Université Euro Méditerranéenne Fès

EuroMed University of Fez

Ecole d'Ingénierie Digitale et d'Intelligence Artificielle (EIDIA)

Filière : 1ème année cyber-security

Semestre : 6

Module : infonuagique

Thème :

Configuration kubeflow sur ubuntu 20.04

Framed by :

Pr. Amammou Ahmed

Prepared by the student:

MALIKI Maroua

Année Universitaire : 2023-2024

Sommaire:

DOCKER :

Définition

Docker est une plateforme de virtualisation logicielle open-source qui permet de créer, d'exécuter et de gérer des conteneurs d'applications. Les conteneurs Docker sont des environnements isolés et légers qui encapsulent une application et ses dépendances, ce qui permet de les exécuter de manière cohérente sur différents systèmes d'exploitation.

Conteneurs :unité logicielle qui regroupe tout ce dont votre app a besoin pour fonctionner.

Rôles

Docker joue plusieurs rôles importants dans le développement et le déploiement d'applications :

- **Création d'images:** Docker permet de créer des images d'applications, qui sont des modèles préconfigurés pour exécuter des applications spécifiques. Les images peuvent être stockées et partagées localement ou dans des registres d'images publics ou privés.
- **Exécution de conteneurs:** Docker permet d'exécuter des conteneurs à partir d'images. Les conteneurs sont des instances exécutables d'une image, et ils encapsulent l'application et ses dépendances.
- **Gestion des conteneurs:** Docker permet de gérer les conteneurs en cours d'exécution. Cela inclut le démarrage, l'arrêt, la suspension, la reprise et la suppression des conteneurs.
- **Orchestrierung des conteneurs:** Docker peut être utilisé pour orchestrer des conteneurs à grande échelle. Cela implique la gestion du déploiement, de la mise à l'échelle et de la mise à jour des conteneurs sur un cluster d'hôtes.

Commande de configuration :

First, update your existing list of packages: **sudo apt update**

install Docker: **sudo apt install docker-ce**

Docker should now be installed, the daemon started, and the process enabled to start on boot. Check that it's running: **sudo systemctl status docker**

The output should be similar to the following, showing that the service is active and running:

```
Output
● docker.service - Docker Application Container Engine
     Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
     Active: active (running) since Tue 2020-05-19 17:00:41 UTC; 17s ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 24321 (dockerd)
      Tasks: 8
     Memory: 46.4M
     CGroup: /system.slice/docker.service
             └─24321 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
```

KUBERNETES:

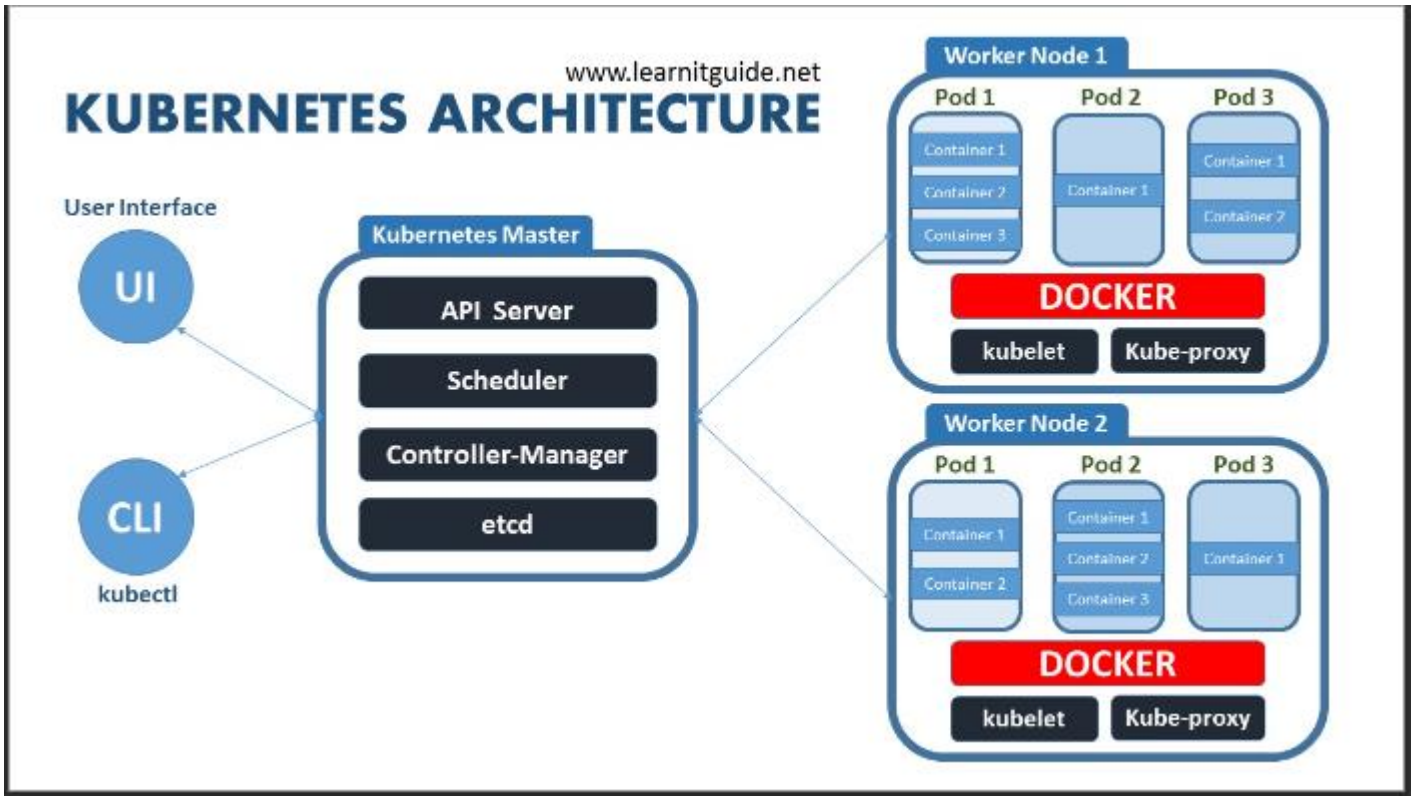
Définition

Kubernetes, souvent abrégé en K8s, est une plateforme open-source de gestion de conteneurs. Elle automatise le déploiement, la mise à l'échelle et la gestion des applications conteneurisées sur des clusters d'hôtes. Kubernetes ne gère pas directement les conteneurs, mais plutôt les ressources nécessaires à leur exécution.

Rôle

Kubernetes joue un rôle essentiel dans l'orchestration des conteneurs à l'echelle de cluster en fournissant les fonctionnalités suivantes :

- **Déploiement:** Kubernetes permet de déployer des applications conteneurisées de manière déclarative. Cela signifie que vous définissez l'état souhaité de votre application, et Kubernetes s'occupe de le réaliser en créant et en gérant les conteneurs nécessaires.
- **Mise à l'échelle:** Kubernetes peut automatiquement mettre à l'échelle vos applications en fonction de la demande. Si le trafic augmente, Kubernetes peut démarrer des instances supplémentaires de vos conteneurs. Inversement, si le trafic diminue, Kubernetes peut arrêter les conteneurs inutilisés.



Kubernetes follows master-slave architecture. Kubernetes architecture has a master node and worker nodes. There are four components of a **master node**:

→ Kube API server

→ controller

→ scheduler

→ etcd

And, the **worker node** has three components:

→ kubelet

→ kube-proxy

→ container runtime

Kubeadm

- **Définition:** Outil en ligne de commande conçu pour initialiser un cluster Kubernetes. Il automatise la configuration des composants essentiels du cluster, tels que les API Server, les etcd et les contrôleurs Kubernetes.
- **Rôle:** Simplifier le processus d'initialisation d'un cluster Kubernetes en automatisant les tâches complexes.

Kubectl

- **Définition:** Outil en ligne de commande pour interagir avec les clusters Kubernetes. Il permet de gérer les ressources du cluster, telles que les pods, les déploiements, les services et les nœuds.
- **Rôle:** Offrir une interface utilisateur pour gérer et contrôler les clusters Kubernetes.

Kubelet

- **Définition:** Agent qui s'exécute sur chaque nœud du cluster et qui est responsable de la gestion des conteneurs. Il reçoit les instructions du maître du cluster et exécute les tâches nécessaires, telles que le démarrage, l'arrêt et le redémarrage des conteneurs.
- **Rôle:** Gérer les conteneurs sur chaque nœud du cluster en fonction des instructions du maître du cluster.

Installation des paquets de base

1. Sur les deux nœuds (maître et worker), commencez par installer le paquet `apt-transport-https` qui permet d'utiliser les protocoles http et https dans les dépôts Ubuntu. Installez également `curl` car il sera nécessaire plus tard :

Bash
```
sudo apt install apt-transport-https curl
```

Ajout de la clé de signature Kubernetes

2. Ajoutez la clé de signature Kubernetes sur les deux systèmes :

Bash
```
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
```

Ajout du dépôt de paquets Kubernetes

3. Ajoutez le dépôt de paquets Kubernetes. Notez que la commande suivante utilise "xenial" car elle correspond à Ubuntu 16.04, la version la plus récente du dépôt Kubernetes disponible au moment de la rédaction de ce guide. Lorsque Ubuntu 20.04 (focal fossa) sera officiellement supporté, vous pourrez remplacer "xenial" par "focal" dans la commande :

Bash
```
sudo apt-add-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main"
```

Installation des outils Kubernetes

4. Installez les outils Kubernetes nécessaires :

Bash
```
sudo apt install kubeadm kubelet kubectl kubernetes-cni
```

Désactivation de la mémoire Swap

5. Kubernetes ne fonctionne pas si la mémoire swap est activée. Désactivez-la sur le nœud maître et le nœud worker avec la commande suivante :

Bash
```
sudo swapoff -a
```

Attention : cette commande désactive la mémoire swap jusqu'au prochain redémarrage du système. Pour rendre ce changement permanent, éditez le fichier `/etc/fstab`.

6. Ouvrez le fichier `/etc/fstab` avec votre éditeur de texte préféré (par exemple `nano`) :

Bash
```
sudo nano /etc/fstab
```

7. Commentez la ligne correspondant au fichier d'échange (swapfile) en ajoutant un symbole "#" au début de la ligne.
8. Enregistrez les modifications et fermez le fichier.

Définition des noms d'hôtes

9. Assurez-vous que chaque nœud possède un nom d'hôte unique. Dans cet exemple, nous utilisons les noms "kubernetes-master" et "kubernetes-worker" pour facilement différencier les rôles de chaque machine. Si nécessaire, changez le nom d'hôte avec la commande suivante :

Sur le noeud maître :

Bash

```
sudo hostnamectl set-hostname kubernetes-master
```

Sur le noeud worker :

Bash

```
sudo hostnamectl set-hostname kubernetes-worker
```

Remarque : les modifications du nom d'hôte ne seront pas prises en compte dans le terminal en cours. Ouvrez une nouvelle session pour voir les changements.

10. Vérifiez que tous les nœuds ont une date et une heure précises. Des certificats TLS invalides peuvent survenir si la date et l'heure ne sont pas synchronisées.

Initialisation du nœud maître Kubernetes

11. Maintenant, nous pouvons initialiser le nœud maître Kubernetes. Exécutez la commande suivante sur le nœud maître :

Bash

```
kubernetes-master:~$ sudo kubeadm init
```

La commande d'initialisation génère un jeton et une configuration que nous utiliserons plus tard pour joindre les nœuds worker au cluster. Prenez note de la sortie de la commande, en particulier du jeton et de la configuration.

Déploiement d'un réseau de pods

12. La prochaine étape consiste à déployer un réseau de pods. Le réseau de pods est utilisé pour la communication entre les hôtes et est nécessaire au bon fonctionnement du cluster Kubernetes. Nous allons utiliser le réseau de pods Flannel. Exécutez les deux commandes suivantes sur le nœud maître :

Bash

```
kubernetes-master:~$ kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
kubernetes-master:~$ kubectl apply -f https://raw.
```

Kubeflow:

Kubeflow est une plateforme open-source permettant de déployer et gérer des pipelines de machine learning (ML) sur Kubernetes. Elle offre un ensemble d'outils, de bibliothèques et de services qui facilitent l'automatisation du cycle de vie du machine learning, de l'expérimentation à la production.

Rôle de Kubeflow

Kubeflow joue un rôle clé dans l'orchestration et l'exécution de pipelines de machine learning en fournissant les fonctionnalités suivantes :

- **Définition de pipelines:** Kubeflow permet de définir des pipelines de machine learning portables et reproductibles. Ces pipelines peuvent inclure des étapes de prétraitement des données, d'entraînement de modèles, d'évaluation et de déploiement.
- **Exécution de pipelines:** Kubeflow gère l'exécution des pipelines sur Kubernetes. Il orchestre les différentes étapes du pipeline, en s'assurant que les tâches sont exécutées dans le bon ordre et que les ressources nécessaires sont disponibles.
- **Gestion des artefacts:** Kubeflow permet de stocker et de gérer les artefacts du machine learning, tels que les données d'entraînement, les modèles et les résultats d'évaluation.
- **Expérimentation de machine learning:** Kubeflow fournit des outils pour faciliter l'expérimentation de machine learning. Cela comprend la possibilité de lancer des expériences à distance, de suivre les résultats et de comparer différentes configurations de modèle.
- **Déploiement de modèles:** Kubeflow permet de déployer des modèles de machine learning en production sur Kubernetes. Il gère la mise à l'échelle et la gestion des ressources pour les modèles déployés.

Commande de configuration :

1 .cert-manager

Kustomize build common/cert-manager/cert-manager/base|kubectl apply –f-

```
clusterrole.rbac.authorization.k8s.io/cert-manager-controller-orders created
clusterrole.rbac.authorization.k8s.io/cert-manager-edit created
clusterrole.rbac.authorization.k8s.io/cert-manager-view created
clusterrole.rbac.authorization.k8s.io/cert-manager-webhook:subjectaccessreviews created
rolebinding.rbac.authorization.k8s.io/cert-manager-webhook:dynamic-serving created
rolebinding.rbac.authorization.k8s.io/cert-manager-cainjector:leaderelection created
rolebinding.rbac.authorization.k8s.io/cert-manager:leaderelection created
clusterrolebinding.rbac.authorization.k8s.io/cert-manager-cainjector created
clusterrolebinding.rbac.authorization.k8s.io/cert-manager-controller-approve:cert-manager-io created
clusterrolebinding.rbac.authorization.k8s.io/cert-manager-controller-certificates created
clusterrolebinding.rbac.authorization.k8s.io/cert-manager-controller-challenges created
clusterrolebinding.rbac.authorization.k8s.io/cert-manager-controller-clusterissuers created
clusterrolebinding.rbac.authorization.k8s.io/cert-manager-controller-ingress-shim created
clusterrolebinding.rbac.authorization.k8s.io/cert-manager-controller-issuers created
clusterrolebinding.rbac.authorization.k8s.io/cert-manager-controller-orders created
clusterrolebinding.rbac.authorization.k8s.io/cert-manager-webhook:subjectaccessreviews created
service/cert-manager created
service/cert-manager-webhook created
deployment.apps/cert-manager created
deployment.apps/cert-manager-cainjector created
deployment.apps/cert-manager-webhook created
mutatingwebhookconfiguration.admissionregistration.k8s.io/cert-manager-webhook created
validatingwebhookconfiguration.admissionregistration.k8s.io/cert-manager-webhook created
```

Kubectl get pod –n cert-manager

```
NAME                                       READY   STATUS    RESTARTS   AGE
cert-manager-7dd5854bb4-q2qbm              1/1     Running   0          3m
cert-manager-cainjector-64c949654c-pqxsr   1/1     Running   0          3m
cert-manager-webhook-6b57b9b886-8sc54      1/1     Running   0          3m
```

```
seokii@seokii:~/manifests$ kustomize build common/cert-manager/kubeflow-issuer/base | kubectl apply -f -
clusterissuer.cert-manager.io/kubeflow-self-signing-issuer created
```

Installed successfully

2.Istio

```
kustomize build common/istio-1-9/istio-crds/base | kubectl apply -f -
kustomize build common/istio-1-9/istio-namespace/base | kubectl apply -f -
```

```
seokii@seokii:~/manifests$ kustomize build common/cert-manager/kubeflow-issuer/base | kubectl apply -f -
clusterissuer.cert-manager.io/kubeflow-self-signing-issuer created
seokii@seokii:~/manifests$ kustomize build common/istio-1-9/istio-crds/base | kubectl apply -f -
Warning: apiextensions.k8s.io/v1beta1 CustomResourceDefinition is deprecated in v1.16+, unavailable in v1.22+; use apiextensions.k8s.io/v1 CustomResourceDefinition
customresourcedefinition.apiextensions.k8s.io/authorizationpolicies.security.istio.io created
customresourcedefinition.apiextensions.k8s.io/destinationrules.networking.istio.io created
customresourcedefinition.apiextensions.k8s.io/envoyfilters.networking.istio.io created
customresourcedefinition.apiextensions.k8s.io/gateways.networking.istio.io created
customresourcedefinition.apiextensions.k8s.io/istiooperators.install.istio.io created
customresourcedefinition.apiextensions.k8s.io/peerauthentications.security.istio.io created
customresourcedefinition.apiextensions.k8s.io/requestauthentications.security.istio.io created
customresourcedefinition.apiextensions.k8s.io/serviceentries.networking.istio.io created
customresourcedefinition.apiextensions.k8s.io/sidecars.networking.istio.io created
customresourcedefinition.apiextensions.k8s.io/virtualservices.networking.istio.io created
customresourcedefinition.apiextensions.k8s.io/workloadentries.networking.istio.io created
customresourcedefinition.apiextensions.k8s.io/workloadgroups.networking.istio.io created
seokii@seokii:~/manifests$ kustomize build common/istio-1-9/istio-namespace/base | kubectl apply -f -
namespace/istio-system created
```

This is the result of installing Istio CRD and the Istio namespace

```
kustomize build common/istio-1-9/istio-install/base | kubectl apply -f -
kubectl get po -n istio-system
```

Install and verify Istio with commands.

```
validatingwebhookconfiguration.admissionregistration.k8s.io/istiod-istio-system created
seokii@seokii:~/manifests$ kubectl get po -n istio-system
NAME                                    READY   STATUS              RESTARTS   AGE
istio-ingressgateway-79b665c95-9cxs9    0/1     ContainerCreating   0          23s
istiod-86457659bb-g4z57                 1/1     Running             0          23s
seokii@seokii:~/manifests$ kubectl get po -n istio-system
NAME                                    READY   STATUS    RESTARTS   AGE
istiod-86457659bb-g4z57                 1/1     Running   0          66s
istio-ingressgateway-79b665c95-9cxs9    1/1     Running   0          66s
seokii@seokii:~/manifests$
```

To verify that both pods are in the running state

3.Dex

```
kustomize build common/dex/overlays/istio | kubectl apply -f -
kubectl get po -n auth
```

```
eokii@seokii:~/manifests$ kustomize build common/dex/overlays/istio | kubectl apply -f -
amespace/auth created
arning: apiextensions.k8s.io/v1beta1 CustomResourceDefinition is deprecated in v1.16+, un
sions.k8s.io/v1 CustomResourceDefinition
ustomresourcedefinition.apiextensions.k8s.io/authcodes.dex.coreos.com created
erviceaccount/dex created
arning: rbac.authorization.k8s.io/v1beta1 ClusterRole is deprecated in v1.17+, unavailabl
on.k8s.io/v1 ClusterRole
lusterrole.rbac.authorization.k8s.io/dex created
arning: rbac.authorization.k8s.io/v1beta1 ClusterRoleBinding is deprecated in v1.17+, una
orization.k8s.io/v1 ClusterRoleBinding
lusterrolebinding.rbac.authorization.k8s.io/dex created
onfigmap/dex created
ecret/dex-oidc-client created
ervice/dex created
eployment.apps/dex created
irtualservice.networking.istio.io/dex created
eokii@seokii:~/manifests$ kubectl get po -n auth
AME                   READY   STATUS    RESTARTS   AGE
ex-5ddf47d88d-pxmjq   1/1     Running   1          17s
eokii@seokii:~/manifests$
```

Install dex and verify the pod

4.OIDC AuthService

```
kustomize build common/oidc-authservice/base | kubectl apply -f -
kubectl get po -n istio-system -w
```

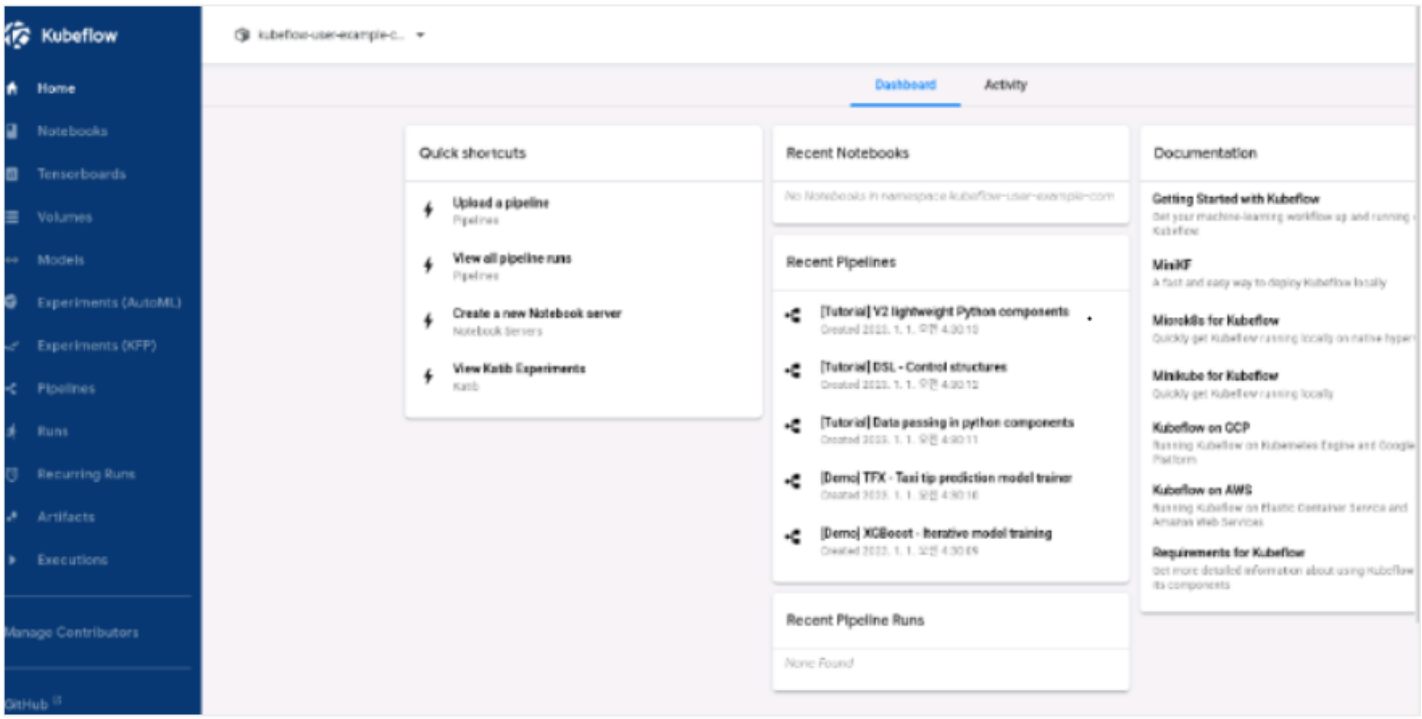
```
seokii@seokii:~/manifests$ kustomize build common/oidc-authservice/base | kubectl apply -f -
configmap/oidc-authservice-parameters created
secret/oidc-authservice-client created
service/authservice created
persistentvolumeclaim/authservice-pvc created
statefulset.apps/authservice created
envoyfilter.networking.istio.io/authn-filter created
seokii@seokii:~/manifests$ kubectl get po -n istio-system -w
NAME                                   READY   STATUS    RESTARTS   AGE
istiod-86457659bb-g4z57                1/1     Running   0          5m
istio-ingressgateway-79b665c95-9cxs9   1/1     Running   0          5m
authservice-0                          1/1     Running   0          23s
```

Install the OIDC AuthService and see the results.

5.installing all the components of kubeflow

6.finalisation

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



Kubeflow enfin installer.

CONCLUSION:

Cette demarche d'installation puisse paraître simple en apparence, elle nécessite une attention particulière à chaque étape pour éviter des erreurs potentielles. Ces erreurs peuvent provenir de différentes sources, telles que :

- **Incompatibilité des versions:** Assurez-vous que les versions des outils Kubernetes et du système d'exploitation sont compatibles.
- **Problèmes de stockage:** Vérifiez que vous disposez de suffisamment d'espace de stockage pour les conteneurs et les données.
- **Limites de ressources:** Assurez-vous que les ressources CPU, mémoire et réseau sont suffisantes pour exécuter les conteneurs et les pods.
- **Permissions insuffisantes:** Vérifiez que les utilisateurs et les processus possèdent les permissions nécessaires pour accéder aux ressources et exécuter les commandes.

La clé du succès réside dans la compréhension des erreurs et la prise de mesures correctives pour les éviter à l'avenir. En suivant attentivement les instructions et en consultant les ressources disponibles, vous pouvez installer et configurer avec succés.