Coaching Kubernetes – Semaine 1 CKAD

Préparation intensive à la certification CKAD avec le Coach Expert Brascens Sedogbo. Une semaine complète dédiée à la maîtrise des fondamentaux pratiques du développement Kubernetes.

Coach Expert CKAD : Brascens Sedogbo

Préparation à la certification CKAD

Date	Durée	Format
1 décembre 2025	1 semaine (Lundi → Vendredi)	2h/jour présentiel + autonomie

Objectif général

Maîtriser les fondamentaux pratiques du développement Kubernetes : créer, debugger et optimiser des applications dans un cluster. Préparer l'examen CKAD 100% pratique (2h, 15-20 tâches, kubectl impératif).

Compétences Semaine 1

Pods/Deployments/Services + ConfigMaps/Secrets + Probes + Multi-containers.

Livrable obligatoire jeudi

CKAD_Semaine1.md unique contenant :

- Tous les TP (commandes + YAML templates + captures)
- 1 CrashLoopBackOff debuggé
- Challenge technique réussi
- Templates YAML réutilisables

Organisation complète

Jour	Format	Contenu	Livrable
Lundi	Présentiel 2h	Pods + Deployments + Services	TP1 dans MD
Mardi	Autonomie	Refaire impératif + debug	TP2 dans MD
Mercredi	Présentiel 2h	ConfigMaps/Secrets/Probes/Multi-containers	TP3-5 dans MD
Jeudi	Présentiel 2h	Démo + Challenge 1h	TP6 + MD final
Vendredi	Autonomie	Consolidation + exercices	MD finalisé

Prérequis (À faire AVANT lundi)
Cluster Kubernetes fonctionnel

Minikube standalone (pour CKAD rapide)

1. Installation Minikube (5 min)

```
# Docker
sudo apt install -y docker.io && sudo usermod -aG docker $USER && newgrp docker

Basculer iptables en mode legacy
sudo update-alternatives --set iptables /usr/sbin/iptables-legacy
sudo update-alternatives --set ip6tables /usr/sbin/ip6tables-legacy
sudo service docker restart

# Minikube + kubectl
curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
sudo install minikube-linux-amd64 /usr/local/bin/minikube
curl -LO "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
sudo install kubectl /usr/local/bin/kubectl

# Start
minikube start --driver=docker --cpus=2 --memory=4096
kubectl get nodes # Ready?
```

3. Alias CKAD (essentiels)

```
echo 'alias k="kubectl"' >> ~/.bashrc
echo 'alias kk="kubectl config set-context --current --namespace"' >> ~/.bashrc
echo 'alias km="minikube"' >> ~/.bashrc
source ~/.bashrc
k version --client
```

✅ **Checklist :** minikube start OK + k run nginx --image=nginx --restart=Never → Pod Running.

LUNDI – Fondamentaux + Pods + Deployments + Services (2h)

Kubernetes, Quoi ? Pourquoi ? Quand ? Où ? Comment démarrer ? (Vue d'ensemble | Kubernetes) (15min)

1. Philosophie CKAD (10 min)

Examen : 2h, 15-20 tâches pratiques, docs kubernetes.io autorisées

Priorités : kubectl impératif (k), YAML editing rapide (vim), debug (describe/logs/exec)

Semaine 1 = 70% du syllabus CKAD (Core Concepts + Configuration + Observability)

1. Pods impératifs (20 min)

```
k run nginx --image=nginx --restart=Never # Pod simple
k get pods -o wide
k describe pod nginx # Événements + status
k logs nginx # Logs
k delete pod nginx
```

2. Deployments (30 min)

```
k create deployment web --image=nginx
k scale deployment web --replicas=3
k set image deployment/web nginx=nginx:1.25
k rollout status deployment/web
k rollout history deployment/web
k rollout undo deployment/web # Revert
k get deployments -o wide
```

3. Services (30 min)

```
k expose deployment web --type=NodePort --port=80
k expose deployment web --type=ClusterIP --port=80 --name=web-internal
k get svc # Notez NodePort
k describe svc web
km service web --url # Testez l'URL
```

Livrable : Section "TP1 – Pods/Deployments/Services" dans CKAD_Semaine1.md.

MARDI – Autonomie impérative

Objectif : Refaire Lundi UNIQUEMENT impératif, sans copier-coller.

TP obligatoire :

> Pod api (nginx) → Deployment front (3→5 replicas) → mauvaise image (CrashLoop) → corriger → NodePort

Debug obligatoire : k get pods, describe, logs, events.

Ressources autorisées :

- CKAD-Exercises (**https://github.com/dgkanatsios/CKAD-exercises**)
- Kubernetes Docs (**https://kubernetes.io/docs/concepts/**)
- Vidéo : "Kubernetes Deployments" Nana (20 min max)

Livrable : Section "TP2 – Autonomie impérative" + 1 CrashLoop debuggé.

MERCREDI – ConfigMaps + Secrets + Probes + Multi-containers (2h)

1. ConfigMaps/Secrets (30 min)

```
k create configmap app-config --from-literal=ENV=prod -o yaml --dry-run=client > configmap.yaml
k create secret generic db-secret --from-literal=password=admin123 -o yaml --dry-run=client > secret.yaml
k apply -f configmap.yaml -f secret.yaml
k get configmap,secret
```

pod-config.yaml (créez-le) :

```
apiVersion: v1
kind: Pod
metadata:
  name: app-pod
spec:
  containers:
  - name: app
    image: nginx
    env:
    - name: ENV
      valueFrom:
        configMapKeyRef:
          name: app-config
          key: ENV
    - name: DB_PASSWORD
      valueFrom:
        secretKeyRef:
          name: db-secret
          key: password
```

```
k apply -f pod-config.yaml && k get pod -o yaml | grep -A5 ENV
```

MERCREDI – ConfigMaps + Secrets + Probes + Multi-containers (2h)

2. Probes + Resources (30 min)

deployment-probes.yaml :

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: probed-app
spec:
  replicas: 2
  selector:
    matchLabels:
      app: probed-app
  template:
    metadata:
      labels:
        app: probed-app
    spec:
      containers:
      - name: nginx
        image: nginx
        resources:
          requests:
            cpu: 100m
            memory: 128Mi
          limits:
            cpu: 200m
            memory: 256Mi
        livenessProbe:
          httpGet:
            path: /
            port: 80
          initialDelaySeconds: 5
          periodSeconds: 10
        readinessProbe:
          httpGet:
            path: /
            port: 80
          initialDelaySeconds: 3
```

```
k apply -f deployment-probes.yaml && k describe pod
```

MERCREDI – ConfigMaps + Secrets + Probes + Multi-containers (2h)

3. Multi-containers Sidecar (30 min)

pod-sidecar.yaml :

```
apiVersion: v1
kind: Pod
metadata:
  name: sidecar-pod
spec:
  containers:
  - name: app
    image: nginx
  - name: logger
    image: busybox
    command: ['sh', '-c', 'while true; do echo "$(date) :: log" >> /var/log/app.log; sleep 5; done']
    volumeMounts:
    - name: varlog
      mountPath: /var/log
  volumes:
  - name: varlog
    emptyDir: {}
```

```
k apply -f pod-sidecar.yaml && k logs sidecar-pod -c logger
```

Livrable : Sections "TP3 Config/Secrets", "TP4 Probes", "TP5 Multi-containers".

JEUDI – Challenge + Validation (2h)

1. Démo obligatoire (30 min)

Prouvez (écran partagé) les éléments suivants pour valider votre maîtrise des concepts fondamentaux :

- [ ] ✔️ Pods/Deployments/Services fonctionnels
- [ ] ✔️ ConfigMap/Secret injectés (vérifiable avec `k get pod -o yaml | grep ENV`)
- [ ] ✔️ Probes actifs (visualisables avec `k describe pod` → LastProbe)
- [ ] ✔️ Logs Sidecar fonctionnels et visibles

Cette démo confirmera votre capacité à déployer, configurer et monitorer des applications de base dans Kubernetes.

2. Challenge technique CKAD (30min, SEUL, sans aide)

Objectif : Dans un namespace dédié `ckad-s1`, déployez une application complète comprenant un Deployment `api` avec 3 réplicas, incluant des Probes et des limites de ressources. Exposez l'application via un Service ClusterIP. Ajoutez un Pod multi-container qui intègre un sidecar. Introduisez volontairement un `CrashLoopBackOff`, puis débuggez et corrigez le problème.

Étapes initiales de mise en place :

```
k create ns ckad-s1
k create deployment api --image=nginx -n ckad-s1
k scale deployment api --replicas=3 -n ckad-s1
k expose deployment api --port=80 --type=ClusterIP -n ckad-s1
```

Réalisez ensuite les actions suivantes :

1. **Édition YAML :** Modifiez le Deployment `api` (`k edit deployment api -n ckad-s1`) pour y ajouter des **probes** (liveness et readiness) et des **limites de ressources** (CPU, mémoire), en vous basant sur les templates vus mercredi.
2. **Provoquer un crash :** Simulez un problème en changeant l'image du container `nginx` pour une image inexistante ou incorrecte : `k set image deployment/api nginx=mauvaise-image -n ckad-s1`. + "CHALLENGE SURPRISE"
3. **Débogage :** Utilisez les commandes de débogage essentielles (`k describe pod -n ckad-s1`, `k logs -n ckad-s1`, événements) pour identifier la cause du `CrashLoopBackOff`.
4. **Correction :** Appliquez la bonne image pour corriger le problème : `k set image deployment/api nginx=nginx:latest -n ckad-s1`.

Critères de réussite : Le challenge doit être complété en moins d'une demi heure, et vous devrez être capable d'expliquer oralement votre démarche de débogage et de correction.

VENDREDI – Consolidation CKA-ready

Finaliser CKAD_Semaine1.md

Runbook + tous YAML templates

5 exercices CKAD-Exercises

Pods/Deployments (https://github.com/dgkanatsios/CKAD-exercises)

Vim CKAD

echo "set nu ts=2 et sw=2" >> ~/.vimrc

Vidéo

"CKAD Debugging" TechWorld with Nana (20 min)

🏁 Validation Finale Semaine 1

Le participant maîtrise :

- ✔️ [ ] kubectl impératif + debug CrashLoopBackOff
- ✔️ [ ] Pods/Deployments/Services + scaling/rollouts
- ✔️ [ ] ConfigMaps/Secrets injection
- ✔️ [ ] Probes (liveness/readiness) + resources
- ✔️ [ ] Multi-containers sidecar pattern
- ✔️ [ ] CKAD_Semaine1.md pro (templates réutilisables)

Ressources Semaine 1 :

- CKAD-Exercises (**https://github.com/dgkanatsios/CKAD-exercises**)
- Kubernetes.io Concepts (**Concepts | Kubernetes**)
- Linux Foundation - Introduction to Kubernetes (LFS158) - edX gratuit (15h)
- Xavki - Kubernetes - Playlist française complète
- KillerCoda - Labs gratuits dans le navigateur

Prochaine étape : Semaine 2 CKAD Jobs/CronJobs + NetworkPolicies + Affinity + Volumes

Partagez votre CKAD_Semaine1.md sur Teams avant vendredi 18h !

Questions ?

Ping Teams. Bonne préparation !