DOCKER

Docker vs Virtual Machines

* To provide some S/W and H/W functionalities virtually, VMs are needed
* VMs run on Hypervisor which are very costly
* To run applications that do not require all the features of the VM, special containers can be used
* This is where Docker comes into the picture
* Docker uses libraries and runs images on the Docker Engine instead of a hypervisor without the use of complicated infrastructure

| **Docker** | **Virtualization** |
| --- | --- |
| It boots in a few seconds. | It takes a few minutes for VMs to boot. |
| Pre-built docker containers are readily available. | Ready-made VMs are challenging to find. |
| Docker has a complex usage mechanism consisting of both third-party and docker-managed tools. | Tools are easy to use and more straightforward to work with. third-party. |
| Limited to Linux. | Can run a variety of guest OS. |
| Dockers make use of the execution engine. | VMs make use of the hypervisor. |
| It is lightweight. | It is heavyweight. |
| Host OS can be different from container OS. | Host OS can be different from guest OS. |
| Can run many docker containers on a laptop. | Cannot run more than a couple of VMS  on an average laptop. |
| Docker can get a virtual network adapter. It can have separate IPs ad Ports. | Each VMS gets its virtual network adapter. |
| Sharing of files is possible. | Sharing library and files are not possible. |
| Lacks security measures. | Security depends on the hypervisor. |
| A container is portable. | VMS is dependent on a hypervisor. |
| It allows running an application in an isolated environment known as a container | It provides easiness in managing applications, recovery mechanisms, and isolation from the host operating system |

Docker Commands:

* To start docker:

**sudo systemctl start docker**

* To see if any applications running on docker

**sudo docker ps**

* Create 3 files: Dockerfile, requirement.txt, app.py
* Add content to these files
* Build docker app by going into the directory

**sudo docker build -t my-python-app** .

* Run Docker App

**sudo docker run my-python-app**

Advantage of Docker Images:

* Makes testing operations easier
* ACR is a repository that can store any type of image

To install Azure CLI

**curl -sL https://aka.ms/InstallAzureCLIDeb | sudo bash**

Azure CLI login

Az login

* Azure Resource Group Login

**az group create --name RG\_shell --location eastus**

* Azure acr(Azure Container Registry) creation

**az acr create --name idacrmani --resource-group RG\_shell --location eastus --sku Standard**

\*\*Note: my-python-app is the name of the image

* Login to azure acr needs to be done

**sudo docker login idacrmani.azurecr.io**

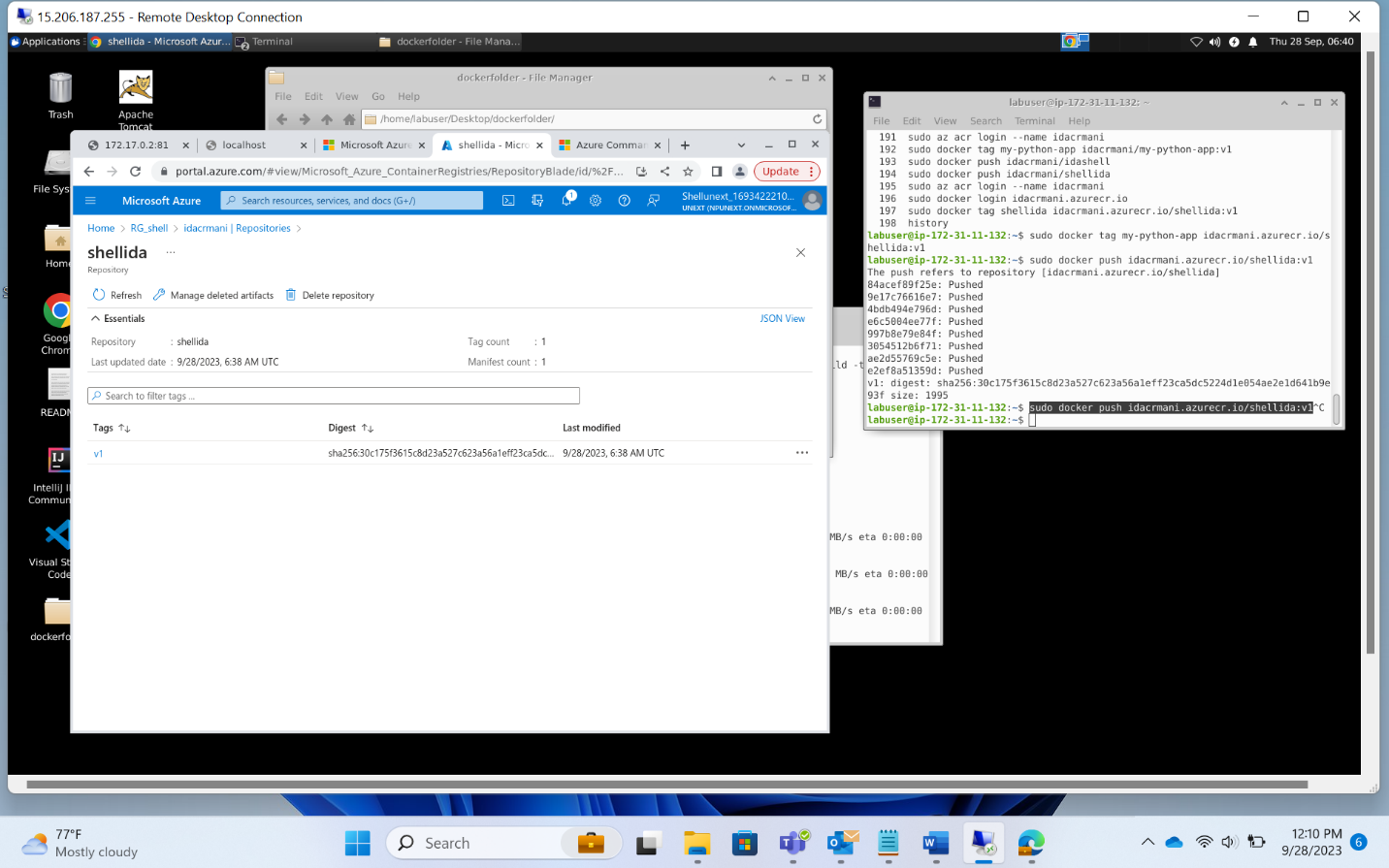
* Image tagging to the acr needs to be done to create a connection before push

**sudo docker tag my-python-app idacrmani.azurecr.io/shellida:v1**

* Push of the image into acr

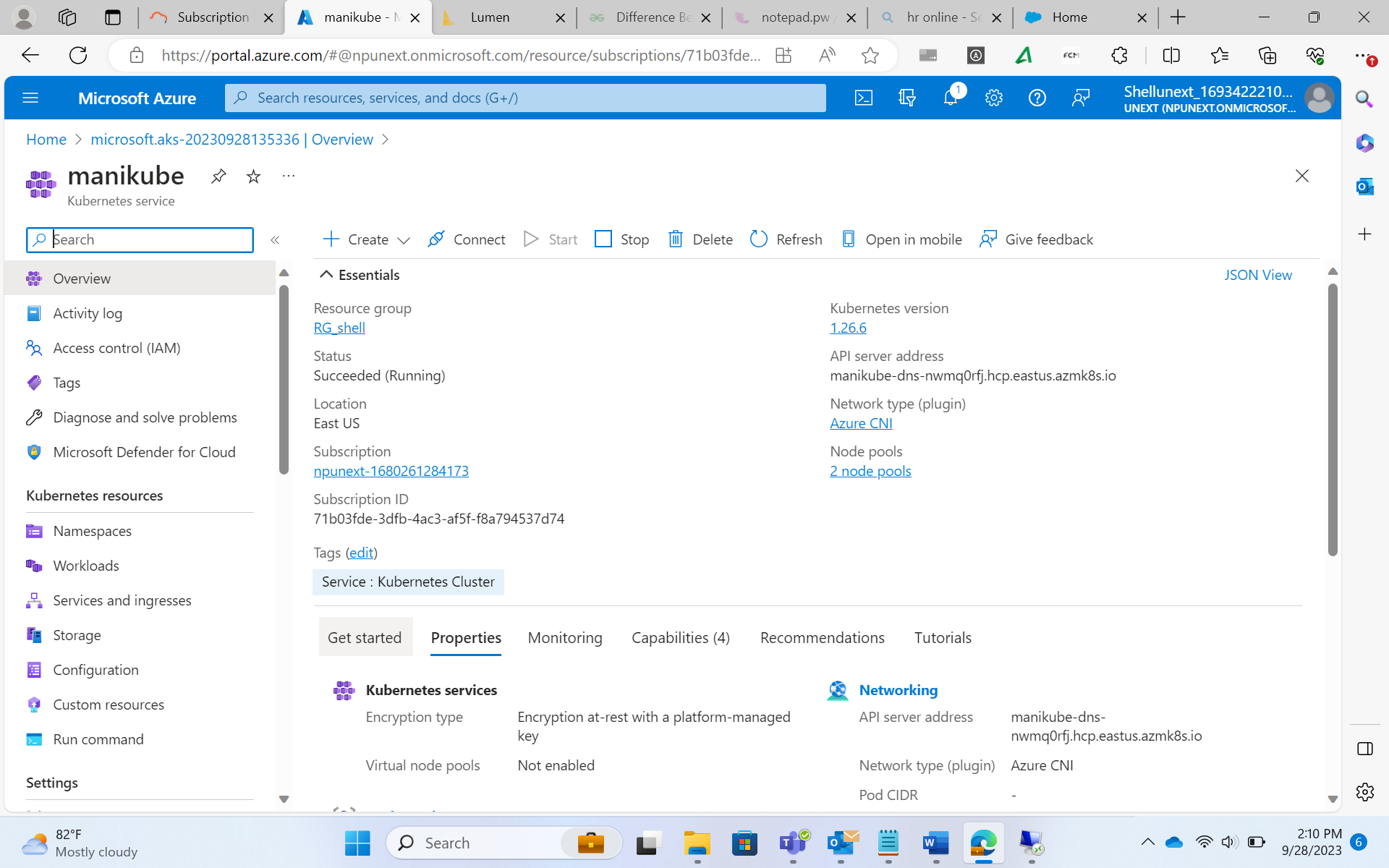
**sudo docker tag my-python-app idacrmani.azurecr.io/shellida:v1**

Final pushed image into docker acr can be seen below



Kubernetes

In Azure Portal, create a Kubernetes Cluster as shown below



* Create a folder called Kubernetes in the VM
* Add two yaml files: deployment.yaml, and service.yaml
* Add the content in the yaml files
* Run the kube control(kubectl) command from inside the Kubernetes directory for deployment.yaml

**kubectl apply -f deployment.yaml**

* Run the kube control command from inside the Kubernetes directory for deployment.yaml

**kubectl apply -f service.yaml**

* attach the kuberneted cluster with the acr

**az aks get-credentials --resource-group RG\_shell --name manikube**

**az aks update -n manikube -g RG\_shell --attach-acr idacrmani**

* Get the IP address of the Kubernetes cluster

**kubectl get svc my-python-app**

or

use the Azure Portal to connect the image to the Kubernetes Cluster

* create a starter application
* create a single image application
* view application