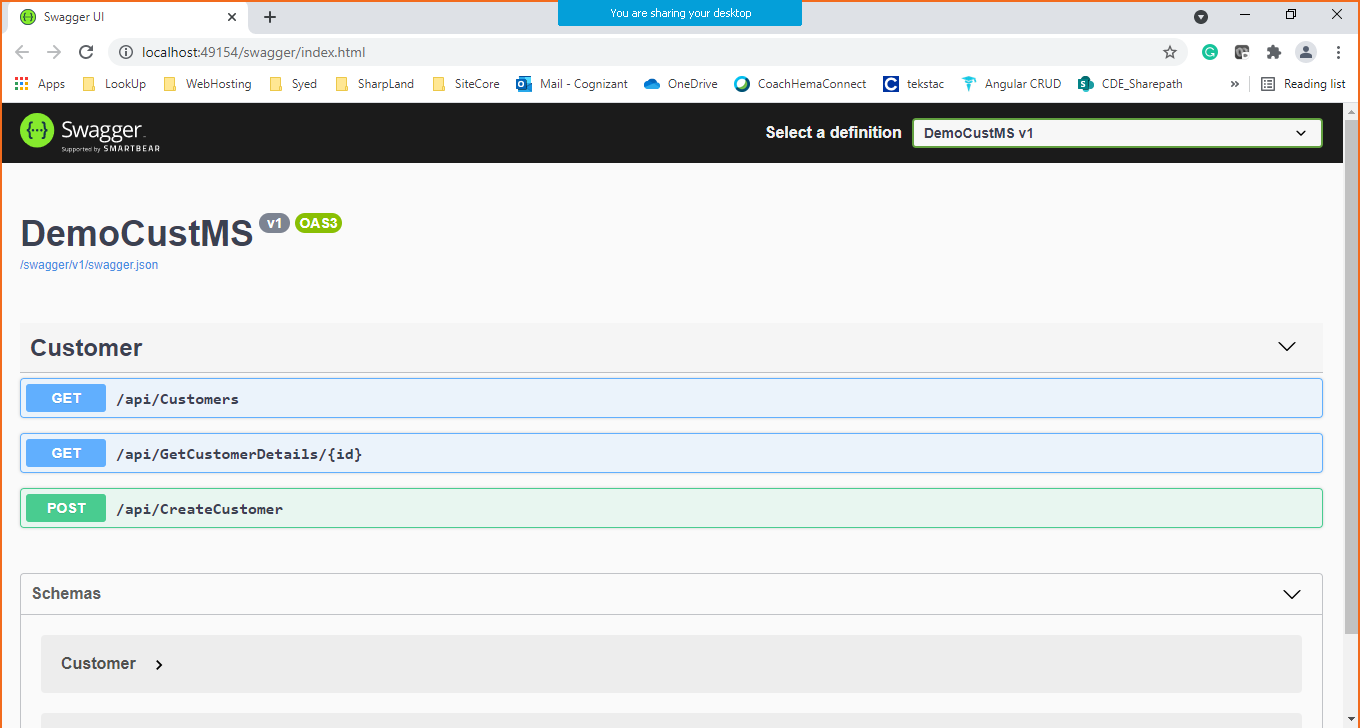
}

}

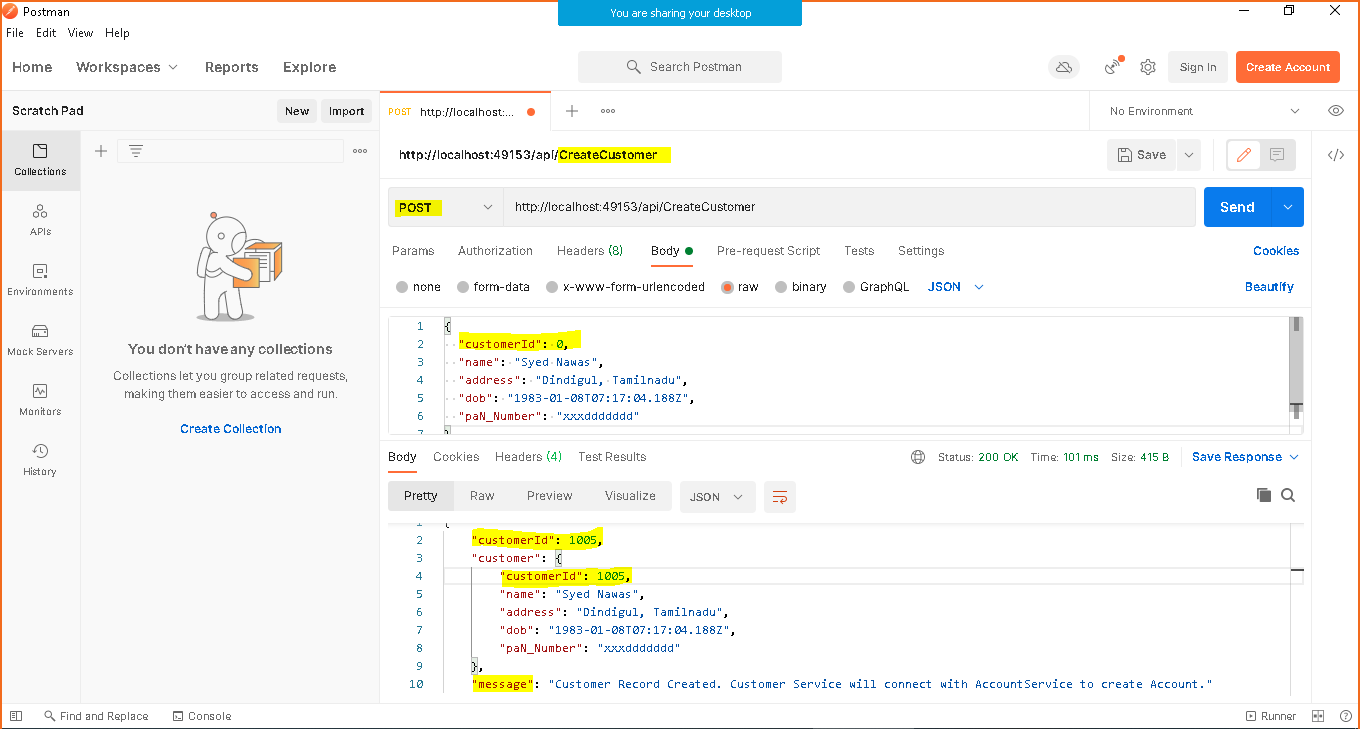
**Stage2 - Run the Application, Select “Docker” as environment instead of IIExpress**

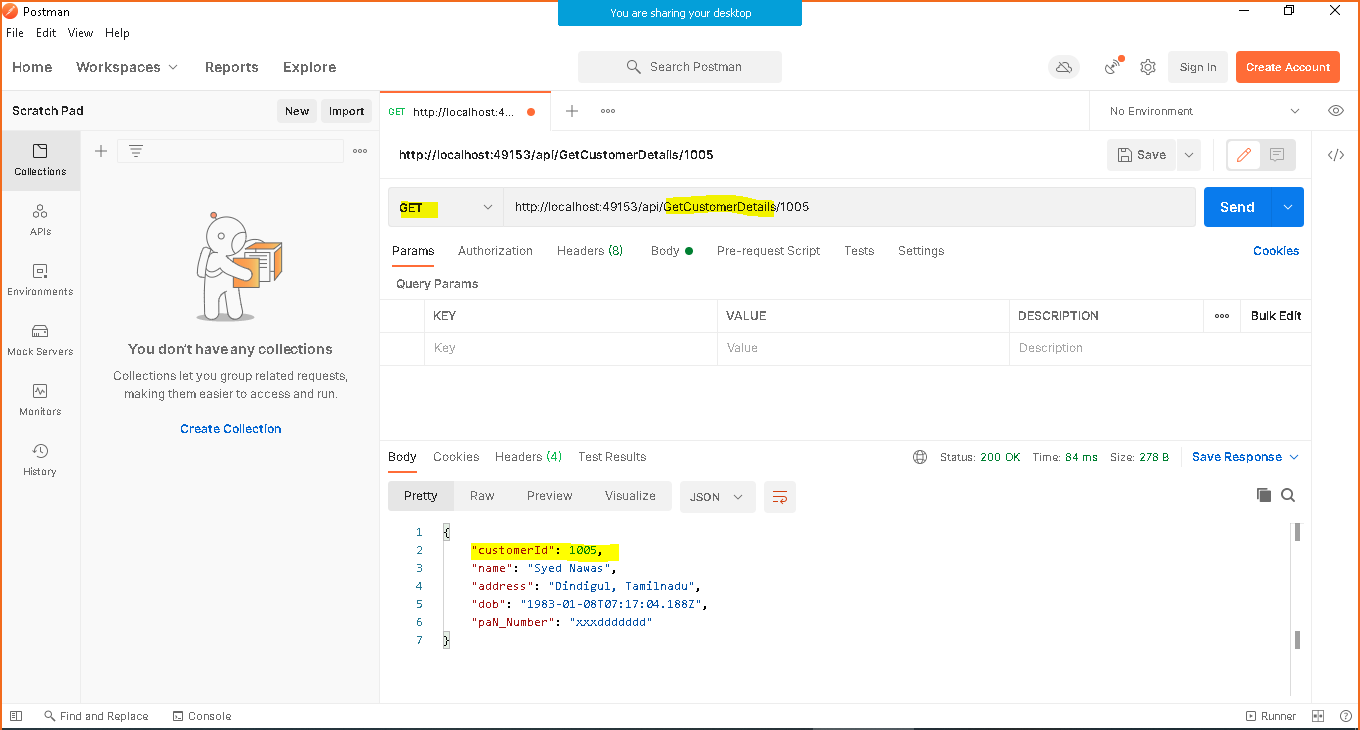
As we have enabled “Docker” for this project, “Dockerfile” would be present in project. We can notice that docker image has added in Docker Desktop and application is running as follows.





**Note: We can test all Customer WebAPI URLs (Web API Action Methods) by postman**





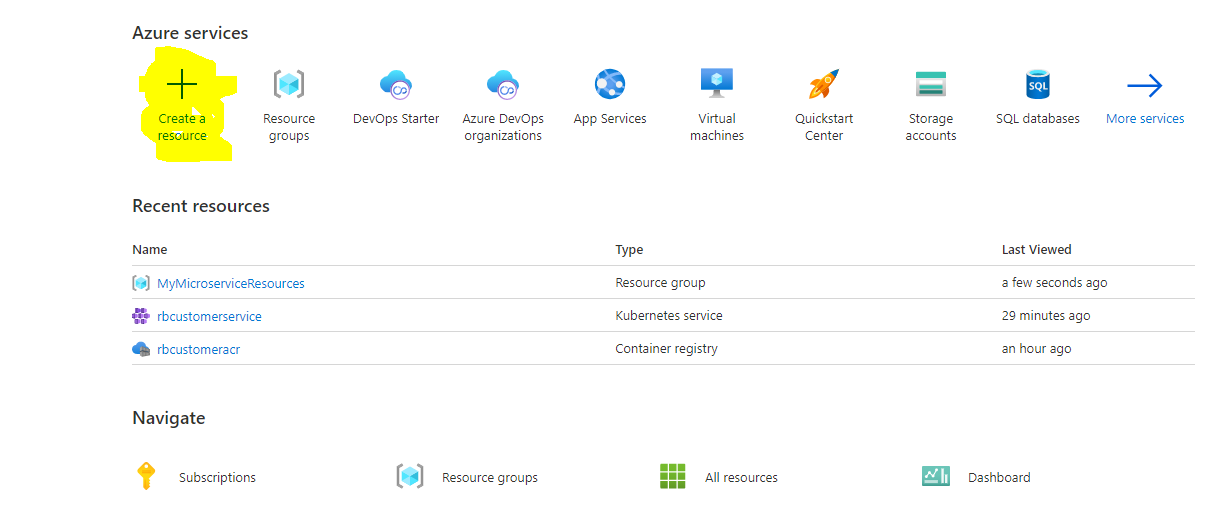
**Chapter 2 – Azure Resources**

**Resource Group, Container Registry, Kubenetes Cluster**

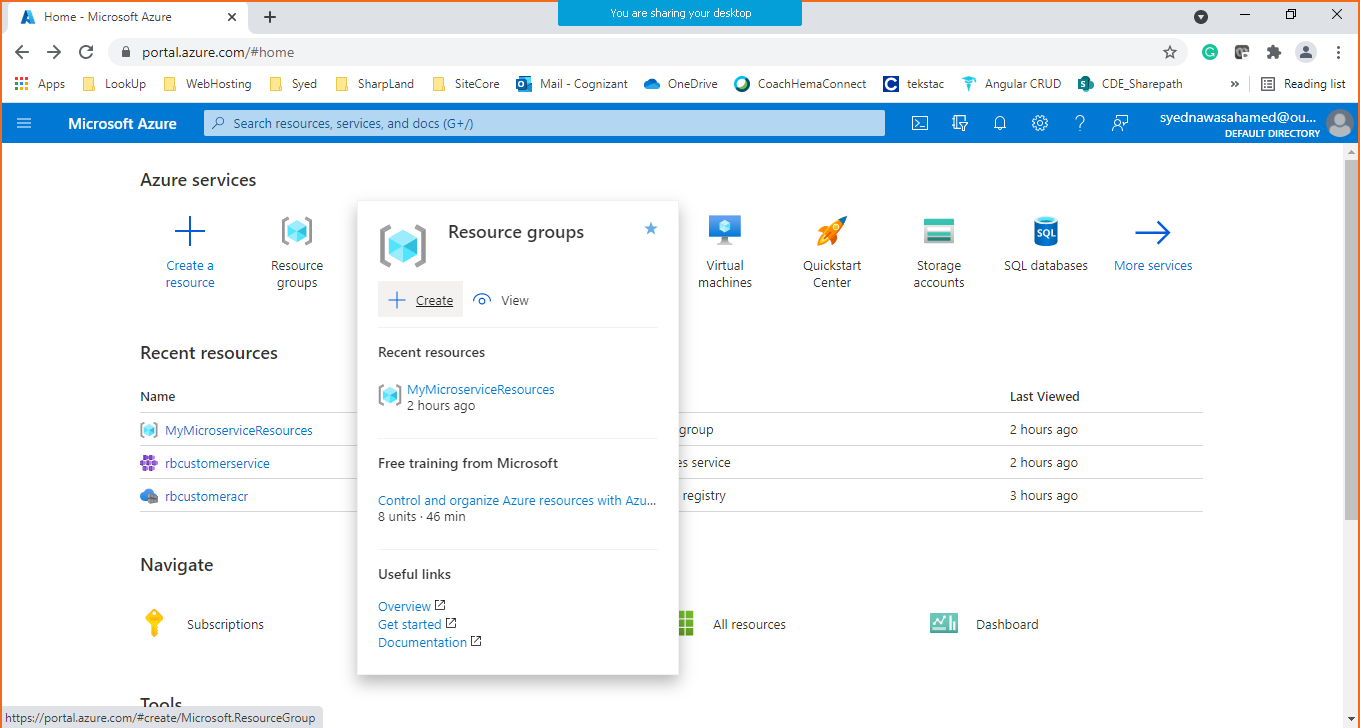
Stage3: Create Azure Resource Group

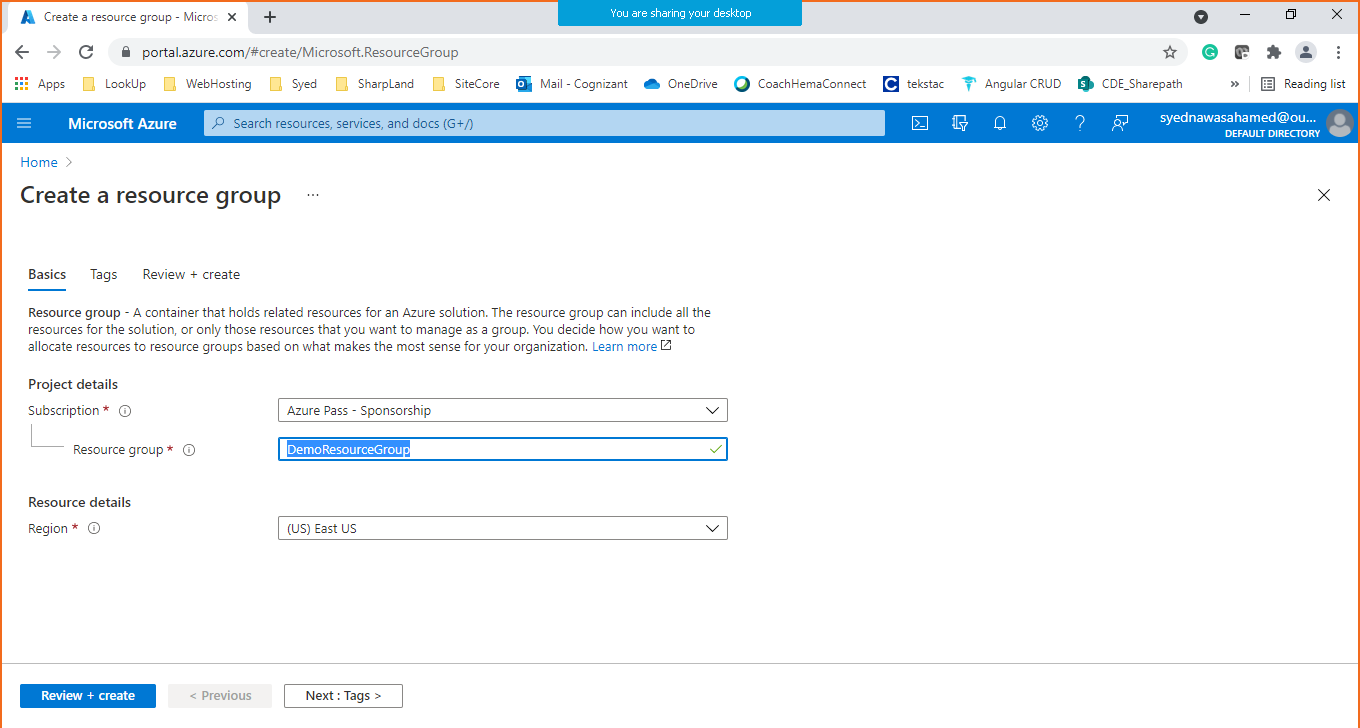
1. Login in to Azure Account with your credentials

2. Create **Azure Resource Group** – DemoResourceGroup (you can use any one of the methods as below)

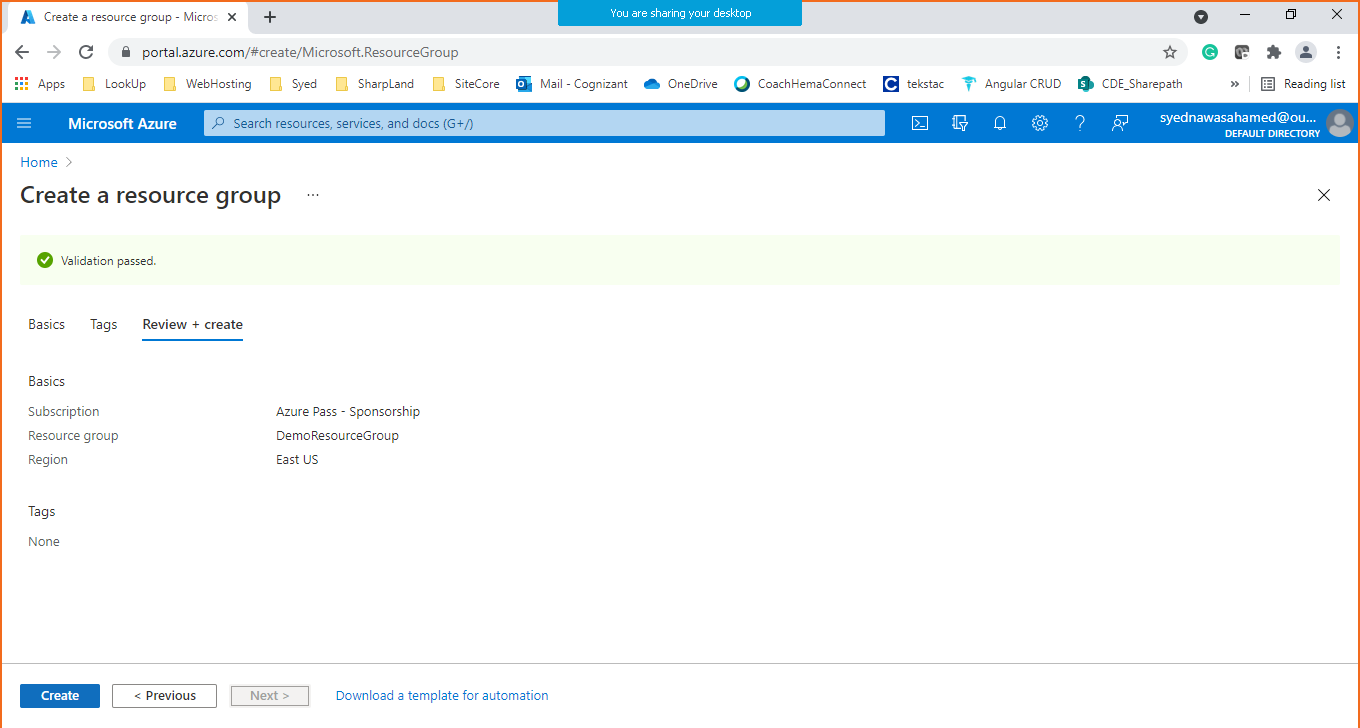


or

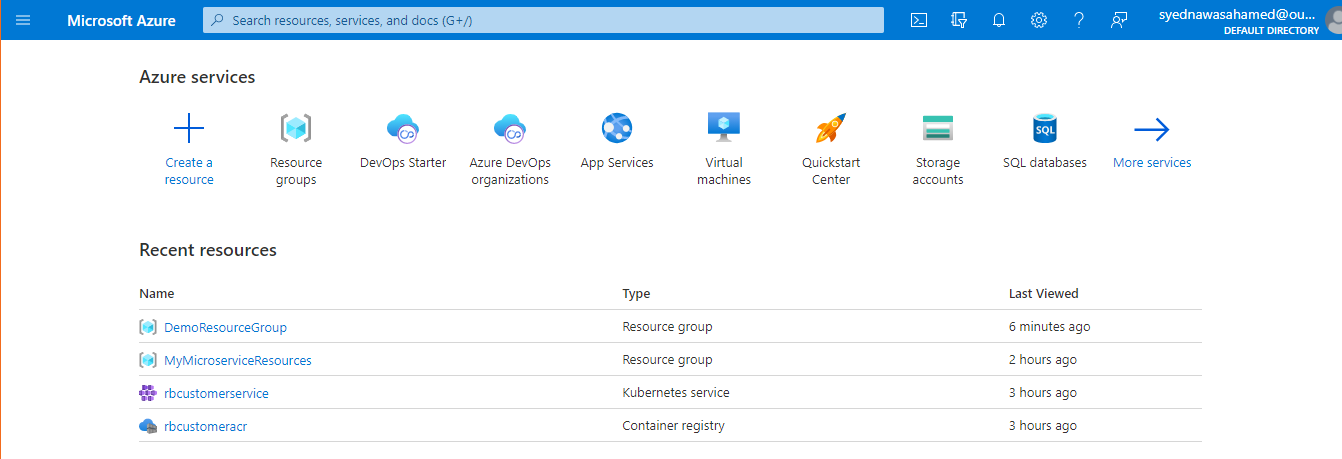




Give the Resource Group Name and Select Next Button.

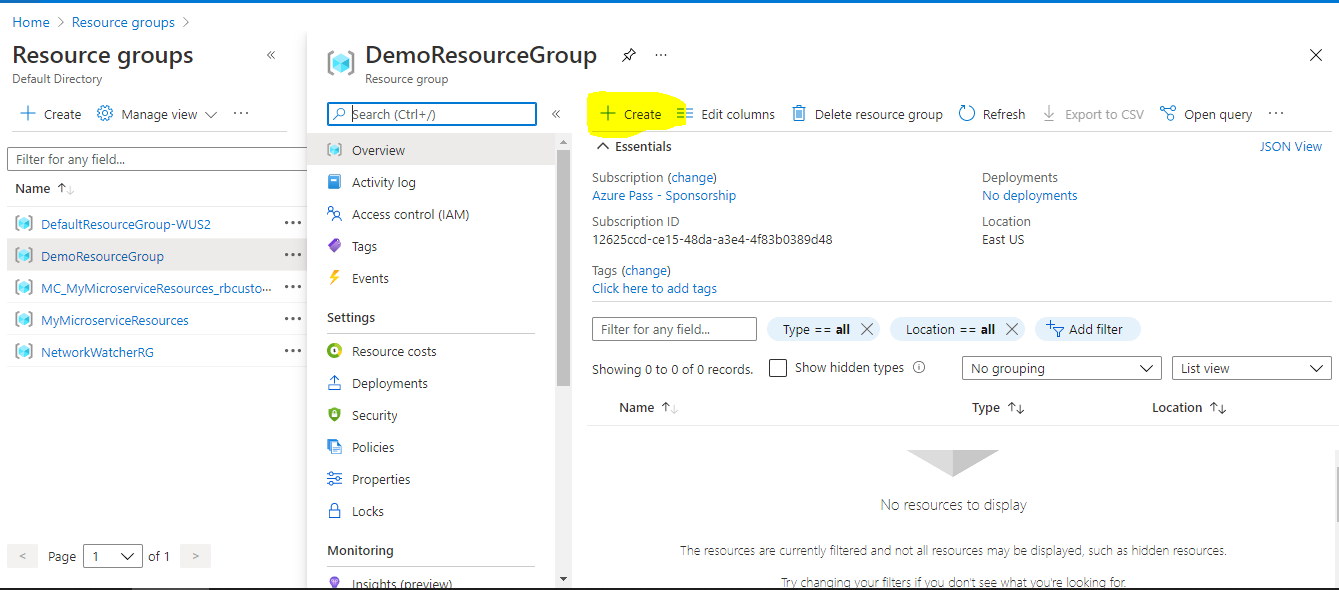


Click **Create** Button to create Resource Group Name, Below confirm that DemoResourceGroup is created

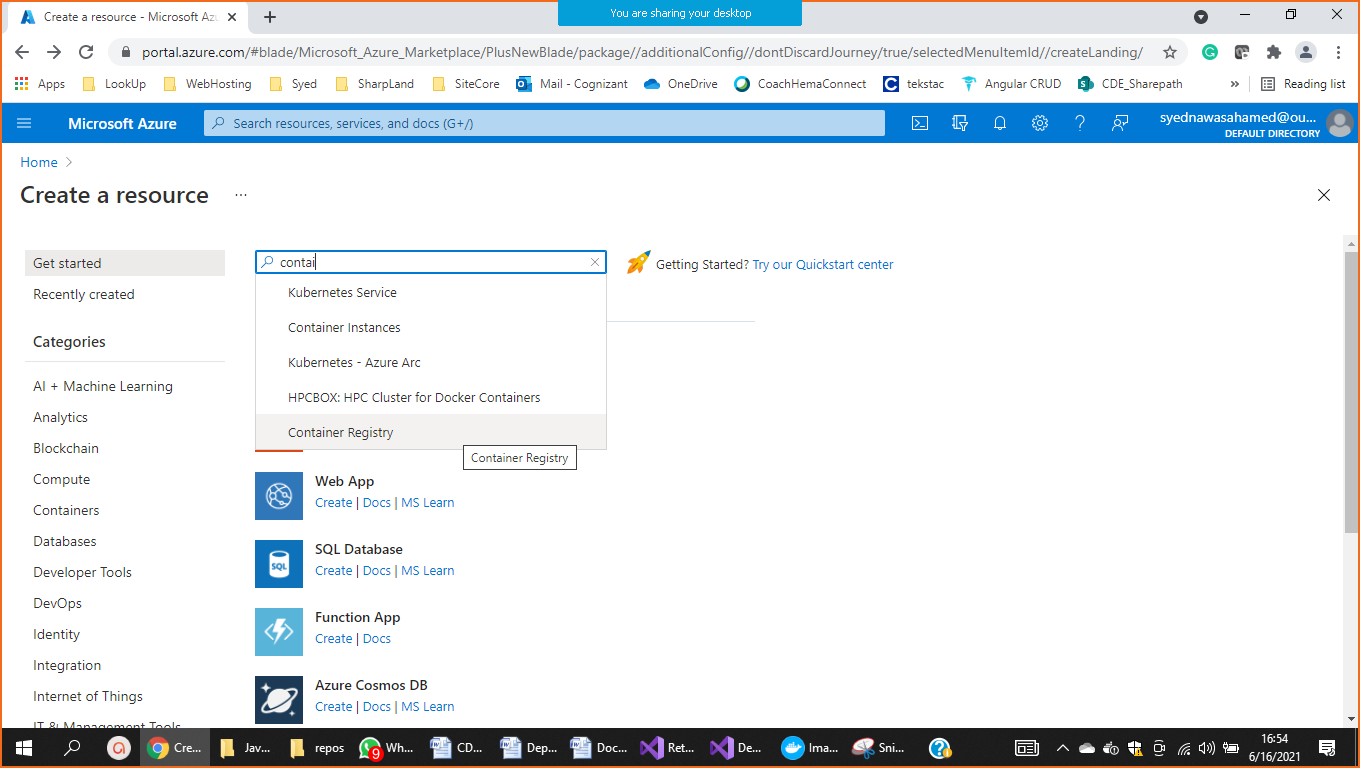


**Stage 4 : Create Container Registry (ACR)**

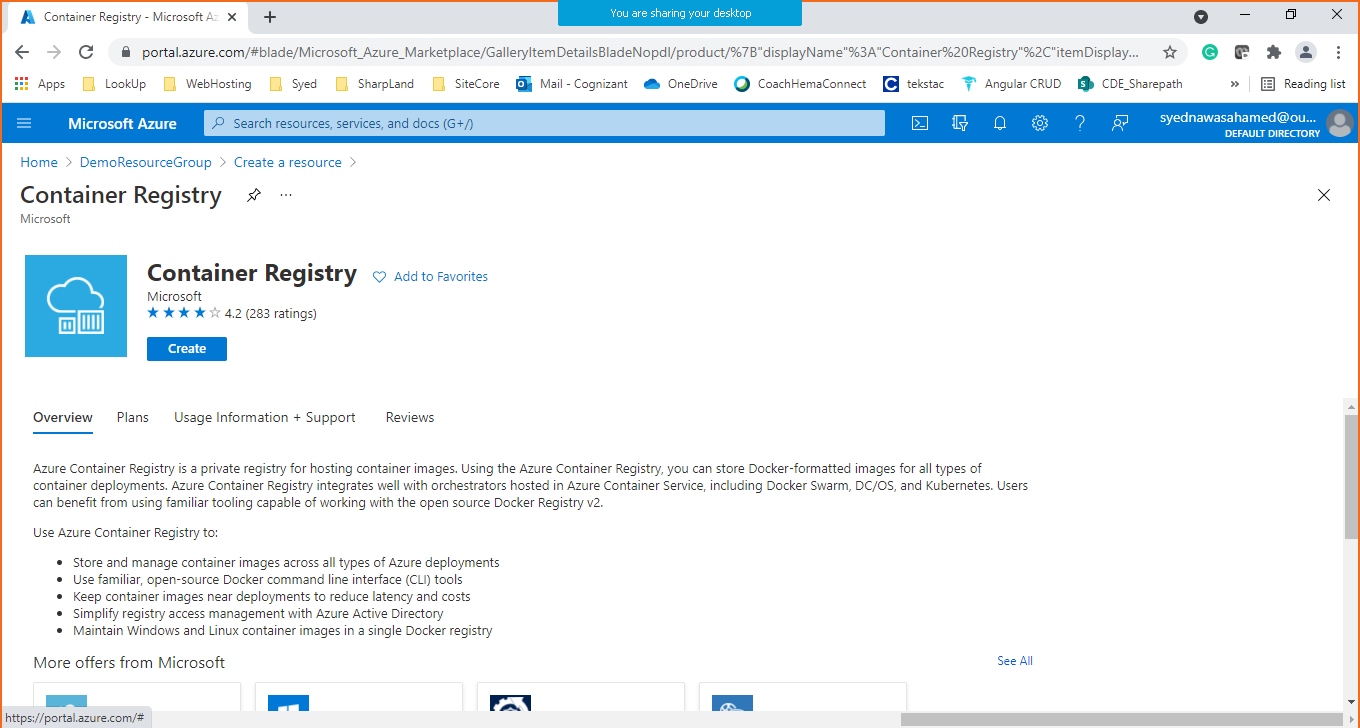
Navigate into **“DemoResourceGroup”** we created just now and there you click **Create** option as below



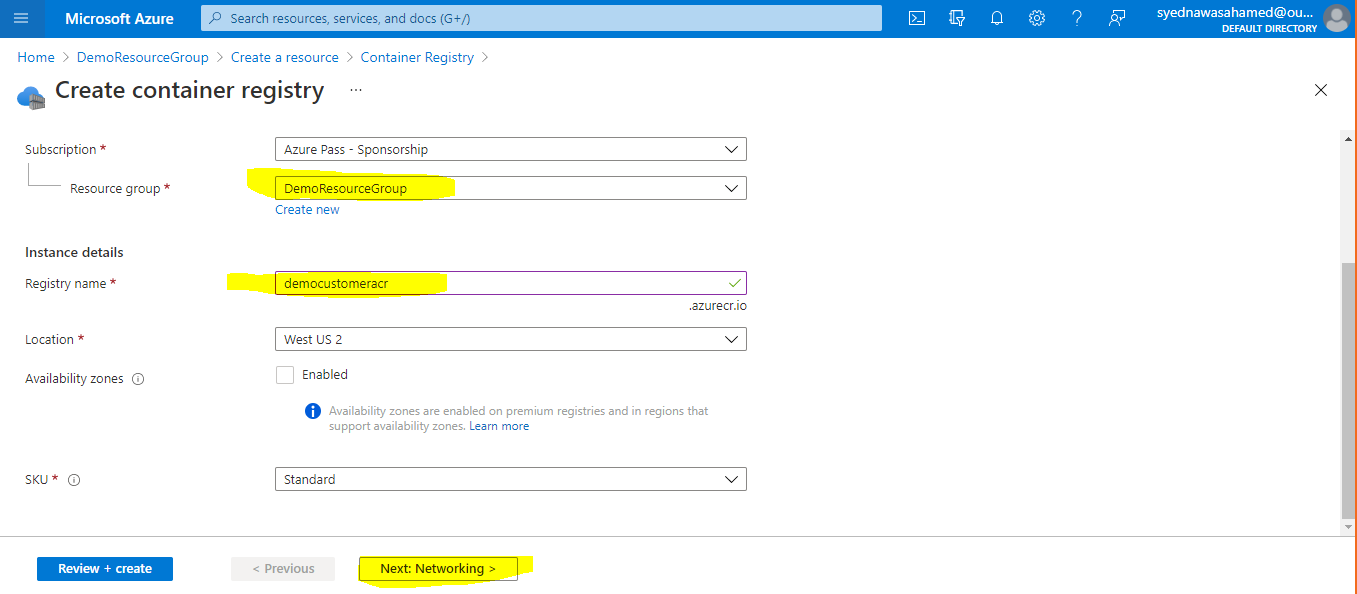
Search for the “container Registry” resource on **Create a resource** page



Select “Container Registry” from the dropdown menu. The you will navigate into following page

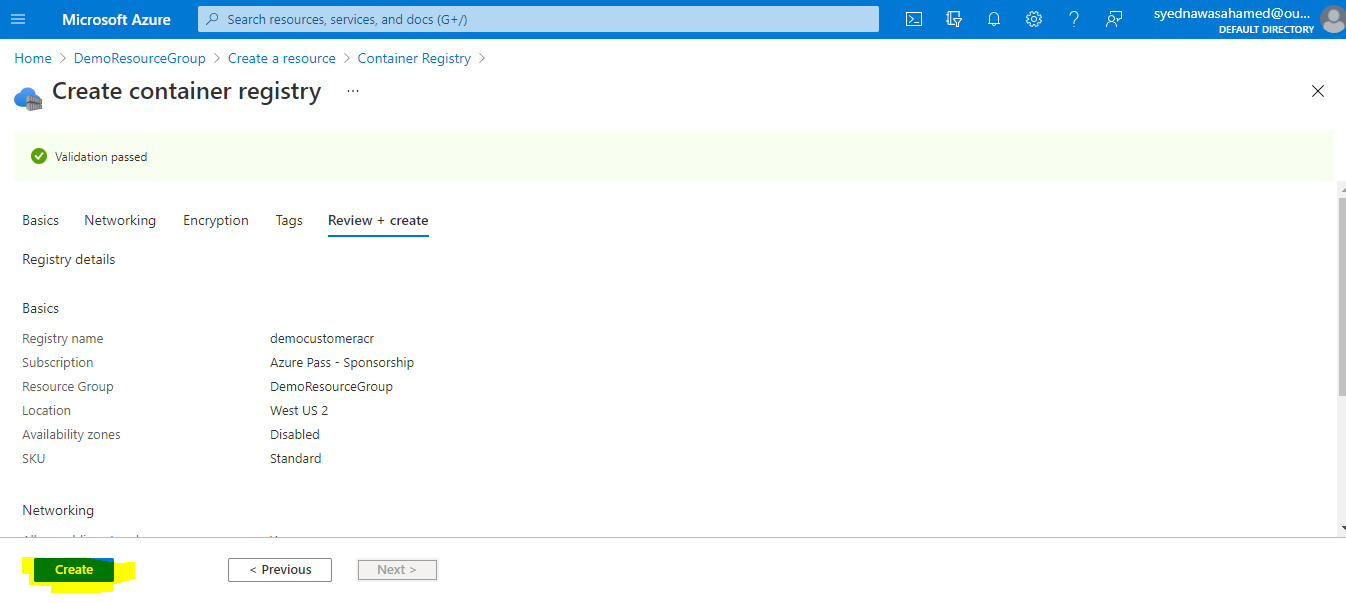


Click on “Create” Button

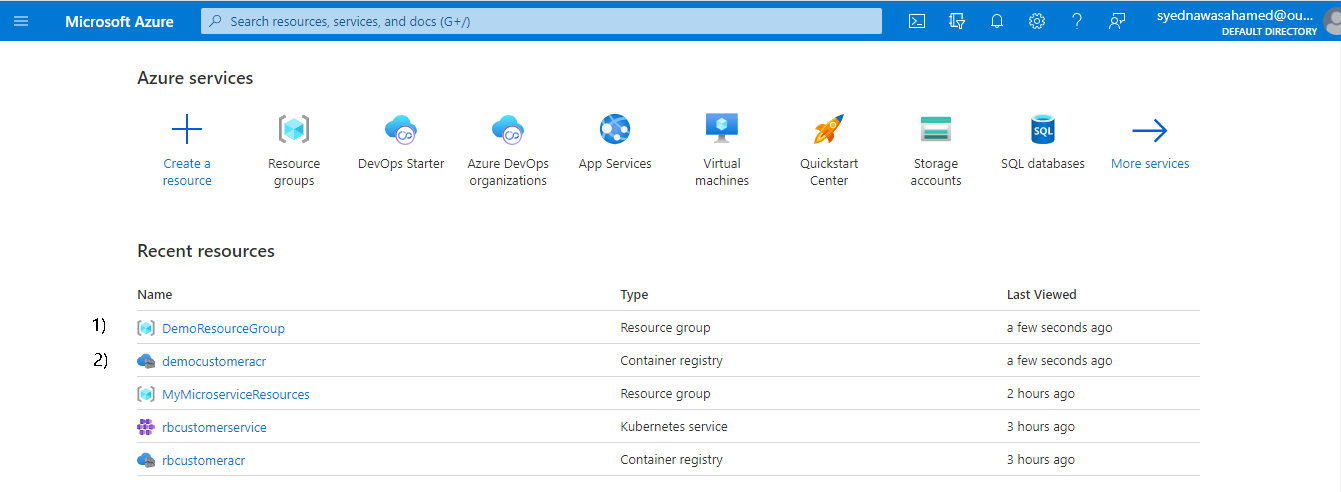


Select the **Resource Group,** Give Registry Name as democustomer**acr**. acr suffix will be added as industry standard to show it is azure container registry. Azure container registry should have unique name across the geo location.

Leave the “Networking”, “Encryption”, “Tags” and click the Review + Create. See the reviews of your selection.

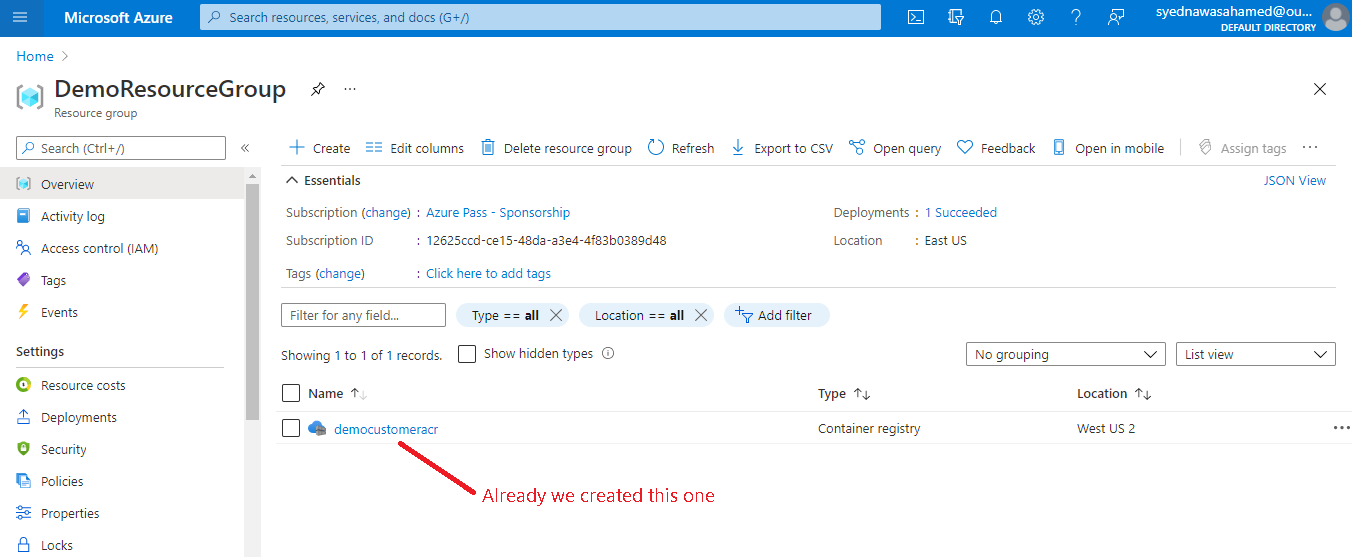


Click “**Create**” Button, then your container registry will be ready on dashboard as below.

 Now, you have Azure Resource Group, Container Registry resources on your Azure account.

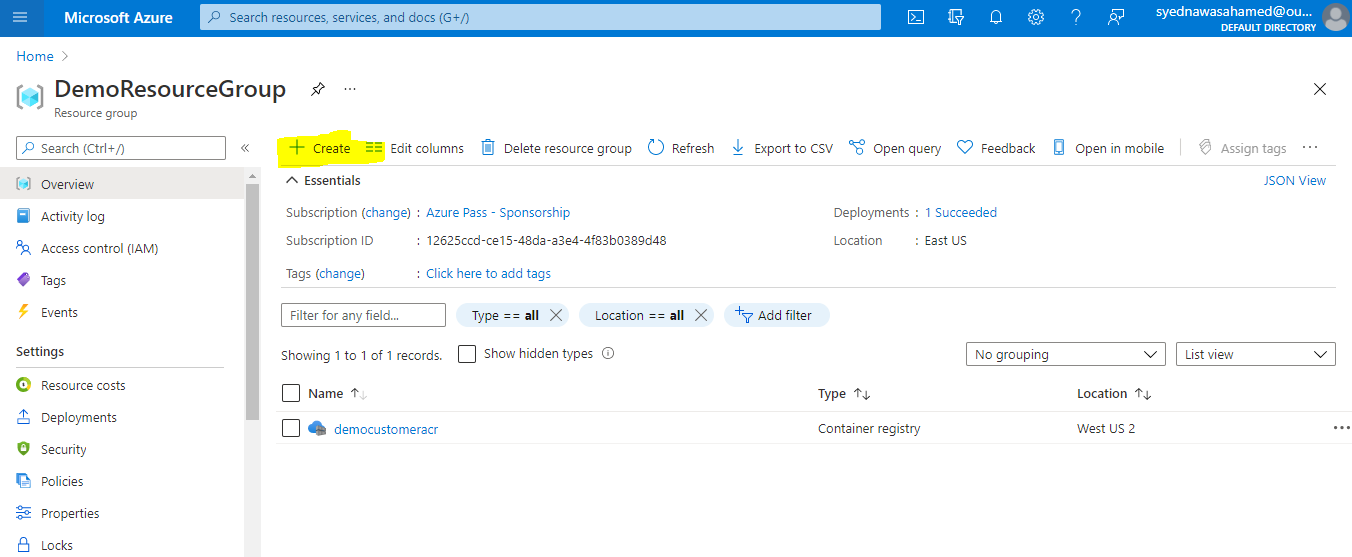
Our container registry has successfully created inside the **DemoResourceGroup** Resource group.

We can confirm by navigating as below.

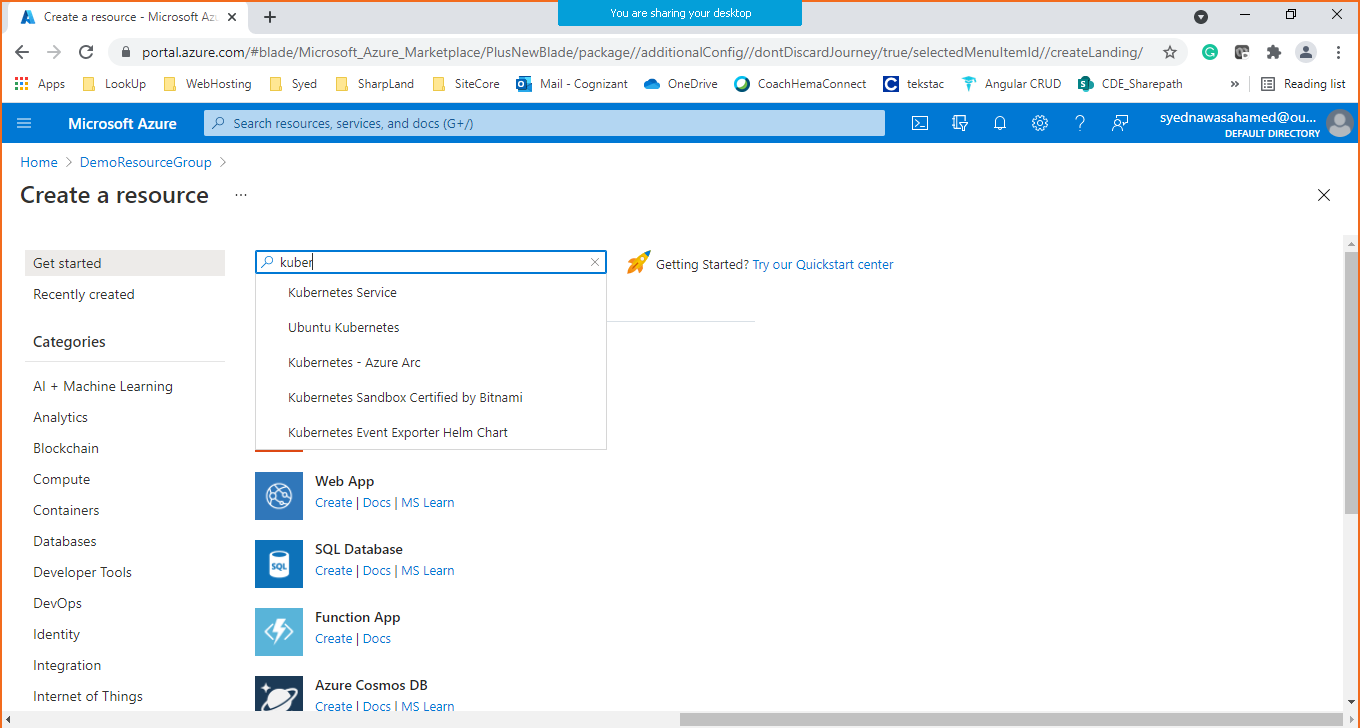


**Stage 5: Create Azure Kubernete Service (AKS) or Kubenetes Cluster**

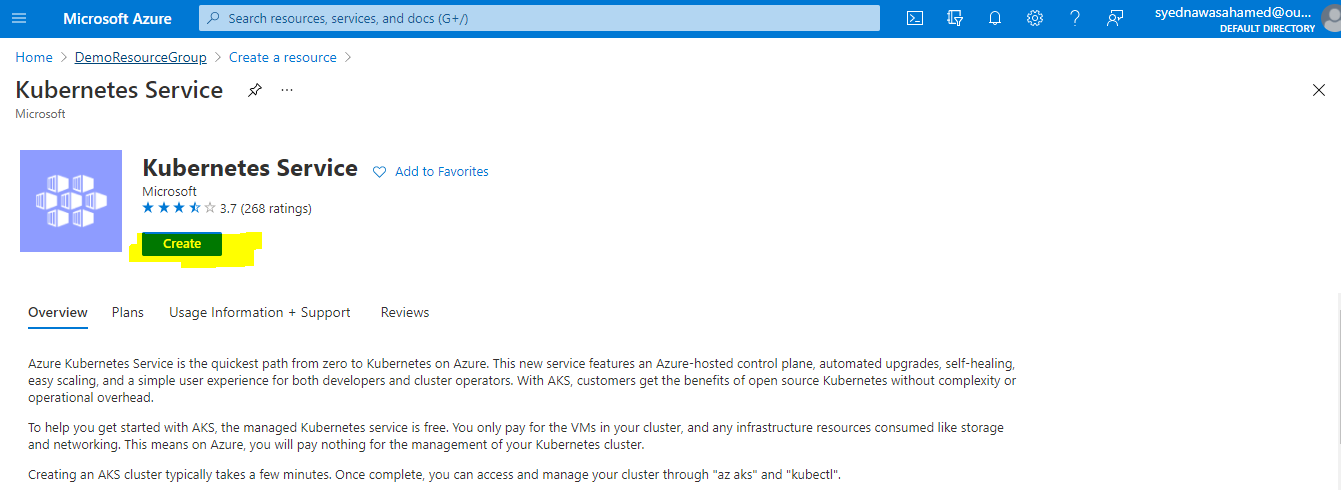
Navigate into **DemoResourceGroup**, Click **Create** Option there in order to create Kubernete Service.



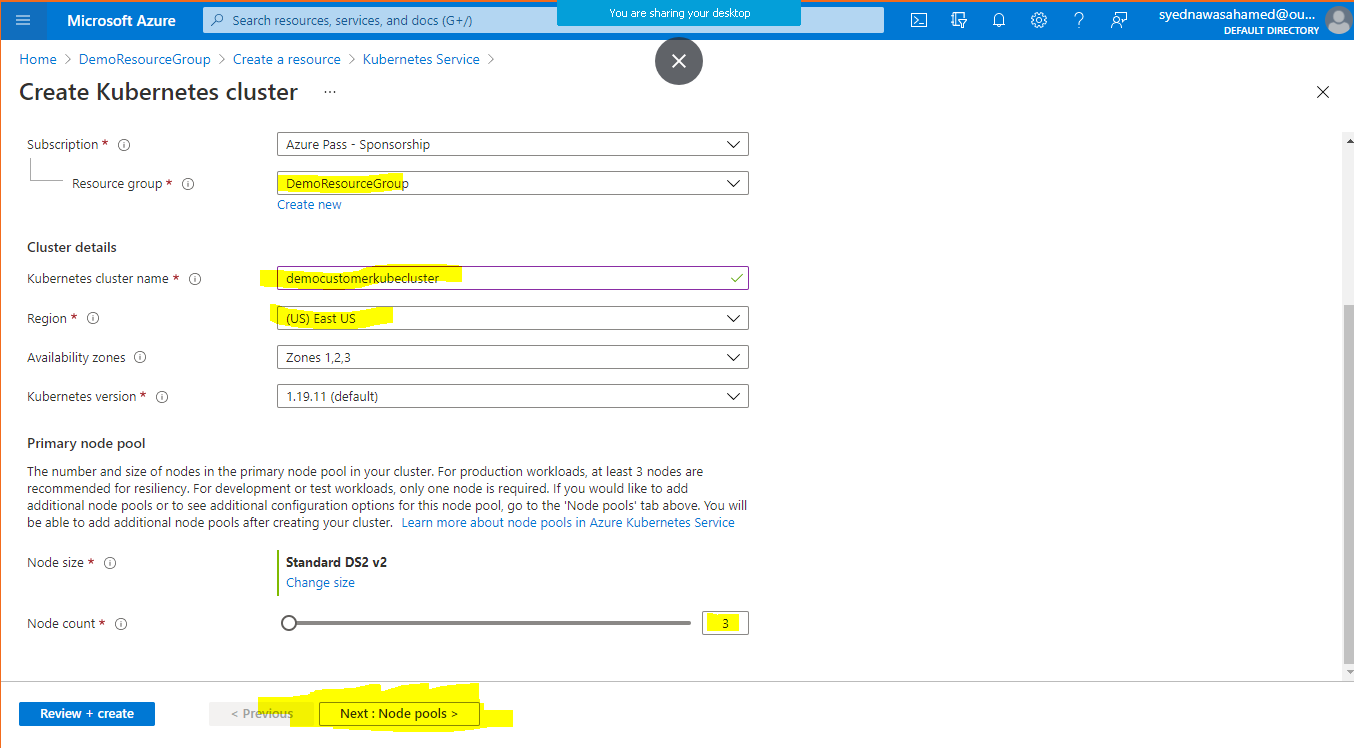
In the following **Create a resource** page, Search for Kubernetes Service, we will get redirected to next page.



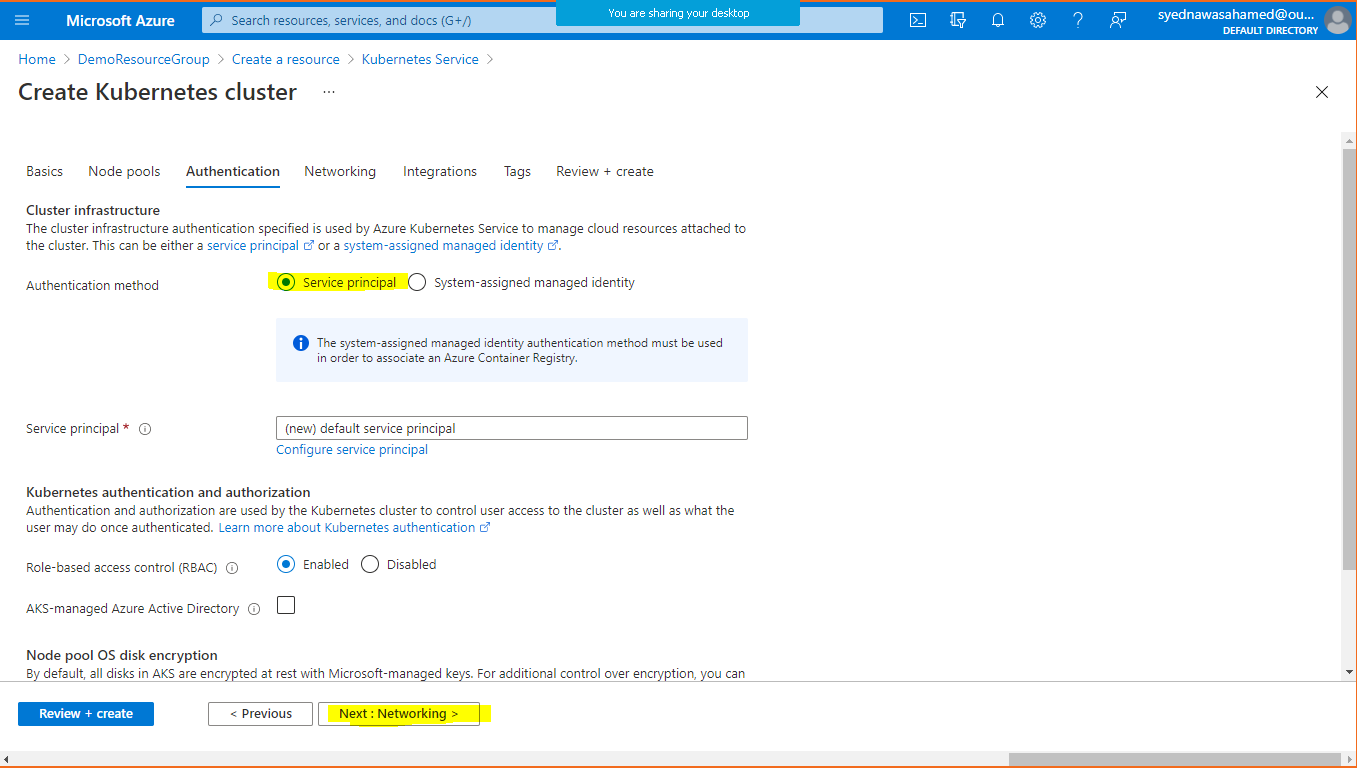
In this **Kubenetes Service** Page, Click Create Button,



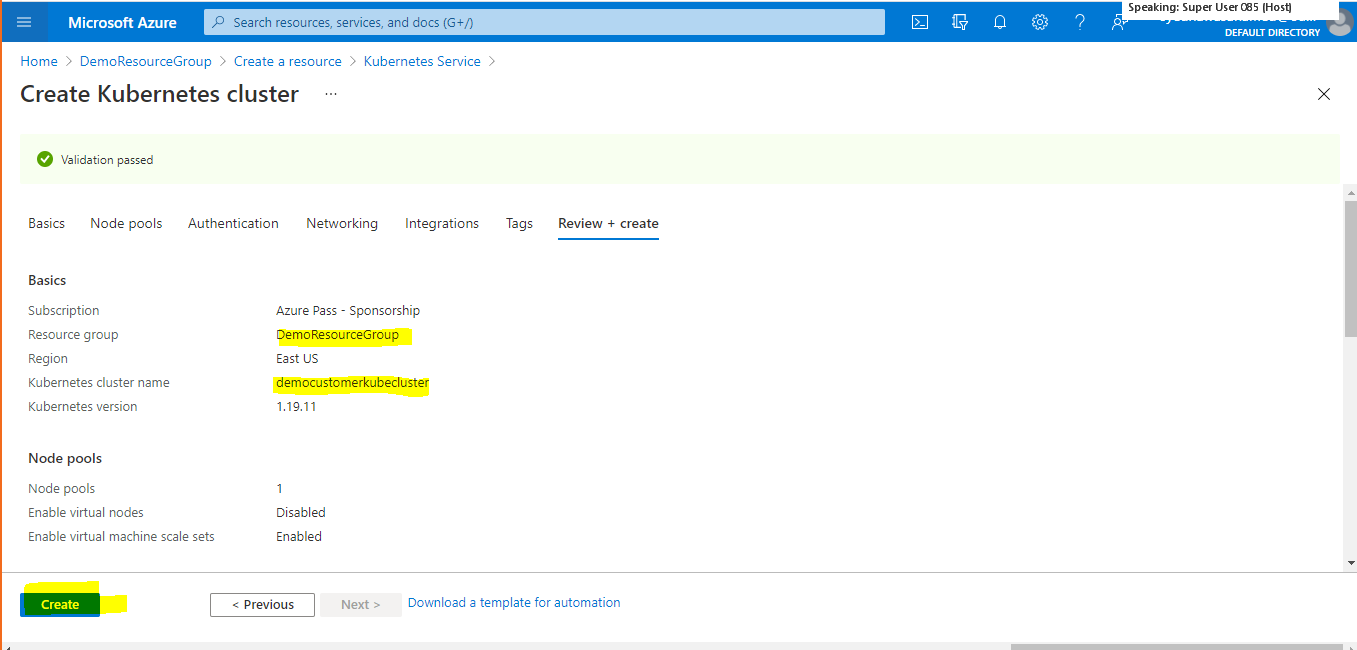
In this following page, choose the Resource Group and Give the Kubernetes Cluster Name as below,



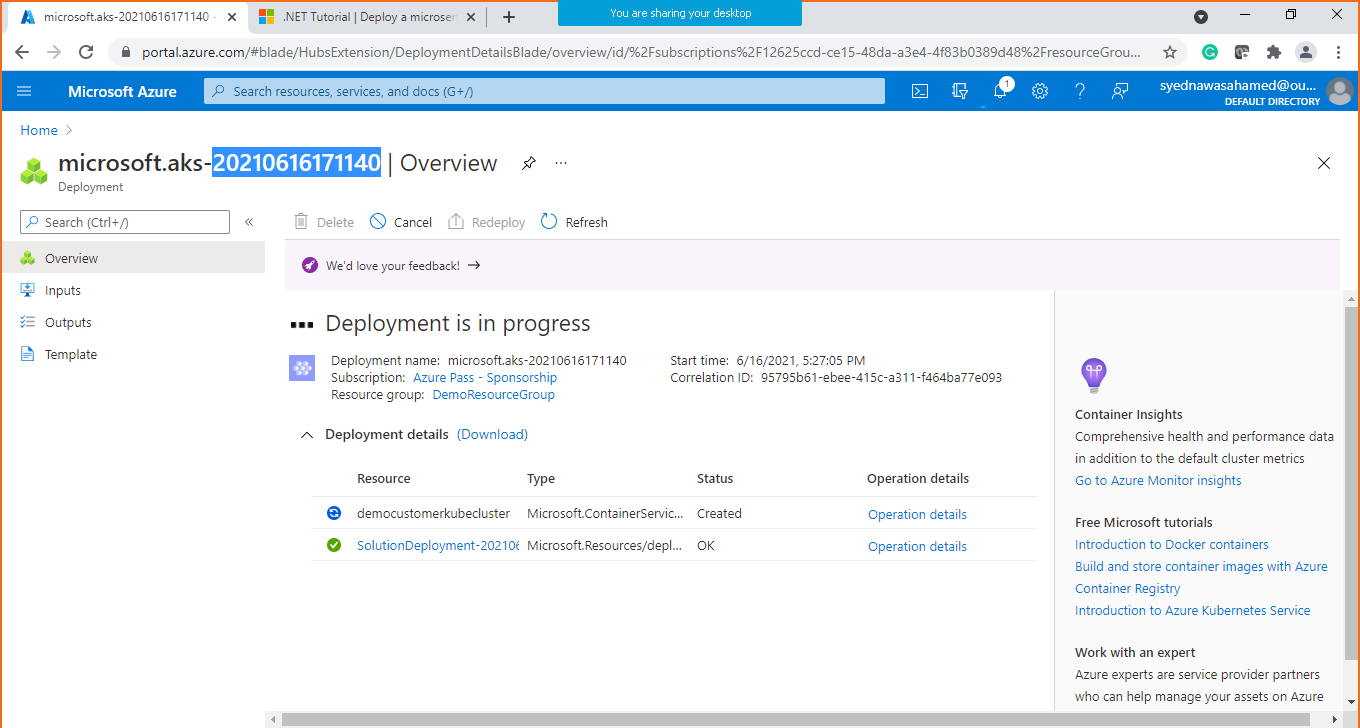
Navigate to **Authentication** tab, Select **Service Principal** authentication.



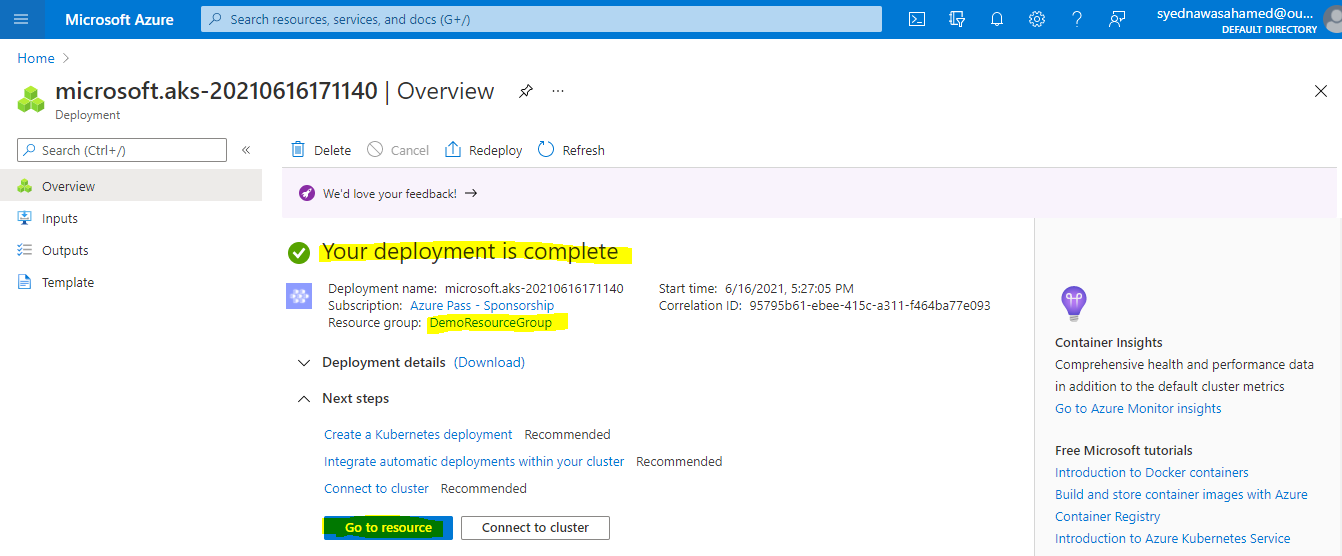
Skip other “Networking”, “Integrations”, “Tags” tabs and Navigate to **Review + Create** to do final review.



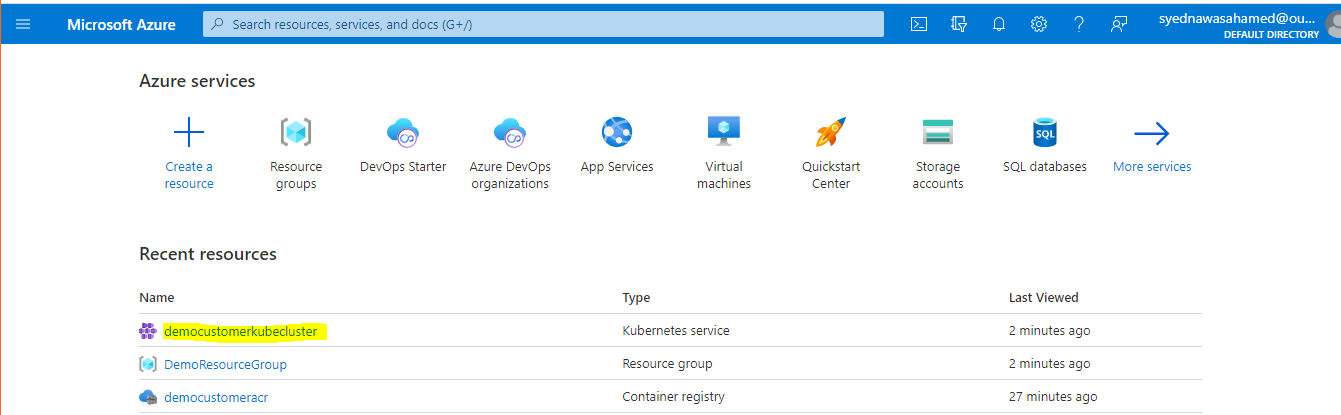
Click **Create** button. Azure will start new **Kubernates cluster deployment** process as follows,



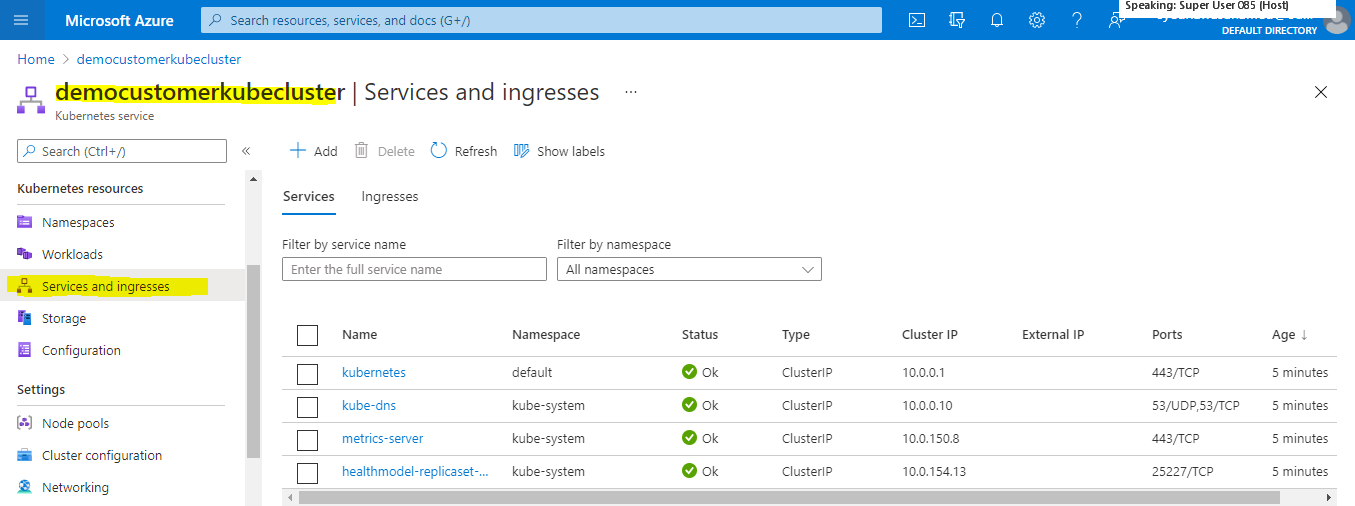
This asynchronous process will take some time around 3 to 4 minutes approximately. We will get following when finished.



Now we have all resources to deploy our microservice over azure kubernetes as listed below,

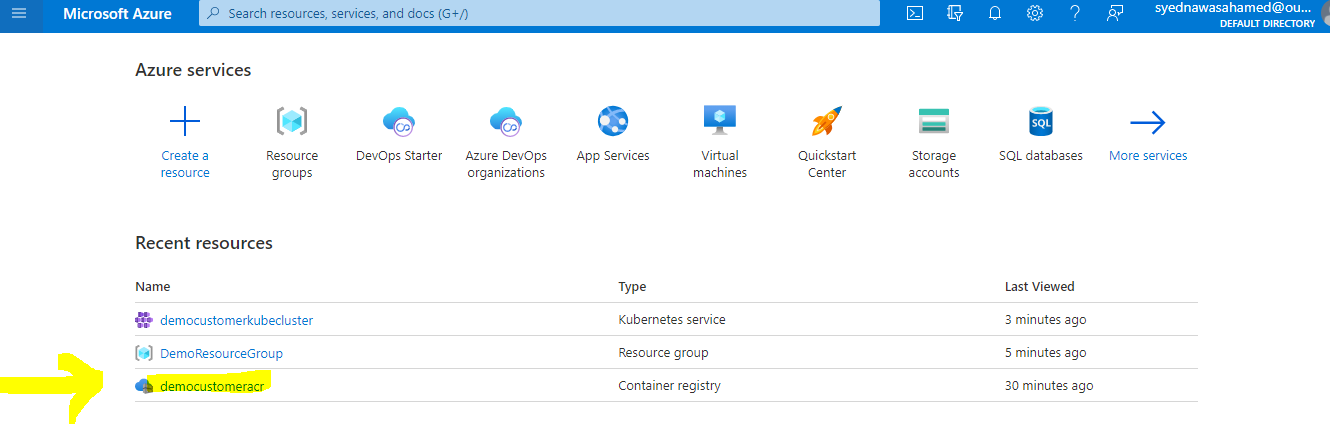


Now Navigate into democustomerkubecluster, Services and ingresses option in left side pane, will show initial namespaces created on new kubernetes cluster. Once we deploy our microservice, here that will be added as on namespace.

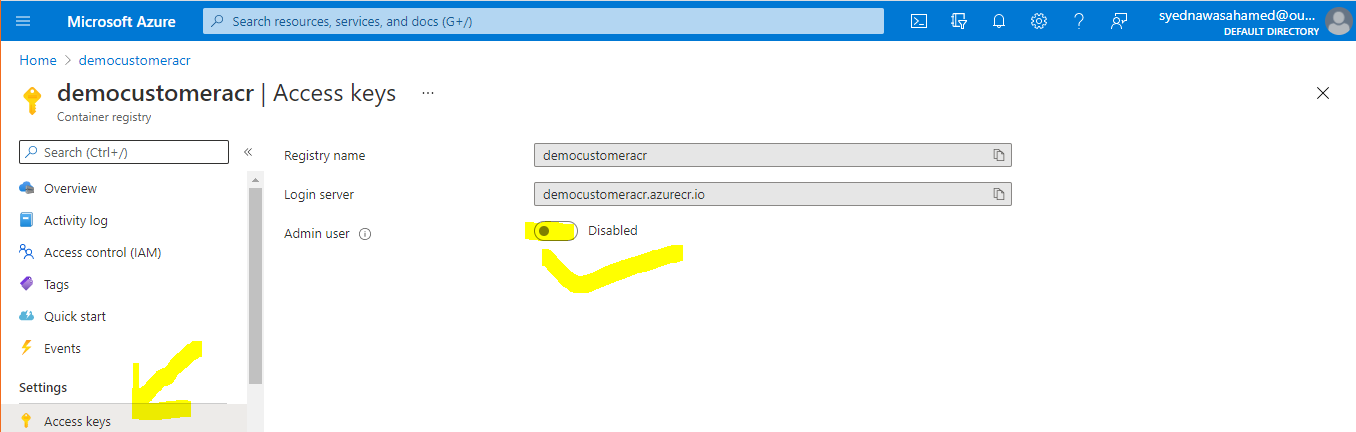


**Stage 6: Enable access keys to Azure Container Registry (ACR)**

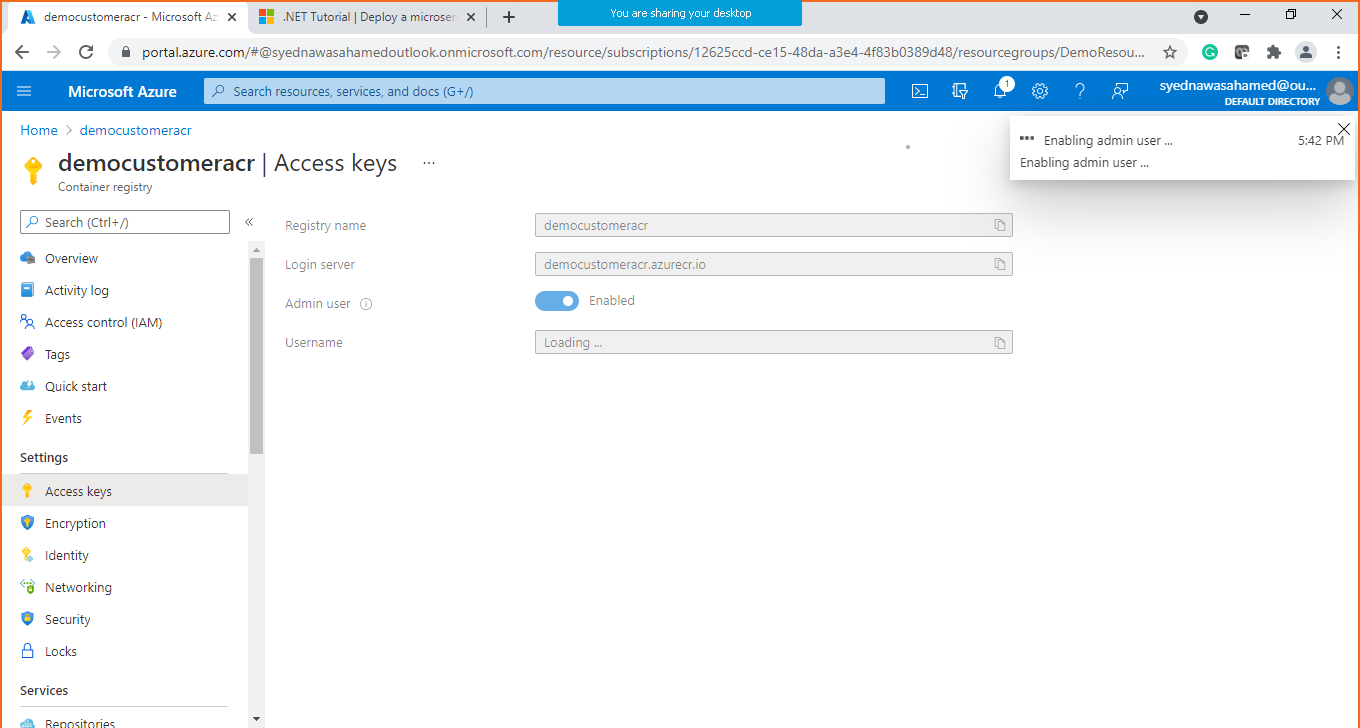
Navigate into Azure Container Registry



Select **Access keys** in Left side pane, Turn on the **Admin User** – by switching from Disable to Enabled



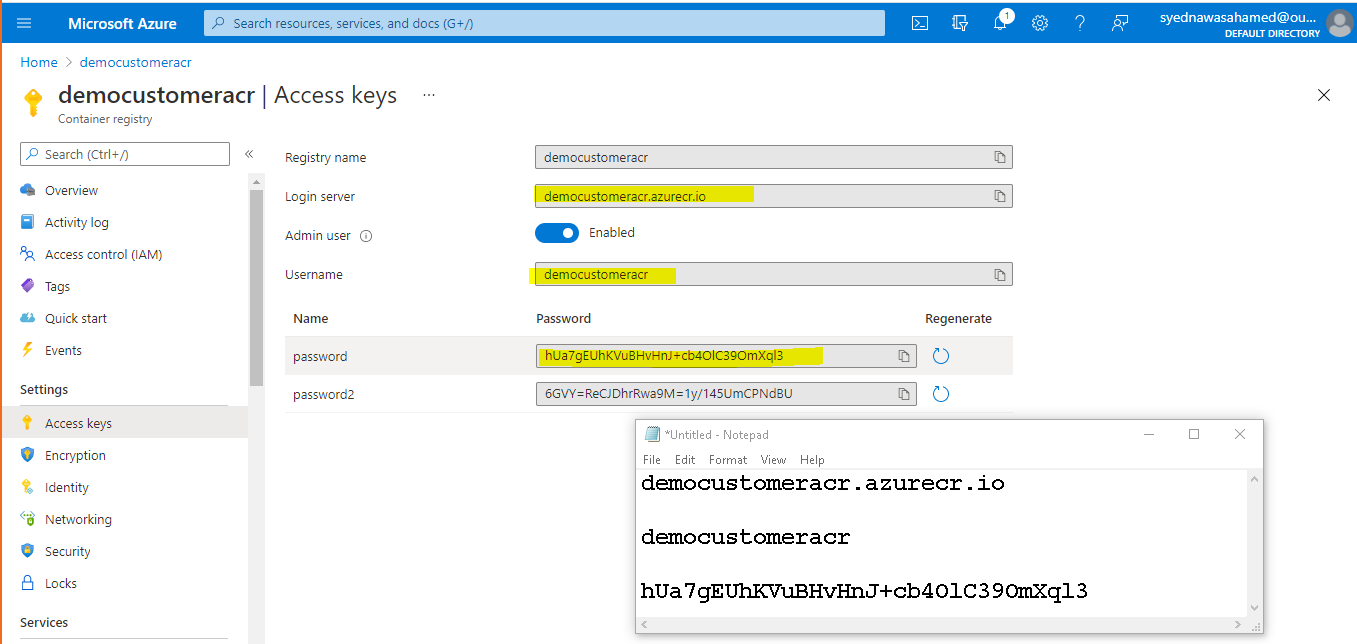
This will enable “Admin user” access to our container registry as follows,



If you enable **Admin user,** Username, password, password2 will be generated for current container registry.

**Hint:** Just note down the following in notepad, It is required to create **container secret key** in next stage

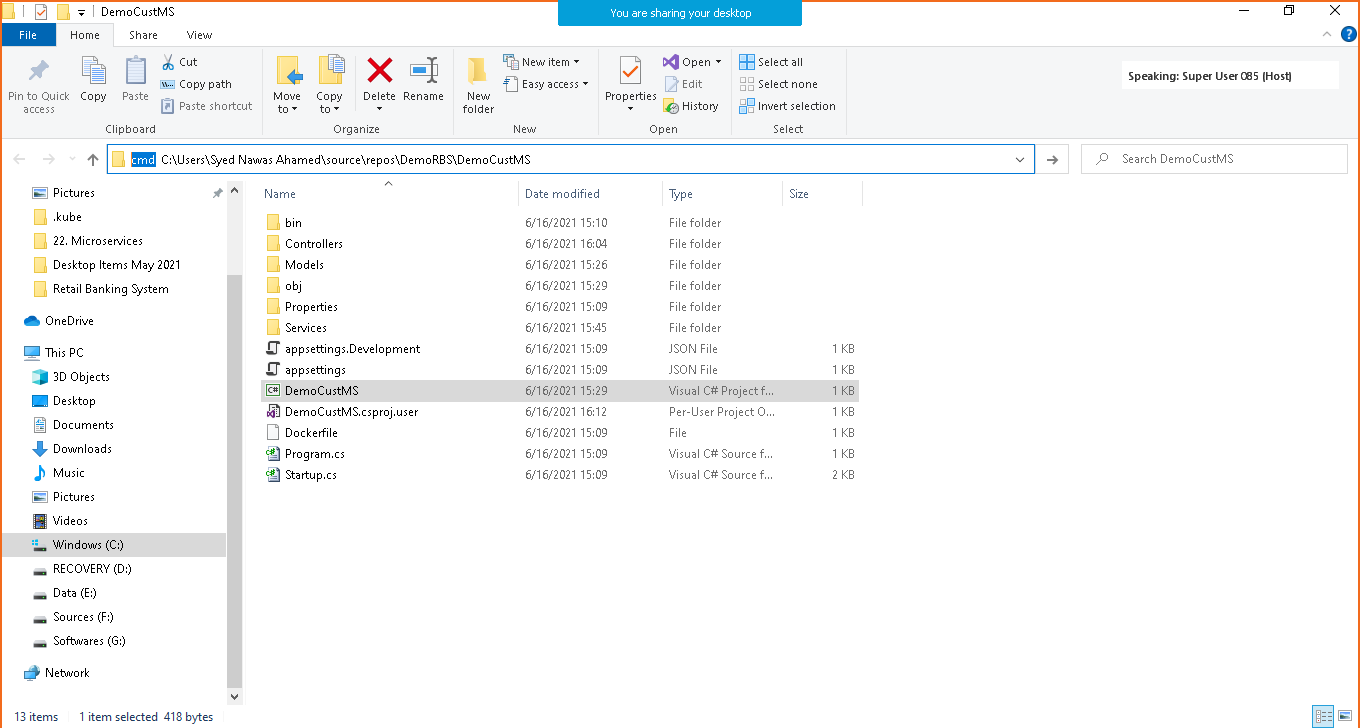
* Login server
* Username
* Password



**Chapter 3 – Kubernetes Deployment process**

**Stage 7 – Login into Azure Container Registry from command window**

Come back to Development Environment and Open Command window on our project working location. For the next steps we are going to proceed now, It is best to open the command window on working directory, just navigate into your project location in file Explorer, type **cmd** on file explorer’s address bar and hit enter.

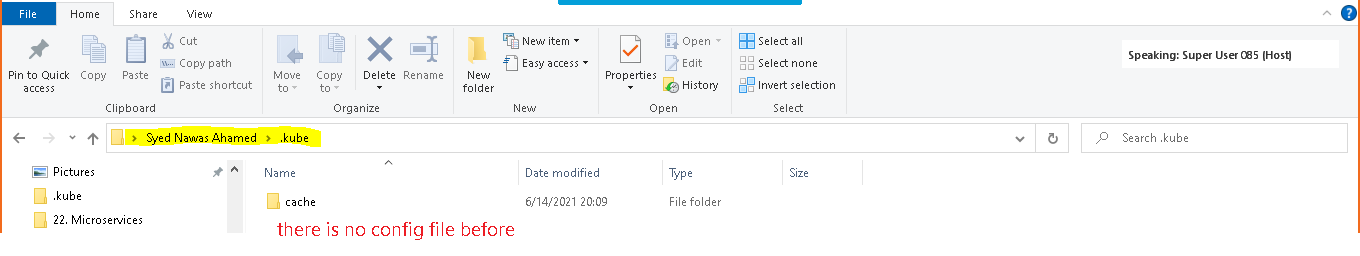


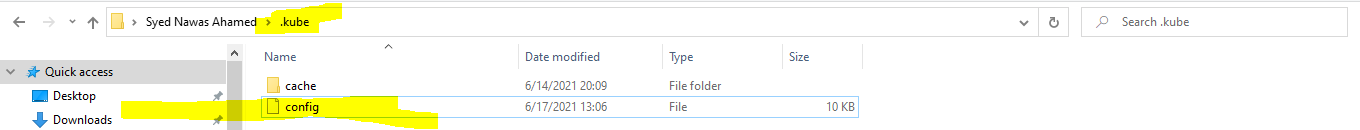
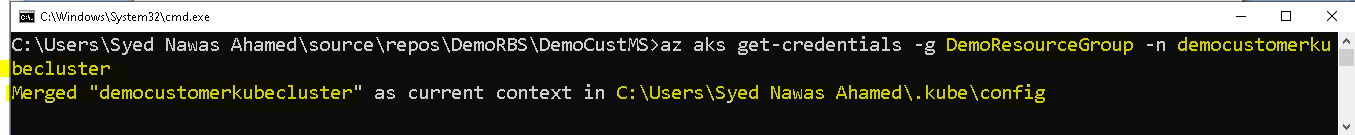
Write azure CLI command to login into Azure container Registry as follows and confirm that you are able to access azure container registry.



**Stage 8 – Get Azure Kubenetes Cluster Credentials into .Kube\config file from command window**

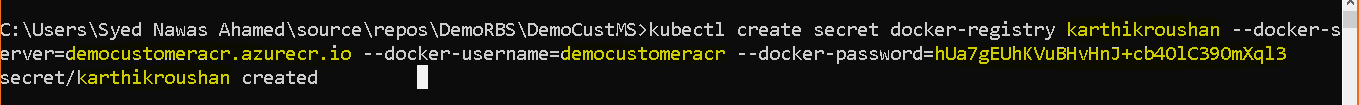
When we retrieve aks credentials, the config file will be created inside the **.Kube** folder of system user. Before running **get-credentials** command first time, there will not be any files inside c:\users\username**\.kube** folder.





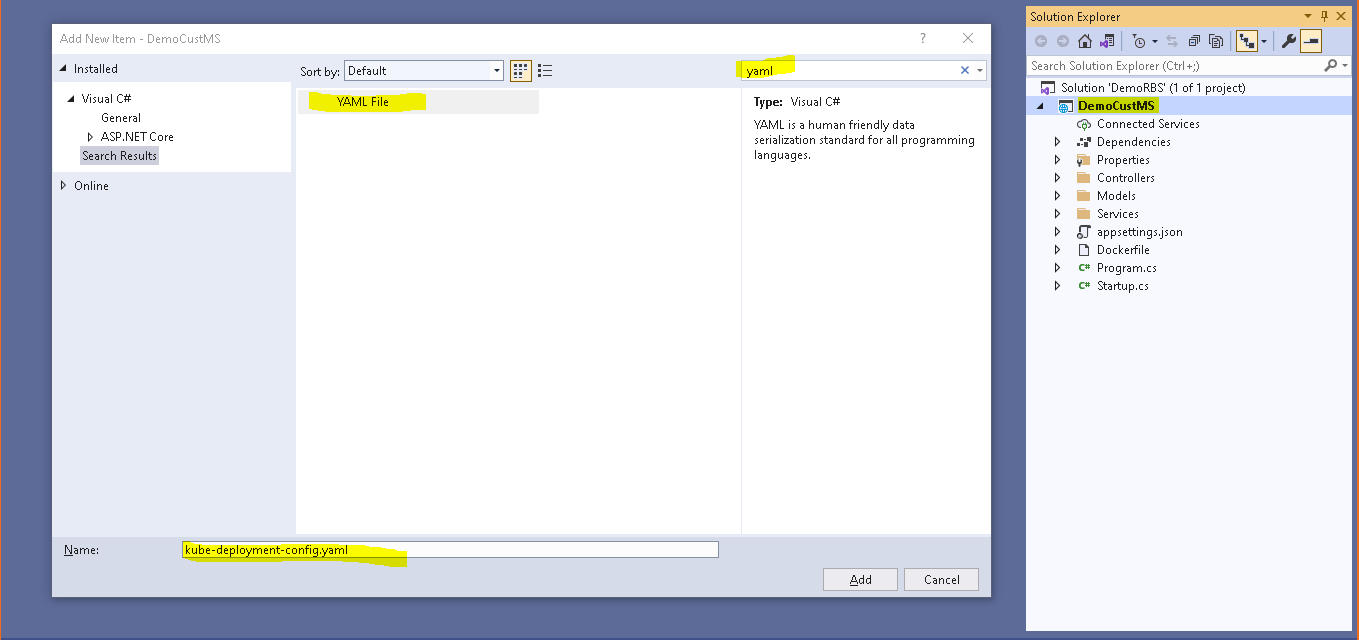
**Stage 9 – Create Secret Key to docker container registry and Publish the Micro- service**

We need to create **secret-key** for docker container registry, that key will be mentioned in **yaml configuration** **file** we are creating to deploy our web api application image into kubernetes as well as the key will be used between docker registry and kubenetes cluster to handshake communications. Issue the following command in azure CLI.



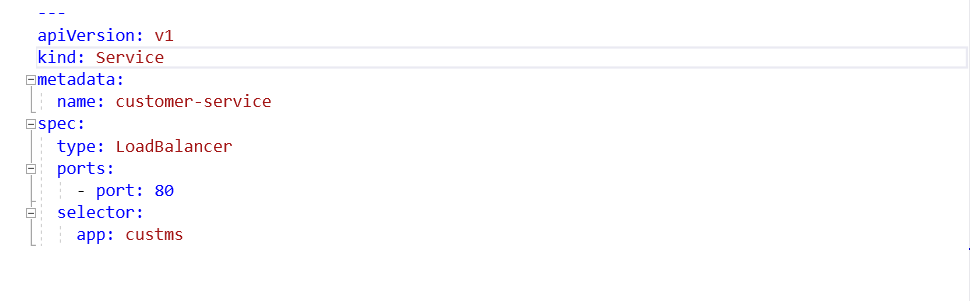
**Stage 10 – Yaml configuration file creation to deploy Micro- service in to kubernetes**

After creating **secret-key** successfully, Add a new **yaml configuration file** in your DemoCustMS Project and customize the yaml file with following code and also mention the **secret key** generated in previous step.



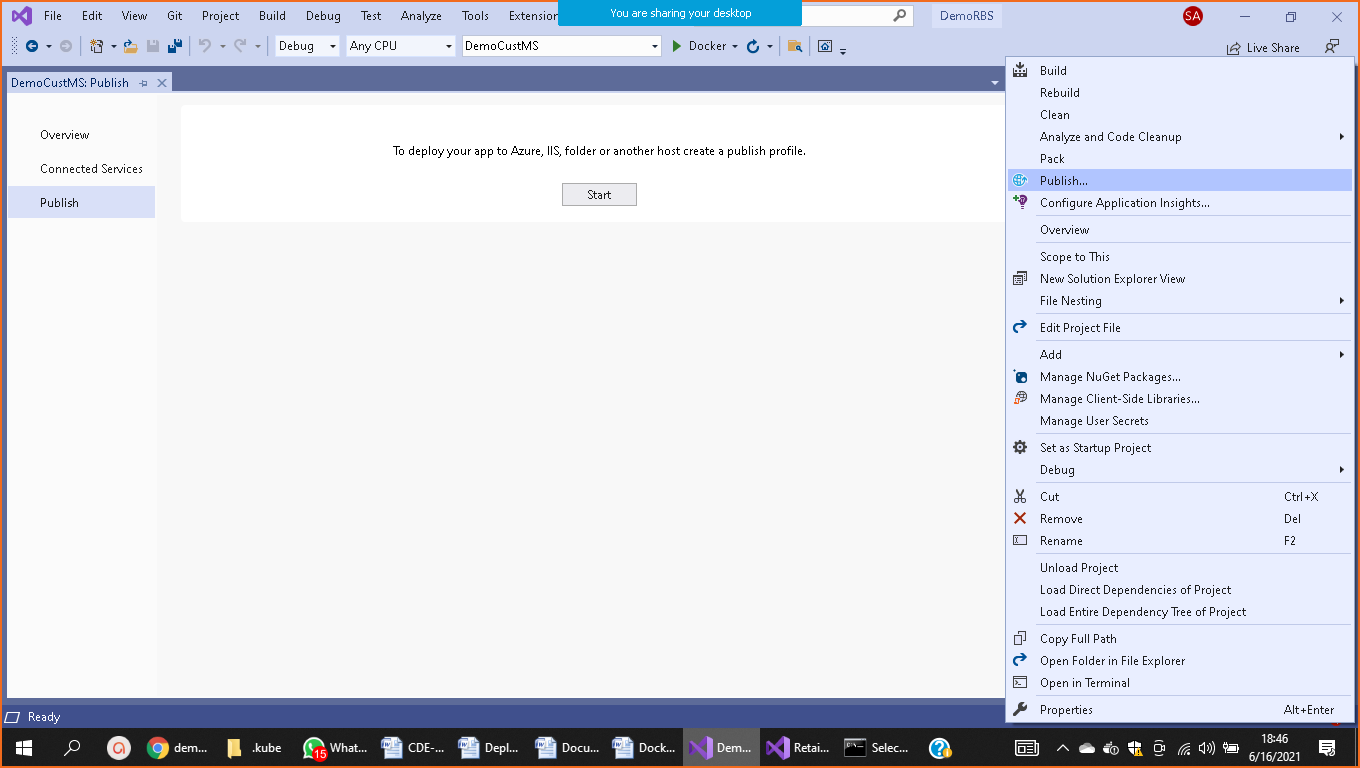
Give any name to configuration file. Here I have given **kube-deployment-conig.yaml** Add the following contents into yaml file you have just created.

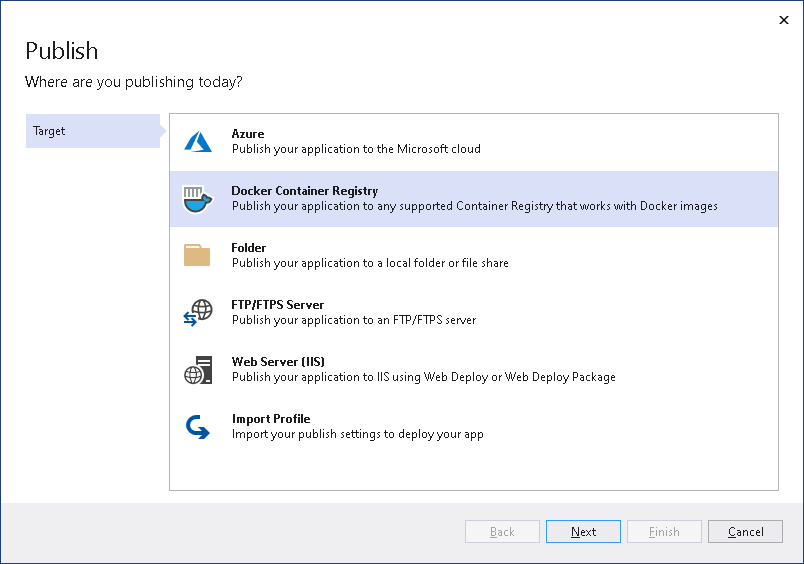




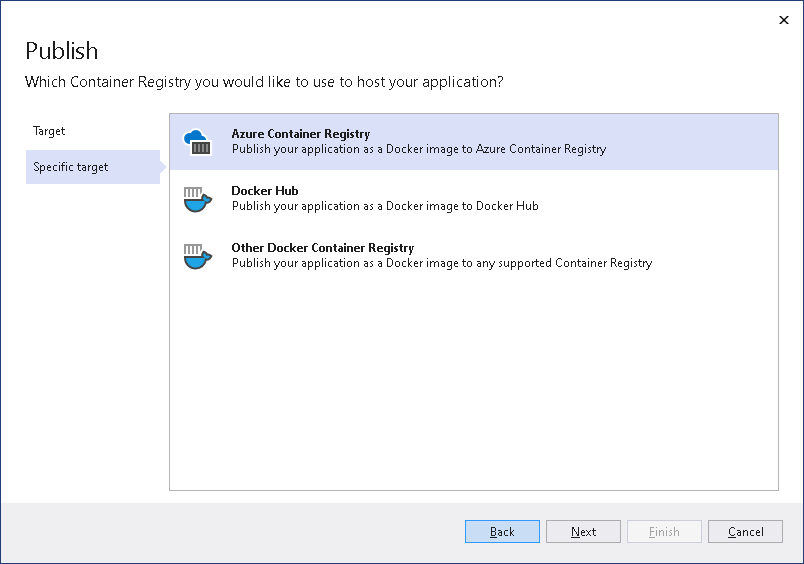
**Stage 11 - Publishing the Micro-service**

1. To publish our Micro-service, Choose **Publish** option from solution explorer.

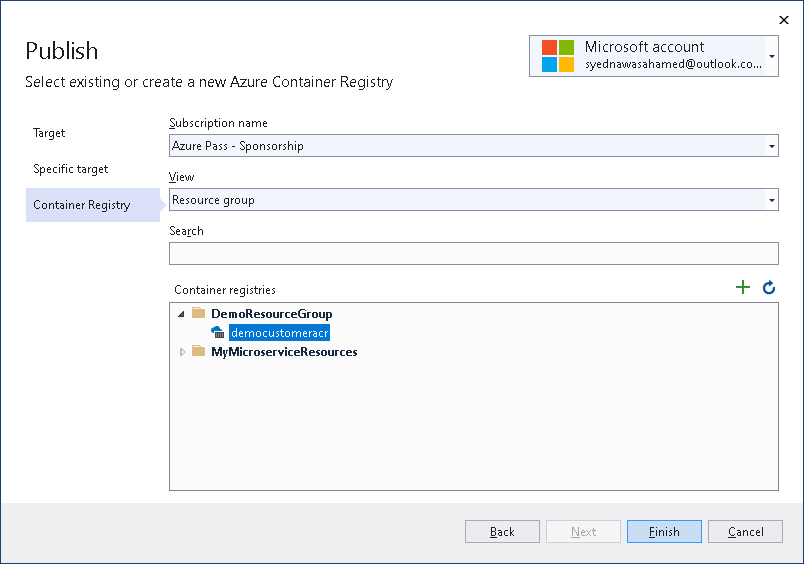
 2. Click Start and Select **Docker Container Registry** as **Target**.



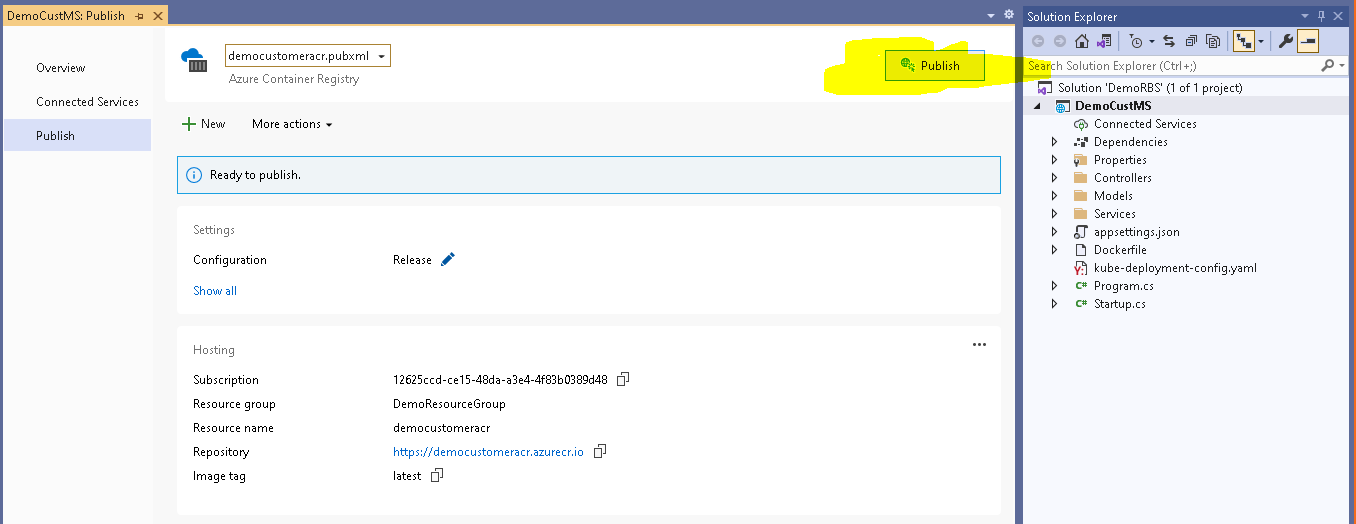
3. Select **Azure container Registry** as **Specific target**

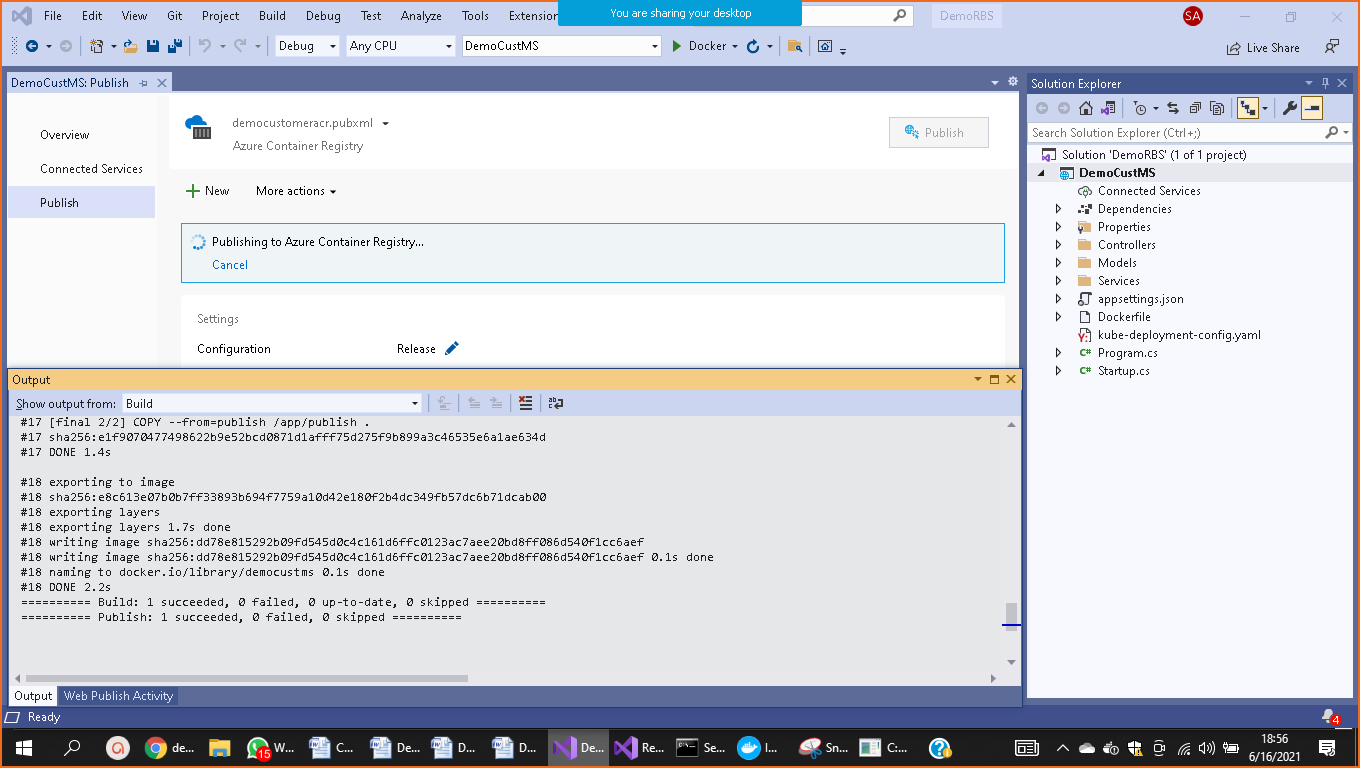


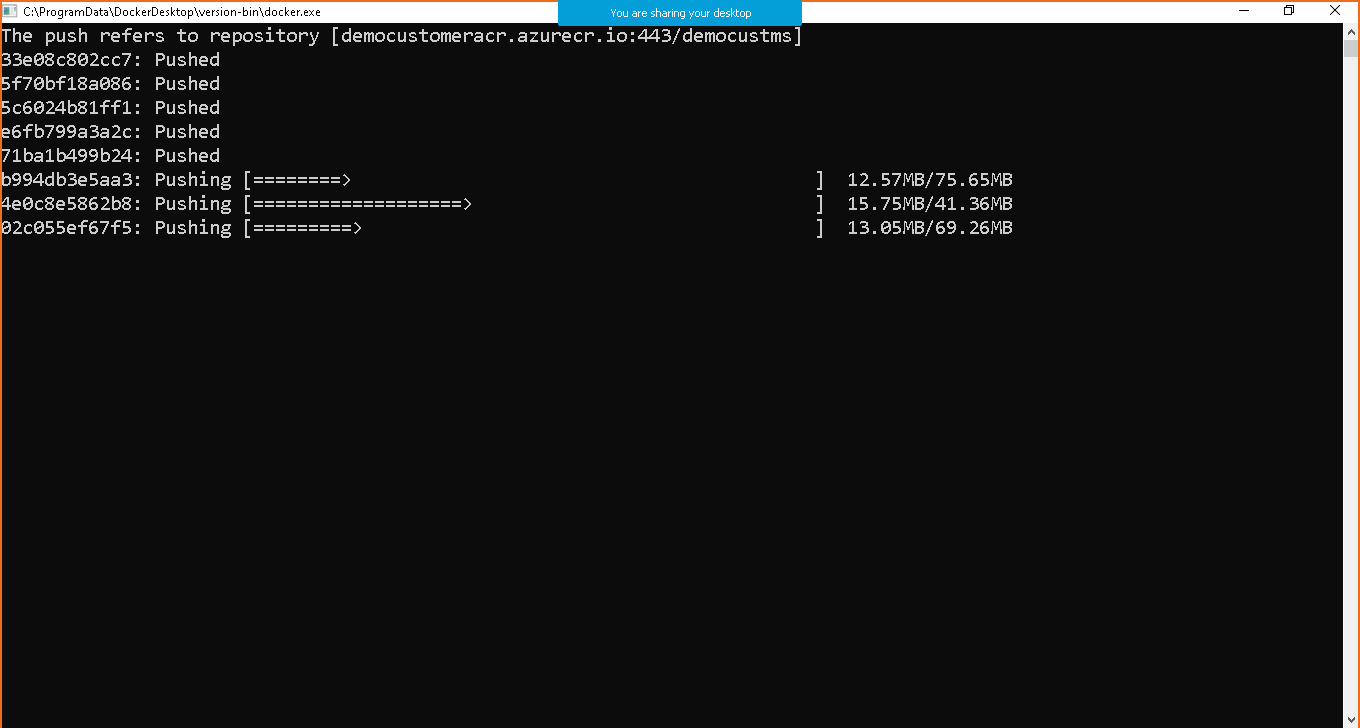
4. Select the azure subscription, resource group and container registry we created inside the resource group.

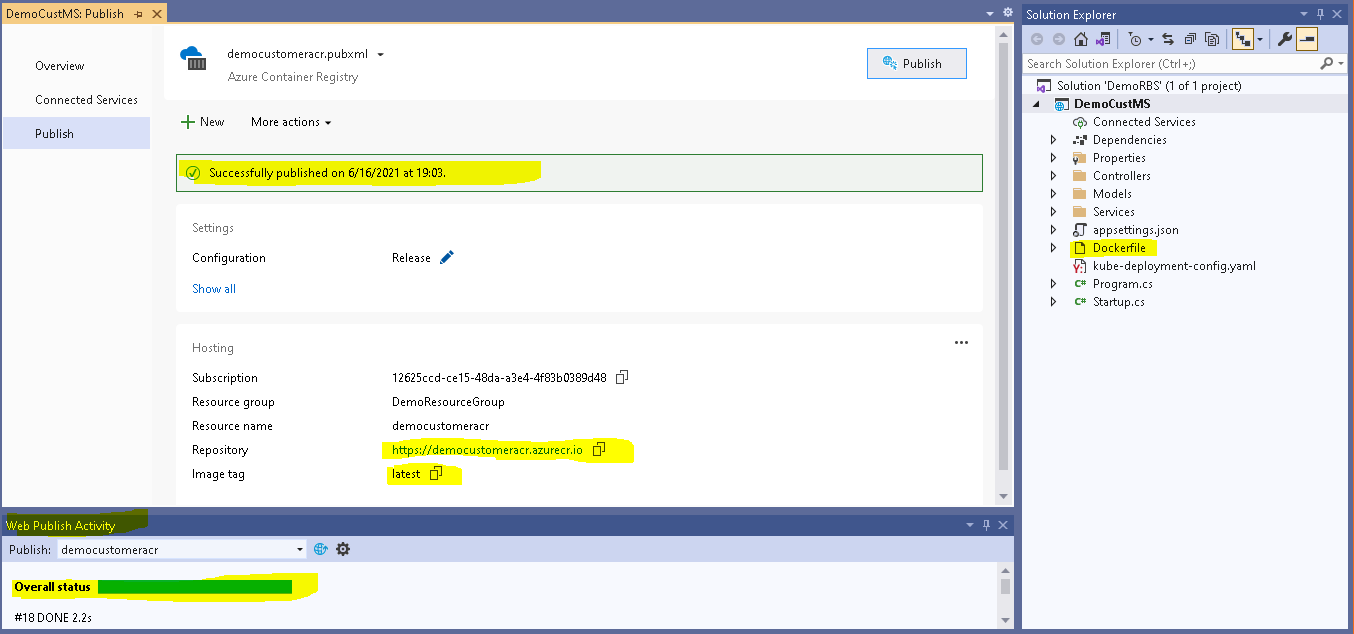


5. Click Publish button to initiate publish operations

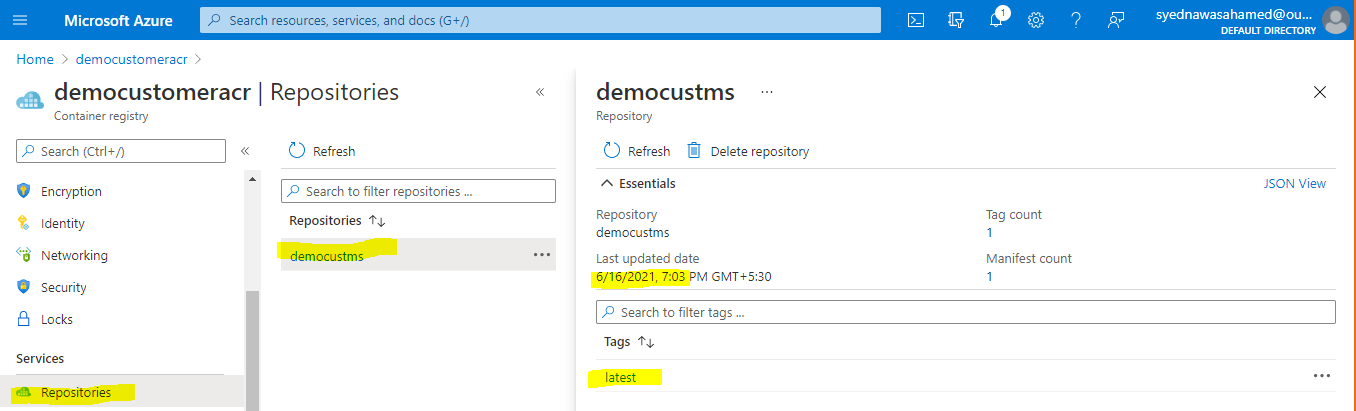








After **publish** process completed with above screens, Just we have to check whether our azure container registry added with our application repository image. Go to Azure Portal and Navigate into Container Registry, Click **Repositories**, See the repositories, and tag-name of image.



In General, repository will be created by our project name (democustms) and tag name will be latest.

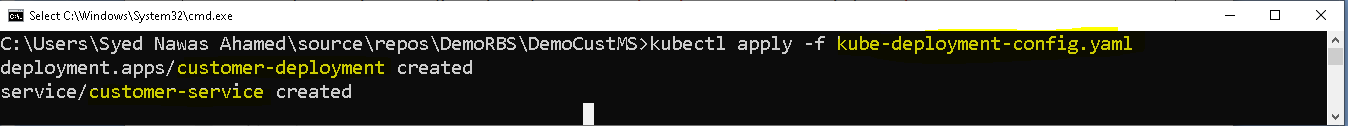
If tag name and repository name are different, then do some changes in **image** configuration of yaml file

Syntax of Image is Image**: containregistryname.azurecr.io/repositoryname:tagname**



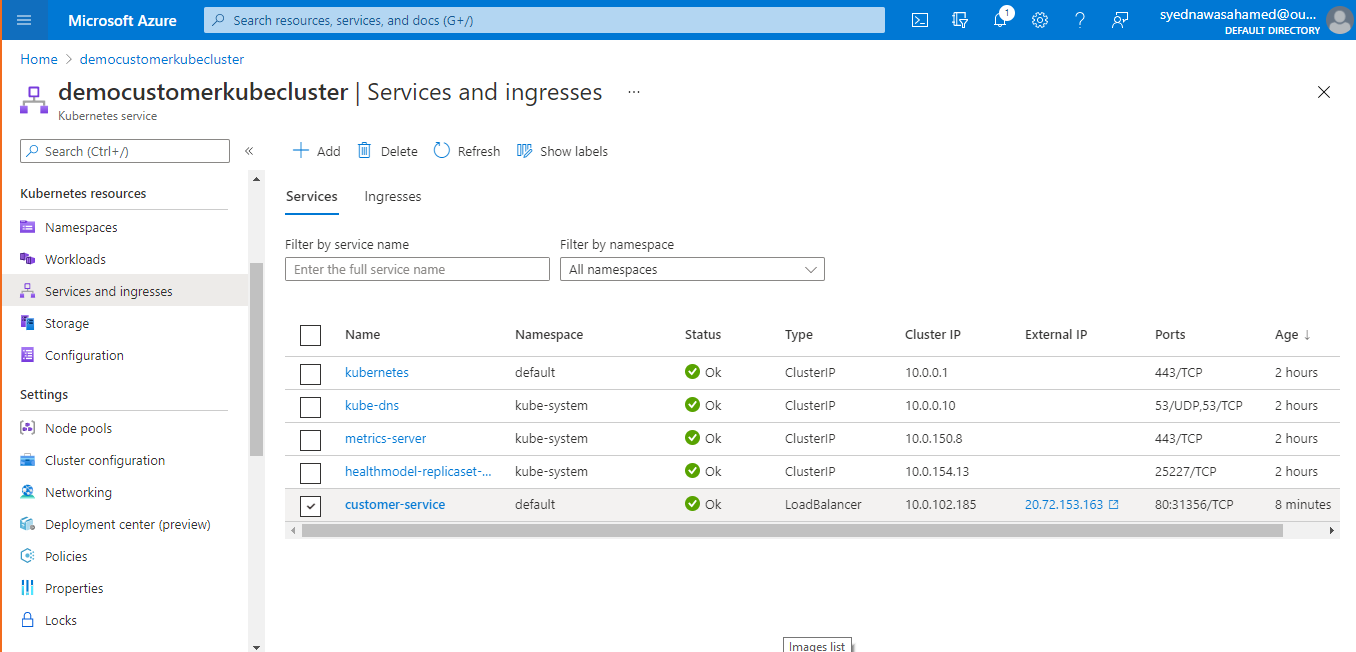
Stage 12 – Deployment in Kubernetes

Type the **kubectl apply command with yaml file as argument** on azure CLI to deploy container registry image in to kubenete cluster. This CLI will use the all the configuration of yaml file for the kubernetes deployment process.



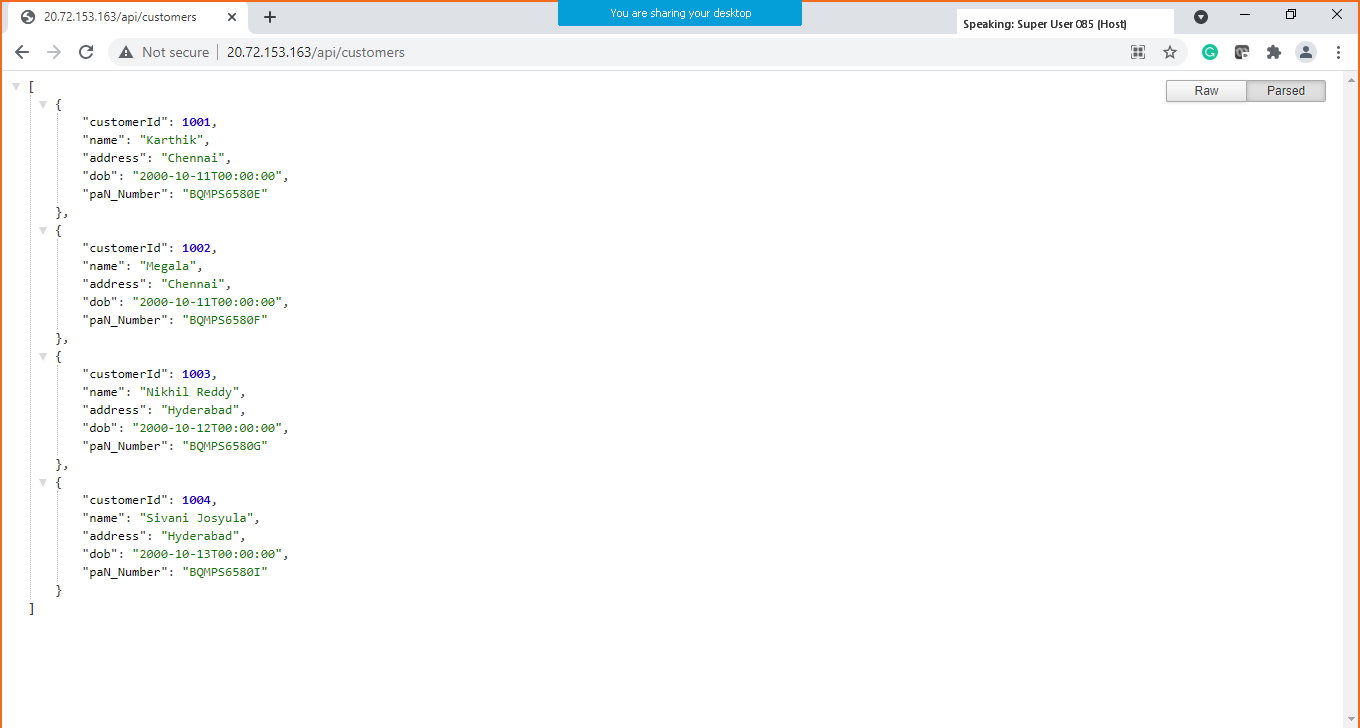
Here **customer-deployment**, **customer-service** are the meta names we gave on yaml file. Now check Azure AKS.

Stage 13 – Observe the External IP exposed by Kubernetes cluster

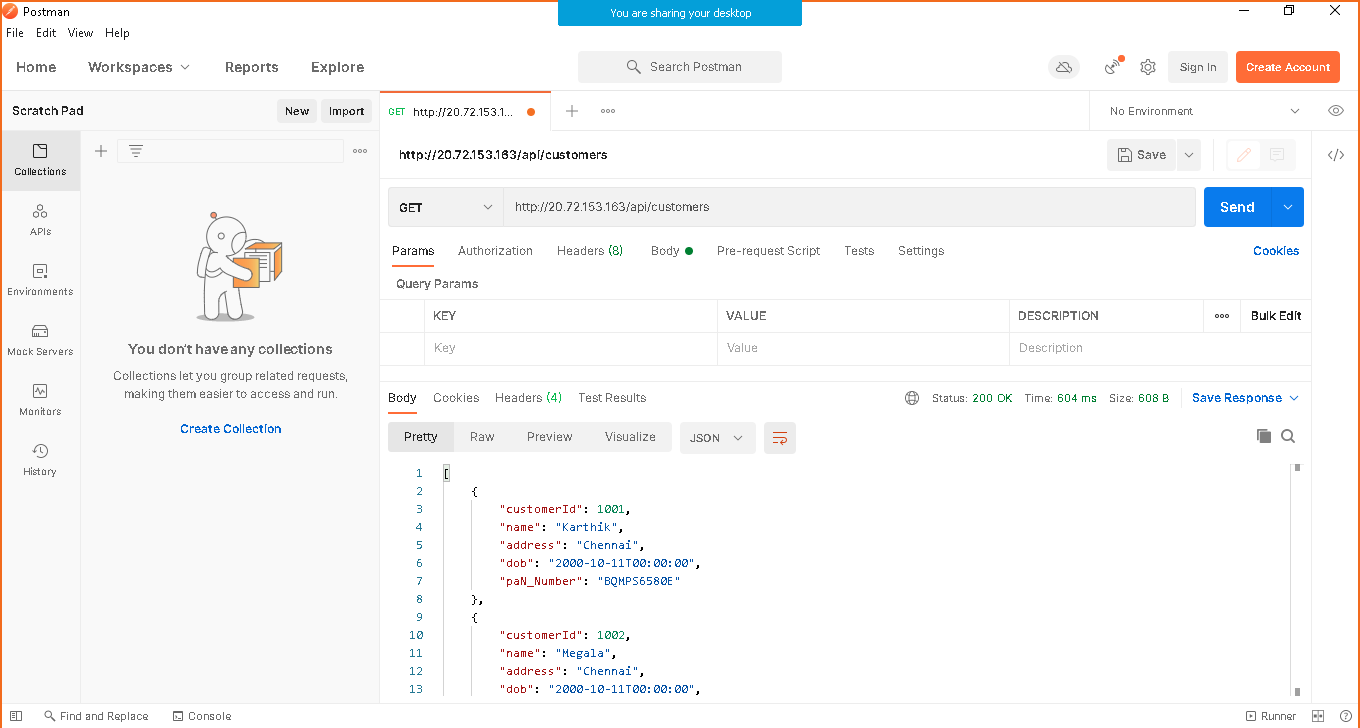


Use the **External IP** suffixed with **api/controller-name** to get end-points of deployed image of kubernetes cluster.

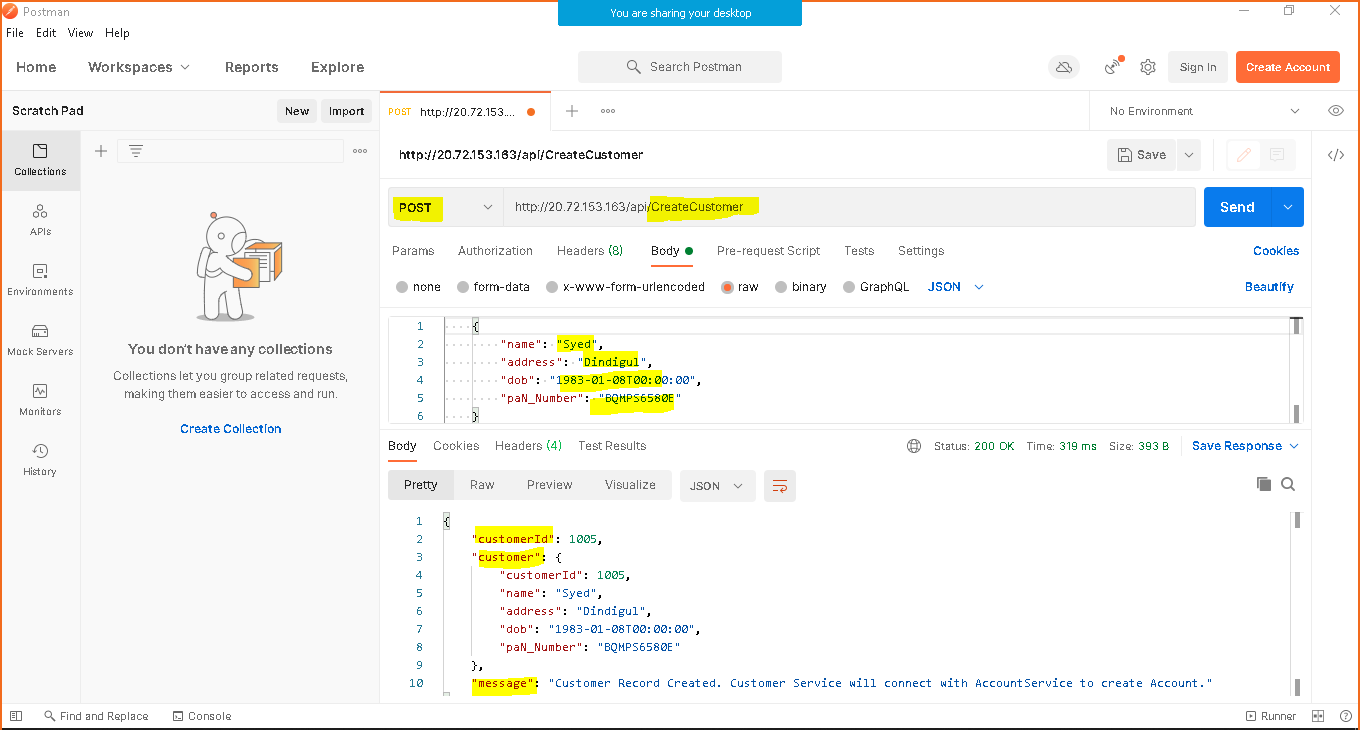
**Test Get Action**



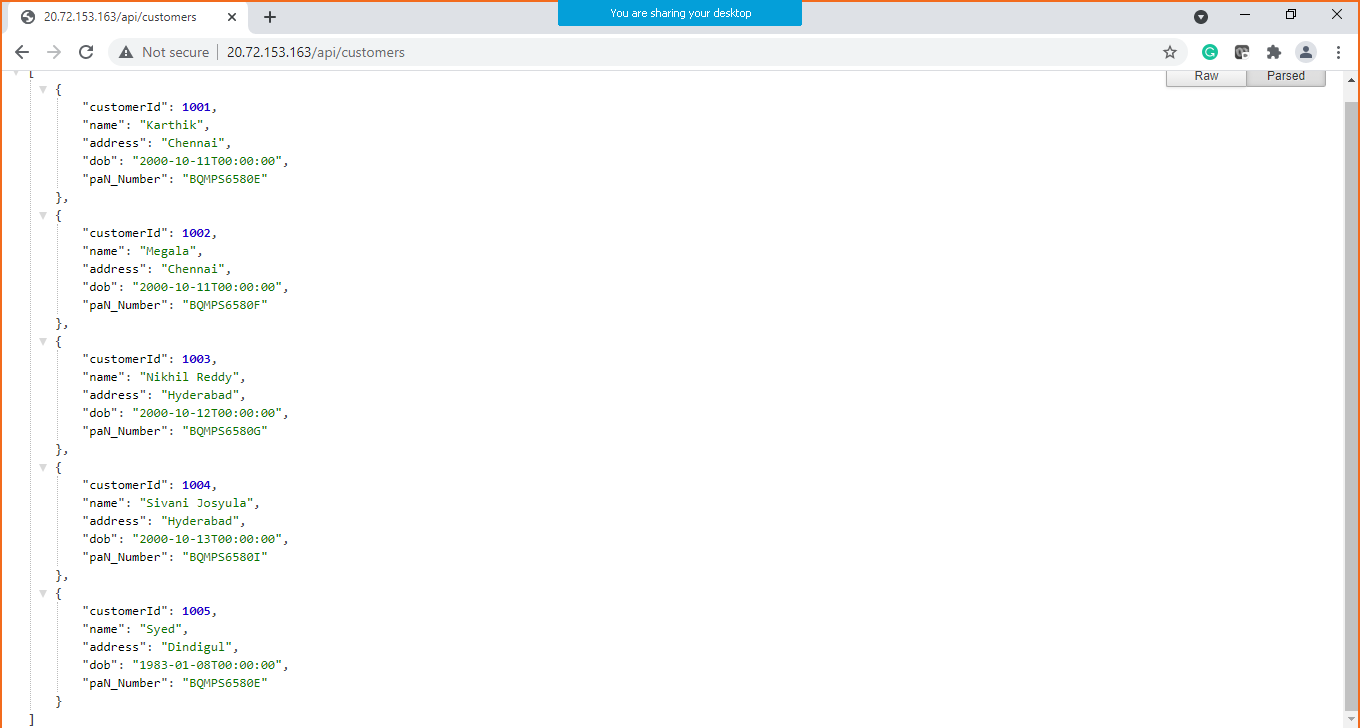
**Test GET using POST Action**

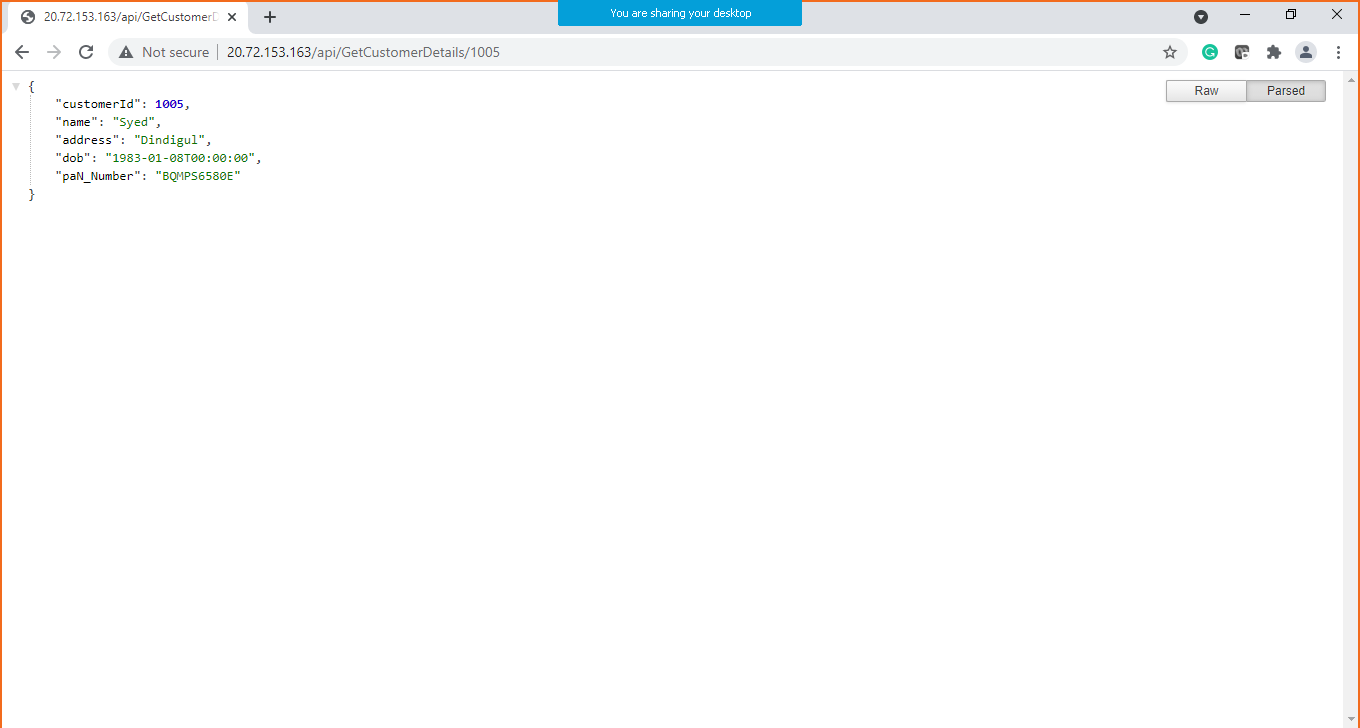


**Test POST using Postman**

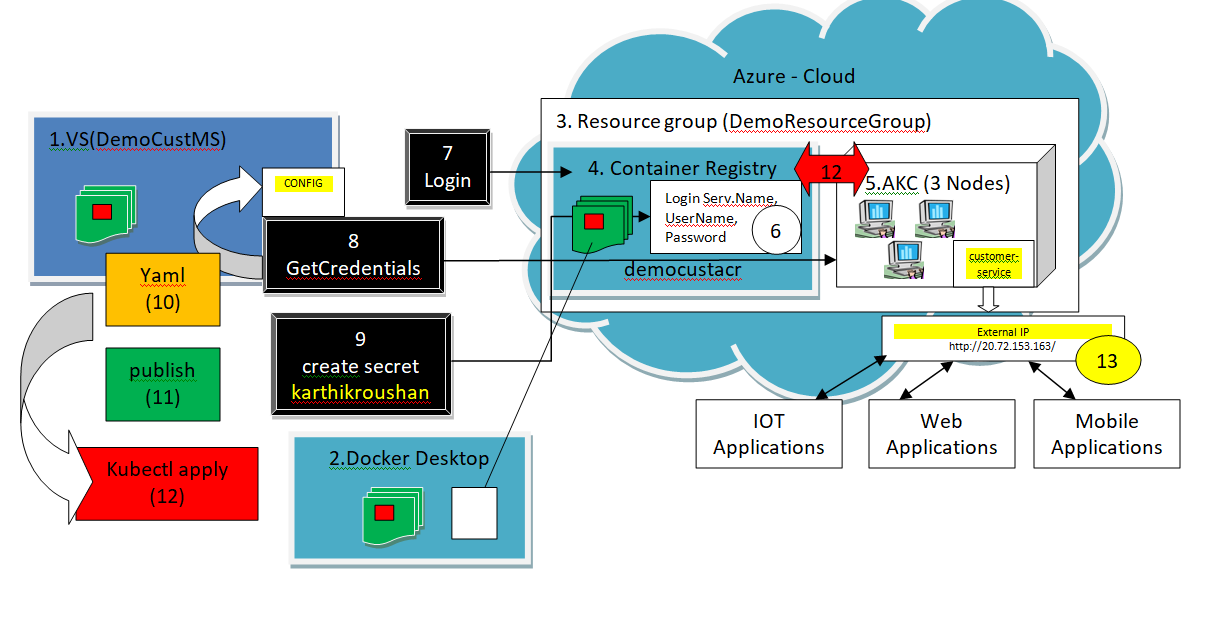


Then record you posted to AKS cluster can be reflected everywhere, just check below,





Summary of All Actions numbered from 1 to 13



**Disclaimer:** This Execution Plan diagram was prepared for understanding the concept in training environment only.

For more technical details please visit official websites of Microsoft Azure and Kubernetes.

Prepared by: **Syed Nawas Ahamed**, Microsoft .NET Technical Trainer.

[syednawashamed@gmail.com](mailto:syednawashamed@gmail.com) +91-9842502235, +91-9787810104.

<https://syed-nawas-codes.blogspot.com/>