In this lab, you will learn how to create a new Azure Kubernetes Services (AKS) cluster using the Azure CLI. This enables you to host complex container-based applications in Azure.

Kubernetes (K8s) is an open source product for simplifying and automating deployment, scaling, and management of Docker-based containerized applications. Docker allows you to image an application consisting of tens or even hundreds of microservices, each deployed to a container, and Kubernetes allows you to deploy and manage these containers with ease.

So what is Azure Kubernetes Services (AKS)? In Microsoft's words, "Azure Kubernetes Service (AKS) simplifies deploying a managed Kubernetes cluster in Azure by offloading the operational overhead to Azure." In simpler words, Azure implemented a version of Kubernetes which is much easier to work with.

Learning Objectives
In this lab, you will learn how to do the following:

Log in to the Azure CLI
Understand Azure Kubernetes Services (AKS)
Provision an AKS cluster
Confirm that the AKS cluster was successfully created
Clean up the AKS cluster

Login to azure :
Az login -u $username -p $password

In this lab, we will provision a new Azure Kubernetes Services (AKS) cluster using Azure CLI.

Understanding AKS
AKS makes your job as a Kubernetes administrator much easier because you don't need to worry about managing the master nodes of your cluster. All you need to worry about are the user nodes, which host your application. Simply provision your AKS cluster and deploy your application to it.

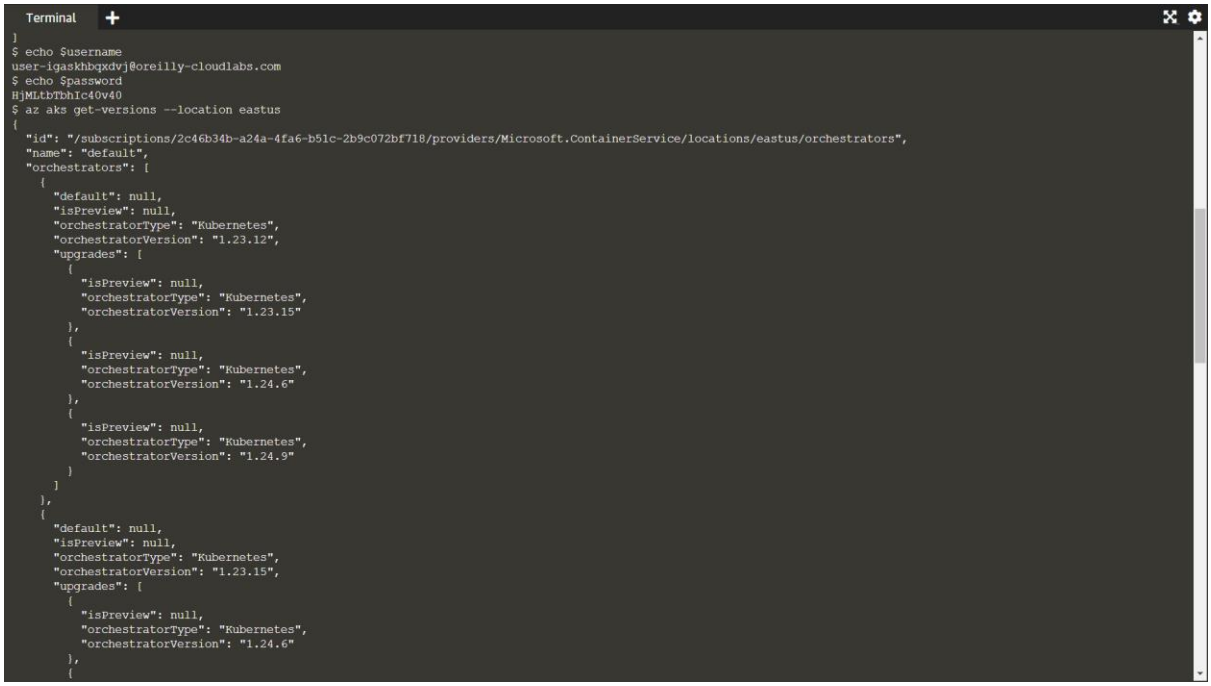
Before jumping into Azure CLI, it is important to have a good understating of core AKS concepts:

Kubernetes cluster: This is the AKS service at the parent level.
Control plane: This provides the core Kubernetes services and orchestration of workloads, a.k.a. master nodes. Azure will manage this portion for you.
Nodes: A Kubernetes node runs your container-based application. An AKS cluster has at least one node.
Node pools: A node pool is a group of nodes. An AKS cluster has at least one node pool.
Pods: A pod is generally mapped to a single container. A node can host multiple pods, and each pod can be accessed using a dedicated IP address.
Deployments via YAML manifests: A deployment contains the configuration details, and the list of containers to deploy. A deployment is represented using a YAML file, also known as a manifest.

Now, let's provision our first AKS cluster using Azure CLI in the next steps.

Provisioning an AKS Cluster
First you need to figure out which Kubernetes version is available in your location. You can use the following command to get the available Kubernetes versions:

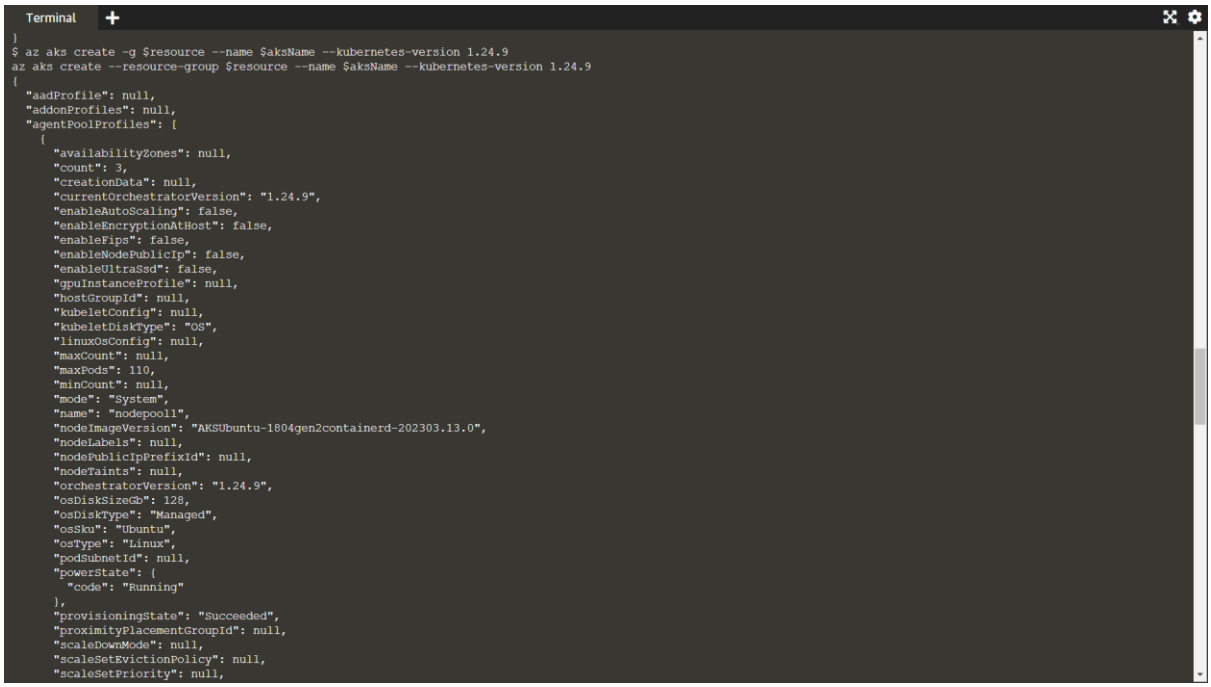
az aks get-versions --location eastus

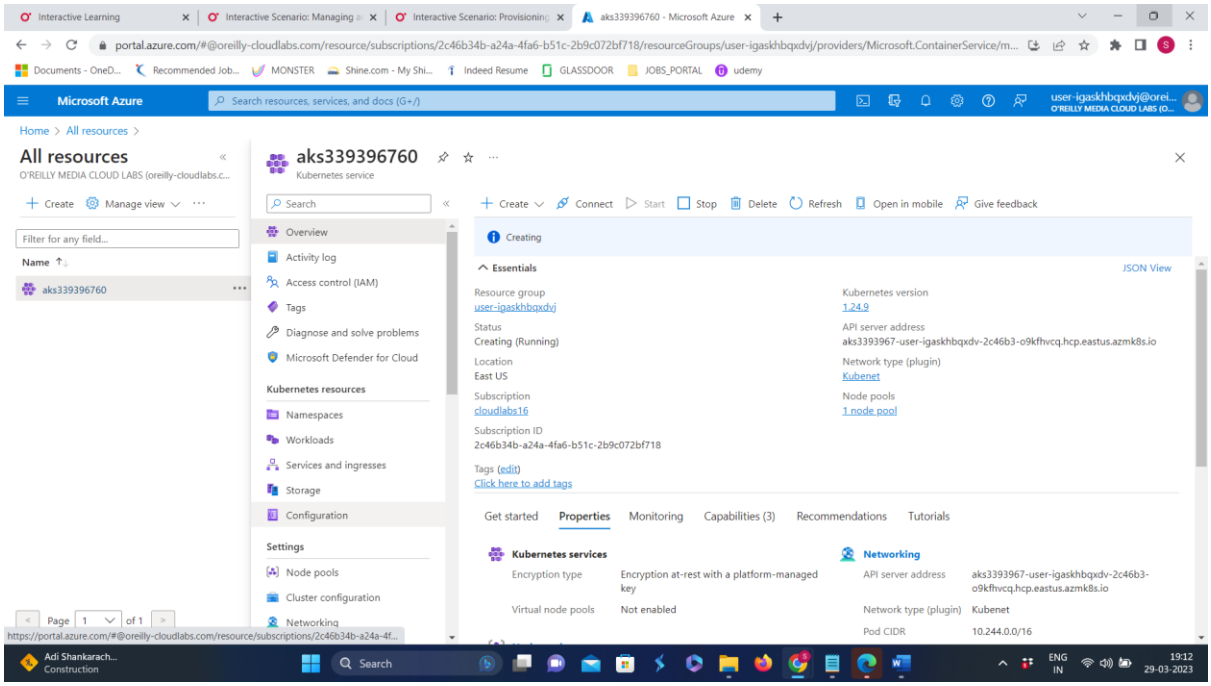


Now use the following command to create an AKS cluster. We are using version 1.24.9, as this was the latest available version for our location at the time:

az aks create --resource-group $resource --name $aksName --kubernetes-version 1.24.9
Let's take a look at the parameters for this command:

--resource-group: Name of the parent resource group
--kubernetes-version: Version of Kubernetes to use, for example "1.24.9"
--name: Name of the new AKS cluster



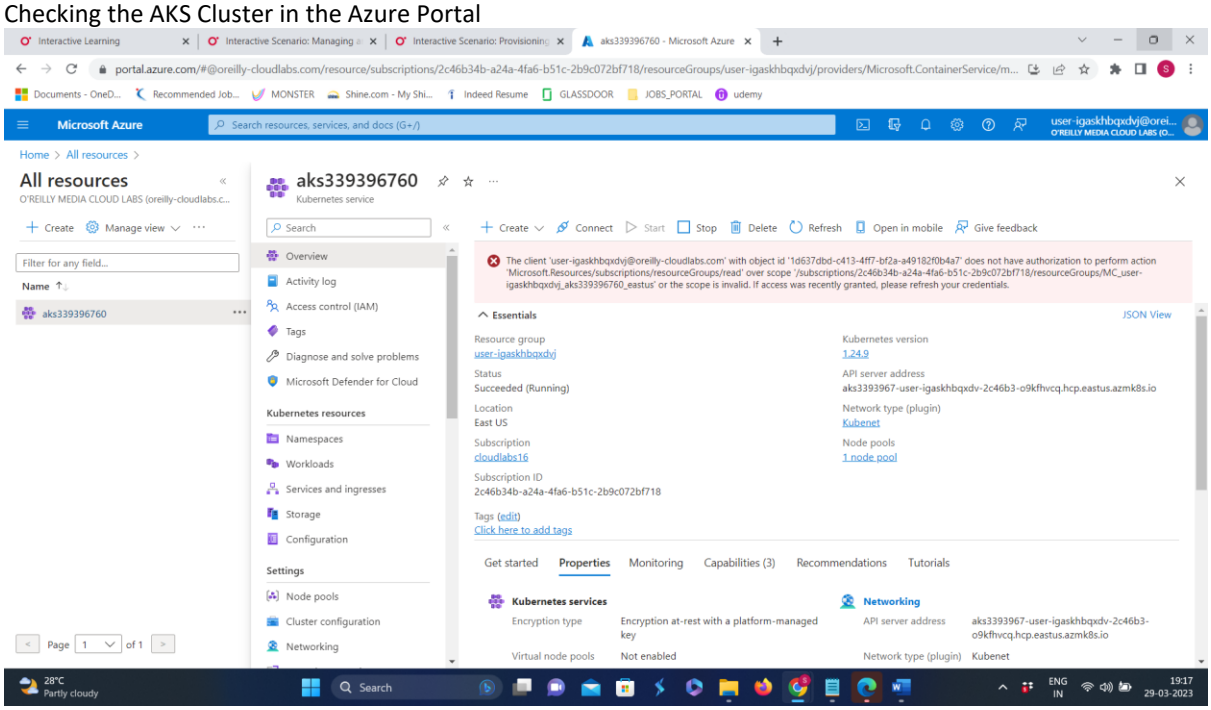


Confirming That the AKS Cluster Is Provisioned
Use the following command to list all the AKS clusters in your temporary Azure subscription:

az aks list --query "[].{Name: name}"

```
Long-running operation wait cancelled.
$ az aks list --query "[].{Name: name}"
[
  {
    "Name": "aks339396760"
  }
]
$
```

Checking the AKS Cluster in the Azure Portal



Deleting the AKS Cluster
The following command will clean up the new AKS cluster:

```
az aks delete --name $aksName --resource-group $resource
```

Now use the list command again to get all the AKS clusters in your temporary Azure subscription:

```
az aks list --query "[].{Name: name}"
```