

Guide d'Administration

Namespaces Moodle - Cluster RBER

Version 1.0 - Janvier 2026

Sommaire

1. Configuration kubectl
2. Prérequis et accès cluster
3. Gestion des namespaces
4. Déploiement d'applications
5. Configuration DNS
6. Certificats SSL
7. Ingress et routage
8. Stockage persistant
9. Base de données PostgreSQL
10. Monitoring et logs
11. Backup et restauration
12. Troubleshooting

1. Configuration kubectl

1.1 Réception du fichier d'accès

L'équipe DevOps fournit un fichier kubeconfig pour accéder au cluster :

- username.kubeconfig.gpg — fichier chiffré GPG
- username.kubeconfig — fichier non chiffré

1.2 Installation kubectl

Ubuntu/Debian

```
curl -LO "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
chmod +x kubectl
sudo mv kubectl /usr/local/bin/
```

macOS

```
brew install kubectl
```

Windows (PowerShell admin)

```
choco install kubernetes-cli
```

1.3 Déchiffrement GPG

Si le fichier est chiffré (.gpg), déchiffrer avec ta clé privée :

```
gpg --decrypt username.kubeconfig.gpg > ~/.kube/config-rber
chmod 600 ~/.kube/config-rber
```

1.4 Configuration de l'environnement

Option A : Fichier unique

```
# Créer le dossier
mkdir -p ~/.kube

# Placer le fichier
mv username.kubeconfig ~/.kube/config-rber

# Définir la variable
export KUBECONFIG=~/.kube/config-rber

# Rendre permanent
echo 'export KUBECONFIG=~/.kube/config-rber' >> ~/.bashrc
source ~/.bashrc
```

Option B : Fichiers multiples (plusieurs clusters)

```
# Fusionner avec config existante
export KUBECONFIG=~/.kube/config:~/.kube/config-rber

# Lister les contextes
kubectl config get-contexts

# Basculer vers RBER
kubectl config use-context username@rber-fidj
```

1.5 Vérification de la connexion

```
# Tester la connexion
kubectl get pods
```

```
# Vérifier les droits
kubectl auth can-i --list

# Contexte actuel
kubectl config current-context

# Infos cluster
kubectl cluster-info
```

1.6 Commandes kubectl essentielles

Action	Commande
Lister les pods	<code>kubectl get pods</code>
Logs d'un pod	<code>kubectl logs <pod></code>
Logs temps réel	<code>kubectl logs -f <pod></code>
Entrer dans un pod	<code>kubectl exec -it <pod> -- bash</code>
Décrire une ressource	<code>kubectl describe pod <pod></code>
Port-forward	<code>kubectl port-forward svc/<svc> 8080:80</code>
Lister les services	<code>kubectl get svc</code>
Lister les deployments	<code>kubectl get deployments</code>
Lister les ingress	<code>kubectl get ingress</code>
Lister les PVC	<code>kubectl get pvc</code>

1.7 Alias recommandés

Ajouter au `.bashrc` ou `.zshrc` :

```
alias k='kubectl'
alias kgp='kubectl get pods'
alias kgs='kubectl get svc'
alias kgd='kubectl get deployments'
alias kgi='kubectl get ingress'
alias kl='kubectl logs'
alias klf='kubectl logs -f'
alias ke='kubectl exec -it'

# Autocomplétion
source <(kubectl completion bash) # bash
source <(kubectl completion zsh)  # zsh
```

1.8 Résolution de problèmes kubectl

Connexion refusée

```
# Vérifier l'URL
grep server ~/.kube/config-rber

# Tester la connectivité
curl -k https://102.222.216.6:6443/healthz
```

Erreur certificat

```
# Vérifier le CA (doit retourner > 1000)
grep certificate-authority-data ~/.kube/config-rber | wc -c
```

Erreur permissions

```
kubectl auth can-i get pods
kubectl auth can-i create deployments
kubectl auth can-i --list
```

1.9 Résumé rapide

```
# 1. Déchiffrer (si .gpg)
gpg --decrypt fichier.kubeconfig.gpg > ~/.kube/config-rber

# 2. Configurer
export KUBECONFIG=~/.kube/config-rber

# 3. Tester
kubectl get pods
```

2. Prérequis et accès cluster

2.1 Informations cluster

Paramètre	Valeur
Cluster	rber-fidj (RKE2 v1.32.x)
API Server externe	https://102.222.216.6:6443
API Server interne	https://10.29.112.100:6443
HAProxy VIP	10.29.112.100 / 102.222.216.6
Registry Harbor	harbor.rber.bj
Storage Class	longhorn

2.2 Namespaces par université

Université	Namespace	Domaine
UNSTIM	moodle-unstim	elearning.unstim.bj
UAC	moodle-uac	elearning.uac.bj
UNA	moodle-una	elearning.una.bj
UP	moodle-up	elearning.up.bj

2.3 Vérification rapide du cluster

```
# État des nodes
kubectl get nodes -o wide

# Ressources disponibles
kubectl top nodes

# Namespaces existants
kubectl get namespaces
```

3. Gestion des namespaces

3.1 Créer un namespace

```
# Créer le namespace
kubectl create namespace moodle-uac

# Ajouter les labels
kubectl label namespace moodle-uac \
  university=UAC \
  app=moodle \
  environment=production
```

3.2 Quotas de ressources

```
# resource-quota.yaml
apiVersion: v1
kind: ResourceQuota
metadata:
  name: moodle-quota
  namespace: moodle-uac
spec:
  hard:
    requests.cpu: "4"
    requests.memory: 8Gi
    limits.cpu: "8"
    limits.memory: 16Gi
    persistentvolumeclaims: "10"
    pods: "20"
kubectl apply -f resource-quota.yaml
```

3.3 Network Policies

```
# network-policy.yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: default-deny-ingress
  namespace: moodle-uac
spec:
  podSelector: {}
  policyTypes:
  - Ingress
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          name: ingress-haproxy
    - podSelector: {}
```

4. Déploiement d'applications

4.1 Déployer Moodle avec Helm

```
# Ajouter le repo Bitnami
helm repo add bitnami https://charts.bitnami.com/bitnami
helm repo update
```

Créer le fichier values :

```
# moodle-uac-values.yaml
image:
  tag: 5.0.0-debian-12-r0

moodleSkipInstall: false
moodleUsername: admin
moodleEmail: admin@uac.bj
moodleSiteName: "E-Learning UAC"

postgresql:
  enabled: false

externalDatabase:
  host: uac-pg-cluster-rw
  port: 5432
  user: moodleuser
  database: moodle
  type: postgresql

persistence:
  enabled: true
  existingClaim: moodle-uac-data

resources:
  requests:
    memory: 2Gi
    cpu: 1000m
  limits:
    memory: 4Gi
    cpu: 2000m

ingress:
  enabled: true
  ingressClassName: haproxy
  hostname: elearning.uac.bj
```

Installer :

```
helm install moodle-uac bitnami/moodle \
  --namespace moodle-uac \
  --values moodle-uac-values.yaml \
  --set externalDatabase.password=MOT_DE_PASSE \
  --set moodlePassword=ADMIN_PASSWORD
```

4.2 Mise à jour et rollback

```
# Upgrade
helm upgrade moodle-uac bitnami/moodle \
  --namespace moodle-uac \
  --values moodle-uac-values.yaml
```

```
# Historique
helm history moodle-uac -n moodle-uac

# Rollback
helm rollback moodle-uac 1 -n moodle-uac
```


5. Configuration DNS

5.1 Enregistrements DNS requis

Domaine	Type	Valeur
elearning.unstim.bj	A	102.222.216.6
elearning.uac.bj	A	102.222.216.6
elearning.una.bj	A	102.222.216.6
elearning.up.bj	A	102.222.216.6

5.2 Vérification DNS

```
# Résolution
nslookup elearning.uac.bj
dig elearning.uac.bj +short

# Test connectivité
curl -I http://elearning.uac.bj
```

5.3 Exemple zone DNS (Bind)

```
; Zone uac.bj
$TTL 3600
@      IN  SOA  ns1.uac.bj. admin.uac.bj. (
        2026010201  ; Serial
        3600        ; Refresh
        1800        ; Retry
        604800      ; Expire
        86400 )     ; Minimum TTL

@      IN  NS   ns1.uac.bj.
ns1     IN  A    IP_SERVEUR_DNS
elearning IN A    102.222.216.6
```

6. Certificats SSL

6.1 Installation cert-manager

```
helm repo add jetstack https://charts.jetstack.io
helm repo update

helm install cert-manager jetstack/cert-manager \
  --namespace cert-manager \
  --create-namespace \
  --set installCRDs=true
```

6.2 ClusterIssuer Let's Encrypt

```
# cluster-issuer.yaml
apiVersion: cert-manager.io/v1
kind: ClusterIssuer
metadata:
  name: letsencrypt-prod
spec:
  acme:
    server: https://acme-v02.api.letsencrypt.org/directory
    email: admin@rber.bj
    privateKeySecretRef:
      name: letsencrypt-prod-key
    solvers:
      - http01:
          ingress:
            class: haproxy
kubectl apply -f cluster-issuer.yaml
```

6.3 Demander un certificat

```
# certificate-uac.yaml
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: elearning-uac-tls
  namespace: moodle-uac
spec:
  secretName: elearning-uac-tls
  issuerRef:
    name: letsencrypt-prod
    kind: ClusterIssuer
  dnsNames:
    - elearning.uac.bj
kubectl apply -f certificate-uac.yaml

# Vérifier
kubectl get certificate -n moodle-uac
kubectl describe certificate elearning-uac-tls -n moodle-uac
```

6.4 Certificat manuel

```
kubectl create secret tls elearning-uac-tls \
  --cert=elearning.uac.bj.crt \
  --key=elearning.uac.bj.key \
  -n moodle-uac
```

6.5 Vérification SSL

```
# Vérifier le certificat
openssl s_client -connect elearning.uac.bj:443 -servername elearning.uac.bj

# Expiration
echo | openssl s_client -connect elearning.uac.bj:443 2>/dev/null | \
  openssl x509 -noout -dates
```

7. Ingress et routage

7.1 Ingress HTTP

```
# ingress-uac.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: moodle-uac-ingress
  namespace: moodle-uac
  annotations:
    haproxy.org/timeout-client: "300s"
    haproxy.org/timeout-server: "300s"
spec:
  ingressClassName: haproxy
  rules:
  - host: elearning.uac.bj
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: moodle-uac
            port:
              number: 80
```

7.2 Ingress HTTPS avec TLS

```
# ingress-uac-tls.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: moodle-uac-ingress
  namespace: moodle-uac
  annotations:
    haproxy.org/ssl-redirect: "true"
    haproxy.org/timeout-client: "300s"
    haproxy.org/timeout-server: "300s"
    cert-manager.io/cluster-issuer: letsencrypt-prod
spec:
  ingressClassName: haproxy
  tls:
  - hosts:
    - elearning.uac.bj
      secretName: elearning-uac-tls
  rules:
  - host: elearning.uac.bj
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: moodle-uac
            port:
              number: 80
```

7.3 Vérifier les Ingress

```
kubect1 get ingress -A  
kubect1 describe ingress moodle-uac-ingress -n moodle-uac
```

8. Stockage persistant

8.1 Créer un PVC

```
# pvc-moodle-uac.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: moodle-uac-data
  namespace: moodle-uac
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: longhorn
  resources:
    requests:
      storage: 100Gi
kubectl apply -f pvc-moodle-uac.yaml
kubectl get pvc -n moodle-uac
```

8.2 Redimensionner un PVC

```
kubectl patch pvc moodle-uac-data -n moodle-uac \
-p '{"spec": {"resources": {"requests": {"storage": "150Gi"}}}}'
```

8.3 État Longhorn

```
kubectl get volumes.longhorn.io -n longhorn-system
kubectl get replicas.longhorn.io -n longhorn-system
```

9. Base de données PostgreSQL

9.1 Créer un cluster CloudNativePG

```
# pg-cluster-uac.yaml
apiVersion: postgresql.cnpg.io/v1
kind: Cluster
metadata:
  name: uac-pg-cluster
  namespace: moodle-uac
spec:
  instances: 3
  imageName: ghcr.io/cloudnative-pg/postgresql:16.3
  storage:
    size: 50Gi
    storageClass: longhorn
  bootstrap:
    initdb:
      database: moodle
      owner: moodleuser
      encoding: UTF8
  postgresql:
    parameters:
      max_connections: "200"
      shared_buffers: "2GB"
  monitoring:
    enablePodMonitor: true
kubectl apply -f pg-cluster-uac.yaml
kubectl get cluster -n moodle-uac
```

9.2 Récupérer les credentials

```
# Mot de passe
kubectl get secret uac-pg-cluster-app -n moodle-uac \
  -o jsonpath='{.data.password}' | base64 -d

# Connection string
kubectl get secret uac-pg-cluster-app -n moodle-uac \
  -o jsonpath='{.data.uri}' | base64 -d
```

9.3 Connexion psql

```
kubectl port-forward -n moodle-uac svc/uac-pg-cluster-rw 5432:5432 &

PGPASSWORD=$(kubectl get secret uac-pg-cluster-app -n moodle-uac \
  -o jsonpath='{.data.password}' | base64 -d) \
psql -h localhost -U moodleuser -d moodle
```

9.4 Failover manuel

```
kubectl cnpg promote uac-pg-cluster uac-pg-cluster-2 -n moodle-uac
```

10. Monitoring et logs

10.1 Logs des pods

```
# Logs d'un deployment
kubectl logs -n moodle-uac deployment/moodle-uac --tail=100

# Temps réel
kubectl logs -n moodle-uac deployment/moodle-uac -f

# Tous les pods d'un label
kubectl logs -n moodle-uac -l app=moodle --all-containers
```

10.2 Métriques

```
kubectl top pods -n moodle-uac
kubectl top nodes
```

10.3 Events

```
kubectl get events -n moodle-uac --sort-by='.lastTimestamp'
kubectl get events -n moodle-uac -w
```


11. Backup et restauration

11.1 Backup PostgreSQL

```
# Backup CloudNativePG
kubectl cnpg backup uac-pg-cluster -n moodle-uac
kubectl get backups -n moodle-uac
```

11.2 Backup manuel pg_dump

```
kubectl port-forward -n moodle-uac svc/uac-pg-cluster-rw 5432:5432 &

PGPASSWORD=$(kubectl get secret uac-pg-cluster-app -n moodle-uac \
  -o jsonpath='{.data.password}' | base64 -d) \
pg_dump -h localhost -U moodleuser -d moodle | gzip > backup_$(date +%Y%m%d).sql.gz
```

11.3 Restauration

```
gunzip -c backup_20260101.sql.gz | \
PGPASSWORD=xxx psql -h localhost -U moodleuser -d moodle
```

12. Troubleshooting

12.1 Pod en CrashLoopBackOff

```
kubect1 logs -n moodle-uac <pod> --previous
kubect1 describe pod -n moodle-uac <pod>
kubect1 get events -n moodle-uac --field-selector involvedObject.name=<pod>
```

12.2 Pod en Pending

```
kubect1 describe nodes | grep -A 5 "Allocated resources"
kubect1 get pvc -n moodle-uac
```

12.3 Problème connexion DB

```
kubect1 exec -it -n moodle-uac deployment/moodle-uac -- bash

# Dans le pod
apt update && apt install -y postgresql-client
psql -h uac-pg-cluster-rw -U moodleuser -d moodle -c '\l'
```

12.4 Problème Ingress/SSL

```
kubect1 describe ingress -n moodle-uac
kubect1 get certificate -n moodle-uac
kubect1 logs -n cert-manager deployment/cert-manager
```

12.5 Redémarrer un déploiement

```
kubect1 rollout restart deployment/moodle-uac -n moodle-uac
kubect1 rollout status deployment/moodle-uac -n moodle-uac
```