

Tp final devops leclerc-vincent from scratch

Lien du tp : https://github.com/bart120/m2cloud/blob/main/projet_k8s/projet.md

Disclaimer : ce tp a été réalisé sur un pc sous **W11 PRO** en pair programming (la fourniture des fichiers + organisation des dossiers a été proposée par l'ia, la verification + tests + validation par moi)

Voici une **décomposition en étapes simples**, dans l'ordre le plus “safe” pour réussir le TP sans se perdre. L'idée : **on valide un bloc à la fois** (build → run local → run K8s → perf → monitoring → sécu → logs → Helm).

Étape 0 — Préparer le dossier du “projet”

Objectif : avoir une structure claire.

- 01-compose-analysis/ (notes + schéma)
- 02-images/ (Dockerfiles + build scripts)
- 03-k8s-manifests/ (YAML bruts)
- 04-helm-chart/ (chart final)
- 05-tests/ (charge test + preuves HPA)
- docs/ (captures + explications)

Livrable attendu : un repo propre + un README.

Étape 1 — Analyser le docker-compose (le cœur du projet)

Objectif : comprendre **qui parle à qui, quels ports, quelles variables d'environnement, quels volumes**.

1. Ouvre docker-compose.yml
2. Pour chaque service (web, applicants.api, identity.api, jobs.api, sql.data, rabbitmq, user.data si présent) note :
 - Port exposé (ex: 8080:80)
 - Variables d'env (strings de connexion, URLs d'API, credentials)
 - Depends_on (dépendances)
 - Volumes (data SQL, rabbitmq)
 - Réseaux

 **Livrable** : un tableau “Service → Port → Env → Dépendances → Stockage”.

Étape 2 — Faire tourner en local avec Docker Compose (baseline)

Objectif : vérifier que tout fonctionne **avant Kubernetes**.

- Lancer : docker compose up -d
- Tester :
 - le web dans navigateur
 - endpoints API (ex: /health, /swagger, etc.)
 - login via identity.api si applicable

 **Livrable** : “preuve” que le stack marche (captures + notes des URLs).

Étape 3 — Construire les images Docker (une par app)

Objectif : produire des images “propres” et reproductibles.

1. Vérifie les Dockerfiles fournis
2. Standardise les tags :
 - your-registry/web:1.0.0
 - your-registry/applicants-api:1.0.0
 - etc.
3. Build toutes les images
4. Test rapide : docker run (au moins pour web et une API)

 **Livrable** : liste des images + tags + commande de build.

Étape 4 — Pousser dans un registry privé (Docker Hub privé ou Azure)

Objectif : tes nodes K8s doivent pouvoir pull tes images.

- Docker Hub privé : docker login, docker push ...
- Azure Container Registry (ACR) : push vers *.azurecr.io/...

 **Livrable** : preuve de push (captures ou output) + tags finaux.

Étape 5 — Choisir la cible Kubernetes (AKS recommandé)

Objectif : avoir ton cluster “prod-like”.

Option A (recommandée) : AKS

- 2–3 nodes minimum (ex : 2 nodes pour commencer)
- Active “metrics server” (souvent déjà ok sur AKS)

Option B : local

- k3d / kind / minikube (ok mais monitoring + ELK parfois plus lourd)

 **Livrable :** cluster accessible + kubectl get nodes OK.

Étape 6 — Déployer la base “data + messaging” (sql + rabbitmq + user.data)

Objectif : démarrer les composants stateful d’abord.

1. Namespace (ex: tp-prod)
2. sql.data :
 - **StatefulSet** (ou Deployment si simple, mais StatefulSet mieux)
 - **PVC** (volume persistant)
 - Service interne ClusterIP
3. rabbitmq :
 - PVC si tu veux persister
 - Service interne ClusterIP
4. user.data si c’est une base/stockage :
 - PVC + Service interne

 **Livrable :** kubectl get pods,svc,pvc -n tp-prod avec tout Running.

Étape 7 — Déployer les API (applicants/jobs/identity)

Objectif : les APIs doivent parler à SQL / Rabbit.

Pour chaque API :

- **Deployment**
- **Service ClusterIP**

- **Config :**
 - variables d'environnement (connection strings, rabbitmq host, urls...)
 - idéalement via **ConfigMap + Secret**
- **Ressources requests/limits** (celles données dans l'énoncé)
- **Restart auto :**
 - restartPolicy: Always (par défaut sur Deployment)
 - probes (étape 10)

 **Livrable :** APIs accessibles **dans le cluster** (test via kubectl port-forward).

Étape 8 — Déployer le web (exposé vers l'extérieur)

Objectif : le web consomme jobs.api + applicants.api et sera la porte d'entrée.

- Deployment + Service
- Ingress (HTTPS obligatoire)
- Variables d'env : URLs internes des APIs (DNS K8s)

 **Livrable :** web accessible depuis Internet (ou depuis ton réseau si local) **en HTTPS**.

Étape 9 — Mettre les ressources CPU/MEM + la HA

Objectif : répondre à la partie “Performance + Haute dispo”.

1. Ajoute sur **chaque Deployment** :
 - resources.requests et resources.limits selon l'énoncé
2. Répliques :
 - web : 2 (ou +)
 - APIs : 2
 - sql/rabbit : 1 (stateful)

 **Livrable :** kubectl describe deploy ... montre bien requests/limits + replicas.

Étape 10 — Liveness & Readiness probes (bilan de santé)

Objectif : redémarrage automatique + pas de trafic tant que pas prêt.

Pour chaque service :

- **readinessProbe** : endpoint “ready” (ou HTTP GET simple)
- **livenessProbe** : endpoint “health” (ou TCP si pas mieux)

Livrable : YAML + preuve que K8s détecte une panne (même simple : mauvais port → pod non ready).

Étape 11 — Affinités (performance demandée)

Objectif : coller les pods qui communiquent sur les mêmes nodes.

Affinités demandées :

- web proche de jobs.api + applicants.api
- jobs.api + applicants.api proche de sql.data
- identity.api proche de user.data

👉 Implémentation simple :

- **podAffinity preferredDuringSchedulingIgnoredDuringExecution**
- basée sur labels (ex: app=jobs-api)

Livrable : YAML + kubectl describe pod montre les règles.

Étape 12 — HPA (scalabilité) + preuve de montée en charge

Objectif : au moins 1 service autoscalé (web ou applicants.api).

1. Installer/valider **metrics-server**
2. Mettre HPA :
 - minReplicas: 2, maxReplicas: 6
 - target CPU: ex 60%
3. Générer charge :
 - hey, k6, ab (au choix)
4. Montrer :
 - kubectl get hpa -w
 - kubectl get pods -w (replicas augmentent)

 **Livrable** : capture avant/après charge prouvant l'augmentation.

Étape 13 — Monitoring Prometheus + kube-state-metrics

Objectif : métriques cluster + nodes + pods + control plane (selon dispo).

1. Installer Prometheus (souvent via Helm)
2. Installer kube-state-metrics
3. Vérifier :
 - o targets UP
 - o métriques CPU/mem pods
 - o métriques état objets K8s (deployments, pods, etc.)

 **Livrable** : dashboard ou captures Prometheus (Targets + une requête).

Étape 14 — Sécurité : HTTPS + ServiceAccount + RBAC + NetworkPolicy

Objectif : respecter les contraintes “sécurité”.

A) HTTPS uniquement (cert auto-signé)

- Créer un cert + secret TLS
- Ingress en TLS
- Redirection HTTP → HTTPS (selon ingress controller)

B) ServiceAccount dédié + RBAC restreint (au moins 1 microservice)

- Créer ServiceAccount
- Role (droits minimalistes)
- RoleBinding
- Associer au Deployment via serviceAccountName

C) NetworkPolicy (au moins 1)

Exemple simple conforme :

- Seule web peut appeler identity.api
- Seules applicants/jobs/identity peuvent parler à sql.data
- Bloquer le reste par défaut (policy “default deny” + exceptions)

-  **Livrable :** YAML + preuve (ex: un pod “test” ne peut plus ping/curl sql).
-

Étape 15 — Logs centralisés (ELK/EFK) + Dashboard + filtres erreurs

Objectif : exploitation logs.

Choix simple :

- **EFK** (Fluent Bit → Elasticsearch → Kibana)
ou
- **OpenSearch stack** (souvent plus léger selon distributions)

Exigences à produire :

- 1 dashboard affichant logs d'un service sur une période
- 1 filtre “erreurs” (level=ERROR, HTTP 5xx, etc.)

-  **Livrable :** captures Kibana/OpenSearch Dashboards (dashboard + filtre).
-

Étape 16 — Helm (déploiement automatisé)

Objectif : tout ton déploiement applicatif en Helm **sans** installer l'ingress controller dedans.

Ton Helm chart doit contenir :

- Deployments / StatefulSets
- Services
- Ingress (TLS)
- ConfigMaps / Secrets
- HPA
- NetworkPolicies
- ServiceAccount / RBAC
- (optionnel) Pod affinity, PDB

Structure type :

- Chart.yaml
- values.yaml

- templates/ (deploy, svc, ingress, hpa, netpol, rbac, pvc...)

 **Livrable :**

- helm install tp-prod ./chart -n tp-prod
- helm upgrade ...
- README avec variables configurables.

Maintenant que le fil conducteur du TP est posé, nous passons maintenant à la réalisation 😊

Tout au long du tp nous allons au fur et à mesure installer ce que nous avons besoins pour réaliser chaque étape.

Au niveau des prérequis, nous avons besoin d'une base minimal accessible à tous :

Docker-desktop (+WSL2) installé et à jours ainsi que le

Windows PowerShell à jour.

Ce TP est en grande parti réalisé en ligne de commande PowerShell, je recommande de prendre l'habitude de l'utiliser.

Au cours du tutoriel, vous allez pouvoir saisir pleinement tout l'intérêt de l'importance d'utiliser le terminal et réaliser à quel point c'est un outil puissant.

Pour commencer vous pouvez créer un dossier où vous allez réaliser le tp, dans mon cas c'est : C:\Users\Dimi ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices

Étape 0 — Initialiser le dossier projet (PowerShell)

0.1 — Créer le dossier racine (se placer dans votre répertoire dans mon cas

C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices)

```
mkdir tp-k8s-microservices
```

```
cd tp-k8s-microservices
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Tous droits réservés.

Installez la dernière version de PowerShell pour de nouvelles fonctionnalités et améliorations ! https://aka.ms/PSWindows

PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent> mkdir tp-k8s-microservices

Répertoire: C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent

Mode           LastWriteTime         Length Name
----           LastWriteTime         Length Name
<--->       23/12/2025      19:34                tp-k8s-microservices

PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent> cd tp-k8s-microservices
```

0.2 — Créer l'arborescence du projet

```
mkdir 01-compose-analysis
```

```
mkdir 02-images
```

```
mkdir 03-k8s-manifests
```

```
mkdir 04-helm-chart
```

```
mkdir 05-tests
```

```
mkdir docs
```

0.3 — Ajouter des sous-dossiers utiles (optionnel mais recommandé)

```
mkdir 02-images\scripts
```

```
mkdir 03-k8s-manifests\base
```

```
mkdir 03-k8s-manifests\apps
```

```
mkdir 03-k8s-manifests\infra
```

```
mkdir 03-k8s-manifests\security
```

```
mkdir 03-k8s-manifests\observability
```

```
mkdir 03-k8s-manifests\logging
```

0.4 — Créer des fichiers “README” vides (pour structurer)

```
New-Item -ItemType File -Path .\README.md -Force | Out-Null
```

```
New-Item -ItemType File -Path .\01-compose-analysis\README.md -Force | Out-Null
```

```
New-Item -ItemType File -Path .\02-images\README.md -Force | Out-Null
```

```
New-Item -ItemType File -Path .\03-k8s-manifests\README.md -Force | Out-Null
```

```
New-Item -ItemType File -Path .\04-helm-chart\README.md -Force | Out-Null
```

```
New-Item -ItemType File -Path .\05-tests\README.md -Force | Out-Null
```

```
New-Item -ItemType File -Path .\docs\README.md -Force | Out-Null
```

0.5 — Ajouter un .gitignore “standard” (le .gitignore doit être à la racine du projet – dans mon cas c'est : C:\Users\Dimi ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices)

```
@"
```

```
# OS / IDE
```

```
.DS_Store
```

```
Thumbs.db
```

```
.vscode/
```

```
.idea/
```

```
# Logs / temp
```

```
*.log
```

```
tmp/
```

```
temp/
```

```
# Node / Python (au cas où)
```

```
node_modules/
```

```
__pycache__/
```

```
.venv/  
  
# Secrets  
.env  
*.pem  
*.key  
*.pfx  
secrets*.yaml
```

```
# Helm packaging  
*.tgz  
  
# Kubernetes local generated  
kubeconfig*  
"@ | Set-Content -Encoding UTF8 .\gitignore
```

0.6 — Initialiser Git (Voir capture d'écran 0.7)

```
git init  
git branch -M main
```

0.7 — Premier commit “structure”

```
git add .  
git commit -m "initialization de la structure du projet"
```

```
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> git init  
Initialized empty Git repository in C:/users/dimitri ANGELY/Desktop/laboratoire/Leclerc-vincent/tp-k8s-microservices/.git/  
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> git branch -M main  
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> git add .  
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> git commit -m "initialisation de la structure du projet"  
[main (root-commit) 47fa21f] initialisation de la structure du projet  
 8 files changed, 30 insertions(+)  
 create mode 100644 .gitignore  
 create mode 100644 01-compose-analysis/README.md  
 create mode 100644 02-images/README.md  
 create mode 100644 03-k8s-manifests/README.md  
 create mode 100644 04-helm-chart/README.md  
 create mode 100644 05-tests/README.md  
 create mode 100644 README.md  
 create mode 100644 docs/README.md
```

0.8 — Vérifier que tout est bien créé

tree /F

Résultat attendu

Nous devons voir un dossier tp-k8s-microservices avec les 6 dossiers principaux + les sous-dossiers K8s.

```
PS C:\Users\Dimi...re\laboratoire\Leclerc-vincent\tp-k8s-microservices> tree /F
Structure du dossier pour le volume OS
Le numéro de série du volume est DA3F-C2F5
C:.
|   .gitignore
|   README.md
|
└── 01-compose-analysis
    README.md
|
└── 02-images
    README.md
    |
    └── scripts
|
└── 03-k8s-manifests
    README.md
    |
    ├── apps
    ├── base
    ├── infra
    ├── logging
    ├── observability
    └── security
|
└── 04-helm-chart
    README.md
|
└── 05-tests
    README.md
|
└── docs
    README.md
```

Pourquoi mettre .gitignore à la racine ?

Parce que :

- un `.gitignore` **s'applique à tout le repo**
 - il ignore :
 - secrets (`.env`, `certs`, clés)
 - fichiers générés (logs, Helm packages)
 - dépendances (`node_modules`, `.venv`)
 - tu **évites les erreurs classiques** (commit de secrets, `kubeconfig`, etc.)
-

🔴 À éviter absolument

✗ Mettre le `.gitignore` :

- dans `03-k8s-manifests/`
- dans `04-helm-chart/`
- dans un sous-dossier “au hasard”

→ Sinon Git **n'ignorera pas** ce qu'il faut globalement.

Bonnes pratiques (optionnel, plus tard)

Tu pourras **ajouter d'autres `.gitignore` locaux** dans :

- `02-images/` (si besoin spécifique)
- `04-helm-chart/` (Helm artefacts)

Mais **le principal reste toujours à la racine**.

La suite logique est d'**importer le code du cours** comme si on importait le code d'un développeur et le ranger proprement dans notre repo.

Objectif : avoir les apps “sources” séparées des fichiers K8s/Helm.

Étape 0.9 — Créer un dossier `src` pour le code du cours

Dans **tp-k8s-microservices** (la racine que tu montres), exécute :

```
mkdir src
```

```

PS C:\Users\Dimi ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> mkdir src

Répertoire: C:\Users\Dimi ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices

Mode          LastWriteTime    Length Name
----          LastWriteTime    Length Name
d-----      23/12/2025       19:43   src

```

Étape 0.10 — Mettre le repo du cours dans src

Cas A (recommandé) : cloner le repo du cours (c'est ce que j'ai choisi de faire)

1. Va dans src :

```
cd .\src
```

2. Clone (remplace l'URL par celle du cours) :

```
git clone <URL_DU_REPO_DU_COURS>
```

3. Reviens à la racine :

```
cd ..
```

```

PS C:\Users\Dimi ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> cd .\src
PS C:\Users\Dimi ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\src> git clone https://github.com/bart120/m2cloud.git
Cloning into 'm2cloud'...
remote: Enumerating objects: 310, done.
remote: Counting objects: 100% (310/310), done.
remote: Compressing objects: 100% (238/238), done.
remote: Total 310 (delta 58), reused 299 (delta 47), pack-reused 0 (from 0)
Receiving objects: 100% (310/310), 2.51 MiB | 21.27 MiB/s, done.
Resolving deltas: 100% (58/58), done.
PS C:\Users\Dimi ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\src> cd ..

```

Cas B : tu as déjà le code en zip/dossier (j'ai laisser volontairement pour avoir une alternative)

1. Copie/colle le dossier du cours dans :

```
tp-k8s-microservices/src/<nom-du-projet-cours>/
```

Étape 0.11 — Vérifier que tu as bien le docker-compose

Tu dois trouver un docker-compose.yml quelque part dans src.

Depuis la racine du projet :

Get-ChildItem -Path .\src -Recurse -Filter docker-compose.yml

```
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> Get-ChildItem -Path .\src -Recurse -Filter docker-compose.yml

Répertoire: C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent
\tp-k8s-microservices\src\m2cloud\compose

Mode                LastWriteTime         Length Name
----              LastWriteTime        Length Name
-a---       23/12/2025     19:49           1332 docker-compose.yml

Répertoire: C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent
\tp-k8s-microservices\src\m2cloud\projet_k8s\appscore

Mode                LastwriteTime         Length Name
----              LastwriteTime        Length Name
-a---       23/12/2025     19:49           1793 docker-compose.yml
```

- Résultat attendu : il affiche le chemin du fichier.

Problème rencontré : Ici, j'ai deux fichier docker-compose.yml, il faut donc choisir le bon fichier.

Comme le TP parle d'un projet complet (web + APIs + sql + rabbitmq), le plus probable est celui dans **projet_k8s\appscore** (mais on va vérifier proprement sans deviner).

Ouvrir les 2 compose et identifier lequel est "le bon"

Exécute ces 2 commandes (elles affichent juste les noms de services) :

Compose #1

```
(Get-Content .\src\m2cloud\compose\docker-compose.yml) | Select-String "^\s{0,2}[a-zA-Z0-9_.-]+:\s*$"
```

Compose #2

```
(Get-Content .\src\m2cloud\projet_k8s\appscore\docker-compose.yml) | Select-String "^\s{0,2}[a-zA-Z0-9_.-]+:\s*$"
```

- Résultat attendu : vous verrez des lignes du style web:, jobs.api:, sql.data: etc.

```
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> (Get-Content .\src\m2cloud\compose\docker-compose.yml) | Select-String "^\s{0,2}[a-zA-Z0-9_.-]+:\s*$"
networks:
  backend:
  frontend:
volumes:
  mysql_data:
services:
  database:
  myadmin:
  wordpress:

PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> (Get-Content .\src\m2cloud\projet_k8s\appscore\docker-compose.yml) | Select-String "^\s{0,2}[a-zA-Z0-9_.-]+:\s*$"
services:
  user_data:
  rabbitmq:
  webmc:
  applicants_api:
  jobs_api:
  identity_api:
  sql_data:
```

Une fois qu'on voit bien lequel contient bien :

```
web  
applicants.api  
identity.api  
jobs.api  
sql.data  
rabbitmq
```

👉 On choisit celui-là comme référence.

Ensuite, on crée un dossier “compose officiel” dans ton repo (pour éviter de chercher dans src/ à chaque fois).

Crée le dossier :

```
mkdir .\01-compose-analysis\compose-source
```

Copie le compose choisi (exemple : si c'est appscore) :

```
Copy-Item .\src\m2cloud\projet_k8s\appscore\docker-compose.yml .\01-compose-analysis\compose-source\docker-compose.yml -Force
```

```
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> mkdir .\01-compose-analysis\compose-source

Répertoire: C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent
\tp-k8s-microservices\01-compose-analysis

Mode                LastWriteTime        Length Name
----                LastWriteTime        Length Name
d----- 23/12/2025      19:56            compose-source

PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> Copy-Item .\src\m2cloud\projet_k8s\appscore\docker-compose.yml .\01-compose-analysis\compose-source\docker-compose.yml -Force
```

Étape 0.12 — Vérifier que la copie est OK

```
Get-ChildItem .\01-compose-analysis\compose-source
```

Tu dois voir docker-compose.yml.

```
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> Get-ChildItem .\01-compose-analysis\compose-source

Répertoire: C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\01-compose-analysis\compose-source

Mode          LastWriteTime      Length Name
----          -----          ----  --
-a--  23/12/2025    19:49        1793 docker-compose.yml

PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> |
```

Option “double check” :

```
(Get-Content .\01-compose-analysis\compose-source\docker-compose.yml) | Select-String "^\s{0,2}[a-zA-Z0-9_-]+:\s*$"
```

Tu dois revoir :

```
user.data, rabbitmq, webmvc, applicants.api, jobs.api, identity.api, sql.data
```

```
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> (Get-Content .\01-compose-analysis\compose-source\docker-compose.yml) | Select-String "^\s{0,2}[a-zA-Z0-9_-]+:\s*$"

services:
  user.data:
  rabbitmq:
  webmvc:
  applicants.api:
  jobs.api:
  identity.api:
  sql.data:
```

Étape 0.13 — Créer un fichier “notes d’analyse” (qui servira à l’étape 1)

Toujours à la racine :

```
New-Item -ItemType File -Path .\01-compose-analysis\compose-notes.md -Force | Out-Null
```

```
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> New-Item -ItemType File -Path .\01-compose-analysis\compose-notes.md -Force | Out-Null
```

Étape 0.14 — Vérifier que ton compose “officiel” est bien en place

```
Get-Content .\01-compose-analysis\compose-source\docker-compose.yml -TotalCount 30
```

```

PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> Get-Content .\01-compose-analysis\compose-source\docker-compose.yml -TotalCount 30
version: '3'

services:
  user_data:
    image: redis
    ports:
      - 6379:6379

  rabbitmq:
    image: rabbitmq:3-management
    ports:
      - 15672:15672
      - 5672:5672
    container_name: rabbitmq

  webmvc:
    image: web
    build:
      context: ./
      dockerfile: Web/Dockerfile
    ports:
      - 8000:80
    container_name: web
    depends_on:
      - applicants.api
      - identity.api
      - jobs.api
    restart: always

  applicants.api:

```

Étape 0.15 — Commit (fin étape 0)

```

git add .\01-compose-analysis\compose-source\docker-compose.yml

git add .\01-compose-analysis\compose-notes.md

git commit -m "ajout du bon docker-compose pour analyse"

```

```

PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> git add .\01-compose-analysis\compose-source\docker-compose.yml
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> git add .\01-compose-analysis\compose-notes.md
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> git commit -m "ajout du bon docker-compose pour analyse"
[main 6207b7e] ajout du bon docker-compose pour analyse
 1 file changed, 84 insertions(+)
 create mode 100644 01-compose-analysis/compose-source/docker-compose.yml

```

Fin de l'étape 0 — Préparer le dossier du “projet”

On commence l'**étape 1 : analyse du docker-compose**. Je te guide **ligne par ligne** et on va produire un fichier de notes clair qui te servira ensuite pour Kubernetes (Services, DNS, Secrets, NetworkPolicy, affinités...).

Étape 1.1 — Créer le fichier d’analyse (notes)

Exécute :

```
@"
```

```
# Analyse docker-compose (source officielle)
```

Fichier: 01-compose-analysis/compose-source/docker-compose.yml

1) Liste des services

- user.data (redis)
- rabbitmq (rabbitmq:3-management)
- webmvc (web)
- applicants.api
- jobs.api
- identity.api
- sql.data (mssql-linux)

2) Ports exposés (compose -> externe)

- user.data: 6379 -> 6379
- rabbitmq: 15672 -> 15672 (UI), 5672 -> 5672 (AMQP)
- webmvc: 8080 -> 80
- applicants.api: 8081 -> 80
- jobs.api: 8083 -> 80
- identity.api: 8084 -> 80
- sql.data: 1433 -> 1433

3) Dépendances (depends_on)

- webmvc dépend de: applicants.api, identity.api, jobs.api
- applicants.api dépend de: sql.data, rabbitmq
- jobs.api dépend de: sql.data, rabbitmq
- identity.api dépend de: user.data, rabbitmq

4) Variables d'environnement (à reproduire en ConfigMap/Secret)

applicants.api

-
ConnectionString=Server=sql.data;User=sa;Password=Pass@word;Database=dotnetgigs.applicants;

- HostRabbitmq=rabbitmq

jobs.api

-
ConnectionString=Server=sql.data;User=sa;Password=Pass@word;Database=dotnetgigs.jobs;

- HostRabbitmq=rabbitmq

identity.api

- RedisHost=user.data:6379

- HostRabbitmq=rabbitmq

5) Implications Kubernetes (traduction)

- Les noms utilisés (sql.data, rabbitmq, user.data) deviendront des DNS de Services Kubernetes.

- Les APIs n'ont pas besoin d'être exposées publiquement : Services internes (ClusterIP).

- Seule l'app web (webmvc) doit être exposée via Ingress (HTTPS).

- Le mot de passe SQL (Pass@word) doit aller dans un Secret (pas en clair).

- Redis + MSSQL + RabbitMQ : idéalement avec stockage persistant (PVC) en prod-like.

"@ | Set-Content -Encoding UTF8 .\01-compose-analysis\compose-notes.md

```
PS C:\Users\Dimi ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> @"
>> # Analyse docker-compose (source officielle)
>>
>> Fichier: 01-compose-analysis/compose-source/docker-compose.yml
>>
>> ## 1) Liste des services
>> - user.data (redis)
>> - rabbitmq (rabbitmq:3-management)
>> - webmvc (web)
>> - applicants.api
>> - jobs.api
>> - identity.api
>> - sql.data (mssql-linux)
>>
>> ## 2) Ports exposés (compose -> externe)
>> - user.data: 6379 -> 6379
>> - rabbitmq: 15672 -> 15672 (UI), 5672 -> 5672 (AMQP)
>> - webmvc: 8080 -> 80
>> - applicants.api: 8081 -> 80
>> - jobs.api: 8083 -> 80
>> - identity.api: 8084 -> 80
>> - sql.data: 1433 -> 1433
>>
>> ## 3) Dépendances (depends_on)
>> - webmvc dépend de: applicants.api, identity.api, jobs.api
>> - applicants.api dépend de: sql.data, rabbitmq
>> - jobs.api dépend de: sql.data, rabbitmq
>> - identity.api dépend de: user.data, rabbitmq
>>
>> ## 4) Variables d'environnement (à reproduire en ConfigMap/Secret)
>> ### applicants.api
>> - ConnectionString=Server=sql.data;User=sa;Password=Pass@word;Database=dotnetgigs.applicants;
>> - HostRabbitmq=rabbitmq
>>
>> ### jobs.api
>> - ConnectionString=Server=sql.data;User=sa;Password=Pass@word;Database=dotnetgigs.jobs;
>> - HostRabbitmq=rabbitmq
>>
>> ### identity.api
>> - RedisHost=user.data:6379
```

2) Vérifier que le fichier est bien rempli

Get-Content .\01-compose-analysis\compose-notes.md

```
PS C:\Users\DimiTRI ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> Get-Content .\01-compose-analysis\compose-notes.md
# Analyse docker-compose (source officielle)

Fichier: 01-compose-analysis/compose-source/docker-compose.yml

## 1) Liste des services
- user_data (redis)
- rabbitmq (rabbitmq:3-management)
- webmvc (web)
- applicants.api
- jobs.api
- identity.api
- sql.data (mssql-linux)

## 2) Ports exposés (compose -> externe)
- user_data: 6379 -> 6379
- rabbitmq: 15672 -> 15672 (UI), 5672 -> 5672 (AMQP)
- webmvc: 8080 -> 80
- applicants.api: 8081 -> 80
- jobs.api: 8083 -> 80
- identity.api: 8084 -> 80
- sql.data: 1433 -> 1433

## 3) Dépendances (depends_on)
- webmvc dépend de: applicants.api, identity.api, jobs.api
- applicants.api dépend de: sql.data, rabbitmq
- jobs.api dépend de: sql.data, rabbitmq
- identity.api dépend de: user.data, rabbitmq

## 4) Variables d'environnement (à reproduire en ConfigMap/Secret)
### applicants.api
- ConnectionString=Server=sql.data;User=sa;Password=Pass@word;Database=dotnetgigs.applicants;
- HostRabbitmq=rabbitmq

### jobs.api
- ConnectionString=Server=sql.data;User=sa;Password=Pass@word;Database=dotnetgigs.jobs;
- HostRabbitmq=rabbitmq

### identity.api
- RedisHost=user.data:6379
```

Étape 1.2 — Faire un mini schéma “qui parle à qui” (simple)

On va ajouter un petit schéma ASCII dans le fichier.

Exécute :

Add-Content -Encoding UTF8 .\01-compose-analysis\compose-notes.md @"

6) Schéma des communications (simplifié)

Internet (HTTPS)

|

V

[Ingress] -> [webmvc]

| \

| \

```

    v   v
[ applicants.api ] [ jobs.api ]
      \     /
      \   /
      v   v
[ sql.data ]

```

[identity.api] -> [user.data (redis)]

Tous (web + APIs) utilisent aussi:

- rabbitmq (AMQP 5672)
- UI RabbitMQ (15672) uniquement si besoin admin

"@

```

PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> Add-Content -Encoding UTF8 .\01-compose-analysis\compose-notes.md @"
>>
>> ## 6) Schéma des communications (simplifié)
>>
>> Internet (HTTPS)
>>   |
>>   v
>> [ Ingress ] -> [ webmvc ]
>>   |   \
>>   |   \
>>   v   v
>>   [ applicants.api ] [ jobs.api ]
>>   \   /
>>   v   v
>>   [ sql.data ]
>>
>> [ identity.api ] -> [ user.data (redis) ]
>>
>> Tous (web + APIs) utilisent aussi:
>> - rabbitmq (AMQP 5672)
>> - UI RabbitMQ (15672) uniquement si besoin admin
>>
>> "@

```

Étape 1.3 — Décisions “Kubernetes” qu’on fixe dès maintenant

On fixe des choix clairs (On pourras les justifier dans le rendu).

Ajoute ceci :

Add-Content -Encoding UTF8 .\01-compose-analysis\compose-notes.md @"

7) Décisions Kubernetes (choix projet)

Namespace:

- tp-prod

Exposition externe:

- webmvc: OUI (Ingress HTTPS)
- applicants.api: NON (interne)
- jobs.api: NON (interne)
- identity.api: NON (interne)
- rabbitmq UI (15672): NON par défaut (optionnel via port-forward)
- sql.data / redis: NON (interne)

Type de workload:

- webmvc / applicants.api / jobs.api / identity.api: Deployment
- rabbitmq: Deployment (ou StatefulSet si persistance)
- user.data (redis): Deployment (ou StatefulSet si persistance)
- sql.data (mssql): StatefulSet + PVC (recommandé)

Secrets:

- SQL SA password: Secret Kubernetes

"@

```
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> Add-Content -Encoding UTF8 .\01-compose-analysis\compose-notes.md @"
>>
>> ## 7) Décisions Kubernetes (choix projet)
>>
>> Namespace:
>> - tp-prod
>>
>> Exposition externe:
>> - webmvc: OUI (Ingress HTTPS)
>> - applicants.api: NON (interne)
>> - jobs.api: NON (interne)
>> - identity.api: NON (interne)
>> - rabbitmq UI (15672): NON par défaut (optionnel via port-forward)
>> - sql.data / redis: NON (interne)
>>
>> Type de workload:
>> - webmvc / applicants.api / jobs.api / identity.api: Deployment
>> - rabbitmq: Deployment (ou StatefulSet si persistance)
>> - user.data (redis): Deployment (ou StatefulSet si persistance)
>> - sql.data (mssql): StatefulSet + PVC (recommandé)
>>
>> Secrets:
>> - SQL SA password: Secret Kubernetes
>> "@"
```

Étape 1.4 — Commit de l'analyse (fin étape 1 “partie analyse”)

```
git add .\01-compose-analysis\compose-notes.md
```

```
git commit -m "analyse des dependances du docker-compose, des ports et de  
l'environnement global"
```

```
PS C:\Users\Dimmitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> git add .\01-compose-analysis\compose-notes.md  
warning: in the working copy of '01-compose-analysis/compose-notes.md', LF will be replaced by CRLF the next time Git touches it  
PS C:\Users\Dimmitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> git commit -m "analyse des dependances du docker-compose, des ports et de l'environnement global"  
[main af7707a] analyse des dependances du docker-compose, des ports et de l'environnement global  
1 file changed, 91 insertions(+)
```

Important (ce qu'on a appris et qui va guider tout le reste)

- Le **DNS interne K8s** doit permettre :
 - sql.data (Service) accessible par les APIs
 - rabbitmq (Service) accessible par les APIs
 - user.data (Service) accessible par identity.api
 - Le **webmvc** sera la seule entrée publique via Ingress HTTPS
 - Les variables d'env vont être converties en **ConfigMap + Secret**
-

Prochaine sous-étape (Étape 1.5)

On va maintenant **identifier les chemins des Dockerfiles** et vérifier qu'on sait construire les images :

- Web/Dockerfile
- Services/Applicants.Api/Dockerfile
- Services/Jobs.Api/Dockerfile
- Services/Identity.Api/Dockerfile
- Database/Dockerfile

Pour identifier les chemins des Dockerfiles:

```
Get-ChildItem .\src\m2cloud\projet_k8s\appscore -Recurse -Filter Dockerfile | Select-Object FullName
```

ou (même résultat avec les 2 commandes)

```
Get-ChildItem .\src\m2cloud\projet_k8s\appscore -Recurse -file -Filter Dockerfile | Select-Object FullName
```

```
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> Get-ChildItem .\src\m2cloud\projet_k8s\appscore -Recurse -Filter Dockerfile | Select-Object FullName
FullName
-----
C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\src\m2cloud\projet_k8s\appscore\Database\Dockerfile
C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\src\m2cloud\projet_k8s\appscore\Services\Applicants.Api\Dockerfile
C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\src\m2cloud\projet_k8s\appscore\Services\Identity.Api\Dockerfile
C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\src\m2cloud\projet_k8s\appscore\Services\Jobs.Api\Dockerfile
C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\src\m2cloud\projet_k8s\appscore\Web\Dockerfile
```

Étape 2.2 — Se placer dans le dossier du projet applicatif (contexte de build)

Le docker-compose.yml utilise context: ./ donc on se place dans appscore.

```
cd .\src\m2cloud\projet_k8s\appscore
```

```
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> cd .\src\m2cloud\projet_k8s\appscore
```

Étape 2.3 — Vérifier que Docker fonctionne

docker version

```
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\src\m2cloud\projet_k8s\appscore> docker version
Client:
  Version:          28.5.2
  API version:      1.51
  Go version:       go1.25.3
  Git commit:       ecc6942
  Built:            Wed Nov  5 14:45:58 2025
  OS/Arch:          windows/amd64
  Context:          desktop-linux

Server: Docker Desktop 4.51.0 (210443)
Engine:
  Version:          28.5.2
  API version:      1.51 (minimum version 1.24)
  Go version:       go1.25.3
  Git commit:       89c5e8f
  Built:            Wed Nov  5 14:43:25 2025
  OS/Arch:          linux/amd64
  Experimental:    false
containerd:
  Version:          v1.7.29
  GitCommit:        442cb34bda9a6a0fed82a2ca7cade05c5c749582
runc:
  Version:          1.3.3
  GitCommit:        v1.3.3-0-gd842d771
docker-init:
  Version:          0.19.0
  GitCommit:        de40ad0
```

Puis :

docker info

```
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc_vincent\tp-k8s-microservices\src\m2cloud\projet_k8s\appscore> docker info
Client:
  Version: 28.5.2
  Context: desktop-linux
  Debug Mode: false
  Plugins:
    ai: Docker AI Agent - Ask Gordon (Docker Inc.)
      Version: v1.9.11
      Path: C:\Program Files\Docker\cli-plugins\docker-ai.exe
    buildx: Docker Buildx (Docker Inc.)
      Version: v0.29.1-desktop.1
      Path: C:\Program Files\Docker\cli-plugins\docker-buildx.exe
    compose: Docker Compose (Docker Inc.)
      Version: v2.40.3-desktop.1
      Path: C:\Program Files\Docker\cli-plugins\docker-compose.exe
    debug: Get a shell into any image or container (Docker Inc.)
      Version: 0.0.45
      Path: C:\Program Files\Docker\cli-plugins\docker-debug.exe
    desktop: Docker Desktop commands (Docker Inc.)
      Version: v0.2.0
      Path: C:\Program Files\Docker\cli-plugins\docker-desktop.exe
    extension: Manages Docker extensions (Docker Inc.)
      Version: v0.2.31
      Path: C:\Program Files\Docker\cli-plugins\docker-extension.exe
    init: Creates Docker-related starter files for your project (Docker Inc.)
      Version: v1.4.0
      Path: C:\Program Files\Docker\cli-plugins\docker-init.exe
  mcp: Docker MCP Plugin (Docker Inc.)
    Version: v0.27.0
    Path: C:\Program Files\Docker\cli-plugins\docker-mcp.exe
  model: Docker Model Runner (Docker Inc.)
    Version: v0.1.46
    Path: C:\Program Files\Docker\cli-plugins\docker-model.exe
  offload: Docker Offload (Docker Inc.)
    Version: v0.5.17
    Path: C:\Program Files\Docker\cli-plugins\docker-offload.exe
  sandbox: Docker Sandbox (Docker Inc.)
    Version: v0.5.3
    Path: C:\Program Files\Docker\cli-plugins\docker-sandbox.exe
  sbom: View the packaged-based Software Bill Of Materials (SBOM) for an image (Anchore Inc.)
    Version: 0.6.0
```

Ça doit répondre sans erreur.

Étape 2.4 — Build local des images (comme le compose)

On va build exactement les mêmes images que dans le fichier docker-compose (image: web, image: applicants.api, etc.).

1) Build web

```
docker build -t web -f .\Web\Dockerfile .
```

```
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\src\m2cloud\projet_k8s\appscore> docker build -t web -f .\Web\Dockerfile .
[+] Building 44.7s (19/19) FINISHED docker:desktop-linux
=> [internal] load build definition from Dockerfile 0.1s
=> transferring dockerfile: 600B 0.0s
=> [internal] load metadata for mcr.microsoft.com/dotnet/core/sdk:2. 0.6s
=> [internal] load metadata for mcr.microsoft.com/dotnet/core/aspnet 0.6s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [internal] load build context 0.8s
=> => transferring context: 2.47MB 0.7s
=> [build 1/8] FROM mcr.microsoft.com/dotnet/core/sdk:2.1-stretch@s 25.7s
=> => mcr.microsoft.com/dotnet/core/sdk:2.1-stretch@sha256:a 0.0s
=> => sha256:d22db8cd915ea0c1414de6e2823aa0c47c 303.36MB / 303.36MB 20.0s
=> => sha256:6e1ebcd381250104f4231c05eaba5a6323 171.29MB / 171.29MB 10.3s
=> => sha256:fa5b7de410ab6a1c3e586b9befec9989a9a563 13.25MB / 13.25MB 1.2s
=> => sha256:c19c058edda50f20bd5b0d638034e29d9f713 49.76MB / 49.76MB 3.8s
=> => sha256:4ae15c65bfa0fdcc9053dc734ee6c034f5394ea0 4.34MB / 4.34MB 0.6s
=> => sha256:83600c1b4583abb5830c6d4a039cab9a102 11.30MB / 11.30MB 0.9s
=> => sha256:e1b8d230e067c240479c6a3f842e7f9d4ff10 45.38MB / 45.38MB 2.9s
=> => extracting sha256:eb18d230e067c240479c6a3f842e7f9d4ff1088e3072 1.1s
=> => extracting sha256:83600c1b4583abb5830c6d4a039cab9a102ac1a7b3b 0.2s
=> => extracting sha256:4ae15c65bfa0fdcc9053dc734ee6c034f5394ea0481de 0.1s
=> => extracting sha256:c19c058edda50f20bd5b0d638034e29d9f7138770d79 1.2s
=> => extracting sha256:fa5b7de410ab6a1c3e586b9befec9989a9a563e090672 0.2s
=> => extracting sha256:6e1ebcd381250104f4231c05eaba5a632388f249c644 1.3s
=> => extracting sha256:d22db8cd915ea0c1414de6e2823aa0c47cedda498400 4.8s
=> [base 1/2] FROM mcr.microsoft.com/dotnet/core/aspnet:2.1-stretch- 5.7s
=> => resolve mcr.microsoft.com/dotnet/core/aspnet:2.1-stretch-slim@ 0.0s
=> => sha256:204631cf5a97bea1887f214519fec9d82d0e4 17.67MB / 17.67MB 1.6s
=> => sha256:f66af43fc3abe3694f2bd46f6936aaadb99ad2 2.99MB / 2.99MB 0.7s
=> => sha256:06e05ce6a34c19717d50d6de009d20a269e522 60.13MB / 60.13MB 4.7s
=> => sha256:33f99cea2b7da8c6e0143c9fd7590c6d56f7d 22.53MB / 22.53MB 1.9s
=> => extracting sha256:33f99cea2b7da8c6e0143c9fd7590c6d56f7d310dd55 0.9s
=> => extracting sha256:2b4631cf5a97bea188f214519fec9d82d0e411e6d47 0.3s
=> => extracting sha256:f66af43fc3abe3694f2bd46f6936aaadb99ad45275 0.1s
=> => extracting sha256:06e05ce6a34c19717d50d6de009d20a269e522e8cfef2 0.7s
=> [base 2/2] WORKDIR /app 0.8s
=> [final 1/2] WORKDIR /app 0.2s
=> [build 2/8] WORKDIR /src 2.9s
=> [build 3/8] COPY Web Web/ 0.1s
=> [build 4/8] COPY Foundation\Http Foundation\Http/ 0.0s
=> [build 5/8] RUN dotnet restore "Web\Web.csproj" 3.9s
=> [build 6/8] COPY . 0.2s
=> [build 7/8] WORKDIR /src/Web 0.0s
=> [build 8/8] RUN dotnet build "Web.csproj" -c Release -o /app/build 2.7s
=> [publish 1/1] RUN dotnet publish "Web.csproj" -c Release -o /app/ 6.0s
=> [final 2/2] COPY --from=publish /app/publish . 0.3s
=> exporting image 1.7s
=> => exporting layers 1.2s
=> => exporting manifest sha256:b161d1b4839ee5b805b3748cfdf52f53fe76 0.0s
=> => exporting config sha256:7a2066840619504009bf6c4ddde3f540af4b7 0.0s
=> => exporting attestation manifest sha256:439ec557fc30be39676daacf 0.0s
=> => exporting manifest list sha256:23d3f97e6e44e74d73f7fd9d43f28fe 0.0s
=> => naming to docker.io/library/web:latest 0.0s
=> => unpacking to docker.io/library/web:latest 0.4s
```

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/zsdml5wra68pyjo12qxtwywpkf

2) Build applicants.api

```
docker build -t applicants.api -f .\Services\Applicants.Api\Dockerfile .
```

```
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\src\m2cloud\projet_k8s\appscore> docker build -t applicants.api -f .\Services\Applicants.Api\Dockerfile .
[+] Building 16.4s (19/19) FINISHED docker:desktop-linux
=> [internal] load build definition from Dockerfile 0.0s
=> transferring dockerfile: 278B 0.0s
=> [internal] load metadata for mcr.microsoft.com/dotnet/aspnet 0.2s
=> [internal] load metadata for mcr.microsoft.com/dotnet/sdk:2. 0.2s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [build 1/8] FROM mcr.microsoft.com/dotnet/core/sdk:2.1-stretch@sh 0.0s
=> => resolve mcr.microsoft.com/dotnet/core/sdk:2.1-stretch@sha256:a 0.0s
=> [internal] load build context 0.0s
=> => transferring context: 9.56kB 0.0s
=> [base 1/2] FROM mcr.microsoft.com/dotnet/core/aspnet:2.1-stretch- 0.0s
=> => resolve mcr.microsoft.com/dotnet/core/aspnet:2.1-stretch-slim@ 0.0s
=> CACHED [build 2/8] WORKDIR /src 0.0s
=> [build 3/8] COPY Services/Applicants.Api Services/Applicants.Api/ 0.1s
=> [build 4/8] COPY Foundation/Events Foundation/Events/ 0.1s
=> [build 5/8] RUN dotnet restore "Services/Applicants.Api/applicant" 7.3s
=> [build 6/8] COPY . 0.2s
=> [build 7/8] WORKDIR /src/Services/Applicants.Api 0.1s
=> [build 8/8] RUN dotnet build "applicants.api.csproj" -c Release - 2.2s
=> [publish 1/1] RUN dotnet publish "applicants.api.csproj" -c Rela 4.3s
=> CACHED [base 2/2] WORKDIR /app 0.0s
=> CACHED [final 1/2] WORKDIR /app 0.0s
=> [final 2/2] COPY --from=publish /app/publish . 0.1s
=> exporting image 1.4s
=> => exporting layers 1.1s
=> => exporting manifest sha256:b36b584ade16a8fbf3965003efb8cf1c4d6 0.0s
=> => exporting config sha256:e56ca480d6e7a82ff2a83f0887418b7de5c1f 0.0s
=> => exporting attestation manifest sha256:cc6ed0d5678fc0af6056a48d 0.0s
=> => exporting manifest list sha256:57476b1sebz0c480541b1a5a215d84d 0.0s
=> => naming to docker.io/library/applicants.api:latest 0.0s
=> => unpacking to docker.io/library/applicants.api:latest 0.2s
```

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/ylg92m0v5m1k5z1hem8rsoveo

3) Build jobs.api

```
docker build -t jobs.api -f .\Services\Jobs.Api\Dockerfile .
```

```
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\src\m2cloud\projet_k8s\appscore> docker build -t jobs.api -f .\Services\Jobs.Api\Dockerfile .
[+] Building 14.5s (19/19) FINISHED                                            docker:desktop:linux
=> [internal] load build definition from Dockerfile                         0.0s
=> => transferring dockerfile: 680B                                         0.0s
=> [internal] load metadata for mcr.microsoft.com/dotnet/core/aspnet          0.1s
=> [internal] load metadata for mcr.microsoft.com/dotnet/core/sdk:2.           0.1s
=> [internal] load .dockerignore                                              0.0s
=> => transferring context: 2B                                             0.0s
=> [build 1/8] FROM mcr.microsoft.com/dotnet/core/sdk:2.1-stretch@sha256:sh   0.0s
=> => resolve mcr.microsoft.com/dotnet/core/sdk:2.1-stretch@sha256:a          0.0s
=> [internal] load build context                                              0.0s
=> => transferring context: 9.56kB                                         0.0s
=> [base 1/2] FROM mcr.microsoft.com/dotnet/core/aspnet:2.1-stretch-          0.0s
=> => resolve mcr.microsoft.com/dotnet/core/aspnet:2.1-stretch-slim@          0.0s
=> CACHED [build 2/8] WORKDIR /src                                           0.0s
=> [build 3/8] COPY Services/Jobs.Api/Services/Jobs.Api/                      0.1s
=> [build 4/8] COPY Foundation/Events/Foundation/Events/                     0.0s
=> [build 5/8] RUN dotnet restore "Services/Jobs.Api/jobs.api.csproj"        5.8s
=> [build 6/8] COPY .                                                       0.1s
=> [build 7/8] WORKDIR /src/Services/Jobs.Api                                0.0s
=> [build 8/8] RUN dotnet build "jobs.api.csproj" -c Release -o /app         2.2s
=> [publish 1/1] RUN dotnet publish "jobs.api.csproj" -c Release -o          4.1s
=> CACHED [base 2/2] WORKDIR /app                                           0.0s
=> CACHED [final 1/2] WORKDIR /app                                         0.0s
=> [final 2/2] COPY --from=publis /app/publish .                           0.2s
=> => exporting to image                                                 1.4s
=> => exporting layers                                                 1.1s
=> => exporting manifest sha256:08a76aca4f446f72914df673995bbe607404 0.0s
=> => exporting config sha256:c62fd6691521d7499592831dfb1e8cff74dfbf5 0.0s
=> => exporting attestation manifest sha256:a6d1e0ad57789334e6fb75d6 0.0s
=> => exporting manifest list sha256:44a0ab035c048da9ea1f031ef639712 0.0s
=> => naming to docker.io/library/jobs.api:latest                          0.0s
=> => unpacking to docker.io/library/jobs.api:latest                        0.2s
View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/yr00spe63t7z682pw708rlq09
```

4) Build identity.api

```
docker build -t identity.api -f .\Services\Identity.Api\Dockerfile .
```

```
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\src\m2cloud\projet_k8s\appscore> docker build -t identity.api -f .\Services\Identity.Api\Dockerfile .
[+] Building 13.1s (20/20) FINISHED                                            docker:desktop:linux
=> [internal] load build definition from Dockerfile                         0.0s
=> => transferring dockerfile: 722B                                         0.0s
=> [internal] load metadata for mcr.microsoft.com/dotnet/core/sdk:2.           0.1s
=> [internal] load metadata for mcr.microsoft.com/dotnet/core/aspnet          0.1s
=> [internal] load .dockerignore                                              0.0s
=> => transferring context: 2B                                             0.0s
=> [build 1/9] FROM mcr.microsoft.com/dotnet/core/sdk:2.1-stretch@sha   0.0s
=> => resolve mcr.microsoft.com/dotnet/core/sdk:2.1-stretch@sha256:a          0.0s
=> [base 1/2] FROM mcr.microsoft.com/dotnet/core/aspnet:2.1-stretch-          0.0s
=> => resolve mcr.microsoft.com/dotnet/core/aspnet:2.1-stretch-slim@          0.0s
=> [internal] load build context                                              0.0s
=> => transferring context: 9.56kB                                         0.0s
=> CACHED [build 2/9] WORKDIR /src                                           0.0s
=> [build 3/9] COPY Services/Identity.Api/Services/Identity.Api/                      0.1s
=> [build 4/9] COPY Foundation/Events/Foundation/Events/                     0.0s
=> [build 5/9] RUN ls -la                                              0.2s
=> [build 6/9] RUN dotnet restore "Services/Identity.Api/Identity.Ap"        3.9s
=> [build 7/9] COPY .                                                       0.2s
=> [build 8/9] WORKDIR /src/Services/Identity.Api                                0.1s
=> [build 9/9] RUN dotnet build "Identity.Api.csproj" -c Release -o          2.2s
=> [publish 1/1] RUN dotnet publish "Identity.Api.csproj" -c Release          4.2s
=> CACHED [base 2/2] WORKDIR /app                                           0.0s
=> CACHED [final 1/2] WORKDIR /app                                         0.0s
=> [final 2/2] COPY --from=publish /app/publish .                           0.3s
=> => exporting to image                                                 1.6s
=> => exporting layers                                                 1.2s
=> => exporting manifest sha256:2e2cbb684b232b97e668ef296fc16e41517 0.0s
=> => exporting config sha256:dc622bd46704289d45fcff75980f3899330b16 0.0s
=> => exporting attestation manifest sha256:f611b9e228a52f8fbaeb199 0.0s
=> => exporting manifest list sha256:9cafefab9725dabc8032e1c816fb6687 0.0s
=> => naming to docker.io/library/identity.api:latest                          0.0s
=> => unpacking to docker.io/library/identity.api:latest                        0.2s
View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/yr00spe63t7z682pw708rlq09
```

5) Build sql.data (image mssql-linux)

⚠ Ici le context est ./Database (comme dans le compose). Donc :

```
docker build -t mssql-linux -f .\Database\ Dockerfile .\Database
```

```
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\src\m2cloud\projet_k8s\appscope> docker build -t mssql-linux -f .\Database\ Dockerfile .\Database
[+] Building 35.1s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 310B          0.0s
=> [internal] load metadata for mcr.microsoft.com/mssql/server:2017-
=> [internal] load .dockerrignore           0.0s
=> => transferring context: 2B            0.0s
=> [1/5] FROM mcr.microsoft.com/mssql/server:2017-latest@sha256:e051      31.2s
=> => resolve mcr.microsoft.com/mssql/server:2017-latest@sha256:e051      0.0s
=> => sha256:6038b4deeb58dc09a3ba444f38a6a184d2d88 69.70MB / 69.70MB  4.1s
=> => sha256:b4b37408036f381e8ebde1fb25cd2e54a 361.39MB / 361.39MB  26.4s
=> => sha256:699451820735fbcb93abbd25e13001df88b726c684 0.6s
=> => extracting sha256:b4b37408036f381e8ebde1fb25cd2e54a2aa4ba2 3.6s
=> => extracting sha256:6038b4deeb58dc09a3ba444f38a6a184d2d88af4bd01 1.1s
=> [internal] load build context          0.1s
=> => transferring context: 12.64kB       0.0s
=> [2/5] COPY entrypoint.sh    entrypoint.sh        2.2s
=> [3/5] COPY SqlCmdStartup.sh  SqlCmdStartup.sh   0.1s
=> [4/5] COPY SqlCmdScint.sql  SqlCmdScint.sql    0.1s
=> [5/5] RUN chmod +x ./SqlCmdStartup.sh 0.4s
=> exporting to image                   0.5s
=> expecting layers                   0.2s
=> exporting manifest sha256:c45d49613c0c99cea4cbbdd1686f5f37cb8f 0.0s
=> exporting config sha256:3a9296d7fe6ac7fc10c5436c0a57585d26 0.0s
=> exporting attestation manifest sha256:958fa3f2ebf84e4b9a19e7fc 0.0s
=> exporting manifest list sha256:40b704095f62c25e6c735d967c637904 0.0s
=> => naming to docker.io/library/mssql-linux:latest      0.0s
=> => unpacking to docker.io/library/mssql-linux:latest     0.1s

2 warnings found (use docker --debug to expand):
- Secrets used in Dockerfile: Do not use ENV or ENV instructions for sensitive data (ENV "SA_PASSWORD") (line 2)
- JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 8)

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/zbtchop7se3z168r7j5ssfumx
```

Étape 2.5 — Vérifier que les images existent

```
docker images | Select-String "web|applicants.api|jobs.api|identity.api|mssql-linux|redis|rabbitmq"
```

Tu dois voir au moins :

- web
- applicants.api
- jobs.api
- identity.api
- mssql-linux

(redis et rabbitmq viennent du hub, pas besoin de build)

```
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\src\m2cloud\projet_k8s\appscope> docker images | Select-String "web|applicants.api|jobs.api|identity.api|mssql-linux|redis|rabbitmq"
mssql-linux
latest
7a04095f62 13 seconds ago  1.81GB
identity.api
latest
9cafe9ab725d About a minute ago  428MB
jobs.api
latest
4a9ab6035c04 About a minute ago  428MB
applicants.api
latest
57476151eb2c About a minute ago  428MB
web
latest
23d3f97e6e44 2 minutes ago  429MB
conteneurorchestrationm2cloud-webserver
latest
1ca6a3b240ac 8 weeks ago  175MB
```

Étape 2.6 — Smoke test rapide en local (option mais recommandé)

On valide que tout démarre comme en compose :

docker compose up -d --build

```
PS C:\Users\Oimtri ANGELY\Desktop\laboratoire\Leclerc\vincent\tp-k8s-microservices\src\m2cloud\projet_k8s\appscore> docker compose up -d --build
time="2025-12-23T20:23:18+01:00" level=warning msg="C:\Users\Oimtri ANGELY\Desktop\laboratoire\Leclerc\vincent\tp-k8s-microservices\src\m2cloud\projet_k8s\appscore\docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove
to avoid potential confusion"
[+] Building 2.2s (6/6) FINISHED
  ✓ mssql-data Pulled          6.4s
    ✓ 0c16123e8fbf Pull complete
    ✓ ae4c94ad6fc Pull complete
    ✓ 1d9a59b95a2 Pull complete
    ✓ 44f9495f52a Pull complete
    ✓ 2cedcf21af7 Pull complete
    ✓ 60bf52e320520 Pull complete
    ✓ 300994812179 Pull complete
  ✓ jobs Pulled                7.0s
    ✓ d68039e4272 Pull complete
    ✓ 5e103b677479 Pull complete
    ✓ 14e209edfaef Pull complete
    ✓ f4e7f62f045 Pull complete
    ✓ 3f1971f111 Pull complete
    ✓ efa5b6560812 Pull complete
    ✓ b6e0b3708438 Pull complete
    ✓ d63c4408fb70 Pull complete
    ✓ 200430663435 Pull complete
    ✓ 7084a8f94ab0 Pull complete
[+] Building 2.2s (6/6) FINISHED
```

```
[+] Running 13/13
✓ mssql-linux              Built          0.0s
✓ jobs.api                  Built          0.0s
✓ applicants.api            Built          0.0s
✓ web                        Built          0.0s
✓ identity.api              Built          0.0s
✓ Network appscore_default  Created        0.2s
✓ Container rabbitmq        Started        1.7s
✓ Container appscore-user.data-1 Started        1.8s
✓ Container mssql-linux      Started        1.7s
✓ Container jobs.api        Started        1.2s
✓ Container applicants.api  Started        1.1s
✓ Container identity.api   Started        1.2s
✓ Container web              Started        1.4s
```

Puis :

docker ps

```
PS C:\Users\Oimtri ANGELY\Desktop\laboratoire\Leclerc\vincent\tp-k8s-microservices\src\m2cloud\projet_k8s\appscore> docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS                               NAMES
0ef1ade...        web                "dotnet Web.dll"   8 seconds ago     Up 8 seconds        0.0.0.0:8080->80/tcp, [::]:8080->80/tcp
a300d1f1...        applicants.api     "dotnet applicants.a..." 8 seconds ago     Up 8 seconds        0.0.0.0:8081->80/tcp, [::]:8081->80/tcp
9e23fc8f49...      jobs.api          "dotnet jobs.api.dll" 8 seconds ago     Up 8 seconds        Restarting (139) Less than a second ago
6e53d05d79c       mssql-linux        "/opt/mssql/bin/nom..." 9 seconds ago     Up 7 seconds        0.0.0.0:1433->1433/tcp, [::]:1433->1433/tcp
3dab737385ddc     rabbitmq3-management "docker-entrypoint.s..." 9 seconds ago     Up 7 seconds        0.0.0.0:5672->5672/tcp, 0.0.0.0:15672->15672/tcp, [::]:15672->15672/tcp
82dfe0390fb5      redis              "docker-entrypoint.s..." 9 seconds ago     Up 7 seconds        0.0.0.0:6379->6379/tcp, [::]:6379->6379/tcp
                                         NAMES
web                                         web
applicants.api                         applicants.api
jobs.api                                jobs.api
mssql-linux                             mssql-linux
rabbitmq3-management                     rabbitmq
redis                                     appscore-user.data-1
```

Et tu testes :

- Web : <http://localhost:8080>
- Applicants : <http://localhost:8081>
- Jobs : <http://localhost:8083>
- Identity : <http://localhost:8084>
- Rabbit UI : <http://localhost:15672>

(identifiants par défaut RabbitMQ souvent guest/guest si non modifié)

Après avoir vérifier que les différents services fonctionne via le navigateur, on va « pousser » les images dans un registre de conteneur.

2 choix s'offre à nous : Docker Hub (simple) et Azure ACR (pro). Tu peux suivre celui que tu veux, dans mon cas j'ai choisi Azure ACR.

Prérequis : être dans le chemin ...\\appscore et avoir les images existantes en local (web, applicants.api, jobs.api, identity.api, mssql-linux).

Étape 2.7 — Tag + Push vers un registry

2.7.0 — Vérifier les images locales (1 commande)

```
docker images | Select-String "web|applicants.api|jobs.api|identity.api|mssql-linux"
```

```
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\leclerc-vincent\tp-k8s-microservices\src\m2cloud\projet_k8s\appscore> docker images | Select-String "web|applicants.api|jobs.api|identity.api|mssql-linux"
mssql-linux
latest
e7107e02d858 7 minutes ago 1.81GB
identity.api
latest
f3eb8d3f4cc9 8 minutes ago 428MB
jobs.api
latest
76087e2b8416 8 minutes ago 428MB
applicants.api
latest
81d88aa95a00 9 minutes ago 428MB
web
latest
f49ea0cebbbd 9 minutes ago 429MB
conteneurorchestration2cloud-webserver
latest
1ca6a3b240ac 8 weeks ago 175MB
```

Pour ce qui va suivre nous allons partir du fait que vous avez des bases dans l'utilisations d'azure, si ce n'est pas le cas je vous invite à apprendre simplement comment créer un registre de conteneur sur azure. Poursuivons 😊

2.7.B.2 — Login Azure + ACR

```
az login --tenant <votre_tenant-id>
```

```
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\leclerc-vincent\tp-k8s-microservices\src\m2cloud\projet_k8s\appscore> az login --tenant
Select the account you want to log in with. For more information on login with Azure CLI, see https://go.microsoft.com/fwlink/?linkid=:
Retrieving subscriptions for the selection...
[Tenant and subscription selection]
No   Subscription name   Subscription ID           Tenant
----  -----  -----  -----
```

2.7.B.2 — Lister tes ACR et choisir le bon (sans deviner)

```
az acr list -o table
```

→ Tu vas voir une colonne **Name**. Repère le tien.

```
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\leclerc-vincent\tp-k8s-microservices\src\m2cloud\projet_k8s\appscore> az acr list -o table
NAME  RESOURCE GROUP  LOCATION  SKU  LOGIN SERVER  CREATION DATE  ADMIN ENABLED
-----  -----  -----  -----  -----  -----  -----
tpfinal  tp-final-conteneur-orchestration  francecentral  Standard  tpfinal.azurecr.io  2025-12-23T21:18:08Z  False
```

2.7.B.3 — Définir les variables (remplace MON_ACR)

⚠ Remplacer MON_ACR par le nom vu à l'étape précédente

```
$ACR="MON_ACR"
```

```
$LOGIN_SERVER="$ACR.azurecr.io"
```

```
$VERSION="1.0.0"
```

Créer un fichier “registry.env” (réutilisable)

Depuis la racine, faire :

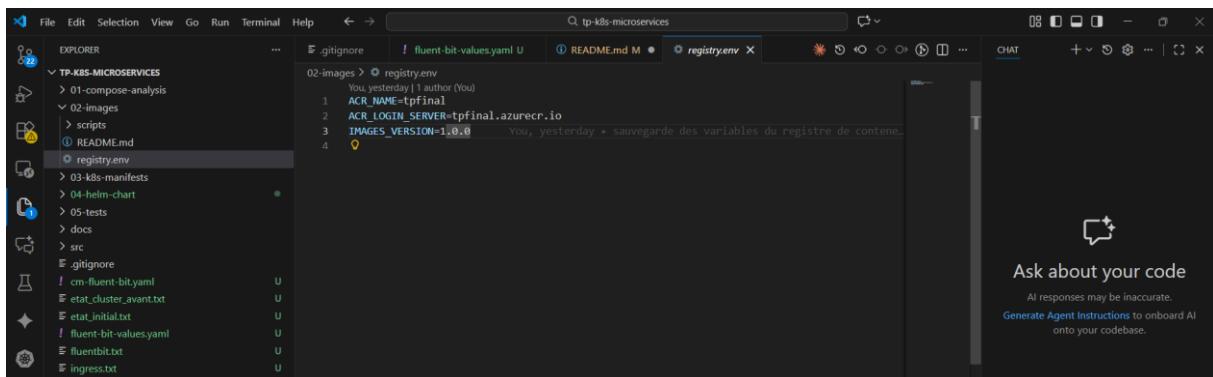
```
@"
```

```
ACR_NAME=tpfinal
```

```
ACR_LOGIN_SERVER=tpfinal.azurecr.io
```

```
IMAGES_VERSION=1.0.0
```

```
"@ | Set-Content -Encoding UTF8 .\02-images\registry.env
```



Commit :

```
git add .\02-images\registry.env
```

```
git commit -m "sauvegarde des variables du registre de conteneur azure"
```

```
PS C:\Users\Dimитри ANGELY\Desktop\laboratoire\leclerc-vincent\tp-k8s-microservices> git add .\02-images\registry.env
warning: in the working copy of '02-images/registry.env', LF will be replaced by CRLF the next time Git touches it
PS C:\Users\Dimитри ANGELY\Desktop\laboratoire\leclerc-vincent\tp-k8s-microservices> git commit -m "sauvegarde des variables du registre de conteneur azure"
[main 34db53d] sauvegarde des variables du registre de conteneur azure
 1 file changed, 3 insertions(+)
 create mode 100644 02-images/registry.env
```

2.7.B.4 — Login ACR

```
az acr login --name $ACR
```

```
PS C:\Users\Dimi ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\src\m2cloud\projet_k8s\appscore> az acr login --name $ACR
The output will be changed in next breaking change release(2.73.0) scheduled for May 2025. Exit code will be 1 if command fails for docker login.
Login Succeeded
```

2.7.B.5 — Vérifier les images locales à tagger

```
docker images | Select-String "web|applicants.api|jobs.api|identity.api|mssql-linux"
```

```
PS C:\Users\Dimi ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\src\m2cloud\projet_k8s\appscore> docker images | Select-String "web|applicants.api|jobs.api|identity.api|mssql-linux"
WARNING: This output is designed for human readability. For machine-readable output, please use --format.



| Image                                          | Size         | Virtual Size | Status   |
|------------------------------------------------|--------------|--------------|----------|
| applicants.api:latest                          | 81d88aa95a00 | 428MB        | 120MB U  |
| conteneurorchestrationm2cloud-webserver:latest | 1ca6a3b240ac | 175MB        | 45.2MB U |
| identity.api:latest                            | f3eb8d3f4cc9 | 428MB        | 120MB U  |
| jobs.api:latest                                | 76887e2b8416 | 428MB        | 120MB U  |
| mssql-linux:latest                             | e7107e02d858 | 1.81GB       | 459MB U  |
| web:latest                                     | f49ea0ceb8bd | 429MB        | 120MB U  |


```

2.7.B.6 — Tagger proprement (noms “k8s-friendly”)

```
docker tag web "$LOGIN_SERVER/webmvc:$VERSION"
```

```
docker tag applicants.api "$LOGIN_SERVER/applicants-api:$VERSION"
```

```
docker tag jobs.api "$LOGIN_SERVER/jobs-api:$VERSION"
```

```
docker tag identity.api "$LOGIN_SERVER/identity-api:$VERSION"
```

```
docker tag mssql-linux "$LOGIN_SERVER/sql-data:$VERSION"
```

```
PS C:\Users\Dimi ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\src\m2cloud\projet_k8s\appscore> docker tag web "$LOGIN_SERVER/webmvc:$VERSION"
PS C:\Users\Dimi ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\src\m2cloud\projet_k8s\appscore> docker tag applicants.api "$LOGIN_SERVER/applicants-api:$VERSION"
PS C:\Users\Dimi ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\src\m2cloud\projet_k8s\appscore> docker tag jobs.api "$LOGIN_SERVER/jobs-api:$VERSION"
PS C:\Users\Dimi ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\src\m2cloud\projet_k8s\appscore> docker tag identity.api "$LOGIN_SERVER/identity-api:$VERSION"
PS C:\Users\Dimi ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\src\m2cloud\projet_k8s\appscore> docker tag mssql-linux "$LOGIN_SERVER/sql-data:$VERSION"
```

2.7.B.7 — Push vers ACR

```
docker push "$LOGIN_SERVER/webmvc:$VERSION"
```

```
docker push "$LOGIN_SERVER/applicants-api:$VERSION"
```

```
docker push "$LOGIN_SERVER/jobs-api:$VERSION"
```

```
docker push "$LOGIN_SERVER/identity-api:$VERSION"
```

```
docker push "$LOGIN_SERVER/sql-data:$VERSION"
```

```

PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\src\m2cloud\projet_k8s\appscore> docker push "$LOGIN_SERVER/webmvc:$VERSION"
The push refers to repository [tpfinal.azurecr.io/webmvc]
4f4fb700ef54: Pushed
06e05c6a34c1: Pushed
33f99ceab7d: Pushed
2b463c1ef5a9: Pushed
006142832367: Pushed
f1ee1a1d9f31: Pushed
27b9514a671f: Pushed
f66af433fc3e: Pushed
1.0.0: digest: sha256:f49ea0cebbbdd16266227a55a7cee2eee24c598ee77b34dd39a2133a35f3489 size: 856
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\src\m2cloud\projet_k8s\appscore> docker push "$LOGIN_SERVER/applicants-api:$VERSION"
The push refers to repository [tpfinal.azurecr.io/applicants-api]
4f4fb700ef54: Mounted from webmvc
27b9514a671f: Mounted from webmvc
33f99ceab7d: Mounted from webmvc
06e05c6a34c1: Mounted from webmvc
2b463c1ef5a9: Mounted from webmvc
4c157858edf6: Pushed
2cd2a9724a83: Pushed
f66af433fc3e: Mounted from webmvc
1.0.0: digest: sha256:81d88a95a00cf01c82b96a3747aff962216cb50a8a1c962d7b078c47f4a5e4a size: 856
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\src\m2cloud\projet_k8s\appscore> docker push "$LOGIN_SERVER/jobs-api:$VERSION"
The push refers to repository [tpfinal.azurecr.io/jobs-api]
06e05c6a34c1: Mounted from webmvc
4133cd2b3b7a: Pushed
d20840fe41d2: Pushed
f66af433fc3e: Mounted from webmvc
27b9514a671f: Mounted from webmvc
33f99ceab7d: Mounted from webmvc
2b463c1ef5a9: Mounted from webmvc
4f4fb700ef54: Mounted from webmvc
1.0.0: digest: sha256:76887e2b841628680c947e1744d3a2ad03cdfc03b77301a2c5615e8cb3dd1e71 size: 856
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\src\m2cloud\projet_k8s\appscore> docker push "$LOGIN_SERVER/identity-api:$VERSION"
The push refers to repository [tpfinal.azurecr.io/identity-api]
4f4fb700ef54: Mounted from webmvc
781716ce5b5d: Pushed
df5c677875c1: Pushed
f66af433fc3e: Mounted from webmvc
27b9514a671f: Mounted from webmvc
33f99ceab7d: Mounted from webmvc
2b463c1ef5a9: Mounted from webmvc
06e05c6a34c1: Mounted from webmvc
1.0.0: digest: sha256:f3eb8d3f4cc1f170a7d72a15c369c5f00c81a76f424fce24dac9cdfcb09d11 size: 856
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\src\m2cloud\projet_k8s\appscore> docker push "$LOGIN_SERVER/sql-data:$VERSION"
The push refers to repository [tpfinal.azurecr.io/sql-data]
2ba40f18690c: Pushed
5ac3ad2a9a5c1: Pushed
b4b3740808036f: Pushed
c64583dd0dc: Pushed
212ad141705d: Pushed
4f4fb700ef54: Mounted from webmvc
699451820735: Pushed
6038b4deeb58: Pushed
1.0.0: digest: sha256:e7107e02d858bc3787ae732dd1b5dfa139144b7fb6281ef9e780a4011ecd7029 size: 856

```

2.7.B.8 — Vérification côté ACR (repos existants)

az acr repository list --name \$ACR -o table

```

PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\src\m2cloud\projet_k8s\appscore> az acr repository list --name $ACR -o table
Result
-----
applicants-api
identity-api
jobs-api
sql-data
webmvc
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\src\m2cloud\projet_k8s\appscore> az acr repository show-tags --name $ACR --repository webmvc -o table
Result
-----
1.0.0

```

Optionnel (voir les tags d'un repo, exemple webmvc) :

az acr repository show-tags --name \$ACR --repository webmvc -o table

```

PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\src\m2cloud\projet_k8s\appscore> az acr repository show-tags --name $ACR --repository webmvc -o table
Result
-----
1.0.0

```

En cas de déploiement par la suite sur AKS (important)

Pour qu'AKS puisse pull depuis ACR facilement :

- soit l'attacher sur l'ACR à l'AKS (az aks update --attach-acr ...)
- soit créer un imagePullSecret (si cluster non lié)

On fera ça au moment Kubernetes.

Étape 3 — Kubernetes (début) : Namespace + base manifests

On va faire minimal mais propre : **namespace + labels + secrets de base**.

3.1 — Vérifier que kubectl parle à ton cluster

Si tu es sur AKS, tu dois déjà avoir le kubeconfig. On teste.

kubectl version --client

```
PS C:\Users\Dimi...> kubectl version --client
Client Version: v1.34.1
Kustomize Version: v5.7.1
```

kubectl get nodes

```
PS C:\Users\Dimi...> kubectl get nodes
NAME      STATUS   ROLES      AGE   VERSION
docker-de...  Ready    control-plane   77d   v1.34.1
```

✓ Si kubectl get nodes marche → parfait.

✗ Si ça échoue, corriger l'erreur avant de continuer.

3.2 — Créer le namespace tp-prod

kubectl create namespace tp-prod

```
PS C:\Users\Dimi...> kubectl create namespace tp-prod
namespace/tp-prod created
```

Vérif :

kubectl get ns | Select-String "tp-prod"

```
PS C:\Users\Dimi...> kubectl get ns | Select-String "tp-prod"
tp-prod        Active   9s
```

3.3 — Ajouter des labels “organisation”

kubectl label namespace tp-prod project=tp-k8s env=prod --overwrite

```
PS C:\Users\Dimi...> kubectl label namespace tp-prod project=tp-k8s env=prod --overwrite
namespace/tp-prod labeled
```

Vérif :

kubectl get namespace tp-prod --show-labels

```
PS C:\Users\Dimi...> kubectl get namespace tp-prod --show-labels
NAME      STATUS   AGE   LABELS
tp-prod  Active   31s   env=prod.kubernetes.io/metadata.name=tp-prod,project=tp-k8s
```

3.4 — Créer le Secret SQL (mot de passe SA) ✓

Dans ton compose : Password=Pass@word

On le sort du clair → Secret K8s.

```
kubectl -n tp-prod create secret generic sql-secrets `
```

```
--from-literal=SA_PASSWORD='Pass@word'
```

```
PS C:\Users\Dimi ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl -n tp-prod create secret generic sql-secrets  
>> --from-literal=SA_PASSWORD='Pass@word'  
secret/sql-secrets created
```

Vérif :

```
kubectl -n tp-prod get secret sql-secrets
```

```
PS C:\Users\Dimi ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl -n tp-prod get secret sql-secrets  
NAME      TYPE     DATA   AGE  
sql-secrets  Opaque    1      7s
```

3.5 — Créer un fichier YAML “base” pour tracer tout ça dans Git

On va éviter de dépendre uniquement des commandes “kubectl create”.

3.5.1 — Créer 03-k8s-manifests\base\namespace.yaml

À la racine du repo :

```
@"  
  
apiVersion: v1  
  
kind: Namespace  
  
metadata:  
  
  name: tp-prod  
  
  labels:  
  
    project: tp-k8s  
  
  env: prod
```

```
"@ | Set-Content -Encoding UTF8 .\03-k8s-manifests\base\namespace.yaml
```

```
PS C:\Users\Dimi ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> @"  
>> apiVersion: v1  
>> kind: Namespace  
>> metadata:  
>>   name: tp-prod  
>>   labels:  
>>     project: tp-k8s  
>>     env: prod  
>> "@ | Set-Content -Encoding UTF8 .\03-k8s-manifests\base\namespace.yaml
```

```

apiVersion: v1
kind: Namespace
metadata:
  name: tp-prod
  labels:
    project: tp-k8s
    env: prod

```

3.5.2 — Créez 03-k8s-manifests\base\sql-secret.yaml

⚠️ Ici on utilise stringData (lisible), K8s encode en base64 automatiquement.

```

@"

apiVersion: v1

kind: Secret

metadata:

  name: sql-secrets

  namespace: tp-prod

type: Opaque

stringData:

  SA_PASSWORD: "Pass@word"

```

"@ | Set-Content -Encoding UTF8 .\03-k8s-manifests\base\sql-secret.yaml

```

PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> @@
>> apiVersion: v1
>> kind: Secret
>> metadata:
>>   name: sql-secrets
>>   namespace: tp-prod
>> type: Opaque
>> stringData:
>>   SA_PASSWORD: "Pass@word"
>> "@ | Set-Content -Encoding UTF8 .\03-k8s-manifests\base\sql-secret.yaml

```

(Oui c'est encore en clair dans Git mais c'est OK pour un TP, en version "pro", privilégié SealedSecrets ou Azure Key Vault)

3.6 — Commit des manifests de base

```
git add .\03-k8s-manifests\base\namespace.yaml
```

```
git add .\03-k8s-manifests\base\sql-secret.yaml
```

```
git commit -m "(k8s): ajout du namespace et secret de base sql"
```

```
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc_vincent\tp-k8s-microservices> git add .\03-k8s-manifests\base\namespace.yaml
warning: in the working copy of '03-k8s-manifests/base/namespace.yaml', LF will be replaced by CRLF the next time Git touches it
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc_vincent\tp-k8s-microservices> git add .\03-k8s-manifests\base\sql-secret.yaml
warning: in the working copy of '03-k8s-manifests/base/sql-secret.yaml', LF will be replaced by CRLF the next time Git touches it
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc_vincent\tp-k8s-microservices> git commit -m "(k8s): ajout du namespace et secret de base sql"
[main 314fb7] (k8s): ajout du namespace et secret de base sql
 2 files changed, 15 insertions(+)
  create mode 100644 03-k8s-manifests/base/namespace.yaml
  create mode 100644 03-k8s-manifests/base/sql-secret.yaml
```

Déployer user.data (Redis) + rabbitmq

3.7.1 — Crée Redis (user.data)

A) Crée 03-k8s-manifests\infra\redis-user-data.yaml

Exécute à la racine du repo :

```
@"

apiVersion: apps/v1

kind: Deployment

metadata:

  name: user-data

  namespace: tp-prod

  labels:

    app: user-data

spec:

  replicas: 1

  selector:

    matchLabels:

      app: user-data

  template:
```

```
metadata:  
labels:  
  app: user-data  
spec:  
  containers:  
    - name: redis  
      image: redis:7  
      ports:  
        - containerPort: 6379  
      resources:  
        requests:  
          cpu: "4m"  
          memory: "500Mi"  
        limits:  
          cpu: "250m"  
          memory: "800Mi"  
      livenessProbe:  
        tcpSocket:  
          port: 6379  
        initialDelaySeconds: 15  
        periodSeconds: 10  
      readinessProbe:  
        tcpSocket:  
          port: 6379  
        initialDelaySeconds: 5  
        periodSeconds: 5  
---  
apiVersion: v1
```

```
kind: Service
metadata:
  name: user-data
  namespace: tp-prod
  labels:
    app: user-data
spec:
  type: ClusterIP
  selector:
    app: user-data
  ports:
    - name: redis
      port: 6379
      targetPort: 6379
"@ | Set-Content -Encoding UTF8 .\03-k8s-manifests\infra\redis-user-data.yaml
```

```
PS C:\Users\Dimi ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> @"
>> apiVersion: apps/v1
>> kind: Deployment
>> metadata:
>>   name: user-data
>>   namespace: tp-prod
>>   labels:
>>     app: user-data
>> spec:
>>   replicas: 1
>>   selector:
>>     matchLabels:
>>       app: user-data
>>   template:
>>     metadata:
>>       labels:
>>         app: user-data
>>     spec:
>>       containers:
>>         - name: redis
>>           image: redis:7
>>           ports:
>>             - containerPort: 6379
>>           resources:
>>             requests:
>>               cpu: "4m"
>>               memory: "500Mi"
>>             limits:
>>               cpu: "250m"
>>               memory: "800Mi"
>>           livenessProbe:
>>             tcpSocket:
>>               port: 6379
>>             initialDelaySeconds: 15
>>             periodSeconds: 10
>>           readinessProbe:
>>             tcpSocket:
>>               port: 6379
>>             initialDelaySeconds: 5
>>             periodSeconds: 5
>> ---
>> apiVersion: v1
>> kind: Service
>> metadata:
>>   name: user-data
>>   namespace: tp-prod
>>   labels:
>>     app: user-data
>> spec:
>>   type: ClusterIP
>>   selector:
>>     app: user-data
>>   ports:
>>     - name: redis
>>       port: 6379
>>       targetPort: 6379
>> "@ | Set-Content -Encoding UTF8 .\03-k8s-manifests\infra\redis-user-data.yaml
```

B) Appliquer

```
kubectl apply -f .\03-k8s-manifests\infra\redis-user-data.yaml
```

```
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl apply -f .\03-k8s-manifests\infra\redis-user-data.yaml
deployment.apps/user-data created
service/user-data created
```

C) Vérifier

```
kubectl -n tp-prod get pods -l app=user-data
```

```
kubectl -n tp-prod get svc user-data
```

```
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl -n tp-prod get pods -l app=user-data
NAME           READY   STATUS    RESTARTS   AGE
user-data-7b697b6c7-jf8bm  1/1     Running   0          12s
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl -n tp-prod get svc user-data
NAME      TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
user-data  ClusterIP  10.99.200.185 <none>        6379/TCP  23s
```

3.7.2 — Créer RabbitMQ (interne)

A) Créer 03-k8s-manifests\infra\rabbitmq.yaml

```
@"
```

```
apiVersion: apps/v1
```

```
kind: Deployment
```

```
metadata:
```

```
  name: rabbitmq
```

```
  namespace: tp-prod
```

```
  labels:
```

```
    app: rabbitmq
```

```
spec:
```

```
  replicas: 1
```

```
  selector:
```

```
    matchLabels:
```

```
      app: rabbitmq
```

```
  template:
```

```
    metadata:
```

```
      labels:
```

```
        app: rabbitmq
```

```
    spec:
```

```
containers:
  - name: rabbitmq
    image: rabbitmq:3-management
    ports:
      - name: amqp
        containerPort: 5672
      - name: mgmt
        containerPort: 15672
    resources:
      requests:
        cpu: "50m"
        memory: "256Mi"
      limits:
        cpu: "500m"
        memory: "768Mi"
    livenessProbe:
      tcpSocket:
        port: 5672
      initialDelaySeconds: 20
      periodSeconds: 10
    readinessProbe:
      tcpSocket:
        port: 5672
      initialDelaySeconds: 10
      periodSeconds: 5
---
apiVersion: v1
kind: Service
```

```
metadata:  
  name: rabbitmq  
  namespace: tp-prod  
  
labels:  
  app: rabbitmq  
  
spec:  
  type: ClusterIP  
  
  selector:  
    app: rabbitmq  
  
  ports:  
    - name: amqp  
      port: 5672  
      targetPort: 5672  
  
    - name: mgmt  
      port: 15672  
      targetPort: 15672  
  
"@ | Set-Content -Encoding UTF8 .\03-k8s-manifests\infra\rabbitmq.yaml
```

```

PS C:\Users\Dimi ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> @"
>> apiVersion: apps/v1
>> kind: Deployment
>> metadata:
>>   name: rabbitmq
>>   namespace: tp-prod
>>   labels:
>>     app: rabbitmq
>> spec:
>>   replicas: 1
>>   selector:
>>     matchLabels:
>>       app: rabbitmq
>>   template:
>>     metadata:
>>       labels:
>>         app: rabbitmq
>>     spec:
>>       containers:
>>         - name: rabbitmq
>>           image: rabbitmq:3-management
>>       ports:
>>         - name: amqp
>>           containerPort: 5672
>>         - name: mgmt
>>           containerPort: 15672
>>       resources:
>>         requests:
>>           cpu: "50m"
>>           memory: "256Mi"
>>         limits:
>>           cpu: "500m"
>>           memory: "768Mi"
>>       livenessProbe:
>>         tcpSocket:
>>           port: 5672
>>         initialDelaySeconds: 20
>>         periodSeconds: 10
>>       readinessProbe:
>>         tcpSocket:
>>           port: 5672
>>         initialDelaySeconds: 10
>>         periodSeconds: 5
>>   ...
>> apiVersion: v1
>> kind: Service
>> metadata:
>>   name: rabbitmq
>>   namespace: tp-prod
>>   labels:
>>     app: rabbitmq
>> spec:
>>   type: ClusterIP
>>   selector:
>>     app: rabbitmq
>>   ports:
>>     - name: amqp
>>       port: 5672
>>       targetPort: 5672
>>     - name: mgmt
>>       port: 15672
>>       targetPort: 15672
>> "@ | Set-Content -Encoding UTF8 .\03-k8s-manifests\infra\rabbitmq.yaml

```

B) Appliquer

kubectl apply -f .\03-k8s-manifests\infra\rabbitmq.yaml

```

PS C:\Users\Dimi ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl apply -f .\03-k8s-manifests\infra\rabbitmq.yaml
deployment.apps/rabbitmq created
service/rabbitmq created

```

C) Vérifier

```
kubectl -n tp-prod get pods -l app=rabbitmq
```

```
kubectl -n tp-prod get svc rabbitmq
```

```
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl -n tp-prod get pods -l app=rabbitmq
NAME          READY   STATUS    RESTARTS   AGE
rabbitmq-78ccfb5c7b-66q7h   0/1     Running   0          13s
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl -n tp-prod get svc rabbitmq
NAME      TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
rabbitmq   ClusterIP  10.98.200.19 <none>        5672/TCP,15672/TCP   22s
```

A) Vérifier DNS interne (rabbitmq et user-data)

On lance un pod temporaire (curl) :

```
kubectl -n tp-prod run netcheck --image=curlimages/curl:8.5.0 -it --rm - sh
```

```
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl -n tp-prod run netcheck --image=curlimages/curl:8.5.0 -it --rm - sh
All commands and output from this session will be recorded in container logs, including credentials and sensitive information passed through the command prompt.
If you don't see a command prompt, try pressing enter.
```

Une fois dedans, exécute :

```
nslookup rabbitmq
```

```
nslookup user-data
```

Puis teste les ports :

```
nc -zv rabbitmq 5672
```

```
nc -zv user-data 6379
```

Puis quitte :

```
Exit
```

```

~ $ nslookup rabbitmq
Server:      10.96.0.10
Address:     10.96.0.10:53

** server can't find rabbitmq.cluster.local: NXDOMAIN
** server can't find rabbitmq.svc.cluster.local: NXDOMAIN
** server can't find rabbitmq.cluster.local: NXDOMAIN

Name:   rabbitmq.tp-prod.svc.cluster.local
Address: 10.98.200.19

** server can't find rabbitmq.svc.cluster.local: NXDOMAIN

~ $ nslookup user-data
Server:      10.96.0.10
Address:     10.96.0.10:53

** server can't find user-data.svc.cluster.local: NXDOMAIN
** server can't find user-data.cluster.local: NXDOMAIN
** server can't find user-data.cluster.local: NXDOMAIN

Name:   user-data.tp-prod.svc.cluster.local
Address: 10.99.200.185

** server can't find user-data.svc.cluster.local: NXDOMAIN

~ $ nc -zv rabbitmq 5672
rabbitmq (10.98.200.19:5672) open
~ $ nc -zv user-data 6379
user-data (10.99.200.185:6379) open
~ $ exit
Session ended, resume using 'kubectl attach netcheck -c netcheck -i -t' command when the pod is running
pod "netcheck" deleted from tp-prod namespace

```

- Si ça marche, on est prêts pour les APIs.

Interprétation du résultat :

Le test est très bon et il dit beaucoup de choses.

- 1) Ce que tes nslookup montrent (et pourquoi tu vois des NXDOMAIN)

Dans Kubernetes, le DNS “complet” d’un Service est :

rabbitmq.tp-prod.svc.cluster.local

user-data.tp-prod.svc.cluster.local

- Ceux-là résolvent bien → tu as bien l’IP du service.

Quand on tapes juste rabbitmq, Kubernetes essaie aussi plusieurs variantes automatiques. Et comme tu es dans le namespace tp-prod, la résolution “courte” attend surtout :

rabbitmq.tp-prod.svc.cluster.local

ou parfois rabbitmq (via search du resolv.conf) / selon configuration

Les NXDOMAIN que tu vois (rabbitmq.svc.cluster.local, rabbitmq.cluster.local, etc.) sont des tentatives “fallback” normales. 🤞 Ce n'est pas grave tant qu'au final il trouve rabbitmq.tp-prod.svc.cluster.local (ce qu'il fait).

En résumé :

- ✓ DNS OK (dans le namespace)
- ✓ Services trouvés

Les NXDOMAIN = des suffixes testés qui n'existent pas, rien d'alarmant.

2) Ce que tes nc -zv prouvent

rabbitmq:5672 open ✓

user-data:6379 open ✓

Ça prouve 3 choses importantes :

Le Service Kubernetes route bien vers les pods

Le réseau intra-cluster fonctionne

Aucune NetworkPolicy ne bloque ces flux (pas encore)

Donc, si une appli “n'arrive pas” à joindre Rabbit ou Redis, ce ne sera pas le réseau de base → plutôt :

mauvaise URL/hostname dans l'app

mauvais namespace ciblé

variable d'environnement manquante

port/proto incorrect

ou DNS recherché sous une forme que l'app n'utilise pas

Réultat : réseau et DNS de base bien validés ✓

3.7.4 — Commit infra

```
git add .\03-k8s-manifests\infra\redis-user-data.yaml  
git add .\03-k8s-manifests\infra\rabbitmq.yaml  
git commit -m "(k8s): ajout des services interne rabbitmq et redis (user.data)"
```

```
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> git add .\03-k8s-manifests\infra\redis-user-data.yaml  
warning: in the working copy of '03-k8s-manifests\infra\redis-user-data.yaml', LF will be replaced by CRLF the next time Git touches it  
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> git add .\03-k8s-manifests\infra\rabbitmq.yaml  
warning: in the working copy of '03-k8s-manifests\infra\rabbitmq.yaml', LF will be replaced by CRLF the next time Git touches it  
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> git commit -m "(k8s): ajout des services interne rabbitmq et redis (user.data)"  
[main e76a053] (k8s): ajout des services interne rabbitmq et redis (user.data)  
 2 files changed, 116 insertions(+)  
create mode 100644 03-k8s-manifests\infra\rabbitmq.yaml  
create mode 100644 03-k8s-manifests\infra\redis-user-data.yaml
```

Pour valider que Redis + RabbitMQ sont OK (pods Running + Services ClusterIP) :

```
kubectl -n tp-prod get pods -o wide
```

```
kubectl -n tp-prod get svc
```

```
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl -n tp-prod get pods -o wide  
NAME READY STATUS RESTARTS AGE IP NODE NOMINATED NODE READINESS GATES  
rabbitmq-78ccfb5c7b-66q7h 1/1 Running 0 4m51s 10.1.0.104 docker-desktop <none> <none>  
user-data-7b6897b6c7-jf8bm 1/1 Running 0 9m49s 10.1.0.103 docker-desktop <none> <none>  
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl -n tp-prod get svc  
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE  
rabbitmq ClusterIP 10.98.200.19 <none> 5672/TCP,15672/TCP 5m11s  
user-data ClusterIP 10.99.200.185 <none> 6379/TCP 10m
```

Étape 3.8 Maintenant : SQL Server (sql.data) en StatefulSet + PVC (indispensable avant les APIs).

Étape 3.8 — Déployer sql.data (SQL Server) en StatefulSet + PVC

3.8.1 — Créer le manifest SQL Server

Crée le fichier :

A) 03-k8s-manifests\infra\mssql-sql-data.yaml

Exécute :

```
@"
```

```
apiVersion: v1
```

```
kind: PersistentVolumeClaim
```

```
metadata:
```

```
  name: sql-data-pvc
```

```
  namespace: tp-prod
```

```
  labels:
```

```
app: sql-data
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 5Gi
---
apiVersion: v1
kind: Service
metadata:
  name: sql-data
  namespace: tp-prod
  labels:
    app: sql-data
spec:
  type: ClusterIP
  selector:
    app: sql-data
  ports:
    - name: mssql
      port: 1433
      targetPort: 1433
---
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: sql-data
```

```
namespace: tp-prod

labels:
  app: sql-data

spec:
  serviceName: sql-data
  replicas: 1
  selector:
    matchLabels:
      app: sql-data
  template:
    metadata:
      labels:
        app: sql-data
    spec:
      containers:
        - name: mssql
          image: tpfinal.azurecr.io/sql-data:1.0.0
          imagePullPolicy: IfNotPresent
        ports:
          - containerPort: 1433
            name: mssql
      env:
        - name: ACCEPT_EULA
          value: "Y"
        - name: SA_PASSWORD
          valueFrom:
            secretKeyRef:
              name: sql-secrets
```

```
key: SA_PASSWORD

volumeMounts:
  - name: sql-data
    mountPath: /var/opt/mssql

resources:
  requests:
    cpu: "4m"
    memory: "500Mi"
  limits:
    cpu: "1"
    memory: "2000Mi"

livenessProbe:
  tcpSocket:
    port: 1433
  initialDelaySeconds: 40
  periodSeconds: 10

readinessProbe:
  tcpSocket:
    port: 1433
  initialDelaySeconds: 20
  periodSeconds: 5

volumes:
  - name: sql-data
    persistentVolumeClaim:
      claimName: sql-data-pvc

"@ | Set-Content -Encoding UTF8 .\03-k8s-manifests\infra\mssql-sql-data.yaml
```

3.8.2 — Déployer SQL Server

```
PS C:\Users\Dimetri ANGELV\Desktop\laboratoire\leclerc-vincent\tp-k8s-microservices> @"
>> apiVersion: v1
>> kind: PersistentVolumeClaim
>> metadata:
>>   name: sql-data-pvc
>>   namespace: tp-prod
>>   labels:
>>     app: sql-data
>> spec:
>>   accessModes:
>>     - ReadWriteOnce
>>   resources:
>>     requests:
>>       storage: 5Gi
>> ---
>> apiVersion: v1
>> kind: Service
>> metadata:
>>   name: sql-data
>>   namespace: tp-prod
>>   labels:
>>     app: sql-data
>> spec:
>>   type: ClusterIP
>>   selector:
>>     app: sql-data
>>   ports:
>>     - name: mssql
>>       port: 1433
>>       targetPort: 1433
>> ---
>> apiVersion: apps/v1
>> kind: StatefulSet
>> metadata:
>>   name: sql-data
>>   namespace: tp-prod
>>   labels:
>>     app: sql-data
>> spec:
>>   serviceName: sql-data
>>   replicas: 1
>>   selector:
>>     matchLabels:
>>       app: sql-data
>>   template:
>>     metadata:
>>       labels:
>>         app: sql-data
>>     spec:
>>       containers:
>>         - name: mssql
>>           image: tpfinal.azurecr.io/sql-data:1.0.0
>>           imagePullPolicy: IfNotPresent
>>           ports:
>>             - containerPort: 1433
>>               name: mssql
>>           env:
>>             - name: ACCEPT_EULA
>>               value: "Y"
>>             - name: SA_PASSWORD
>>               valueFrom:
>>                 secretKeyRef:
>>                   name: sql-secrets
>>                   key: SA_PASSWORD
>>             volumeMounts:
>>               - name: sql-data
>>                 mountPath: /var/opt/mssql
>>             resources:
>>               requests:
>>                 cpu: "4m"
>>                 memory: "500Mi"
>>               limits:
>>                 cpu: "1"
>>                 memory: "2000Mi"
>>             livenessProbe:
>>               tcpSocket:
>>                 port: 1433
>>               initialDelaySeconds: 40
>>               periodSeconds: 10
>>             readinessProbe:
>>               tcpSocket:
>>                 port: 1433
>>               initialDelaySeconds: 20
>>               periodSeconds: 5
>>             volumes:
>>               - name: sql-data
>>                 persistentVolumeClaim:
>>                   claimName: sql-data-pvc
>>   "@ | Set-Content -Encoding UTF8 .\03-k8s-manifests\infra\mssql\sql-data.yaml
```

```
kubectl apply -f .\03-k8s-manifests\infra\mssql-sql-data.yaml
```

```
PS C:\Users\Dimi...tp-k8s-microservices> kubectl apply -f .\03-k8s-manifests\infra\mssql-sql-data.yaml
persistentvolumeclaim/sql-data-pvc created
service/sql-data created
statefulset.apps/sql-data created
```

3.8.3 — Vérifier PVC / Pod / Service

```
kubectl -n tp-prod get pvc
```

```
kubectl -n tp-prod get pods -l app=mysql-data
```

```
kubectl -n tp-prod get svc sql-data
```

```
PS C:\Users\Dimi...tp-k8s-microservices> kubectl -n tp-prod get pvc
NAME      STATUS  VOLUME          CAPACITY  ACCESS MODES  STORAGECLASS  VOLUMEATTRIBUTESCLASS  AGE
sql-data-pvc  Bound  pvc-d7810ed0-1ca0-4a85-a657-62c3f70b25ec  5Gi       Rwo        hostpath      <unset>           12s
PS C:\Users\Dimi...tp-k8s-microservices> kubectl -n tp-prod get pods -l app=mysql-data
NAME    READY  STATUS   RESTARTS  AGE
sql-data-0  0/1   Running  0          12s
PS C:\Users\Dimi...tp-k8s-microservices> kubectl -n tp-prod get svc sql-data
NAME        TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)  AGE
sql-data   ClusterIP  10.110.116.207  <none>        1433/TCP  12s
```

3.8.4 — Vérifier les logs SQL Server (important)

Si le pod ne passe pas Running, on check les logs :

```
kubectl -n tp-prod logs -l app=mysql-data --tail=80
```

```
PS C:\Users\Dimi...tp-k8s-microservices> kubectl -n tp-prod logs -l app=mysql-data --tail=80
2025-12-24 10:02:09.79 spid7s  Resource governor reconfiguration succeeded
2025-12-24 10:02:09.79 spid7s  SQL Server Audit is starting the audits. This is an informational message. No user action is required.
2025-12-24 10:02:09.79 spid7s  SQL Server Audit has started the audits. This is an informational message. No user action is required.
2025-12-24 10:02:09.81 spid7s  SQL Trace ID 1 was started by login "sa".
2025-12-24 10:02:09.83 spid7s  Server name is 'sql-data-0'. This is an informational message only. No user action is required.
2025-12-24 10:02:09.83 spid24s  Password policy update was successful.
2025-12-24 10:02:09.84 spid24s  Always On: The availability replica manager is starting. This is an informational message only. No user action is required.
2025-12-24 10:02:09.84 spid7s  Starting up database 'msdb'.
2025-12-24 10:02:09.84 spid12s  Starting up database 'mssqlsystemresource'.
2025-12-24 10:02:09.84 spid24s  Always On: The availability replica manager is waiting for the instance of SQL Server to allow client connections. This is an informational message only. No user action is required.
2025-12-24 10:02:09.84 spid12s  The resource database build version is 14.00.3515. This is an informational message only. No user action is required.
2025-12-24 10:02:09.85 spid12s  Starting up database 'model'.
2025-12-24 10:02:09.93 spid7s  Converting database 'msdb' from version 862 to the current version 869.
2025-12-24 10:02:09.94 spid7s  Database 'msdb' running the upgrade step from version 862 to version 863.
2025-12-24 10:02:09.94 spid22s  A self-generated certificate was successfully loaded for encryption.
2025-12-24 10:02:09.94 spid22s  Server is listening on [ 'any' <ipv6> 1433].
2025-12-24 10:02:09.94 spid22s  Server is listening on [ 'any' <ipv4> 1433].
2025-12-24 10:02:09.95 Server  Server is listening on [ ::1 <ipv6> 1434].
2025-12-24 10:02:09.95 Server  Server is listening on [ 127.0.0.1 <ipv4> 1434].
2025-12-24 10:02:09.95 Server  Dedicated admin connection support was established for listening locally on port 1434.
```

3.8.5 — Test réseau rapide : port 1433 accessible en interne

```
kubectl -n tp-prod run netcheck2 --image=busybox:1.36 -it --rm -- sh
```

Dans le shell :

```
nslookup sql-data
```

```
nc -zv sql-data 1433
```

```
exit
```

```

PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclecc-vincent\tp-k8s-microservices> kubectl -n tp-prod run netcheck2 --image=busybox:1.36 -it --rm -- sh
All commands and output from this session will be recorded in container logs, including credentials and sensitive information passed through the command prompt.
If you don't see a command prompt, try pressing enter.
/ #
/ #
/ # nslookup sql-data
Server:      10.96.0.10
Address:     10.96.0.10:53

** server can't find sql-data.svc.cluster.local: NXDOMAIN
Name:   sql-data.tp-prod.svc.cluster.local
Address: 10.110.116.207

** server can't find sql-data.cluster.local: NXDOMAIN

** server can't find sql-data.cluster.local: NXDOMAIN
** server can't find sql-data.svc.cluster.local: NXDOMAIN

/ # nc -zv sql-data 1433
sql-data (10.110.116.207:1433) open
/ # exit
Session ended, resume using 'kubectl attach netcheck2 -c netcheck2 -i -t' command when the pod is running
pod "netcheck2" deleted from tp-prod namespace

```

Interprétation du résultat :

Parfait — même conclusion que tout à l'heure, et c'est une excellente preuve.

Ce que ça prouve

1) DNS OK (dans tp-prod)

nslookup sql-data finit par résoudre :

- sql-data.tp-prod.svc.cluster.local -> 10.110.116.207

Les lignes **NXDOMAIN** (sql-data.svc.cluster.local, sql-data.cluster.local) sont juste des suffixes testés automatiquement qui n'existent pas. Tant que le FQDN avec le namespace est trouvé, c'est bon.

2) Réseau OK + port SQL ouvert

nc -zv sql-data 1433 → **open**

 Donc :

- le Service sql-data fonctionne
- le port 1433 est accessible depuis un pod du namespace
- pas de blocage réseau (NetworkPolicy) sur ce flux

Ce que ça implique pour ton problème “webmvc / apps”

Si une appli n'arrive pas à se connecter à SQL (ou dit “No such device or address” / “Name or service not known”), ce n'est **pas** :

- Kubernetes DNS global
- ni le réseau intra-cluster

- ni le Service SQL

C'est très probablement :

- un **mauvais hostname** dans la config (sql-data vs sql-data.tp-prod.svc.cluster.local)
- un hostname **dans un autre namespace** (default)
- une **variable d'environnement non injectée** (tu avais déjà vu Environment: <none>)
- ou un format d'URL/connection string invalide

3.8.6 — Commit

```
git add .\03-k8s-manifests\infra\mssql-sql-data.yaml
```

```
git commit -m "(k8s): add sql.data mssql statefulset with pvc and secret"
```

```
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc_vincent\tp-k8s-microservices> git add .\03-k8s-manifests\infra\mssql-sql-data.yaml
warning: in the working copy of '03-k8s-manifests\infra\mssql-sql-data.yaml', LF will be replaced by CRLF the next time Git touches it
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc_vincent\tp-k8s-microservices> git commit -m "(k8s): ajout de sql.data mssql statefulset avec pvc et secret"
[main c4b5c7e] (k8s): ajout de sql.data mssql statefulset avec pvc et secret
1 file changed, 87 insertions(+)
create mode 100644 03-k8s-manifests\infra\mssql-sql-data.yaml
```

On valide que SQL Server est **Running**, **PVC Bound**, et les DB dotnetgigs.applicants + dotnetgigs.jobs existent déjà pour la suite :

```
kubectl -n tp-prod get pods -o wide
```

```
kubectl -n tp-prod get pvc
```

```
kubectl -n tp-prod logs -l app=sql-data --tail=40
```

```
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\leclerc-vincent\tp-k8s-microservices> kubectl -n tp-prod get pods -o wide
NAME           READY   STATUS    RESTARTS   AGE   IP          NODE   NOMINATED NODE   READINESS GATES
rabbitmq-78cfb5c7b-66q7h   1/1    Running   0          12m   10.1.0.104   docker-desktop   <none>        <none>
sql-data-0      1/1    Running   0          4m36s  10.1.0.106   docker-desktop   <none>        <none>
user-data-7b6897b6c7-jf8bm 1/1    Running   0          17m   10.1.0.103   docker-desktop   <none>        <none>
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\leclerc-vincent\tp-k8s-microservices> kubectl -n tp-prod get pvc
NAME           STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   VOLUME ATTRIBUTESCLASS   AGE
sql-data-pvc   Bound    pvc-d7810ed0-1ca0-4a85-a657-62c3f70b25ec   5Gi        Rwo          hostpath       <unset>        4m50s
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\leclerc-vincent\tp-k8s-microservices> kubectl -n tp-prod logs -l app=sql-data --tail=40
2025-12-24 10:02:08.71 spid12s   The tempdb database has 1 data file(s).
2025-12-24 10:02:08.73 spid24s   The Service Broker endpoint is in disabled or stopped state.
2025-12-24 10:02:08.73 spid24s   The Database Mirroring endpoint is in disabled or stopped state.
2025-12-24 10:02:08.74 spid24s   Service Broker manager has started.
2025-12-24 10:02:08.99 spid51   Starting up database 'dotnetgigs.applicants'.
2025-12-24 10:02:09.03 spid51   Parallel redo is started for database 'dotnetgigs.applicants' with worker pool size [10].
2025-12-24 10:02:09.05 spid51   Parallel redo is shutdown for database 'dotnetgigs.applicants' with worker pool size [10].
Changed database context to 'master'.
2025-12-24 10:02:09.11 spid7s   Recovery is complete. This is an informational message only. No user action is required.
2025-12-24 10:02:09Changed database context to 'dotnetgigs.applicants'.
.11 spid18s   The default language (LCID 0) has been set for engine and full-text services.
2025-12-24 10:02:09.26 spid51   Starting up database 'dotnetgigs.jobs'.
2025-12-24 10:02:09.34 spid51   Parallel redo is started for database 'dotnetgigs.jobs' with worker pool size [10].
2025-12-24 10:02:09.36 spid51   Parallel redo is shutdown for database 'dotnetgigs.jobs' with worker pool size [10].
(10 rows affected)
(10 rows affected)
Changed database context to 'dotnetgigs.jobs'.
(1 rows affected)
Using 'dbghelp.dll' version '4.0.5'
2025-12-24 10:06:51.11 spid51
```

Étape 3.9 : déployer les APIs (applicants/jobs/identity) avec leurs env vars + ressources + probes + Services internes (ClusterIP)

Nous allons maintenant déployer les APIs (applicants.api, jobs.api, identity.api) avec :

- Services **ClusterIP**
- Env vars depuis compose (ConnectionString, HostRabbitmq, RedisHost)
- Resources requests/limits (celles de l'énoncé)
- Probes (simple TCP sur port 80)
- **Affinité** (on la mettra juste après, étape 3.10)

3.9.1 — Créer applicants.api

A) Fichier 03-k8s-manifests\apps\applicants-api.yaml

@"

apiVersion: apps/v1

kind: Deployment

metadata:

name: applicants-api

namespace: tp-prod

labels:

app: applicants-api

spec:

replicas: 2

selector:

matchLabels:

app: applicants-api

template:

metadata:

labels:

app: applicants-api

spec:

containers:

- name: applicants-api

image: tpfinal.azurecr.io/applicants-api:1.0.0

imagePullPolicy: IfNotPresent

ports:

- containerPort: 80

name: http

env:

```
- name: ConnectionString
  value: "Server=sql-
data;User=sa;Password=Pass@word;Database=dotnetgigs.applicants;"

- name: HostRabbitmq
  value: "rabbitmq"

resources:
requests:
  cpu: "500m"
  memory: "500Mi"

limits:
  cpu: "1"
  memory: "1500Mi"

readinessProbe:
tcpSocket:
  port: 80
initialDelaySeconds: 10
periodSeconds: 5

livenessProbe:
tcpSocket:
  port: 80
initialDelaySeconds: 20
periodSeconds: 10

---
apiVersion: v1
kind: Service
metadata:
  name: applicants-api
  namespace: tp-prod
```

```
labels:  
  app: applicants-api  
  
spec:  
  type: ClusterIP  
  
  selector:  
    app: applicants-api  
  
  ports:  
    - name: http  
      port: 80  
      targetPort: 80  
  
"@ | Set-Content -Encoding UTF8 .\03-k8s-manifests\apps\applicants-api.yaml
```

```

PS C:\Users\Dimity ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> @"
>> apiVersion: apps/v1
>> kind: Deployment
>> metadata:
>>   name: applicants-api
>>   namespace: tp-prod
>>   labels:
>>     app: applicants-api
>> spec:
>>   replicas: 2
>>   selector:
>>     matchLabels:
>>       app: applicants-api
>>   template:
>>     metadata:
>>       labels:
>>         app: applicants-api
>>     spec:
>>       containers:
>>         - name: applicants-api
>>           image: tpfinal.azurecr.io/applicants-api:1.0.0
>>           imagePullPolicy: IfNotPresent
>>       ports:
>>         - containerPort: 80
>>           name: http
>>       env:
>>         - name: ConnectionString
>>           value: "Server=sql-data;User=sa;Password=Pass@word;Database=dotnetgigs.applicants;"
>>         - name: HostRabbitmq
>>           value: "rabbitmq"
>>       resources:
>>         requests:
>>           cpu: "500m"
>>           memory: "500Mi"
>>         limits:
>>           cpu: "1"
>>           memory: "1500Mi"
>>       readinessProbe:
>>         tcpSocket:
>>           port: 80
>>         initialDelaySeconds: 10
>>         periodSeconds: 5
>>       livenessProbe:
>>         tcpSocket:
>>           port: 80
>>         initialDelaySeconds: 20
>>         periodSeconds: 10
>> ---
>> apiVersion: v1
>> kind: Service
>> metadata:
>>   name: applicants-api
>>   namespace: tp-prod
>>   labels:
>>     app: applicants-api
>> spec:
>>   type: ClusterIP
>>   selector:
>>     app: applicants-api
>>   ports:
>>     - name: http
>>       port: 80
>>       targetPort: 80
>> "@ | Set-Content -Encoding UTF8 .\03-k8s-manifests\apps\applicants-api.yaml

```

⚠ Note : ici la ConnectionString contient encore le mot de passe en clair (comme ton compose). Après validation, on le mettra proprement via Secret.

B) Apply

```
kubectl apply -f .\03-k8s-manifests\apps\applicants-api.yaml
```

```
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl apply -f .\03-k8s-manifests\apps\applicants-api.yaml
deployment.apps/applicants-api created
service/applicants-api created
```

C) Vérif

```
kubectl -n tp-prod get pods -l app=applicants-api
```

```
kubectl -n tp-prod get svc applicants-api
```

```
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl -n tp-prod get pods -l app=applicants-api
NAME           READY   STATUS    RESTARTS   AGE
applicants-api-698c5b59df-lkv6s  1/1     Running   0          23s
applicants-api-698c5b59df-sk9n4  1/1     Running   0          22s
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl -n tp-prod get svc applicants-api
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
applicants-api   ClusterIP   10.100.5.242   <none>       80/TCP    23s
```

3.9.2 — Créer jobs.api

A) Fichier 03-k8s-manifests\apps\jobs-api.yaml

```
@"
apiVersion: apps/v1
kind: Deployment
metadata:
  name: jobs-api
  namespace: tp-prod
  labels:
    app: jobs-api
spec:
  replicas: 2
  selector:
    matchLabels:
      app: jobs-api
  template:
    metadata:
      labels:
```

```
app: jobs-api

spec:

containers:
  - name: jobs-api

    image: tpfinal.azurecr.io/jobs-api:1.0.0
    imagePullPolicy: IfNotPresent

  ports:
    - containerPort: 80
      name: http

  env:
    - name: ConnectionString
      value: "Server$sql-
data;User=sa;Password=Pass@word;Database=dotnetgigs.jobs;"

    - name: HostRabbitmq
      value: "rabbitmq"

resources:
  requests:
    cpu: "500m"
    memory: "500Mi"

  limits:
    cpu: "1"
    memory: "1500Mi"

readinessProbe:
  tcpSocket:
    port: 80

  initialDelaySeconds: 10
  periodSeconds: 5

livenessProbe:
```

```
tcpSocket:  
  port: 80  
initialDelaySeconds: 20  
periodSeconds: 10  
  
---  
  
apiVersion: v1  
kind: Service  
metadata:  
  name: jobs-api  
  namespace: tp-prod  
labels:  
  app: jobs-api  
spec:  
  type: ClusterIP  
  selector:  
    app: jobs-api  
  ports:  
    - name: http  
      port: 80  
      targetPort: 80  
"@ | Set-Content -Encoding UTF8 .\03-k8s-manifests\apps\jobs-api.yaml
```

```
PS C:\Users\Dimi ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> @"
>> apiVersion: apps/v1
>> kind: Deployment
>> metadata:
>>   name: jobs-api
>>   namespace: tp-prod
>>   labels:
>>     app: jobs-api
>> spec:
>>   replicas: 2
>>   selector:
>>     matchLabels:
>>       app: jobs-api
>>   template:
>>     metadata:
>>       labels:
>>         app: jobs-api
>>     spec:
>>       containers:
>>         - name: jobs-api
>>           image: tpfinal.azurecr.io/jobs-api:1.0.0
>>           imagePullPolicy: IfNotPresent
>>       ports:
>>         - containerPort: 80
>>           name: http
>>       env:
>>         - name: ConnectionString
>>           value: "Server=sql-data;User=sa;Password=Pass@word;Database=dotnetgigs.jobs;"
>>         - name: HostRabbitmq
>>           value: "rabbitmq"
>>       resources:
>>         requests:
>>           cpu: "500m"
>>           memory: "500Mi"
>>         limits:
>>           cpu: "1"
>>           memory: "1500Mi"
>>       readinessProbe:
>>         tcpSocket:
>>           port: 80
>>           initialDelaySeconds: 10
>>           periodSeconds: 5
>>       livenessProbe:
>>         tcpSocket:
>>           port: 80
>>           initialDelaySeconds: 20
>>           periodSeconds: 10
>> ---
>> apiVersion: v1
>> kind: Service
>> metadata:
>>   name: jobs-api
>>   namespace: tp-prod
>>   labels:
>>     app: jobs-api
>> spec:
>>   type: ClusterIP
>>   selector:
>>     app: jobs-api
>>   ports:
>>     - name: http
>>       port: 80
>>       targetPort: 80
>> "@ | Set-Content -Encoding UTF8 .\03-k8s-manifests\apps\jobs-api.yaml
```

B) Apply + vérif

```
kubectl apply -f .\03-k8s-manifests\apps\jobs-api.yaml
```

```
kubectl -n tp-prod get pods -l app=jobs-api
```

```
kubectl -n tp-prod get svc jobs-api
```

```
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl apply -f .\03-k8s-manifests\apps\jobs-api.yaml
deployment.apps/jobs-api created
service/jobs-api created
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl -n tp-prod get pods -l app=jobs-api
NAME           READY   STATUS    RESTARTS   AGE
jobs-api-575578675-bz6v6  0/1     ContainerCreating   0          0s
jobs-api-575578675-lwbwp  0/1     ContainerCreating   0          0s
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl -n tp-prod get svc jobs-api
NAME      TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
jobs-api   ClusterIP  10.110.39.160 <none>        80/TCP    0s
```

3.9.3 — Crée identity.api

A) Fichier 03-k8s-manifests\apps\identity-api.yaml

```
@"
```

```
apiVersion: apps/v1
```

```
kind: Deployment
```

```
metadata:
```

```
  name: identity-api
```

```
  namespace: tp-prod
```

```
  labels:
```

```
    app: identity-api
```

```
spec:
```

```
  replicas: 2
```

```
  selector:
```

```
    matchLabels:
```

```
      app: identity-api
```

```
  template:
```

```
    metadata:
```

```
      labels:
```

```
        app: identity-api
```

```
    spec:
```

```
      containers:
```

```
- name: identity-api
  image: tpfinal.azurecr.io/identity-api:1.0.0
  imagePullPolicy: IfNotPresent
  ports:
    - containerPort: 80
      name: http
  env:
    - name: RedisHost
      value: "user-data:6379"
    - name: HostRabbitmq
      value: "rabbitmq"
  resources:
    requests:
      cpu: "500m"
      memory: "500Mi"
    limits:
      cpu: "1"
      memory: "1500Mi"
  readinessProbe:
    tcpSocket:
      port: 80
    initialDelaySeconds: 10
    periodSeconds: 5
  livenessProbe:
    tcpSocket:
      port: 80
    initialDelaySeconds: 20
    periodSeconds: 10
```

```
apiVersion: v1
kind: Service
metadata:
  name: identity-api
  namespace: tp-prod
labels:
  app: identity-api
spec:
  type: ClusterIP
  selector:
    app: identity-api
  ports:
    - name: http
      port: 80
      targetPort: 80
"@ | Set-Content -Encoding UTF8 .\03-k8s-manifests\apps\identity-api.yaml
```

```
PS C:\Users\Dimi ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> @"
>> apiVersion: apps/v1
>> kind: Deployment
>> metadata:
>>   name: identity-api
>>   namespace: tp-prod
>>   labels:
>>     app: identity-api
>> spec:
>>   replicas: 2
>>   selector:
>>     matchLabels:
>>       app: identity-api
>>   template:
>>     metadata:
>>       labels:
>>         app: identity-api
>>     spec:
>>       containers:
>>         - name: identity-api
>>           image: tpfinal.azurecr.io/identity-api:1.0.0
>>           imagePullPolicy: IfNotPresent
>>       ports:
>>         - containerPort: 80
>>           name: http
>>       env:
>>         - name: RedisHost
>>           value: "user-data:6379"
>>         - name: HostRabbitmq
>>           value: "rabbitmq"
>>       resources:
>>         requests:
>>           cpu: "500m"
>>           memory: "500Mi"
>>         limits:
>>           cpu: "1"
>>           memory: "1500Mi"
>>       readinessProbe:
>>         tcpSocket:
>>           port: 80
>>         initialDelaySeconds: 10
>>         periodSeconds: 5
>>       livenessProbe:
>>         tcpSocket:
>>           port: 80
>>         initialDelaySeconds: 20
>>         periodSeconds: 10
>>   ...
>> apiVersion: v1
>> kind: Service
>> metadata:
>>   name: identity-api
>>   namespace: tp-prod
>>   labels:
>>     app: identity-api
>> spec:
>>   type: ClusterIP
>>   selector:
>>     app: identity-api
>>   ports:
>>     - name: http
>>       port: 80
>>       targetPort: 80
>> "@ | Set-Content -Encoding UTF8 .\03-k8s-manifests\apps\identity-api.yaml
```

B) Apply + vérif

```
kubectl apply -f .\03-k8s-manifests\apps\identity-api.yaml
```

```
kubectl -n tp-prod get pods -l app=identity-api
```

```
kubectl -n tp-prod get svc identity-api
```

```
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl apply -f .\03-k8s-manifests\apps\identity-api.yaml
deployment.apps/identity-api created
service/identity-api created
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl -n tp-prod get pods -l app=identity-api
NAME          READY   STATUS    RESTARTS   AGE
identity-api-7bbc6f6cb8-ccr6w  0/1     ContainerCreating   0          1s
identity-api-7bbc6f6cb8-sx8jq  0/1     ContainerCreating   0          1s
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl -n tp-prod get svc identity-api
NAME            TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
identity-api   ClusterIP   10.99.169.115   <none>        80/TCP        1s
```

3.9.4 — Test “DNS + accès HTTP interne” (super utile)

On fait un pod curl temporaire :

```
kubectl -n tp-prod run curltest --image=curlimages/curl:8.5.0 -it --rm -- sh
```

Dans le shell :

```
curl -i http://applicants-api/
```

```
curl -i http://jobs-api/
```

```
curl -i http://identity-api/
```

```
exit
```

✓ Même si tu ne connais pas les routes exactes, au minimum tu dois voir un HTTP/1.1 200 ou 302 ou même 404 (mais pas “couldn’t resolve host” / “connection refused”).

```
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl -n tp-prod run curltest --image=curlimages/curl:8.5.0 -it -- sh
All commands and output from this session will be recorded in container logs, including credentials and sensitive information passed through the command prompt.
If you don't see a command prompt, try pressing enter.
~ $ curl -i http://applicants-api/
HTTP/1.1 404 Not Found
Date: Wed, 24 Dec 2025 10:11:31 GMT
Server: Kestrel
Content-Length: 0

~ $ curl -i http://jobs-api/
HTTP/1.1 404 Not Found
Date: Wed, 24 Dec 2025 10:11:45 GMT
Server: Kestrel
Content-Length: 0

~ $ curl -i http://identity-api/
HTTP/1.1 404 Not Found
Date: Wed, 24 Dec 2025 10:12:11 GMT
Server: Kestrel
Content-Length: 0

~ $ exit
Session ended, resume using 'kubectl attach curltest -c curltest -i -t' command when the pod is running
pod "curltest" deleted from tp-prod namespace
```

3.9.5 — Commit

```
git add .\03-k8s-manifests\apps\applicants-api.yaml
```

```
git add .\03-k8s-manifests\apps\jobs-api.yaml
```

```
git add .\03-k8s-manifests\apps\identity-api.yaml
```

git commit -m """(k8s): deployment des APIs comme services interne avec les ressources et les probes""

```
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> git add .\03-k8s-manifests\apps\applicants-api.yaml
warning: in the working copy of '03-k8s-manifests\apps\applicants-api.yaml', LF will be replaced by CRLF the next time Git touches it
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> git add .\03-k8s-manifests\apps\jobs-api.yaml
warning: in the working copy of '03-k8s-manifests\apps\jobs-api.yaml', LF will be replaced by CRLF the next time Git touches it
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> git add .\03-k8s-manifests\apps\identity-api.yaml
warning: in the working copy of '03-k8s-manifests\apps\identity-api.yaml', LF will be replaced by CRLF the next time Git touches it
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> git commit -m "(k8s): deployment des APIs comme services interne avec les ressources et les probes"
[main e5eb7d8] (k8s): deployment des APIs comme services interne avec les ressources et les probes
 3 files changed, 186 insertions(+)
 create mode 100644 03-k8s-manifests\apps\applicants-api.yaml
 create mode 100644 03-k8s-manifests\apps\identity-api.yaml
 create mode 100644 03-k8s-manifests\apps\jobs-api.yaml
```

On valide que tout est **Running** et les Services internes sont OK :

```
kubectl -n tp-prod get pods
```

```
kubectl -n tp-prod get svc
```

```
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl -n tp-prod get pods
NAME           READY   STATUS    RESTARTS   AGE
applicants-api-698c5b59df-lkv6s  1/1     Running   0          5m36s
applicants-api-698c5b59df-sk9n4  1/1     Running   0          5m35s
identity-api-7bbc6f6cb8-ccr6w   1/1     Running   0          3m50s
identity-api-7bbc6f6cb8-sx8jq   1/1     Running   0          3m50s
jobs-api-575578675-bz6v6       1/1     Running   0          4m16s
jobs-api-575578675-lwbwp       1/1     Running   0          4m16s
rabbitmq-78ccfb5c7b-66q7h      1/1     Running   0          20m
sql-data-0                      1/1     Running   0          12m
user-data-7b6897b6c7-jf8bm      1/1     Running   0          25m
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl -n tp-prod get svc
NAME        TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)        AGE
applicants-api  ClusterIP  10.100.5.242 <none>        80/TCP        5m36s
identity-api   ClusterIP  10.99.169.115 <none>        80/TCP        3m50s
jobs-api       ClusterIP  10.110.39.160 <none>        80/TCP        4m16s
rabbitmq       ClusterIP  10.98.200.19  <none>        5672/TCP,15672/TCP 20m
sql-data        ClusterIP  10.110.116.207 <none>       1433/TCP      12m
user-data       ClusterIP  10.99.200.185 <none>       6379/TCP      25m
```

Étape 3.10 : déployer webmvc, puis on ajoute l'**affinité demandée** (web proche de jobs + applicants). Ensuite on fera HTTPS/Ingress.

Étape 3.10 — Déployer webmvc (service interne + port-forward)

3.10.1 — Créer le manifest webmvc

A) Fichier 03-k8s-manifests\apps\webmvc.yaml

Exécute :

```
@"
```

```
apiVersion: apps/v1
```

```
kind: Deployment
```

```
metadata:
  name: webmvc
  namespace: tp-prod
  labels:
    app: webmvc
spec:
  replicas: 2
  selector:
    matchLabels:
      app: webmvc
  template:
    metadata:
      labels:
        app: webmvc
    spec:
      containers:
        - name: webmvc
          image: tpfinal.azurecr.io/webmvc:1.0.0
          imagePullPolicy: IfNotPresent
      ports:
        - containerPort: 80
          name: http
      resources:
        requests:
          cpu: "4m"
          memory: "900Mi"
        limits:
          cpu: "1"
```

```
    memory: "2000Mi"

  readinessProbe:
    tcpSocket:
      port: 80

    initialDelaySeconds: 10
    periodSeconds: 5

  livenessProbe:
    tcpSocket:
      port: 80

    initialDelaySeconds: 20
    periodSeconds: 10

---
apiVersion: v1
kind: Service
metadata:
  name: webmvc
  namespace: tp-prod
  labels:
    app: webmvc
spec:
  type: ClusterIP
  selector:
    app: webmvc
  ports:
    - name: http
      port: 80
      targetPort: 80
"@ | Set-Content -Encoding UTF8 .\03-k8s-manifests\apps\webmvc.yaml
```

```

PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> @"
>> apiVersion: apps/v1
>> kind: Deployment
>> metadata:
>>   name: webmvc
>>   namespace: tp-prod
>>   labels:
>>     app: webmvc
>> spec:
>>   replicas: 2
>>   selector:
>>     matchLabels:
>>       app: webmvc
>>   template:
>>     metadata:
>>       labels:
>>         app: webmvc
>>     spec:
>>       containers:
>>         - name: webmvc
>>           image: tpfinal.azurecr.io/webmvc:1.0.0
>>           imagePullPolicy: IfNotPresent
>>       ports:
>>         - containerPort: 80
>>           name: http
>>       resources:
>>         requests:
>>           cpu: "4m"
>>           memory: "900Mi"
>>         limits:
>>           cpu: "1"
>>           memory: "2000Mi"
>>       readinessProbe:
>>         tcpSocket:
>>           port: 80
>>       initialDelaySeconds: 10
>>       periodSeconds: 5
>>     livenessProbe:
>>       tcpSocket:
>>         port: 80
>>       initialDelaySeconds: 20
>>       periodSeconds: 10
>>   ...
>> apiVersion: v1
>> kind: Service
>> metadata:
>>   name: webmvc
>>   namespace: tp-prod
>>   labels:
>>     app: webmvc
>> spec:
>>   type: ClusterIP
>>   selector:
>>     app: webmvc
>>   ports:
>>     - name: http
>>       port: 80
>>       targetPort: 80
>> "@ | Set-Content -Encoding UTF8 .\03-k8s-manifests\apps\webmvc.yaml

```

B) Apply

kubectl apply -f .\03-k8s-manifests\apps\webmvc.yaml

```

PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl apply -f .\03-k8s-manifests\apps\webmvc.yaml
deployment.apps/webmvc created
service/webmvc created

```

C) Vérifier

```
kubectl -n tp-prod get pods -l app=webmvc
```

```
kubectl -n tp-prod get svc webmvc
```

```
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl -n tp-prod get pods -l app=webmvc
NAME           READY   STATUS    RESTARTS   AGE
webmvc-757695d44d-lrfd7  1/1     Running   0          22s
webmvc-757695d44d-pkrdr  1/1     Running   0          22s
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl -n tp-prod get svc webmvc
NAME      TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
webmvc   ClusterIP  10.96.199.96 <none>        80/TCP    23s
```

3.10.2 — Tester le web depuis ta machine (sans Ingress)

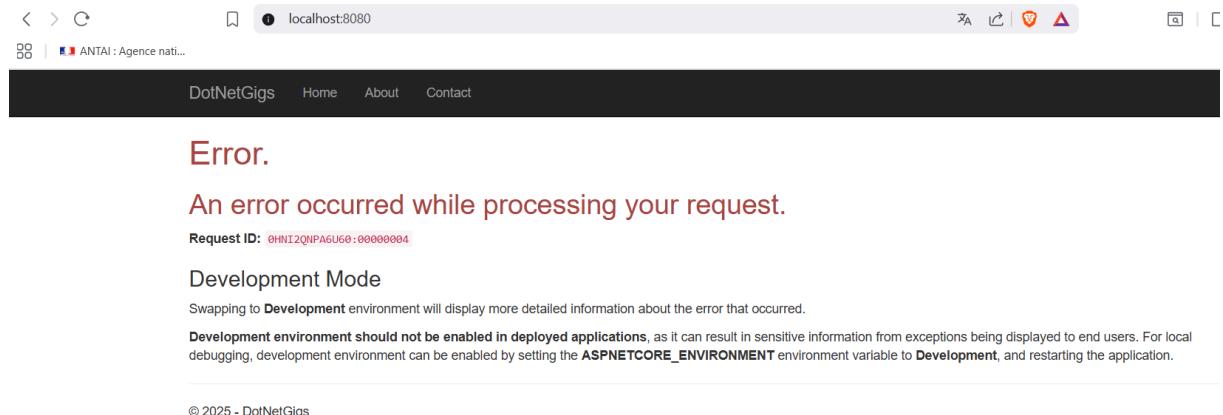
On fait un port-forward (temporaire).

```
kubectl -n tp-prod port-forward svc/webmvc 8080:80
```

```
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl -n tp-prod port-forward svc/webmvc 8080:80
Forwarding from 127.0.0.1:8080 -> 80
```

Garde cette commande ouverte et teste dans ton navigateur :

- <http://localhost:8080>



Ensuite stop le port-forward :

- CTRL + C

Si le web charge, on est bons.

NB : à partir de l'étape suivante j'ai eu énormément d'erreur que je n'afficherai pas afin d'avoir un tutoriel propre et reproductible par tous.

Étape 3.11 — Ajouter l'affinité demandée (webmvc proche des APIs)

On met une **preferred podAffinity** (souple, pas bloquante).

```
@'
```

```
{
```

```
  "spec": {
```

```
"template":{  
    "spec":{  
        "affinity":{  
            "podAffinity":{  
                "preferredDuringSchedulingIgnoredDuringExecution": [  
                    {  
                        "weight": 80,  
                        "podAffinityTerm":{  
                            "labelSelector":{ "matchLabels":{ "app": "jobs-api" } },  
                            "topologyKey": "kubernetes.io/hostname"  
                        }  
,  
                    {  
                        "weight": 80,  
                        "podAffinityTerm":{  
                            "labelSelector":{ "matchLabels":{ "app": "applicants-api" } },  
                            "topologyKey": "kubernetes.io/hostname"  
                        }  
,  
                ]  
            }  
        }  
    }  
}  
}@ | Set-Content -Path patch.json -Encoding ascii
```

```

1  {
2    "spec": {
3      "template": {
4        "spec": {
5          "affinity": {
6            "podAffinity": {
7              "preferredDuringSchedulingIgnoredDuringExecution": [
8                {
9                  "weight": 80,
10                 "podAffinityTerm": {
11                   "labelSelector": { "matchLabels": { "app": "jobs-api" } },
12                   "topologyKey": "kubernetes.io/hostname"
13                 }
14               },
15               {
16                 "weight": 80,
17                 "podAffinityTerm": {
18                   "labelSelector": { "matchLabels": { "app": "applicants-api" } },
19                   "topologyKey": "kubernetes.io/hostname"
20                 }
21               }
22             ]
23           }
24         }
25       }
26     }
27   }
28 }
```

Appliquer le patch

Une fois le fichier créé :

```
kubectl -n tp-prod patch deployment webmvc --type merge --patch-file .\patch.json
```

Vérifier que l'affinity est bien appliquée

```
kubectl -n tp-prod get deployment webmvc -o yaml | Select-String affinity -Context 0,30
```

```

1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: webmvc
5    namespace: tp-prod
6    labels:
7      app: webmvc
8  spec:
9    replicas: 2
10   selector:
11     matchLabels:
12       app: webmvc
13   template:
14     metadata:
15       labels:
16         app: webmvc
17     spec:
18       affinity:
19         podAffinity:
20           preferredDuringSchedulingIgnoredDuringExecution:
21             - weight: 80
22               podAffinityTerm:
23                 labelSelector:
24                   matchLabels:
25                     app: jobs-api
26                     topologyKey: kubernetes.io/hostname
27             - weight: 80
28               podAffinityTerm:
29                 labelSelector:
30                   matchLabels:
31                     app: applicants-api
32                     topologyKey: kubernetes.io/hostname
33     containers:
34       - name: webmvc
35         image: tpfinal.azurecr.io/webmvc:1.0.0
36         imagePullPolicy: IfNotPresent
```

La ligne 18 à 20 :

affinity.podAffinity.preferredDuringSchedulingIgnoredDuringExecution prouve que le patch a bien été appliqué, les affinités sont visible à la ligne 25 et 31 (jobs-api-applicants-api).

Vérifier que ça a bien un effet sur le scheduling (optionnel mais utile)

1) Vérifier que les pods cibles existent et ont bien les labels

```
kubectl -n tp-prod get pods -l app=jobs-api -o wide
```

```
kubectl -n tp-prod get pods -l app=applicants-api -o wide
```

```
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl -n tp-prod get pods -l app=jobs-api -o wide
NAME          READY   STATUS    RESTARTS   AGE     IP           NODE      NOMINATED NODE   READINESS GATES
jobs-api-575578675-bz6v6  1/1    Running   0          35h    10.1.0.111  docker-desktop <none>        <none>
jobs-api-575578675-lwbwp  1/1    Running   0          35h    10.1.0.110  docker-desktop <none>        <none>
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl -n tp-prod get pods -l app=applicants-api -o wide
NAME          READY   STATUS    RESTARTS   AGE     IP           NODE      NOMINATED NODE   READINESS GATES
applicants-api-698c5b59df-lkv6s  1/1    Running   0          35h    10.1.0.109  docker-desktop <none>        <none>
applicants-api-698c5b59df-sk9n4  1/1    Running   0          35h    10.1.0.108  docker-desktop <none>        <none>
```

2) Voir où sont les pods webmvc (sur quels nodes)

```
kubectl -n tp-prod get pods -l app=webmvc -o wide
```

3) Forcer un redeploy pour que le scheduler “reconsidère” la règle

Le patch change le template → normalement ça déclenche déjà un rollout, mais tu peux forcer :

```
kubectl -n tp-prod rollout restart deployment webmvc
```

```
kubectl -n tp-prod rollout status deployment webmvc
```

```
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl -n tp-prod rollout restart deployment webmvc
deployment.apps/webmvc restarted
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl -n tp-prod rollout status deployment webmvc
Waiting for deployment "webmvc" rollout to finish: 1 out of 2 new replicas have been updated...
Waiting for deployment "webmvc" rollout to finish: 1 out of 2 new replicas have been updated...
Waiting for deployment "webmvc" rollout to finish: 1 out of 2 new replicas have been updated...
Waiting for deployment "webmvc" rollout to finish: 1 old replicas are pending termination...
Waiting for deployment "webmvc" rollout to finish: 1 old replicas are pending termination...
deployment "webmvc" successfully rolled out
```

3.11.2 — Mettre à jour ton fichier webmvc.yaml avec l'affinité

On remplace ton fichier par une version complète (avec l'affinité intégrée).

```
@"
```

```
apiVersion: apps/v1
```

```
kind: Deployment
```

```
metadata:
```

```
name: webmvc
namespace: tp-prod
labels:
  app: webmvc
spec:
  replicas: 2
  selector:
    matchLabels:
      app: webmvc
  template:
    metadata:
      labels:
        app: webmvc
    spec:
      affinity:
        podAffinity:
          preferredDuringSchedulingIgnoredDuringExecution:
            - weight: 80
              podAffinityTerm:
                labelSelector:
                  matchLabels:
                    app: jobs-api
                topologyKey: kubernetes.io/hostname
            - weight: 80
              podAffinityTerm:
                labelSelector:
                  matchLabels:
                    app: applicants-api
```

```
    topologyKey: kubernetes.io/hostname

  containers:
    - name: webmvc

      image: tpfinal.azurecr.io/webmvc:1.0.0
      imagePullPolicy: IfNotPresent

    ports:
      - containerPort: 80
        name: http

    resources:
      requests:
        cpu: "4m"
        memory: "900Mi"

      limits:
        cpu: "1"
        memory: "2000Mi"

    readinessProbe:
      tcpSocket:
        port: 80
      initialDelaySeconds: 10
      periodSeconds: 5

    livenessProbe:
      tcpSocket:
        port: 80
      initialDelaySeconds: 20
      periodSeconds: 10

  ---
  apiVersion: v1
  kind: Service
```

```
metadata:  
  name: webmvc  
  namespace: tp-prod  
  labels:  
    app: webmvc  
spec:  
  type: ClusterIP  
  selector:  
    app: webmvc  
  ports:  
    - name: http  
      port: 80  
      targetPort: 80  
"@ | Set-Content -Encoding UTF8 .\03-k8s-manifests\apps\webmvc.yaml
```

3.11.3 — Re-apply depuis fichier (propre)

```
kubectl apply -f .\03-k8s-manifests\apps\webmvc.yaml
```

3.11.4 — Commit

```
git add .\03-k8s-manifests\apps\webmvc.yaml
```

```
git commit -m "(k8s): déploiement de webmvc et des affinités aux APIs"
```

Étape 3.11 (vérif affinité après màj du fichier) — commande fiable

3.11.5 — Vérifier l'affinité dans le YAML du Deployment

Exécute :

```
kubectl -n tp-prod get deploy webmvc -o yaml | Select-String -Pattern  
"affinity|podAffinity|preferredDuringSchedulingIgnoredDuringExecution" -Context 0,6
```

 Tu dois voir des lignes avec affinity: puis podAffinity: etc.

```

PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl -n tp-prod get deploy webmvc -o yaml | Select-String -Pattern "affinity|podAffinity|preferredDuringSchedulingIgnoredDuringExecution" -Context 0,6
>     {"apiVersion": "apps/v1", "kind": "Deployment", "metadata": {"annotations": {}, "labels": {"app": "webmvc"}, "name": "webmvc", "namespace": "tp-prod"}, "spec": {"replicas": 2, "selector": {"matchLabels": {"app": "webmvc"}}, "template": {"metadata": {"labels": {"app": "webmvc"}}, "spec": {"affinity": {"podAffinity": {"preferredDuringSchedulingIgnoredDuringExecution": [{"podAffinityTerm": {"labelSelector": {"matchLabels": {"app": "jobs-api"}}, "topologyKey": "kubernetes.io/hostname"}, "weight": 80}], "podAffinityTerm": {"labelSelector": {"matchLabels": {"app": "applicants-api"}}, "topologyKey": "kubernetes.io/hostname"}, "weight": 80}}}, "containers": [{"image": "tpfinal.azurecr.io/webmvc:1.0.0", "imagePullPolicy": "IfNotPresent", "livenessProbe": {"initialDelaySeconds": 20, "periodSeconds": 10, "tcpSocket": {"port": 80}}, "name": "webmvc", "ports": [{"containerPort": 80, "name": "http"}], "readinessProbe": {"initialDelaySeconds": 10, "periodSeconds": 5, "tcpSocket": {"port": 80}}, "resources": {"limits": {"cpu": "1", "memory": "2000Mi"}, "requests": {"cpu": "4m", "memory": "900Mi"}}}]}, "creationTimestamp": "2025-12-24T10:18:43Z", "generation": 14, "labels": {"app": "webmvc", "name": "webmvc", "namespace": "tp-prod"}, "affinity": {"podAffinity": {"preferredDuringSchedulingIgnoredDuringExecution": [{"podAffinityTerm": {"labelSelector": {"matchLabels": {"app": "jobs-api"}, "topologyKey": "kubernetes.io/hostname"}, "weight": 80}, {"podAffinityTerm": {"labelSelector": {"matchLabels": {"app": "applicants-api"}, "topologyKey": "kubernetes.io/hostname"}, "weight": 80}}]}}, "containers": []

```

Au cas où :

Si l'affinité n'apparaît pas : correction (apply from file)

Comme le fichier 03-k8s-manifests\apps\webmvc.yaml contient déjà l'affinité, refais juste :

kubectl apply -f .\03-k8s-manifests\apps\webmvc.yaml

kubectl -n tp-prod rollout restart deploy/webmvc

kubectl -n tp-prod rollout status deploy/webmvc

```

PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl -n tp-prod rollout restart deployment webmvc deployment.apps/webmvc restarted
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl -n tp-prod rollout status deployment webmvc
Waiting for deployment "webmvc" rollout to finish: 1 out of 2 new replicas have been updated...
Waiting for deployment "webmvc" rollout to finish: 1 out of 2 new replicas have been updated...
Waiting for deployment "webmvc" rollout to finish: 1 out of 2 new replicas have been updated...
Waiting for deployment "webmvc" rollout to finish: 1 old replicas are pending termination...
Waiting for deployment "webmvc" rollout to finish: 1 old replicas are pending termination...
deployment "webmvc" successfully rolled out

```

Puis re-teste la commande de vérif (kubectl -n tp-prod get deploy webmvc -o yaml |

Select-String -Pattern

"affinity|podAffinity|preferredDuringSchedulingIgnoredDuringExecution" -Context 0,6).

Exposition HTTPS via Ingress (cert auto-signé)

Sur Docker Desktop, le plus simple est **ingress-nginx** + secret TLS auto-signé.

4.1 — Installer ingress-nginx

A) Ajouter le repo Helm ingress-nginx

helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx

helm repo update

B) Installer ingress-nginx (namespace dédié)

```
helm install ingress-nginx ingress-nginx/ingress-nginx  
--namespace ingress-nginx  
--create-namespace
```

C) Vérifier

```
kubectl -n ingress-nginx get pods
```

```
kubectl -n ingress-nginx get svc
```

```
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl -n ingress-nginx get pods  
NAME           READY   STATUS    RESTARTS   AGE  
ingress-nginx-controller-6c657c6487-b2mmr  1/1     Running   0          34h  
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl -n ingress-nginx get svc  
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE  
ingress-nginx-controller   LoadBalancer   10.99.57.136  localhost     80:32190/TCP,443:31103/TCP   34h  
ingress-nginx-controller-admission   ClusterIP   10.98.44.58    <none>       443/TCP   34h
```

NB : Ici j'ai du apprendre à installer openssl sous powershell, j'ai fais le choix de passer l'installation car ce n'est pas le sujet du tp (pensez à ajouter openssl aux variables d'environnement système).

4.2 — Générer un certificat auto-signé + secret TLS

On va faire un cert pour un hostname local (ex: tp.local).

Sur Windows, le plus simple est openssl.

```
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> openssl version  
OpenSSL 3.6.0 1 Oct 2025 (Library: OpenSSL 3.6.0 1 Oct 2025)
```

A) Créer un dossier certs

```
mkdir .\03-k8s-manifests\security\certs
```

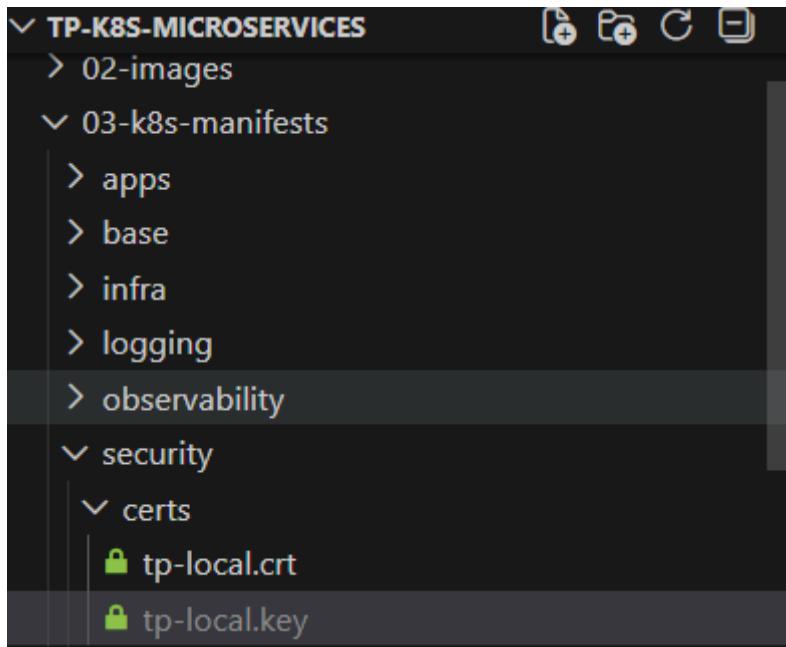
B) Générer cert + key (avec openssl)

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048  
-keyout .\03-k8s-manifests\security\certs\tp-local.key  
-out .\03-k8s-manifests\security\certs\tp-local.crt  
-subj "/CN=tp.local/O=tp"
```

📁 Résultat :

tp-local.key → clé privée

tp-local.crt → certificat auto-signé



C) Créer le secret TLS dans tp-prod

```
kubectl -n tp-prod create secret tls tp-local-tls `  
--cert .\03-k8s-manifests\security\certs\tp-local.crt `  
--key .\03-k8s-manifests\security\certs\tp-local.key
```

Vérif :

```
kubectl -n tp-prod get secret tp-local-tls
```

```
PS C:\Users\Dimитри ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl -n tp-prod get secret tp-local-tls  
NAME          TYPE        DATA   AGE  
tp-local-tls  kubernetes.io/tls  2      34h
```

💡 Vérification rapide des fichiers générés

```
ls .\03-k8s-manifests\security\certs\
```

```
PS C:\Users\Dimитри ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> ls .\03-k8s-manifests\security\certs\  
Répertoire : C:\Users\Dimитри ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\03-k8s-manifests\security\certs  
  
Mode                LastWriteTime         Length Name  
----                -----         ----  --  
-a---    24/12/2025     12:39           1166 tp-local.crt  
-a---    24/12/2025     12:39          1732 tp-local.key
```

La commande openssl x509 -text -noout permet d'inspecter un certificat TLS afin de vérifier son identité, sa validité, ses extensions et sa conformité avec la configuration Ingress Kubernetes

```
openssl x509 -in .\03-k8s-manifests\security\certs\tp-local.crt -text -noout
```

1 Décomposition simple

openssl

👉 outil standard pour la **cryptographie**, les certificats TLS, le chiffrement

x509

👉 indique que tu travailles sur un **certificat X.509**
(c'est le format standard des certificats HTTPS)

-in .\03-k8s-manifests\security\certs\tp-local.crt

👉 chemin vers le **certificat à analyser**

- ici : tp-local.crt
 - fichier public (⚠ pas la clé privée)
-

-text

👉 affiche **TOUTES les informations détaillées**

- lisible par un humain
 - pas encodé en base64
-

-noout

👉 **n'affiche pas le certificat brut**

- évite d'afficher le contenu encodé
 - affiche uniquement les métadonnées
-

2 Ce que tu vois quand tu l'exécutes

Typiquement, tu verras :

Identité

- **Subject** → à qui appartient le certificat
 - **Issuer** → qui l'a signé (CA)
 - **Common Name (CN)** → nom de domaine
 - **SAN (Subject Alternative Names)** → domaines couverts
-

Validité

- Not Before
- Not After

 permet de savoir si le certificat est **valide ou expiré**

Clés & algorithmes

- algorithme (RSA / ECDSA)
 - taille de clé (2048 / 4096 bits)
 - hash (SHA256, etc.)
-

Extensions importantes

- Key Usage
 - Extended Key Usage
 - Basic Constraints
 - Subject Alternative Name
-

3 Pourquoi c'est CRUCIAL en Kubernetes / Ingress

Dans ton contexte (Ingress + TLS) cette commande sert à :

Vérifier que :

- le **CN ou SAN correspond bien au host de l'ingress**

- le certificat n'est pas expiré
 - le certificat est bien **X.509**
 - la clé utilisée est correcte
 - tu n'as pas inversé .crt et .key
-

4 Exemple concret avec Ingress

Ingress :

tls:

- hosts:

- tp.local

secretName: tp-local-tls

Certificat :

Subject: CN = tp.local

X509v3 Subject Alternative Name:

DNS:tp.local

👉 Match parfait

👉 TLS fonctionnera

5 Ce que cette commande NE FAIT PAS

- ✗ ne crée pas de certificat
- ✗ ne signe rien
- ✗ ne modifie pas le fichier
- ✗ n'installe rien dans Kubernetes

🔒 Alternative Kubernetes-native (sans OpenSSL)

Si l'objectif est **uniquement Kubernetes**, on pouvait aussi faire :

```
kubectl create secret tls tp-local-tls `  
  --cert=tp-local.crt`
```

```
--key=tp-local.key`
```

```
-n tp-prod
```

Ou même utiliser **cert-manager** (plus propre en prod).

4.3 — Créer l’Ingress HTTPS pour webmvc

A) Fichier 03-k8s-manifests\security\ingress-webmvc.yaml

```
@"
```

```
apiVersion: networking.k8s.io/v1
```

```
kind: Ingress
```

```
metadata:
```

```
  name: webmvc-ingress
```

```
  namespace: tp-prod
```

```
  annotations:
```

```
    nginx.ingress.kubernetes.io/ssl-redirect: "true"
```

```
spec:
```

```
  ingressClassName: nginx
```

```
  tls:
```

```
    - hosts:
```

```
      - tp.local
```

```
    secretName: tp-local-tls
```

```
  rules:
```

```
    - host: tp.local
```

```
      http:
```

```
        paths:
```

```
          - path: /
```

```
            pathType: Prefix
```

```
          backend:
```

```
            service:
```

```
name: webmvc
```

```
port:
```

```
    number: 80
```

```
"@ | Set-Content -Encoding UTF8 .\03-k8s-manifests\security\ingress-webmvc.yaml
```

B) Apply

```
kubectl apply -f .\03-k8s-manifests\security\ingress-webmvc.yaml
```

C) Vérifier

```
kubectl -n tp-prod get ingress
```

```
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl -n tp-prod get ingress
NAME      CLASS   HOSTS   ADDRESS   PORTS   AGE
webmvc-ingress  nginx  tp.local  localhost  80, 443  34h
```

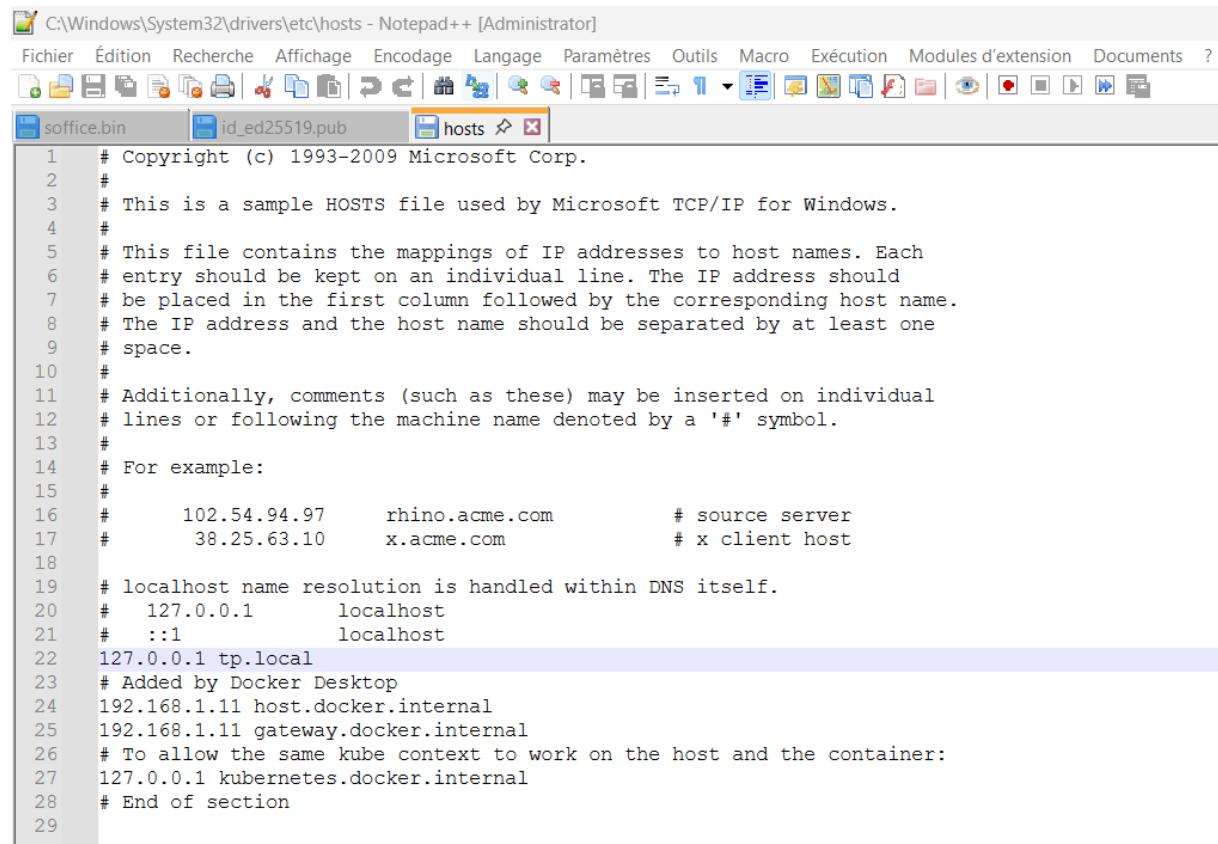
4.4 — Résolution DNS locale (Windows hosts)

Pour accéder à https://tp.local, ajoute dans ton fichier hosts :

1. Ouvre Notepad en admin, édite :
C:\Windows\System32\drivers\etc\hosts

2. Ajoute :

127.0.0.1 tp.local



The screenshot shows the Windows hosts file in Notepad++. The file contains comments about the format and examples of entries. A new entry '127.0.0.1 tp.local' is added at the bottom of the list.

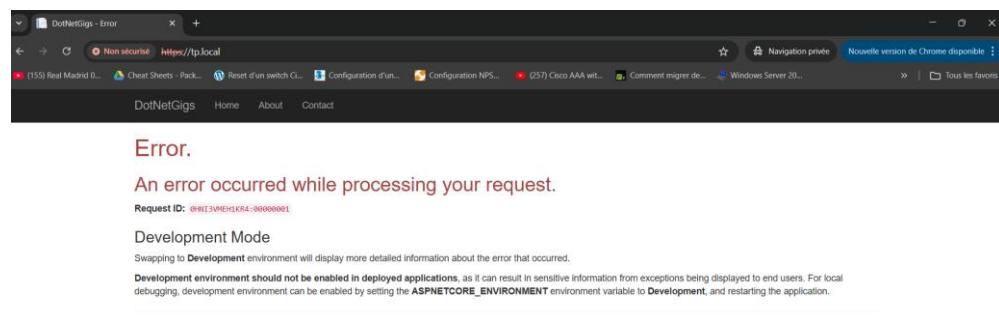
```
# Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#      102.54.94.97    rhino.acme.com        # source server
#      38.25.63.10    x.acme.com            # x client host
#
# localhost name resolution is handled within DNS itself.
# 127.0.0.1      localhost
# ::1            localhost
127.0.0.1 tp.local
# Added by Docker Desktop
192.168.1.11 host.docker.internal
192.168.1.11 gateway.docker.internal
# To allow the same kube context to work on the host and the container:
127.0.0.1 kubernetes.docker.internal
# End of section
```

4.5 — Test HTTPS

Dans ton navigateur :

- https://tp.local

⚠️ Le cert est auto-signé → tu auras un avertissement, c'est normal.



4.6 — Commit des manifests (sans committer la clé privée)

⚠ Recommandation “pro” : **ne commit pas** tp-local.key.

Ajoute ça dans .gitignore :

```
Add-Content -Encoding UTF8 '.\gitignore "`n# TLS keys`n03-k8s-manifests/security/certs/*.key`n"
```

Commit :

```
git add .\03-k8s-manifests\security\ingress-webmvc.yaml
```

```
git add .\03-k8s-manifests\security\certs\tp-local.crt
```

```
git add .\.gitignore
```

```
git commit -m "(k8s): ajout de ingress-nginx et de l'ingress https avec un certificat auto-signé (openssl)"
```

On valide que ingress-nginx est OK (LoadBalancer sur localhost)

```
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl -n ingress-nginx get svc
NAME           TYPE        CLUSTER-IP      EXTERNAL-IP    PORT(S)          AGE
ingress-nginx-controller   LoadBalancer   10.99.57.136  localhost     80:32190/TCP,443:31103/TCP  35h
ingress-nginx-controller-admission ClusterIP  10.98.44.58   <none>        443/TCP         35h
```

on continue avec NetworkPolicy + ServiceAccount/RBAC, puis HPA + Metrics Server, puis Prometheus/kube-state-metrics, puis EFK/ELK, puis Helm

1) ServiceAccount + RBAC (sur identity-api)

1.1 — Créer le manifest RBAC

Crée ce fichier : 03-k8s-manifests\security\identity-rbac.yaml

```
@"
```

```
apiVersion: v1
```

```
kind: ServiceAccount
```

```
metadata:
```

```
  name: sa-identity
```

```
  namespace: tp-prod
```

```
  labels:
```

```
    app: identity-api
```

```
---
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: role-identity-read
  namespace: tp-prod
rules:
  - apiGroups: [""]
    resources: ["configmaps"]
    verbs: ["get", "list"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: rb-identity-read
  namespace: tp-prod
subjects:
  - kind: ServiceAccount
    name: sa-identity
    namespace: tp-prod
roleRef:
  kind: Role
  name: role-identity-read
  apiGroup: rbac.authorization.k8s.io
"@ | Set-Content -Encoding UTF8 .\03-k8s-manifests\security\identity-rbac.yaml
```

1.2 — Appliquer

```
kubectl apply -f .\03-k8s-manifests\security\identity-rbac.yaml
```

1.3 — Associer le ServiceAccount au Deployment identity-api

- 1) Crée le fichier

```

@'

{
  "spec": {
    "template": {
      "spec": {
        "serviceAccountName": "sa-identity"
      }
    }
  }
}

'@ | Set-Content -Path .\patch-identity.json -Encoding ascii

```

2) Applique le patch

```
kubectl -n tp-prod patch deployment identity-api --type merge --patch-file .\patch-identity.json
```

1.4 — Vérifier

```
kubectl -n tp-prod get deploy identity-api -o
jsonpath="{.spec.template.spec.serviceAccountName}\n"
```

```
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl -n tp-prod get deploy identity-api -o jsonpath=".spec.template.spec.serviceAccountName\n"
sa-identity
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> |
```

1.5 — Commit

```
git add .\03-k8s-manifests\security\identity-rbac.yaml
```

```
git commit -m "(securite): Compte de service dedie + RBAC pour identity-api"
```

2) NetworkPolicy (limiter les communications)

⚠ Note : sur certains clusters “locaux”, l’application des NetworkPolicy dépend du plugin réseau. Même si l’enforcement n’est pas actif, on remplit la demande du TP en **les définissant** (et sur AKS ça fonctionne).

On va faire :

- **default deny** (ingress + egress)
- - policies d’autorisation “juste ce qu’il faut”
- - autoriser DNS vers kube-dns

2.1 — Default deny (tp-prod)

Fichier : 03-k8s-manifests\security\netpol-default-deny.yaml

@"

```
apiVersion: networking.k8s.io/v1
```

```
kind: NetworkPolicy
```

```
metadata:
```

```
  name: default-deny-all
```

```
  namespace: tp-prod
```

```
spec:
```

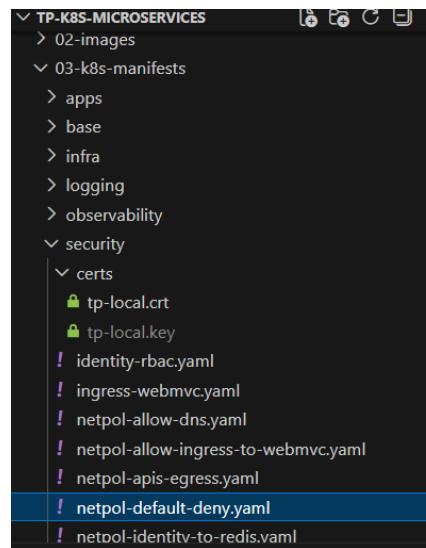
```
  podSelector: {}
```

```
  policyTypes:
```

```
    - Ingress
```

```
    - Egress
```

"@ | Set-Content -Encoding UTF8 .\03-k8s-manifests\security\netpol-default-deny.yaml



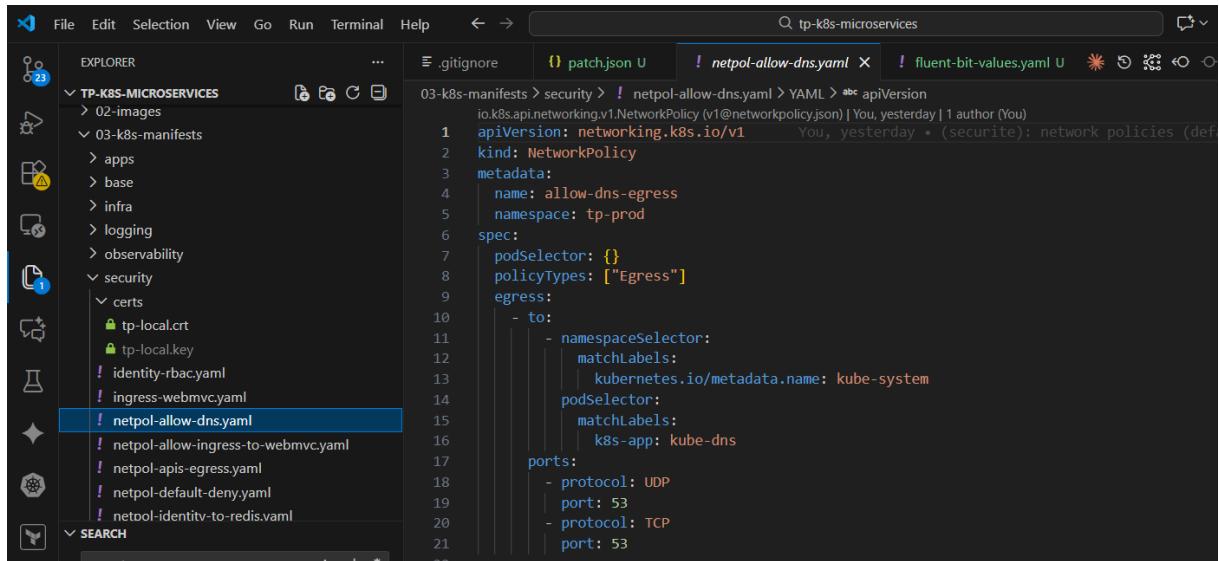
```
03-k8s-manifests > security > ! netpol-default-deny.yaml > YAML > abc apiVersion
io.k8s.api.networking.v1.NetworkPolicy (v1@networkpolicy.json) | You, yesterday | 1 author (You)
1  apiVersion: networking.k8s.io/v1      You, yesterday • (securite): r
2  kind: NetworkPolicy
3  metadata:
4    name: default-deny-all
5    namespace: tp-prod
6  spec:
7    podSelector: {}
8    policyTypes:
9      - Ingress
10     - Egress
11
```

Autoriser DNS (obligatoire sinon tout casse)

Fichier : 03-k8s-manifests\security\netpol-allow-dns.yaml

@"

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-dns-egress
  namespace: tp-prod
spec:
  podSelector: {}
  policyTypes: ["Egress"]
  egress:
    - to:
        - namespaceSelector:
            matchLabels:
              kubernetes.io/metadata.name: kube-system
        podSelector:
          matchLabels:
            k8s-app: kube-dns
  ports:
    - protocol: UDP
      port: 53
    - protocol: TCP
      port: 53
"@ | Set-Content -Encoding UTF8 .\03-k8s-manifests\security\netpol-allow-dns.yaml
```



```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-dns-egress
  namespace: tp-prod
spec:
  podSelector: {}
  policyTypes: ["Egress"]
  egress:
    - to:
        - namespaceSelector:
            matchLabels:
              kubernetes.io/metadata.name: kube-system
        podSelector:
            matchLabels:
              k8s-app: kube-dns
      ports:
        - protocol: UDP
          port: 53
        - protocol: TCP
          port: 53
```

2.3 — Autoriser Ingress vers webmvc depuis ingress-nginx

Fichier : 03-k8s-manifests\security\netpol-allow-ingress-to-webmvc.yaml

@"

apiVersion: networking.k8s.io/v1

kind: NetworkPolicy

metadata:

name: allow-ingress-to-webmvc

namespace: tp-prod

spec:

podSelector:

matchLabels:

app: webmvc

policyTypes: ["Ingress"]

ingress:

- from:

- namespaceSelector:

matchLabels:

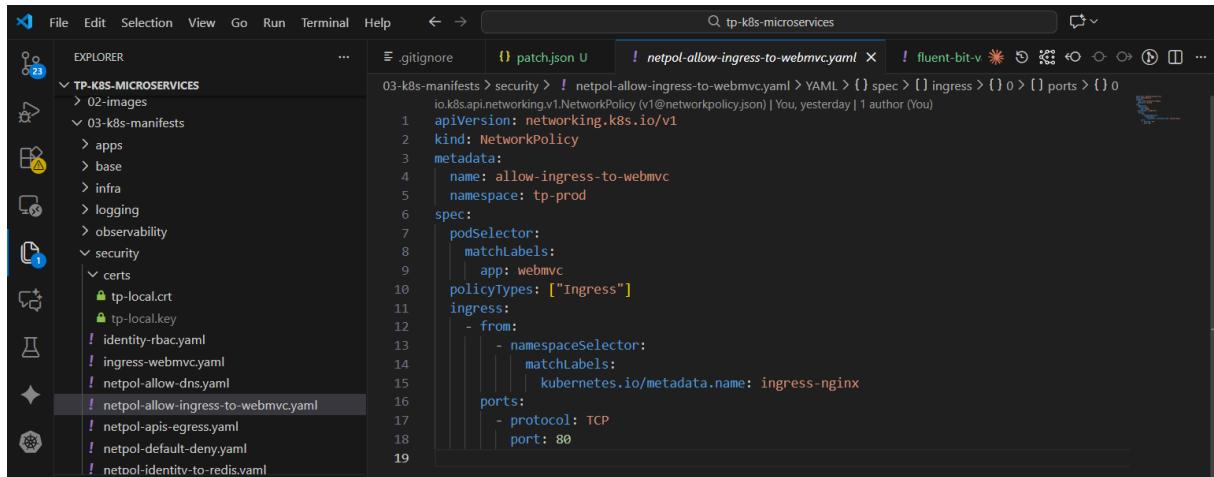
kubernetes.io/metadata.name: ingress-nginx

ports:

- protocol: TCP

port: 80

"@ | Set-Content -Encoding UTF8 .\03-k8s-manifests\security\netpol-allow-ingress-to-webmvc.yaml



```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-ingress-to-webmvc
  namespace: tp-prod
spec:
  podSelector:
    matchLabels:
      app: webmvc
  policyTypes: [ "ingress" ]
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              kubernetes.io/metadata.name: ingress-nginx
      ports:
        - protocol: TCP
          port: 80
```

2.4 — Autoriser web → APIs (80)

Fichier : 03-k8s-manifests\security\netpol-web-to-apis.yaml

@"

apiVersion: networking.k8s.io/v1

kind: NetworkPolicy

metadata:

name: allow-web-to-apis

namespace: tp-prod

spec:

podSelector:

matchExpressions:

- key: app

operator: In

values: ["applicants-api","jobs-api","identity-api"]

```
policyTypes: ["Ingress"]
```

```
ingress:
```

```
- from:
```

```
  - podSelector:
```

```
    matchLabels:
```

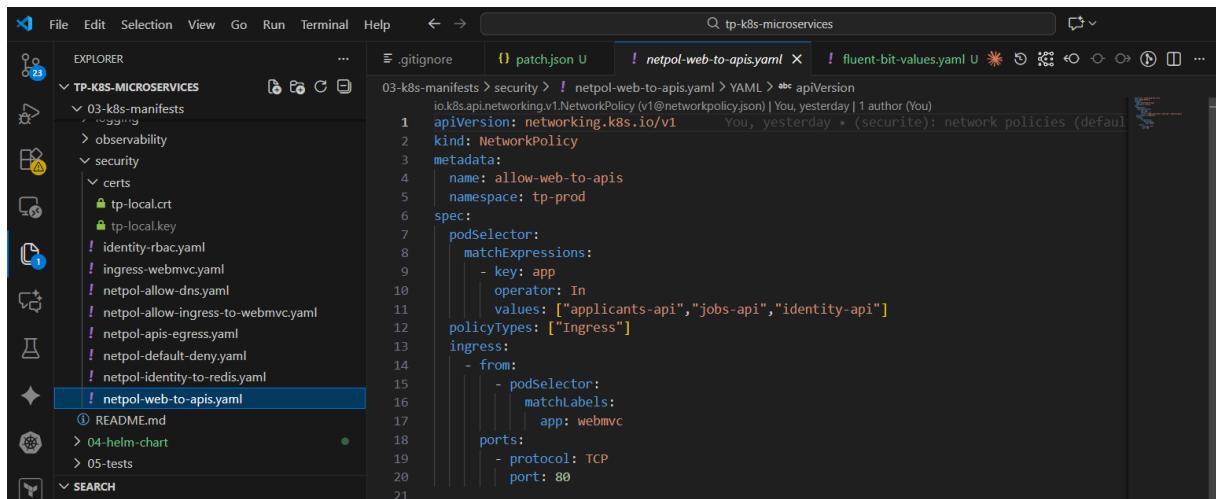
```
      app: webmvc
```

```
ports:
```

```
  - protocol: TCP
```

```
    port: 80
```

```
"@ | Set-Content -Encoding UTF8 .\03-k8s-manifests\security\netpol-web-to-apis.yaml
```



```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-web-to-apis
  namespace: tp-prod
spec:
  podSelector:
    matchExpressions:
      - key: app
        operator: In
        values: ["applicants-api","jobs-api","identity-api"]
  policyTypes: ["Ingress"]
  ingress:
    - from:
        - podSelector:
            matchLabels:
              app: webmvc
        ports:
          - protocol: TCP
          port: 80
```

2.5 — Autoriser APIs → sql-data + rabbitmq

Fichier : 03-k8s-manifests\security\netpol-apis-egress.yaml

```
@"
```

```
apiVersion: networking.k8s.io/v1
```

```
kind: NetworkPolicy
```

```
metadata:
```

```
  name: allow-apis-egress
```

```
  namespace: tp-prod
```

```
spec:  
  podSelector:  
    matchExpressions:  
      - key: app  
        operator: In  
        values: ["applicants-api","jobs-api","identity-api"]  
  policyTypes: ["Egress"]  
  
  egress:  
    - to:  
      - podSelector:  
          matchLabels:  
            app: sql-data  
  
      ports:  
        - protocol: TCP  
          port: 1433  
  
        - to:  
          - podSelector:  
              matchLabels:  
                app: rabbitmq  
  
      ports:  
        - protocol: TCP  
          port: 5672  
  
"@ | Set-Content -Encoding UTF8 .\03-k8s-manifests\security\netpol-apis-egress.yaml
```

```

apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-apis-egress
  namespace: tp-prod
spec:
  podSelector:
    matchExpressions:
      - key: app
        operator: In
        values: ["identity-api"]
  policyTypes: [ "Egress" ]
  egress:
    - to:
        - podSelector:
            matchLabels:
              app: sql-data
        ports:
          - protocol: TCP
            port: 1433
    - to:
        - podSelector:
            matchLabels:
              app: rabbitmq
        ports:
          - protocol: TCP
            port: 5672

```

2.6 — Autoriser identity → user-data (redis)

Fichier : 03-k8s-manifests\security\netpol-identity-to-redis.yaml

```

@"

apiVersion: networking.k8s.io/v1

kind: NetworkPolicy

metadata:

  name: allow-identity-to-redis

  namespace: tp-prod

spec:

  podSelector:

    matchLabels:

      app: identity-api

  policyTypes: ["Egress"]

  egress:

    - to:

        - podSelector:

            matchLabels:

              app: redis

```

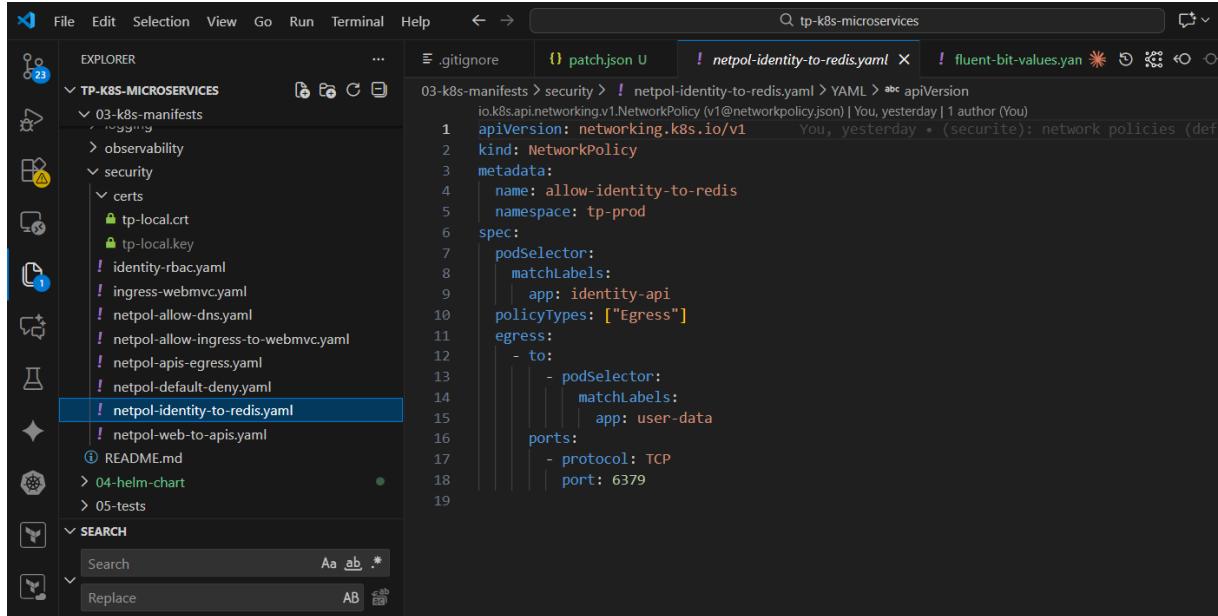
app: user-data

ports:

- protocol: TCP

- port: 6379

```
"@ | Set-Content -Encoding UTF8 .\03-k8s-manifests\security\netpol-identity-to-redis.yaml
```



```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-identity-to-redis
  namespace: tp-prod
spec:
  podSelector:
    matchLabels:
      app: identity-api
  policyTypes: ["Egress"]
  egress:
  - to:
    - podSelector:
        matchLabels:
          app: user-data
    ports:
    - protocol: TCP
      port: 6379
```

2.7 — Appliquer toutes les “policies”

```
kubectl apply -f .\03-k8s-manifests\security\netpol-default-deny.yaml
```

```
kubectl apply -f .\03-k8s-manifests\security\netpol-allow-dns.yaml
```

```
kubectl apply -f .\03-k8s-manifests\security\netpol-allow-ingress-to-webmvc.yaml
```

```
kubectl apply -f .\03-k8s-manifests\security\netpol-web-to-apis.yaml
```

```
kubectl apply -f .\03-k8s-manifests\security\netpol-apis-egress.yaml
```

```
kubectl apply -f .\03-k8s-manifests\security\netpol-identity-to-redis.yaml
```

2.8 — Vérifier

```
kubectl -n tp-prod get networkpolicy
```

```
PS C:\Users\Dimi ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl -n tp-prod get networkpolicy
NAME                POD-SELECTOR           AGE
allow-apis-egress   app in (applicants-api,identity-api,jobs-api)  34h
allow-dns-egress    <none>                  34h
allow-identity-to-redis  app=identity-api  34h
allow-ingress-to-webmvc  app=webmvc       34h
allow-web-to-apis   app in (applicants-api,identity-api,jobs-api)  34h
default-deny-all    <none>                  34h
```

2.9 — Commit

```
git add .\03-k8s-manifests\security\netpol-*yaml  
git commit -m "(securite): network policies (default deny + allowed flows)"
```

3) Metrics Server + HPA

3.1 — Vérifier si metrics-server est déjà là

```
kubectl -n kube-system get deploy metrics-server
```

```
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl -n kube-system get deploy metrics-server  
NAME      READY   UP-TO-DATE   AVAILABLE   AGE  
metrics-server   1/1     1          1           34h
```

3.2 — Vérifier que kubectl top marche

Attends ~30s puis :

```
kubectl top nodes
```

```
kubectl -n tp-prod top pods
```

```
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl top nodes  
NAME      CPU(cores)   CPU(%)    MEMORY(bytes)   MEMORY(%)  
docker-desktop   2002m       10%      11036Mi       34%  
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl -n tp-prod top pods  
NAME      CPU(cores)   MEMORY(bytes)  
applicants-api-698c5b59df-lkv6s   2m        213Mi  
applicants-api-698c5b59df-sk9n4   2m        199Mi  
identity-api-86dc4dd8-j8nmx      3m        106Mi  
identity-api-86dc4dd8-wk8dq      3m        105Mi  
jobs-api-575578675-bz6v6       2m        226Mi  
jobs-api-575578675-lwbwp       2m        164Mi  
rabbitmq-78ccfb5c7b-66q7h      5m        205Mi  
sql-data-0                      13m       1969Mi  
user-data-7b6897b6c7-jf8bm      3m        7Mi  
webmvc-77c9686847-rgs8d       1m        52Mi  
webmvc-77c9686847-zrmbh       1m        46Mi
```

3.3 — Crée un HPA sur webmvc (CPU 60%)

Fichier : 03-k8s-manifests\apps\webmvc-hpa.yaml

```
@"
```

```
apiVersion: autoscaling/v2
```

```
kind: HorizontalPodAutoscaler
```

```
metadata:
```

```
  name: webmvc-hpa
```

```
  namespace: tp-prod
```

```
spec:
```

```
  scaleTargetRef:
```

```
    apiVersion: apps/v1
```

```
kind: Deployment
```

```
  name: webmvc
```

```
  minReplicas: 2
```

```
  maxReplicas: 6
```

```
  metrics:
```

```
    - type: Resource
```

```
      resource:
```

```
        name: cpu
```

```
      target:
```

```
        type: Utilization
```

```
        averageUtilization: 60
```

```
"@ | Set-Content -Encoding UTF8 .\03-k8s-manifests\apps\webmvc-hpa.yaml
```

Apply :

```
kubectl apply -f .\03-k8s-manifests\apps\webmvc-hpa.yaml
```

Vérif :

```
kubectl -n tp-prod get hpa
```

```
PS C:\Users\Dimi ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl -n tp-prod get hpa
NAME      REFERENCE      TARGETS      MINPODS      MAXPODS      REPLICAS      AGE
webmvc-hpa  Deployment/webmvc  cpu: 25%/60%  2           6           2           30h
```

3.4 — Démonstration charge (simple)

On lance un pod qui bombarde webmvc en interne :

```
kubectl -n tp-prod run loadgen --image=busybox:1.36 -it --rm -- sh
```

Dans le shell :

```
while true; do wget -q -O- http://webmvc/ > /dev/null; done
```

```
PS C:\Users\Dimi ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl -n tp-prod run loadgen --image=busybox:1.36 -it --rm -- sh
All commands and output from this session will be recorded in container logs, including credentials and sensitive information passed through the command prompt.
If you don't see a command prompt, try pressing enter.
/ #
/ #
/ # while true; do wget -q -O- http://webmvc/ > /dev/null; done
wget: server returned error: HTTP/1.1 500 Internal Server Error
wget: server returned error: HTTP/1.1 500 Internal Server Error
wget: server returned error: HTTP/1.1 500 Internal Server Error
wget: server returned error: HTTP/1.1 500 Internal Server Error
wget: server returned error: HTTP/1.1 500 Internal Server Error
```

Dans **un autre terminal PowerShell**, observe :

```
kubectl -n tp-prod get hpa -w
```

NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS	AGE
webmvc-hpa	Deployment/webmvc	cpu: 25%/60%	2	6	2	30h
webmvc-hpa	Deployment/webmvc	cpu: 7037%/60%	2	6	2	30h
webmvc-hpa	Deployment/webmvc	cpu: 25%/60%	2	6	4	30h
webmvc-hpa	Deployment/webmvc	cpu: 25%/60%	2	6	6	30h

Quand tu vois les replicas monter, stop loadgen (CTRL+C) puis exit.

3.5 — Commit

```
git add .\03-k8s-manifests\apps\webmvc-hpa.yaml
```

```
git commit -m "(scale): ajout de l'HPA pour webmvc utilisant metrics-server"
```

```
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> git add .\03-k8s-manifests\apps\webmvc-hpa.yaml
warning: in the working copy of '03-k8s-manifests/apps/webmvc-hpa.yaml', LF will be replaced by CRLF the next time Git touches it
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> git commit -m "(scale): ajout de l'HPA pour webmvc utilisant metrics-server"
[main 85f7526] (scale): ajout de l'HPA pour webmvc utilisant metrics-server
 1 file changed, 19 insertions(+)
 create mode 100644 03-k8s-manifests/apps/webmvc-hpa.yaml
```

4) Prometheus + kube-state-metrics (kube-prometheus-stack)

On va installer via Helm : kube-prometheus-stack (inclut Prometheus + Grafana + kube-state-metrics). [artifacthub.io+1](#)

```
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
"prometheus-community" has been added to your repositories
```

4.1 — Ajouter repo Helm Prometheus Community

```
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
```

```
helm repo update
```

```
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
"prometheus-community" has been added to your repositories
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "ingress-nginx" chart repository
...Successfully got an update from the "prometheus-community" chart repository
Update Complete. ⚡Happy Helming!⚡
```

4.2 — Installer le stack dans monitoring

```
kubectl create namespace monitoring
```

```
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl create namespace monitoring
namespace/monitoring created
```

```
helm install monitoring prometheus-community/kube-prometheus-stack `
```

```
--namespace monitoring
```

```

PS C:\Users\Dimi ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> helm install monitoring prometheus-community/kube-prometheus-stack \
--namespace monitoring
level=warn msg="unable to find exact version; falling back to closest available version" chart=kube-prometheus-stack requested="" selected=80.6.0
I1224 17:43:47.140923 33172 warnings.go:110] "Warning: spec.SessionAffinity is ignored for headless services"
I1224 17:43:47.140923 33172 warnings.go:110] "Warning: spec.SessionAffinity is ignored for headless services"
I1224 17:43:47.140923 33172 warnings.go:110] "Warning: spec.SessionAffinity is ignored for headless services"
I1224 17:43:47.140923 33172 warnings.go:110] "Warning: spec.SessionAffinity is ignored for headless services"
I1224 17:43:47.140923 33172 warnings.go:110] "Warning: spec.SessionAffinity is ignored for headless services"
NAME: monitoring
LAST DEPLOYED: Wed Dec 24 17:43:38 2025
NAMESPACE: monitoring
STATUS: deployed
REVISION: 1
DESCRIPTION: Install complete
NOTES:
kube-prometheus-stack has been installed. Check its status by running:
  kubectl --namespace monitoring get pods -l "release=monitoring"

Get Grafana 'admin' user password by running:
  kubectl --namespace monitoring get secrets monitoring-grafana -o jsonpath="{.data.admin-password}" | base64 -d ; echo

Access Grafana local instance:
  export POD_NAME=$(kubectl --namespace monitoring get pod -l "app.kubernetes.io/name=grafana,app.kubernetes.io/instance=monitoring" -o name)
  kubectl --namespace monitoring port-forward $POD_NAME 3000

Get your grafana admin user password by running:
  kubectl get secret --namespace monitoring -l app.kubernetes.io/component=admin-secret -o jsonpath=".items[0].data.admin-password" | base64 --decode ; echo

Visit https://github.com/prometheus-operator/kube-prometheus for instructions on how to create & configure Alertmanager and Prometheus instances using the Operator.

```

4.3 — Vérifier

kubectl -n monitoring get pods

```

PS C:\Users\Dimi ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl -n monitoring get pods
NAME                  READY   STATUS    RESTARTS   AGE
alertmanager-monitoring-kube-prometheus-alertmanager-0   2/2     Running   0          30h
monitoring-grafana-bcd96cc6-wct4b                         3/3     Running   0          30h
monitoring-kube-prometheus-operator-558d87d77c-x8xfh       1/1     Running   0          30h
monitoring-kube-state-metrics-69b6c7f8c-rzzfb             1/1     Running   0          30h
monitoring-prometheus-node-exporter-s8qdl                 0/1     CrashLoopBackOff   252 (117s ago)   30h
prometheus-monitoring-kube-prometheus-prometheus-0        2/2     Running   0          30h

```

4.4 — Ouvrir Grafana en local (port-forward)

kubectl -n monitoring port-forward svc/monitoring-grafana 3000:80

Login admin (récupérer le password) :

kubectl -n monitoring get secret monitoring-grafana -o jsonpath=".data.admin-password" | %{{ [Text.Encoding]::UTF8.GetString([Convert]::FromBase64String(\$_)) }}

Puis navigateur : <http://localhost:3000>

Nom d'utilisateur -> admin

Mdp -> voir résultat de la commande : kubectl -n monitoring get secret monitoring-grafana -o jsonpath=".data.admin-password" | %{{ [Text.Encoding]::UTF8.GetString([Convert]::FromBase64String(\$_)) }}

Quand tu arrives sur Grafana, **n'explore pas au hasard.**

Suis **toujours le même chemin mental.**

1 Dashboards (le point d'entrée principal)

 C'est L'ONGLET à ouvrir en premier

💡 Menu de gauche → **Dashboards**

Pourquoi

- 100 % de l'observabilité exploitable est ici
- Tout le reste (Explore, Alerting...) vient **après**

● 2 Dans Dashboards → ouvrir ceux-ci (dans cet ordre)

✓ A. Kubernetes / Compute Resources / Cluster

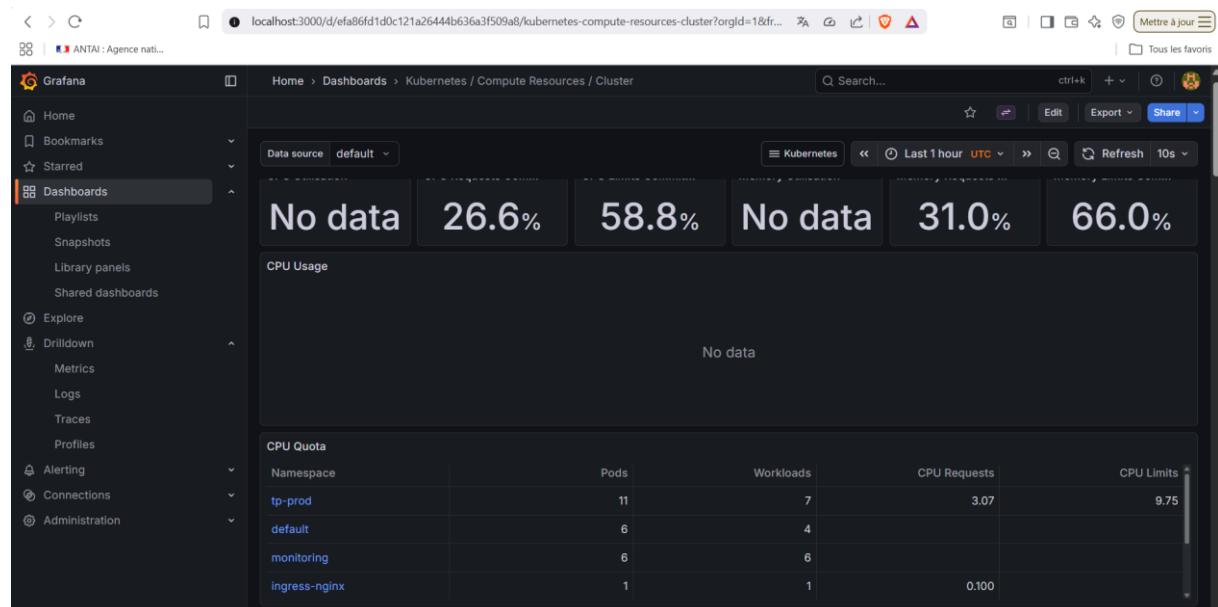
💡 Chemin typique :

Dashboards

└ Kubernetes

 └ Compute Resources

 └ Cluster



Ce que tu observes ici

- CPU global
- Mémoire globale
- Pods Running / Pending
- Nodes Ready

🎯 Question à laquelle ce dashboard répond :

“Est-ce que mon cluster va bien ?”

👉 Si ici tout est stable → le problème n'est PAS l'infra globale.

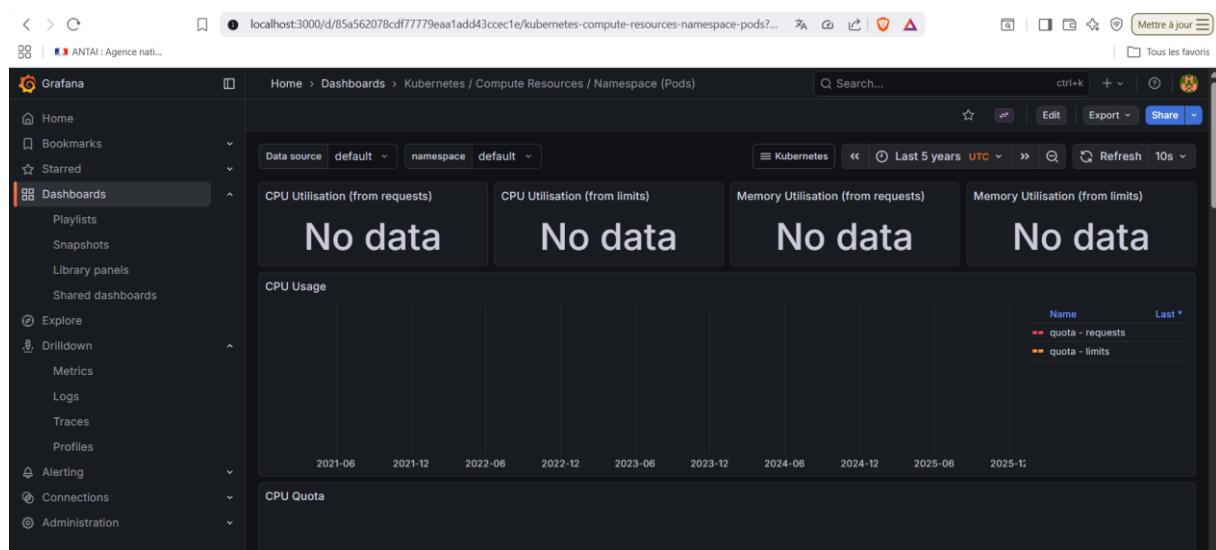
✓ B. Kubernetes / Compute Resources / Namespace (Pods)

📍 Chemin :

Kubernetes

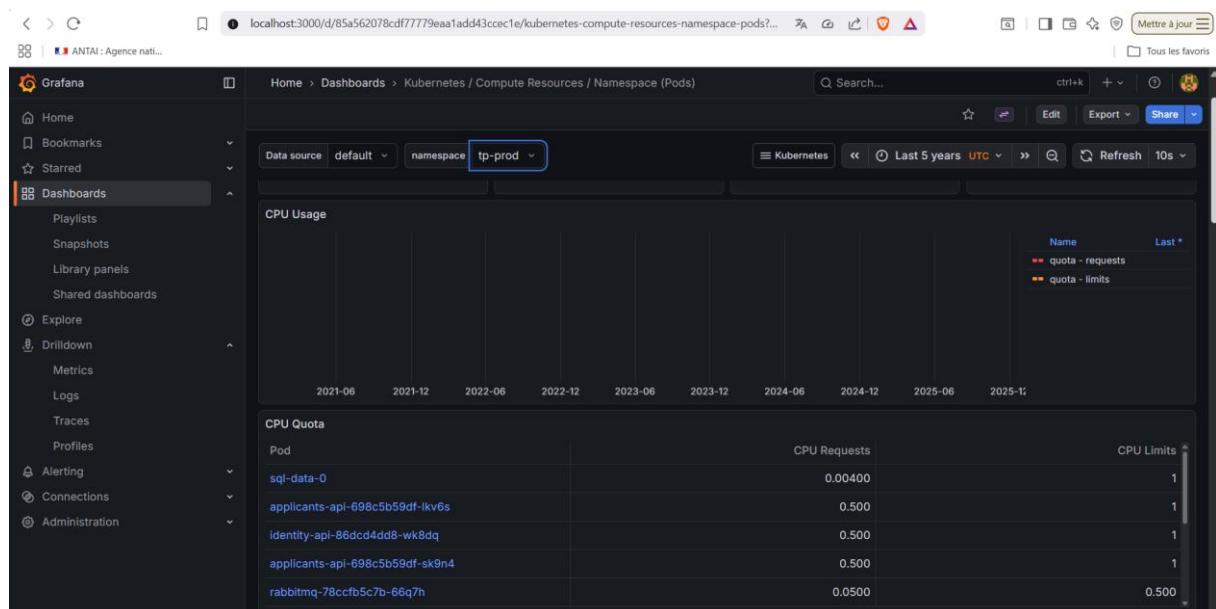
└ Compute Resources

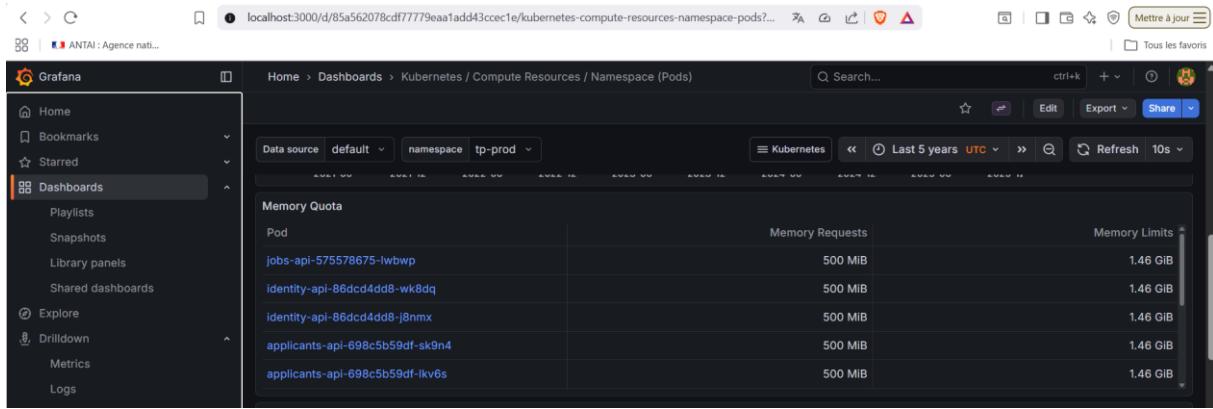
 └ Namespace (Pods)



🔧 Filtre en haut du dashboard :

- Namespace = tp-prod





Ce que l'on observe

- CPU par pod
- Mémoire par pod
- Restart count
- Pods non ready

👉 Question :

“Quel service consomme / dysfonctionne ?”

👉 C'est LE dashboard applicatif principal côté infra.

C. Kubernetes / Compute Resources / Pod

📍 Chemin :

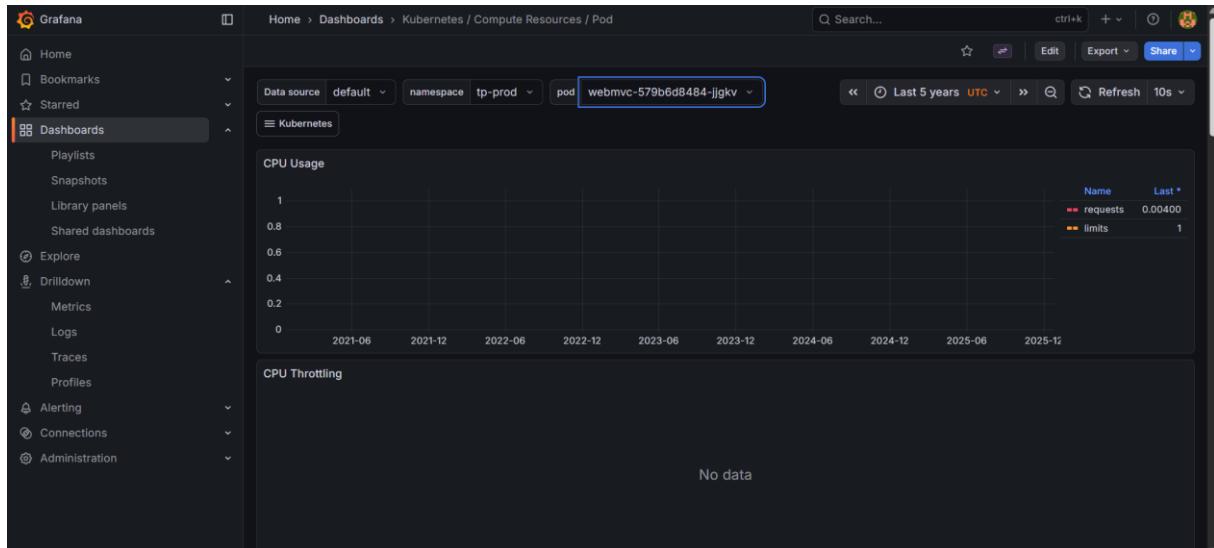
Kubernetes

└ Compute Resources

 └ Pod

🔧 Filtre :

- Namespace = tp-prod
- Pod = webmvc-* (ou jobs-api, etc.)



Ce que tu observes

- CPU / mémoire **au niveau d'un pod précis**
- Pic soudain ?
- Comportement anormal ?

🎯 Question :

“Ce pod est-il sain individuellement ?”

D. Kubernetes / Horizontal Pod Autoscaler (même sans HPA)

📍 Chemin :

Kubernetes

└─ Horizontal Pod Autoscaler

Ce que tu observes

- CPU requests vs usage
- Memory requests vs usage

🎯 Question :

“Mes ressources sont-elles bien dimensionnées ?”

👉 Même sans HPA actif, **ce dashboard t'apprend à raisonner capacité.**

 **3 Explore** (à utiliser ponctuellement, pas en continu)

 Menu de gauche → **Explore**

Quand l'utiliser

- Pour creuser **une métrique précise**
- Pour vérifier une intuition
- Pour corréler avec un moment précis (incident)

 Ce n'est **pas un tableau de bord**, c'est une **loupe**.

 **Ce que tu peux ignorer pour l'instant (et c'est NORMAL)**

- Alerting (tu n'as pas encore de SLO)
- Admin / Configuration
- Plugins
- Dashboards hyper bas niveau (kernel, filesystem...)

 Les bons ingénieurs savent **ce qu'ils n'ont pas besoin de regarder**.

 **Ton “rituel Grafana” idéal (retenable)**

À chaque session :

1. **Dashboards → Cluster**
2. **Dashboards → Namespace (tp-prod)**
3. **Dashboards → Pod (si besoin)**
4. **Kibana si erreur utilisateur**
5. Fin.

 Si tu suis ce rituel, tu es **opérationnel en prod**.

 **Résumé ultra-court**

Si tu dois retenir **3 clics seulement** :

- 1 Dashboards → Kubernetes → Compute Resources → Cluster**
- 2 Dashboards → Kubernetes → Compute Resources → Namespace (Pods)**
- 3 Dashboards → Kubernetes → Compute Resources → Pod**

5) Logs centralisés (EFK : Fluent Bit + Elasticsearch + Kibana)

5.1 — Ajouter repo Elastic + Fluent

```
helm repo add elastic https://helm.elastic.co
```

```
helm repo add fluent https://fluent.github.io/helm-charts
```

```
helm repo update
```

(Elastic charts) [GitHub+1](#)

(Fluent charts) fluent.github.io+2artifacthub.io+2

```
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> helm repo add elastic https://helm.elastic.co
"elastic" has been added to your repositories
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> helm repo add fluent https://fluent.github.io/helm-charts
"fluent" has been added to your repositories
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "ingress-nginx" chart repository
...Successfully got an update from the "fluent" chart repository
...Successfully got an update from the "elastic" chart repository
...Successfully got an update from the "prometheus-community" chart repository
Update Complete. ⚡Happy Helming!⚡
```

5.2 — Créer namespace logging

```
kubectl create namespace logging
```

```
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl create namespace logging
namespace/logging created
```

5.3 — Installer Elasticsearch (single-node pour Docker Desktop)

```
helm install elasticsearch elastic/elasticsearch `
```

```
--namespace logging `
```

```
--set replicas=1 `

--set minimumMasterNodes=1 `

--set resources.requests.cpu=200m `

--set resources.requests.memory=1Gi `

--set resources.limits.cpu=1 `

--set resources.limits.memory=2Gi
```

```
PS C:\Users\Dimitri ANGELEY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> helm install elasticsearch elastic/elasticsearch `

>> --namespace logging `

>> --set replicas=1 `

>> --set minimumMasterNodes=1 `

>> --set resources.requests.cpu=200m `

>> --set resources.requests.memory=1Gi `

>> --set resources.limits.cpu=1 `

>> --set resources.limits.memory=2Gi

level=WARN msg="unable to find exact version; falling back to closest available version" chart=elasticsearch requested="" selected=8.5.1
I1224 17:53:48.931742    3204 warnings.go:110] "Warning: spec.SessionAffinity is ignored for headless services"
NAME: elasticsearch
LAST DEPLOYED: Wed Dec 24 17:53:48 2025
NAMESPACE: logging
STATUS: deployed
REVISION: 1
DESCRIPTION: Install complete
NOTES:
1. Watch all cluster members come up.
$ kubectl get pods --namespace=logging -l app=elasticsearch-master -w
2. Retrieve elastic user's password.
$ kubectl get secrets --namespace=logging elasticsearch-master-credentials -ojsonpath='{.data.password}' | base64 -d
3. Test cluster health using Helm test.
$ helm --namespace=logging test elasticsearch
```

5.4 — Installer Kibana

```
helm install kibana elastic/kibana `

--namespace logging

--timeout 10m --wait `

--set elasticsearchHosts=https://elasticsearch-master:9200
```

(J'ai malheureusement fermé la fenêtre powershell avant d'avoir faire une capture mais j'avais envoyé la réponse à chatgt d'où la capture suivante)

```
j'ai : helm install kibana elastic/kibana
>> --namespace logging --create-namespace
>> --timeout 10m --wait
>> --set elasticsearchHosts="https://elasticsearch-master:9200"
level=WARN msg="unable to find exact version; falling back to closest
available version" chart=kibana requested="" selected=8.5.1
NAME: kibana
LAST DEPLOYED: Wed Dec 24 18:18:11 2025
NAMESPACE: logging
STATUS: deployed
REVISION: 1
DESCRIPTION: Install complete
TEST SUITE: None
NOTES:
1. Watch all containers come up.
$ kubectl get pods --namespace=logging -l release=kibana -w
2. Retrieve the elastic user's password.
$ kubectl get secrets --namespace=logging elasticsearch-master-
credentials -ojsonpath='{.data.password}' | base64 -d
3. Retrieve the kibana service account token.
$ kubectl get se(↓) --namespace=logging kibana-kibana-es-token -
ojsonpath='{.data.token}' | base64 -d
```

5.5 — Installer Fluent Bit (DaemonSet)

```
helm install fluent-bit fluent/fluent-bit `

--namespace logging `

--set backend.type=es `

--set backend.es.host=elasticsearch-master `

--set backend.es.port=9200
```

```

PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> helm install fluent-bit fluent/fluent-bit \
--namespace logging \
--set backend.type=es \
--set backend.es.host=elasticsearch-master \
--set backend.es.port=9200
level=WARNING msg="unable to find exact version; falling back to closest available version" chart=fluent-bit requested="" selected=0.54.0
NAME: fluent-bit
LAST DEPLOYED: Wed Dec 24 18:26:32 2025
NAMESPACE: logging
STATUS: deployed
REVISION: 1
DESCRIPTION: Install complete
NOTES:
Get Fluent Bit build information by running these commands:
export POD_NAME=$(kubectl get pods --namespace logging -l "app.kubernetes.io/name=fluent-bit,app.kubernetes.io/instance=fluent-bit" -o jsonpath="{.items[0].metadata.name}")
kubectl --namespace logging port-forward $POD_NAME 2020:2020
curl http://127.0.0.1:2020

```

5.6 — Vérifier pods

kubectl -n logging get pods

```

PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl -n logging get pods
NAME          READY   STATUS    RESTARTS   AGE
elasticsearch-master-0   1/1     Running   0          31h
fluent-bit-2zzld   1/1     Running   0          30h
kibana-kibana-666dbcfd9d-9sd4c  1/1     Running   0          31h

```

5.7 — Ouvrir Kibana (port-forward)

1) Vérifier que Kibana démarre

kubectl -n logging get pods -l release=kibana -w

Attends que le pod soit Running et READY 1/1.

```

PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl -n logging get pods -l release=kibana -w
NAME          READY   STATUS    RESTARTS   AGE
kibana-kibana-666dbcfd9d-9sd4c  1/1     Running   0          31h

```

2) Trouver le service Kibana

kubectl -n logging get svc | Select-String kibana

Souvent le service s'appelle kibana-kibana.

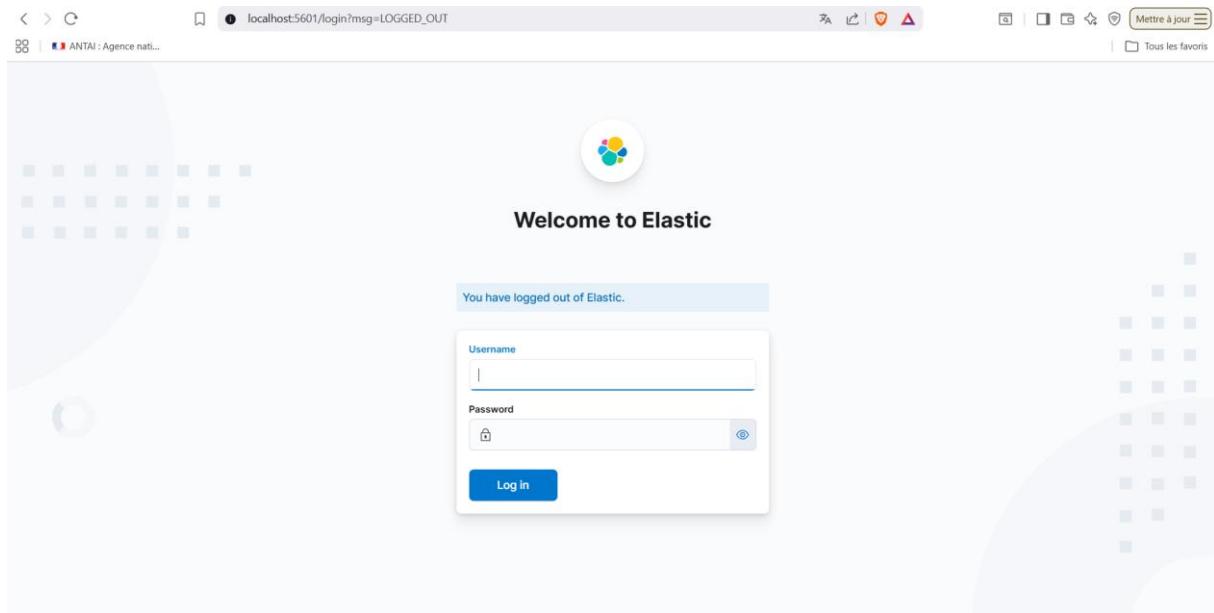
```

PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl -n logging get svc | Select-String kibana
kibana-kibana           ClusterIP  10.111.158.4    <none>        5601/TCP      31h

```

```
kubectl -n logging port-forward svc/kibana-kibana 5601:5601
```

Navigateur : <http://localhost:5601>



Utiliser l'utilisateur elastic

Mot de passe du user elastic :

```
kubectl -n logging get secret elasticsearch-master-credentials -o  
jsonpath="{.data.password}"`
```

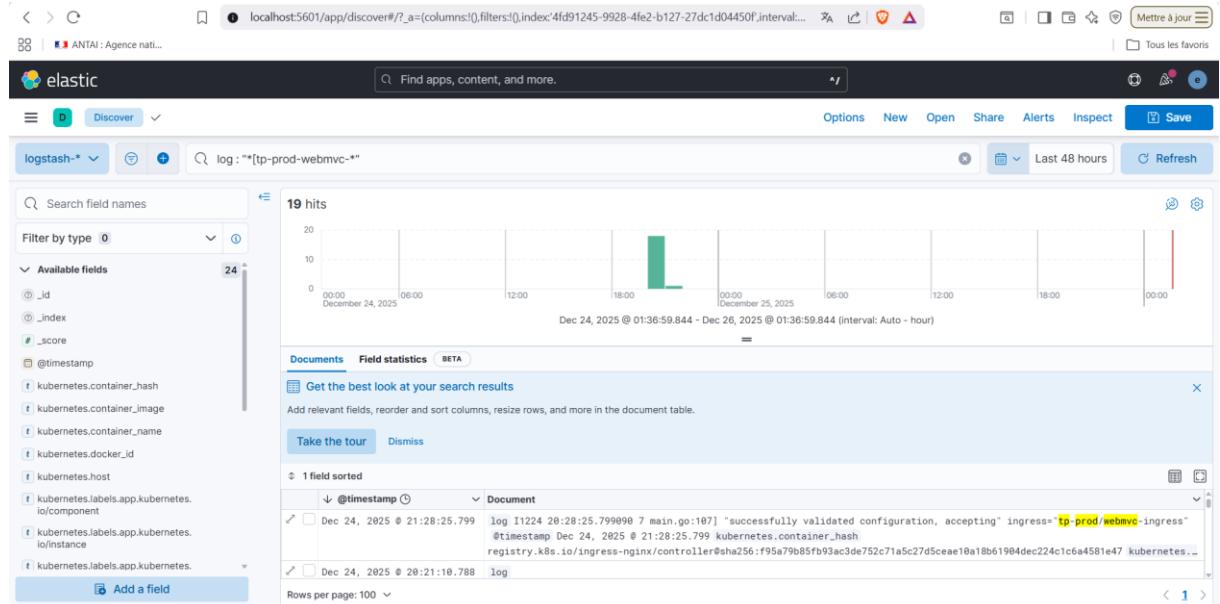
```
| % { [Text.Encoding]::UTF8.GetString([Convert]::FromBase64String($_)) }
```

Ensuite login Kibana :

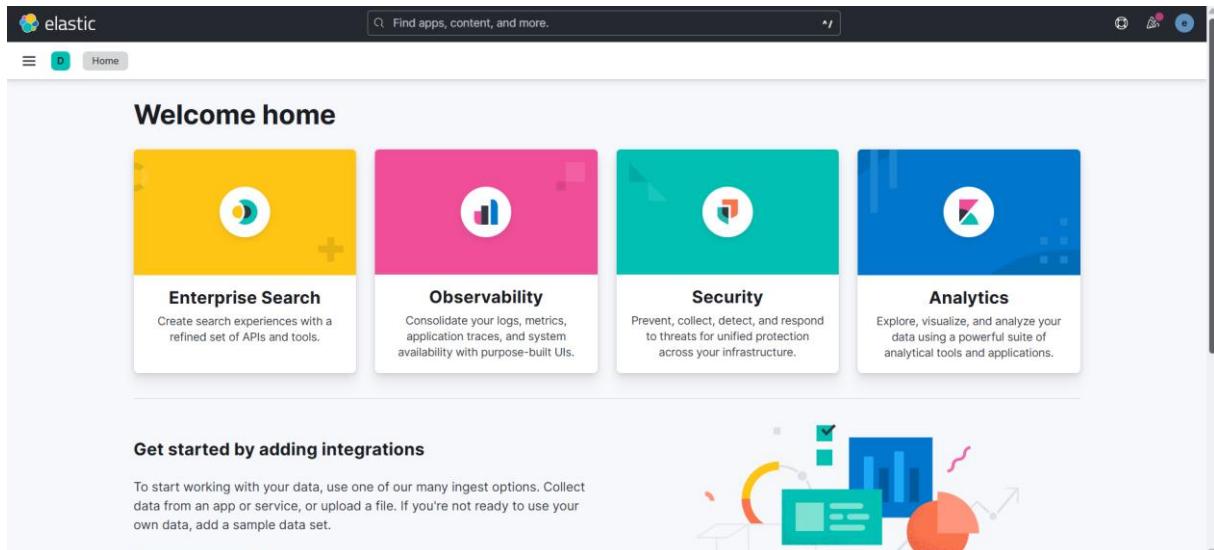
- user: elastic
- password: (valeur récupérée)

Ce que tu dois faire dans Kibana (pour le rendu TP)

- Créer un “Data view” sur l’index Fluent Bit (souvent logstash-* ou fluent-bit* selon config)
- Dashboard : logs d’un service (ex: app=webmvc) sur 15 min
- Filtre erreurs : level:ERROR ou status:[500 TO 599] si présent dans les logs



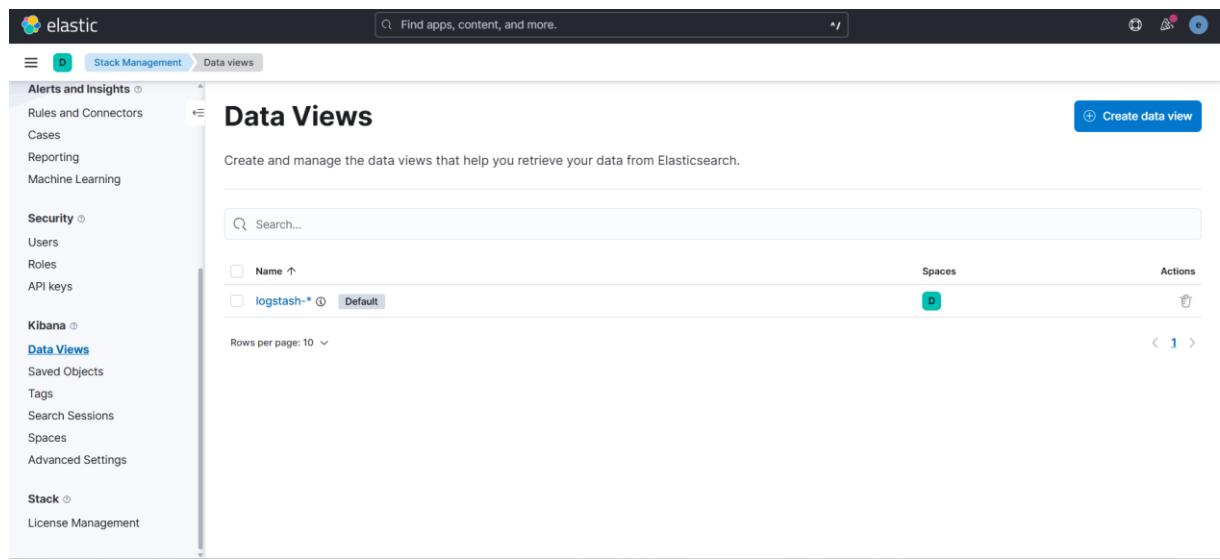
Normalement on doit atterrir sur cette page après la première connexion :



Créer le Dashboard

1. Menu 
2. Onglet management -> stack management
3. Onglet kibana -> dataview

L'ayant déjà fais j'ai :



The screenshot shows the Elasticsearch Stack Management interface. The left sidebar has sections for Alerts and Insights, Security, Kibana (with Data Views selected), and Stack. The main area is titled "Data Views" and contains a search bar and a table. The table has columns for Name, Spaces, and Actions. One row is shown: "logstash-*" under the "Default" space. There are buttons for "Create data view" and "Rows per page: 10".

mais normalement c'est :

Lien **en bas de la page** :

create a data view against hidden, system or default indices

(C'est écrit en bleu, sous “Think you already have data?”)

1 Champ Index pattern

logstash-*

2 Champ **Timestamp field**

@timestamp

(C'est indispensable pour les graphiques et le temps réel)

3 Clique sur Create data view

 C'est terminé.

NB : cela n'a pas marché du coup il a fallu suivre les étapes suivantes avant de pouvoir entrer logstash-* puis timestamp.

Le problème étais le suivant : **Fluent Bit essaie d'envoyer vers Elasticsearch en clair (HTTP/9200)** alors que ton Elasticsearch Helm est en **HTTPS/TLS** (tu l'as déjà vu avec Kibana qui refusait http:). Résultat : “**broken connection**” et **aucun index créé**, donc Kibana ne voit rien.

A faire (resultat d'un très long débogage que j'ai pris le temps de nettoyé) :

1 Récupérer le mot de passe Elasticsearch

```
$ES_PASS = kubectl -n logging get secret elasticsearch-master-credentials -o  
jsonpath=".data.password" `  
| % { [Text.Encoding]::UTF8.GetString([Convert]::FromBase64String($_)) }
```

```
$ES_PASS  
@"  
config:  
outputs: |  
[OUTPUT]  
  Name es  
  Match kube.*  
  Host elasticsearch-master  
  Port 9200  
  Logstash_Format On  
  Retry_Limit False  
  TLS On  
  TLS.Verify Off  
  HTTP_User elastic  
  HTTP_Passwd $ES_PASS  
  Suppress_Type_Name On  
  
[OUTPUT]
```

```

Name es

Match host.*

Host elasticsearch-master

Port 9200

Logstash_Format On

Logstash_Prefix node

Retry_Limit False

TLS On

TLS.Verify Off

HTTP_User elastic

HTTP_Passwd $ES_PASS

Suppress_Type_Name On

"@ | Set-Content -Path .\fluent-bit-values.yaml -Encoding utf8

```

2) Applique la config de façon persistante (Helm upgrade)

```
helm upgrade fluent-bit fluent-bit -n logging -f .\fluent-bit-values.yaml
```

```

PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> helm upgrade fluent-bit fluent/fluent-bit -n logging -f .\fluent-bit-values.yaml
level=INFO msg="Upgrading chart fluent-bit"
level=INFO msg="Chart version: 0.54.0"
level=INFO msg="Selected version: 0.54.0"
level=INFO msg="Values file: C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\values.yaml"
level=INFO msg="Releasing 'fluent-bit' has been upgraded. Happy Helming!"
NAME: fluent-bit
LAST DEPLOYED: Wed Dec 24 19:26:52 2025
NAMESPACE: logging
STATUS: deployed
REVISION: 2
DESCRIPTION: Upgrade complete
NOTES:
Get Fluent Bit build information by running these commands:
export POD_NAME=$(kubectl get pods --namespace logging -l "app.kubernetes.io/name=fluent-bit,app.kubernetes.io/instance=fluent-bit" -o jsonpath=".items[0].metadata.name")
kubectl --namespace logging port-forward $POD_NAME 2020:2020
curl http://127.0.0.1:2020

```

Puis redémarre le DaemonSet :

```
kubectl -n logging rollout restart ds/fluent-bit
```

```
kubectl -n logging rollout status ds/fluent-bit
```

```

PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl -n logging rollout restart ds/fluent-bit
daemonset.apps/fluent-bit restarted
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl -n logging rollout status ds/fluent-bit
Waiting for daemon set "fluent-bit" rollout to finish: 0 out of 1 new pods have been updated...
Waiting for daemon set "fluent-bit" rollout to finish: 0 of 1 updated pods are available...
daemon set "fluent-bit" successfully rolled out

```

3) Vérifie que l'erreur a disparu

```
kubectl -n logging logs ds/fluent-bit --tail=50
```

Tu ne dois plus voir de HTTP status=400 ... _type.

Capture avant correction de l'erreur

```
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl -n logging logs ds/fluent-bit --tail=50
[2025/12/24 18:29:39.836303596] [ warn ] [engine] failed to flush chunk '1-1766600979.459677259.flb', retry in 11 seconds: task_id=230, input=tail.0 > output
=es.0 (out_id=0)
[2025/12/24 18:29:39.836758094] [error ] [output:es:es.0] HTTP status=400 URI=/_bulk, response:
{"error":{"root_cause":[{"type":"illegal_argument_exception","reason":"Action/metadata line [1] contains an unknown parameter [_type]"}, {"type":"illegal_argument_exception","reason":"Action/metadata line [1] contains an unknown parameter [_type]"}, "status":400}}
```

Capture après correction de l'erreur

```
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> kubectl -n logging logs ds/fluent-bit --tail=50
[2025/12/26 01:24:17.683240760] [ warn ] [engine] failed to flush chunk '1-1766604283.474774435.flb', retry in 1404 seconds: task_id=1684, input=tail.0 > out
put=es.0 (out_id=0)
[2025/12/26 01:24:17.683298130] [ warn ] [engine] failed to flush chunk '1-1766604190.915296460.flb', retry in 635 seconds: task_id=1637, input=tail.0 > outp
ut=es.0 (out_id=0)
[2025/12/26 01:24:17.683347055] [ warn ] [engine] failed to flush chunk '1-1766601216.431645637.flb', retry in 1977 seconds: task_id=35, input=tail.0 > outpu
t=es.0 (out_id=0)
```

4) Vérifie que des index apparaissent (Kibana Dev Tools)

GET _cat/indices?v

Tu devrais voir apparaître des index style :

- logstash-2025.12.24
- node-2025.12.24

1	health	status	index	uuid	pri	rep	docs.count	docs.deleted	200 - OK	117 ms
2	yellow	open	logstash-2025.12.24	-C1aIwL2Sq2G26gAgvOzKQ	1	1	76404	0		
3				24.3mb						

Parfait ✅ 🎉

Le screenshot montre que les index existent maintenant : tu as **logstash-2025.12.24** avec des documents dedans. Donc Fluent Bit → Elasticsearch fonctionne.

👉 Maintenant, pour faire ce que l'on voulais (Data view + dashboard + filtres), voici exactement quoi faire.

1 Créer le Data view (index Fluent Bit)

1. Kibana → **Stack Management** → **Data views**
2. **Create data view**
3. **Name / Index pattern :**
4. **logstash-***

5. Time field :

6. @timestamp

7. Create data view

- Tu pourras maintenant voir les logs dans Discover.

2 Voir les logs sur 15 minutes

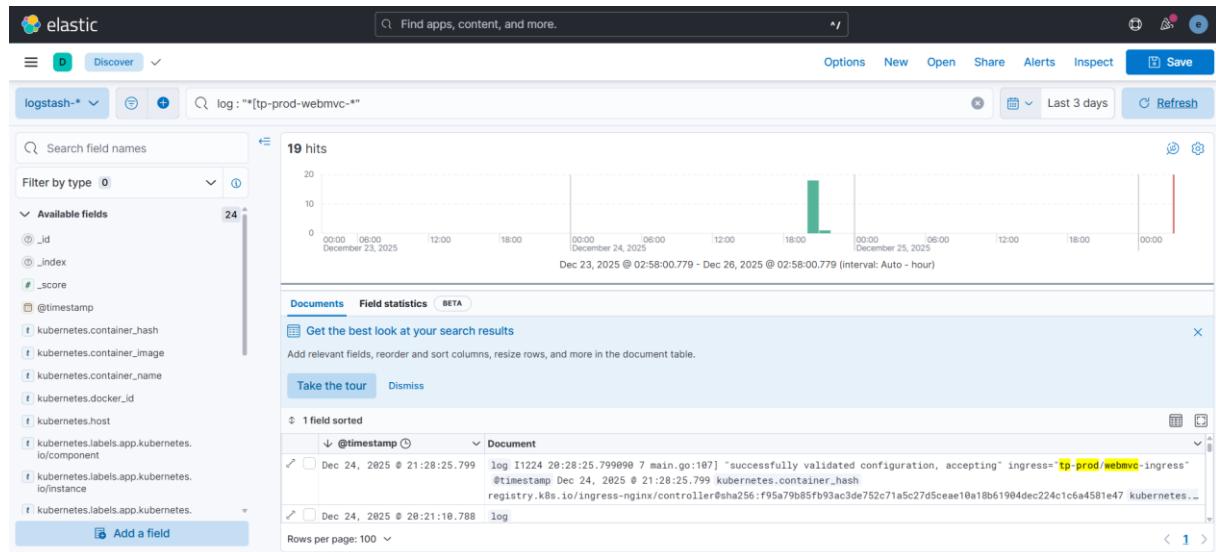
1. Menu  → Discover

2. En haut à droite → sélectionne **Last 15 minutes**

3. Assure-toi que le Data view sélectionné est bien **logstash-***

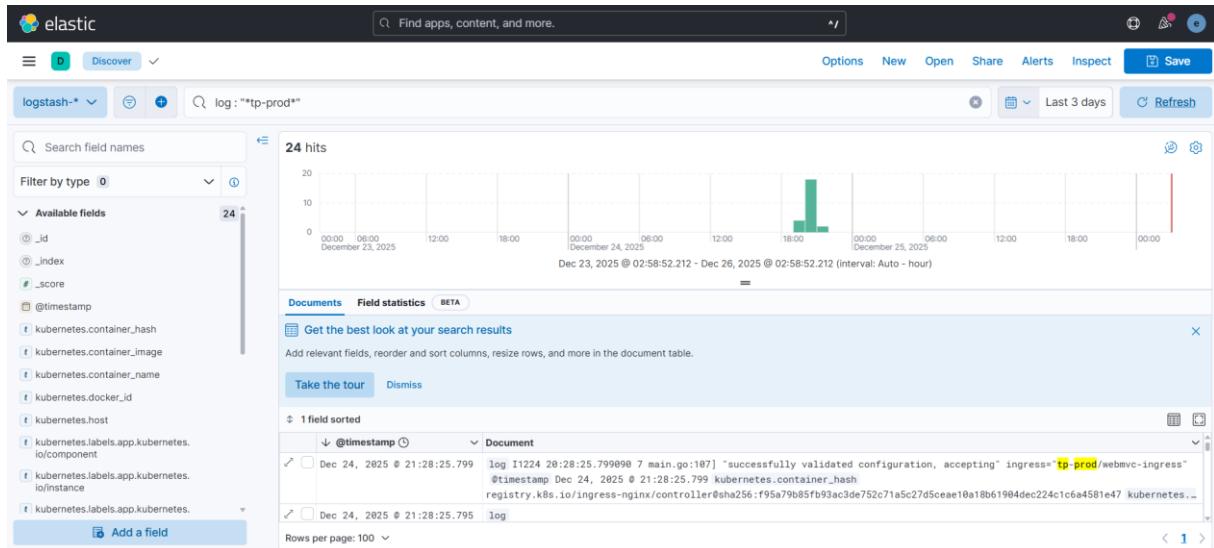
Filtre KQL

log : "*[tp-prod-webmvc-*"



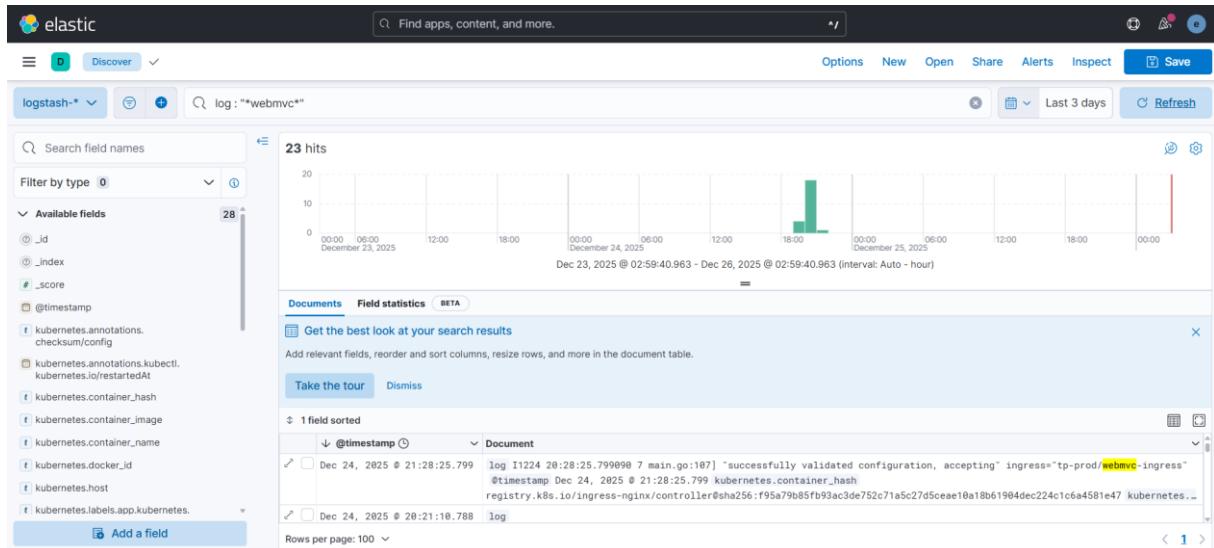
Variante la plus large

log : "*tp-prod*"



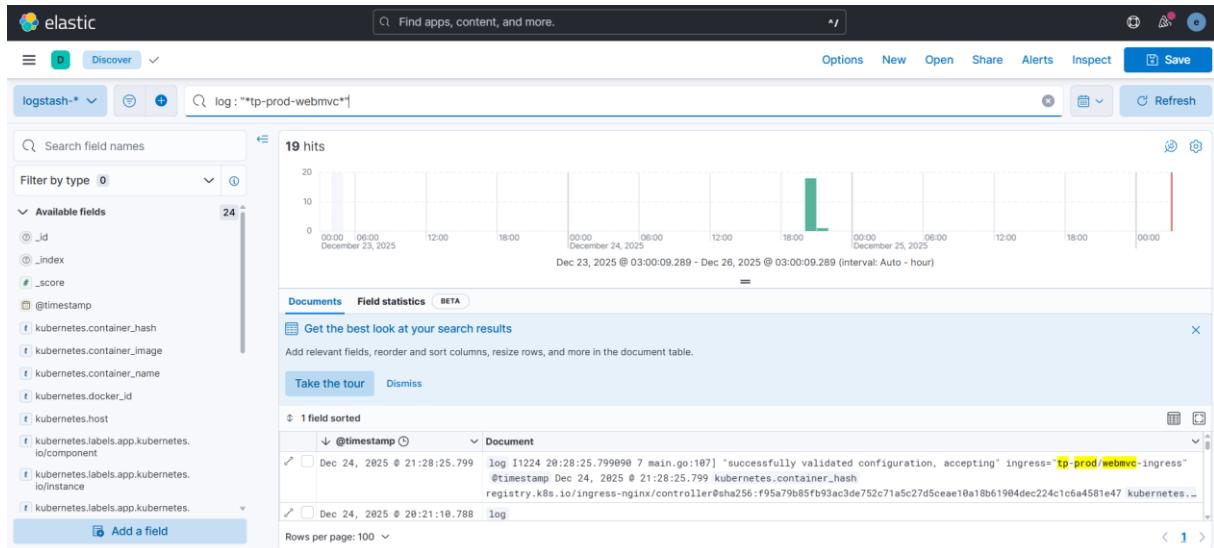
Puis plus précise

log : "webmvc"



Puis la version “ingress” exacte

log : "tp-prod-webmvc"



à quoi s'attendre exactement pour chaque variante, **ce que ça inclut, ce que ça exclut**, et **quand l'utiliser**.

🧠 Principe général à garder en tête

Plus ton filtre est **large** :

- plus tu vois de choses
- plus tu dois **interpréter**

Plus ton filtre est **précis** :

- moins tu vois de choses
 - plus ce que tu vois est **directement actionable**
-

💡 1 Variante la plus large

log : "***tp-prod***"

🔍 Ce que tu vas voir

Tout ce qui mentionne tp-prod quelque part dans la ligne de log, par exemple :

- logs ingress ([tp-prod-webmvc-80])
- logs applicatifs s'ils contiennent le nom du namespace
- logs d'erreurs qui mentionnent tp-prod

- logs système liés au namespace
- parfois des faux positifs (commentaires, messages, debug)

👉 C'est un **filet large**.

🎯 **À quoi ça sert**

- Vérifier que **les logs de tp-prod arrivent bien**
- Déterminer **quels types de logs existent**
- Découvrir **les patterns de nommage**
- Explorer un namespace inconnu

🧠 **Lecture “ingénieur”**

“Dis-moi tout ce que le système sait à propos de tp-prod.”

Quand l'utiliser

- diagnostic initial
- exploration
- validation de pipeline Fluent Bit / ES

Quand NE PAS l'utiliser

- dashboards
- alertes
- analyses fines

➡ **2 Variante plus précise**

log : "*webmvc"

🔍 **Ce que tu vas voir**

Tout ce qui mentionne webmvc, **quel que soit le contexte** :

- ingress webmvc
- logs applicatifs webmvc
- erreurs mentionnant webmvc
- parfois des chemins (/webmvc)
- parfois des images Docker / noms de pod

👉 C'est un **filtre par composant**, pas par rôle.

⌚ À quoi ça sert

- savoir **ce que fait un service donné**
- corrélérer plusieurs sources de logs autour d'un même composant
- comprendre l'impact global d'un service

⚠ Limite importante

Tu peux voir :

- plusieurs namespaces
- plusieurs environnements (si un jour tu as tp-preprod, tp-dev...)

Donc **ce n'est pas encore "production safe"**.

🧠 Lecture "ingénieur"

"Montre-moi tout ce qui concerne webmvc, peu importe d'où ça vient."

Quand l'utiliser

- debug applicatif
- analyse transverse
- corrélation multi-flux

▀ 3 Variante "ingress" exacte (la plus propre pour toi aujourd'hui)

log : "*tp-prod-webmvc*

🔍 Ce que tu vas voir

UNIQUEMENT :

- les logs ingress **du service webmvc**
- pour **le namespace tp-prod**
- pour **le port exposé (80)**

Exemples typiques :

[tp-prod-webmvc-80] "GET /jobs HTTP/1.1" 404

👉 C'est **chirurgical**.

⌚ À quoi ça sert

- dashboards
- suivi du trafic réel utilisateur
- erreurs HTTP
- performance perçue
- incidents “le site ne marche pas”

🧠 Lecture “ingénieur”

“Montre-moi ce que les utilisateurs vivent réellement sur webmvc en prod.”

Quand l'utiliser

- exploitation quotidienne
- démonstration
- KPI / SLA
- post-mortem

💡 Résumé clair (à mémoriser)

Filtre	Niveau	Sert à
tp-prod	Namespace	Explorer, vérifier l'ingestion
webmvc	Service	Debug applicatif transverse
tp-prod-webmvc	Route ingress	Exploitation, dashboards

Etat des lieux : helm list -A

```
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> helm list -A
NAME      NAMESPACE   REVISION  UPDATED           STATUS        CHART          APP VERSION
elasticsearch  logging    1         2025-12-24 17:53:48.7408172 +0100 CET  deployed     .elasticsearch-8.5.1  8.5.1
fluent-bit     logging    3         2025-12-24 19:32:22.5235248 +0100 CET  deployed      fluent-bit-0.54.0   4.1.0
ingress-nginx  ingress-nginx 1         2025-12-24 12:23:29.0037826 +0100 CET  deployed      ingress-nginx-4.14.1 1.14.1
kibana        logging    1         2025-12-24 18:18:11.8243562 +0100 CET  deployed      kibana-8.5.1       8.5.1
monitoring     monitoring  1         2025-12-24 17:43:38.7526411 +0100 CET  deployed      kube-prometheus-stack-80.6.0 v0.87.1
```

6) Helm (packaging de ton déploiement applicatif)

Objectif : ton chart doit créer

pods/deployments/statefulsets/services/ingress/hpa/rbac/netpol, sans installer ingress-nginx.

n fait **Helm module par module : INFRA** d'abord. Tu vas copier-coller des commandes PowerShell qui créent les fichiers, puis helm template / helm install.

Prérequis : tu as déjà 04-helm-chart\tp-appscore créé et nettoyé les templates (comme dans mon étape 6.1/6.2).

Si ce n'est pas fait, exécute exactement ça :

```
cd .\04-helm-chart
```

```
helm create tp-appscore
```

```
Remove-Item -Recurse -Force .\tp-appscore\templates\tests
```

```
Remove-Item -Force .\tp-appscore\templates\*
```

```
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> cd .\04-helm-chart
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart> helm create tp-appscore
Creating tp-appscore
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart> Remove-Item -Recurse -Force .\tp-appscore\templates\tests
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart> Remove-Item -Force .\tp-appscore\templates\*
```

HELM — Module 1 : INFRA (redis + rabbitmq + sql + secret + pvc + svc + statefulset)

1.1 — Écrire values.yaml (infra uniquement)

Toujours dans 04-helm-chart\tp-appscore :

```
@"
```

```
global:
```

```
  namespace: tp-prod
```

```
  registry: tpfinal.azurecr.io
```

```
  tag: "1.0.0"
```

```
infra:
```

```
  redis:
```

```
    enabled: true
```

```
    image: redis:7
```

```
    servicePort: 6379
```

```
    requests:
```

```
      cpu: "4m"
```

```
      memory: "500Mi"
```

```
    limits:
```

```
cpu: "250m"  
memory: "800Mi"
```

```
rabbitmq:  
  enabled: true  
  image: rabbitmq:3-management  
  servicePorts:  
    amqp: 5672  
    mgmt: 15672  
  requests:  
    cpu: "50m"  
    memory: "256Mi"  
  limits:  
    cpu: "500m"  
    memory: "768Mi"
```

```
mssql:  
  enabled: true  
  imageName: sql-data  
  servicePort: 1433  
  pvcSize: 5Gi  
  acceptEula: "Y"  
  saPassword: "Pass@word"  
  requests:  
    cpu: "4m"  
    memory: "500Mi"  
  limits:  
    cpu: "1"
```

```
memory: "2000Mi"
```

```
"@ | Set-Content -Encoding UTF8 .\values.yaml
```

```
1 global:
2   namespace: tp-prod
3   registry: tpfinal.azurecr.io
4   tag: "1.0.0"
5
6   infra:
7     redis:
8       enabled: true
9       image: redis:7
10      servicePort: 6379
11      requests:
12        cpu: "4m"
13        memory: "500Mi"
14      limits:
15        cpu: "250m"
16        memory: "800Mi"
17
18     rabbitmq:
19       enabled: true
20       image: rabbitmq:3-management
21       servicePorts:
22         amqp: 5672
23         mgmt: 15672
24       requests:
25         cpu: "50m"
26         memory: "256Mi"
27       limits:
28         cpu: "50m"
29         memory: "768Mi"
30
31     mssql:
32       enabled: true
33       imageName: sql-data
34       servicePort: 1433
35       pvcsSize: 5Gi
36       acceptEula: "Y"
```

1.2 — Créer les helpers _helpers.tpl

Fichier : templates_helpers.tpl

```
@"
```

```
{%- define "tp.name" -%}
```

```
tp-appscore
```

```
{%- end -%}
```

```
{%- define "tp.ns" -%}
```

```
{{ .Values.global.namespace | default .Release.Namespace }}
```

```
{%- end -%}
```

```
{%- define "tp.fullname" -%}
```

```
{{ include "tp.name" . }}
```

```
{%- end -%}
```

```
"@ | Set-Content -Encoding UTF8 .\templates\_helpers.tpl
```

```
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> @"
>> {{- define "tp.name" -}}
>> tp-appscore
>> {{- end -}}
>>
>> {{- define "tp.ns" -}}
>> {{ .Values.global.namespace | default .Release.Namespace }}
>> {{- end -}}
>>
>> {{- define "tp.fullname" -}}
>> {{ include "tp.name" . }}
>> {{- end -}}
>> @" | Set-Content -Encoding UTF8 .\templates\_helpers.tpl
```

1.3 — Template INFRA : Redis (Deployment + Service)

Fichier : templates\infra-redis.yaml

@"

{{- if .Values.infra.redis.enabled }}

apiVersion: apps/v1

kind: Deployment

metadata:

name: user-data

namespace: {{ include "tp.ns" . }}

labels:

app: user-data

spec:

replicas: 1

selector:

matchLabels:

app: user-data

template:

metadata:

labels:

app: user-data

spec:

containers:

- name: redis

```
image: {{ .Values.infra.redis.image }}

ports:
  - containerPort: {{ .Values.infra.redis.servicePort }}

resources:
  requests:
    cpu: {{ .Values.infra.redis.requests.cpu | quote }}
    memory: {{ .Values.infra.redis.requests.memory | quote }}

  limits:
    cpu: {{ .Values.infra.redis.limits.cpu | quote }}
    memory: {{ .Values.infra.redis.limits.memory | quote }}

livenessProbe:
  tcpSocket:
    port: {{ .Values.infra.redis.servicePort }}

    initialDelaySeconds: 15
    periodSeconds: 10

readinessProbe:
  tcpSocket:
    port: {{ .Values.infra.redis.servicePort }}

    initialDelaySeconds: 5
    periodSeconds: 5

---

apiVersion: v1
kind: Service
metadata:
  name: user-data
  namespace: {{ include "tp.ns" . }}
labels:
  app: user-data
```

spec:

type: ClusterIP

selector:

app: user-data

ports:

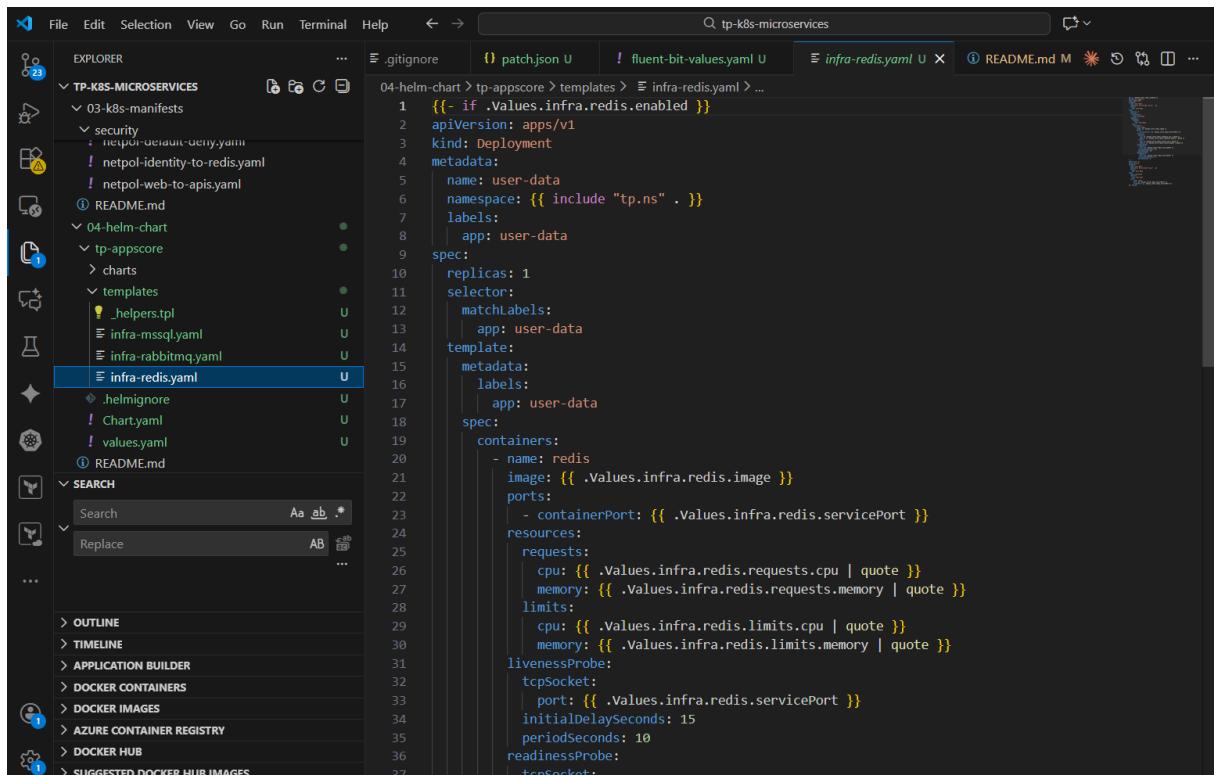
- name: redis

port: {{ .Values.infra.redis.servicePort }}

targetPort: {{ .Values.infra.redis.servicePort }}

{{- end }}

"@ | Set-Content -Encoding UTF8 .\templates\infra-redis.yaml



The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows the project structure under "TP-K8S-MICROSERVICES".
- Editor:** Displays the content of "infra-redis.yaml".
- Search Bar:** Contains "tp-k8s-microservices".
- Bottom Status Bar:** Shows file paths like ".gitignore", "patch.json", "fluent-bit-values.yaml", "infra-redis.yaml", "README.md", and "READMD.md".

```
04-helm-chart > tp-appscore > templates > infra-redis.yaml > ...
1  {{- if .Values.infra.redis.enabled }}
2    apiVersion: apps/v1
3    kind: Deployment
4    metadata:
5      name: user-data
6      namespace: {{ include "tp.ns" . }}
7      labels:
8        app: user-data
9    spec:
10      replicas: 1
11      selector:
12        matchLabels:
13          app: user-data
14      template:
15        metadata:
16          labels:
17            app: user-data
18        spec:
19          containers:
20            - name: redis
21              image: {{ .Values.infra.redis.image }}
22              ports:
23                - containerPort: {{ .Values.infra.redis.servicePort }}
24              resources:
25                requests:
26                  cpu: {{ .Values.infra.redis.requests.cpu | quote }}
27                  memory: {{ .Values.infra.redis.requests.memory | quote }}
28                limits:
29                  cpu: {{ .Values.infra.redis.limits.cpu | quote }}
30                  memory: {{ .Values.infra.redis.limits.memory | quote }}
31              livenessProbe:
32                tcpSocket:
33                  port: {{ .Values.infra.redis.servicePort }}
34              initialDelaySeconds: 15
35              periodSeconds: 10
36              readinessProbe:
37                tcpSocket:
```

1.4 — Template INFRA : RabbitMQ (Deployment + Service)

Fichier : templates\infra-rabbitmq.yaml

@"

{{- if .Values.infra.rabbitmq.enabled }}

apiVersion: apps/v1

kind: Deployment

```
metadata:
  name: rabbitmq
  namespace: {{ include "tp.ns" . }}
  labels:
    app: rabbitmq
spec:
  replicas: 1
  selector:
    matchLabels:
      app: rabbitmq
  template:
    metadata:
      labels:
        app: rabbitmq
    spec:
      containers:
        - name: rabbitmq
          image: {{ .Values.infra.rabbitmq.image }}
          ports:
            - name: amqp
              containerPort: {{ .Values.infra.rabbitmq.servicePorts.amqp }}
            - name: mgmt
              containerPort: {{ .Values.infra.rabbitmq.servicePorts.mgmt }}
      resources:
        requests:
          cpu: {{ .Values.infra.rabbitmq.requests.cpu | quote }}
          memory: {{ .Values.infra.rabbitmq.requests.memory | quote }}
      limits:
```

```
cpu: {{ .Values.infra.rabbitmq.limits.cpu | quote }}

memory: {{ .Values.infra.rabbitmq.limits.memory | quote }}

livenessProbe:

tcpSocket:

port: {{ .Values.infra.rabbitmq.servicePorts.amqp }}

initialDelaySeconds: 20

periodSeconds: 10

readinessProbe:

tcpSocket:

port: {{ .Values.infra.rabbitmq.servicePorts.amqp }}

initialDelaySeconds: 10

periodSeconds: 5
```

```
---
```

```
apiVersion: v1

kind: Service

metadata:

name: rabbitmq

namespace: {{ include "tp.ns" . }}

labels:

app: rabbitmq

spec:

type: ClusterIP

selector:

app: rabbitmq

ports:

- name: amqp

port: {{ .Values.infra.rabbitmq.servicePorts.amqp }}

targetPort: {{ .Values.infra.rabbitmq.servicePorts.amqp }}
```

```

- name: mgmt

port: {{ .Values.infra.rabbitmq.servicePorts.mgmt }}

targetPort: {{ .Values.infra.rabbitmq.servicePorts.mgmt }}

{{- end }}

```

"@ | Set-Content -Encoding UTF8 .\templates\infra-rabbitmq.yaml

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: rabbitmq
  namespace: {{ include "tp.ns" . }}
  labels:
    app: rabbitmq
spec:
  replicas: 1
  selector:
    matchLabels:
      app: rabbitmq
  template:
    metadata:
      labels:
        app: rabbitmq
    spec:
      containers:
        - name: rabbitmq
          image: {{ .Values.infra.rabbitmq.image }}
          ports:
            - name: amqp
              containerPort: {{ .Values.infra.rabbitmq.servicePorts.amqp }}
            - name: mgmt
              containerPort: {{ .Values.infra.rabbitmq.servicePorts.mgmt }}
          resources:
            requests:
              cpu: {{ .Values.infra.rabbitmq.requests.cpu | quote }}
              memory: {{ .Values.infra.rabbitmq.requests.memory | quote }}
            limits:
              cpu: {{ .Values.infra.rabbitmq.limits.cpu | quote }}
              memory: {{ .Values.infra.rabbitmq.limits.memory | quote }}
          livenessProbe:
            tcpSocket:
              port: {{ .Values.infra.rabbitmq.servicePorts.amqp }}
        initialDelaySeconds: 20

```

1.5 — Template INFRA : SQL (Secret + PVC + Service + StatefulSet)

Fichier : templates\infra-mssql.yaml

```

@"

{{- if .Values.infra.mssql.enabled }}

apiVersion: v1

kind: Secret

metadata:

  name: sql-secrets

  namespace: {{ include "tp.ns" . }}

type: Opaque

```

```
stringData:  
  SA_PASSWORD: {{ .Values.infra.mssql.saPassword | quote }}  
---  
apiVersion: v1  
kind: PersistentVolumeClaim  
metadata:  
  name: sql-data-pvc  
  namespace: {{ include "tp.ns" . }}  
  labels:  
    app: sql-data  
spec:  
  accessModes:  
    - ReadWriteOnce  
  resources:  
    requests:  
      storage: {{ .Values.infra.mssql.pvcSize | quote }}  
---  
apiVersion: v1  
kind: Service  
metadata:  
  name: sql-data  
  namespace: {{ include "tp.ns" . }}  
  labels:  
    app: sql-data  
spec:  
  type: ClusterIP  
  selector:  
    app: sql-data
```

```
ports:
  - name: mssql
    port: {{ .Values.infra.mssql.servicePort }}
    targetPort: {{ .Values.infra.mssql.servicePort }}

---
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: sql-data
  namespace: {{ include "tp.ns" .}}
  labels:
    app: sql-data
spec:
  serviceName: sql-data
  replicas: 1
  selector:
    matchLabels:
      app: sql-data
  template:
    metadata:
      labels:
        app: sql-data
    spec:
      containers:
        - name: mssql
          image: {{ .Values.global.registry }}/{{ .Values.infra.mssql.imageName }}:{{ .Values.global.tag }}
          imagePullPolicy: IfNotPresent
```

```
ports:
  - containerPort: {{ .Values.infra.mssql.servicePort }}
    name: mssql

env:
  - name: ACCEPT_EULA
    value: {{ .Values.infra.mssql.acceptEula | quote }}
  - name: SA_PASSWORD
    valueFrom:
      secretKeyRef:
        name: sql-secrets
        key: SA_PASSWORD

volumeMounts:
  - name: sql-data
    mountPath: /var/opt/mssql

resources:
  requests:
    cpu: {{ .Values.infra.mssql.requests.cpu | quote }}
    memory: {{ .Values.infra.mssql.requests.memory | quote }}
  limits:
    cpu: {{ .Values.infra.mssql.limits.cpu | quote }}
    memory: {{ .Values.infra.mssql.limits.memory | quote }}

livenessProbe:
  tcpSocket:
    port: {{ .Values.infra.mssql.servicePort }}
  initialDelaySeconds: 40
  periodSeconds: 10

readinessProbe:
  tcpSocket:
```

```

port: {{ .Values.infra.mssql.servicePort }}

initialDelaySeconds: 20

periodSeconds: 5

volumes:

- name: sql-data

  persistentVolumeClaim:

    claimName: sql-data-pvc

{{- end }}

```

"@ | Set-Content -Encoding UTF8 .\templates\infra-mssql.yaml

```

File Edit Selection View Go Run Terminal Help < > tp-k8s-microservices
EXPLORER .gitignore patchjson U fluent-bit-values.yaml U infra-mssql.yaml U README.md ...
TP-K8S-MICROSERVICES 03-k8s-manifests
  security
    netpol-default-deny.yaml
    netpol-identity-to-redis.yaml
    netpol-web-to-apis.yaml
  README.md
  04-helm-chart
    tp-appscore
      charts
      templates
        .helpers.tpl
        infra-mssql.yaml
        infra-rabbitmq.yaml
        infra-redis.yaml
        .helmignore
        Chart.yaml
        values.yaml
      README.md
  SEARCH
    Search Aa ab *
    Replace AB ...
  ...
  OUTLINE
  TIMELINE
  APPLICATION BUILDER
  DOCKER CONTAINERS
  DOCKER IMAGES
  AZURE CONTAINER REGISTRY
  DOCKER HUB
  SUGGESTED DOCKER HUB IMAGES

```

```

04-helm-chart > tp-appscore > templates > infra-mssql.yaml > ...
1  {{- if .Values.infra.mssql.enabled }}
2  apiVersion: v1
3  kind: Secret
4  metadata:
5    name: sql-secrets
6    namespace: {{ include "tp.ns" . }}
7  type: opaque
8  stringData:
9    SA_PASSWORD: {{ .Values.infra.mssql.saPassword | quote }}
10   ...
11  apiVersion: v1
12  kind: PersistentVolumeClaim
13  metadata:
14    name: sql-data-pvc
15    namespace: {{ include "tp.ns" . }}
16    labels:
17      app: sql-data
18  spec:
19    accessModes:
20      - ReadWriteOnce
21    resources:
22      requests:
23        storage: {{ .Values.infra.mssql.pvcSize | quote }}
24   ...
25  apiVersion: v1
26  kind: Service
27  metadata:
28    name: sql-data
29    namespace: {{ include "tp.ns" . }}
30    labels:
31      app: sql-data
32  spec:
33    type: ClusterIP
34    selector:
35      app: sql-data
36    ports:
37      - name: mssql

```

1.7 — Tester le rendu Helm (sans installer)

Depuis 04-helm-chart\tp-appscore :

```

helm template tp-appscore . --namespace tp-prod | Select-String
"kind:|name:|namespace:" -Context 0,1

```

Tu dois voir Deployment/Service/StatefulSet/PVC/Secret.

```

PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> helm template tp-appscore . --namespace tp-prod | Select-String "kind:[name:]namespace:" -Context 0,1
> kind: Secret
  metadata:
>   name: sql-secrets
>   namespace: tp-prod
  type: Opaque
> kind: PersistentVolumeClaim
  metadata:
>   name: sql-data-pvc
>   namespace: tp-prod
  labels:
> kind: Service
  metadata:
>   name: sql-data
>   namespace: tp-prod
  labels:
>     - name: mssql
>       port: 1433
> kind: Service
  metadata:
>   name: rabbitmq
>   namespace: tp-prod
  labels:
>     - name: amqp
>       port: 5672
>     - name: mgmt
>       port: 15672
> kind: Service
  metadata:
>   name: user-data
>   namespace: tp-prod
  labels:
>     - name: redis
>       port: 6379
> kind: Deployment
  metadata:
>   name: rabbitmq
>   namespace: tp-prod
  labels:

```

1.8 — Installer le module INFRA (nouvelle release Helm)

⚠️ Comme tu as déjà déployé l'infra en YAML, le plus propre est de **supprimer l'infra actuelle** OU d'installer Helm dans un autre namespace.

Le plus simple pour toi : **on supprime les ressources infra existantes** (redis/rabbit/sql + pvc_attach) puis on installe Helm.

1.8.1 — Supprimer infra actuelle (tp-prod)

kubectl -n tp-prod delete deploy user-data rabbitmq

kubectl -n tp-prod delete sts sql-data

kubectl -n tp-prod delete svc user-data rabbitmq sql-data

kubectl -n tp-prod delete pvc sql-data-pvc

kubectl -n tp-prod delete secret sql-secrets

```

PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> kubectl -n tp-prod delete deploy user-data rabbitmq
deployment.apps "user-data" deleted from tp-prod namespace
deployment.apps "rabbitmq" deleted from tp-prod namespace
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> kubectl -n tp-prod delete sts sql-data
statefulset.apps "sql-data" deleted from tp-prod namespace
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> kubectl -n tp-prod delete svc user-data rabbitmq sql-data
service "user-data" deleted from tp-prod namespace
service "rabbitmq" deleted from tp-prod namespace
service "sql-data" deleted from tp-prod namespace
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> kubectl -n tp-prod delete pvc sql-data-pvc
persistentvolumeclaim "sql-data-pvc" deleted from tp-prod namespace
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> kubectl -n tp-prod delete secret sql-secrets
secret "sql-secrets" deleted from tp-prod namespace
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> |

```

1.8.2 — Installer Helm

```
helm install tp-appscore-infra . --namespace tp-prod
```

```
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> helm install tp-appscore-infra . --namespace tp-prod
NAME: tp-appscore-infra
LAST DEPLOYED: Fri Dec 26 03:46:26 2025
NAMESPACE: tp-prod
STATUS: deployed
REVISION: 1
DESCRIPTION: Install complete
TEST SUITE: None
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> |
```

1.8.3 — Vérifier

```
kubectl -n tp-prod get pods
```

```
kubectl -n tp-prod get svc
```

```
kubectl -n tp-prod get pvc
```

```
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> kubectl -n tp-prod get pods
NAME          READY   STATUS    RESTARTS   AGE
applicants-api-698c5b59df-lkv6s  1/1     Running   0          40h
applicants-api-698c5b59df-sk9n4  1/1     Running   0          40h
identity-api-86dc4dd8-j8nmx     1/1     Running   0          38h
identity-api-86dc4dd8-wk8dq     1/1     Running   0          38h
jobs-api-575578675-bz6v6       1/1     Running   0          40h
jobs-api-575578675-lwbwp       1/1     Running   0          40h
rabbitmq-78ccfb5c7b-qnxvm     1/1     Running   0          41s
sql-data-0                      1/1     Running   0          41s
user-data-7b6897b6c7-xkqst      1/1     Running   0          41s
webmvc-77c9686847-rgs8d        1/1     Running   0          5h11m
webmvc-77c9686847-zrmbh        1/1     Running   0          5h11m
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> kubectl -n tp-prod get svc
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
applicants-api ClusterIP  10.100.5.242 <none>        80/TCP          40h
identity-api   ClusterIP  10.99.169.115 <none>        80/TCP          40h
jobs-api       ClusterIP  10.110.39.160 <none>        80/TCP          40h
rabbitmq       ClusterIP  10.98.32.229 <none>        5672/TCP,15672/TCP  42s
sql-data        ClusterIP  10.96.1.125  <none>        1433/TCP        42s
user-data       ClusterIP  10.98.161.39 <none>        6379/TCP        42s
webmvc         ClusterIP  10.96.199.96 <none>        80/TCP          40h
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> kubectl -n tp-prod get pvc
NAME           STATUS    VOLUME          CAPACITY   ACCESS MODES  STORAGECLASS   VOLUMEATRIBUTESCLASS   AGE
sql-data-pvc   Bound    pvc-5f480b7a-4618-4597-bd4b-269582ea4410  5Gi        RWO          hostpath      <unset>          42s
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> |
```

1.9 — Commit Helm infra

Reviens à la racine repo puis commit :

```
cd ..\..
```

```
git add .\04-helm-chart\tp-appscore
```

```
git commit -m "feat(helm): infra module (redis, rabbitmq, mssql) templates"
```

```
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> git add .\04-helm-chart\tp-appscore
warning: in the working copy of '04-helm-chart/tp-appscore/.helmignore', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of '04-helm-chart/tp-appscore/Chart.yaml', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of '04-helm-chart/tp-appscore/templates/_helpers.tpl', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of '04-helm-chart/tp-appscore/templates/_infra-mssql.yaml', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of '04-helm-chart/tp-appscore/templates/_infra-rabbitmq.yaml', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of '04-helm-chart/tp-appscore/templates/_infra-redis.yaml', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of '04-helm-chart/tp-appscore/values.yaml', LF will be replaced by CRLF the next time Git touches it
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> git commit -m "(helm): ajout des modules de templates de l'infra (redis
, rabbitmq, mssql)"
[main 4a2cb32] (helm): ajout des modules de templates de l'infra (redis, rabbitmq, mssql)
 7 files changed, 319 insertions(+)
 create mode 100644 04-helm-chart/tp-appscore/.helmignore
 create mode 100644 04-helm-chart/tp-appscore/Chart.yaml
 create mode 100644 04-helm-chart/tp-appscore/templates/_helpers.tpl
 create mode 100644 04-helm-chart/tp-appscore/templates/_infra-mssql.yaml
 create mode 100644 04-helm-chart/tp-appscore/templates/_infra-rabbitmq.yaml
 create mode 100644 04-helm-chart/tp-appscore/templates/_infra-redis.yaml
 create mode 100644 04-helm-chart/tp-appscore/values.yaml
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> |
```

Helm **infra** est déployé et les pods infra ont été recréés via Helm (âge ~7 min). On passe au **Module 2 : APPS** (web + APIs) sous Helm comme on peut le voir grâce au commande suivante :

```
helm list -n tp-prod
```

```
kubectl -n tp-prod get pods
```

NAME	NAMESPACE	REVISION	UPDATED	STATUS	CHART	APP VERSION
tp-appscore-infra	tp-prod	1	2025-12-26 03:46:26.5150635 +0100 CET	deployed	tp-appscore-0.1.0	1.16.0
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> helm list -n tp-prod						
NAME		READY	STATUS	RESTARTS	AGE	
applicants-api-698c5b59df-lkv6s	1/1	Running	0	0h	40h	
applicants-api-698c5b59df-sk9n4	1/1	Running	0	0h	40h	
identity-api-86dc4dd8-j8nmx	1/1	Running	0	0h	38h	
identity-api-86dc4dd8-wk8dq	1/1	Running	0	0h	38h	
jobs-api-575578675-bz6v6	1/1	Running	0	0h	40h	
jobs-api-575578675-lwbwp	1/1	Running	0	0h	40h	
rabbitmq-78ccfb05c7b-qnxvm	1/1	Running	0	0h	7m11s	
sql-data-0	1/1	Running	0	0h	7m11s	
user-data-7b6897b6c7-xkqst	1/1	Running	0	0h	7m11s	
webmvc-77c9686847-rgs8d	1/1	Running	0	0h	5h17m	
webmvc-77c9686847-zrmbh	1/1	Running	0	0h	5h17m	

on passe au **Module 2 : APPS** (applicants/jobs/identity/webmvc + services + HPA + ingress + affinity + (rbac/netpol en option dans chart)).

Objectif : créer une **deuxième release Helm** tp-appscore-apps qui gère :

- Deployments + Services : applicants-api, jobs-api, identity-api, webmvc
- HPA (webmvc)
- Ingress HTTPS (webmvc)
- Affinités (web ↔ apis)
- (Optionnel dans ce module) ServiceAccount + RBAC (identity)
- (Optionnel dans ce module) NetworkPolicies

Comme tu as déjà RBAC/netpol en YAML, on peut les intégrer Helm ensuite. Là on fait d'abord APPS “core”.

HELM — Module 2 : APPS

2.1 — Mettre à jour values.yaml (ajouter section apps)

Va dans le chart :

```
cd .\04-helm-chart\tp-appscore
```

```
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> cd .\04-helm-chart\tp-appscore
```

Puis **ajoute** à la fin de values.yaml (sans casser l'infra) :

```
Add-Content -Encoding UTF8 .\values.yaml @"
```

```
apps:
```

```
    applicantsApi:
```

```
        enabled: true
```

```
        name: applicants-api
```

```
        imageName: applicants-api
```

```
        replicas: 2
```

```
        containerPort: 80
```

```
        env:
```

```
            hostRabbitmq: rabbitmq
```

```
            database: dotnetgigs.applicants
```

```
        requests:
```

```
            cpu: "500m"
```

```
            memory: "500Mi"
```

```
        limits:
```

```
            cpu: "1"
```

```
            memory: "1500Mi"
```

```
jobsApi:
```

```
    enabled: true
```

```
    name: jobs-api
```

```
    imageName: jobs-api
```

```
    replicas: 2
```

```
    containerPort: 80
```

```
    env:
```

```
        hostRabbitmq: rabbitmq
```

```
database: dotnetgigs.jobs
```

```
requests:
```

```
  cpu: "500m"
```

```
  memory: "500Mi"
```

```
limits:
```

```
  cpu: "1"
```

```
  memory: "1500Mi"
```

```
identityApi:
```

```
  enabled: true
```

```
  name: identity-api
```

```
  imageName: identity-api
```

```
  replicas: 2
```

```
  containerPort: 80
```

```
  env:
```

```
    hostRabbitmq: rabbitmq
```

```
    redisHost: user-data:6379
```

```
  requests:
```

```
    cpu: "500m"
```

```
    memory: "500Mi"
```

```
  limits:
```

```
    cpu: "1"
```

```
    memory: "1500Mi"
```

```
webmvc:
```

```
  enabled: true
```

```
  name: webmvc
```

```
  imageName: webmvc
```

```
replicas: 2
containerPort: 80
requests:
  cpu: "4m"
  memory: "900Mi"
limits:
  cpu: "1"
  memory: "2000Mi"
affinity:
  enabled: true

hpa:
  enabled: true
  targetDeployment: webmvc
  minReplicas: 2
  maxReplicas: 6
  cpuUtilization: 60

ingress:
  enabled: true
  className: nginx
  host: tp.local
  tlsSecret: tp-local-tls
"@
```

```

PS C:\Users\Dimi\ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> Add-Content -Encoding UTF8 .\values.yaml @"
>>
>> apps:
>>   applicantsApi:
>>     enabled: true
>>     name: applicants-api
>>     imageName: applicants-api
>>     replicas: 2
>>     containerPort: 80
>>     env:
>>       hostRabbitmq: rabbitmq
>>       database: dotnetgigs.applicants
>>     requests:
>>       cpu: "500m"
>>       memory: "500Mi"
>>     limits:
>>       cpu: "1"
>>       memory: "1500Mi"
>>
>> jobsApi:
>>   enabled: true
>>   name: jobs-api
>>   imageName: jobs-api
>>   replicas: 2
>>   containerPort: 80
>>   env:
>>     hostRabbitmq: rabbitmq
>>     database: dotnetgigs.jobs
>>   requests:
>>     cpu: "500m"
>>     memory: "500Mi"
>>   limits:
>>     cpu: "1"
>>     memory: "1500Mi"
>>
>> identityApi:
>>   enabled: true
>>   name: identity-api
>>   imageName: identity-api
>>   replicas: 2
>>   containerPort: 80

```

2.2 — Templates APPS : Applicants API

Créer templates\app-applicants-api.yaml

@"

`{{- if .Values.apps.applicantsApi.enabled }}`

apiVersion: apps/v1

kind: Deployment

metadata:

name: {{ .Values.apps.applicantsApi.name }}

namespace: {{ include "tp.ns" . }}

labels:

app: {{ .Values.apps.applicantsApi.name }}

spec:

replicas: {{ .Values.apps.applicantsApi.replicas }}

selector:

matchLabels:

app: {{ .Values.apps.applicantsApi.name }}

template:

```
metadata:
  labels:
    app: {{ .Values.apps.applicantsApi.name }}

spec:
  containers:
    - name: {{ .Values.apps.applicantsApi.name }}
      image: {{ .Values.global.registry }}/{{ .Values.apps.applicantsApi.imageName }}:{{ .Values.global.tag }}
      imagePullPolicy: IfNotPresent
    ports:
      - containerPort: {{ .Values.apps.applicantsApi.containerPort }}
        name: http
    env:
      - name: HostRabbitmq
        value: {{ .Values.apps.applicantsApi.env.hostRabbitmq | quote }}
      - name: ConnectionString
        value: {{ printf "Server=sql-data;User=sa;Password=%s;Database=%s;" .Values.infra.mssql.saPassword .Values.apps.applicantsApi.env.database | quote }}
  resources:
    requests:
      cpu: {{ .Values.apps.applicantsApi.requests.cpu | quote }}
      memory: {{ .Values.apps.applicantsApi.requests.memory | quote }}
    limits:
      cpu: {{ .Values.apps.applicantsApi.limits.cpu | quote }}
      memory: {{ .Values.apps.applicantsApi.limits.memory | quote }}
  readinessProbe:
    tcpSocket:
      port: {{ .Values.apps.applicantsApi.containerPort }}
  initialDelaySeconds: 10
```

```
    periodSeconds: 5

    livenessProbe:
      tcpSocket:
        port: {{ .Values.apps.applicantsApi.containerPort }}

    initialDelaySeconds: 20
    periodSeconds: 10

---
apiVersion: v1
kind: Service
metadata:
  name: {{ .Values.apps.applicantsApi.name }}
  namespace: {{ include "tp.ns" .}}
  labels:
    app: {{ .Values.apps.applicantsApi.name }}
spec:
  type: ClusterIP
  selector:
    app: {{ .Values.apps.applicantsApi.name }}
  ports:
    - name: http
      port: 80
      targetPort: {{ .Values.apps.applicantsApi.containerPort }}

{{- end }}

"@ | Set-Content -Encoding UTF8 .\templates\app-applicants-api.yaml
```

```

PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> @"
>> {{- if .Values.apps.applicantsApi.enabled }}
>> apiVersion: apps/v1
>> kind: Deployment
>> metadata:
>>   name: {{ .Values.apps.applicantsApi.name }}
>>   namespace: {{ include "tp.ns" . }}
>>   labels:
>>     app: {{ .Values.apps.applicantsApi.name }}
>> spec:
>>   replicas: {{ .Values.apps.applicantsApi.replicas }}
>>   selector:
>>     matchLabels:
>>       app: {{ .Values.apps.applicantsApi.name }}
>>   template:
>>     metadata:
>>       labels:
>>         app: {{ .Values.apps.applicantsApi.name }}
>>     spec:
>>       containers:
>>         - name: {{ .Values.apps.applicantsApi.name }}
>>           image: {{ .Values.global.registry }}/{{ .Values.apps.applicantsApi.imageName }}:{{ .Values.global.tag }}
>>           imagePullPolicy: IfNotPresent
>>         ports:
>>           - containerPort: {{ .Values.apps.applicantsApi.containerPort }}
>>             name: http
>>           env:
>>             - name: HostRabbitmq
>>               value: {{ .Values.apps.applicantsApi.env.hostRabbitmq | quote }}
>>             - name: ConnectionString
>>               value: {{ printf "Server=%s;User=%s;Password=%s;Database=%s;" .Values.infra.mssql.saPassword .Values.apps.applicantsApi.env.database | quote }}
>>             resources:
>>               requests:
>>                 cpu: {{ .Values.apps.applicantsApi.requests.cpu | quote }}
>>                 memory: {{ .Values.apps.applicantsApi.requests.memory | quote }}
>>               limits:
>>                 cpu: {{ .Values.apps.applicantsApi.limits.cpu | quote }}
>>                 memory: {{ .Values.apps.applicantsApi.limits.memory | quote }}
>>             readinessProbe:
>>               tcpSocket:

```

2.3 — Templates APPS : Jobs API

Créer templates\app-jobs-api.yaml

@"

{{- if .Values.apps.jobsApi.enabled }}

apiVersion: apps/v1

kind: Deployment

metadata:

name: {{ .Values.apps.jobsApi.name }}

namespace: {{ include "tp.ns" . }}

labels:

app: {{ .Values.apps.jobsApi.name }}

spec:

replicas: {{ .Values.apps.jobsApi.replicas }}

selector:

matchLabels:

app: {{ .Values.apps.jobsApi.name }}

template:

```
metadata:
  labels:
    app: {{ .Values.apps.jobsApi.name }}

spec:
  containers:
    - name: {{ .Values.apps.jobsApi.name }}
      image: {{ .Values.global.registry }}/{{ .Values.apps.jobsApi.imageName }}:{{ .Values.global.tag }}
      imagePullPolicy: IfNotPresent
  ports:
    - containerPort: {{ .Values.apps.jobsApi.containerPort }}
      name: http
  env:
    - name: HostRabbitmq
      value: {{ .Values.apps.jobsApi.env.hostRabbitmq | quote }}
    - name: ConnectionString
      value: {{ printf "Server=sql-data;User=sa;Password=%s;Database=%s;" .Values.infra.mssql.saPassword .Values.apps.jobsApi.env.database | quote }}
  resources:
    requests:
      cpu: {{ .Values.apps.jobsApi.requests.cpu | quote }}
      memory: {{ .Values.apps.jobsApi.requests.memory | quote }}
    limits:
      cpu: {{ .Values.apps.jobsApi.limits.cpu | quote }}
      memory: {{ .Values.apps.jobsApi.limits.memory | quote }}
  readinessProbe:
    tcpSocket:
      port: {{ .Values.apps.jobsApi.containerPort }}
  initialDelaySeconds: 10
```

```
    periodSeconds: 5

    livenessProbe:
        tcpSocket:
            port: {{ .Values.apps.jobsApi.containerPort }}

    initialDelaySeconds: 20
    periodSeconds: 10

---
apiVersion: v1
kind: Service
metadata:
    name: {{ .Values.apps.jobsApi.name }}
    namespace: {{ include "tp.ns" . }}
    labels:
        app: {{ .Values.apps.jobsApi.name }}
spec:
    type: ClusterIP
    selector:
        app: {{ .Values.apps.jobsApi.name }}
    ports:
        - name: http
          port: 80
          targetPort: {{ .Values.apps.jobsApi.containerPort }}
{{- end }}

"@ | Set-Content -Encoding UTF8 .\templates\app-jobs-api.yaml
```

```

PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> @"
>> {{- if .Values.apps.jobsApi.enabled }}
>>   apiVersion: apps/v1
>>   kind: Deployment
>>   metadata:
>>     name: {{ .Values.apps.jobsApi.name }}
>>     namespace: {{ include "tp.ns" . }}
>>     labels:
>>       app: {{ .Values.apps.jobsApi.name }}
>>   spec:
>>     replicas: {{ .Values.apps.jobsApi.replicas }}
>>     selector:
>>       matchLabels:
>>         app: {{ .Values.apps.jobsApi.name }}
>>     template:
>>       metadata:
>>         labels:
>>           app: {{ .Values.apps.jobsApi.name }}
>>       spec:
>>         containers:
>>           - name: {{ .Values.apps.jobsApi.name }}
>>             image: {{ .Values.global.registry }}/{{ .Values.apps.jobsApi.imageName }}:{{ .Values.global.tag }}
>>             imagePullPolicy: IfNotPresent
>>             ports:
>>               - containerPort: {{ .Values.apps.jobsApi.containerPort }}
>>             name: http
>>             env:
>>               - name: HostRabbitmq
>>                 value: {{ .Values.apps.jobsApi.env.hostRabbitmq | quote }}
>>               - name: ConnectionString
>>                 value: {{ printf "Server=%s;User=%s;Password=%s;Database=%s;" .Values.infra.mssql.saPassword .Values.apps.jobsApi.env.database | quote }}
>>             resources:
>>               requests:
>>                 cpu: {{ .Values.apps.jobsApi.requests.cpu | quote }}
>>                 memory: {{ .Values.apps.jobsApi.requests.memory | quote }}
>>               limits:
>>                 cpu: {{ .Values.apps.jobsApi.limits.cpu | quote }}

```

2.4 — Templates APPS : Identity API

Créer templates\app-identity-api.yaml

@"

```

{{- if .Values.apps.identityApi.enabled }}

apiVersion: apps/v1

kind: Deployment

metadata:

  name: {{ .Values.apps.identityApi.name }}

  namespace: {{ include "tp.ns" . }}

  labels:

    app: {{ .Values.apps.identityApi.name }}

spec:

  replicas: {{ .Values.apps.identityApi.replicas }}

  selector:

    matchLabels:

      app: {{ .Values.apps.identityApi.name }}

  template:

    metadata:

```

```
labels:
  app: {{ .Values.apps.identityApi.name }}

spec:
  containers:
    - name: {{ .Values.apps.identityApi.name }}
      image: {{ .Values.global.registry }}/{{ .Values.apps.identityApi.imageName }}:{{ .Values.global.tag }}
      imagePullPolicy: IfNotPresent
  ports:
    - containerPort: {{ .Values.apps.identityApi.containerPort }}
      name: http
  env:
    - name: RedisHost
      value: {{ .Values.apps.identityApi.env.redisHost | quote }}
    - name: HostRabbitmq
      value: {{ .Values.apps.identityApi.env.hostRabbitmq | quote }}
  resources:
    requests:
      cpu: {{ .Values.apps.identityApi.requests.cpu | quote }}
      memory: {{ .Values.apps.identityApi.requests.memory | quote }}
    limits:
      cpu: {{ .Values.apps.identityApi.limits.cpu | quote }}
      memory: {{ .Values.apps.identityApi.limits.memory | quote }}
  readinessProbe:
    tcpSocket:
      port: {{ .Values.apps.identityApi.containerPort }}
    initialDelaySeconds: 10
    periodSeconds: 5
```

```
livenessProbe:  
  tcpSocket:  
    port: {{ .Values.apps.identityApi.containerPort }}  
  initialDelaySeconds: 20  
  periodSeconds: 10  
  
---  
  
apiVersion: v1  
kind: Service  
  
metadata:  
  name: {{ .Values.apps.identityApi.name }}  
  namespace: {{ include "tp.ns" . }}  
  
labels:  
  app: {{ .Values.apps.identityApi.name }}  
  
spec:  
  type: ClusterIP  
  
  selector:  
    app: {{ .Values.apps.identityApi.name }}  
  
  ports:  
    - name: http  
      port: 80  
      targetPort: {{ .Values.apps.identityApi.containerPort }}  
  
{{- end }}  
"@ | Set-Content -Encoding UTF8 .\templates\app-identity-api.yaml
```

```

PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> @"
>> {{- if .Values.apps.identityApi.enabled }}
>> apiVersion: apps/v1
>> kind: Deployment
>> metadata:
>>   name: {{ .Values.apps.identityApi.name }}
>>   namespace: {{ include "tp.ns" . }}
>>   labels:
>>     app: {{ .Values.apps.identityApi.name }}
>> spec:
>>   replicas: {{ .Values.apps.identityApi.replicas }}
>>   selector:
>>     matchLabels:
>>       app: {{ .Values.apps.identityApi.name }}
>>   template:
>>     metadata:
>>       labels:
>>         app: {{ .Values.apps.identityApi.name }}
>>     spec:
>>       containers:
>>         - name: {{ .Values.apps.identityApi.name }}
>>           image: {{ .Values.global.registry }}/{{ .Values.apps.identityApi.imageName }}:{{ .Values.global.tag }}
>>           imagePullPolicy: IfNotPresent
>>           ports:
>>             - containerPort: {{ .Values.apps.identityApi.containerPort }}
>>               name: http
>>             env:
>>               - name: RedisHost
>>                 value: {{ .Values.apps.identityApi.env.redisHost | quote }}
>>               - name: HostRabbitmq
>>                 value: {{ .Values.apps.identityApi.env.hostRabbitmq | quote }}
>>             resources:
>>               requests:
>>                 cpu: {{ .Values.apps.identityApi.requests.cpu | quote }}
>>                 memory: {{ .Values.apps.identityApi.requests.memory | quote }}
>>               limits:
>>                 cpu: {{ .Values.apps.identityApi.limits.cpu | quote }}
>>                 memory: {{ .Values.apps.identityApi.limits.memory | quote }}
>>             readinessProbe:
>>               tcpSocket:
>>                 port: {{ .Values.apps.identityApi.containerPort }}

```

2.5 — Template APPS : WebMVC (+ affinité optionnelle)

Créer templates\app-webmvc.yaml

```

@"

{{- if .Values.apps.webmvc.enabled }}

apiVersion: apps/v1

kind: Deployment

metadata:

  name: {{ .Values.apps.webmvc.name }}

  namespace: {{ include "tp.ns" . }}

  labels:

    app: {{ .Values.apps.webmvc.name }}

spec:

  replicas: {{ .Values.apps.webmvc.replicas }}

  selector:

    matchLabels:

```

```
app: {{ .Values.apps.webmvc.name }}

template:

metadata:

labels:

app: {{ .Values.apps.webmvc.name }}

spec:

{{- if .Values.apps.webmvc.affinity.enabled }}

affinity:

podAffinity:

preferredDuringSchedulingIgnoredDuringExecution:

- weight: 80

podAffinityTerm:

labelSelector:

matchLabels:

app: jobs-api

topologyKey: kubernetes.io/hostname

- weight: 80

podAffinityTerm:

labelSelector:

matchLabels:

app: applicants-api

topologyKey: kubernetes.io/hostname

{{- end }}

containers:

- name: {{ .Values.apps.webmvc.name }}

image: {{ .Values.global.registry }}/{{ .Values.apps.webmvc.imageName }}:{{ .Values.global.tag }}

imagePullPolicy: IfNotPresent
```

```
ports:
  - containerPort: {{ .Values.apps.webmvc.containerPort }}
    name: http

resources:
  requests:
    cpu: {{ .Values.apps.webmvc.requests.cpu | quote }}
    memory: {{ .Values.apps.webmvc.requests.memory | quote }}

  limits:
    cpu: {{ .Values.apps.webmvc.limits.cpu | quote }}
    memory: {{ .Values.apps.webmvc.limits.memory | quote }}

readinessProbe:
  tcpSocket:
    port: {{ .Values.apps.webmvc.containerPort }}

  initialDelaySeconds: 10
  periodSeconds: 5

livenessProbe:
  tcpSocket:
    port: {{ .Values.apps.webmvc.containerPort }}

  initialDelaySeconds: 20
  periodSeconds: 10

---
apiVersion: v1
kind: Service
metadata:
  name: {{ .Values.apps.webmvc.name }}
  namespace: {{ include "tp.ns" . }}

labels:
  app: {{ .Values.apps.webmvc.name }}
```

spec:

type: ClusterIP

selector:

app: {{ .Values.apps.webmvc.name }}

ports:

- name: http

port: 80

targetPort: {{ .Values.apps.webmvc.containerPort }}

{{- end }}

"@ | Set-Content -Encoding UTF8 .\templates\app-webmvc.yaml

```
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> @"
>> {{- if .Values.apps.webmvc.enabled }}
>> apiVersion: apps/v1
>> kind: Deployment
>> metadata:
>>   name: {{ .Values.apps.webmvc.name }}
>>   namespace: {{ include "tp.ns" . }}
>>   labels:
>>     app: {{ .Values.apps.webmvc.name }}
>> spec:
>>   replicas: {{ .Values.apps.webmvc.replicas }}
>>   selector:
>>     matchLabels:
>>       app: {{ .Values.apps.webmvc.name }}
>>   template:
>>     metadata:
>>       labels:
>>         app: {{ .Values.apps.webmvc.name }}
>>     spec:
>>       {{- if .Values.apps.webmvc.affinity.enabled }}
>>       affinity:
>>         podAffinity:
>>           preferredDuringSchedulingIgnoredDuringExecution:
>>             - weight: 80
>>               podAffinityTerm:
>>                 labelSelector:
>>                   matchLabels:
>>                     app: jobs-api
>>                     topologyKey: kubernetes.io/hostname
>>             - weight: 80
>>               podAffinityTerm:
>>                 labelSelector:
>>                   matchLabels:
>>                     app: applicants-api
>>                     topologyKey: kubernetes.io/hostname
>>       {{- end }}
>>       containers:
>>         - name: {{ .Values.apps.webmvc.name }}
>>           image: {{ .Values.global.registry }}/{{ .Values.apps.webmvc.imageName }}:{{ .Values.global.tag }}
>>           imagePullPolicy: IfNotPresent
>>           ports:
```

2.6 — Template APPS : HPA webmvc

Créer templates\app-hpa.yaml

@"

{{- if .Values.apps.hpa.enabled }}

```

apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: {{ .Values.apps.hpa.targetDeployment }}-hpa
  namespace: {{ include "tp.ns" . }}
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: {{ .Values.apps.hpa.targetDeployment }}
  minReplicas: {{ .Values.apps.hpa.minReplicas }}
  maxReplicas: {{ .Values.apps.hpa.maxReplicas }}
metrics:
  - type: Resource
    resource:
      name: cpu
    target:
      type: Utilization
      averageUtilization: {{ .Values.apps.hpa.cpuUtilization }}
{{- end }}

```

```
"@ | Set-Content -Encoding UTF8 .\templates\app-hpa.yaml
```

```

PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> @"
>> {{- if .Values.apps.hpa.enabled }}
>>   apiVersion: autoscaling/v2
>>   kind: HorizontalPodAutoscaler
>>   metadata:
>>     name: {{ .Values.apps.hpa.targetDeployment }}-hpa
>>     namespace: {{ include "tp.ns" . }}
>>   spec:
>>     scaleTargetRef:
>>       apiVersion: apps/v1
>>       kind: Deployment
>>       name: {{ .Values.apps.hpa.targetDeployment }}
>>     minReplicas: {{ .Values.apps.hpa.minReplicas }}
>>     maxReplicas: {{ .Values.apps.hpa.maxReplicas }}
>>   metrics:
>>     - type: Resource
>>       resource:
>>         name: cpu
>>       target:
>>         type: Utilization
>>         averageUtilization: {{ .Values.apps.hpa.cpuUtilization }}
>> {{- end }}
>> "@ | Set-Content -Encoding UTF8 .\templates\app-hpa.yaml

```

2.7 — Template APPS : Ingress HTTPS webmvc

Créer templates\app-ingress.yaml

```
@"  
{%- if .Values.apps.ingress.enabled %}  
apiVersion: networking.k8s.io/v1  
kind: Ingress  
metadata:  
  name: webmvc-ingress  
  namespace: {{ include "tp.ns" . }}  
  annotations:  
    nginx.ingress.kubernetes.io/ssl-redirect: "true"  
spec:  
  ingressClassName: {{ .Values.apps.ingress.className }}  
  tls:  
    - hosts:  
        - {{ .Values.apps.ingress.host }}  
      secretName: {{ .Values.apps.ingress.tlsSecret }}  
  rules:  
    - host: {{ .Values.apps.ingress.host }}  
      http:  
        paths:  
          - path: /  
            pathType: Prefix  
        backend:  
          service:  
            name: {{ .Values.apps.webmvc.name }}  
          port:
```

```

number: 80

{{- end }}

"@ | Set-Content -Encoding UTF8 .\templates\app-ingress.yaml

```

```

PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> @"
>> {{- if .Values.apps.ingress.enabled }}
>> apiVersion: networking.k8s.io/v1
>> kind: Ingress
>> metadata:
>>   name: webmvc-ingress
>>   namespace: {{ include "tp.ns" . }}
>>   annotations:
>>     nginx.ingress.kubernetes.io/ssl-redirect: "true"
>> spec:
>>   ingressClassName: {{ .Values.apps.ingress.className }}
>>   tls:
>>     - hosts:
>>       - {{ .Values.apps.ingress.host }}
>>       secretName: {{ .Values.apps.ingress.tlsSecret }}
>>   rules:
>>     - host: {{ .Values.apps.ingress.host }}
>>       http:
>>         paths:
>>           - path: /
>>             pathType: Prefix
>>             backend:
>>               service:
>>                 name: {{ .Values.apps.webmvc.name }}
>>                 port:
>>                   number: 80
>> {{- end }}
>> "@ | Set-Content -Encoding UTF8 .\templates\app-ingress.yaml

```

2.8 — Vérifier le rendu Helm (sans installer)

```
helm template tp-appscore-apps . --namespace tp-prod | Select-String "kind:|name:" -Context 0,1
```

```

PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> helm template tp-appscore-apps . --namespace tp-prod | Select-String "kind:|name:" -Context 0,1
> kind: Secret
>   metadata:
>     name: sql-secrets
>     namespace: tp-prod
> kind: PersistentVolumeClaim
>   metadata:
>     name: sql-data-pvc
>     namespace: tp-prod
> kind: Service
>   metadata:
>     name: applicants-api
>     namespace: tp-prod
>     - name: http
>       port: 80
> kind: Service
>   metadata:
>     name: identity-api
>     namespace: tp-prod
>     - name: http
>       port: 80
> kind: Service
>   metadata:
>     name: jobs-api
>     namespace: tp-prod
>     - name: http
>       port: 80
> kind: Service
>   metadata:
>     name: webmvc
>     namespace: tp-prod
>     - name: http
>       port: 80
> kind: Service
>   metadata:
>     name: sql-data
>     namespace: tp-prod
>     - name: mssql
>       port: 1433

```

2.9 — Supprimer les apps “YAML” existantes (pour laisser Helm gérer)

```
kubectl -n tp-prod delete deploy webmvc applicants-api jobs-api identity-api
```

```
kubectl -n tp-prod delete svc webmvc applicants-api jobs-api identity-api
```

```
kubectl -n tp-prod delete hpa webmvc-hpa 2>$null
```

```
kubectl -n tp-prod delete ingress webmvc-ingress 2>$null
```

(On ne touche pas l'infra, elle est déjà Helm.)

```
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> kubectl -n tp-prod delete deploy webmvc applicants-api jobs-api identity-api deployment.apps "webmvc" deleted from tp-prod namespace deployment.apps "applicants-api" deleted from tp-prod namespace deployment.apps "jobs-api" deleted from tp-prod namespace deployment.apps "identity-api" deleted from tp-prod namespace PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> kubectl -n tp-prod delete svc webmvc applicants-api jobs-api identity-api service "webmvc" deleted from tp-prod namespace service "applicants-api" deleted from tp-prod namespace service "jobs-api" deleted from tp-prod namespace service "identity-api" deleted from tp-prod namespace PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> kubectl -n tp-prod delete hpa webmvc-hpa 2>$null horizontalpodautoscaler.autoscaling "webmvc-hpa" deleted from tp-prod namespace PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> kubectl -n tp-prod delete ingress webmvc-ingress 2>$null ingress.networking.k8s.io "webmvc-ingress" deleted from tp-prod namespace PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> |
```

2.10 — Installer la release Helm APPS

```
helm install tp-appscore-apps . --namespace tp-prod
```

NB : à partir de ce moment j'ai eu de multiples erreur -> j'ai choisi de faire l'option A

Cas **classique Helm** : un objet K8s (ici le **Secret sql-secrets**) existe déjà dans tp-prod, mais il “appartient” déjà à **un autre release Helm** (tp-appscore-infra). Du coup Helm refuse de l’“adopter” pour tp-appscore-apps.

👉 Donc **ce n'est pas une erreur Kubernetes**, c'est un **conflit d'ownership Helm** via ces annotations :

- meta.helm.sh/release-name: tp-appscore-infra
- alors que toi tu installles tp-appscore-apps

✓ Objectif : installer tp-appscore-apps sans casser tp-appscore-infra

Tu as 3 options propres. Je te donne la plus safe en premier.

Option A (recommandée) : ne PAS gérer sql-secrets dans apps

Si sql-secrets fait partie de l'infra (logique), alors dans ton chart tp-appscore-apps **tu ne dois pas recréer ce Secret**.

1) Trouve le template qui crée sql-secrets dans tp-appscore-apps

Dans ton dossier chart :

```
Select-String -Path .\templates\*.yaml -Pattern "name:\s*sql-secrets" -Context 0,5
```

2) Mets ce secret sous condition (values)

Dans le template du secret, entoure avec :

```
{%- if .Values.sqlSecrets.create %}  
... secret yaml ...  
{%- end %}
```

Dans values.yaml du chart apps :

```
sqlSecrets:  
  create: false  
  name: sql-secrets
```

Et là tes apps référencent juste sql-secrets existant.

 C'est le modèle “infra crée les secrets, apps consomment”.

Option B : installer apps dans un autre namespace

Si tu veux vraiment séparer infra/apps par namespace, alors :

- tp-appscore-infra dans tp-infra
- tp-appscore-apps dans tp-prod

Mais toi tu as déjà infra dans tp-prod, donc ça implique de déplacer / réinstaller l'infra, donc pas idéal.

Option C : “forcer l’adoption” du Secret par apps (dangereux)

Tu peux modifier les annotations du Secret existant pour qu'il appartienne au nouveau release.

 MAIS : tu casses potentiellement le release infra et ses upgrades.

Si tu veux quand même (je déconseille), voilà la logique :

1. vérifier annotations actuelles

```
kubectl -n tp-prod get secret sql-secrets -o yaml | Select-String "meta.helm.sh" -Context 0,2
```

2. patcher les annotations/labels Helm vers tp-appscore-apps
(je te le donnerai si tu décides cette option)
-

Conclusion simple

Ton infra tp-appscore-infra **possède déjà** sql-secrets.

Donc ton chart tp-appscore-apps doit **arrêter d'essayer de le créer**.

Pour ce faire :

Étape 1 — Se placer dans le bon dossier

Fais exactement ceci :

```
cd .\04-helm-chart\tp-appscore
```

Étape 2 — Rechercher sql-secrets dans le chart

Maintenant seulement :

```
Select-String -Path .\templates\*.yaml -Pattern "sql-secrets" -Context 0,5
```

ou plus large (recommandé) :

```
Select-String -Path .\templates\*.yaml -Pattern "name:\s*sql-secrets" -Context 0,5
```

(option choisi)

```
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices> cd .\04-helm-chart\tp-appscore
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> Select-String -Path .\templates\*.yaml -Pattern "name:\s*sql-secrets" -Context 0,5
> templates\infra-mssql.yaml:5:   name: sql-secrets
templates\infra-mssql.yaml:6:   namespace: {{ include "tp.ns" . }}
templates\infra-mssql.yaml:7:type: Opaque
templates\infra-mssql.yaml:8:stringData:
templates\infra-mssql.yaml:9:   SA_PASSWORD: {{ .Values.infra.mssql.saPassword | quote }}
templates\infra-mssql.yaml:10:---
> templates\infra-mssql.yaml:72:           name: sql-secrets
templates\infra-mssql.yaml:73:           key: SA_PASSWORD
templates\infra-mssql.yaml:74:           volumeMounts:
templates\infra-mssql.yaml:75:             - name: sql-data
templates\infra-mssql.yaml:76:               mountPath: /var/opt/mssql
templates\infra-mssql.yaml:77:             resources:
```

Parfait -> on a trouvé **exactement** pourquoi Helm bloque.

Ce que ta sortie prouve

Dans **templates/infra-mssql.yaml**, le chart **crée** le Secret :

- name: sql-secrets (ligne ~5)

et plus bas, il est **utilisé** (ligne ~72).

Donc quand tu fais helm install tp-appscore-apps, Helm essaye de **recréer** sql-secrets... sauf que ce Secret existe déjà et est “possédé” par la release **tp-appscore-infra** (annotation meta.helm.sh/release-name), donc Helm refuse.

 Conclusion : **infra-mssql.yaml ne doit pas être rendu (templated) dans la release “apps”.**

Solution propre (recommandée) : “infra” et “apps” séparés via un flag

Garder infra-mssql.yaml dans le chart, mais l'entourer d'une condition.

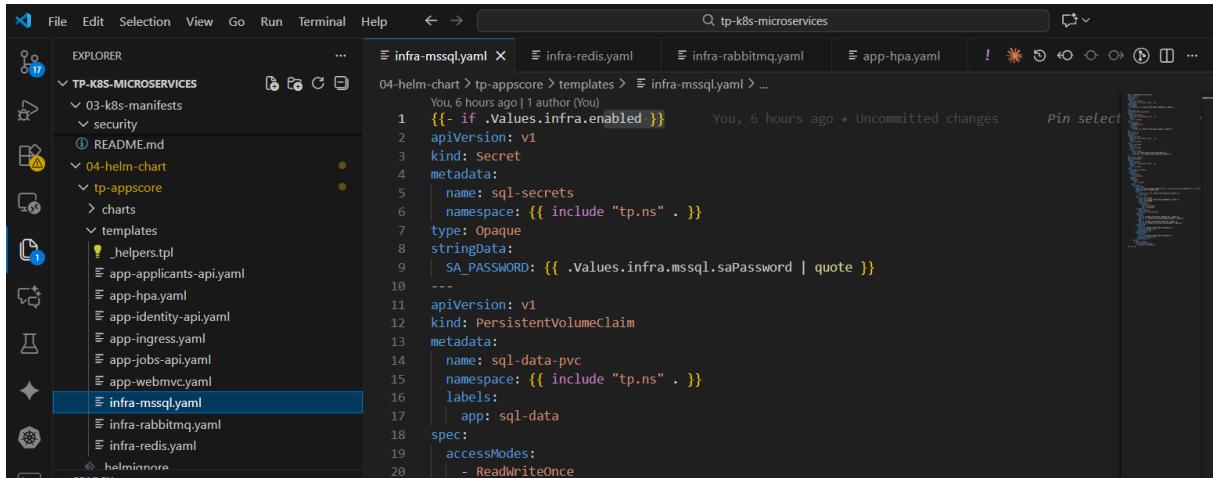
1) Modifie templates/infra-mssql.yaml

Ajoute tout en haut du fichier :

```
{%- if .Values.infra.enabled %}
```

Et tout en bas du fichier :

```
{%- end %}
```



```
1  {{- if .Values.infra.enabled }}          You, 6 hours ago | 1 author (You)
2  apiVersion: v1
3  kind: Secret
4  metadata:
5    name: sql-secrets
6    namespace: {{ include "tp.ns" . }}
7  type: Opaque
8  stringData:
9    SA_PASSWORD: {{ .Values.infra.mssql.saPassword | quote }}
10   ...
11  apiVersion: v1
12  kind: PersistentVolumeClaim
13  metadata:
14    name: sql-data-pvc
15    namespace: {{ include "tp.ns" . }}
16  labels:
17    app: sql-data
18  spec:
19    accessModes:
20      - ReadWriteOnce
```

✓ Comme ça :

- tp-appscore-infra => infra.enabled=true → crée sql-secrets, déploie MSSQL, etc.
- tp-appscore-apps => infra.enabled=false → **ne touche pas** aux secrets infra (pas eu besoin de faire cela mais je laisse au cas ou)

2) Mets un défaut dans values.yaml

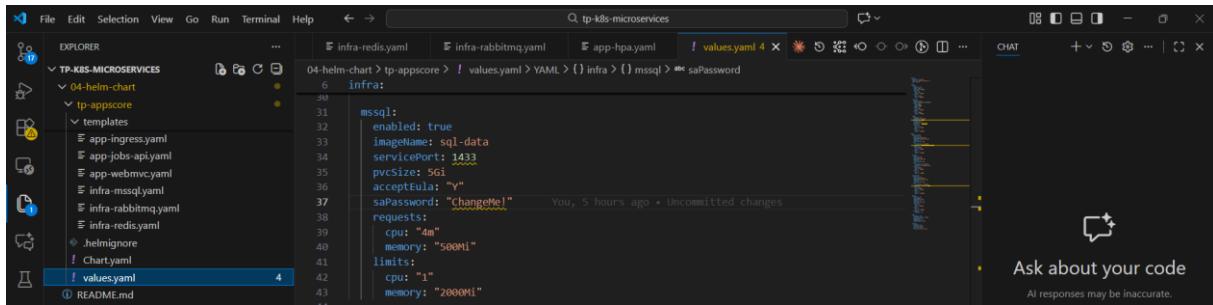
Dans values.yaml (à la racine du chart) :

infra:

```
enabled: true
```

```
mssql:
```

```
saPassword: "ChangeMe!"
```



3) Installe les 2 releases correctement

Infra :

```
helm upgrade --install tp-appscore-infra . -n tp-prod --set infra.enabled=true
```

```
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> helm upgrade --install tp-appscore-infra . -n tp-prod --set infra.enabled=true
Release "tp-appscore-infra" has been upgraded. Happy Helming!
NAME: tp-appscore-infra
LAST DEPLOYED: Sun Dec 28 10:22:40 2025
NAMESPACE: tp-prod
STATUS: deployed
REVISION: 2
DESCRIPTION: Upgrade complete
TEST SUITE: None
```

Apps (nouvelle erreur que j'ai du corriger) :

```
helm upgrade --install tp-appscore-apps . -n tp-prod --set infra.enabled=false
```

```
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> helm upgrade --install tp-appscore-apps . -n tp-prod --set infra.enabled=false
Release "tp-appscore-apps" does not exist. Installing it now.
Error: unable to continue with install: Service "applicants-api" in namespace "tp-prod" exists and cannot be imported into the current release: invalid ownership metadata; annotation validation error: key "meta.helm.sh/release-name" must equal "tp-appscore-apps": current value is "tp-appscore-infra"
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> Select-String -Path .\templates\*.yaml -Pattern "name:s*applicants-api" -Context 0..6
```

Nickel : le problème “sql-secrets” est réglé, mais maintenant on voit le même problème sur **Service applicants-api**.

Pourquoi Helm refuse ?

Parce que applicants-api a été créé par tp-appscore-infra, donc Helm considère qu'il appartient à cette release. Une autre release (tp-appscore-apps) ne peut pas le “récupérer” sans modification, sinon Helm ne saurait plus qui doit le gérer lors des upgrades/uninstalls.

→ Explication simple : la release tp-appscore-infra déploie encore des ressources “apps” (au moins Service applicants-api, et probablement aussi deploy applicants-api, jobs-api, identity-api, webmvc, etc.).

Donc il faut faire **exactement la même séparation** que pour MSSQL... mais cette fois sur les manifests “apps”.

Fix propre : 2 flags infra.enabled et apps.enabled

on veut :

- **Release infra** : ne rend que redis/rabbit/sql/pvc etc.
- **Release apps** : ne rend que les microservices (deployments + services + ingress)

A. Mets ce défaut dans values.yaml (normal déjà fais mais je met au cas ou)

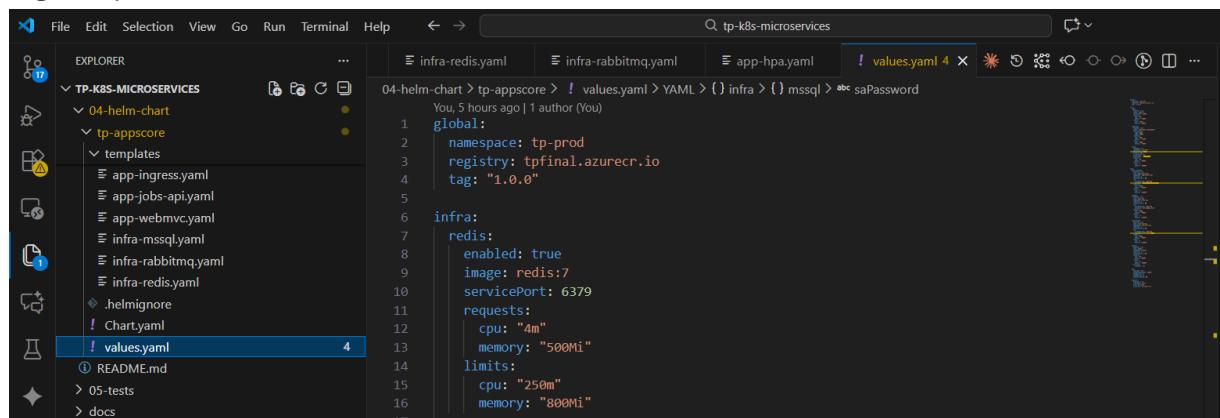
infra:

```
enabled: true
```

apps:

```
enabled: true
```

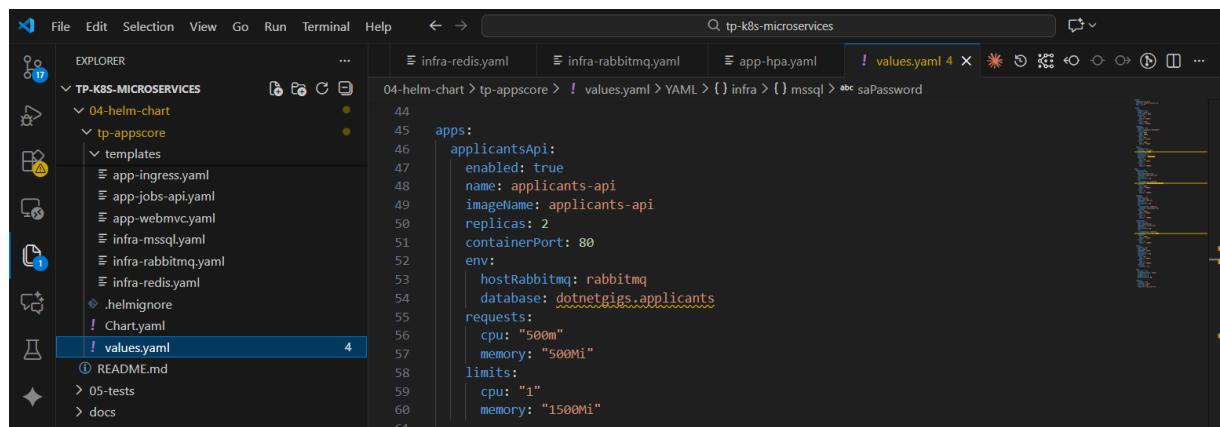
Ligne 8 pour infra



```
global:
  namespace: tp-prod
  registry: tpfinal.azurecr.io
  tag: "1.0.0"

infra:
  redis:
    enabled: true
    image: redis:7
    servicePort: 6379
    requests:
      cpu: "4m"
      memory: "500Mi"
    limits:
      cpu: "250m"
      memory: "800Mi"
```

Ligne 47 pour infra



```
apps:
  applicantsApi:
    enabled: true
    name: applicants-api
    imageName: applicants-api
    replicas: 2
    containerPort: 80
    env:
      hostRabbitmq: rabbitmq
      database: dotnetgigs.applicants
    requests:
      cpu: "500m"
      memory: "500Mi"
    limits:
      cpu: "1"
      memory: "1500Mi"
```

Entoure tes templates “apps”

Dans chaque template qui concerne les apps (services/deployments/ingress des APIs + webmvcs), ajoute :

En haut :

```
{{- if .Values.apps.enabled }}
```

En bas :

```
{{- end }}
```

⚠️ Important : tp-appscore-infra doit avoir apps.enabled=false sinon il “possède” encore les services.

5) corriger fichier par fichier

Récupéré la liste de fichiers templates\ pour savoir quel fichiers **entourer** par apps.enabled :

```
Get-ChildItem .\templates | Select-Object Name
```

```
Select-String -Path .\templates\*.yaml -Pattern "applicants-api|jobs-api|identity-api|webmvc" -Context 0,2
```

```
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> Get-ChildItem .\templates | Select-Object Name
Name
-----
app-applicants-api.yaml
app-hpa.yaml
app-identity-api.yaml
app-ingress.yaml
app-jobs-api.yaml
app-webmvc.yaml
infra-mssql.yaml
infra-rabbitmq.yaml
infra-redis.yaml
_helpers.tpl

PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> Select-String -Path .\templates\*.yaml -Pattern "applicants-api|jobs-api|identity-api|webmvc" -Context 0,2
> templates\app-ingress.yaml:5:   name: webmvc-ingress
> templates\app-ingress.yaml:6:   namespace: {{ include "tp.ns" . }}
> templates\app-ingress.yaml:7:   annotations:
> templates\app-ingress.yaml:23:           name: {{ .Values.apps.webmvc.name }}
> templates\app-ingress.yaml:24:           port:
> templates\app-ingress.yaml:25:           number: 80
> templates\app-webmvc.yaml:1:{{- if .Values.apps.webmvc.enabled }}
> templates\app-webmvc.yaml:2:apiVersion: apps/v1
> templates\app-webmvc.yaml:3:kind: Deployment
> templates\app-webmvc.yaml:5:   name: {{ .Values.apps.webmvc.name }}
> templates\app-webmvc.yaml:6:   namespace: {{ include "tp.ns" . }}
> templates\app-webmvc.yaml:7:   labels:
> templates\app-webmvc.yaml:8:   app: {{ .Values.apps.webmvc.name }}
> templates\app-webmvc.yaml:9:spec:
> templates\app-webmvc.yaml:10:   replicas: {{ .Values.apps.webmvc.replicas }}
> templates\app-webmvc.yaml:11:   selector:
> templates\app-webmvc.yaml:12:     matchLabels:
> templates\app-webmvc.yaml:13:   app: {{ .Values.apps.webmvc.name }}
> templates\app-webmvc.yaml:14:   template:
> templates\app-webmvc.yaml:15:     metadata:
> templates\app-webmvc.yaml:17:       app: {{ .Values.apps.webmvc.name }}
```

Diagnostic

- **On as bien séparé infra (infra-mssql.yaml, infra-rabbitmq.yaml, infra-redis.yaml)**
- **On as des templates apps séparés (app-*yaml)**

- MAIS l'install “infra” a déjà créé au moins un Service applicants-api → donc les templates apps sont rendus quand tu déploies infra (par défaut), ou bien apps.enabled est resté à true.

👉 Il manque une garde globale apps.enabled autour de **tous** les fichiers app-*yaml (pas seulement webmvc).

Correctif propre et durable

1) Ajoute un flag global apps.enabled (déjà fais mais je laisse au cas ou)

Dans values.yaml (racine du chart), assure-toi d'avoir :

infra:

 enabled: true

apps:

 enabled: true

 applicantsApi:

 enabled: true

 identityApi:

 enabled: true

 jobsApi:

 enabled: true

 webmvc:

 enabled: true

(Les sous-flags existent peut-être déjà chez toi, mais le principal c'est apps.enabled.)

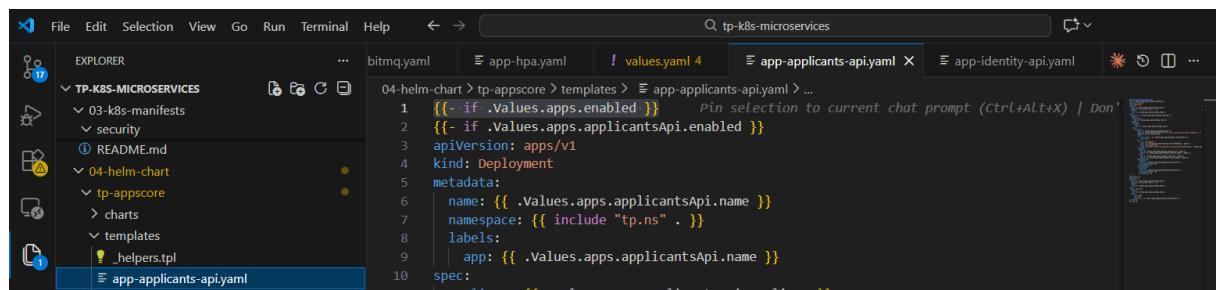
2) *Entoure TOUS les templates “app-*” avec if .Values.apps.enabled

* Structure à appliquer: faire un double if (imbriqué) valeur en exemple disponible dans les captures d'écran :

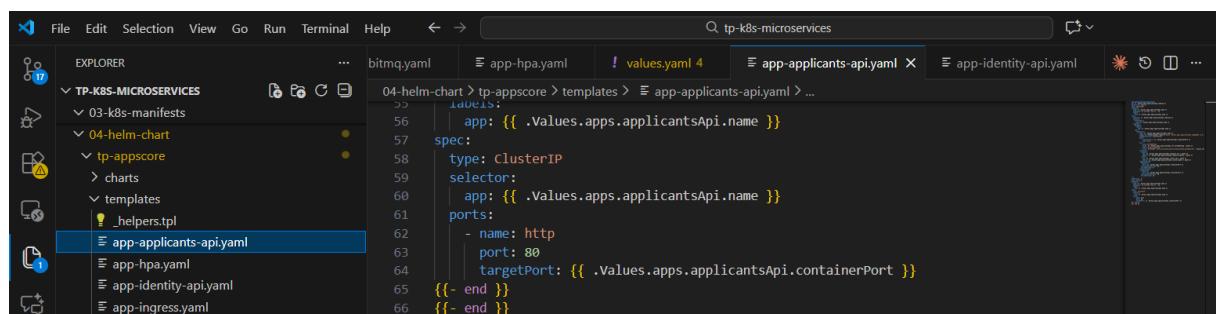
```
{ {- if .Values.apps.enabled }}  
{ {- if .Values.apps.webmvc.enabled }}  
...  
{ {- end }}  
{ {- end }}
```

Fichiers concernés

- templates/app-applicants-api.yaml

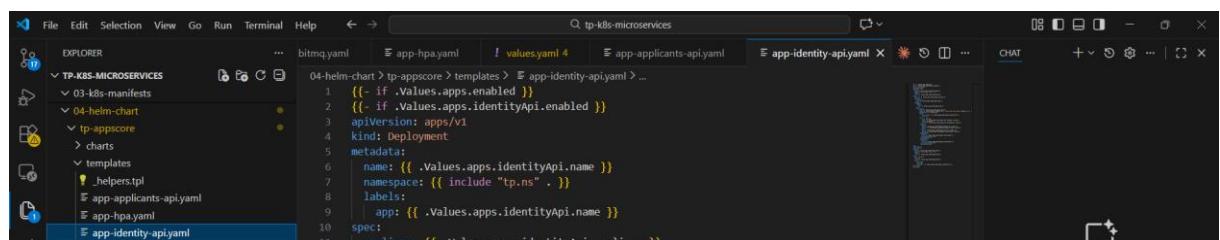


```
04-helm-chart > tp-appscore > templates > app-applicants-api.yaml ...  
1  { {- if .Values.apps.enabled }}  
2  { {- if .Values.apps.applicantsApi.enabled }}  
3  apiVersion: apps/v1  
4  kind: Deployment  
5  metadata:  
6    name: {{ .Values.apps.applicantsApi.name }}  
7    namespace: {{ include "tp.ns" . }}  
8  labels:  
9    app: {{ .Values.apps.applicantsApi.name }}  
10 spec:  
11   replicas: {{ .Values.apps.applicantsApi.replicas }}
```



```
04-helm-chart > tp-appscore > templates > app-applicants-api.yaml ...  
56  labels:  
57    app: {{ .Values.apps.applicantsApi.name }}  
58  spec:  
59    type: ClusterIP  
60    selector:  
61      app: {{ .Values.apps.applicantsApi.name }}  
62    ports:  
63      - name: http  
64        port: 80  
65        targetPort: {{ .Values.apps.applicantsApi.containerPort }}  
66  {{- end }}  
{{- end }}
```

- templates/app-identity-api.yaml



```
04-helm-chart > tp-appscore > templates > app-identity-api.yaml ...  
1  { {- if .Values.apps.enabled }}  
2  { {- if .Values.apps.identityApi.enabled }}  
3  apiVersion: apps/v1  
4  kind: Deployment  
5  metadata:  
6    name: {{ .Values.apps.identityApi.name }}  
7    namespace: {{ include "tp.ns" . }}  
8  labels:  
9    app: {{ .Values.apps.identityApi.name }}  
10 spec:  
11   replicas: {{ .Values.apps.identityApi.replicas }}
```

```

    ...
    ports:
      - name: http
        port: 80
        targetPort: {{ .values.apps.identityApi.containerPort }}
    ...
  
```

- templates/app-jobs-api.yaml
- templates/app-webmvc.yaml (déjà conditionné, mais on va l'imbriquer proprement)
- templates/app-ingress.yaml
- templates/app-hpa.yaml

NB : Les captures sont là pour démontrer, les modifications sont à faire dans tous les autres fichiers

3) Entoure TOUS les templates “infra-*” avec if .Values.infra.enabled

Tu l'as déjà fait pour infra-mssql.yaml. Fais pareil pour :

- infra-rabbitmq.yaml

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: rabbitmq
  namespace: {{ include "tp.ns" . }}
spec:
  replicas: 1
  selector:
    matchLabels:
      app: rabbitmq
  template:
    metadata:
      labels:
        app: rabbitmq
  spec:
    ...
  
```

- infra-redis.yaml

```

1  {{> if .Values.infra.enabled }} You, 5 hours ago | author (You)
2    apiVersion: apps/v1
3    kind: Deployment
4    metadata:
5      name: user-data
6      namespace: {{> include "tp.ns" . }}
7      labels:
8        app: user-data
9    spec:
10      replicas: 1
11      selector:
12        matchLabels:
13          app: user-data
14      template:
15        metadata:
16          labels:
17            app: user-data
18      spec:
19        containers:

```

✓ Vérifie que c'est bien séparé AVANT de redéployer

1) Infra ne doit PLUS sortir applicants-api

helm template tp-appscore-infra . -n tp-prod --set infra.enabled=true --set apps.enabled=false | Select-String "applicants-api"

→ doit être vide.

2) Apps doit sortir applicants-api

helm template tp-appscore-apps . -n tp-prod --set infra.enabled=false --set apps.enabled=true | Select-String "applicants-api"

→ doit afficher des ressources.

```

PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> helm template tp-appscore-infra . -n tp-prod
--set infra.enabled=true --set apps.enabled=false | Select-String "applicants-api"
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> helm template tp-appscore-apps . -n tp-prod
--set infra.enabled=false --set apps.enabled=true | Select-String "applicants-api"

# Source: tp-appscore/templates/app-applicants-api.yaml
name: applicants-api
  app: applicants-api
  app: applicants-api
# Source: tp-appscore/templates/app-applicants-api.yaml
name: applicants-api
  app: applicants-api
  app: applicants-api
  app: applicants-api
  - name: applicants-api
    image: tpfinal.azurecr.io/applicants-api:1.0.0
    app: applicants-api

```

⚠ Très important : on a déjà des apps “possédées” par tp-appscore-infra

Même après correction, si j'essai d'installer tp-appscore-apps, Helm va encore se plaindre **tant que les objets existent et appartiennent à infra**.

Donc il faut faire une étape de “nettoyage ownership” (propre) :

Option la plus propre : laisser infra désinstaller les apps

1. Mets à jour **infra** avec apps.enabled=false (ça va retirer les apps de la release infra)

```
helm upgrade tp-appscore-infra . -n tp-prod --set infra.enabled=true --set apps.enabled=false
```

```
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> helm upgrade tp-appscore-infra . -n tp-prod --set infra.enabled=true --set apps.enabled=false
Release "tp-appscore-infra" has been upgraded. Happy Helming!
NAME: tp-appscore-infra
LAST DEPLOYED: Sun Dec 28 10:42:20 2025
NAMESPACE: tp-prod
STATUS: deployed
REVISION: 3
DESCRIPTION: Upgrade complete
TEST SUITE: None
```

2. Vérifie que les services apps ont disparu (ou au moins applicants-api)

```
kubectl -n tp-prod get svc applicants-api jobs-api identity-api webmvc
```

```
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> kubectl -n tp-prod get svc applicants-api job-s-api identity-api webmvc
Error from server (NotFound): services "applicants-api" not found
Error from server (NotFound): services "jobs-api" not found
Error from server (NotFound): services "identity-api" not found
Error from server (NotFound): services "webmvc" not found
```

3. Puis installe apps

```
helm upgrade --install tp-appscore-apps . -n tp-prod --set infra.enabled=false --set apps.enabled=true ( j'ai encore eu une erreur)
```

```
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> helm upgrade --install tp-appscore-apps . -n tp-prod --set infra.enabled=false --set apps.enabled=true
Release "tp-appscore-apps" does not exist. Installing it now.
Error: unable to continue with install: HorizontalPodAutoscaler "webmvc-hpa" in namespace "tp-prod" exists and cannot be imported into the current release: invalid ownership metadata; annotation validation error: key "meta.helm.sh/release-name" must equal "tp-appscore-apps": current value is "tp-appscore-infra"
```

C'est la méthode “Helm propre” : l'ancien owner supprime, le nouveau owner crée.

Bilan :

- apps.enabled=false dans **infra** → les Service apps ont bien été supprimés (ils n'existent plus).
- Mais il reste un **objet orphelin** appartenant encore à tp-appscore-infra : **HorizontalPodAutoscaler webmvc-hpa**.

C'est normal si le template HPA n'était pas correctement “gated” (ou si une valeur le rend quand même).

Vérifie si webmvc-hpa est encore dans la sortie Helm infra

On veut savoir si la release infra **rende encore** un HPA malgré apps.enabled=false.

```
helm template tp-appscore-infra . -n tp-prod --set infra.enabled=true --set  
apps.enabled=false | Select-String "webmvc-hpa"
```

```
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart> helm template tp-appscore-infra . -n tp-prod  
--set infra.enabled=true --set apps.enabled=false | Select-String "webmvc-hpa"  
name: webmvc-hpa
```

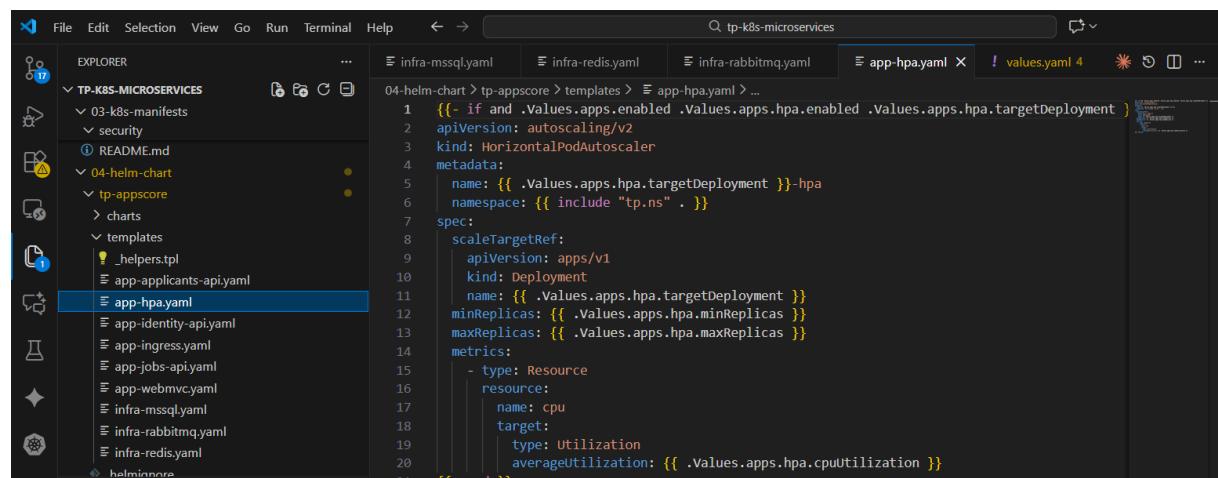
Fix durable : mettre app-hpa.yaml sous apps.enabled

Dans templates/app-hpa.yaml, entoure tout le fichier :

```
{%- if .Values.apps.enabled %}  
  
# ... contenu du HPA ...  
  
{%- end %}
```

Et si tu as des flags par app, double condition (recommandé) : j'ai choisi de faire cette solution

```
{%- if and .Values.apps.enabled.Values.apps.hpa.enabled.Values.apps.hpa.targetDeployment %}  
  
...  
  
{%- end %}
```



The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows the project structure under "TP-K8S-MICROSERVICES". The "app-hpa.yaml" file is currently selected.
- Terminal:** Shows the command "helm template tp-appscore-infra . -n tp-prod --set infra.enabled=true --set apps.enabled=false | Select-String "webmvc-hpa"" being run.
- Code Editor:** Displays the YAML content of "app-hpa.yaml". The code uses a double condition to check if both .Values.apps.enabled and .Values.apps.hpa.enabled are true before applying the targetDeployment setting.

```
1  {{- if and .Values.apps.enabled .Values.apps.hpa.enabled .values.apps.hpa.targetDeployment }}  
2    apiVersion: autoscaling/v2  
3    kind: HorizontalPodAutoscaler  
4    metadata:  
5      name: {{ .Values.apps.hpa.targetDeployment }}-hpa  
6      namespace: {{ include "tp.ns" . }}  
7    spec:  
8      scaleTargetRef:  
9        apiVersion: apps/v1  
10       kind: Deployment  
11       name: {{ .Values.apps.hpa.targetDeployment }}  
12       minReplicas: {{ .Values.apps.hpa.minReplicas }}  
13       maxReplicas: {{ .Values.apps.hpa.maxReplicas }}  
14     metrics:  
15       - type: Resource  
16         resource:  
17           name: cpu  
18           target:  
19             type: Utilization  
20             averageUtilization: {{ .Values.apps.hpa.cpuUtilization }}
```

Pourquoi il reste un HPA alors que les Services ont disparu ?

Parce que Helm supprime **uniquement** ce qui faisait partie du rendu **de la release au moment de l'upgrade**.

Si ton HPA était rendu avant, mais plus maintenant, il peut rester si :

- il n'a pas été "touché" par la modification de rendu (cas rare)
- ou tu as plusieurs HPA/conditions différentes
- ou le HPA a été créé hors Helm/patché manuellement
- ou le template app-hpa.yaml n'a pas été protégé correctement à l'époque

Dans tous les cas, le clean ciblé kubectl delete hpa webmvc-hpa est la bonne action.

Ensuite, quoi faire maintenant dans ton cluster ?

Mets à jour infra en gardant apps désactivé :

```
helm upgrade tp-appscore-infra . -n tp-prod --set infra.enabled=true --set apps.enabled=false
```

```
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> helm upgrade tp-appscore-infra . -n tp-prod --set infra.enabled=true --set apps.enabled=false
Release "tp-appscore-infra" has been upgraded. Happy Helming!
NAME: tp-appscore-infra
LAST DEPLOYED: Sun Dec 28 10:54:54 2025
NAMESPACE: tp-prod
STATUS: deployed
REVISION: 4
DESCRIPTION: Upgrade complete
TEST SUITE: None
```

Installe apps :

```
helm upgrade --install tp-appscore-apps . -n tp-prod --set infra.enabled=false --set apps.enabled=true
```

```
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> helm upgrade --install tp-appscore-apps . -n tp-prod --set infra.enabled=false --set apps.enabled=true
Release "tp-appscore-apps" does not exist. Installing it now.
NAME: tp-appscore-apps
LAST DEPLOYED: Sun Dec 28 10:55:38 2025
NAMESPACE: tp-prod
STATUS: deployed
REVISION: 1
DESCRIPTION: Install complete
TEST SUITE: None
```

✓ Vérification rapide

kubectl -n tp-prod get hpa

kubectl -n tp-prod get svc

helm list -n tp-prod

```
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> kubectl -n tp-prod get hpa
NAME          REFERENCE   TARGETS    MINPODS   MAXPODS   REPLICAS   AGE
webmvc-hpa   Deployment/webmvc  cpu: <unknown>/60%  2           6           0           31s
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> kubectl -n tp-prod get svc
NAME          TYPE        CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
applicants-api ClusterIP  10.110.149.24  <none>           80/TCP          31s
identity-api   ClusterIP  10.102.43.31   <none>           80/TCP          31s
jobs-api       ClusterIP  10.106.172.235 <none>           80/TCP          31s
rabbitmq       ClusterIP  10.98.32.229  <none>           5672/TCP,15672/TCP  2d7h
sql-data        ClusterIP  10.96.1.125   <none>           1433/TCP        2d7h
user-data       ClusterIP  10.98.161.39  <none>           6379/TCP        2d7h
webmvc         ClusterIP  10.103.133.153 <none>           80/TCP          31s
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> helm list -n tp-prod
NAME            NAMESPACE      REVISION      UPDATED             STATUS      CHART          APP VERSION
tp-appscore-apps  tp-prod        1           2025-12-28 10:55:38 +0100 CET  deployed   tp-appscore-0.1.0  1.16.0
tp-appscore-infra tp-prod        4           2025-12-28 10:54:54 +0100 CET  deployed   tp-appscore-0.1.0  1.16.0
```

5) Check final “tout est OK”

Fais ces 4 commandes :

kubectl -n tp-prod get pods

kubectl -n tp-prod top pods

```
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> kubectl -n tp-prod get pods
NAME          READY   STATUS    RESTARTS   AGE
applicants-api-79b87859f-q4mkj  1/1     Running   0          7m6s
applicants-api-79b87859f-qxn9h  1/1     Running   0          7m6s
identity-api-7bbc6f6cb8-5n27m  1/1     Running   0          7m6s
identity-api-7bbc6f6cb8-8xs9l  1/1     Running   0          7m6s
jobs-api-d9fcf88dc-8pnv6     1/1     Running   0          7m6s
jobs-api-d9fcf88dc-nxb99     1/1     Running   0          7m6s
rabbitmq-78ccfb5c7b-qnxvm   1/1     Running   0          2d7h
sql-data-0                   1/1     Running   0          2d7h
user-data-7b6897b6c7-xkqst   1/1     Running   0          2d7h
webmvc-579b6d8484-2gbp4     1/1     Running   0          7m6s
webmvc-579b6d8484-xx45w     1/1     Running   0          7m6s
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> kubectl -n tp-prod top pods
NAME          CPU(cores)   MEMORY(bytes)
applicants-api-79b87859f-q4mkj  2m          44Mi
applicants-api-79b87859f-qxn9h  2m          44Mi
identity-api-7bbc6f6cb8-5n27m  2m          46Mi
identity-api-7bbc6f6cb8-8xs9l  2m          46Mi
jobs-api-d9fcf88dc-8pnv6     2m          42Mi
jobs-api-d9fcf88dc-nxb99     2m          42Mi
rabbitmq-78ccfb5c7b-qnxvm   4m          135Mi
sql-data-0                   12m         1261Mi
user-data-7b6897b6c7-xkqst   3m          3Mi
webmvc-579b6d8484-2gbp4     1m          36Mi
webmvc-579b6d8484-xx45w     1m          36Mi
```

kubectl -n tp-prod describe hpa webmvc-hpa

kubectl -n tp-prod get ingress

```
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> kubectl -n tp-prod describe hpa webmvc-hpa
Name:          webmvc-hpa
Namespace:    tp-prod
Labels:        app.kubernetes.io/managed-by=Helm
Annotations:  meta.helm.sh/release-name: tp-appscore-apps
               meta.helm.sh/release-namespace: tp-prod
CreationTimestamp: Sun, 28 Dec 2025 10:55:38 +0100
Reference:    Deployment/webmvc
Metrics:      ( current / target )
  resource cpu on pods  (as a percentage of request):  25% (1m) / 60%
Min replicas: 2
Max replicas: 6
Deployment pods: 2 current / 2 desired
Conditions:
  Type      Status  Reason           Message
  ----      ----   ----           -----
  AbleToScale  True    ReadyForNewScale recommended size matches current size
  ScalingActive True    ValidMetricFound  the HPA was able to successfully calculate a replica count from cpu resource utilization (percentage of request)
  ScalingLimited True    TooFewReplicas   the desired replica count is less than the minimum replica count
Events:        <none>
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> kubectl -n tp-prod get ingress
NAME          CLASS   HOSTS   ADDRESS      PORTS   AGE
webmvc-ingress nginx  tp.local  localhost  80, 443  7m7s
```

Tout est **parfait** ✅ — on a maintenant une stack Kubernetes **propre, scalable, observable**.

HPA fonctionne vraiment (et ce que ça signifie)

On voit :

- **Metrics OK** : 25% (1m) / 60%
- **ScalingActive = True (ValidMetricFound)** ✅ → le HPA arrive à lire les métriques CPU
- **Requests CPU = 4m** ✅ → indispensable pour calculer le %
- **2 current / 2 desired** ✅

La ligne importante :

ScalingLimited: TooFewReplicas

“desired replica count is less than the minimum replica count”

Ça veut dire :

- Avec la charge actuelle (faible), Kubernetes voudrait probablement descendre à **1 replica**
- Mais ton **minReplicas=2** l'en empêche
 - Donc il reste à **2 pods**, ce qui est ce que tu veux en général.

👉 Conclusion : ton HPA est opérationnel et sain.

2) Pourquoi ton CPU “%” est 25% alors que kubectl top montre ~1m ?

Parce que le HPA calcule le % **par rapport au request** :

- Request CPU = **4m**
- Consommation actuelle = **1m**
- $1m / 4m = 25\%$

C'est exactement ce que tu vois.

3) Ce que tu nous avons maintenant (niveau réel)

Mise en place d'un pattern utilisé en entreprise :

Deux releases Helm avec ownership clean :

- tp-appscore-infra (stateful / composants partagés)
- tp-appscore-apps (stateless / microservices + HPA + ingress)

Autoscaling fonctionnel

Ingress HTTPS/443 et host tp.local

Metrics Server OK

Services + pods OK

Ressources requests/limits cohérentes

Franchement, ça correspond à un niveau **DevOps / Cloud Engineer intermédiaire**, parce que tu as résolu des problèmes que beaucoup ne savent pas gérer (dis par chatgpt – est-ce réellement vrai ?) :

- conflits Helm ownership
- séparation infra/apps
- HPA + requests CPU
- debug réseau/DNS intra cluster
- observabilité (metrics + logs)

4) “Quoi observer maintenant” dans l'interface (ce qui a du sens)

Vu l'état actuel, les 3 observations utiles sont :

HPA / Scaling

- Est-ce que la charge monte → est-ce que le nombre de replicas suit ?
- Observe :
 - current / target
 - desired vs min/max

→ Si tu veux une démo : provoquer la charge.

B) Santé applicative (tp-prod)

- restarts
- readiness
- latences/erreurs côté ingress (Kibana)

C) Capacité / dimensionnement

- requests vs usage :
 - tu vois déjà que webmvc consomme ~1m pour un request 4m → OK
 - SQL consomme beaucoup de mémoire → normal

5) Si tu veux “voir” le scaling en live (preuve visuelle)

Tu peux faire un test simple.

Service webmvc expose 80/TCP (HTTP), pas 443.

<https://tp.local> (Ingress) = 443 ✓

webmvc.tp-prod.svc.cluster.local (Service interne) = 80 ✓

Donc depuis le pod, il faut appeler HTTP (ou préciser le port).

✓ Commande correcte (interne au cluster)

Génère la charge depuis Kubernetes (meilleure pratique).

kubectl -n tp-prod run loadtest --image=curlimages/curl:8.5.0 -it --rm -- sh

Puis dans le pod :

```
for i in $(seq 1 500); do
```

```
    curl -s http://webmvc.tp-prod.svc.cluster.local/ > /dev/null
```

```
PS C:\Users\Dimetri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> kubectl -n tp-prod run loadtest --image=curlimages/curl:8.5.0 -it --rm -- sh
All commands and output from this session will be recorded in container logs
, including credentials and sensitive information passed through the command
prompt.
If you don't see a command prompt, try pressing enter.
~ $ for i in $(seq 1 500); do
>     curl -s http://webmvc.tp-prod.svc.cluster.local/ > /dev/null
> done
~ $ |
```

Là, tu testes :

- réseau interne
 - service discovery
 - HPA en conditions réelles
-

Pendant le test, observe le scaling

Dans un autre terminal :

```
kubectl -n tp-prod get hpa webmvc-hpa -w
```

Tu devrais voir :

- le % CPU monter
- desired augmenter
- des pods supplémentaires apparaître

```
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> kubectl -n tp-prod get hpa webmvc-hpa -w
NAME      REFERENCE   TARGETS          MINPODS   MAXPODS   REPLICAS   AGE
webmvc-hpa Deployment/webmvc  cpu: 25%/60%  2         6          2          6h44m
webmvc-hpa Deployment/webmvc  cpu: 1775%/60% 2         6          2          6h44m
webmvc-hpa Deployment/webmvc  cpu: 25%/60%  2         6          4          6h45m
webmvc-hpa Deployment/webmvc  cpu: 25%/60%  2         6          6          6h46m
```

Tu devrais voir le CPU % et le nombre de replicas augmenter (si assez de charge).

Si on veut tester le chemin “Ingress HTTPS” depuis le cluster

Là, on doit viser l'ingress-controller (ou le host), pas le service.

Le plus simple (si DNS/hosts dans le pod ne résout pas tp.local) :

1. récupère l'IP du service ingress-nginx-controller

```
kubectl -n ingress-nginx get svc
```

```
PS C:\Users\Dimitri ANGELY\Desktop\laboratoire\Leclerc-vincent\tp-k8s-microservices\04-helm-chart\tp-appscore> kubectl -n ingress-nginx get svc
NAME           TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)          AGE
ingress-nginx-controller  LoadBalancer  10.99.57.136  <none>        80:32190/TCP,443:31103/TCP  4d5h
ingress-nginx-admission   ClusterIP   10.98.44.58   <none>        443/TCP         4d5h
```

2. puis dans le pod :

```
curl -k -H "Host: tp.local" https://<IP_INGRESS>/ ->
```

dans mon cas

```
curl -k -H "Host: tp.local" https://10.99.57.136/
```

Mais pour provoquer le HPA, le plus direct et propre est **HTTP vers le Service**, comme au-dessus.

```
Windows PowerShell

~ $ curl -k -H "Host: tp.local" https://10.99.57.136/
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>DotNetGigs - Error</title>

        <link rel="stylesheet" href="https://ajax.aspnetcdn.com/ajax/bootstrap/3.3.7/css/bootstrap.min.css" />
<meta name="x-stylesheet-fallback-test" content="" class="sr-only" /><script>!function(a,b,c,d){var e,f=document,g=f.getElementsByTagName("SCRIPT"),h=g[g.length-1].previousElementSibling,i=f.defaultView&&f.defaultView.getComputedStyle?f.defaultView.getComputedStyle(h):h.currentStyle;if(i&&i[a]!==b)for(e=0;e<c.length;e++)f.write('<link href="'+c[e]+'" '+d+'"/>')}"position","absolute",["\lib\bootstrap\dist\css\bootstrap.min.css"], "rel=\u0022stylesheet\u0022");</script>
        <link rel="stylesheet" href="/css/site.min.css?v=d54Uo600LFrPD8938Lds-OCM37D_Tfrzs_T0wjuZb5M" />

</head>
<body>
    <nav class="navbar navbar-inverse navbar-fixed-top">
        <div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-toggle" data-toggle="col

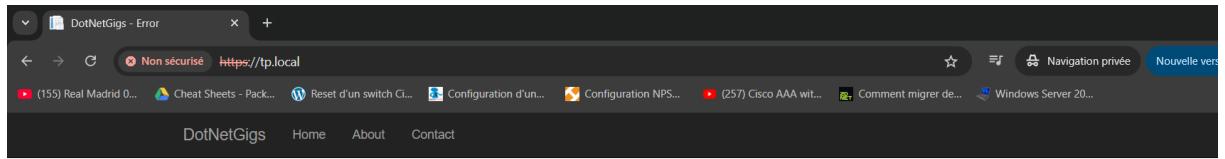
<h1 class="text-danger">Error.</h1>
<h2 class="text-danger">An error occurred while processing your request.</h2>

<p>
    <strong>Request ID:</strong> <code>0HNI5UT4U1NNG:00000001</code>
</p>

<h3>Development Mode</h3>
<p>
    Swapping to <strong>Development</strong> environment will display more detailed information about the error that occurred.
</p>
<p>
    <strong>Development environment should not be enabled in deployed applications</strong>, as it can result in sensitive information from exceptions being displayed to end users. For local debugging, development environment can be enabled by setting the <strong>ASPNETCORE_ENVIRONMENT</strong> environment variable to <strong>Development</strong>, and restarting the application.
</p>

<hr />
<footer>
    <p>&copy; 2025 - DotNetGigs</p>
</footer>
</div>
```

Qui correspond à la capture d'écran ci-dessous dans le navigateur :



Error.

An error occurred while processing your request.

Request ID: 0HN13VMEH1KR4:00000001

Development Mode

Swapping to **Development** environment will display more detailed information about the error that occurred.

Development environment should not be enabled in deployed applications, as it can result in sensitive information from exceptions being displayed to end users. For local debugging, development environment can be enabled by setting the **ASPNETCORE_ENVIRONMENT** environment variable to **Development**, and restarting the application.

© 2025 - DotNetGigs

Le tp est présent terminé.