*CLOUD ASSIGNMENT*

ASSIGNMENTS:

* Create a Virtual Network with 2 subnets. Each subnet should have 16 Ips only.

1. So firstly, we are going to create a resource named Virtual network.

Graphical user interface, text, application, email

Description automatically generated

1. After adding the basic details for creating the virtual network we will head toward Ip addresses settings and click on “Add subnet” to create 2 subnets.

Graphical user interface, text, application, email

Description automatically generated

1. We selected the subnet address range such that it should contain 16 Ips only (10.0.0.0/28 and 10.0.1.0/28). 11 Ips + 5 reserved addresses.
2. After creating the subnets, we keep the remaining settings default and click on review and create to create the virtual network.

Graphical user interface, text, application, email

Description automatically generated

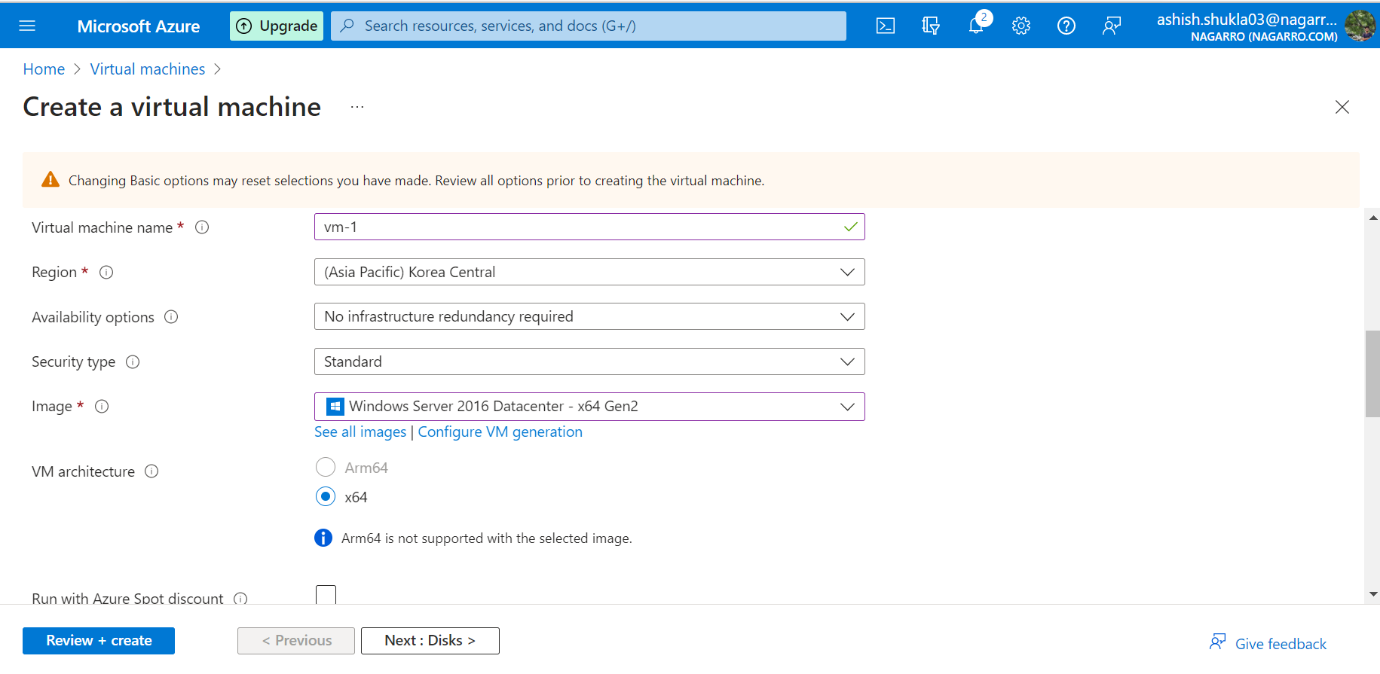
1. After the validation passes, we just must click on create and our resource will be created successfully.

Graphical user interface, application

Description automatically generated

* Inside one of the subnets, create a VM and deploy MVC application code inside it and it should leverage the database on the cloud.

1. So firstly, we are going to create a virtual machine by clicking on create a resource and selecting virtual machine. We must select the same resource group which we have created before while creating virtual network to create the virtual machine in one of the subnets.



1. After filling the basic details, we will click on review and create for creating the virtual machine.

Graphical user interface, text, application

Description automatically generated

1. After the validation passes, we just must click on create and our virtual machine resource will start getting deployed (Here we have created Linux operating system virtual machine).

Graphical user interface, text, application, email

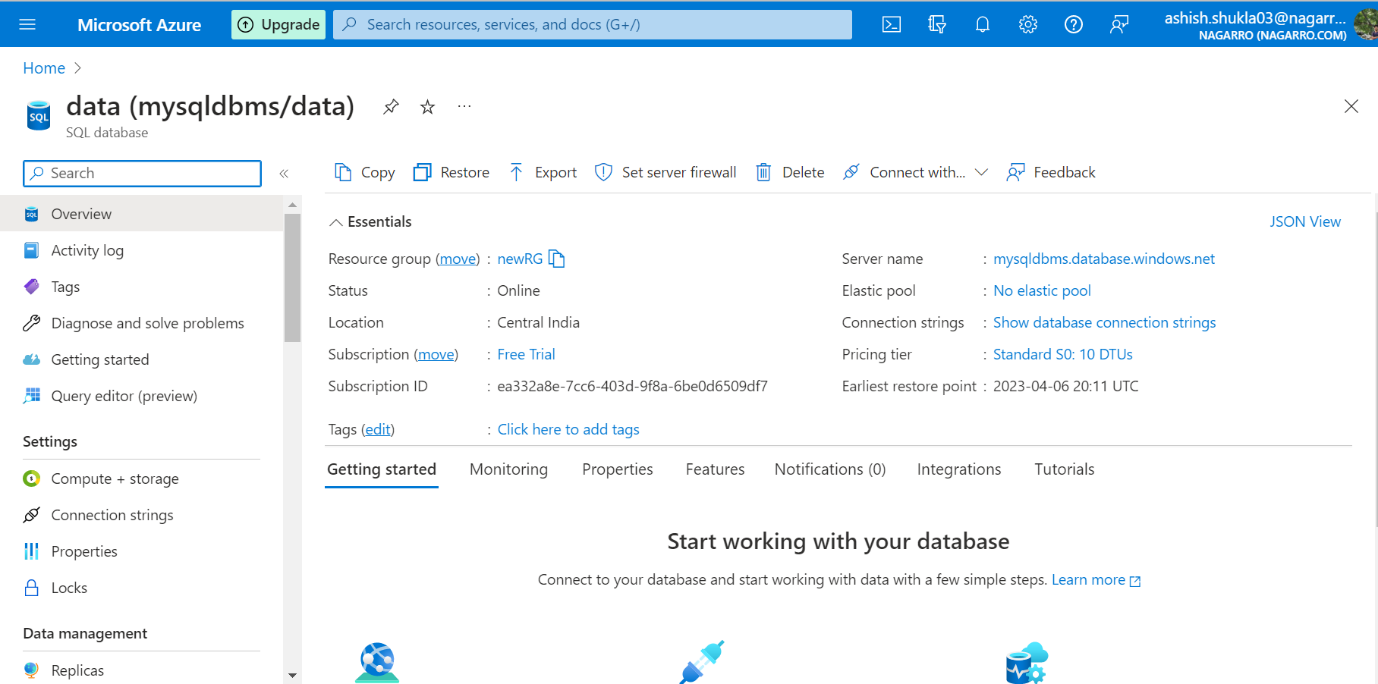
Description automatically generated

1. Now we must deploy our MVC application on this virtual machine and connect it database. So, firstly we will connect our MVC application with SQL Database which we will create on Azure cloud.
2. So, after selecting the SQL database resource, we must fill the basic details regarding creation of the database.

Graphical user interface, application

Description automatically generated

1. After filling the basic details, we will just click on review and create for validation. As the validation passes, we will click on create to deploy our SQL database resource on azure cloud platform.



1. Now after creating the SQL database, we will click on connection string and add the string, database username and password to our MVC application settings to create a connection between our application and Azure SQL database.

Graphical user interface, text, application, email

Description automatically generated

1. As u can see in the below screenshot that all the details of the Azure SQL database have been added to our MVC application (Here we are using a spring boot application of library management which keeps the records of users, authors, and books in the database)

A screenshot of a computer

Description automatically generated

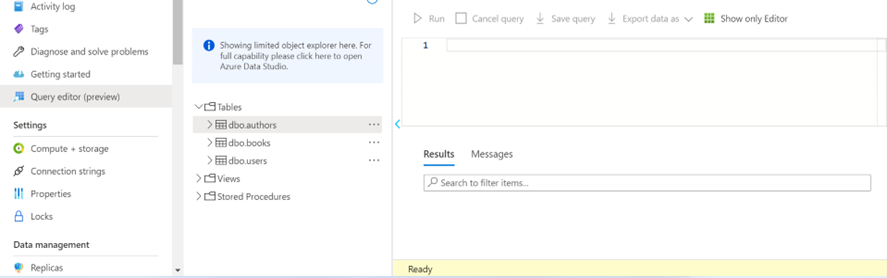
1. So now after running our application and navigating back to our azure SQL database we will click on “Query Editor option” present on the left side and add our login and password details for our database.
2. After adding the login and password we will click on OK.

Graphical user interface, text, application, email

Description automatically generated

1. In the database we will be able to see the authors, books and user’s column below tables as shown in below screenshot which clearly shows that our spring boot application is connected to azure SQL database.

Text

Description automatically generated

1. We have created a user by running a query in our Azure SQL database. We will login in our application with this user and try to add a new record of book in our application and after adding the record we will notice that it is automatically getting stored in our Azure SQL Database.
2. Now we will login to our application with this username and password

Graphical user interface, text, application

Description automatically generated

1. Adding a new record of book and submitting.

Graphical user interface, text, application, email

Description automatically generated

As we can see that the record is added successfully.

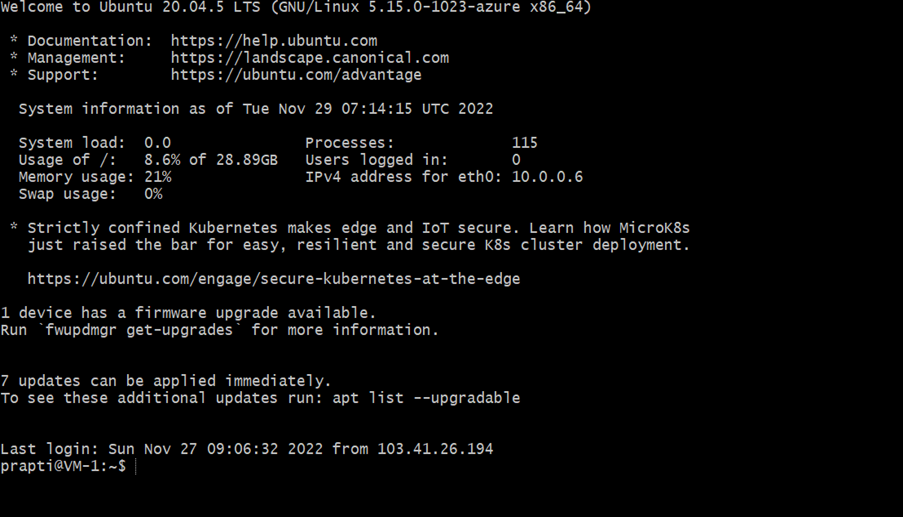
Graphical user interface, application

Description automatically generated

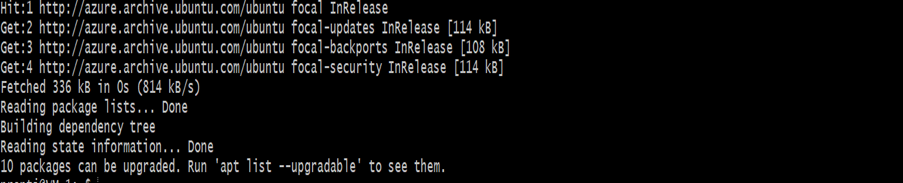
15.Now our next step is to deploy our application to Virtual machine which we have created.

For this firstly we must install and configure tomcat on azure VM.

So, we will follow the steps below to configure tomcat.



Firstly, we will update the package manager catche by running “sudo apt update”.



For security purposes, tomcat should run under a separate, unprivileged user for which we will use the command “sudo useradd -m -d /opt/tomcat -U -s /bin/false tomcat” to create a new user called tomcat.

Now, we will get the URL of tar from the tomcat 9 download page and use wget command to download the latest version.

Text

Description automatically generated

and extract the downloaded file by running “sudo tar xzvf apache-tomcat-9\*tar.gz -C /opt/tomcat –strip-components=1” command.

Now we will grant tomcat user the ownership over the extracted /opt/tomcat folder and executable permissions to contents in /opt/tomcat/bin using “sudo chown -R tomcat:tomcat /opt/tomcat/” and “sudo chmod -R u+x /opt/tomcat/bin” commands.

For configuring the admin users, we will open the tomcat user configuration file and define privileged manager and admin users in the tomcat user configuration, by adding the following lines before ending tags.

**<role rolename=”manager-gui” />  
<user username=”manager” password=”< Ashishshukla123>” roles=”manager-gui” /><role rolename=”admin-gui” />  
<user username=”admin” password=”< Ashishshukla123>” roles=”manager-gui,admin-gui” />**

Now we will open manager page config profile by using “sudo nano /opt/tomcat/webapps/manager/META-INF/context.xml” and comment out valve definition to enable access to manager page with the tomcat user.

Text

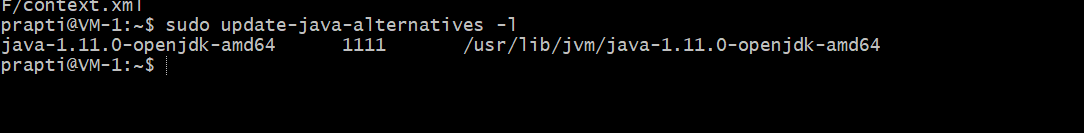
Description automatically generated

Again, we will open the host-manager page config file by running “sudo nano /opt/tomcat/webapps/host-manager/META-INF/context.xml” command and comment out valve definition to enable access to host-manager page with the tomcat server.



Now we must create “systemd” service which will keep tomcat quietly running in the background and restart tomcat automatically in case or error.

Tomcat, being a java application itself, requires java runtime. We will run the “sudo update-java-alternatives -l” command to get the location of java.



Creating a tomcat service file named tomcat.service under /etc/system/system using “sudo nano /etc/systemd/system/tomcat.service” command and adding the following lines.

**[Unit]  
Description=Tomcat 9 servlet container  
After=network.target[Service]  
Type=forkingUser=tomcat  
Group=tomcatEnvironment=”JAVA\_HOME=/usr/lib/jvm/java-1.11.0-openjdk-amd64"  
Environment=”JAVA\_OPTS=-Djava.security.egd=file:///dev/urandom -Djava.awt.headless=true”Environment=”CATALINA\_BASE=/opt/tomcat”  
Environment=”CATALINA\_HOME=/opt/tomcat”  
Environment=”CATALINA\_PID=/opt/tomcat/temp/tomcat.pid”  
Environment=”CATALINA\_OPTS=-Xms512M -Xmx1024M -server -XX:+UseParallelGC”ExecStart=/opt/tomcat/bin/startup.sh  
ExecStop=/opt/tomcat/bin/shutdown.sh[Install]  
WantedBy=multi-user.target**

Text

Description automatically generated

Now, reloading the system daemon so it become aware of the new service using “sudo systemctl daemon-reload” and “sudo systemctl start tomcat ” to start the tomcat service.

We can also use “sudo systemctl status tomcat” to check the status of tomcat .

To enable tomcat to start up the system, we can run “sudo systemctl enable tomcat” command.

Text

Description automatically generated

16. If we try to access tomcat webpage from outside using VM’s public IP, we will get an unreachable response. For accessing the tomcat web service using public IP, we need to enable HTTP and tomcat service port by adding new inbound security rule in VM’s networking section.

Graphical user interface, text, application

Description automatically generated

Adding inbound security rule to allow tomcat server port 8086.

Graphical user interface, text, application, email

Description automatically generated

1. Now we will be able to access Tomcat welcome page from outside using VM’s public IP.

Graphical user interface, application, website

Description automatically generated

1. Now we must generate a war file of our spring boot application and deploy that war file in the tomcat server.

So, after creating the war file we will login to the manager page using the manager credentials and under war file to deploy section, choose the spring boot war file and deploy.

Table

Description automatically generated with medium confidence

1. Once the deployment is successful, the library.management.war will be available in the list of applications.

Table

Description automatically generated

1. Now we will be able to access the deployed spring boot application app.

Graphical user interface, text, application

Description automatically generated

* Deploy the same MVC application to Azure App Service.

1. For deploying the same MVC application to azure app service, we will create a new resource called App Services.

Graphical user interface, text, application, email

Description automatically generated

Adding all the basic details and clicking on review and create to deploy

Our app service.

Graphical user interface, text, application, email

Description automatically generated

1. Now we must add our spring boot application to GitHub for accessing the application for deployment.

The screenshot below shows that the application is successfully added to GitHub.

1. After adding our application to GitHub, we will move back to our azure app service and click on Deployment Center option.
2. After selecting the deployment center option, we must select source code as GitHub and add the repository, organization, branch and runtime stack for our application which is present on GitHub and save it.
3. After clicking on save, the setting up deployment process will start, and we can click on logs option to see the logs.
4. After clicking on logs, you will notice that the build of our deployment is started and as the build gets completed, it will start to deploy.

Here in the below screenshot, we can see that the deployment is completed successfully.

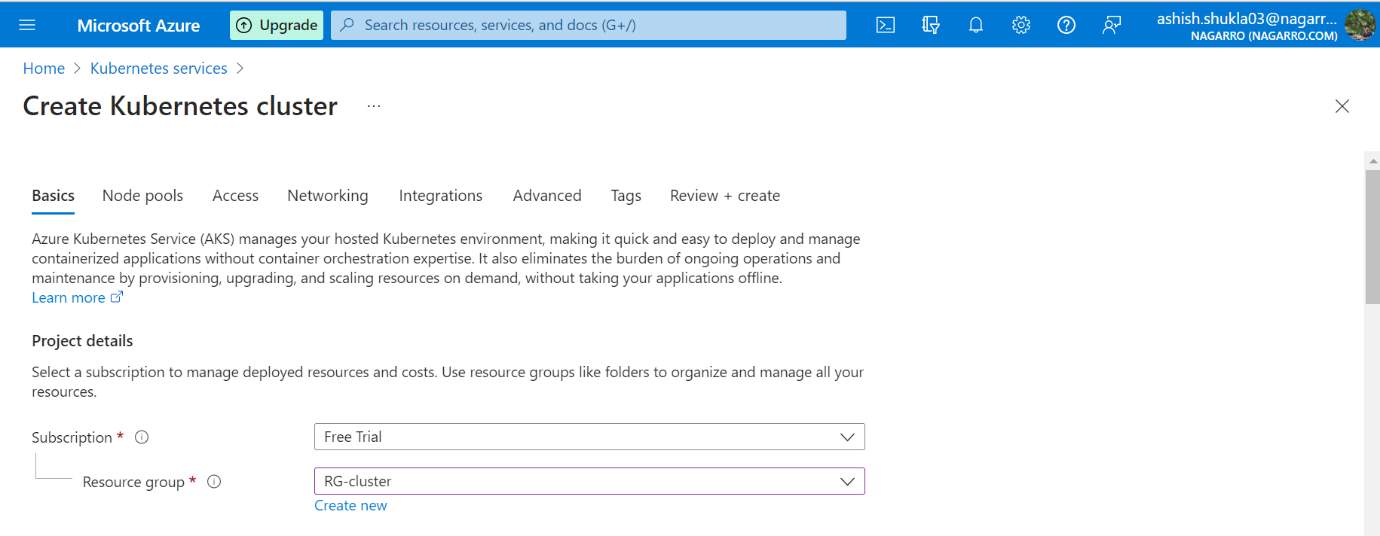
1. After the completion of the deployment, a link will be generated which we can use to access our application.
2. Here we can see that our application is successfully deployed on Azure web apps, and we are able to access it.

Graphical user interface, text, application

Description automatically generated

* Create the AKS cluster (2 Nodes) and deploy any service on it. Service should be accessible from the internet.

1. Firstly, we must create an AKS cluster with 2 nodes for which we will click on create a resource and search for Azure Kubernetes Service.
2. After selecting create Kubernetes cluster we will start filling the basic details to create a Kubernetes cluster.



1. While filling the basic details below Primary node pool option we must select nodes as 2.

Graphical user interface, text, application, email

Description automatically generated

1. After filling all the details just click on review and create and our AKS cluster will start getting deployed.
2. So now we will be using the Azure CLI to deploy our spring boot application to Azure Kubernetes cluster using one of the services of AKS i.e., Load Balancer.
3. Now to connect our AKS cluster to Azure CLI to use kubectl (command line for Kubernetes), we will use “az account set –subscription …” and “az aks get credentials –resource-group … ” commands.
4. We have also created docker file to create the docker image of our spring boot application which needs to be deployed and after creating the docker image we have saved it in public repository known as Docker Hub.

. Graphical user interface, text, application, email

Description automatically generated

1. Now we will create a deployment in our cluster by running “kubectl create deployment productservice –image=ashishshukla03/product-service: latest”.
2. We can’t access our application after deployment too till the time we do not create any service to it.

Now, to create a service we will run a command as “kubectl expose deployment productservice –type=LoadBalancer –port=8086”

1. After our service is created, we will type the command “kubectl get services ” to know the external IP and ports on which our application is being deployed.
2. As our deployment process is done, we can easily access our application deployed on AKS cluster using External Ip mentioned.

Graphical user interface, text, application, email

Description automatically generated

* 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.

1. Firstly, we must create Azure storage for which we will click on create a resource and search for storage account.
2. After selecting the storage account, we will fill the basic details and click on create and review to create an Azure storage account.

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

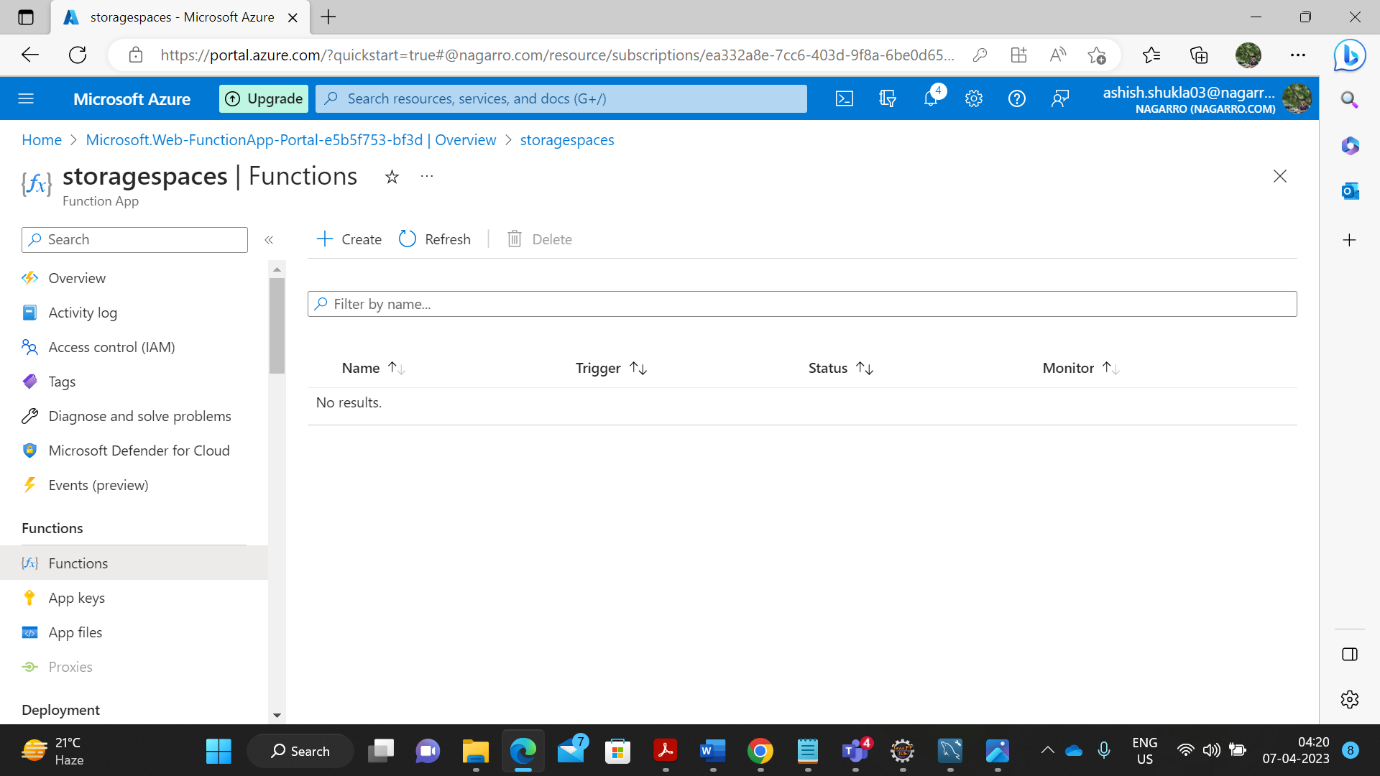
Description automatically generated

1. Now we will create one more resource called as Azure function.

Graphical user interface, text, application, email

Description automatically generated

1. Now after creating both the resources, we must go to Azure function app resource and click on function option to build a trigger for Azure blog storage.

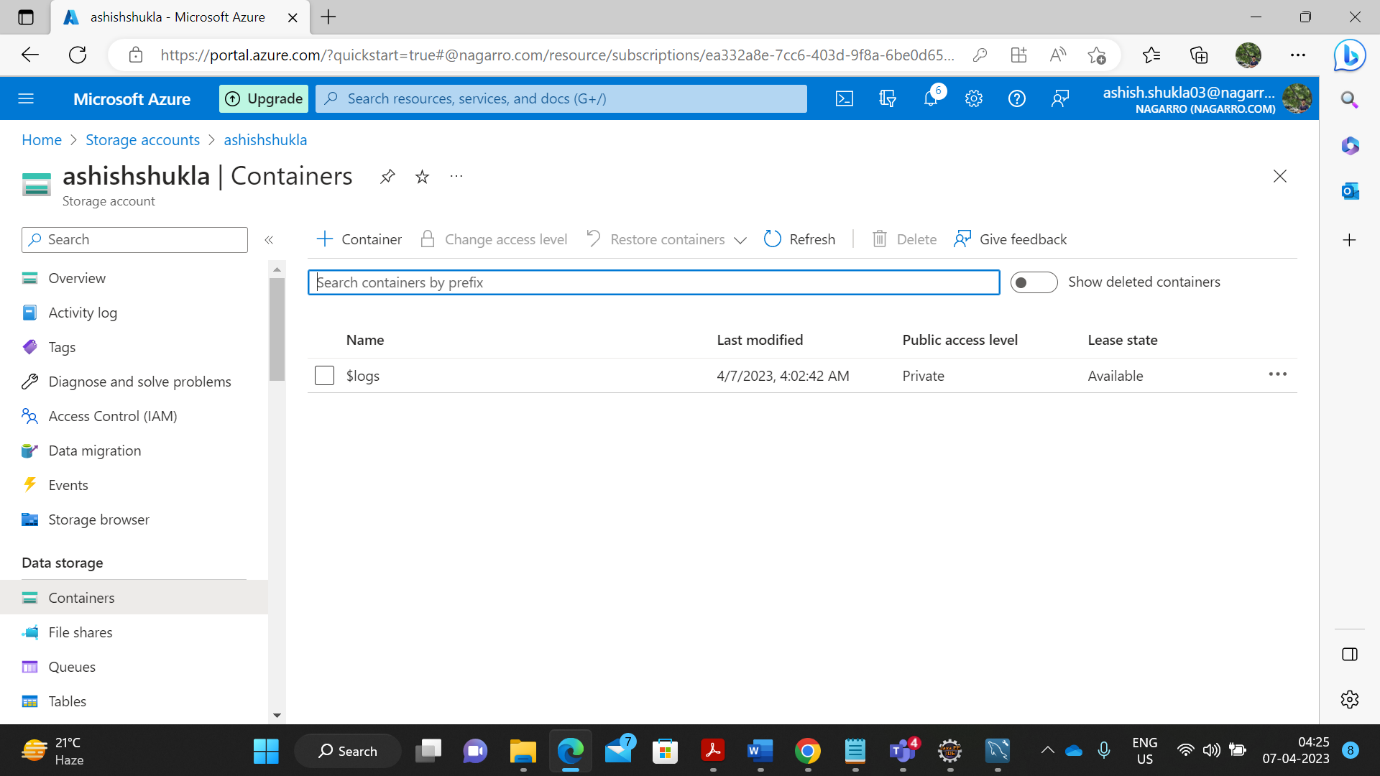


1. After clicking on create, we must select the Azure blog storage trigger and create a storage account connection with the azure storage which we have created recently.

Graphical user interface, text, application, email

Description automatically generated

1. Now when the Azure blob storage is created, we should head back to our storage account and create a blob storage container where we will upload any file and trigger will occur automatically.



Graphical user interface, text, application, email

Description automatically generated

1. Uploading a file in blob storage container.



1. As soon as the file is uploaded in the azure blob storage, an automatic trigger will occur in azure function with the name of the file uploaded in the blob storage in the log section.

Graphical user interface, text, website

Description automatically generated