Repository GitHub : https://github.com/GaelleBriet/nodejs-k8s-app

Première question :
Consultez le rapport de couverture dans coverage/lcov-report/index.html.
Quel est le taux de couverture ?

Réponse : Le taux de couverture est de 93.54%.

```
<div class='fl pad1y space-right2'>
    <span class="strong">93.54% </span>
    <span class="quiet">Lines</span>
    <span class='fraction'>29/31</span>
</div>
```

Seconde question :
Allez dans l'onglet "Actions" de votre repo.
Les tests passent-ils tous ?

Ils sont tous passés.

Exercice 1 : Observer le comportement des pods

Questions :

- Que se passe-t-il quand vous supprimez un pod ?
- Combien de temps faut-il pour que le nouveau pod soit prêt ?
- Quelle est la différence entre les états Running et Ready ?

Réponses :

- Kubernetes recréé automatiquement un pod pour maintenir le nombre de réplicas : `minReplicas: 2` dans hpa.yaml

  Séquence observée :
  Pod csvcz passe en état Terminating (5m20s)
  Immédiatement, un nouveau pod vmsft est créé (Pending → ContainerCreating → Running → Ready)
  L'ancien pod passe par Completed puis est supprimé définitivement
  Le ReplicaSet (géré par le Deployment) garantit qu'il y a toujours le bon nombre de pods actifs.

- 8 secondes ont été nécessaires.

```
⚡kubectl get pods -n nodejs-app -w
NAME                          READY   STATUS    RESTARTS   AGE
nodejs-app-8566cbbcb6-csvcz   1/1     Running   0          3m56s
nodejs-app-8566cbbcb6-jh9nv   1/1     Running   0          3m56s
nodejs-app-8566cbbcb6-vwfsk   1/1     Running   0          3m56s
nodejs-app-8566cbbcb6-csvcz   1/1     Terminating   0          5m20s
nodejs-app-8566cbbcb6-vmsft   0/1     Pending       0          0s
nodejs-app-8566cbbcb6-csvcz   1/1     Terminating   0          5m20s
nodejs-app-8566cbbcb6-vmsft   0/1     Pending       0          0s
nodejs-app-8566cbbcb6-vmsft   0/1     ContainerCreating   0          0s
nodejs-app-8566cbbcb6-vmsft   0/1     Running             0          2s
nodejs-app-8566cbbcb6-csvcz   0/1     Completed           0          5m25s
nodejs-app-8566cbbcb6-csvcz   0/1     Completed           0          5m26s
nodejs-app-8566cbbcb6-csvcz   0/1     Completed           0          5m26s
nodejs-app-8566cbbcb6-vmsft   1/1     Running             0          8s
nodejs-app-8566cbbcb6-vmsft   1/1     Terminating         0          43s
nodejs-app-8566cbbcb6-vmsft   1/1     Terminating         0          43s
nodejs-app-8566cbbcb6-vmsft   0/1     Completed           0          49s
nodejs-app-8566cbbcb6-vmsft   0/1     Completed           0          49s
nodejs-app-8566cbbcb6-vmsft   0/1     Completed           0          49s
```

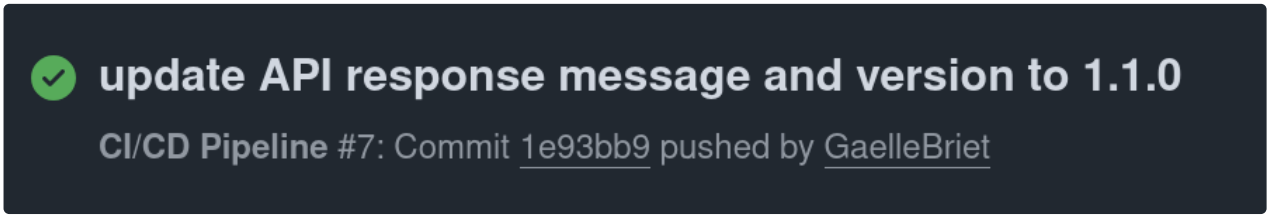
- **Running** : Le container tourne, mais l'application n'est pas encore prête à recevoir du trafic
- **Ready** : L'application a passé toutes les readiness probes et peut recevoir du trafic

---

Exercice 2 : Tester le Rolling Update

Questions :

- Observez-vous des erreurs pendant le déploiement ?
- Les deux versions coexistent-elles pendant un moment ?
- pas d'erreurs




```
nodejs-k8s-app on  main
⚡kubectl rollout status deployment/nodejs-app -n nodejs-app
kubectl rollout history deployment/nodejs-app -n nodejs-app
deployment "nodejs-app" successfully rolled out
deployment.apps/nodejs-app
REVISION  CHANGE-CAUSE
1         <none>
```

```
nodejs-k8s-app on  main
⚡kubectl rollout restart deployment/nodejs-app -n nodejs-app

deployment.apps/nodejs-app restarted
```

- oui le temps que les nouveaux pods avec la nouvelle image soient prêts

---

Exercice 3 : Tester l'autoscaling

Questions :

- Combien de temps faut-il pour que les pods se créent ?
- Quel est le nombre maximum de pods créés ?
- Que se passe-t-il après l'arrêt du test de charge ?
- environ 8 secondes

```
nodejs-app-7c6d8578d7-vrsvd   0/1     ContainerCreating   0              0s
nodejs-app-7c6d8578d7-vrsvd   0/1     Running             0              2s
nodejs-app-7c6d8578d7-vrsvd   1/1     Running             0              8s
```

- 3 pods max

```
⚡kubectl get hpa -n nodejs-app
NAME              REFERENCE                TARGETS                        MINPODS   MAXPODS   REPLICAS   AGE
nodejs-app-hpa    Deployment/nodejs-app    cpu: 1%/70%, memory: 10%/80%   2         10        3          102m
```

- Le HPA descale vers le nombre minimum de réplicas, les pods excédentaires passent en état `Terminating` puis disparaissent, l'utilisation CPU diminue progressivement

```
⚡kubectl get hpa -n nodejs-app -w
NAME              REFERENCE                TARGETS                        MINPODS   MAXPODS   REPLICAS   AGE
nodejs-app-hpa    Deployment/nodejs-app    cpu: 1%/70%, memory: 10%/80%   2         10        3          103m
nodejs-app-hpa    Deployment/nodejs-app    cpu: 46%/70%, memory: 14%/80%  2         10        3          104m
nodejs-app-hpa    Deployment/nodejs-app    cpu: 46%/70%, memory: 14%/80%  2         10        3          105m
nodejs-app-hpa    Deployment/nodejs-app    cpu: 34%/70%, memory: 10%/80%  2         10        2          105m
nodejs-app-hpa    Deployment/nodejs-app    cpu: 1%/70%, memory: 10%/80%   2         10        2          106m
nodejs-app-hpa    Deployment/nodejs-app    cpu: 1%/70%, memory: 10%/80%   2         10        2          107m
```

```
⚡kubectl get pods -n nodejs-app -w
NAME                              READY   STATUS    RESTARTS        AGE
nodejs-app-7c6d8578d7-tvvrv   1/1     Running   0               16m
nodejs-app-7c6d8578d7-vjxkx   1/1     Running   1 (9m34s ago)   16m
nodejs-app-7c6d8578d7-vrsvd   1/1     Running   0               3m41s
nodejs-app-7c6d8578d7-tvvrv   1/1     Terminating   0               18m
nodejs-app-7c6d8578d7-tvvrv   1/1     Terminating   0               18m
nodejs-app-7c6d8578d7-tvvrv   0/1     Completed     0               18m
nodejs-app-7c6d8578d7-tvvrv   0/1     Completed     0               18m
nodejs-app-7c6d8578d7-tvvrv   0/1     Completed     0               18m
```

Exercice 4 : Ajouter une nouvelle fonctionnalité

```
nodejs-k8s-app on  main
⚡kubectl set image deployment/nodejs-app \
  nodejs-app=yuki82/nodejs-k8s-app:latest \
  -n nodejs-app

nodejs-k8s-app on  main
⚡curl http://localhost:8080/api/stats | jq
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
curl: (52) Empty reply from server

nodejs-k8s-app on  main
⚡kubectl get pods -n nodejs-app

NAME                          READY   STATUS    RESTARTS   AGE
nodejs-app-79dd65b868-647lg   1/1     Running   0          22s
nodejs-app-79dd65b868-xtkw2   1/1     Running   0          22s
```

Questions de validation

Questions théoriques / Réponses
Docker :

1. **Expliquez l'avantage du multi-stage build**
   - Réduire la taille de l'image (stage 1 avec les outils de build, stage 2 avec uniquement les dépendances de production)
   - Séparation des responsabilités
2. **Pourquoi utilise-t-on un utilisateur non-root ?**
   - principe du moindre privilège
   - limitation des dégâts si l'app est attaquée (l'attaquant n'aura pas les accès root)
   - bonne pratiques de scanners de sécurité
3. **À quoi sert le health check dans le Dockerfile ?**
   - surveillance de la santé pour docker / kubernetes
   - redémarrage automatique
   - load balancing (traffic envoyé uniquement sur les conteneurs healthy)

Tests :

1. **Quelle est la différence entre un test unitaire et un test d'intégration ?**
   - Test unitaire : teste une fonction ou un composant isolé, dépendances externes mockées, tests rapides, permet de vérifier la logique métier
   - Test d'intégration : teste plusieurs composants ensemble, dépendances réelles utilisées, tests plus lents.
2. **Pourquoi viser 80% de code coverage ?**
   - Viser 80% de code coverage permet de s'assurer d'une bonne qualité de code, de détecter rapidement le régressions, seuil communément recommandé.
3. **Comment mocker les dépendances externes en Jest ?**
   - On peut utiliser `jest.mock()`

CI/CD :

1. **Expliquez le workflow GitHub Actions du projet**
   Déclenché à chaque push / PR

```
on:
push:
branches:
  - main
  - develop
pull_request:
branches:
  - main
```

- **1 : Lint** (vérifier la qualité du code), déclenché a chaque push/PR
- **2 : Tests,** attends que le lint passe, strategy matrix (tests en parallère sur node 20 et 22)
- **3 : build-and-push**, construction de l'image docker, attends que les tests soient passés, et uniquement sur les push sur main. Push vers DockerHub
- **4 : security-scan,** analyse de sécurité, attends que le build-and-push soit OK, uniquement après un push sur main. Détecte les vulnérabilités dans l'image
- **5 : Notify**, affiche un message de succès si le build est réussi, affiche une erreur et fait échouer le workflow si le build échoue

2. **Pourquoi tester sur plusieurs versions de Node.js ?**
   - Tester sur plusieurs versions de node pour assurer la compatibilité entre les versions

3. **Qu'est-ce que le cache dans les GitHub Actions ?**

- le cache permet de sauvegarder et réutiliser des fichiers entre les éxécutions de workflow pour accélérer les builds

Kubernetes :

1. **Différence entre Deployment et StatefulSet ?**
   Deployment : pour une application stateless (sans état persistant)
   StatefulSet : pour une application stateful avec un état persistant (comme une base de données)
2. **Expliquez livenessProbe vs readinessProbe**
   livenessProbe : sert à détecter si l'app est bloquée ou morte, si en échec kubernetes redémarre le pod
   readinessProbe : sert à détecter si l'app peut recevoir du traffic, si en échec le pod est retiré du service load balancer
3. **Comment fonctionne le HPA ?**
   Le HorizontalPodAutoscaler scale automatiquement le nombre de pods selon des métriques.
4. **Qu'est-ce qu'un Rolling Update ?**
   Stratégie de déploiement par défaut qui met à jour les pods progressivement sans downtime

Architecture :

1. Pourquoi utiliser un Service de type NodePort ?
   - expose sur un port statique , évite le recours à un load balancer, pour les environnements de dev / test
2. À quoi sert un Ingress ?
   - objet lubernetes qui gère l'accès HTTP/HTTPS
3. Comment gérer les secrets sensibles ?
   - les chiffrer et stocker dans git

Questions pratiques

- Montrez comment rollback vers une version précédente

```
# Afficher l'historique complet
kubectl rollout history deployment/nodejs-app -n nodejs-app

# Voir les détails d'une révision spécifique
kubectl rollout history deployment/nodejs-app --revision=2 -n nodejs-app

# Rollback vers la révision précédente (n-1)
kubectl rollout undo deployment/nodejs-app -n nodejs-app

# Rollback vers une révision spécifique
kubectl rollout undo deployment/nodejs-app --to-revision=3 -n nodejs-app
```

- Augmentez les replicas à 5 manuellement

```
# Scale à 5 replicas
kubectl scale deployment/nodejs-app --replicas=5 -n nodejs-app

# Vérifier
kubectl get deployment nodejs-app -n nodejs-app
kubectl get pods -n nodejs-app -w  # -w pour watch en temps réel


# OU ----

# Éditer k8s/deployment.yaml
# Changer replicas: 2 → replicas: 5
```

```
# Appliquer
kubectl apply -f k8s/deployment.yaml
```

- Ajoutez une variable d'environnement au déploiement

```
# Ajouter une variable
kubectl set env deployment/nodejs-app NEW_VAR=production -n nodejs-app

# Vérifier
kubectl describe deployment nodejs-app -n nodejs-app | grep -A 10 Environment

# OU Modifier deployment.yaml
# puis l'appliquer
kubectl apply -f k8s/deployment.yaml

# Vérifier dans un pod
kubectl exec -it <pod-name> -n nodejs-app -- env | grep NEW_VAR
```

- Créez un pod de debug dans le namespace

```
# Pod avec shell bash
kubectl run debug-pod \
  --image=busybox \
  --restart=Never \
  --namespace=nodejs-app \
  --rm -it \
  -- /bin/sh

# wget -O- http://nodejs-app-service:80/
# nslookup nodejs-app-service
# exit
```

- Exportez tous les manifestes d'un namespace

```
# Exporter toutes les ressources
kubectl get all -n nodejs-app -o yaml > nodejs-app-backup.yaml

# Plus sélectif
kubectl get deployment,service,configmap,secret -n nodejs-app -o yaml > backup.yaml
```