**Azure Kubernetes Service**

**Azure Kubernetes Service** is the quickest path from zero to Kubernetes on Azure. This new service features an Azure-hosted control plane, automated upgrades, self-healing, easy scaling, and simple user experience for both developers and cluster operators.

To help you get started with AKS, the managed Kubernetes service is free. You only pay for the VMs in your cluster, and any infrastructure resources consumed like storage and networking. This means on Azure, you will pay nothing for the management of your Kubernetes cluster.

**\* AKS is currently available for Linux virtual machines.**

Creating an AKS cluster typically takes a few minutes. Once complete, you can access and manage your cluster through "az aks" and "kubectl".

It also eliminates the burden of ongoing operations and maintenance by provisioning, upgrading, and scaling resources on demand, without taking your applications offline.

**We are using the Command line tool (Power Shell ) for now.**

**Step 1: We need to create a Resource Group.**

Resource groups (RG)?

Ø Resource groups (RG) in Azure is a new approach to group a collection of assets in logical groups for easy or even automatic provisioning, monitoring, and access control, and for more effective management of their costs.

Ø One benefit of using RGs in Azure is grouping related resources that belong to an application together, as they share a unified lifecycle from creation to usage and finally, de-provisioning.

**PS /home/abhinav> az group create --name TestAKS --location eastus**

*{*

*"id": "/subscriptions/359459cc-dd4e-40b0-9ca4-8cfa604036b6/resourceGroups/TestAKS",*

*"location": "eastus",*

*"managedBy": null,*

*"name": "TestAKS",*

*"properties": {*

*"provisioningState": "Succeeded"*

*},*

*"tags": null,*

*"type": "Microsoft.Resources/resourceGroups"*

*}*

Ø **Get-AzResourceGroup || To check AzResourceGroup information**

ResourceGroupName : TestAKS

Location         : eastus

ProvisioningState : Succeeded

Tags             :

ResourceId       : /subscriptions/359459cc-dd4e-40b0-9ca4-8cfa604036b6/resourceGroups/TestAKS

**Step 2:**Creating a cluster having cluster name**‘TestAKScluster’,**Number of node taking**‘2’,** Enabling Monitoring as well and doing all the things under **‘TestAKS**’ Resource Group.

Ø **PS /home/abhinav> az aks create --resource-group TestAKS --name TestAKScluster --node-count 2 --enable-addons monitoring --generate-ssh-**keys  ( It will take some time around 5 mins, you will see output like this).

**SSH key files '/home/abhinav/.ssh/id\_rsa' and '/home/abhinav/.ssh/id\_rsa.pub' have been generated under ~/.ssh to allow SSH access to the VM. Ifusing machines without permanent storage like Azure Cloud Shell without an attached file share, back up your keys to a safe location**

**AAD role propagation done[############################################] 100.0000%{**

**"aadProfile": null,**

**"addonProfiles": {**

**"KubeDashboard": {**

**"config": null,**

**"enabled": true,**

**"identity": null**

**},**

**"omsagent": {**

**"config": {**

**"logAnalyticsWorkspaceResourceID": "/subscriptions/359459cc-dd4e-40b0-9ca4-8cfa604036b6/resourcegroups/defaultresourcegroup-eus/providers/microsoft.operationalinsights/workspaces/defaultworkspace-359459cc-dd4e-40b0-9ca4-8cfa604036b6-eus"**

**},**

**"enabled": true,**

**"identity": null**

**}**

**},**

**"agentPoolProfiles": [**

**{**

**"availabilityZones": null,**

**"count": 2,**

**"enableAutoScaling": null,**

**"enableNodePublicIp": false,**

**"maxCount": null,**

**"maxPods": 110,**

**"minCount": null,**

**"mode": "System",**

**"name": "nodepool1",**

**"nodeLabels": {},**

**"nodeTaints": null,**

**"orchestratorVersion": "1.15.11",**

**"osDiskSizeGb": 128,**

**"osType": "Linux",**

**"provisioningState": "Succeeded",**

**"scaleSetEvictionPolicy": null,**

**"scaleSetPriority": null,**

**"spotMaxPrice": null,**

**"tags": null,**

**"type": "VirtualMachineScaleSets",**

**"vmSize": "Standard\_D2s\_v3",**

**"vnetSubnetId": null**

**}**

**],**

**"apiServerAccessProfile": null,**

**"autoScalerProfile": null,**

**"diskEncryptionSetId": null,**

**"dnsPrefix": "TestAKSclu-TestAKS-359459",**

**"enablePodSecurityPolicy": null,**

**"enableRbac": true,**

**"fqdn": "testaksclu-testaks-359459-d7777bcf.hcp.eastus.azmk8s.io",**

**"id": "/subscriptions/359459cc-dd4e-40b0-9ca4-8cfa604036b6/resourcegroups/TestAKS/providers/Microsoft.ContainerService/managedClusters/TestAKScluster",**

**"identity": null,**

**"identityProfile": null,**

**"kubernetesVersion": "1.15.11",**

**"linuxProfile": {**

**"adminUsername": "azureuser",**

**"ssh": {**

**"publicKeys": [**

**{**

**"keyData": "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCmPLwkJ4y3UQIeKmMPWPQtAZXi+I2ZtaVgrfI6FbG/4cHJGAwkHBjDjDTwqep2FJZ47KDbEkOx9ZyQ5MnXG/cJtHT2nykHOutDmxtWrchVixPz3Om0M74TaoKEsEFeSTJ+LePmDmknlikuloHV8vIVFX2lv+xX2rM3CgjMYTcRKiT8fUJ7dQW3g3jBfZqEVH2sbUxSmwxbiGiiYuLhOAXFwRfOEAyh4yV6pbBAjjh98UXwyLXmGyIYzI0qRBp4fUrLE5El45m7lycmiCxPcFCiDiYytUwp36N8U0iSaK5T8xulhnyLjJhU7NBap6TABUWMbH3Z7HYQFLPy+3i0/1Ih"**

**}**

**]**

**}**

**},**

**"location": "eastus",**

**"maxAgentPools": 10,**

**"name": "TestAKScluster",**

**"networkProfile": {**

**"dnsServiceIp": "10.0.0.10",**

**"dockerBridgeCidr": "172.17.0.1/16",**

**"loadBalancerProfile": {**

**"allocatedOutboundPorts": null,**

**"effectiveOutboundIps": [**

**{**

**"id": "/subscriptions/359459cc-dd4e-40b0-9ca4-8cfa604036b6/resourceGroups/MC\_TestAKS\_TestAKScluster\_eastus/providers/Microsoft.Network/publicIPAddresses/33f772e4-577a-4af3-8424-d0c4ee0c7d0c",**

**"resourceGroup": "MC\_TestAKS\_TestAKScluster\_eastus"**

**}**

**],**

**"idleTimeoutInMinutes": null,**

**"managedOutboundIps": {**

**"count": 1**

**},**

**"outboundIpPrefixes": null,**

**"outboundIps": null**

**},**

**"loadBalancerSku": "Standard",**

**"networkMode": null,**

**"networkPlugin": "kubenet",**

**"networkPolicy": null,**

**"outboundType": "loadBalancer",**

**"podCidr": "10.244.0.0/16",**

**"serviceCidr": "10.0.0.0/16"**

**},**

**"nodeResourceGroup": "MC\_TestAKS\_TestAKScluster\_eastus",**

**"privateFqdn": null,**

**"provisioningState": "Succeeded",**

**"resourceGroup": "TestAKS",**

**"servicePrincipalProfile": {**

**"clientId": "6e1fe0b4-f236-4a12-883a-dcaffdf17391",**

**"secret": null**

**},**

**"sku": {**

**"name": "Basic",**

**"tier": "Free"**

**},**

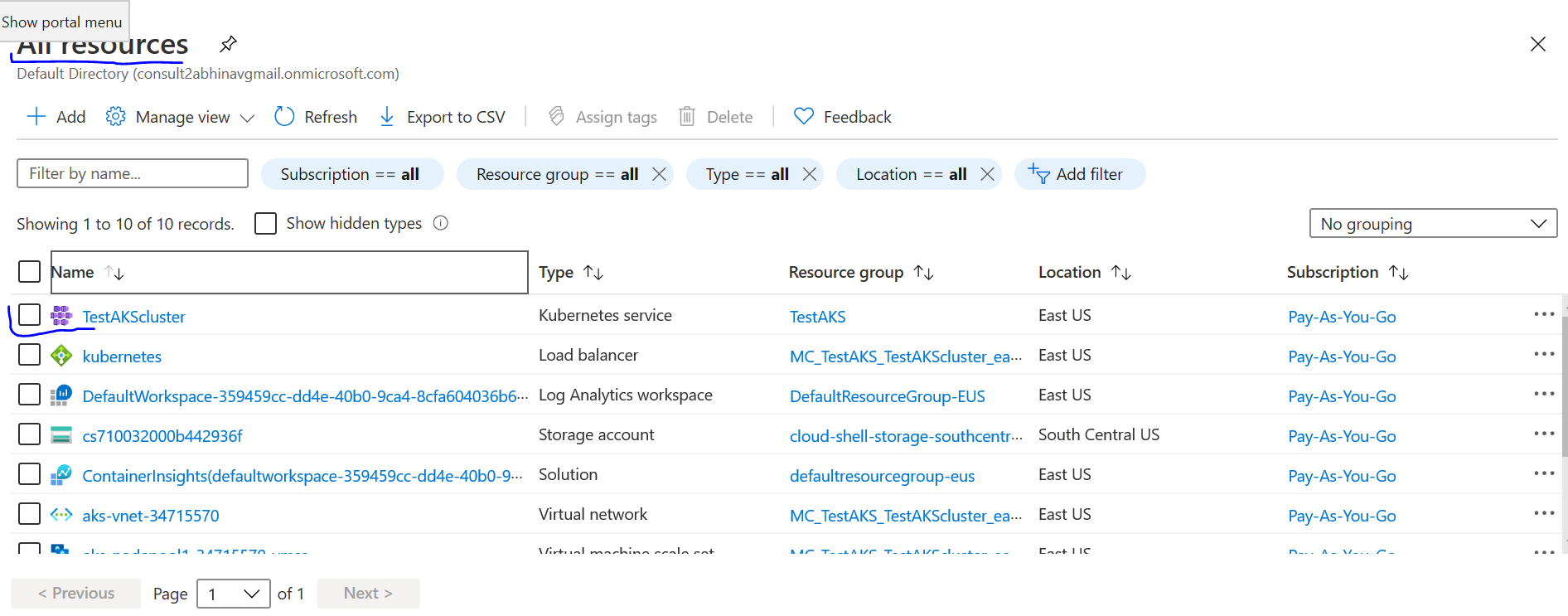
**"tags": null,**

**"type": "Microsoft.ContainerService/ManagedClusters",**

**"windowsProfile": null**

**}**

**Step 3: Now Go to Azure Portal check All Resources, you will see output like this.**



**Step 4: Now we need to install kubectl tool to work with cluster service.**

PS /home/abhinav> **az aks install-cli --install-location=./kubectl**

Downloading client to "./kubectl" from "https://storage.googleapis.com/kubernetes-release/release/v1.18.4/bin/linux/amd64/kubectl"

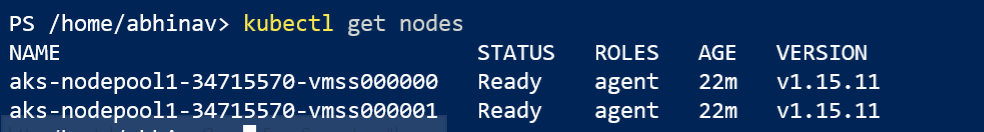
Please ensure that. Is in your search PATH, so the `kubectl` command can be found.

**Step 5: Now we will get the credentials of our cluster.**

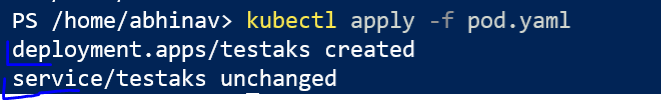
* PS /home/abhinav> **az aks get-credentials --resource-group TestAKS --name TestAKScluster**

Merged "TestAKScluster" as current context in /home/abhinav/.kube/config

**Step 6: You can get information about Nodes for this cluster via this command.**



**Step 7: We will deploy an apache container.**



This is the yaml file which I used.

Pod.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

 name: testaks

spec:

 replicas: 1

 selector:

   matchLabels:

     app: testaks

 template:

   metadata:

     labels:

       app: testaks

   spec:

     containers:

     - name: testaksserver

       image: httpd

       ports:

       - containerPort: 80

---

apiVersion: v1

kind: Service

metadata:

 name: testaks

spec:

 type: LoadBalancer

 ports:

 - port: 80

 selector:

   app: testaks

**Step 8: To see all resources of Kubernetes cluster which we have defined in our yaml file.**

As we can see our cluster (testaks) is having ip ‘External-IP’ means through which we will access what it has from external way, go to new explorer tab and hit this IP ‘http://52.146.56.90/’ **you will get a message like below, it means now we have the apache service running has a container Kubernetes cluster.**

