

# 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 : 172.18.0.2

```
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
```

Modification du service correspondant :

```
! productpage_service.yaml x
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](https://hub.docker.com/r/istio/examples-bookinfo-mongodb)**

**N'oubliez pas d'indiquer un volume dans votre Deployment, afin de conserver le contenu de /data/db.**

# Technologie des conteneurs

```
! mongodb.yaml u x
kubernetes > fil-rouge > ! mongodb.yaml > {} spec > {} template > {} spec > [ ] containers > {} 0 >
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: mongodb-deployment
5    labels:
6      app: mongodb
7  spec:
8    replicas: 1
9    selector:
10     matchLabels:
11       app: mongodb
12    template:
13     metadata:
14       labels:
15         app: mongodb
16     spec:
17       containers:
18         - name: mongodb
19           image: docker.io/istio/examples-bookinfo-mongodb:1.16.2
20           ports:
21             - containerPort: 27017
22           volumeMounts:
23             - mountPath: /data/db
24               name: mongodb-volume
25       volumes:
26         - name: mongodb-volume
27           emptyDir:
28             medium: Memory
29
```

**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 x
kubernetes > fil-rouge > ! mysql.yaml > {} spec > {} template > {} spec > [ ] containers > {} 0 > [ ] env > {} 0 > {}
1  apiVersion: v1
2  kind: Secret
3  metadata:
4    name: mysql-credentials
5  type: Opaque
6  data:
7    rootpasswd: cGlsaXBpbGk2Mw==
8  ---
9  apiVersion: apps/v1
10 kind: Deployment
11 metadata:
12   name: mysql-deployment
13   labels:
14     app: mysql
15 spec:
16   replicas: 1
17   selector:
18     matchLabels:
19       app: mysql
20   template:
21     metadata:
22       labels:
23         app: mysql
24     spec:
25       containers:
26       - name: mysql
27         image: docker.io/istio/examples-bookinfo-mysqldb:1.16.2
28         ports:
29         - containerPort: 3306
30         env:
31         - name: MYSQL_ROOT_PASSWORD
32           valueFrom:
33             secretKeyRef:
34               name: mysql-credentials
35               key: rootpasswd
36         args: ["--default-authentication-plugin","mysql_native_password"]
37         volumeMounts:
38         - mountPath: /var/lib/mysql
39           name: mysql-volume
40       volumes:
41       - name: mysql-volume
42         emptyDir:
43           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 utilise 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 x
kubernetes > fil-rouge > ! ratings_service.yaml > {} spec > [] ports > {} 0 > # targetPort
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: ratings
5  spec:
6    selector:
7      app: ratings-blue
8    ports:
9      - protocol: TCP
10        port: 9080
11        targetPort: 9080

! ratings.yaml u x
kubernetes > fil-rouge > ! ratings.yaml > {} spec > {} template > {} metadata
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: ratings-deployment-blue
5    labels:
6      app: ratings-blue
7  spec:
8    replicas: 5
9    selector:
10     matchLabels:
11       app: ratings-blue
12   template:
13     metadata:
14       labels:
15         app: ratings-blue
16     spec:
17       containers:
18         - name: ratings
19           image: localhost:5001/ratings:1.0
20           ports:
21             - containerPort: 9080
22           env:
23             - name: SERVICE_VERSION
24               value: "v1"
25           readinessProbe:
26             httpGet:
27               path: /health
28               port: 9080
29             initialDelaySeconds: 10
30             periodSeconds: 10
31           livenessProbe:
32             exec:
33               command:
34                 - ls
35             initialDelaySeconds: 10
36             periodSeconds: 10
37   ...
```

```
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>**

```
5 # Auteurs
6
7 Yassine BENGANA
8 Julien POLYCARPE
9
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
root@osboxes:/home/osboxes# helm install helm-mongodb bitnami/mongodb --set architecture="replicaset"
NAME: helm-mongodb
LAST DEPLOYED: Sun Mar 27 06:47:22 2022
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
CHART NAME: mongodb
CHART VERSION: 11.1.1
APP VERSION: 4.4.13

** Please be patient while the chart is being deployed **

MongoDB&reg; can be accessed on the following DNS name(s) and ports from within your cluster:

    helm-mongodb-0.helm-mongodb-headless.default.svc.cluster.local:27017
    helm-mongodb-1.helm-mongodb-headless.default.svc.cluster.local:27017

To get the root password run:

    export MONGODB_ROOT_PASSWORD=$(kubectl get secret --namespace default helm-mongodb -o jsonpath="{.data.mongodb-root-password}" | base64 --decode)

To connect to your database, create a MongoDB&reg; client container:

    kubectl run --namespace default helm-mongodb-client --rm --tty -i --restart='Never' --env="MONGODB_ROOT_PASSWORD=$MONGODB_ROOT_PASSWORD" --image docker.io/bitnami/mongo
b:4.4.13-debian-10-r14 --command -- bash

Then, run the following command:

    mongo admin --host "helm-mongodb-0.helm-mongodb-headless.default.svc.cluster.local:27017,helm-mongodb-1.helm-mongodb-headless.default.svc.cluster.local:27017" --authenti
cationDatabase admin -u root -p $MONGODB_ROOT_PASSWORD
root@osboxes:/home/osboxes#
```

**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](mailto:benjamin@kiowy.com)  
**avant le ~~13/03/2022 à 23h59~~ ⇒ 27/03/2022 à 23h59.**

Pour infos : Je vais tester "l'exécutabilité" 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  $\frac{2}{3}$  de la note, le tier restant est fonction de la qualité de vos Dockerfile, docker-compose, manifests Kubernetes...