Exercice : Déploiement Image Open shift , Github via Terraform

But de l'exercice : déployer une page HTML statique sur **OpenShift Sandbox** en utilisant une **image non-root**, avec le code source stocké sur **GitHub**, et automatiser la création des ressources avec **Terraform**.

Rôles :

- **GitHub** : dépôt du code et (facultatif) des fichiers Terraform ; gestion de versions et collaboration.
- **Terraform** : décrit l'infrastructure comme du code (Deployment, Service). Il applique ces ressources vers le cluster (idempotent).
- **OpenShift** : plateforme d'exécution (basée sur Kubernetes) qui crée les Pods, Services et expose l'application par Route.

Remarque Sandbox : dans Developer Sandbox, le provider Terraform « kubernetes » ne peut pas lister les CRDs cluster-scope. On crée donc la **Route** via `oc expose` (provisionner en `local-exec`) au lieu d'un objet Terraform « Route ».

Énoncé

1. Créer un dépôt GitHub nommé `site-web` contenant un fichier `index.html` simple.
2. Déployer l'application sur OpenShift en utilisant une image **non-root** (ex. `registry.access.redhat.com/ubi8/httpd-24:latest`).
3. Exposer l'application via une **Route** publique.
4. Écrire un fichier `main.tf` qui crée automatiquement le **Deployment** et le **Service** (image non-root), puis exécuter un `oc expose` via Terraform pour créer la **Route**.

Correction détaillée

A. Préparer le dépôt GitHub

Créez un répertoire de travail et un fichier HTML minimal :

```
mkdir site-web
cd site-web
notepad index.html
```

Contenu proposé de `index.html` :

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Hello OpenShift</title>
  </head>
  <body>
    <h1>Hello depuis OpenShift (image non-root) !</h1>
  </body>
</html>
```

Initialiser Git et pousser sur GitHub (branche principale `main`) :

```
git init
git add .
git commit -m "Initial commit site-web"
git branch -M main
git remote add origin https://github.com/<votre-user>/site-web.git
# Première fois : Git va demander un username + password => utilisez le PAT
GitHub comme "password"
git push -u origin main
```

Vérifiez sur GitHub que `index.html` est visible dans `main`.

B. Déployer manuellement sur OpenShift (validation rapide)

Connexion (si nécessaire) :

```
oc login --token=<TON_TOKEN> --server=<URL_CLUSTER>
```

Création de l'application en image non-root (httpd UBI8) avec source GitHub :

```
oc new-app registry.access.redhat.com/ubi8/httpd-24:latest~https://
github.com/<votre-user>/site-web.git --name=site-web
```

Si erreur de tag, essayez sans tag explicite :

```
oc new-app registry.access.redhat.com/ubi8/httpd-24~https://github.com/
<votre-user>/site-web.git --name=site-web
```

Vérification :

```
oc get pods -n <votre-namespace>
oc logs -f deploy/site-web -n <votre-namespace>
```

Exposer le service :

```
oc expose service site-web -n <votre-namespace>
oc get route site-web -n <votre-namespace> -o jsonpath="{.spec.host}{'\n'}"
```

Ouvrez l'URL renvoyée dans votre navigateur pour valider le rendu. Une fois validé, vous pouvez supprimer ces ressources manuelles si vous souhaitez ne garder que la version « codée » par Terraform :

```
oc delete all -l app=site-web -n <votre-namespace>
```

C. Automatiser avec Terraform (Deployment + Service) + Route via `local-exec`

Dans le répertoire `site-web`, créez `main.tf` :

```
terraform {
  required_providers {
    kubernetes = {
      source  = "hashicorp/kubernetes"
      version = ">= 2.29"
    }
  }
}

provider "kubernetes" {
  # Terraform réutilise le contexte kubeconfig actuel (après oc login)
  config_path = "~/.kube/config"
}

# Deployment (image non-root httpd UBI8)
resource "kubernetes_deployment" "site_web" {
  metadata {
    name      = "site-web-tf"
    namespace = "<votre-namespace>"        # ex: contact-walid-labidi-dev
    labels = { app = "site-web-tf" }
  }
  spec {
    replicas = 1
    selector { match_labels = { app = "site-web-tf" } }
    template {
      metadata { labels = { app = "site-web-tf" } }
      spec {
```

```
        container {
          name  = "httpd"
          image = "registry.access.redhat.com/ubi8/httpd-24:latest"
          port  { container_port = 8080 }
        }
      }
    }
  }
}

# Service (expose le port 80 vers le container 8080)
resource "kubernetes_service" "site_web" {
  metadata {
    name      = "site-web-tf"
    namespace = "<votre-namespace>"
  }
  spec {
    selector = { app = "site-web-tf" }
    port {
      port        = 80
      target_port = 8080
    }
    type = "ClusterIP"
  }
}

# Création de la Route via oc (local-exec) car les CRDs Route ne sont pas
listables en Sandbox
resource "null_resource" "create_route" {
  provisioner "local-exec" {
    command = "oc expose service site-web-tf -n <votre-namespace>"
  }
  provisioner "local-exec" {
    command = "oc get route site-web-tf -n <votre-namespace> -o
jsonpath={.spec.host}"
  }
  depends_on = [kubernetes_service.site_web]
}
```

Appliquer Terraform :

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

Récupérer l'URL (si non imprimée) :

```
oc get route site-web-tf -n <votre-namespace> -o jsonpath="{.spec.host}
{'\n'}"
```

Ouvrez l’URL dans le navigateur et validez l’affichage.

Nettoyage (facultatif)

Supprimer les objets créés par Terraform :

```
terraform destroy -auto-approve
```

Si la Route a été créée en dehors du state Terraform (via `local-exec`), supprimez-la au besoin :

```
oc delete route site-web-tf -n <votre-namespace>
```
