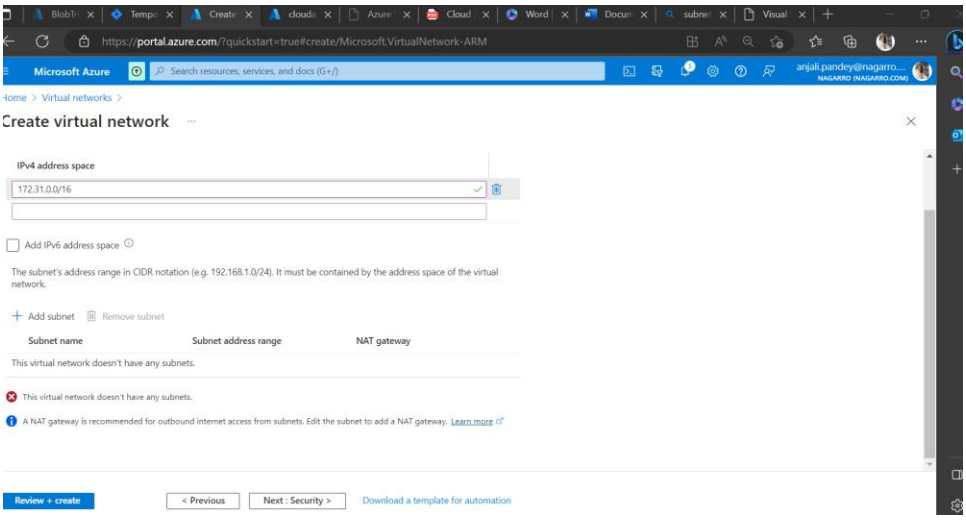
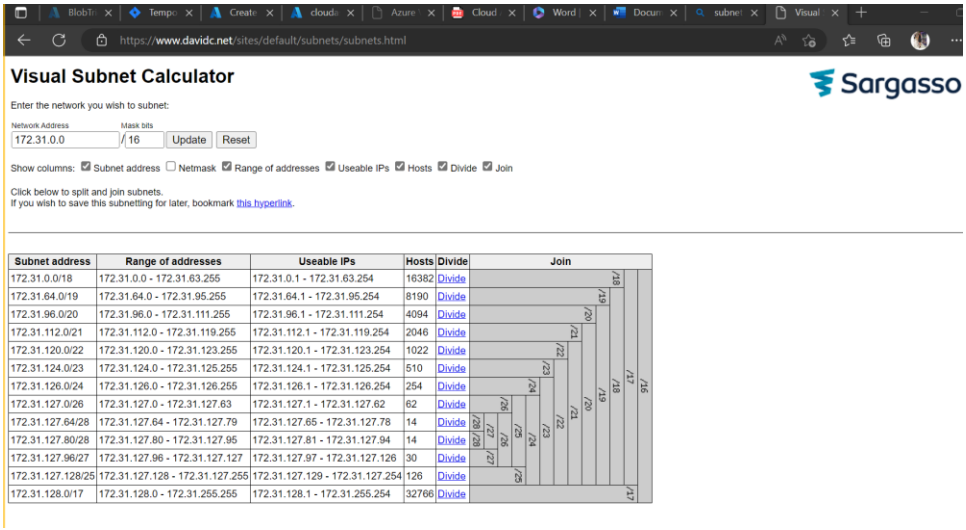
1). Create a virtual network with 2 subnets. Each subnet should have 16 Ips only

- **I went to Vnet (virtual network) and configured it with a cidr range 172.31.0.0/16.**

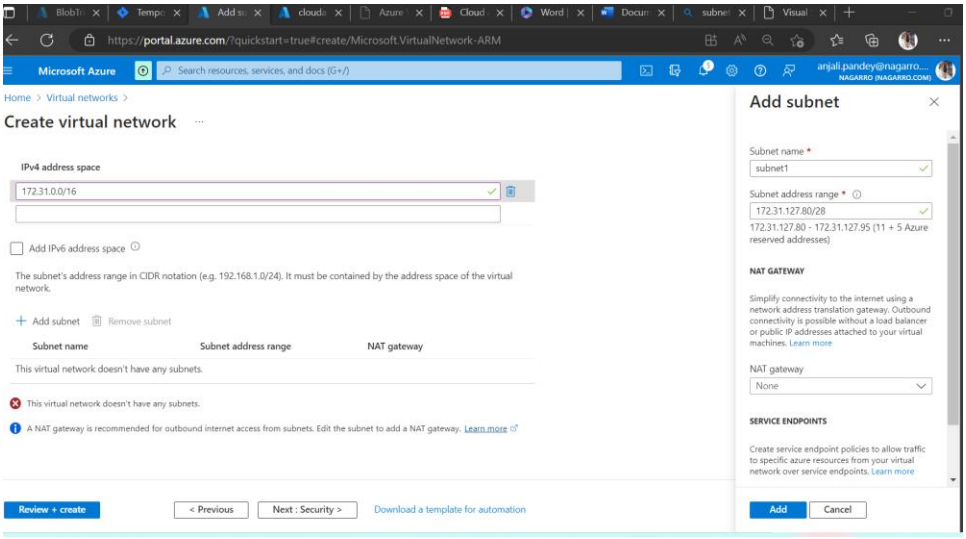


- **Then used tool for calculating subnet for 16 ips for this cidr range**

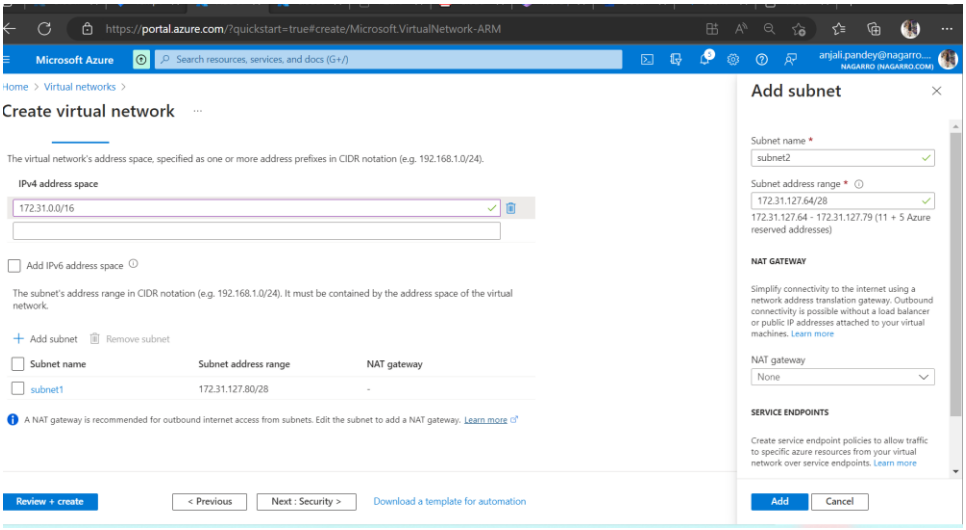


Subnet address	Range of addresses	Useable IPs	Hosts	Divide
172.31.0.0/18	172.31.0.0 - 172.31.63.255	172.31.0.1 - 172.31.63.254	16382	Divide
172.31.64.0/19	172.31.64.0 - 172.31.95.255	172.31.64.1 - 172.31.95.254	8190	Divide
172.31.96.0/20	172.31.96.0 - 172.31.111.255	172.31.96.1 - 172.31.111.254	4094	Divide
172.31.112.0/21	172.31.112.0 - 172.31.119.255	172.31.112.1 - 172.31.119.254	2046	Divide
172.31.120.0/22	172.31.120.0 - 172.31.123.255	172.31.120.1 - 172.31.123.254	1022	Divide
172.31.124.0/23	172.31.124.0 - 172.31.125.255	172.31.124.1 - 172.31.125.254	510	Divide
172.31.126.0/24	172.31.126.0 - 172.31.126.255	172.31.126.1 - 172.31.126.254	254	Divide
172.31.127.0/26	172.31.127.0 - 172.31.127.63	172.31.127.1 - 172.31.127.62	62	Divide
172.31.127.64/28	172.31.127.64 - 172.31.127.79	172.31.127.65 - 172.31.127.78	14	Divide
172.31.127.80/28	172.31.127.80 - 172.31.127.95	172.31.127.81 - 172.31.127.94	14	Divide
172.31.127.96/27	172.31.127.96 - 172.31.127.127	172.31.127.97 - 172.31.127.126	30	Divide
172.31.127.128/25	172.31.127.128 - 172.31.127.255	172.31.127.129 - 172.31.127.254	126	Divide
172.31.128.0/17	172.31.128.0 - 172.31.255.255	172.31.128.1 - 172.31.255.254	32766	Divide

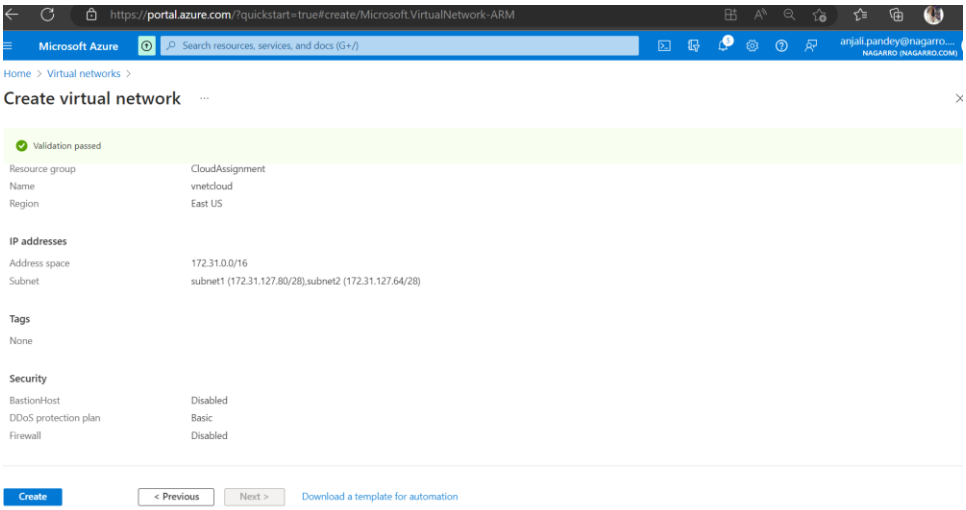
- **Calculated 2 subnets for 16 ips 172.31.127.64/28 and 172.31.127.80/28, created one subnet with no nat gateway because we does not wanted to private subnet.**



- **Created 2nd subnet**

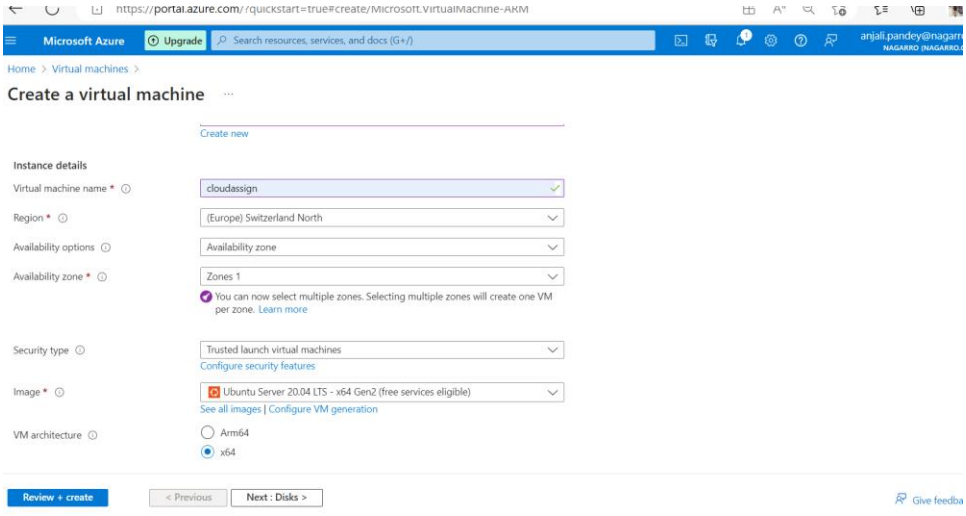


- **Now we can create and whole setup is done for vnet and subnet.**

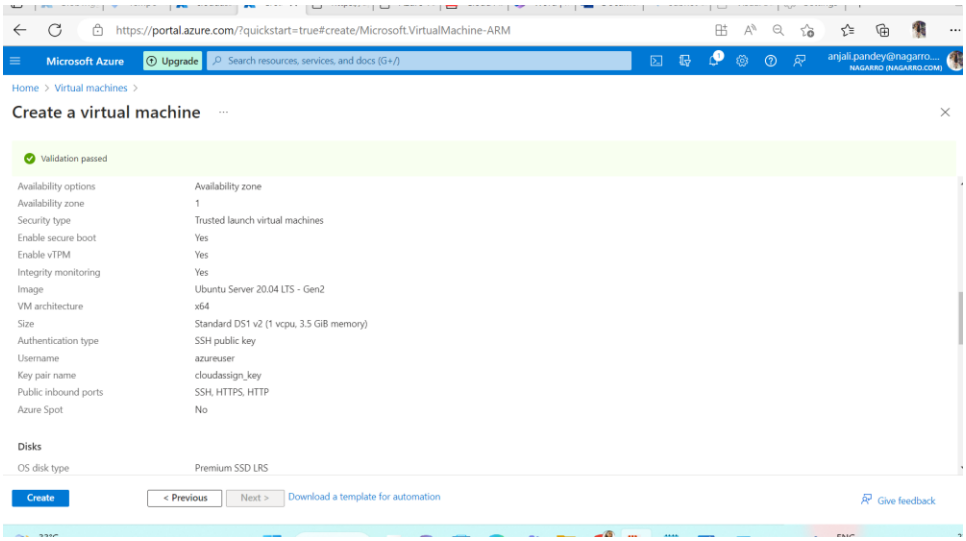


2). Inside one of the subnets, create a VM and deploy an application code inside it and it should leverage the database on the cloud (any existing application created by you before)

- **For creating vm search vm and the change the configration**

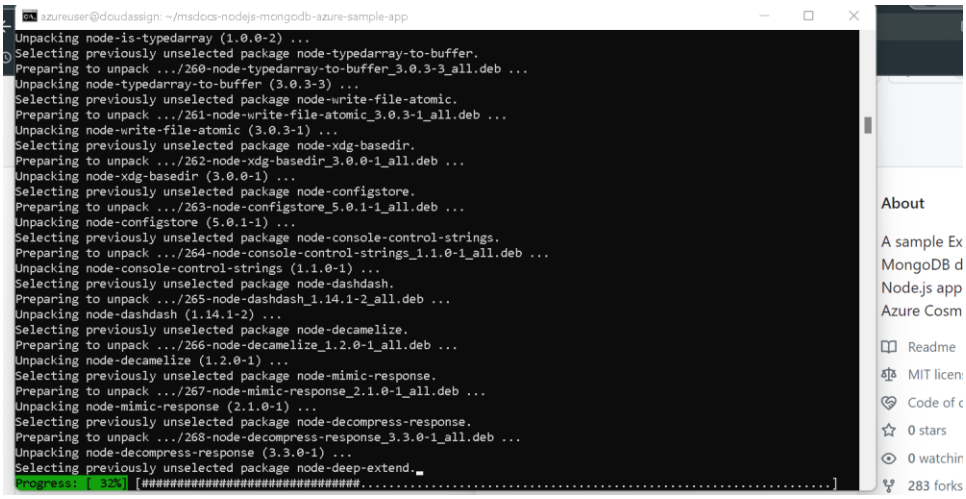


- **Choose the same subdomain and click on create with default configuration added two more ports 80 and 443**

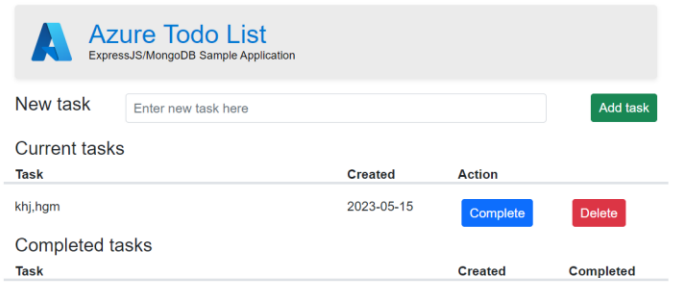


- **Then ssh into vm and start setting up application**

- **Then cloned application from my github repo and installing node and npm.**

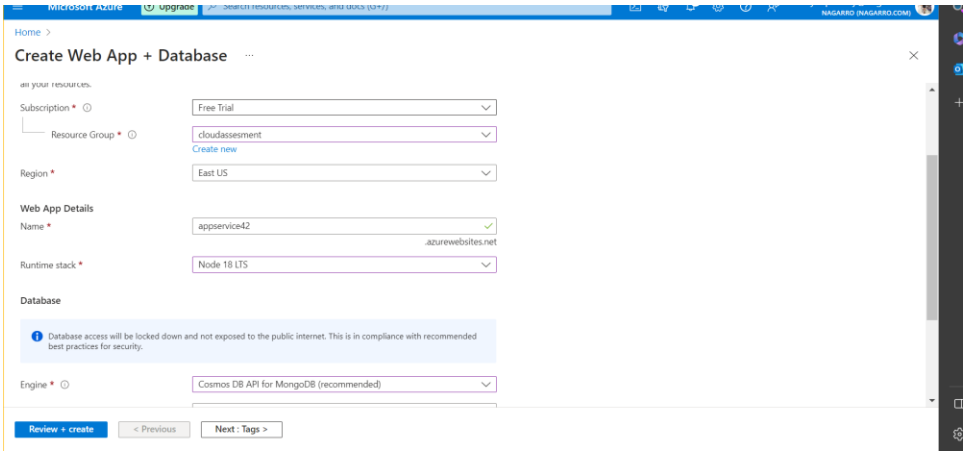


- **Then Npm I and npm run start and application started working**

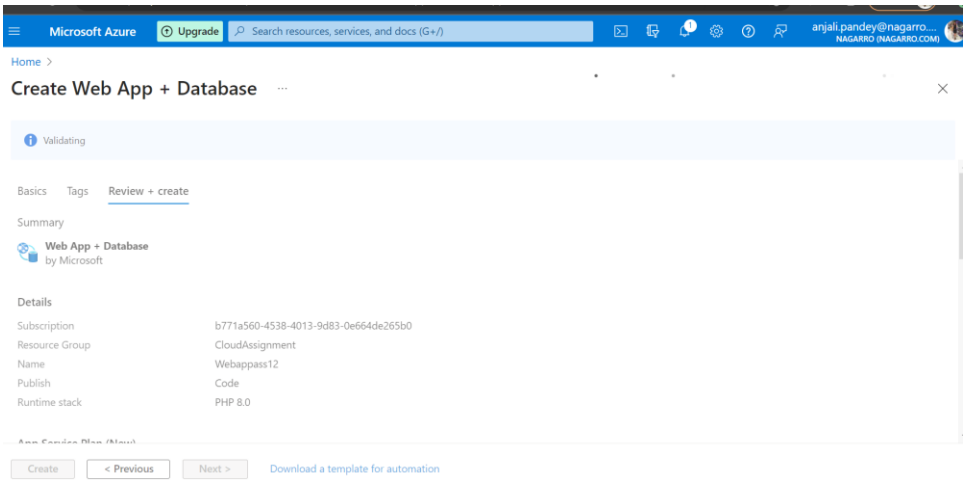


3). Deploy the same application to Azure App Service. It should also leverage the database on the cloud.

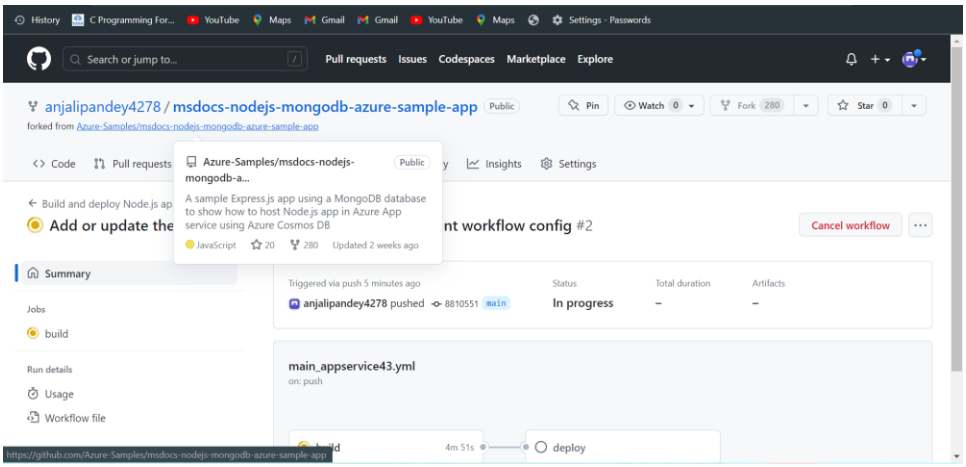
- **For creating this we choose webapp + database and provided information there**



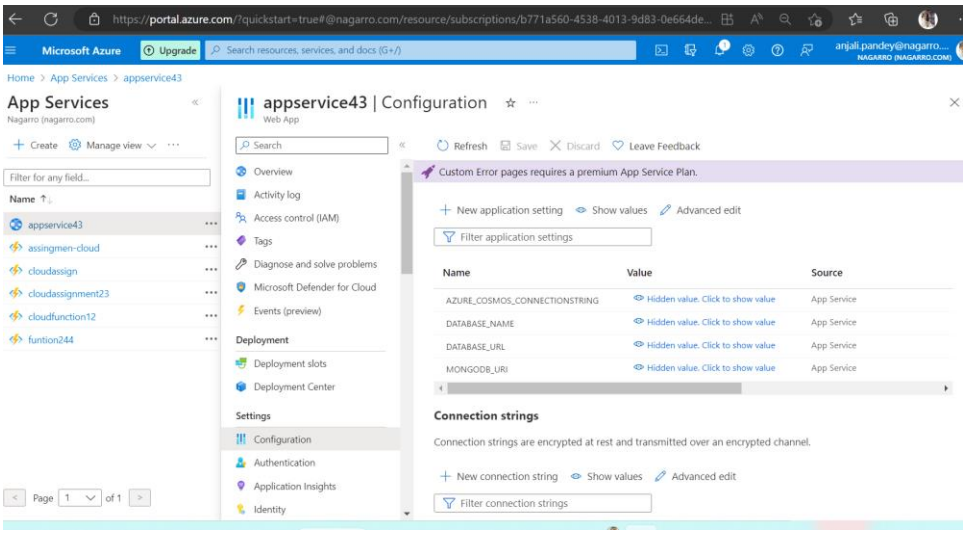
- **Then clicked on review + create**



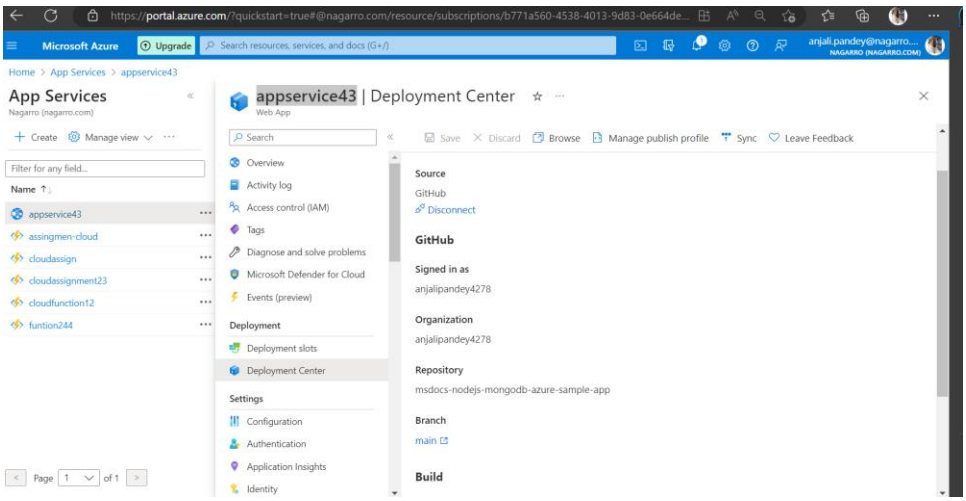
- **Used sample application for express and mongo db**



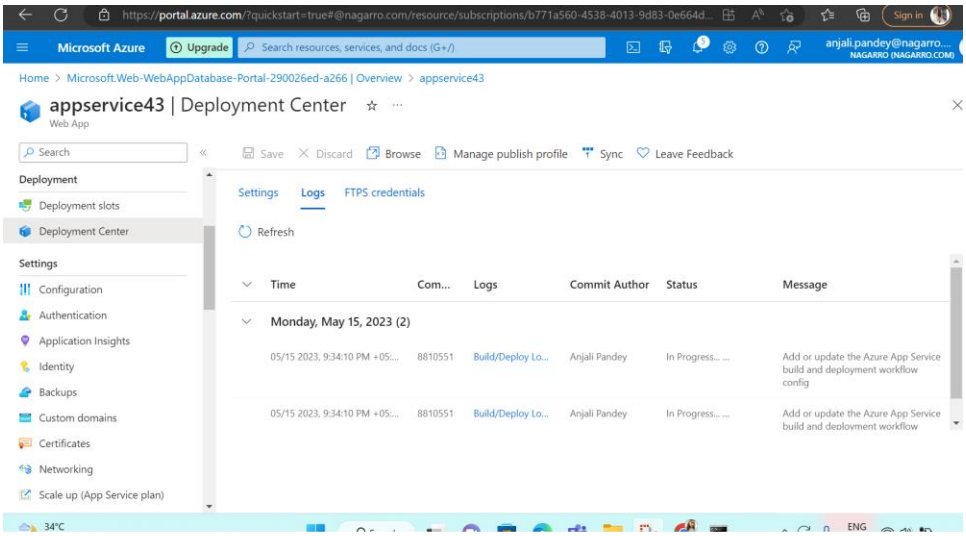
- **Added env for database string and name**



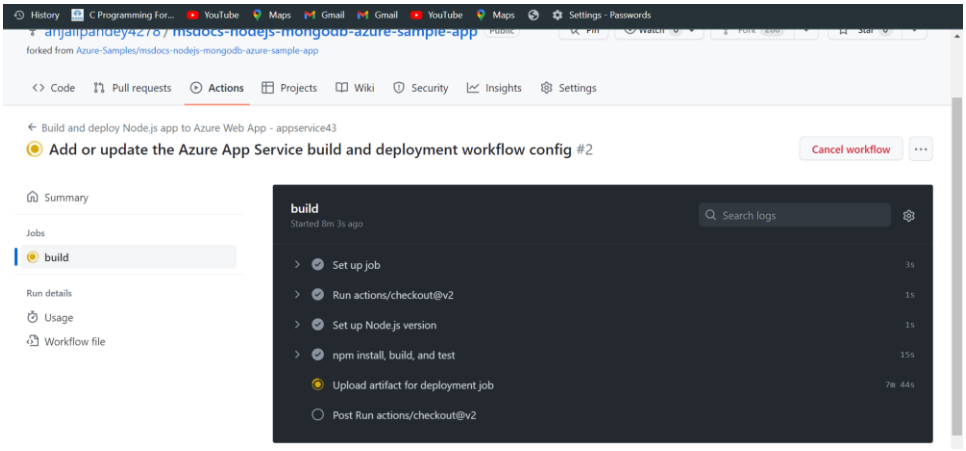
- **Went to deployment center and configured the github and added the branch**



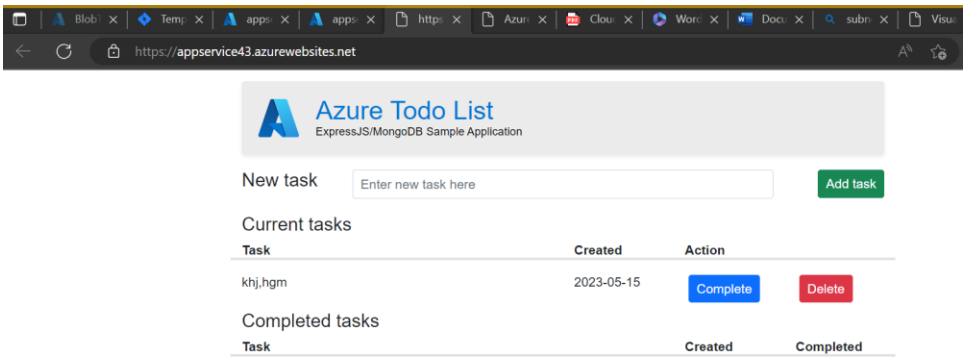
- **Then deploy the one ci/cd will run as a deployment part which we can take a look in portal**



- **We can check in github track that also**



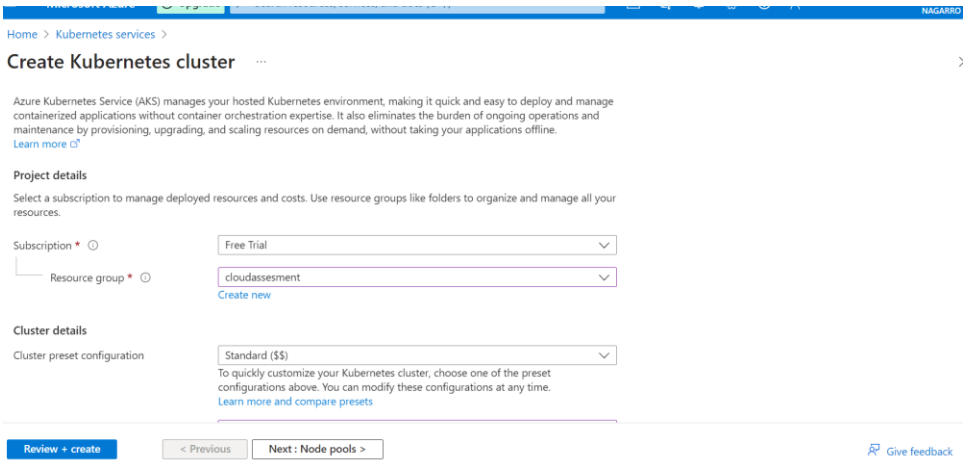
- **After deployment we can use the app domain appservice43.azurewebsites.net**



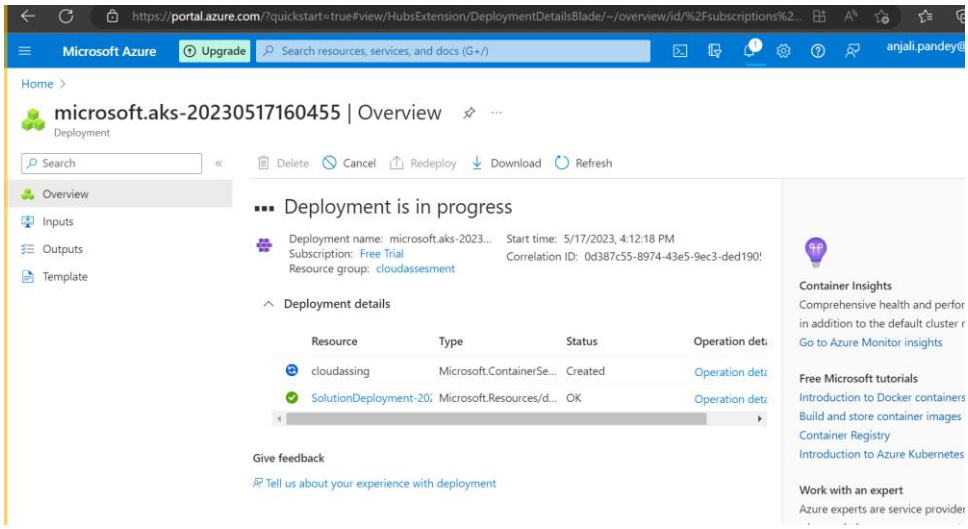
We can check there.

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

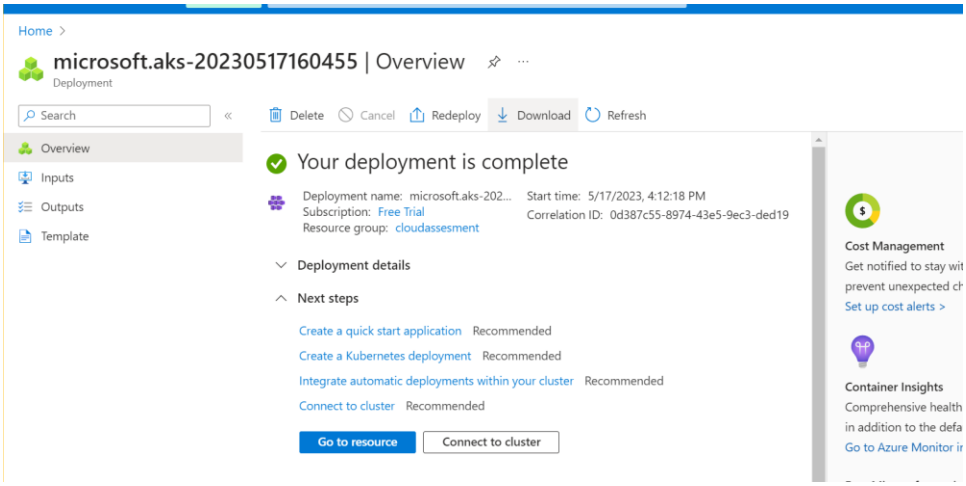
- **Went to Kubernetes cluster and click there**



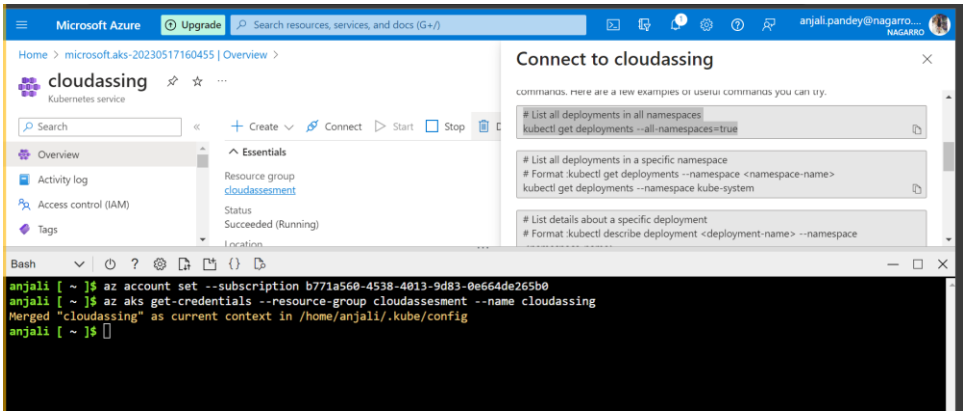
- **And choose 2 node which we want use so core will be 4 then with default setting and create**



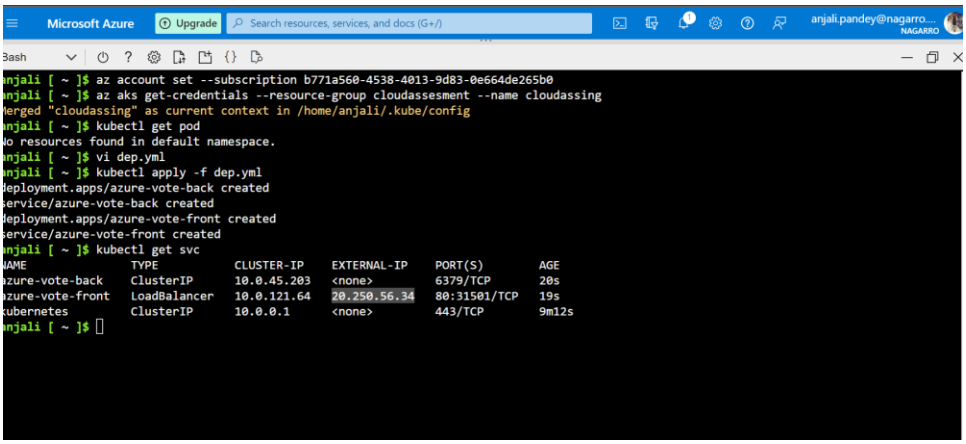
- **When deployment completed we can go to resource**



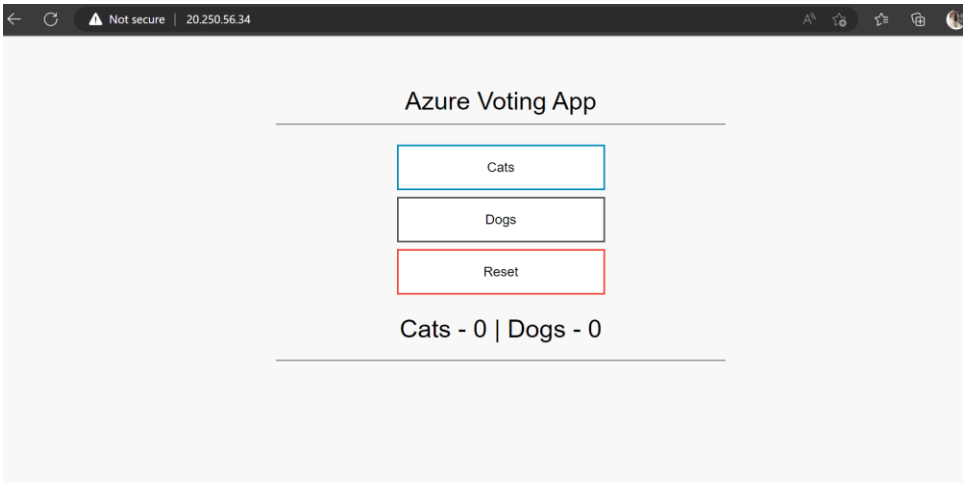
- **Then I have used cloudshell for deployment and accessing the cluster**



- **Then created one dep.yml file where created 2 service which are fronted and backend**

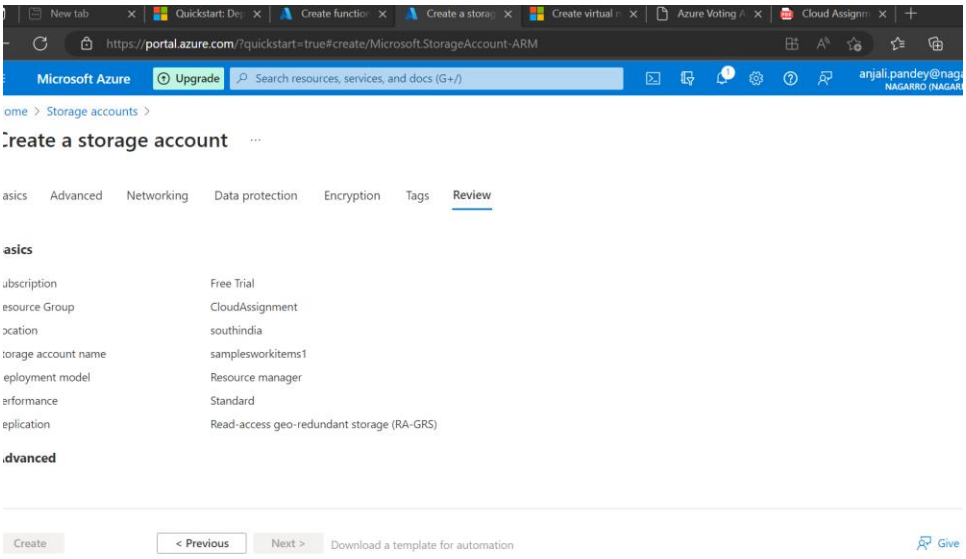


- **We created frontend with loadbalancer so got the external ip 20.250.56.34**

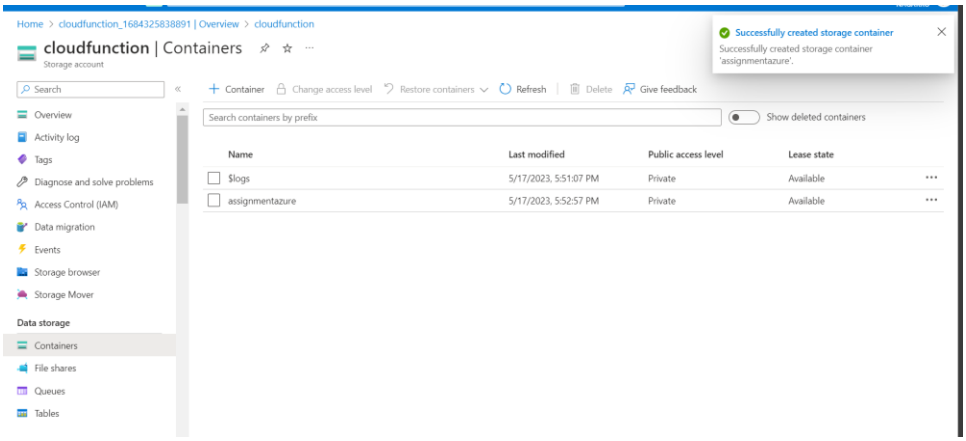


5). Create an Azure function that should trigger as soon as you upload a file in the blob storage. Function should be able to print the name of the file uploaded in the function.

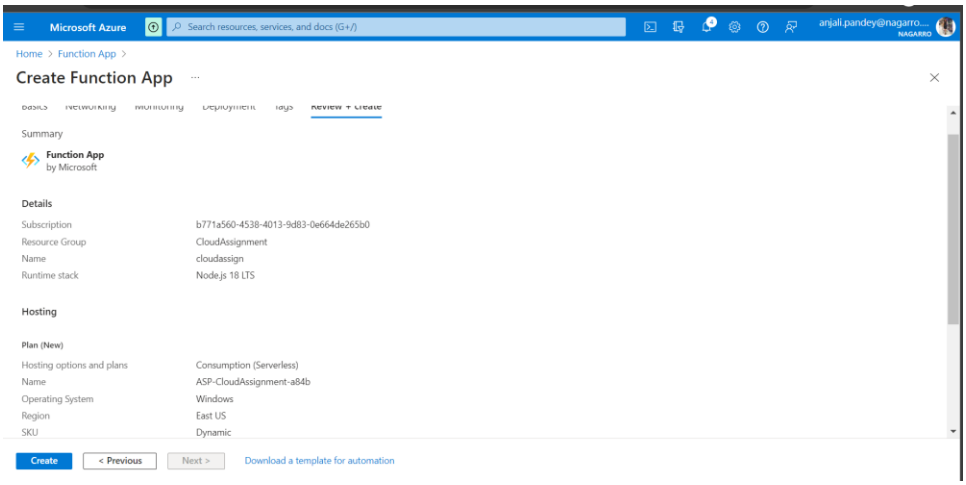
- **Created storage blob with name**



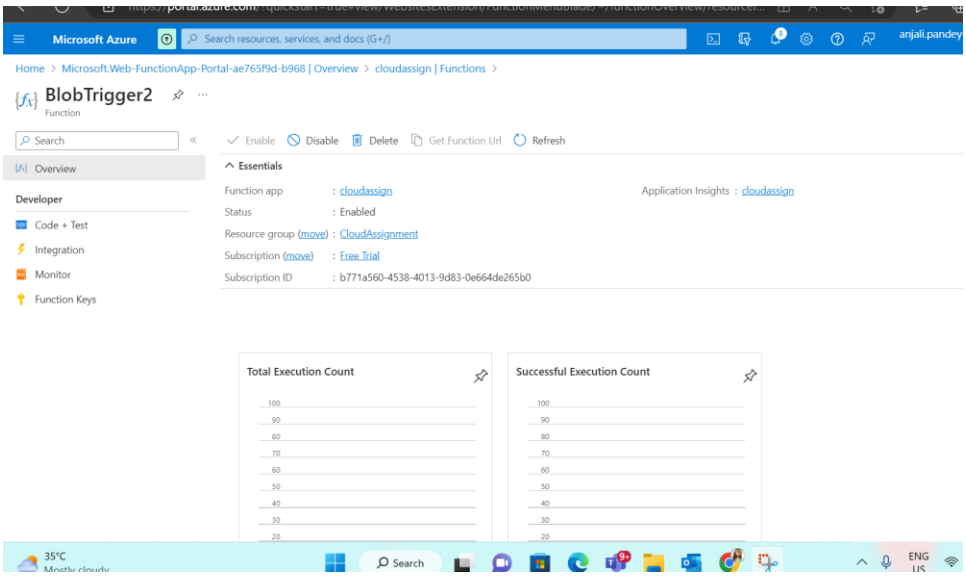
- **Then created container inside the storage blob**



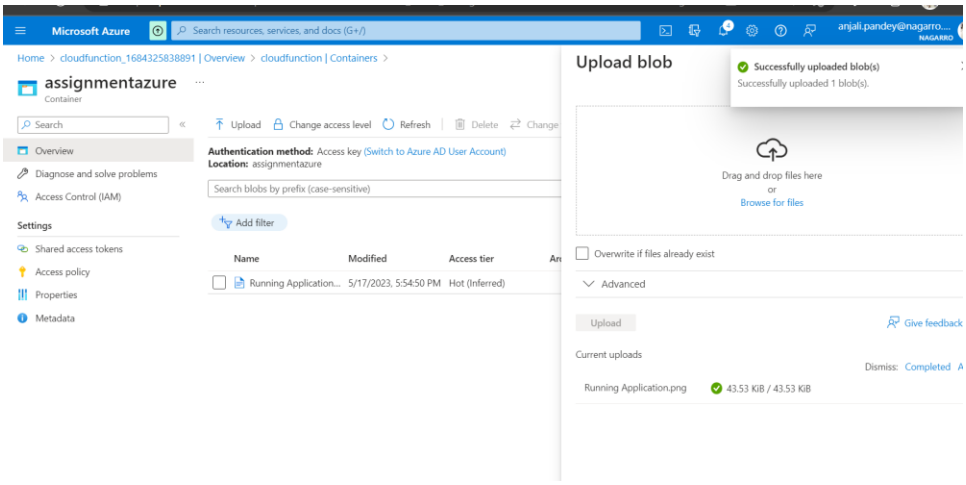
- **Then created function**



- **Then created a trigger for blob upload and created function for that**



- **Then upload the object and function will trigger.**



- **Then function will trigger and give output**