Exercice Professionnel : Déploiement d'un Nginx Non-Root avec OpenShift et Terraform

🎯Objectif

Mettre en place une application web basée sur **Nginx non-root** dans l'environnement OpenShift Sandbox, en automatisant la création des ressources nécessaires avec **Terraform**. L'application sera ensuite exposée au public via une **Route**.

---

1. Pré-requis

- Un compte actif sur OpenShift Sandbox.
- `oc` CLI configuré (`oc login ...`).
- `terraform` installé et ajouté au `PATH` .
- Droits développeur dans le projet par défaut : `contact-walid-labidi-dev` .

---

2. Étapes du Déploiement

Étape 1 : Initialisation du projet Terraform

Dans PowerShell :

```
mkdir C:\terraform\nginx
cd C:\terraform\nginx
notepad main.tf
```

Étape 2 : Écriture du fichier `main.tf`

```
terraform {
  required_providers {
    kubernetes = {
      source  = "hashicorp/kubernetes"
      version = "2.27.0"
    }
  }
}

provider "kubernetes" {
  config_path = "~/.kube/config"
}

# Deployment Nginx (non-root)
```

```
resource "kubernetes_deployment" "nginx_nonroot" {
  metadata {
    name      = "nginx-nonroot"
    namespace = "contact-walid-labidi-dev"
    labels = {
      app = "nginx-nonroot"
    }
  }

  spec {
    replicas = 1
    selector {
      match_labels = {
        app = "nginx-nonroot"
      }
    }

    template {
      metadata {
        labels = {
          app = "nginx-nonroot"
        }
      }

      spec {
        container {
          name  = "nginx"
          image = "nginxinc/nginx-unprivileged:1.25-alpine"

          port {
            container_port = 8080
          }
        }
      }
    }
  }
}

# Service
resource "kubernetes_service" "nginx_svc" {
  metadata {
    name      = "nginx-nonroot"
    namespace = "contact-walid-labidi-dev"
  }

  spec {
    selector = {
      app = "nginx-nonroot"
    }

    port {
```

```
      port        = 80
      target_port = 8080
    }

    type = "ClusterIP"
  }
}

# Route OpenShift via local-exec
resource "null_resource" "nginx_route" {
  provisioner "local-exec" {
    command = "oc -n contact-walid-labidi-dev expose service/nginx-nonroot"
  }

  depends_on = [kubernetes_service.nginx_svc]
}
```

---

3. Exécution des Commandes

Initialiser Terraform

```
terraform init
```

Appliquer le plan

```
terraform apply -auto-approve
```

---

4. Vérification

Lister les pods

```
oc get pods -n contact-walid-labidi-dev
```

👉Résultat attendu :

```
nginx-nonroot-xxxxx   1/1   Running   0   20s
```

Lister les services

```
oc get svc -n contact-walid-labidi-dev
```

👉Résultat attendu :

```
nginx-nonroot   ClusterIP   172.30.xx.xx   <none>   80/TCP   20s
```

Lister les routes

```
oc get route -n contact-walid-labidi-dev
```

👉Résultat attendu :

```
nginx-nonroot   nginx-nonroot-contact-walid-labidi-dev.apps.sandbox...
nginx-nonroot   8080   None
```

---

5. Test de l'Application

Ouvrir dans le navigateur :

```
http://nginx-nonroot-contact-walid-labidi-dev.apps.sandbox-
m2.1530.p1.openshiftapps.com
```

👉La page d'accueil **Welcome to Nginx** s'affiche.

Résultat Final

- Déploiement automatisé de Nginx **non-root** avec Terraform.
- Service accessible dans le cluster.
- Route exposée automatiquement via `oc expose`.
- Application consultable publiquement via une URL OpenShift.

Explications Importantes

- **Pourquoi** `nginxinc/nginx-unprivileged` **?** → Image officielle de Nginx fonctionnant en mode utilisateur (UID 101, port 8080), compatible avec OpenShift (qui refuse les containers root par défaut).
- **Pourquoi** `local-exec` **?** → Le provider Kubernetes ne gère pas directement les Routes OpenShift. On contourne avec `oc expose` exécuté automatiquement après la création du Service.
- **Pourquoi namespace fixe ?** → Dans OpenShift Sandbox, l'utilisateur n'a pas le droit de créer des namespaces, il doit travailler dans celui qui lui est attribué (`contact-walid-labidi-dev`).