# Kubeflow Deployment

# Table of contents

# Kubeflow Deployment

## Kubeflow Deployment

### Introduction

This document provides instructions for deploying Kubeflow on Tanzu Kubernetes cluster.

### Scope and Steps

Kubeflow provides components for each stage in the machine learning lifecycle, from exploration through to training and deployment. Operators can choose what is best for their users, there is no requirement to deploy every component of Kubeflow.

### Prerequisites

**NOTE:** All prerequisites must be installed and configured before creating the Tanzu Kubernetes cluster.

Perform the following steps:

1. Download and Install kubectl for vSphere in our validation for Kubeflow version 1.5 of kubectl requires v1.21+.
2. Make sure you first create a Tanzu Kubernetes cluster and install GPU Operator on your Tanzu Kubernetes cluster in the configuration session.
3. Install Kustomize for Kubeflow installation

### Deploy Kubeflow

We used the manifests for installation, perform the following steps to deploy Kubeflow 1.5.0 on your Tanzu Kubernetes cluster:

1. The following kubectl command creates a ClusterRoleBinding that grants access to authenticated users to run a privileged set of workloads using the default PSP vmware-system-privileged.

```
kubectl create clusterrolebinding default-tkg-admin-privileged-binding --clusterrole=psp:vmware-system-privileged --group=system:authenticated
```

2. Set the default storageclass for pv claims of kubeflow components such as MinIO and MySQL:

```
kubectl patch storageclass seletedstorageclassname -p '{"metadata": {"annotations"{"storageclass.kubernetes.io/is-default-class":"true"}}}'
```

```
 root@photon-HCIBench [ ~/manifests/example ]# kubectl get sc
NAME                PROVISIONER                                    RECLAIMPOLICY           VOLUMEBINDINGMODE             ALLOWVOLUMEEXPANSION          AGE

nfs-external        cluster.local/nfs-subdir-external-provisioner   Delete                  Immediate                     true                          26d

stripe              csi.vsphere.vmware.com                         Delete                  Immediate                     true                          23d

vsan-r1 (default)   csi.vsphere.vmware.com                         Delete                  Immediate                     true                          33d
```

**Figure 1: Set Default Storageclass**

3. Download the scripts to deploy kubeflow by cloning the Github repository:

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

 git checkout v1.5-branch
```

4. You can install kubeflow official components by using either of the two options, Install with a single command or Install individual components. **Note:** Individual components may have dependencies. If all the individual commands are executed, the result is the same as the single command installation.
5. Verify all the pods are running. The kubectl apply commands may fail on the first try. This is inherent in how Kubernetes and kubectl work. Try to rerun the command until it succeeds.
   To check that all Kubeflow-related pods are ready, use the following commands:

```
kubectl get pods -n cert-manager

kubectl get pods -n istio-system

kubectl get pods -n auth

kubectl get pods -n knative-eventing

kubectl get pods -n knative-serving

kubectl get pods -n kubeflow

kubectl get pods -n kubeflow-user-example-com
```

The following diagram shows the pods deployed in the Istio namespace:

```
kubectl get pod -n istio-system
NAME                                  READY        STATUS.    RESTARTS   AGE
authservice-0                         1/1          RUNNING     0         23h
cluster-local-gateway-7796d7bc87-9qb5v 1/1         Running     0         24h
istio-ingressgateway-64b7899489-ft5gn  1/1         Running     0         24h
istio-5d9bb9cb4-5zvzz                 1/1          Running     0         24h
```

**Figure 2: Pods in istio-system Namespace**

**Figure 3** shows the pods deployed in the kubeflow namespace:

```
ubuntu@vmware-tanzu-jumpbox            kubectl get pod -n kubeflow

NAME                                                READY        STATUS    RESTARTS    AGE
admission-webhook-deployment-7df7558c67
-gltpf                                               1/1         Running    0          23d
cache-deployer-deployment-6f4bcc969-7j2jk            1/1         Running    0          23d
cache-server-7cc6cbbf55-8f6m9                        1/1         Running    0          23d
centraldashboard-5dd4f57bbd-2k7f7                    2/2         Running    0          22d
jupyter-web-app-deployment-8d96db4cd-7n4g5           1/1         Running    0          23d
katib-controller-58ddb4b856-fafhw                    1/1         Running    0          23d
katib-db-manager-6df878f5b8-27545                    1/1         Running    0          23d
katib-mysql-6dcb447c6f-xp8fc                         1/1         Running    0          23d
katib-ui-f787b9d88-gglr5                             1/1         Running    0          23d
kfserving-controller-manager-                        1/1         Running    0          23d
kfserving-models-web-app-5d6cd6b5dd-58q6d            1/1         Running    0          23d
kserve-models-web-app-6f45769bb6-5adpz               1/1         Running    0          23d
kubeflow-pipelines-profile-controller-7fd7c77c5d-kx4591/1        Running    0          23d
metacontroller-0                                     1/1         Running    0          23d
metadata-envoy-deployment-76847ff6c5-2bdbz           1/1         Running    0          23d
metadata-grpc-deployment-6f6f7776c5-btchf            2/2         Running    0          23d
metadata-writer-78fc7d5bb8-7s9c9                     1/1         Running    0          23d
minio-5665df66c9-hfjm8                               2/2         Running    0          23d
ml-pipeline-6bccbd7bd-5m6n6                          2/2         Running    0          23d
ml-pipeline-persistenceagent-87b6888c4-bxlcb         2/2         Running    0          23d
ml-pipeline-scheduledworkflow-665847bb9-pj91m        2/2         Running    0          23d
ml-pipeline-ui-68cc764f66-w7gww                      2/2         Running    0          23d
ml-pipeline-viewer-crd-68777557fb-g7sms              2/2         Running    0          23d
ml-pipeline-visualizationserver-58ccb76855-dlmwn     2/2         Running    0          23d
mysql-f7b9b7dd4-k65vv                                2/2         Running    0          23d
notebook-controller-deployment-5d9c6c656c-4prq4      2/2         Running    0          23d
profiles-deployment-78ffd649f5-q7bk9                 3/3         Running    0          22d
tensorboard-controller-controller-manager-6848cb6846- 3/3        Running    0          23d
9h4sn
tensorboards-web-app-deployment-7c5db448d7-9ggp7     1/1         Running    0          23d
training-operator-7b8cc9865d-hffbp                   1/1         Running    0          23d
volumes-web-app-deployment-87484c848-62t9n           1/1         Running    0          23d
workflow-controller-6fc6f67d66-5zpgx                 2/2         Running    2          22d
```

**Figure 3: Pods in Kubeflow Namespace**

6. Access the Kubeflow central dashboard:
   - **Option** 1: Port forward: The default way of accessing Kubeflow is via port-forward.

```
kubectl port-forward svc/istio-ingressgateway -n istio-system 8080:80
```

Example: http://localhost:8080

   - **Option 2:** NodePort/LoadBalancer/Ingress: since many of the Kubeflow web apps (for example, Tensorboard Web App, Jupyter Web App, Katib UI) use secure cookies, we need to set up HTTPS.

     We can access the dashboard using the LoadBalancer external IP address 

     - Change the type of the istio-ingressgateway service to LoadBalancer:

```
kubectl -n istio-system patch service istio-ingressgateway -p '{"spec": {"type": "LoadBalancer"}}'
kubectl get svc -n istio-system
NAME            TYPE           CLUSTER-IP     EXTERNAL-IP    PORT (S)
```

  

```
Authservice   ClusterIP       10.100.82.68    <none>         8080/TCP
cluster-local-gateway ClusterIP 10.101.213.134 <none> 15020/TCP,80/TCP
istio-ingressgateway LoadBalancer 10.104.45.33 172.16.20.72 15021:32506/TCP,80:31917/TCP,443:32332/TCP,314
istiod ClusterIP  10.103.211.151 (none>
5010/TCP.15012/TCP,443/TCP,15014/TCP
knative-local-gateway ClusterIP 10.111.221.131 <none> 80/TCP
```

**Figure 4: Change istio-ingressgateway Service Type to Loadbalancer**

And make changes to set up HTTPS configuration.

## Configure HTTPS

Make the following changes:

- Update Istio Gateway to expose port 443 with HTTPS and make port 80 redirected to 443:

```
kubectl -n kubeflow edit gateways.networking.istio.io kubeflow-gateway
servers:
- hosts:
  -“*”
  port:
    name: http
    number: 80
    protocol: HTTP
  tls:
    httpsRedirect: true
-hosts:
 -“*”
 port:
   name: https
   number: 443
   protocol: HTTPS
tls:
  mode: SIMPLE
  privatekey:/etc/istio/ingressgateway-certs/tls.key
  serverCertificate:/etc/istio/ingressgateway-certs/tls.crt
```

**Figure 5: Update istio Gateway Attributes**

- Change the REDIRECT_URL in oidc-authservice-parameters configmap.

  In our example, 172.16.20.72 is the IP address of the istio-ingressgateway.

```
kubectl -n istio-system edit configmap oidc-authservice-parameters
OIDC SCOPES: profile email groups
PORT: ‘"8080”’
REDIRECT URL: https://172.16.20.72/login/oide
SKIP AUTH URI: / dex
STORE PATH: /var/lib/authservice/data.db
```

**Figure 6: Change REDIRECT_URL to Loadbalancer IP Address**

Append the same to the redirectURIs list in dex configmap:

```
kubectl -n auth edit configmap dex
```

- Rollout restart authservice and dex

```
kubectl -n istio-system rollout restart statefulset authservice
kubectl -n auth rollout restart deployment dex
```

- Create a certificate.yaml with the YAML in **Figure 7** to create a self-signed certificate:

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

```
apiVersion:
cert-manager.io/vlalpha2
kind: Certificate
metadata:
name: istio-ingressgateway-certs
namespace: istio-system
spec:
commonName: istio-ingressgateway.istio-system.svc
ipAddresses:
- 172.16.20.72
isCA: true
issuerRef:
kind: ClusterIssuer
name: kubeflow-self-signing-issuer
secretName:istio-ingressgateway-certs
```

**Figure 7: Create istio-ingressgateway Certificate**

- We can access the Kubeflow Central Dashboard from https:// IP address of the istio-ingressgateway.

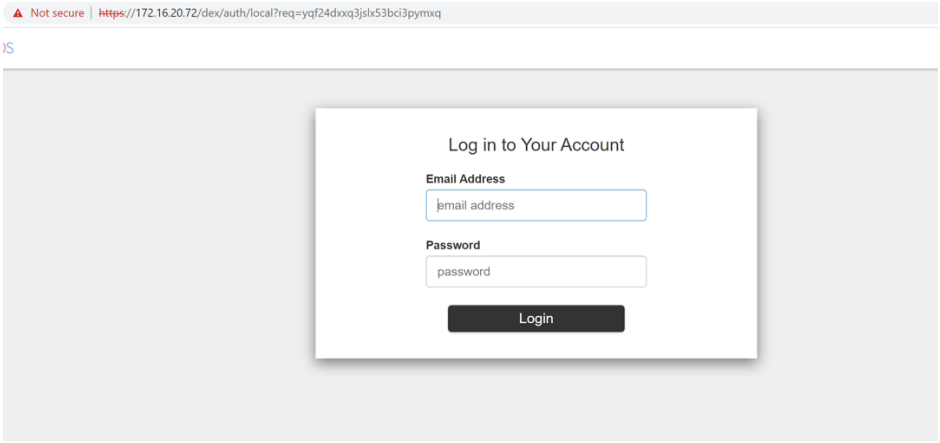


**Figure 8: Kubeflow Login Page**

Log in with the default user's credential. The default email address is user@example.com and the default password is 12341234. The default user's namespace is Kubeflow-user-example-com.

-

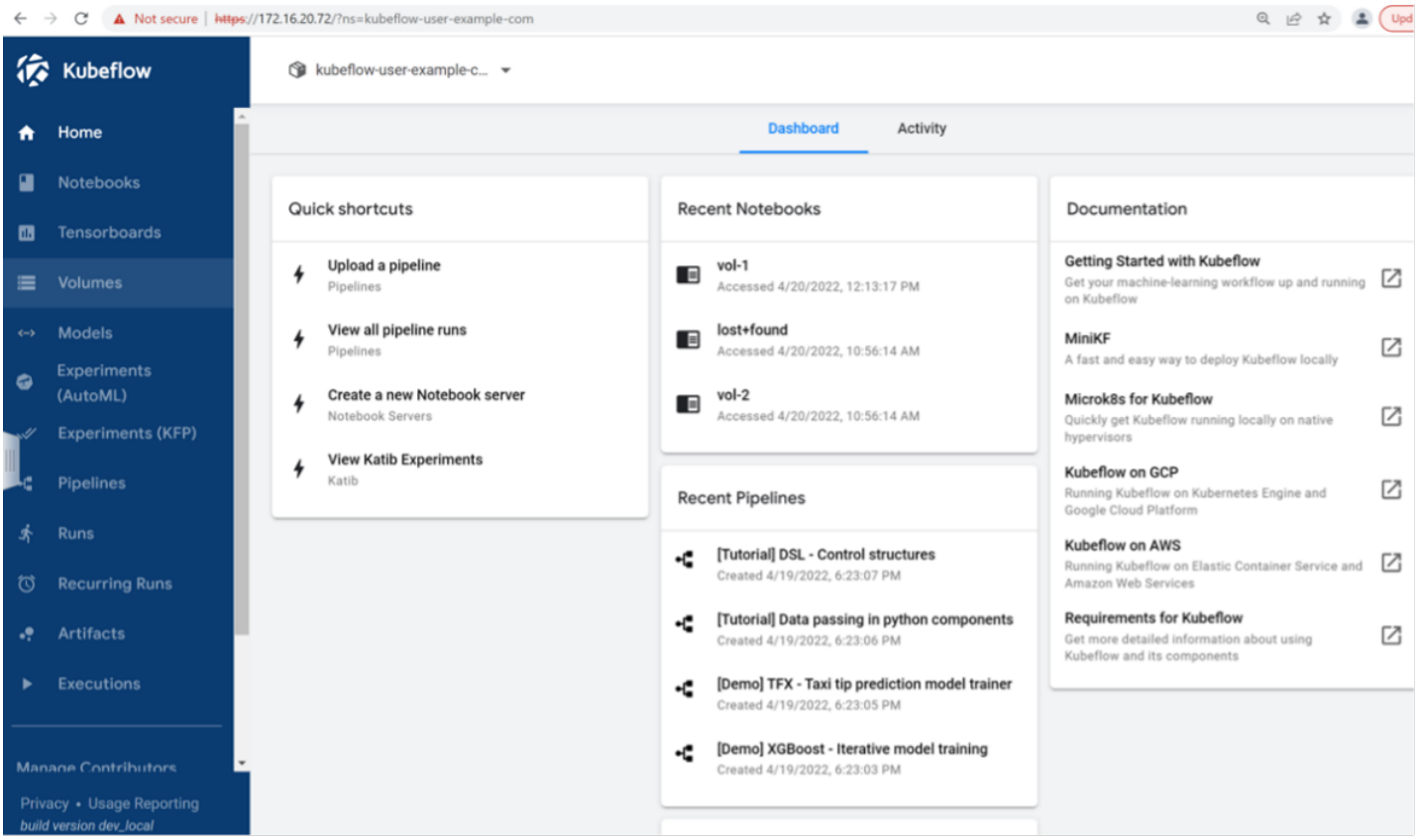


**Figure 9: Kubeflow Central Dashboard**

## Add New Users

Add new user: users are managed by Kubeflow profile module:

```
cat <<EOF | kubectl apply -f
apiVersion: kubeflow.org/v1beta1
kind: Profile
metadata:
name: newuser’s namesmespacename   # replace with the name of profile you want
spec:
owner:
kind: User
name: newuser@example.com   # replace with the user email
EOF
```

Add the user credentials in dex in Kubeflow for basic authentication. Generate the hash by using bcrypt in the dex configmap:

```
kubectl edit cm dex -o yaml -n auth
```

Add the new user under the staticPasswords section:

```
-email: newuser@example.com
 hash: $2v$12$4K/VkmDdla10rb3xAt82zu8qk7Ad6ReFR4ICP9UeYE9ONLiN9D£72
 username: newuser
```

**Figure 10: Add New User in Dex Configmap**

For more information, refer to Kubeflow Getting Started.

Check out the solution Home Page for more information.

Previous page: Kubeflow Configuration

Next page: Kubeflow Validation