Guide de Déploiement d'un Cluster Kubernetes Multi-Nœuds

Réalisé par :

M. Fode Mangane

Table des matières

1. Introduction

Dans ce guide, nous allons procéder à l'installation et à la configuration d'un cluster Kubernetes composé d'un nœud maître et de deux nœuds workers. Pour ce déploiement, nous utiliserons Containerd comme runtime de conteneurs afin d'assurer stabilité et performance.

Pour celles et ceux qui souhaitent bien maîtriser les bases, j'ai déjà publié un article dédié aux concepts fondamentaux de Kubernetes. Ici, nous passerons directement à la pratique à travers la mise en place concrète d'un cluster fonctionnel.

2. Prérequis Techniques

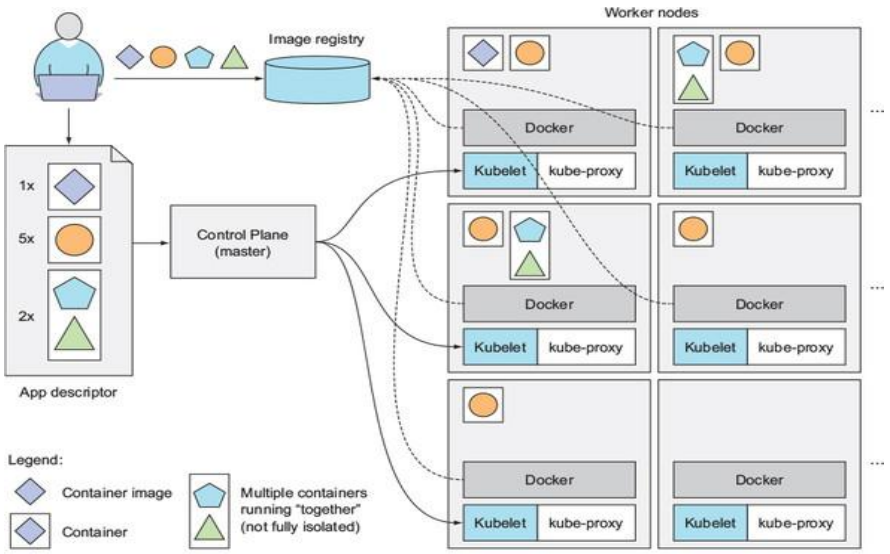
Avant de commencer, assurons-nous que notre environnement répond aux exigences suivantes :

- **Système d'exploitation : Ubuntu 22.04 LTS ou version plus récente**
- **Mémoire RAM : Minimum 2 Go par nœud (4 Go recommandés pour le maître)**
- **Processeur : Minimum 2 cœurs par nœud**
- **Réseau : Connectivité réseau entre tous les nœuds**
- **Privilèges : Accès administrateur (root) sur tous les nœuds**

3. Architecture du Cluster

Notre infrastructure Kubernetes sera organisée de la manière suivante :

- **1 Nœud Maître** : Responsable de la gestion du cluster, héberge l'API Server, etcd et le scheduler
- **2 Nœuds Workers** : Dédiés à l'exécution des pods applicatifs



4. Préparation de l'Environnement

4.1 Configuration Initiale

Commençons par préparer tous nos nœuds avec les outils essentiels. Nous allons exécuter les commandes suivantes sur chaque nœud (maître et workers) :

apt update -y

apt-get install -y curl openssh-server vim

4.2 Configuration de l'Accès SSH

Pour faciliter la gestion à distance de nos nœuds, nous allons configurer SSH :

sed -e 's/^.*PermitRootLogin prohibit-password/PermitRootLogin yes/g' -i /etc/ssh/sshd_config

systemctl restart ssh

Note : Cette configuration nous permettra de nous connecter facilement via des outils comme MobaXterm.

4.3 Désactivation du Pare-feu

Pour éviter les conflits de réseau pendant l'installation, nous allons désactiver le pare-feu :

Note : Nous configurons ici le nœud maître. Dans mon cas, j'ai initialement nommé mon maître "fode-mangane", puis je l'ai renommé en "k8sMaster".

```
root@fode-mangane:~# systemctl disable --now ufw
Synchronizing state of ufw.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install disable ufw
Removed "/etc/systemd/system/multi-user.target.wants/ufw.service".
root@fode-mangane:~#
```

4.4 Configuration des Modules Kernel

Nous devons activer certains modules kernel nécessaires au bon fonctionnement de Kubernetes :

```
root@fode-mangane:~# tee /etc/modules-load.d/containerd.conf <<EOF
overlay
br_netfilter
EOF
overlay
br_netfilter
root@fode-mangane:~#
```

4.5 Configuration Réseau Avancée

La configuration réseau est cruciale pour le fonctionnement du cluster :

```
root@fode-mangane:~# tee /etc/sysctl.d/kubernetes.conf<<EOF
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
EOF
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
root@fode-mangane:~#
```

```
root@fode-mangane:~# sysctl --system
* Applique /usr/lib/sysctl.d/10-apparmor.conf …
* Applique /etc/sysctl.d/10-bufferbloat.conf …
* Applique /etc/sysctl.d/10-console-messages.conf …
* Applique /etc/sysctl.d/10-ipv6-privacy.conf …
* Applique /etc/sysctl.d/10-kernel-hardening.conf …
* Applique /etc/sysctl.d/10-magic-sysrq.conf …
* Applique /etc/sysctl.d/10-map-count.conf …
* Applique /etc/sysctl.d/10-network-security.conf …
* Applique /etc/sysctl.d/10-ptrace.conf …
```

5. Installation de Containerd

5.1 Ajout du Dépôt Docker

Nous allons d'abord ajouter le dépôt officiel Docker pour installer Containerd :

```
root@fode-mangane:~# mkdir -m 0755 -p /etc/apt/keyrings
root@fode-mangane:~# curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.
gpg
root@fode-mangane:~#
```

```
root@fode-mangane:~# echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
root@fode-mangane:~#
```

5.2 Installation et Configuration de Containerd

Procédons maintenant à l'installation de Containerd :

```
root@fode-mangane:~# apt-get update && apt-get install -y containerd.io
Atteint :1 http://sn.archive.ubuntu.com/ubuntu noble InRelease
Atteint :2 http://security.ubuntu.com/ubuntu noble-security InRelease
Réception de :3 http://sn.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Atteint :4 http://sn.archive.ubuntu.com/ubuntu noble-backports InRelease
Atteint :5 https://download.docker.com/linux/ubuntu noble InRelease
```

Configurons ensuite Containerd pour qu'il fonctionne correctement avec Kubernetes :

```
root@fode-mangane:~# mkdir -p /etc/containerd
root@fode-mangane:~# containerd config default>/etc/containerd/config.toml
root@fode-mangane:~# sed -e 's/SystemdCgroup = false/SystemdCgroup = true/g' -i /etc/containerd/config.toml
root@fode-mangane:~#
```

Finalisons en redémarrant et activant le service :

```
root@fode-mangane:~# systemctl daemon-reload
root@fode-mangane:~# systemctl restart containerd
root@fode-mangane:~# systemctl enable containerd
root@fode-mangane:~#
```

6. Installation de Kubernetes

6.1 Configuration du Dépôt Kubernetes

Préparons notre système pour l'installation des composants Kubernetes :

```
root@fode-mangane:~# apt-get update && apt-get install -y apt-transport-https ca-certificates curl
Atteint :1 http://sn.archive.ubuntu.com/ubuntu noble InRelease
Atteint :2 http://security.ubuntu.com/ubuntu noble-security InRelease
Atteint :3 http://sn.archive.ubuntu.com/ubuntu noble-updates InRelease
Atteint :4 http://sn.archive.ubuntu.com/ubuntu noble-backports InRelease
Atteint :5 https://download.docker.com/linux/ubuntu noble InRelease
Lecture des listes de paquets... Fait
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
```

Ajoutons la clé de signature et le dépôt Kubernetes :

```
root@fode-mangane:~# curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.30/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyri
ngs/kubernetes-apt-keyring.gpg
root@fode-mangane:~#
```

```
root@fode-mangane:~# echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.3
0/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.30/deb/ /
root@fode-mangane:~#
```

6.2 Installation des Composants Kubernetes

Installons maintenant les outils essentiels de Kubernetes :

```
root@fode-mangane:~# apt-get update && apt-get install -y kubelet kubeadm kubectl
apt-mark hold kubelet kubeadm kubectl
Atteint :1 http://security.ubuntu.com/ubuntu noble-security InRelease
Atteint :2 https://download.docker.com/linux/ubuntu noble InRelease
Atteint :3 http://sn.archive.ubuntu.com/ubuntu noble InRelease
Atteint :4 http://sn.archive.ubuntu.com/ubuntu noble-updates InRelease
Atteint :5 http://sn.archive.ubuntu.com/ubuntu noble-backports InRelease
Réception de :6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.30/deb  InRelease [1 192 B]
Réception de :7 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.30/deb  Packages [19,0 kB]
```

Puis nous activons le service kubelet :

systemctl enable kubelet

7. Configuration du Nœud Maître

7.1 Initialisation du Cluster

Sur le nœud maître uniquement, nous allons initialiser notre cluster Kubernetes :

kubeadm init

Résultat attendu :

- Génération des certificats
- Configuration de l'API Server
- Création du token de join pour les workers

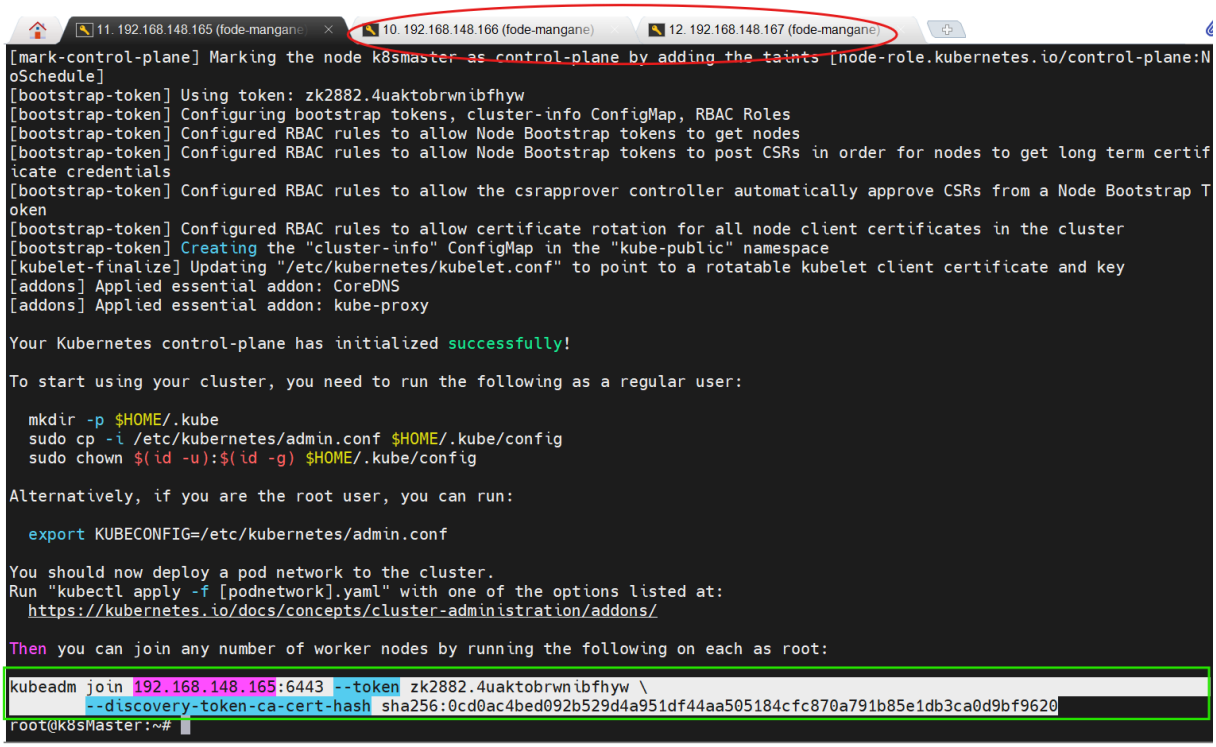
```
[bootstrap-token] Configured RBAC rules to allow the csrapprover controller automatically approve CSRs from a Node Bootstrap T
oken
[bootstrap-token] Configured RBAC rules to allow certificate rotation for all node client certificates in the cluster
[bootstrap-token] Creating the "cluster-info" ConfigMap in the "kube-public" namespace
[kubelet-finalize] Updating "/etc/kubernetes/kubelet.conf" to point to a rotatable kubelet client certificate and key
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 192.168.148.165:6443 --token mvysoc.04q7tlkxgxaz0aw0 \
        --discovery-token-ca-cert-hash sha256:5743958756e84e46cf8bcf865b3e0d77728c1895649cd34fd98aa58f1e5abd4e
```

8. Configuration des Nœuds Workers

8.1 Connexion des Workers au Cluster

Sur chaque nœud worker, nous allons utiliser la commande fournie lors de l'initialisation du maître :

kubeadm join <MASTER_IP>:6443 --token <TOKEN> \ --discovery-token-ca-cert-hash sha256:<HASH>



8.2 Configuration du Premier Worker

Important : Pour assurer une communication fluide, tous les nœuds doivent être connectés au même réseau que le nœud maître.

```
# Let NetworkManager manage all devices on this system
network:
  version: 2
  renderer: NetworkManager
  ethernets:
    ens33:
      dhcp4: no
      addresses:
        - 192.168.148.166/24
      gateway4: 192.168.148.1
      nameservers:
        addresses:
          - 8.8.8.8
          - 1.1.1.1
~
~
```

```
root@Worker1:~# kubeadm join 192.168.148.165:6443 --token zk2882.4uaktobrwnibfhyw         --discovery-token-ca-cert-hash sha256:0cd0ac4bed092b529d4a951df44aa505184cfc870a791b85e1db3ca0d9bf9620
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-check] Waiting for a healthy kubelet at http://127.0.0.1:10248/healthz. This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 502.088941ms
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.

root@Worker1:~#
```

8.3 Configuration du Deuxième Worker

```
# Let NetworkManager manage all devices on this system
network:
  version: 2
  renderer: NetworkManager
  ethernets:
    ens33:
      dhcp4: no
      addresses:
        - 192.168.148.167/24
      gateway4: 192.168.148.1
      nameservers:
        addresses:
          - 8.8.8.8
          - 1.1.1.1
~
~
~
```

```
root@Worker2:~# kubeadm join 192.168.148.165:6443 --token zk2882.4uaktobrwnibfhyw \
        --discovery-token-ca-cert-hash sha256:0cd0ac4bed092b529d4a951df44aa505184cfc870a791b85e1db3ca0d9bf9620
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-check] Waiting for a healthy kubelet at http://127.0.0.1:10248/healthz. This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 502.391949ms
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.

root@Worker2:~#
```

9. Vérification du Cluster

9.1 Vérification de l'État des Nœuds

Vérifions que tous nos nœuds sont correctement connectés :

```
root@k8sMaster:~# kubectl get nodes
NAME        STATUS     ROLES           AGE     VERSION
k8smaster   NotReady   control-plane   2m12s   v1.30.13
worker1     NotReady   <none>          63s     v1.30.13
worker2     NotReady   <none>          60s     v1.30.13
root@k8sMaster:~#
```

9.2 Vérification des Pods Système

Contrôlons le bon fonctionnement des composants système :

```
root@k8sMaster:~# kubectl get pods -A
NAMESPACE     NAME                                READY   STATUS    RESTARTS        AGE
kube-system   coredns-55cb58b774-pdxgj            0/1     Pending   0               3m3s
kube-system   coredns-55cb58b774-vdzw7            0/1     Pending   0               3m3s
kube-system   etcd-k8smaster                      1/1     Running   5               3m3s
kube-system   kube-apiserver-k8smaster            1/1     Running   6 (3m22s ago)   3m3s
kube-system   kube-controller-manager-k8smaster   1/1     Running   2               3m3s
kube-system   kube-proxy-4vtvc                    1/1     Running   0               115s
kube-system   kube-proxy-qnqs8                    1/1     Running   0               112s
kube-system   kube-proxy-xdvbd                    1/1     Running   0               3m3s
kube-system   kube-scheduler-k8smaster            1/1     Running   6               3m3s
root@k8sMaster:~#
```

État attendu :

- Tous les nœuds doivent afficher le statut "Ready"
- Tous les pods système doivent être en statut "Running"

10. Commandes de Maintenance Utiles

10.1 Surveillance du Cluster

Voici quelques commandes essentielles pour surveiller votre cluster :

❖ Voir tous les nœuds avec détails

kubectl get nodes -o wide

❖ Consulter les événements du cluster

kubectl get events

❖ Obtenir des informations détaillées sur un nœud

kubectl describe node <nom-du-noeud>

```
root@k8sMaster:~# kubectl describe node worker1
Name:               worker1
Roles:              <none>
Labels:             beta.kubernetes.io/arch=amd64
                    beta.kubernetes.io/os=linux
                    kubernetes.io/arch=amd64
                    kubernetes.io/hostname=worker1
                    kubernetes.io/os=linux
Annotations:        kubeadm.alpha.kubernetes.io/cri-socket: unix:///var/run/containerd/containerd.sock
                    node.alpha.kubernetes.io/ttl: 0
                    volumes.kubernetes.io/controller-managed-attach-detach: true
CreationTimestamp:  Thu, 22 May 2025 20:07:21 +0000
Taints:             node.kubernetes.io/not-ready:NoSchedule
Unschedulable:      false
Lease:
  HolderIdentity:  worker1
  AcquireTime:     <unset>
  RenewTime:       Thu, 22 May 2025 20:10:54 +0000
Conditions:
  Type             Status  LastHeartbeatTime                 LastTransitionTime                Reason
Message
  ----             ------  -----------------                 ------------------                ------
-------
  MemoryPressure   False   Thu, 22 May 2025 20:07:29 +0000   Thu, 22 May 2025 20:07:18 +0000   KubeletHasSufficientMemory
kubelet has sufficient memory available
  DiskPressure     False   Thu, 22 May 2025 20:07:29 +0000   Thu, 22 May 2025 20:07:18 +0000   KubeletHasNoDiskPressure
kubelet has no disk pressure
  PIDPressure      False   Thu, 22 May 2025 20:07:29 +0000   Thu, 22 May 2025 20:07:18 +0000   KubeletHasSufficientPID
kubelet has sufficient PID available
  Ready            False   Thu, 22 May 2025 20:07:29 +0000   Thu, 22 May 2025 20:07:18 +0000   KubeletNotReady
container runtime network not ready: NetworkReady=false reason:NetworkPluginNotReady message:Network plugin returns error: c
ni plugin not initialized
Addresses:
  InternalIP:  192.168.148.166
```

11. Conclusion

Ce guide de déploiement d'un cluster Kubernetes multi-nœuds a permis de réaliser l'ensemble des étapes nécessaires à la mise en place d'une infrastructure distribuée, fiable et évolutive. De la préparation des nœuds à l'intégration des workers au cluster principal, chaque étape a été soigneusement exécutée afin d'assurer un environnement de conteneurisation optimisé et prêt pour des déploiements futurs. Voici les points clés à retenir :

- **Préparation complète de l'environnement** : Configuration des nœuds Ubuntu avec tous les prérequis
- **Installation et configuration de Containerd** : Runtime de conteneurs optimisé pour Kubernetes
- **Déploiement du cluster multi-nœuds** : Un maître et deux workers parfaitement synchronisés

- **Vérification et maintenance** : Outils et commandes pour surveiller votre infrastructure

Cette réalisation marque le début d'une aventure passionnante dans l'univers de l'orchestration de conteneurs. Votre cluster est maintenant prêt à héberger des applications complexes, gérer l'auto-scaling, et orchestrer des déploiements à grande échelle.

Et ce n'est qu'un début ! Dans les prochains tutoriels, nous explorerons ensemble des concepts plus avancés tels que le déploiement d'applications via **GitLab CI/CD**, la réalisation de tests de performance avec **JMeter**, l'analyse de la qualité du code avec **SonarQube**, l'automatisation des déploiements continus avec **ArgoCD**, et bien d'autres pratiques essentielles pour renforcer vos environnements **Kubernetes** et **DevOps**.

Si ce guide vous a été utile et que vous souhaitez poursuivre votre apprentissage avec des contenus de qualité sur les technologies cloud natives, n'hésitez pas à suivre mon compte linkedin (Fode Mangane | LinkedIn) pour ne manquer aucune publication.

Partagez vos réussites et vos questions en commentaires, j'ai hâte de voir vos déploiements et d'échanger avec vous sur vos projets Kubernetes !

Remerciements

Je tiens à exprimer ma sincère gratitude à Monsieur **Massamba Lo**, mon enseignant, pour son accompagnement et ses précieux conseils tout au long de ce projet. Son expertise et sa pédagogie ont été déterminantes dans la réussite de ce déploiement Kubernetes.