# Formation Kubernetes - Travaux pratiques

## TP 3 - DÉPLOYER EN UTILISANT DES FICHIERS RESSOURCE.

**Objectif** : Écrire et utiliser des fichiers de configuration YAML pour déployer les applications.

Commencez par supprimer les ressources **demonstration** et **demonstration-service** du TP2.

Dans ce TP nous allons redéployer notre application **demonstration** du TP2 mais cette fois en utilisant **kubectl apply -f** et en visualisant le résultat dans minikube dashboard.

1. Créez un dossier **TP2_deploy_using_files** sur le bureau de la machine et ouvrez le avec VSCode. Nous allons d'abord déployer notre application comme un simple Pod (non recommandé mais montré ici pour l'exercice).
2. Créez un fichier **demo-pod.yaml** avec à l'intérieur le code d'exemple du TP (à voir plus bas)
3. Appliquez le ficher avec **kubectl apply -f <fichier>**
4. Constatez dans Minikube dashboard.
5. Modifiez le nom du pod dans la description précédente et réappliquez la configuration. Kubernetes met à jour le nom.
6. Modifier le nom du conteneur **rancher-demo** et réappliquez la configuration. Que se passe-t-il ?

Kubernetes refuse d'appliquer le nouveau nom de conteneur car un pod est largement immutable. Pour changer d'une quelquonque façon les conteneurs du pod il faut supprimer (kubectl delete -f <fichier>) et recréer le pod. Mais ce travail de mise à jour devrait être géré par un déploiement pour automatiser et pour garantir la haute disponibilité de notre application demonstration.

Kubernetes fournit un ensemble de commande pour débugger des conteneurs :

**kubectl logs <pod-name> -c <conteneur_name>**
(le nom du conteneur est inutile si un seul)

**kubectl exec -it <pod-name> -c <conteneur_name> -- bash**

**kubectl attach -it <pod-name>**

Explorez le pod avec la commande **kubectl exec -it <pod-name> -c <conteneur_name> -- bash** écrite plus haut.

Supprimez le pod.

**Avec un déploiement (méthode à utiliser)**

Créez un fichier demo-deploy.yaml avec à l'intérieur le code suivant à compléter:

1. Appliquez ce nouvel objet avec kubectl.
2. Inspectez le déploiement dans Minikube dashboard .
3. Changez le nom d'un conteneur et réappliquez: Cette fois le déploiement se charge de créer un nouveau pod avec les bonnes caractéristiques et de supprimer l'ancien.
4. Changez le nombre de réplicats.

**Ajoutons un service en mode NodePort**

1. Créez un fichier demo-svc.yaml avec à l'intérieur le code suivant à compléter:
2. Appliquez ce nouvel objet avec kubectl.
3. Inspectez le service dans Lens.
4. Visitez votre application avec l'Internal ip du noeud (à trouver dans les information du node) et le nodeport (port 3xxxx) associé au service, le nombre de réplicat devrait apparaître.
5. Pour tester, changez le label du selector dans le service (lignes nom-app: demonstration et partie: les-petits-pods-demo à remplacer dans le fichier ) et réappliquez.
6. Constatez que l'application n'est plus accessible dans le navigateur. Pourquoi ?

Les services kubernetes redirigent le trafic basés sur les étiquettes (labels) appliquées sur les pods du cluster. Il faut donc éviter d'utiliser deux fois le même label pour des parties différentes de l'application.

## Un manifeste de Pods (demo-pod.yaml)

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: rancher-demo-pod
spec:
  containers:
    - image: monachus/rancher-demo:latest
      name: rancher-demo-container
      ports:
        - containerPort: 8080
          name: http
          protocol: TCP
    - image: redis
      name: redis-container
      ports:
        - containerPort: 6379
          name: http
          protocol: TCP
```

## Deployment

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  strategy:
    type: Recreate
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.7.9
          ports:
            - containerPort: 80
```

## Service (demo-svc.yaml)

```yaml
apiVersion: v1
kind: Service
metadata:
  name: demo-service
  labels:
    nom-app: demonstration
    partie: le-fameux-service-demo
spec:
  ports:
    - port: <port>
  selector:
    nom-app: demonstration
    partie: les-petits-pods-demo
  type: NodePort
```