Technologie des conteneurs

TP4 : ZZ Book en production

Dans ce TP, vous devez écrire l'ensemble des manifests nécessaires au fonctionnement de votre application.

Q1 / Dans le TP3, vous devez avoir créé les objets nécessaires à l'exécution de vos micro-services, mais les pods ne peuvent pas communiquer entre eux ! Ajoutez les objets K8S Services afin que la communication entre pods soit possible.

Voir les fichiers xxxx_service.yaml dans le dossier kubernetes/fil-rouge/

Q2 / Afin d'accéder au micro-service productpage, modifiez le Service correspondant. Quel type devez vous utiliser ? À quelle IP pouvez-vous accéder à l'application ?

On doit utiliser un service de type NodePort.
On peut accéder l'application via l'internal ip de la node control-plane : 127.18.0.2

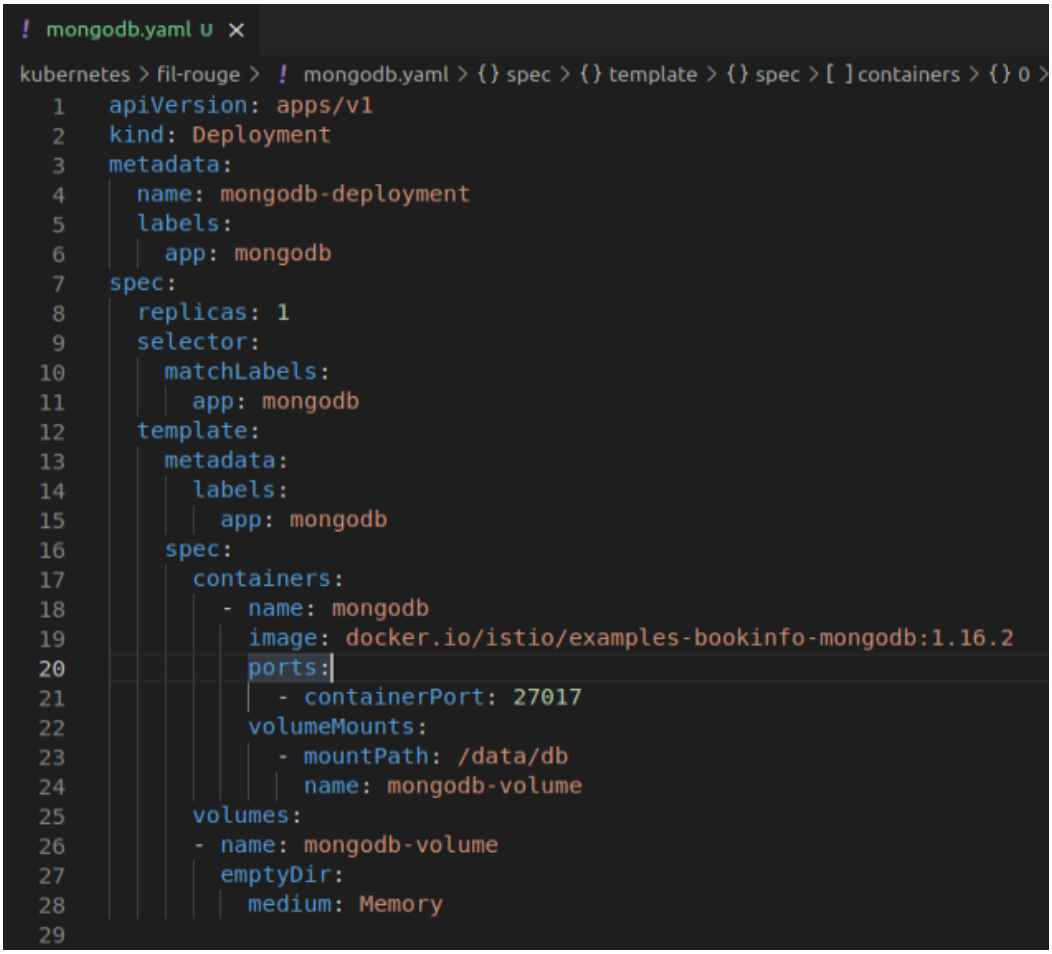
```
root@osboxes:/home/osboxes/tp-containers/kubernetes/fil-rouge# kubectl get nodes -o wide
NAME                 STATUS   ROLES                  AGE    VERSION   INTERNAL-IP   EXTERNAL-IP   OS-IMAGE       KERNEL-VERSION      CONTAINER-RUNTIME
kind-control-plane   Ready    control-plane,master   8m47s   v1.23.4   172.18.0.2    <none>        Ubuntu 21.10   5.13.0-19-generic   containerd://1.5.10
root@osboxes:/home/osboxes/tp-containers/kubernetes/fil-rouge#
```

Modification du service correspondant :

```
! productpage_service.yaml U ×
kubernetes > fil-rouge > ! productpage_s
  1  apiVersion: v1
  2  kind: Service
  3  metadata:
  4    name: productpage
  5  spec:
  6    type: NodePort
  7    selector:
  8      app: productpage
  9    ports:
 10      - protocol: TCP
 11        port: 9080
 12        nodePort: 30000
```

Q3 / Ajoutez les manifests nécessaires au déploiement d'une base mongodb. Utilisez un Deployment avec l'image *docker.io/istio/examples-bookinfo-mongodb:1.16.2*
N'oubliez pas d'indiquer un volume dans votre Deployment, afin de conserver le contenu de /data/db.

Technologie des conteneurs



Q4 / De la même manière, déployez mysql via un Deployment et l'image docker.io/istio/examples-bookinfo-mysqldb:1.16.2
Vous devrez créer un Secret afin de stocker le mot de passe root, et monter ce secret dans un variable d'environnement MYSQL_ROOT_PASSWORD
Pour démarrer le conteur, il aura besoin des arguments suivants : ["--default-authentication-plugin","mysql_native_password"]
Enfin, n'oubliez pas de spécifier un volume pour le dossier /var/lib/mysql

Technologie des conteneurs

mysql.yaml U

kubernetes > fil-rouge > mysql.yaml > {} spec > {} template > {} spec > [ ] containers > {} 0 > [ ] env > {} 0 > {}

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: mysql-credentials
type: Opaque
data:
  rootpasswd: cGlsaXBpbGk2Mw==
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql-deployment
  labels:
    app: mysql
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mysql
  template:
    metadata:
      labels:
        app: mysql
    spec:
      containers:
        - name: mysql
          image: docker.io/istio/examples-bookinfo-mysqldb:1.16.2
          ports:
            - containerPort: 3306
          env:
            - name: MYSQL_ROOT_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: mysql-credentials
                  key: rootpasswd
          args: ["--default-authentication-plugin","mysql_native_password"]
          volumeMounts:
            - mountPath: /var/lib/mysql
              name: mysql-volume
      volumes:
      - name: mysql-volume
        emptyDir:
          medium: Memory
```

Q5 / Ajoutez 5 replicas à ratings, puis mettez à jour l'image vers la v2, observez le déploiement des pods. Vous pouvez également tenter d'accéder à l'application afin de constater que le load-balancing fonctionne (en rechargeant la page).

Q6 / Ajoutez un Deployment de ratings avec la v2, afin d'avoir à la fois 5 pods v1 et 5 pods v2 dans le cluster. Utilisez les labels et le Service afin de faire un déploiement Blue/Green (changement du trafic de la v1 à la v2 instantané).

Technologie des conteneurs

On utitilise deux deployments, appelés ratings-blue et ratings-green et disposant du label app : ratings-blue ou app : ratings-green. Un deployment est en V1, l'autre en V2.

Dans le service qui gère le routage de ratings, on utilise le selector app : ratings-blue ou ratings-green pour acheminer le trafic vers la version que l'on souhaite.

ratings_service.yaml U

kubernetes > fil-rouge > ratings_service.yaml > {} spec > [ ] ports > {} 0 > # targetPort

```yaml
apiVersion: v1
kind: Service
metadata:
  name: ratings
spec:
  selector:
    app: ratings-blue
  ports:
    - protocol: TCP
      port: 9080
      targetPort: 9080
```

ratings.yaml U

kubernetes > fil-rouge > ratings.yaml > {} spec > {} template > {} metadata

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ratings-deployment-blue
  labels:
    app: ratings-blue
spec:
  replicas: 5
  selector:
    matchLabels:
      app: ratings-blue
  template:
    metadata:
      labels:
        app: ratings-blue
    spec:
      containers:
        - name: ratings
          image: localhost:5001/ratings:1.0
          ports:
            - containerPort: 9080
          env:
            - name: SERVICE_VERSION
              value: "v1"
          readinessProbe:
            httpGet:
              path: /health
              port: 9080
            initialDelaySeconds: 10
            periodSeconds: 10
          livenessProbe:
            exec:
              command:
                - ls
            initialDelaySeconds: 10
            periodSeconds: 10
---
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ratings-deployment-green
  labels:
    app: ratings-green
spec:
  replicas: 5
  selector:
    matchLabels:
      app: ratings-green
  template:
    metadata:
      labels:
        app: ratings-green
    spec:
      containers:
        - name: ratings
          image: localhost:5001/ratings:1.0
          ports:
            - containerPort: 9080
          env:
            - name: SERVICE_VERSION
              value: "v2"
          readinessProbe:
            httpGet:
              path: /health
              port: 9080
            initialDelaySeconds: 10
            periodSeconds: 10
          livenessProbe:
            exec:
              command:
                - ls
            initialDelaySeconds: 10
            periodSeconds: 10
```

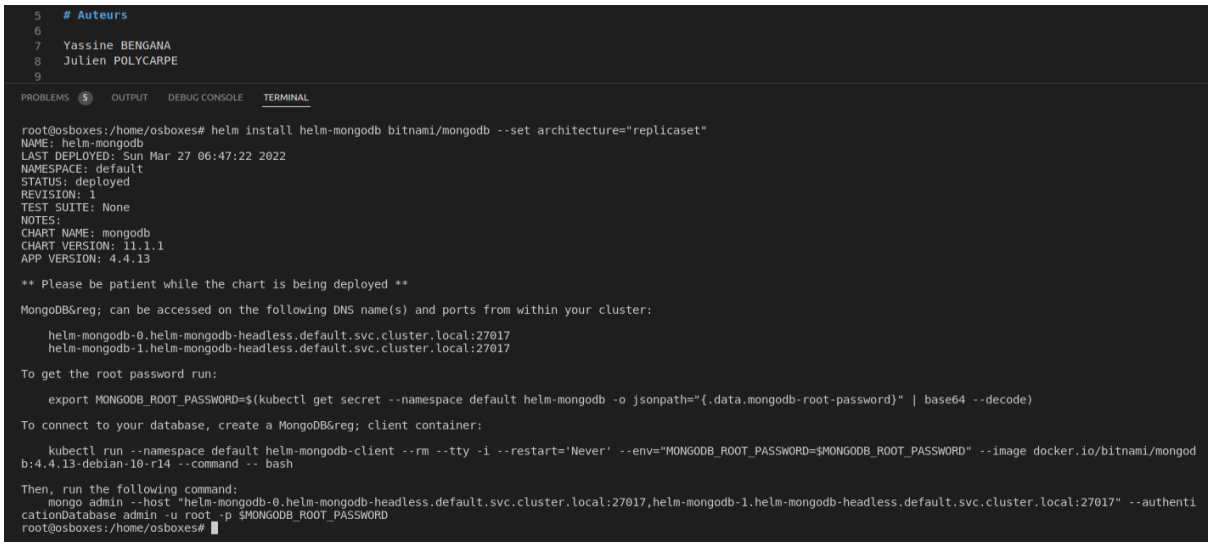
Technologie des conteneurs

Q7 / Ajoutez une liveness probe et une readiness probe dans vos Deployment (pas nécessaire sur les DB dans le projet).

Voir dans chaque deployment, exemple ci-dessous :

```
62          readinessProbe:
63            httpGet:
64              path: /health
65              port: 9080
66            initialDelaySeconds: 10
67            periodSeconds: 10
68          livenessProbe:
69            exec:
70              command:
71                - ls
72            initialDelaySeconds: 10
73            periodSeconds: 10
```

Q8 (Bonus) / Mettez le nombre de replicas à 0 sur les Deployments de Mysql et de MongoDB. Installez MongoDB en tant que ReplicaSet à l'aide de Helm https://github.com/bitnami/charts/tree/master/bitnami/mongodb



Q9 (Bonus) / Écrivez Ingress afin de permettre l'accès à votre application, avec des fonctionnalités de couche 7. Sur kind, il faudra vous aider de cette documentation : https://kind.sigs.k8s.io/docs/user/ingress/

Voir dossier ingress_test

IMPORTANT : Pour le rendu du TP, vérifiez que :

- Les fichiers (code, Dockerfile, docker-compose.yml, manifests k8s) sont bien présents
- Un README à la racine indique :

Technologie des conteneurs

    - **Vos noms**
    - L'emplacement des différents éléments s'ils sont dans des sous-dossiers.
    - Une documentation si vous le jugez nécessaire
- Les éventuels compte-rendus des TPs sont présents (format texte, markdown, pdf, essayez d'éviter le docx…)

Envoyez le lien de votre dépôt (si il est public) ou un .zip à benjamin@kiowy.com **avant le ~~13/03/2022 à 23h59~~ ⇒ 27/03/2022 à 23h59.**

Pour infos : Je vais tester "l'éxécutabilité" de vos projets, c.a.d. que je vais :

- Récupérer votre dépôt/dézipper
- Construire vos images à l'aide de Docker (docker build)
- Démarrer votre projet avec docker-compose up et tenter d'y accéder
- Pousser les images construites vers un registre privé Google Cloud
- Faire kubectl apply -f . de vos manifests
- Tenter d'accéder à votre application (et vérifier l'exécution des pods)

L'ensemble de ces points représentent ⅔ de la note, le tier restant est fonction de la qualité de vos Dockerfile, docker-compose, manifests Kubernetes…