

Table of Contents

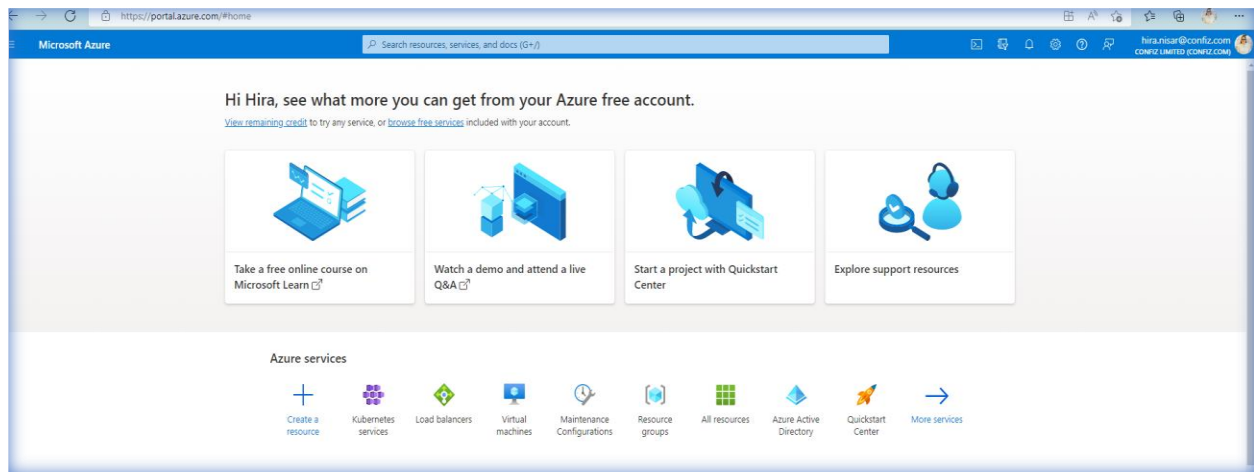
AZURE QUICK SETUP	2
Resource group	2
Virtual Machine.....	5
Kubernetes Service (Cluster).....	6
Install Kubeflow	10
Prerequisites	10
Installations of Prerequisites	10
Logging in the Azure.....	10
Connecting to the cluster.....	11
Installation of prerequisites	12
Kubeflow Installation	13
Set up and Kubeflow.	13
Deploy Kubeflow	13
Rules.....	14
Kustomization	14

DOCUMENTATION

AZURE KUBEFLOW

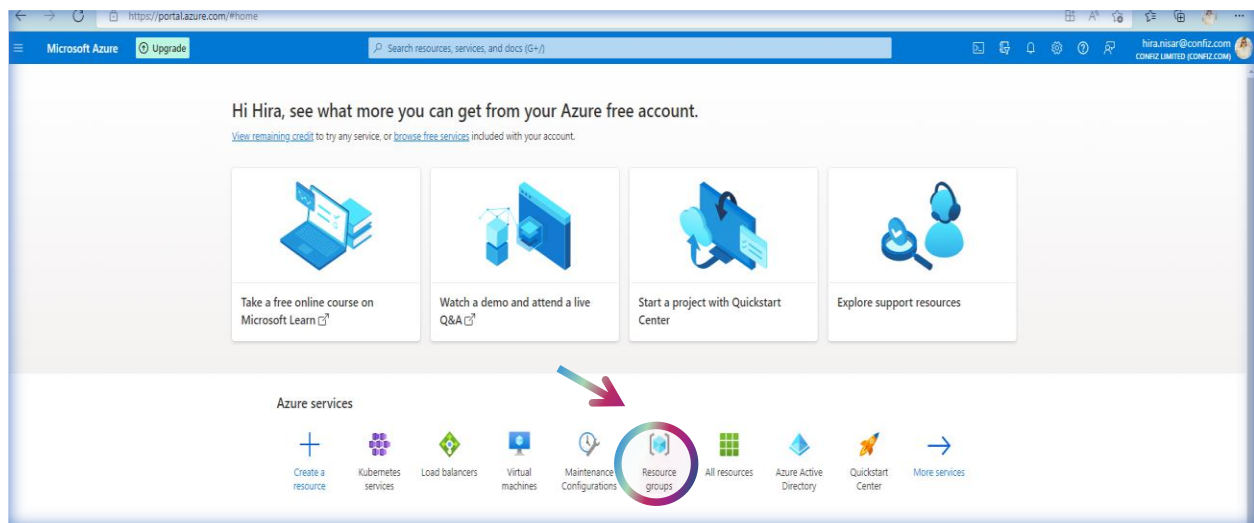
AZURE QUICK SETUP

- ✓ Open the **Azure**: <https://portal.azure.com>
- ✓ Register for a **subscription of a free account**, create your free account and access the **portal home page**.

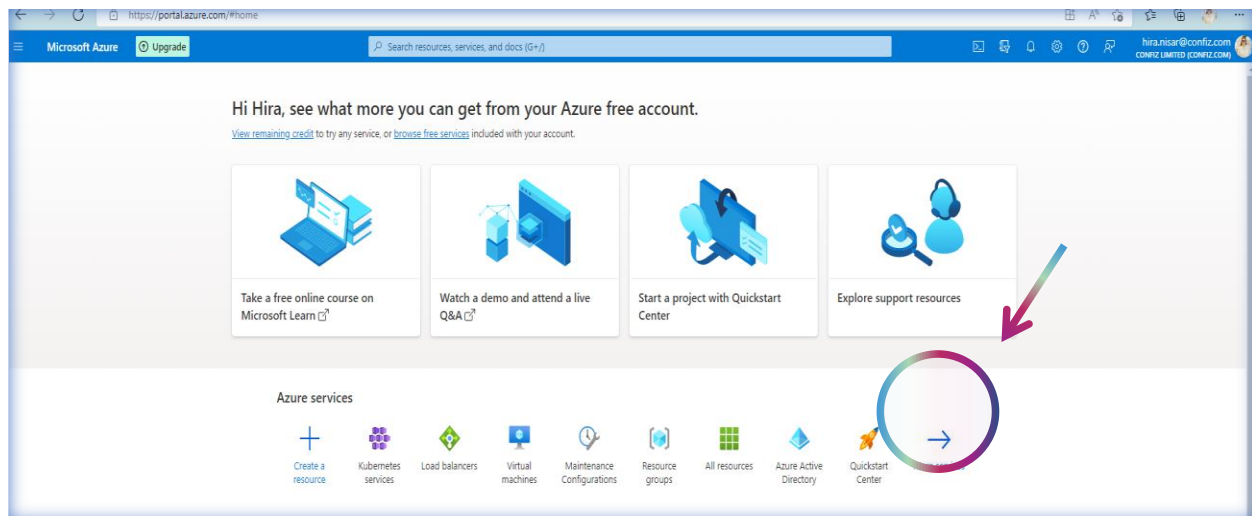


Resource group

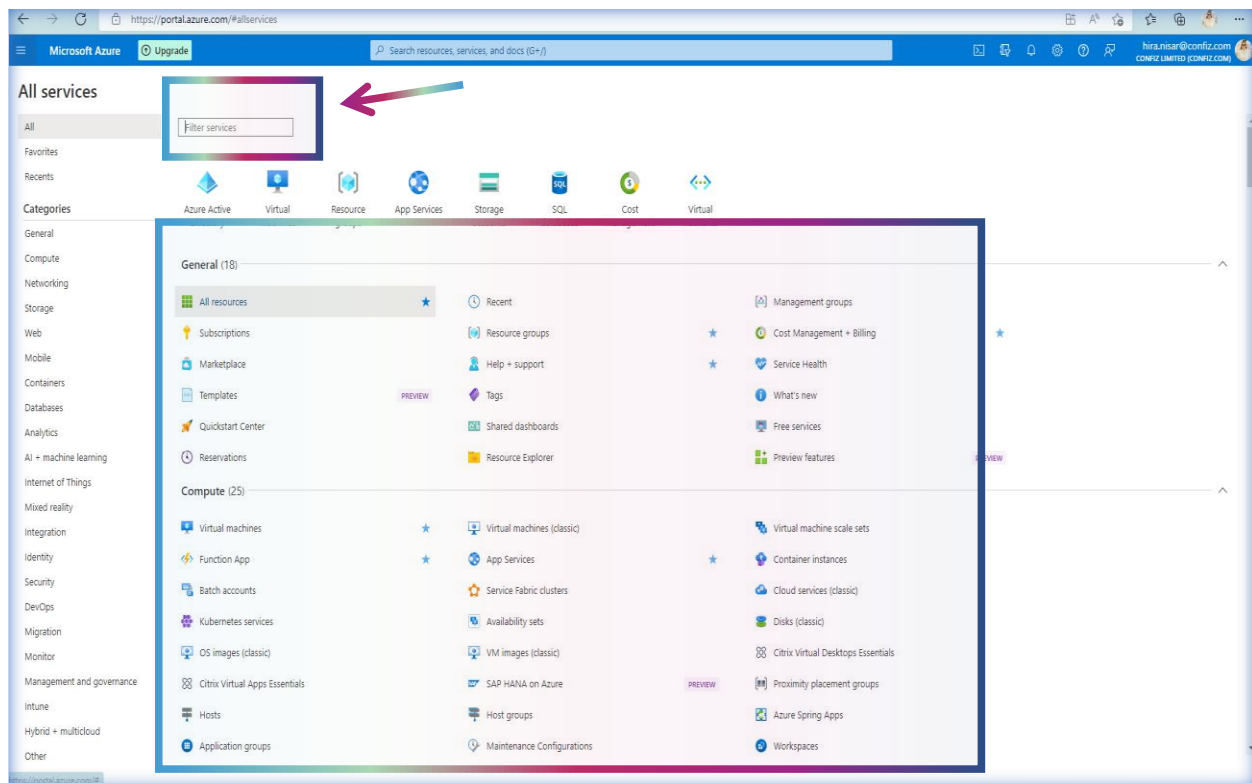
- ✓ Create a **resource group** with this **subscription**, with the highlighted icon on the screen.



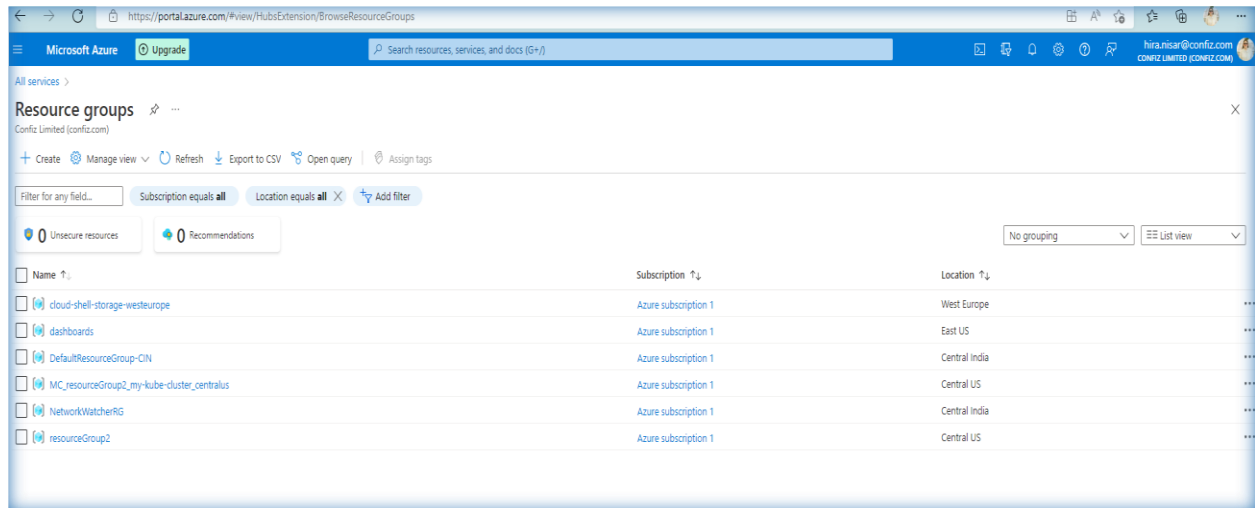
If you do not see that icon in the quick start icons, then browse the **“more services”** and manually create one.



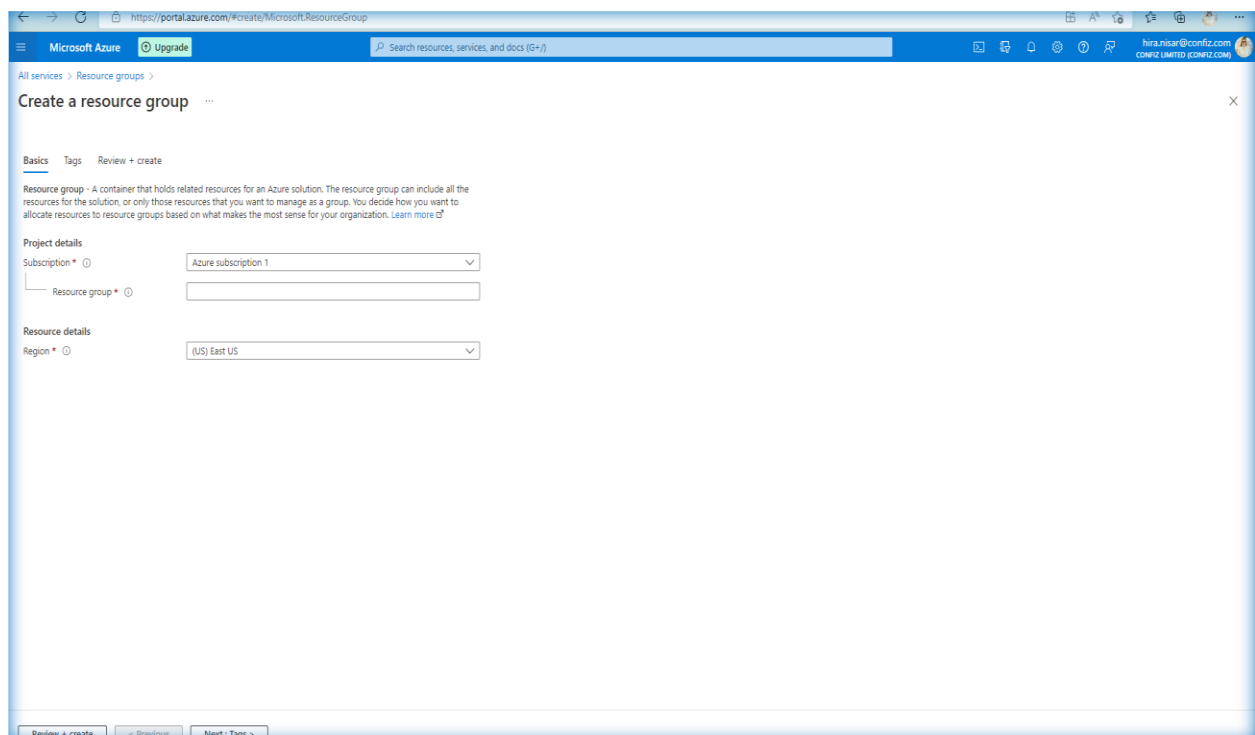
✓ Type in the field and search for a **“resource group”**.



- ✓ Here you can view and create your **resource group**.

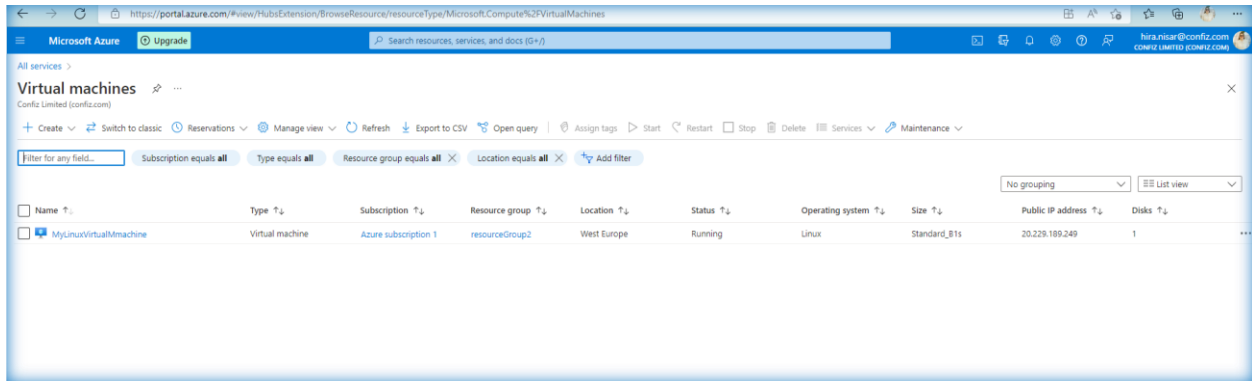


- ✓ Add in the details of your azure subscription and the name you want for your resource group and then follow the default settings.



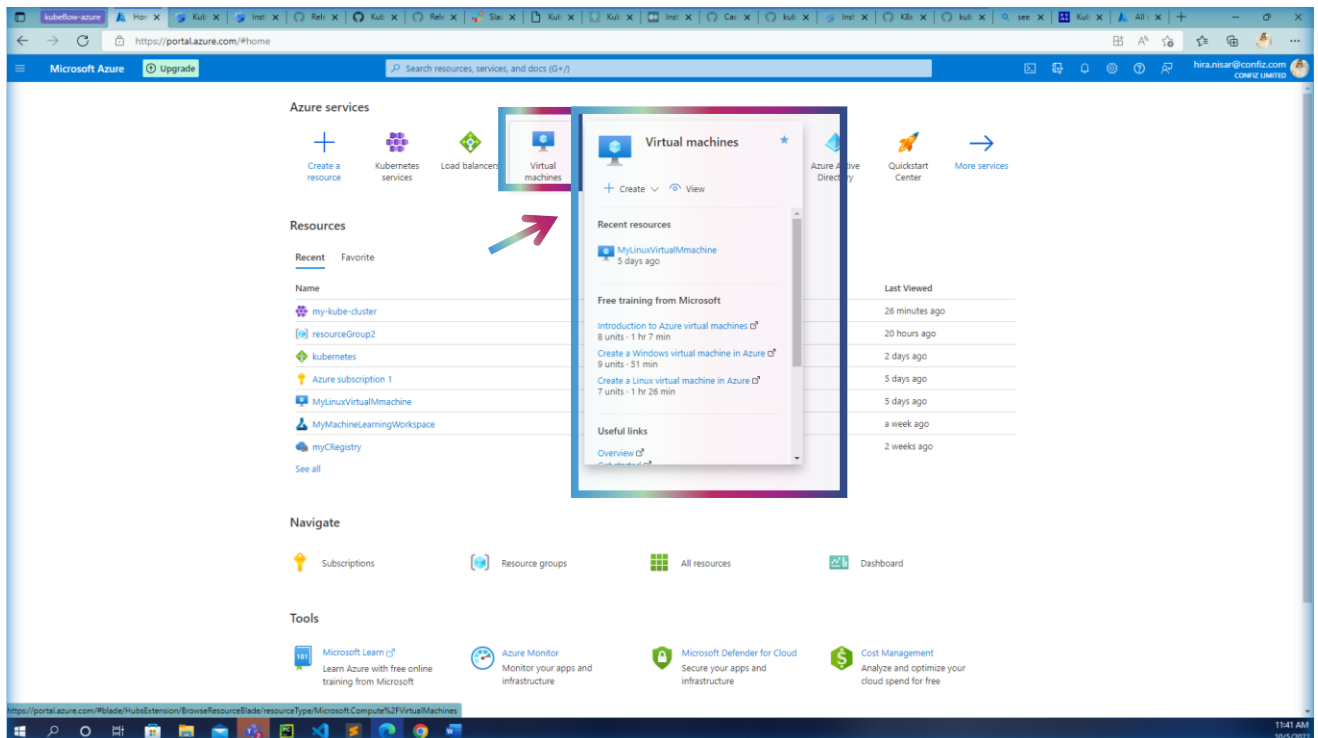
Virtual Machine

- ✓ Once its created and deployed, setup a **virtual machine** in the same way. You can view my virtual machine and itsb details below.



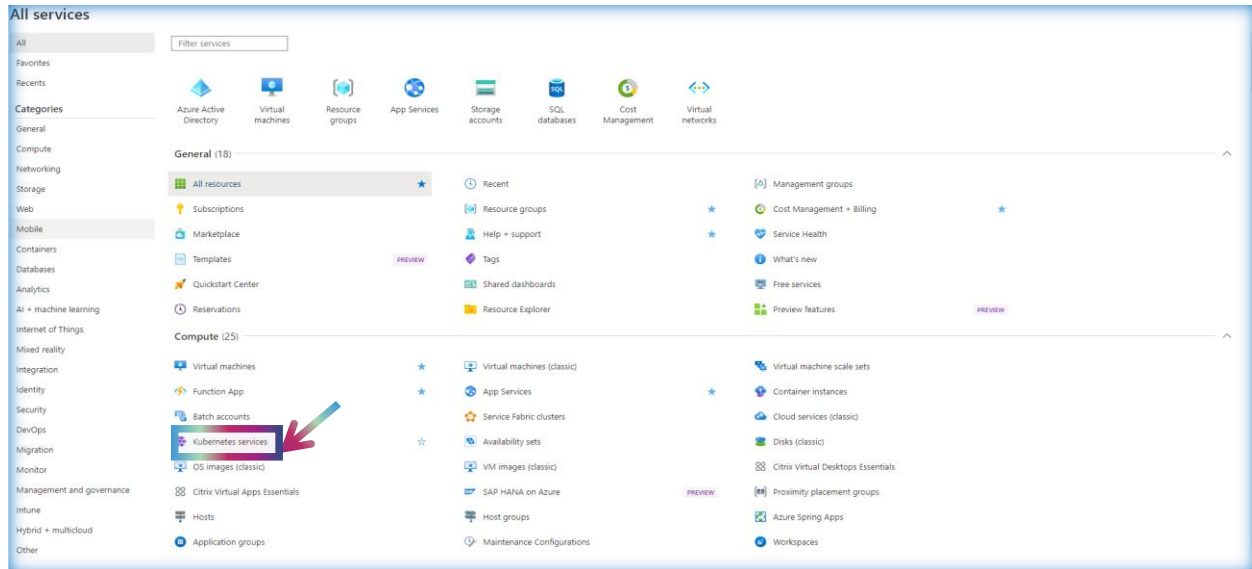
Name	Type	Subscription	Resource group	Location	Status	Operating system	Size	Public IP address	Disks
MyLinuxVirtualMachine	Virtual machine	Azure subscription 1	resourceGroup2	West Europe	Running	Linux	Standard_B1s	20.229.189.249	1

If you face any difficulty in creating a resource, just place the mouse over its icon and view the **documentation and tutorials** provided by azure.

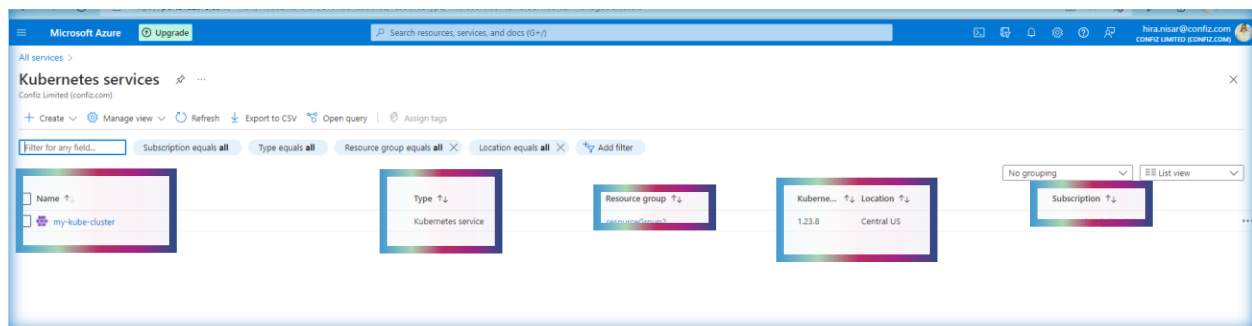


Kubernetes Service (Cluster)

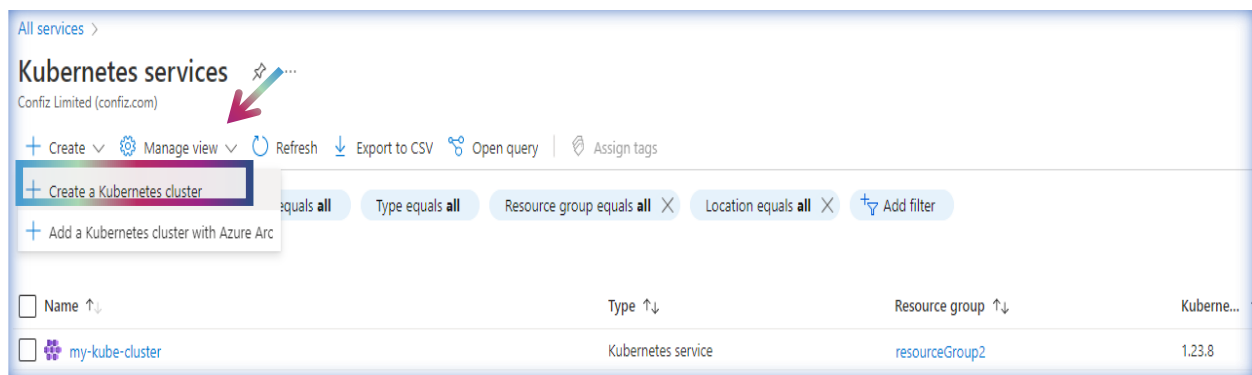
- ✓ Once done with these basis resources, you now need to create a **kubernetes service**.



- ✓ Add the details as shown in the following snapshot of my kube cluster and follow the default settings.



- ✓ You need to select the first option for this guide.



✓ Add 2 min and 2 max nodes.

As it is the free trial, you will be able to create only 2 nodes and with specific zones.

Microsoft Azure Upgrade Search resources, services, and docs (G+/)

All services > Kubernetes services >

Create Kubernetes cluster

Cluster preset configuration: Standard (\$\$)

To quickly customize your Kubernetes cluster, choose one of the preset configurations above. You can modify these configurations at any time. [Learn more and compare presets](#)

Kubernetes cluster name * ⓘ

Region * ⓘ: (US) East US

Availability zones ⓘ: Zones 1,2,3

Kubernetes version * ⓘ: 1.23.12 (default)

API server availability ⓘ:
☒ 99.95% Optimize for availability.
☐ 99.5% Optimize for cost.
99.95% API server availability is recommended for standard configuration.

Primary node pool

The number and size of nodes in the primary node pool in your cluster. For production workloads, at least 3 nodes are recommended for resiliency. For development or test workloads, only one node is required. If you would like to add additional node pools or to see additional configuration options for this node pool, go to the 'Node pools' tab above. You will be able to add additional node pools after creating your cluster. [Learn more about node pools in Azure Kubernetes Service](#)

Node size * ⓘ: Standard DS2 v2
Standard DS2_v2 is recommended for standard configuration. [Change size](#)

Scale method * ⓘ:
☐ Manual
☒ Autoscale
Autoscaling is recommended for standard configuration.

Node count range * ⓘ: 2 2

[Review + create](#) < Previous Next : Node pools >

- ✓ After this “Basics” section, go to the node pools section and create a node pool for 2 nodes and 110 pods for each node.

Microsoft Azure Upgrade Search resources, services, and docs (G+)

All services > Kubernetes services >

Create Kubernetes cluster ...

Basics **Node pools** Access Networking Integrations Advanced Tags Review + create

Node pools

In addition to the required primary node pool configured on the Basics tab, you can also add optional node pools to handle a variety of workloads [Learn more about node pools](#)

+ Add node pool Delete

Name	Mode	OS type	Node count	Node size
<input type="checkbox"/> agentpool	System	Linux	2-2	Standard_DS2_v2

Enable virtual nodes

Virtual nodes allow burstable scaling backed by serverless Azure Container Instances. [Learn more about virtual nodes](#)

Enable virtual nodes ☐

Node pool OS disk encryption

By default, all disks in AKS are encrypted at rest with Microsoft-managed keys. For additional control over encryption, you can supply your own keys using a disk encryption set backed by an Azure Key Vault. The disk encryption set will be used to encrypt the OS disks for all node pools in the cluster. [Learn more](#)

Encryption type (Default) Encryption at-rest with a platform-managed key

- ✓ For the access part

← → ↻ https://portal.azure.com/#create/microsoft.aks

Microsoft Azure Upgrade Search resources, services, and docs (G+)

All services > Kubernetes services >

Create Kubernetes cluster ...

Basics Node pools **Access** Networking Integrations Advanced Tags Review + create

Resource identity ☐ System-assigned managed identity
By default, Azure uses a managed identity. To use a service principal, use the CLI. [Learn more](#)

Choose between local accounts or Azure AD for authentication and Azure RBAC or Kubernetes RBAC for your authorization needs.

Authentication and Authorization ☐ Local accounts with Kubernetes RBAC

i Once the cluster is deployed, use the Kubernetes CLI to manage RBAC configurations. [Learn more](#)

✓ For networking

All services > Kubernetes services >

Create Kubernetes cluster

Basics Node pools Access **Networking** Integrations Advanced Tags Review + create

You can change networking settings for your cluster, including enabling HTTP application routing and configuring your network using either the 'Kubenet' or 'Azure CNI' options:

- The **kubenet** networking plug-in creates a new VNet for your cluster using default values.
- The **Azure CNI** networking plug-in allows clusters to use a new or existing VNet with customizable addresses. Application pods are connected directly to the VNet, which allows for native integration with VNet features.

[Learn more about networking in Azure Kubernetes Service](#)

Network configuration ⓘ ☒ Kubenet ☐ Azure CNI

DNS name prefix * ⓘ

Traffic routing

Load balancer ⓘ Standard

Enable HTTP application routing ⓘ ☐

Security

Enable private cluster ⓘ ☐

Set authorized IP ranges ⓘ ☐

Network policy ⓘ ☒ None ☐ Calico ☐ Azure

i The Azure network policy is not compatible with kubenet networking.

All services > Kubernetes services >

Create Kubernetes cluster

Basics Node pools Access Networking **Integrations** Advanced Tags Review + create

Connect your AKS cluster with additional services.

Azure Container Registry
Connect your cluster to an Azure Container Registry to enable seamless deployments from a private image registry. You can create a new registry or choose one you already have. [Learn more about Azure Container Registry](#)

Container registry [Create new](#)

Azure Monitor
In addition to the CPU and memory metrics included in AKS by default, you can enable Container Insights for more comprehensive data on the overall performance and health of your cluster. Billing is based on data ingestion and retention settings. [Learn more about container performance and health monitoring](#)
[Learn more about pricing](#)

Container monitoring ☒ Enabled ☐ Disabled
🔗 Azure monitor is recommended for standard configuration.

Log Analytics workspace ⓘ [Create new](#)

Use managed identity (preview) ⓘ ☐

Azure Policy
Apply at-scale enforcements and safeguards for AKS clusters in a centralized, consistent manner through Azure Policy. [Learn more about Azure Policy for AKS](#)

Azure Policy ☐ Enabled ☒ Disabled

And then for all the sections follow the defaults.

Press “**Review and Create**” and wait for the service to be deployed.

Install Kubeflow

Prerequisites

Kubernetes compatible on k8s 1.22

Kustomize version 3.2.0

Install and configure the **Azure Command Line Interface (Az)**

Installations of Prerequisites

Logging in the Azure

According to this guide you are logged into Azure portal with the subscription you got from the free trail.

If you are working locally you need to follow:

Login to Azure

```
az login
```

Initial cluster setup for new cluster

Create a resource group:

```
az group create -n <RESOURCE_GROUP_NAME> -l <LOCATION>
```

Example variables:

- RESOURCE_GROUP_NAME=KubeTest
- LOCATION=westus

Create a specifically defined cluster:

```
az aks create -g <RESOURCE_GROUP_NAME> -n <NAME> -s <AGENT_SIZE> -c  
<AGENT_COUNT> -l <LOCATION> --generate-ssh-keys
```

Example variables:

- NAME=KubeTestCluster
- AGENT_SIZE=Standard_D4_v3
- AGENT_COUNT=2
- Use the same resource group and name from the previous step

NOTE: If you are using a GPU based AKS cluster (For example: AGENT_SIZE=Standard_NC6), you also need to [install the NVidia drivers](#) on the cluster nodes before you can use GPUs with Kubeflow

Connecting to the cluster

Open the cluster/ Kubernetes service you created, click on the **Connect** tab

Home > Recent >

my-kube-cluster ☆ ☆ ...

Kubernetes service

Search

+ Create ▾ Connect ▸ Start □ Stop 🗑 Delete 🔄 Refresh 🗨 Give feedback

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Microsoft Defender for Cloud

Kubernetes resources

Namespaces

Workloads

Services and ingresses

Essentials

Resource group : [resourceGroup2](#)

Status : Succeeded (Running)

Location : Central US

Subscription : [Azure subscription 1](#)

Subscription ID : 653c88cd-dff3-4b5c-b717-a78b8467fe1e

Tags [\(edit\)](#) : [Click here to add tags](#)

Get started **Properties** Monitoring Capabilities (3) Recommendations Tutorials

Kubernetes services

Encryption type Encryption at-rest with a platform-managed key

Networking

API server address my-kube-cluster-dns-2c93fe48.hcp.centralus.azmk8s.io

Kubernetes version : 1.23.8

API server address : my-kube-cluster-dns-2c93fe48.hcp.centralus.azmk8s.io

Network type (plugin) : [Kubenet](#)

Node pools : [1 node pool](#)

Copy the first command from the pane that just opened.

Microsoft Azure Upgrade

Search resources, services, and docs (5/5)

Home > Recent >

my-kube-cluster ☆ ☆ ...

Kubernetes service

Search

+ Create ▾ Connect ▸ Start □ Stop 🗑 Delete 🔄 Refresh 🗨 Give feedback

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Microsoft Defender for Cloud

Kubernetes resources

Namespaces

Workloads

Services and ingresses

Storage

Configuration

Settings

Node pools

Cluster configuration

Networking

Open Service Mesh

Essentials

Resource group : [resourceGroup2](#)

Status : Succeeded (Running)

Location : Central US

Subscription : [Azure subscription 1](#)

Subscription ID : 653c88cd-dff3-4b5c-b717-a78b8467fe1e

Tags [\(edit\)](#) : [Click here to add tags](#)

Get started **Properties** Monitoring Capabilities (3) Recommendations Tutorials

Kubernetes services

Encryption type Encryption at-rest with a platform-managed key

Virtual node pools Not enabled

Node pools

Node pools 1 node pool

Kubernetes versions 1.23.8

Node sizes Standard_D2ads_v5

Configuration

Kubernetes version 1.23.8

Authentication and Authorization Local accounts with Kubernetes RBAC

Networking

API server address my-kube-cluster-dns-2c93fe48.hcp.centralus.azmk8s.io

Network type (plugin) [Kubenet](#)

Pod CIDR

Service CIDR

DNS service IP

Docker bridge CIDR

Network Policy

Load balancer

HTTP application routing

Private cluster

Connect to my-kube-cluster

Connect to your cluster using command line tooling to interact directly with cluster using kubectl, the command line tool for Kubernetes. Kubectl is available within the Azure Cloud Shell by default and can also be installed locally. [Learn more](#) if

1. Open Cloud Shell or the Azure CLI

2. Run the following commands

`az account set --subscription 653c88cd-dff3-4b5c-b717-a78b8467fe1e`

`az aks get-credentials --resource-group resourceGroup2 --name my-kube-cluster`

Sample commands

Once you have run the command above to connect to the cluster, you can run any kubectl commands. Here are a few examples of useful commands you can try.

List all deployments in all namespaces

`kubectl get deployments --all-namespaces=true`

List all deployments in a specific namespace

`kubectl get deployments --namespace <namespace-name>`

Format kubectl get deployments --namespace <namespace-name>

`kubectl get deployments --namespace kube-system`

List details about a specific deployment

`kubectl describe deployment <deployment-name> --namespace <namespace-name>`

Format kubectl describe deployment my-dep --namespace kube-system

`kubectl describe deployment my-dep --namespace kube-system`

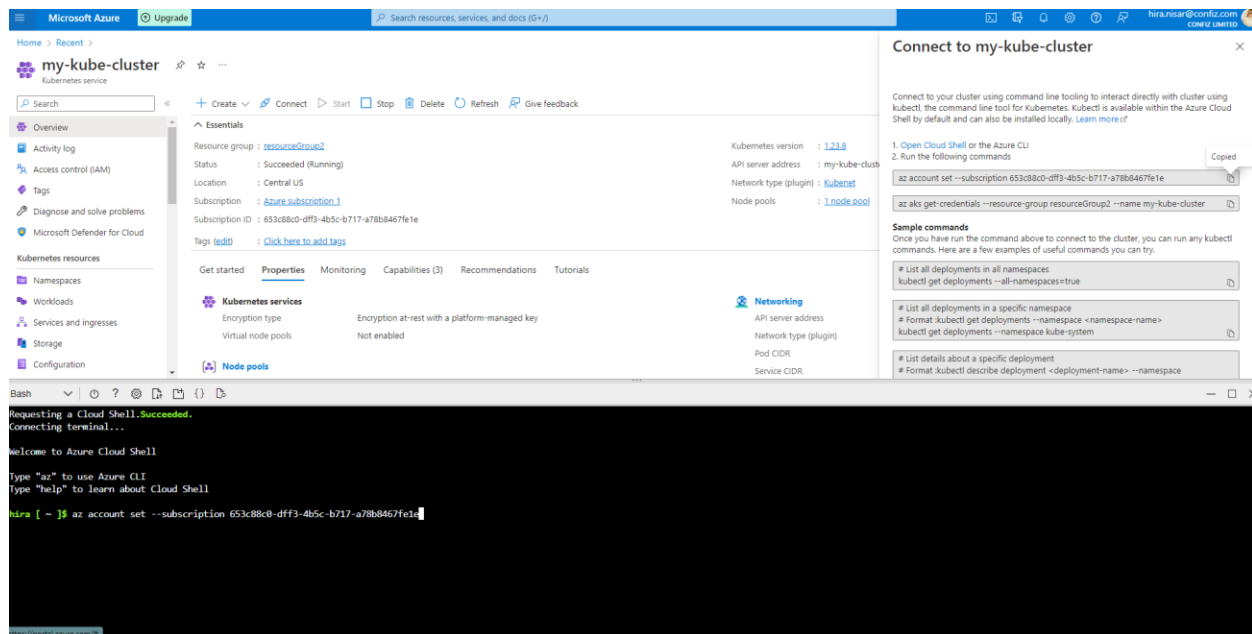
List pods using a specific label

`kubectl get pods -l <label-key>=<label-value> --all-namespaces=true`

Get logs for all pods with a specific label

`kubectl logs -l <label-key>=<label-value>`

Open the **Cloud Shell** and paste and run the command. Then do the same with the second command.



Now you have started your cluster service successfully.

Installation of prerequisites

Install kubectl

Install for Linux, as our virtual machine is Ubuntu.

[Install Tools | Kubernetes](#)

Download the latest release with the command:

```
curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
```

To download a specific version, replace the `$(curl -L -s https://dl.k8s.io/release/stable.txt)` portion of the command with the specific version.

For example, to download version v1.25.0 on Linux, type:

```
curl -LO https://dl.k8s.io/release/v1.25.0/bin/linux/amd64/kubectl
```

There are other methods too provided on the webpage:

[Install and Set Up kubectl on Linux | Kubernetes](#)

Now verify the kubectl configuration

In order for kubectl to find and access a Kubernetes cluster, it needs a [kubeconfig file](#), which is created automatically when you create a cluster. By default, kubectl configuration is located at `~/.kube/config`. Check that kubectl is properly configured by getting the cluster state:

```
kubectl cluster-info
```

Install Azure Command Line Interface (Az)

As, we are already working on the azure cloud service, we are already logged in and Azure cli is already preinstalled.

If you are working locally you need to login with the command: `az login`

Install Docker (Optional)

As, this is optional, I am not providing the detailed procedure for this but if you like to install it, follow the following links.

[Setting Up Docker for Windows and WSL to Work Flawlessly — Nick Janetakis](#)

Kubeflow Installation

Set up and Kubeflow.

- ✓ Create user credentials. You only need to run this command once:
 - `az aks get-credentials -n <NAME> -g <RESOURCE_GROUP_NAME>`
- ✓ Download the kfctl v0.7.1 release from the [Kubeflow releases page](#)
- ✓ Unpack the tar ball:
 - `tar -xvf kfctl_v0.7.1_<platform>.tar.gz`

Deploy Kubeflow

The code below includes an optional command to add the binary kfctl to your path. If you don't add the binary to your path, you must use the full path to the kfctl binary each time you run it.

- ✓ The following command is optional, to make kfctl binary easier to use.
 - `export PATH=$PATH:<path to where kfctl was unpacked>`
- ✓ Set KF_NAME to the name of your Kubeflow deployment. This also becomes the of the directory containing your configuration. For example, your deployment name can be 'my-kubeflow' or 'kf-test'.
 - `export KF_NAME=<your choice of name for the Kubeflow deployment>`
- ✓ Set the path to the base directory where you want to store one or more Kubeflow deployments. For example, /opt/. Then set the Kubeflow application directory for this deployment.

- export BASE_DIR=<path to a base directory>
- export KF_DIR=\${BASE_DIR}/\${KF_NAME}
- ✓ Set the configuration file to use, such as the file specified below:
 - export
 CONFIG_URI=https://raw.githubusercontent.com/kubeflow/manifests/v0.7-branch/kfdef/kfctl_k8s_istio.0.7.1.yaml
- ✓ Generate and deploy Kubeflow:
 - mkdir -p \${KF_DIR}
 - cd \${KF_DIR}
 - kfctl apply -V -f \${CONFIG_URI}

Rules

\${KF_NAME} - The name of your Kubeflow deployment. If you want a custom deployment name, specify that name here. For example, my-kubeflow or kf-test. The value of KF_NAME must consist of lower-case alphanumeric characters or '-', and must start and end with an alphanumeric character. The value of this variable cannot be greater than 25 characters. It must contain just a name, not a directory path. This value also becomes the name of the directory where your Kubeflow configurations are stored, that is, the Kubeflow application directory.

\${KF_DIR} - The full path to your Kubeflow application directory.

Kustomization

The above process will not allow you to deploy kubeflow fully, so you need not to worry seeing the warnings and the errors in the above installations.

[Configuring Kubeflow with kfctl and kustomize | Kubeflow](#)

We will use **kustomize** to further complete our installations.

- ✓ Install kustomize version 3.2.0: [Release v3.2.0 · kubernetes-sigs/kustomize \(github.com\)](#)
[Configuring Kubeflow with kfctl and kustomize | Kubeflow](#)
 Check the installed version with the command:

```
kustomize version
```

You can also use other methods to install "kustomize"

[Kustomize | SIG CLI \(kubernetes.io\)](#)

Easiest way to do it is to install the latest binaries using the command:

```
curl -s "https://raw.githubusercontent.com/kubernetes-sigs/kustomize/master/hack/install_kustomize.sh" | bash
```

or

```
curl -s https://api.github.com/repos/kubernetes-sigs/kustomize/releases
|\
grep browser_download |\
grep download/kustomize |\
grep -m 1 $opsys |\
cut -d '"' -f 4 |\
xargs curl -O -L
```

[Binaries | SIG CLI \(kubernetes.io\)](#)

If you want to install ARM binaries, visit the releases page: [Releases · kubernetes-sigs/kustomize \(github.com\)](#)

- ✓ Clone the manifest file:

```
git clone https://github.com/kubeflow/manifests.git
cd manifests
```

- ✓ In the manifest repo, navigate to:

```
common/istio-1-14/istio-install/base/install.yaml.
```

Go to:

```
apiVersion: admissionregistration.k8s.io/v1beta1
kind: MutatingWebhookConfiguration
metadata:
  name: istio-sidecar-injector
  labels:
    istio.io/rev: default
    install.operator.istio.io/owning-resource: unknown
    operator.istio.io/component: "Pilot"
    app: sidecar-injector
    release: istio
....
```

- ✓ And add notation:
admissions.enforcer/disabled: "true"

as follows:

```
apiVersion: admissionregistration.k8s.io/v1beta1
kind: MutatingWebhookConfiguration
metadata:
```

```

name: istio-sidecar-injector
annotations:
  admissions.enforcer/disabled: "true"
labels:
  istio.io/rev: default
  install.operator.istio.io/owning-resource: unknown
  operator.istio.io/component: "Pilot"
  app: sidecar-injector
  release: istio
....

```

- ✓ Save your changes to yaml file and, now install Kubeflow:

```

while ! kustomize build example | kubectl apply -f -; do echo "Retrying
to apply resources"; sleep 10; done

```

- ✓ Run this command to check that the resources have been deployed correctly in namespace kubeflow:

```

kubectl get all -n kubeflow

```

- ✓ Expose load balancer

To expose Kubeflow with a load balancer service, change the type of the istio-ingressgateway service to LoadBalancer.

```

kubectl patch service -n istio-system istio-ingressgateway -p '{"spec":
{"type": "LoadBalancer"}}'

```

After that, obtain the LoadBalancer IP address (this may take more than 15 minutes)

```

kubectl get svc -n istio-system istio-ingressgateway -o
jsonpath='{.status.loadBalancer.ingress[0]}'

```

Running it should return you with a response containing loadbalancer IP where Kubeflow is running.

- ✓ create a self-signed certificate with cert-manager:

```

nano certificate.yaml:

```

This will open a GNU console within bash where you need to create a certificate by typing the following:

```

apiVersion: cert-manager.io/v1
kind: Certificate
metadata:

```



```
name: istio-ingressgateway-certs
namespace: istio-system
spec:
  commonName: istio-ingressgateway.istio-system.svc
  # Use ipAddresses if your LoadBalancer issues an IP address
  ipAddresses:
    - <LoadBalancer IP>
  isCA: true
  issuerRef:
    kind: ClusterIssuer
    name: kubeflow-self-signing-issuer
  secretName: istio-ingressgateway-certs
```

✓ After creating the certificate apply it:

```
kubectl apply -f certificate.yaml -n istio-system
```

So, now, Kubeflow has been deployed

To open the Kubeflow Dashboard, run the following command to get the dashboard IP:

```
kubectl get svc -n istio-system istio-ingressgateway -o
jsonpath='{.status.loadBalancer.ingress[0]}'
```

Next, open <loadbalancerip> in your browser.

Login to Kubeflow using default email: user@example.com and password: 12341234.