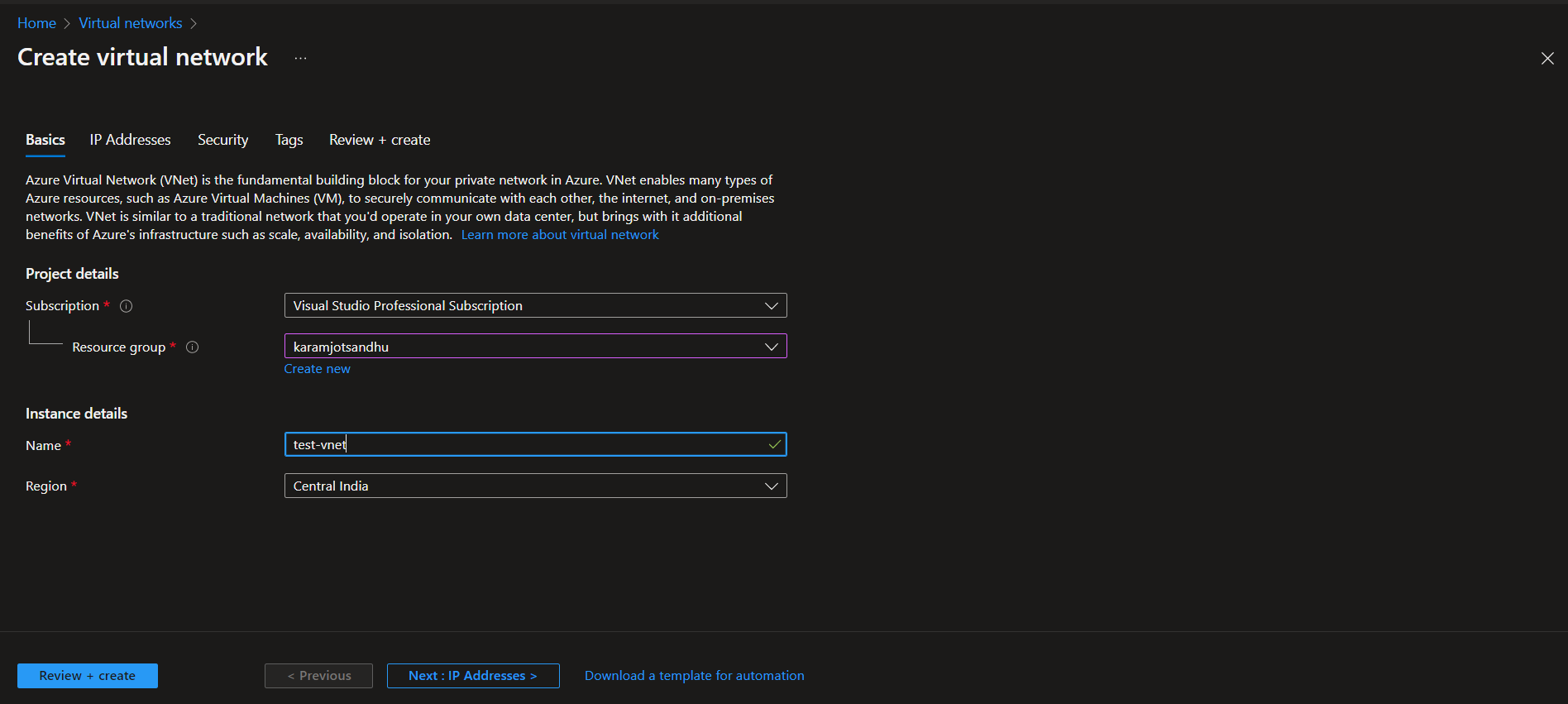
**Cloud Assignment by Karamjot Sandhu**

1. Create a virtual network with 2 subnets. Each subnet should have 16 Ips only.

* Step 1: Search Virtual Network in the search box, then click on Create button to create virtual network. Then, select subscription and Resource Group if not present create one, given a name to the virtual network and select region.



Step 2: Add the IPv4 address space and if you want to add subnet then click on the Add Subnet button.

Text

Description automatically generated

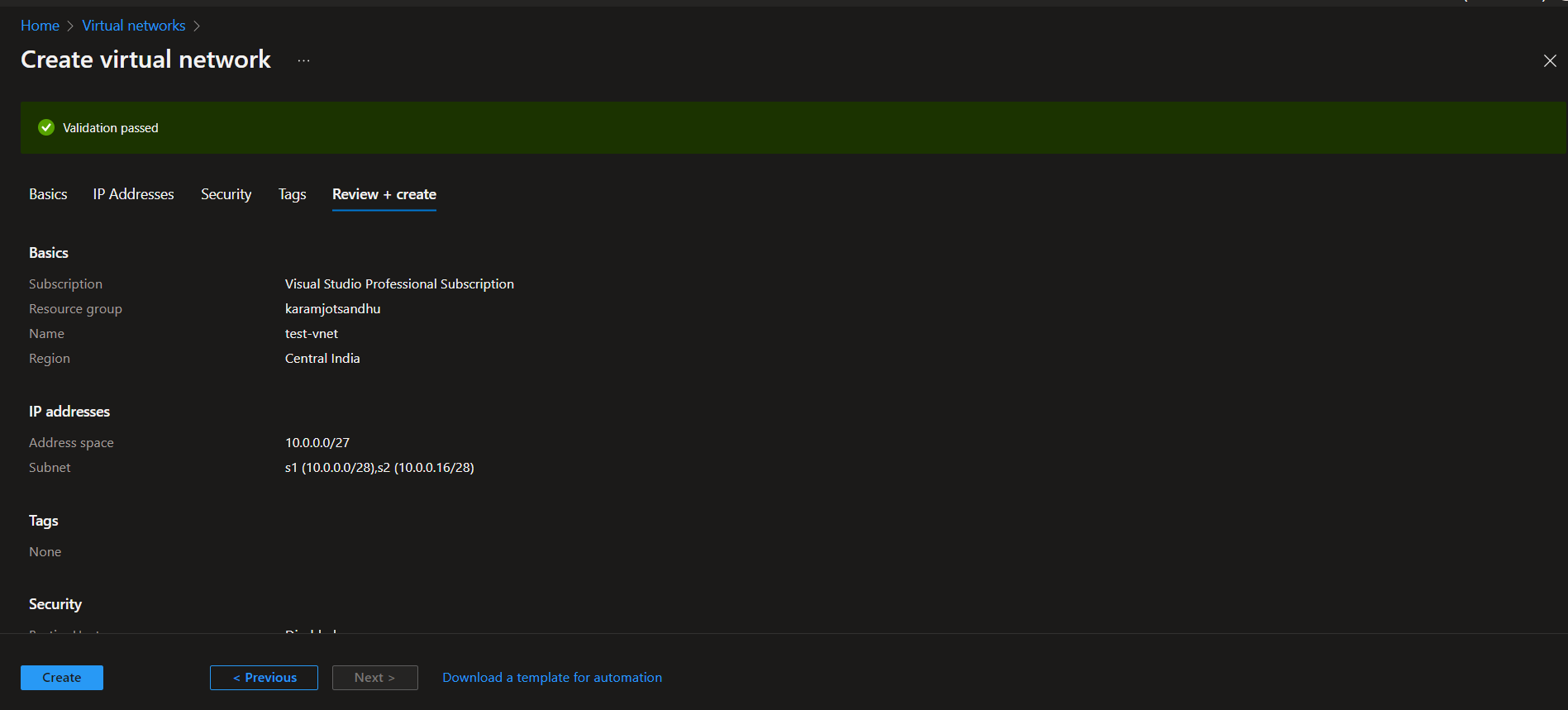
Step 3: Added two subnet each having 16lps only

A screenshot of a computer screen

Description automatically generated with medium confidenceA screenshot of a computer screen

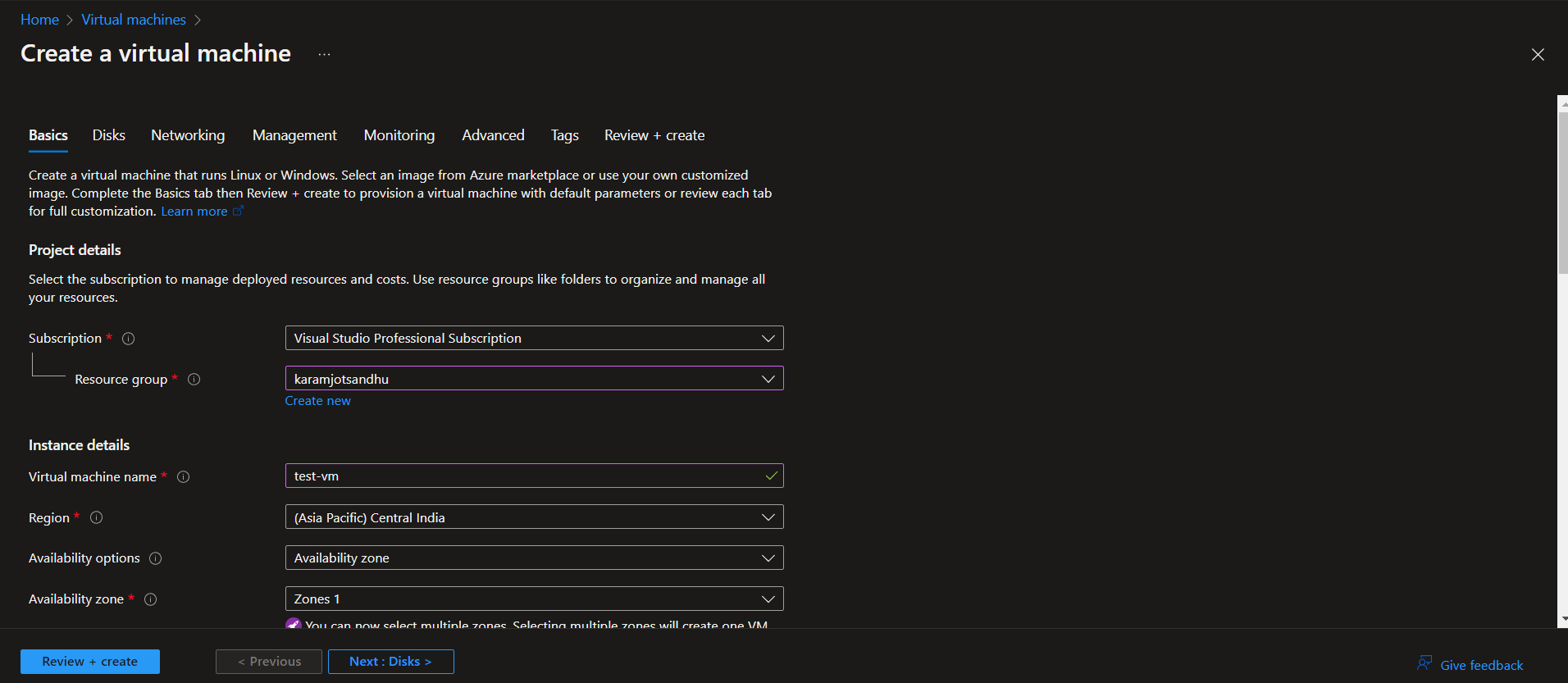
Description automatically generated with medium confidence

Step 4: Review the settings and click on the Create Button to create the virtual network.



1. Inside one of the subnets, create a VM and deploy any application code inside it and it should leverage the database on the cloud (any existing application created by you before).

* Step 1: Create a virtual machine in one of the subnets, then select the resource group and the subscription, then appropriate image.



Step 2: Select the appropriate disk space.

A screenshot of a computer

Description automatically generated with medium confidence

Step 3: Select the one subnet of the last created virtual network in question 1 .

A screenshot of a computer

Description automatically generated with medium confidence

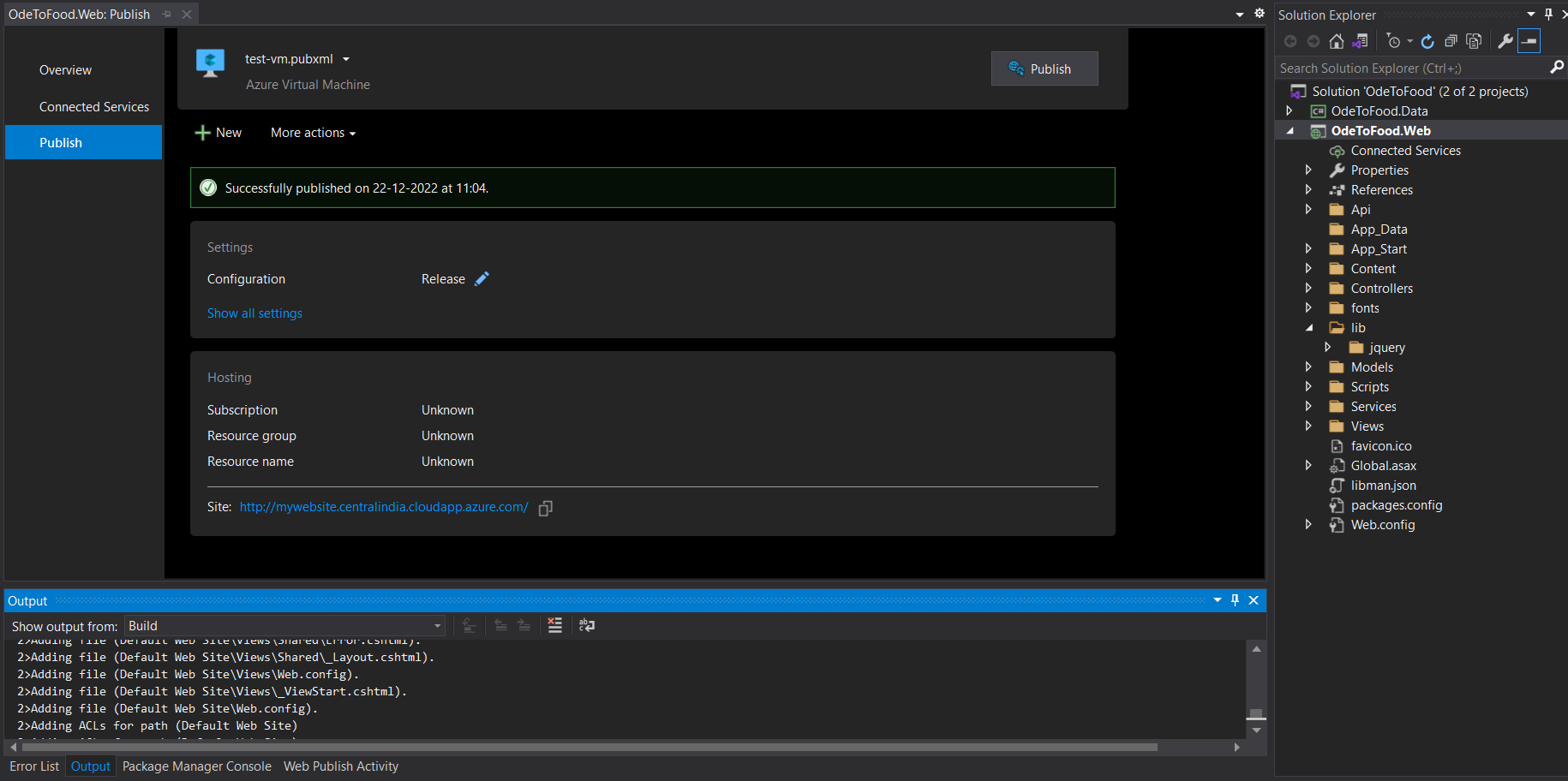
Step 4: Review and Create the virtual machine.

A screenshot of a computer

Description automatically generated with medium confidence

Step 5: Connect to the VM using RDP, then configure the IIS and install Web Deploy 3.6 for web applications.

Step 6: Open project in Visual Studio and right click the project, then click on Publish, then select Azure VM and then select the resource group and VM and finish. Then, click on the Publish button to publish the application



Step 7: Enter the destination URL to get the application up and running.

Graphical user interface, text, application

Description automatically generated

1. Deploy the same application to Azure App Service. It should also leverage the database on the cloud.

* Step 1: Search for Azure App Service, then click Create Web App, then select subscription and resource group, then name it and select publish options as code, then select runtime stack as ASP.NET V4.8, then operating system as windows, then select region and storage option according to the requirements.

A screenshot of a computer

Description automatically generated with medium confidence

Step 2: Review and create the web app, within few seconds your resource is ready to use.

A screenshot of a computer screen

Description automatically generated

Step 3: Open the MVC project in Visual Studio, then right click the project and select Publish, then select Azure and then select Azure App Service (Windows), then select the resource group and finish. Then click on the publish button to publish the application.

A screenshot of a computer

Description automatically generated

Step 4: Application is up and running on the specified URL by the Azure App Service.

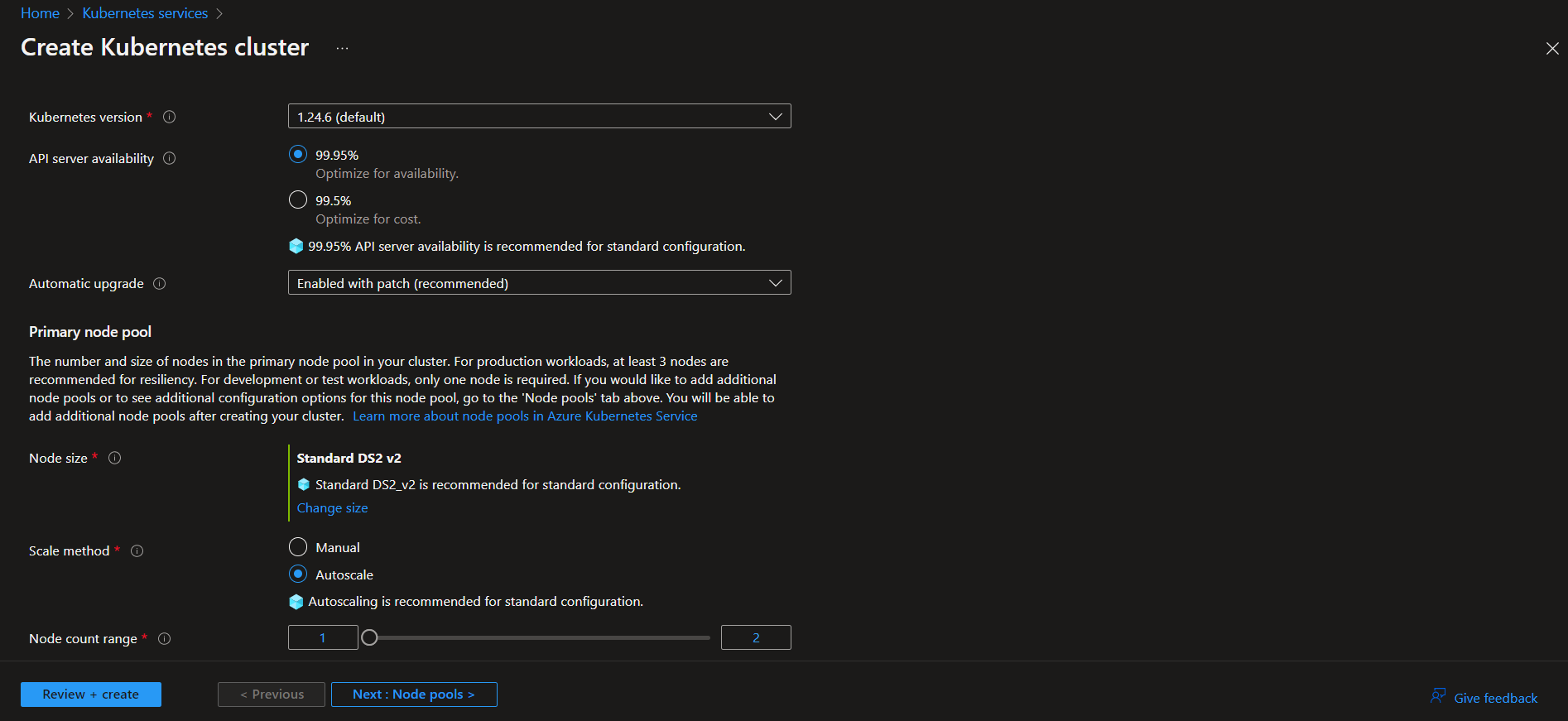
Graphical user interface, text, application

Description automatically generated

1. Create the AKS cluster (2 nodes, smallest size VM) and deploy any two services on it. Services should be accessible from the internet.

* Step 1: Search for Kubernetes Services, then click on the add button to create new Kubernetes Cluster and then select the resource group, name, Kubernetes version ,

node size and node count.



Step 2: Select the default authentication settings and proceed to the networking where select the Azure CNI, which will create the new VNet automatically.

Text

Description automatically generated

Step 3: Review and create the Azure Kubernetes Cluster.

A screenshot of a computer screen

Description automatically generated

Step 4: In the search bar, search for the Azure Container Registry, then create a new registry, by selecting the resource group and the size as standard. Then, open the Azure CLI PowerShell and run the command:

git clone https://github.com/jot11sandhu/simple-helloapi.git

This can clone the whole project with Docker file.

Graphical user interface, text

Description automatically generated

Step 5: Go into the folder which contains Docker file of the last project by running the command :

cd simple-helloapi/Hello/Hello

then run the command to build it:

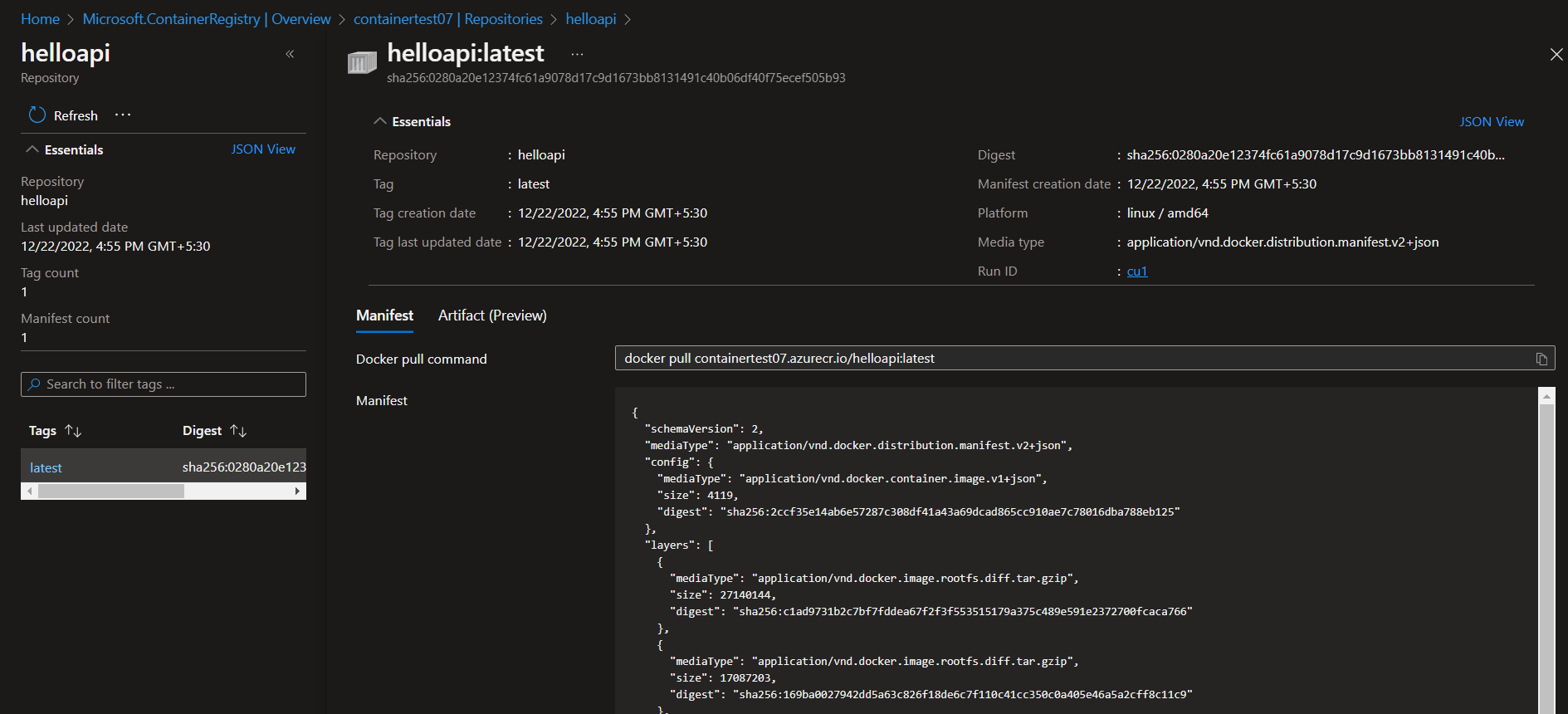
az acr build –image helloapi –registry containertest07 .

it will take few minutes to complete.

Graphical user interface, text

Description automatically generated

Step 6: Go to the Azure Container Registry which you created and in the repository, check your cloned project is copied along the docker pull command .



Step 7: Then run the command:

touch deployment.yaml

code .

this could create new yaml file in the folder named ‘deployment.yaml’ and copy the following content in it :

apiVersion: apps/v1

kind: Deployment

metadata:

name: helloapi-deployment

spec:

selector:

matchLabels:

app: helloapi-pod

template:

metadata:

labels:

app: helloapi-pod

spec:

containers:

- name: helloapi-container

image: containertest07.azurecr.io/helloapi:latest

resources:

limits:

cpu: 500m

memory: 128Mi

ports:

- containerPort: 80

then create one new file by running the following command:

touch service.yaml

code.

then copy the following code in it:

apiVersion: v1

kind: Service

metadata:

name: helloapi-service

spec:

ports:

- port: 8080

targetPort: 80

selector:

app: helloapi-pod

type: LoadBalancer

Step 8: Then run the following command to connect the Azure Container Registry with the Azure Kubernetes Cluster:

az aks update -n aks-test -g karamjotsandhu --attach-acr containertest07

then run the following commands:

i) To deploy the application

kubectl apply -f deployment.yaml

ii) To check the deployed application

kubectl get deployment

iii) To check the status (running)

kubectl get pods

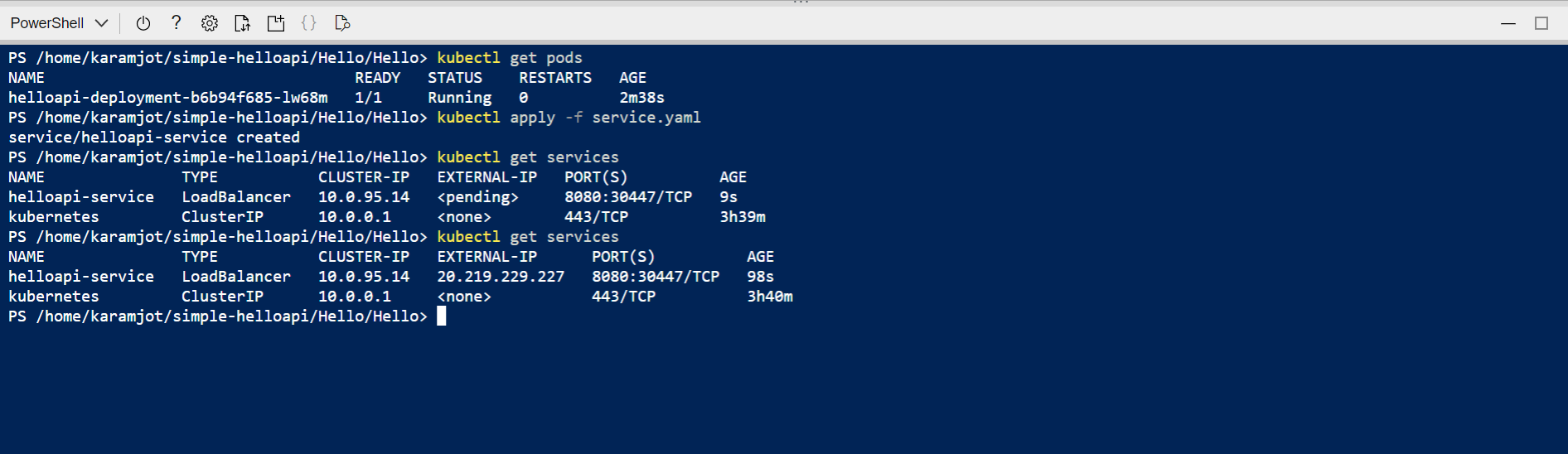
iv) To up the service

kubectl apply -f service.yaml

v) To check services

kubectl get services

Last command will show the EXTERNAL-IP and PORT(S) at which the application is accessible by the end-user.



Step 9: To see the result, copy the following address in the browser:

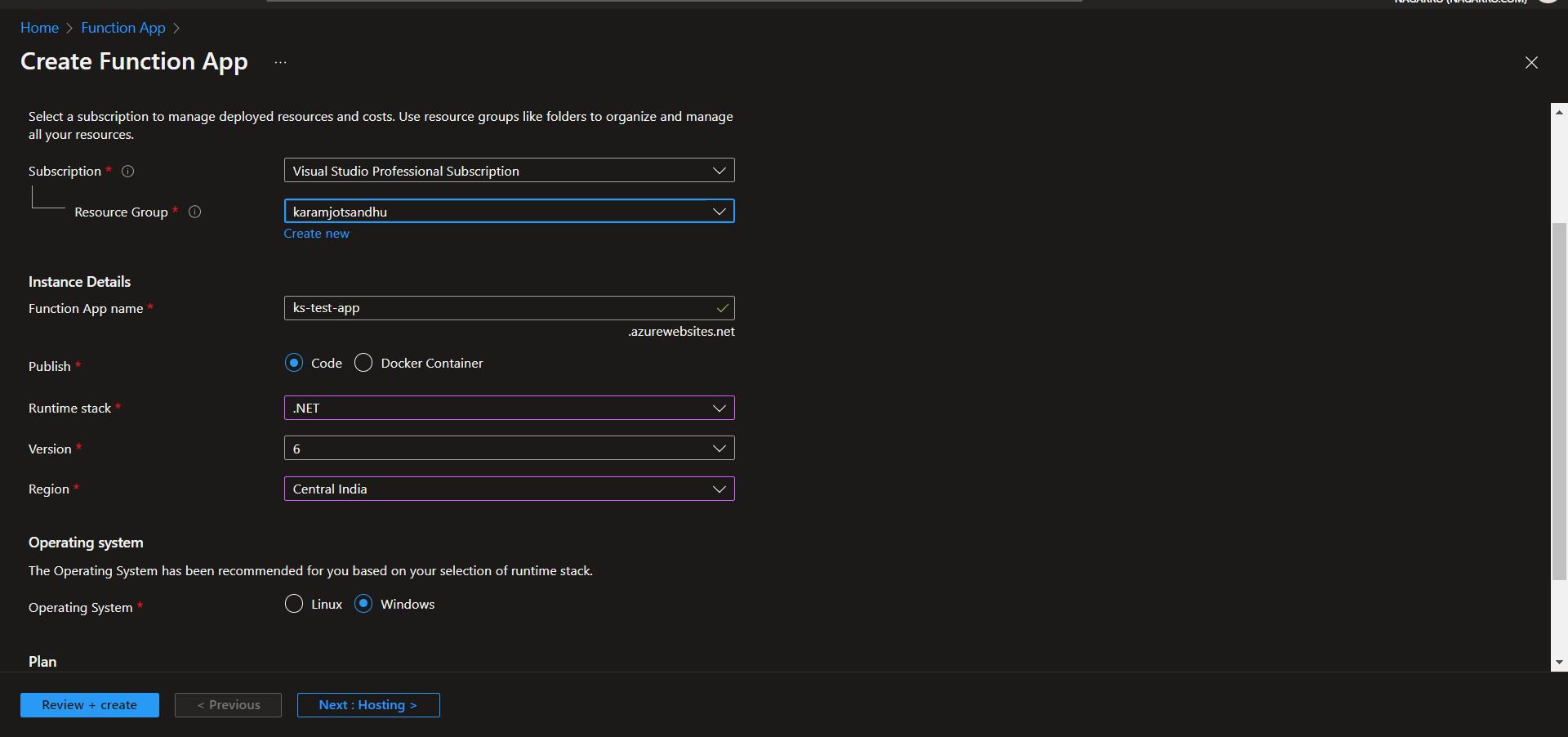
http://20.219.229.227:8080/Hello/hello

Text

Description automatically generated

1. Create an Azure function that should trigger as soon as you upload a file in the blob storage. Function should be able to print the name of the file uploaded in the function.

* Step 1: Search the Function App in the search bar of the Azure portal, then select the subscription and resource group, then give it a name, then select publish as a code, then select runtime stack .NET, then select region.



Step 2: Select the existing storage account or create a new one, then select the plan type as Consumption (Serverless) .

Graphical user interface, text

Description automatically generated

Step 3: Review and create the function app.

A screenshot of a computer screen

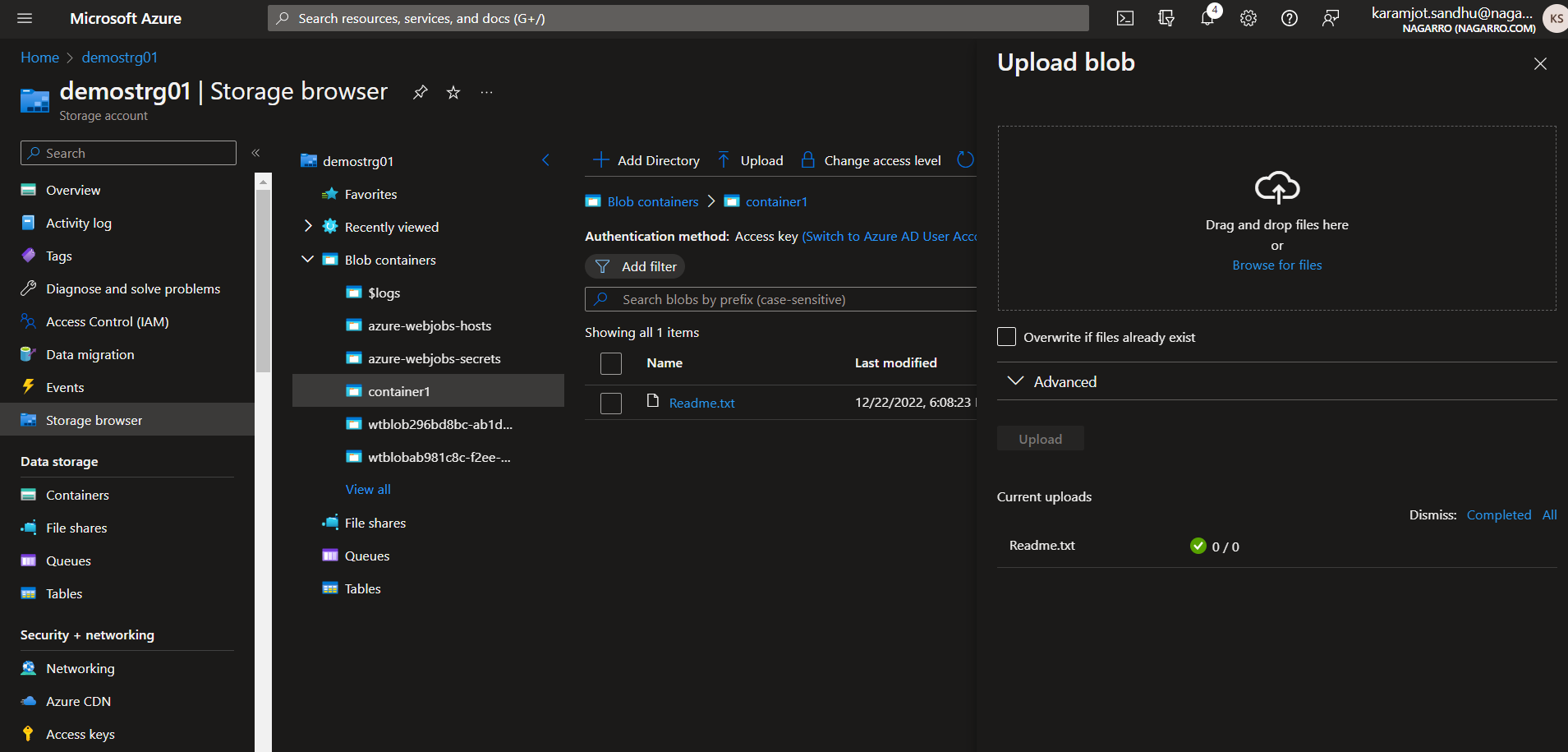
Description automatically generated with medium confidence

Step 4: Go to the function app you created, then open the functions from the left side bar, then click on the add new function, then select the template of the function as Azure Blob Storage Trigger.

Graphical user interface, text

Description automatically generated

Step 5: Then go to the storage account, select the Storage Explorer from the left-pane, then add container of the same name which name is given during the trigger creation, or see that from the integration in the trigger and add to the Blob Containers, then upload any file to the container.



Step 6: The trigger has printed the name of the file which you have uploaded to the Blob container, see in the log of the specific hit.