B.M.S. COLLEGE OF ENGINEERING

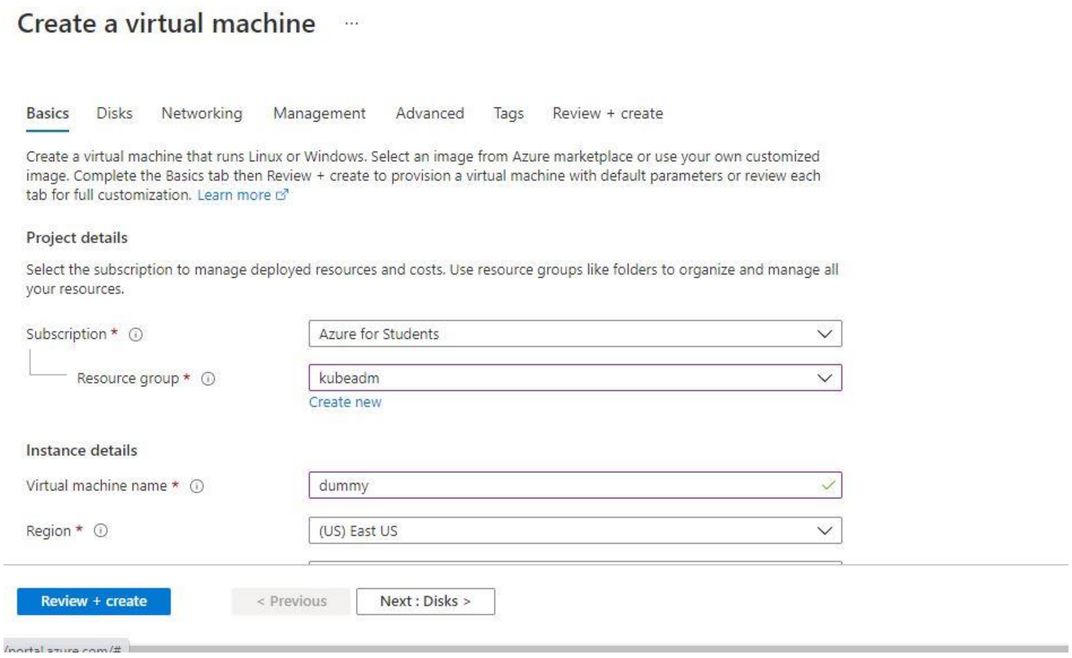
Autonomous Institute, Affiliated to VTU

Estd. 1946

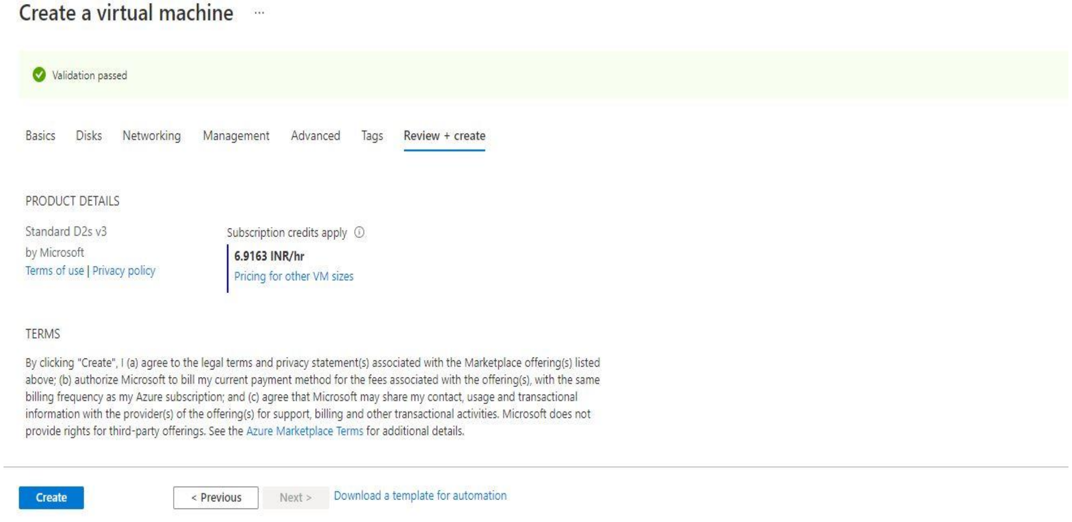
DEPARTMENT OF CSE

CTY Project Work In collaboration with HPE

Project Title	Open source monitoring and observability stack on Kubernetes		
Student Team	Name: Pooja Srinivasan USN:1BM18CS069 SEM:UG-6th SEM	Name: Anusree Manoj K USN:1BM18CS017 SEM:UG-6th SEM	
	Name: Shikha N USN:1BM18CS149 SEM:UG-6th SEM	Name: Niha USN:1BM18CS060 SEM:UG-6th SEM	
Faculty Mentor	Dr. Nandhini V Associate Professor	**HPE Mentors**	Divakar Padiyar Sonu Sudhakaran
Review for the Period	19-03-2021	25-03-2021	
Task Given	Implementing a K8 Cluster		
Difficulties Faced	Virtual Machines unable to communicate with each other.		
Libraries Used	None		
Github Link for the code:	None		
Code: Implementation in Microsoft Azure.			
Steps for installation Of K8 cluster [1]	Steps for Installation: 1. Create Virtual Machines 2. Virtual machines are created in a virtual network so that they communicate with each other 3. Install Docker,Kubelet,Kubeadm,Kubectl 4. Initialize and join the k8s clusters using Kubeadm 5. Install CNI (Weave) for Cluster DNS and Pod communication 6. Check Node and Pod Status		
Creating Virtual	Note:-We are using Microsoft Azure as our cloud platform.		

Machines [1]	1. For creating a Virtual Machine click on “Virtual Machines” and click on the add option to create a virtual machine. 2. This displays a list of options to personalize and build our VM. 3. For the first vm, we created a resource group and selected it, and added instance details such as vm name, region, availability option, image, size. 4. Then we entered the administrator account details which are authentication type which set it with an ssh public key source or password, in our case we have set it up with a password. 5. The basic settings can be changed as per need but in our case it is retained with the default settings. 6. We can configure the advanced options like networking,disc,management etc.We have retained it with default settings and only configured the basic settings. 7. Once the machine is configured as per our requirement we can review the same,create and deploy it. 8. In the same manner we have created 3 machines. One is a master node and the other 2 are worker nodes. 9. Typically, we can create all 3 nodes in the same region but due to limitations of our account we have created the two worker nodes in separate regions.  Create a virtual machine ··· Basics Disks Networking Management Advanced Tags Review + create Create a virtual machine that runs Linux or Windows. Select an image from Azure marketplace or use your own customized image. Complete the Basics tab then Review + create to provision a virtual machine with default parameters or review each tab for full customization. Learn more Project details Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources. Subscription * : Azure for Students Resource group * : kubeadm Create new Instance details Virtual machine name * : dummy ✓ Region * : (US) East US /portal.azure.com/#

<table>
<tr>
<td></td>
<td>



Create a virtual machine

Validation passed

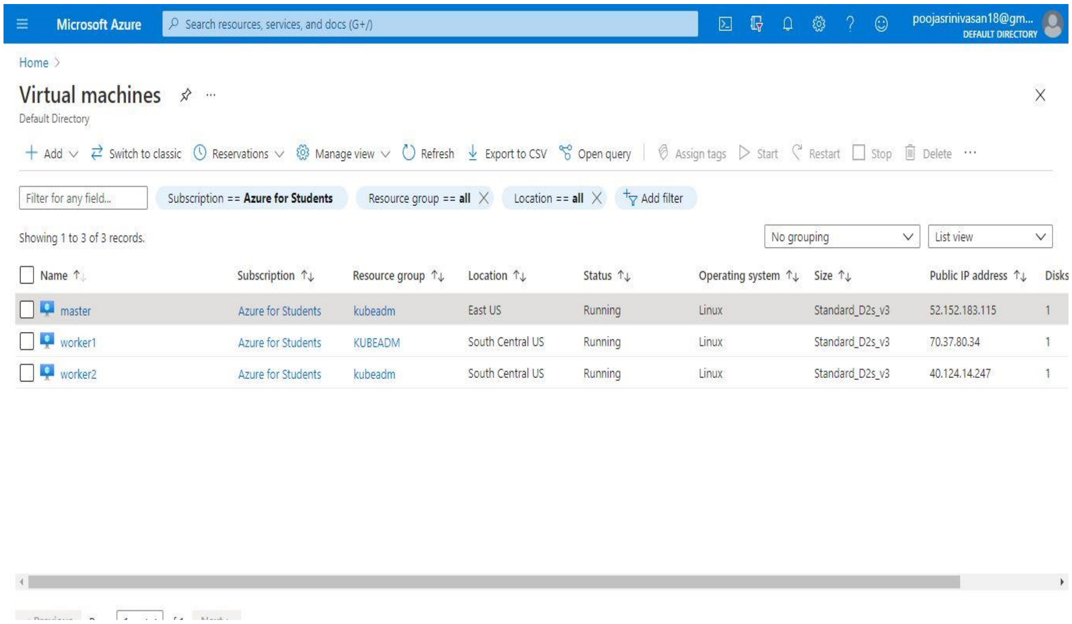
PRODUCT DETAILS

Standard D2s v3
by Microsoft
Terms of use | Privacy policy

Subscription credits apply
6.9163 INR/hr
Pricing for other VM sizes

TERMS

By clicking "Create", I (a) agree to the legal terms and privacy statement(s) associated with the Marketplace offering(s) listed above; (b) authorize Microsoft to bill my current payment method for the fees associated with the offering(s), with the same billing frequency as my Azure subscription; and (c) agree that Microsoft may share my contact, usage and transactional information with the provider(s) of the offering(s) for support, billing and other transactional activities. Microsoft does not provide rights for third-party offerings. See the Azure Marketplace Terms for additional details.



Virtual machines
Default Directory

Subscription == Azure for Students | Resource group == all | Location == all

Showing 1 to 3 of 3 records.

Name	Subscription	Resource group	Location	Status	Operating system	Size	Public IP address	Disks
master	Azure for Students	kubeadm	East US	Running	Linux	Standard_D2s_v3	52.152.183.115	1
worker1	Azure for Students	KUBEADM	South Central US	Running	Linux	Standard_D2s_v3	70.37.80.34	1
worker2	Azure for Students	kubeadm	South Central US	Running	Linux	Standard_D2s_v3	40.124.14.247	1

</td>
</tr>
<tr>
<td>

To allow the machines to communicate with each other [1]

</td>
<td>

A virtual network is created so that the machines can communicate with each other. In our case we have two networks namely:
1.kubeadm (for master node)
2.kubeadmvnet505 (for worker node)
Now we want our master node to communicate with the worker nodes hence we add a network peering (for the creation of a virtual network) to kubeadm.
3.For the machines to communicate with each other we add inbound rules to the network.Since we are creating the inbound rules for testing purposes we allow all incoming ports.The inbound rule is added to all the virtual machines since they have to communicate with each other.

To check if the machines can communicate with each other we use PuTTY (an open

</td>
</tr>
</table>

source terminal emulator).

We now use the ssh protocol to do so.We use the public IP address of the master and worker node to get them running on our physical machine.The we can communication can be achieved between the master and worker

Commands used are as follows:

1.For login give a suitable name.We have given our login as “ubuntu”

2.Enter the password of the machine which we have initially set while creating the machine under authentication.

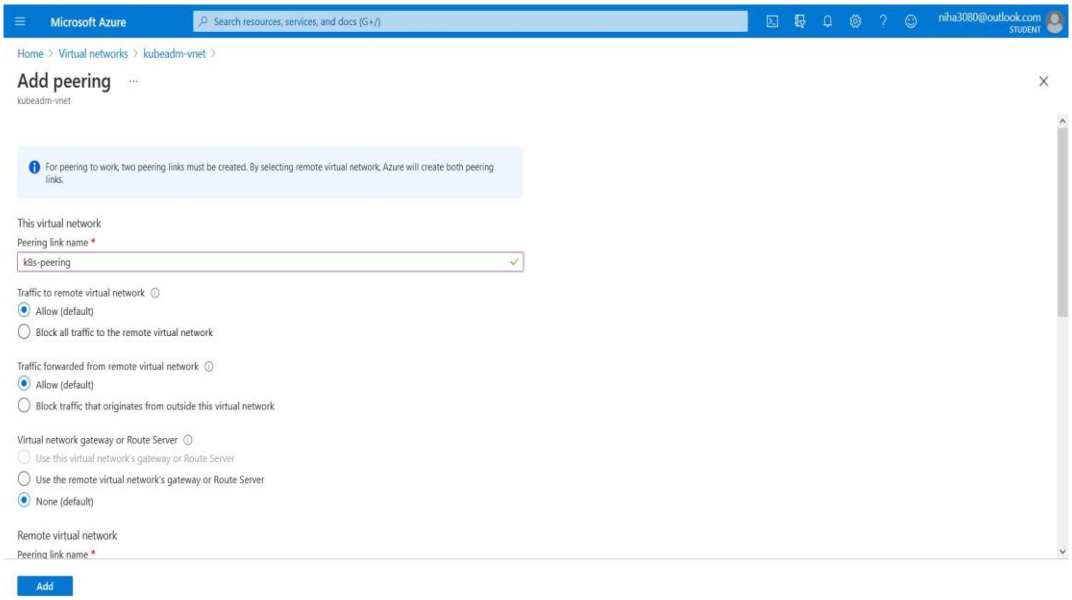
3.hostename (this gives the name of the vm)

4.sudo su - (change directory to master)

5.ssh ubuntu@public ip address of worker node

6.We accept the fingerprint to continue

Communication is established.


Microsoft Azure
Search resources, services, and docs (G+/)
niha3080@outlook.com
STUDENT
Home > Virtual networks > kubeadm-vnet >

Add peering

kubeadm-vnet

For peering to work, two peering links must be created. By selecting remote virtual network, Azure will create both peering links.

This virtual network

Peering link name *: k8s-peering

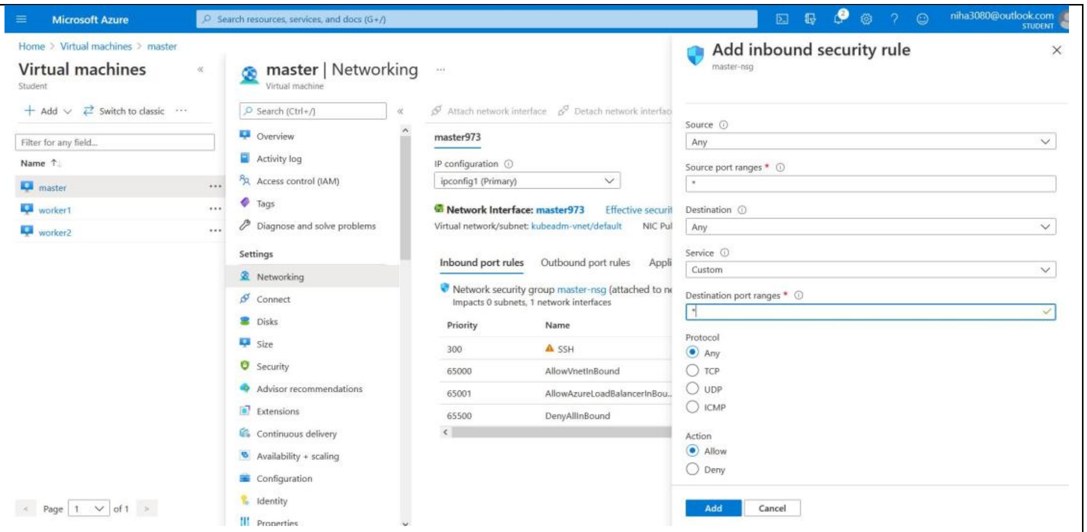
Traffic to remote virtual network
● Allow (default)
○ Block all traffic to the remote virtual network

Traffic forwarded from remote virtual network
● Allow (default)
○ Block traffic that originates from outside this virtual network

Virtual network gateway or Route Server
○ Use this virtual network's gateway or Route Server
○ Use the remote virtual network's gateway or Route Server
● None (default)

Remote virtual network

Peering link name *



Priority	Name
300	⚠ SSH
65000	AllowVnetInBound
65001	AllowAzureLoadBalancerInBou...
65500	DenyAllInBound

Add inbound security rule

master-nsg

Source ⓘ: Any

Source port ranges * ⓘ: *

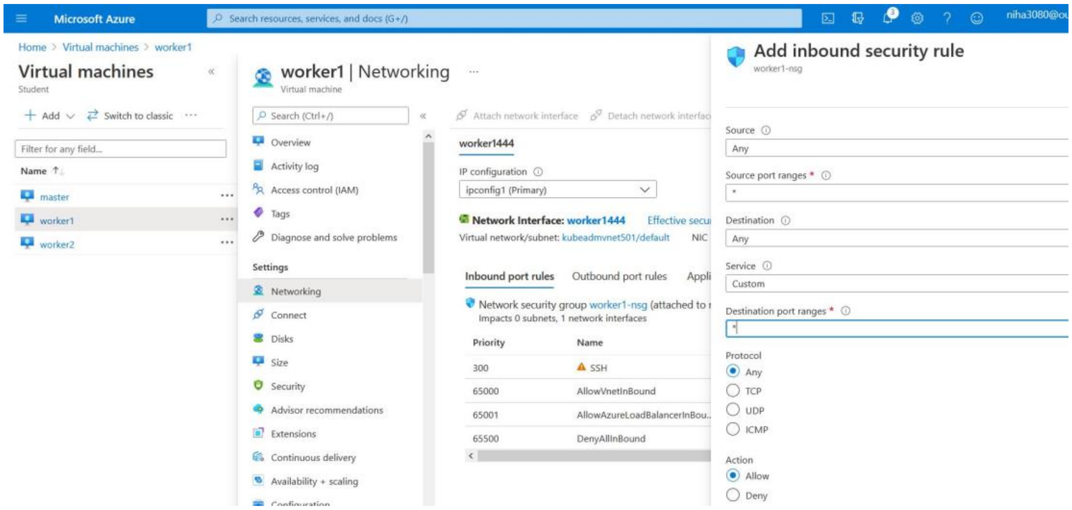
Destination ⓘ: Any

Service ⓘ: Custom

Destination port ranges * ⓘ: *

Protocol
- ● Any
- ○ TCP
- ○ UDP
- ○ ICMP

Action
- ● Allow
- ○ Deny



Priority	Name
300	⚠ SSH
65000	AllowVnetInBound
65001	AllowAzureLoadBalancerInBou...
65500	DenyAllInBound

Add inbound security rule

worker1-nsg

Source ⓘ: Any

Source port ranges * ⓘ: *

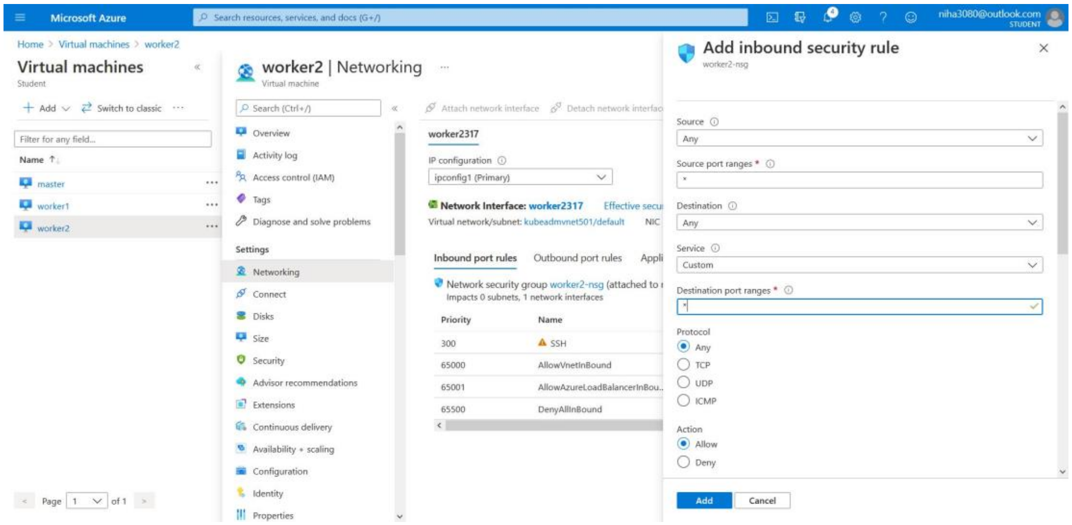
Destination ⓘ: Any

Service ⓘ: Custom

Destination port ranges * ⓘ: *

Protocol
- ● Any
- ○ TCP
- ○ UDP
- ○ ICMP

Action
- ● Allow
- ○ Deny



Priority	Name
300	⚠ SSH
65000	AllowVnetInBound
65001	AllowAzureLoadBalancerInBou...
65500	DenyAllInBound

Add inbound security rule

worker2-nsg

Source ⓘ: Any

Source port ranges * ⓘ: *

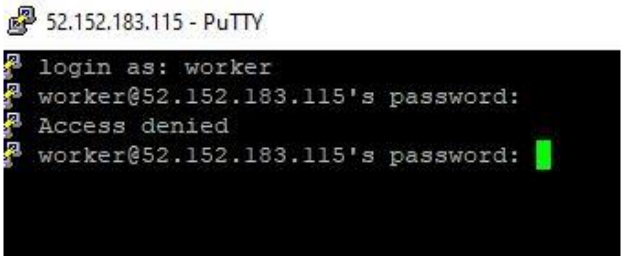
Destination ⓘ: Any

Service ⓘ: Custom

Destination port ranges * ⓘ: *

Protocol
- ● Any
- ○ TCP
- ○ UDP
- ○ ICMP

Action
- ● Allow
- ○ Deny

	 52.152.183.115 - PuTTY login as: worker worker@52.152.183.115's password: Access denied worker@52.152.183.115's password:
Install Docker,Kubelet,Kubeadm,Kubectl [1]	After creating a virtual machine and establishing connection, we need to install some packages on master as well as worker nodes. ● **Docker** – is a software responsible for running the containers. ● **kubeadm** – a CLI tool that will install and configure the various components of a cluster in a standard way. ● **kubelet** – a system service/program that runs on all nodes and handles node-level operations. ● **kubectl** – a CLI tool used for issuing commands to the cluster through its API Server. We have installed them using the following commands: ● Step1: We have to SSH to our virtual machines with the username and password. So to access our nodes we are using Putty as we are working on windows ● Step2: Then, we installed docker package to all three nodes using the following command:**apt-get update && apt-get install-y docker.io** ● Step3: After installation of docker is done, we installed Kubelet, Kubeadm, and kubectl packages into our machines using the following command: **apt-get install -y kubelet kubeadm kubectl** And for configuration settings for Kubelet, Kubeadm, and kubectl **echo"Environment="KUBELET_CGROUP_ARGS--cgroup-driver-cgroupfs" >>letc/systemd/system/kubelet.service.d/10-kubeadm.conf**
Commands to initialize master and join worker [1]	1. kubeadm init Note: If cluster initialisation has succeeded, then we will see a cluster join command.This command will be used by the worker nodes to join the Kubernetes cluster, 2. Kube join followed by output received in the previous command. 3. To use the cluster we need to set the environment variables. Join Worker Nodes to the Kubernetes Cluster 1. SSH into the Worker node with the username and password. **ssh <external ip of worker node>** 2. Run the **kubeadm join** command that we have received and saved.

Testing the Kubernetes Cluster[1]	Once the cluster is created and joined with the worker nodes,the nodes are tested. 1. The status of the node is checked using the following command: **kubectl get nodes** 2. Next the below command is executed if the status is not ready: **kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version= $(kubectl version | base64 | tr -d '\n')"** 3. After waiting for a few minutes and verifying the cluster status by using the **kubectl get nodes** on master node again,it is noticed that nodes come to the ready state. 4. To verify the status of the system pods like coreDNS, weave-net, Kube-proxy, and all other master node system processes,the following command is used:**kubectl get pods -n kube-system**
References	1. https://k21academy.com/docker-kubernetes/three-node-kubernetes-cluster/