INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES HAUTS-DE-FRANCE

Université Polytechnique HAUTS-DE-FRANCE

Architecture Orientée Services :

TP Kubernetes EDA

Simon Dumoulin

FISE ICY 5A

2025-2026

Sommaire

Étape 1 : Manipulations

Gestion de minikube

Avant toute chose, on démarre minikube.

```
ubuntu@ubuntu2204:~$ minikube start
😄  minikube v1.37.0 on Ubuntu 24.04 (vbox/amd64)
✨  Automatically selected the docker driver. Other choices: none, ssh
📌  Using Docker driver with root privileges
👍  Starting "minikube" primary control-plane node in "minikube" cluster
🚜  Pulling base image v0.0.48 ...
💾  Downloading Kubernetes v1.34.0 preload ...
    > preloaded-images-k8s-v18-v1...:  337.07 MiB / 337.07 MiB  100.00% 13.54 M
    > gcr.io/k8s-minikube/kicbase...:  488.52 MiB / 488.52 MiB  100.00% 9.03 Mi
🔥  Creating docker container (CPUs=2, Memory=3072MB) ...
🐳  Preparing Kubernetes v1.34.0 on Docker 28.4.0 ...
🔗  Configuring bridge CNI (Container Networking Interface) ...
🔎  Verifying Kubernetes components...
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
🌟  Enabled addons: storage-provisioner, default-storageclass
🏄  Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

Vérifiez que minikube pointe correctement vers le moteur docker

```
ubuntu@ubuntu2204:~$ minikube profile list
```

PROFILE	DRIVER	RUNTIME	IP	VERSION	STATUS	NODES	ACTIVE PROFILE	ACTIVE KUBECONTEXT
minikube	docker	docker	192.168.49.2	v1.34.0	OK	1	*	*

En regardant la colonne “DRIVER”, on voit bien qu’elle indique “docker”.

Grâce à la commande ci-dessous, on peut configurer notre terminal pour que le client Docker local communique directement avec le démon Docker situé à l'intérieur du conteneur Minikube, plutôt qu'avec le démon de notre machine hôte.

```
ubuntu@ubuntu2204:~$ eval $(minikube docker-env)
```

Durant la phase de développement du projet, cela nous a permis de build des images dans kubernetes en local plutôt que de devoir passer par DockerHub (push à chaque modification).

Quels sont les addons actuellement installé ?

```
ubuntu@ubuntu2204:~$ minikube addons list
```

ADDON NAME	PROFILE	STATUS	MAINTAINER
ambassador	minikube	disabled	3rd party (Ambassador)
amd-gpu-device-plugin	minikube	disabled	3rd party (AMD)
auto-pause	minikube	disabled	minikube
cloud-spanner	minikube	disabled	Google
csi-hostpath-driver	minikube	disabled	Kubernetes
dashboard	minikube	disabled	Kubernetes
default-storageclass	minikube	enabled ✅	Kubernetes
efk	minikube	disabled	3rd party (Elastic)
freshpod	minikube	disabled	Google
gcp-auth	minikube	disabled	Google
gvisor	minikube	disabled	minikube
headlamp	minikube	disabled	3rd party (kinvolk.io)
inaccel	minikube	disabled	3rd party (InAccel [info@inaccel.com])
ingress	minikube	disabled	Kubernetes
ingress-dns	minikube	disabled	minikube
inspektor-gadget	minikube	disabled	3rd party (inspektor-gadget.io)
istio	minikube	disabled	3rd party (Istio)
istio-provisioner	minikube	disabled	3rd party (Istio)
kong	minikube	disabled	3rd party (Kong HQ)
kubeflow	minikube	disabled	3rd party
kubetail	minikube	disabled	3rd party (kubetail.com)
kubevirt	minikube	disabled	3rd party (KubeVirt)
logviewer	minikube	disabled	3rd party (unknown)
metallb	minikube	disabled	3rd party (MetalLB)
metrics-server	minikube	disabled	Kubernetes
nvidia-device-plugin	minikube	disabled	3rd party (NVIDIA)
nvidia-driver-installer	minikube	disabled	3rd party (NVIDIA)
nvidia-gpu-device-plugin	minikube	disabled	3rd party (NVIDIA)
olm	minikube	disabled	3rd party (Operator Framework)
pod-security-policy	minikube	disabled	3rd party (unknown)
portainer	minikube	disabled	3rd party (Portainer.io)
registry	minikube	disabled	minikube
registry-aliases	minikube	disabled	3rd party (unknown)
registry-creds	minikube	disabled	3rd party (UPMC Enterprises)
storage-provisioner	minikube	enabled ✅	minikube
storage-provisioner-gluster	minikube	disabled	3rd party (Gluster)
storage-provisioner-rancher	minikube	disabled	3rd party (Rancher)
volcano	minikube	disabled	third-party (volcano)
volumesnapshots	minikube	disabled	Kubernetes
yakd	minikube	disabled	3rd party (marcnuri.com)

Les seules addons qui sont actuellement installés (par défaut) sont default-storageclass et storage-provisioner.

Installez celle qui vous semble intéressante, pourquoi ?

```
ubuntu@ubuntu2204:~$ minikube addons enable dashboard
💡  dashboard is an addon maintained by Kubernetes. For any concerns contact minikube on GitHub.
You can view the list of minikube maintainers at: https://github.com/kubernetes/minikube/blob/master/OWNERS
    ▪ Using image docker.io/kubernetesui/dashboard:v2.7.0
    ▪ Using image docker.io/kubernetesui/metrics-scraper:v1.0.8
💡  Some dashboard features require the metrics-server addon. To enable all features please run:

        minikube addons enable metrics-server

🌟  The 'dashboard' addon is enabled
```

Nous avons choisi d'activer l'addon "dashboard" car cela facilite l'appréhension de kubernetes au début. En effet, cet addon fournit une interface graphique pour visualiser l'état du cluster, gérer les ressources, etc, ce qui peut nous aider à comprendre le fonctionnement de kubernetes.

Lister les profils actif sous minikube avec toutes leurs caractéristiques

```
ubuntu@ubuntu2204:~$ minikube profile list
```

PROFILE	DRIVER	RUNTIME	IP	VERSION	STATUS	NODES	ACTIVE PROFILE	ACTIVE KUBECONTEXT
minikube	docker	docker	192.168.49.2	v1.34.0	OK	1	*	*

On voit qu'il n'y a que le profil "minikube" par défaut. Le symbole "*" dans la colonne "ACTIVE PROFILE" indique qu'il est actif.

Quels sont les profils en cours ?

```
ubuntu@ubuntu2204:~$ minikube profile
minikube
```

On peut également utiliser cette commande pour voir le profil en cours, qui est bien "minikube".

Comment puis je créer un nouveau profile, que représente un profile ?

Il suffit d'utiliser l'option "-p" de "minikube start" qui nous permet de sélectionner un profile ou de le créer s'il n'existe pas.

```
ubuntu@ubuntu2204:~$ minikube start -p mon-nouveau-profil
😄  [mon-nouveau-profil] minikube v1.37.0 on Ubuntu 24.04 (vbox/amd64)
✨  Automatically selected the docker driver. Other choices: none, ssh
📌  Using Docker driver with root privileges
👍  Starting "mon-nouveau-profil" primary control-plane node in "mon-nouveau-profil" cluster
🚜  Pulling base image v0.0.48 ...
🔥  Creating docker container (CPUs=2, Memory=3072MB) ...
🐳  Preparing Kubernetes v1.34.0 on Docker 28.4.0 ...
🔗  Configuring bridge CNI (Container Networking Interface) ...
🔎  Verifying Kubernetes components...
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
🌟  Enabled addons: storage-provisioner, default-storageclass
🏄  Done! kubectl is now configured to use "mon-nouveau-profil" cluster and "default" namespace by default
```

Un profil représente une instance de cluster Kubernetes totalement indépendante avec sa propre configuration.

Afficher le statut de minikube

```
ubuntu@ubuntu2204:~$ minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured
docker-env: in-use
```

Cela nous indique que tout est bien démarré et configuré.

Comment puis-je accéder au dashboard de minikube ?

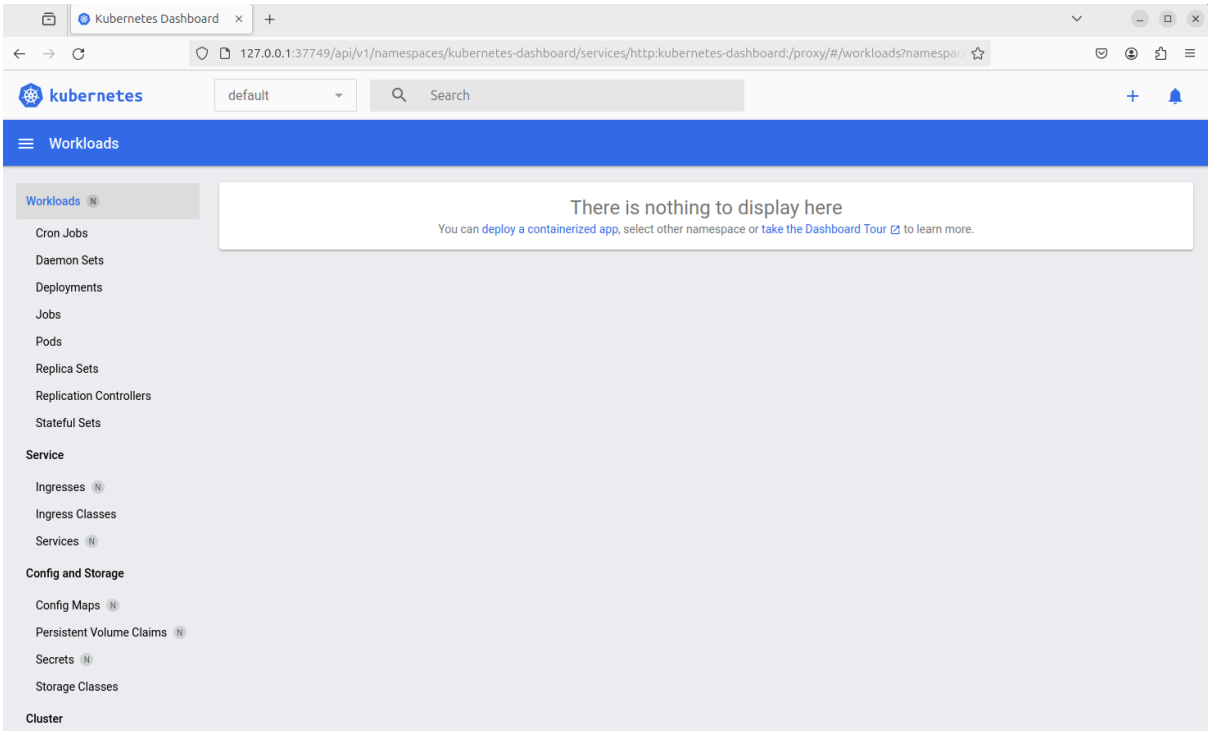
On peut simplement faire “minikube dashboard”.

```
ubuntu@ubuntu2204:~$ minikube dashboard
🤔  Verifying dashboard health ...
🚀  Launching proxy ...
🤔  Verifying proxy health ...
🎉  Opening http://127.0.0.1:37749/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/pr
oxy/ in your default browser...
update.go:85: cannot change mount namespace according to change mount (/run/user/1000/doc/by-app/snap.firefox /r
un/user/1000/doc none bind,rw,x-snapd.ignore-missing 0 0): cannot inspect "/run/user/1000/doc": lstat /run/user/
1000/doc: permission denied
Gtk-Message: 10:51:06.362: Not loading module "atk-bridge": The functionality is provided by GTK natively. Pleas
e try to not load it.
[36928, Main Thread] WARNING: GTK+ module /snap/firefox/4173/gnome-platform/usr/lib/gtk-2.0/modules/libcanberra-
gtk-module.so cannot be loaded.
GTK+ 2.x symbols detected. Using GTK+ 2.x and GTK+ 3 in the same process is not supported.: 'glib warning', file
 /build/firefox/parts/firefox/build/toolkit/xre/nsSigHandlers.cpp:187

(firefox:36928): Gtk-WARNING **: 10:51:06.504: GTK+ module /snap/firefox/4173/gnome-platform/usr/lib/gtk-2.0/mod
ules/libcanberra-gtk-module.so cannot be loaded.
GTK+ 2.x symbols detected. Using GTK+ 2.x and GTK+ 3 in the same process is not supported.
Gtk-Message: 10:51:06.504: Failed to load module "canberra-gtk-module"
[36928, Main Thread] WARNING: GTK+ module /snap/firefox/4173/gnome-platform/usr/lib/gtk-2.0/modules/libcanberra-
gtk-module.so cannot be loaded.
GTK+ 2.x symbols detected. Using GTK+ 2.x and GTK+ 3 in the same process is not supported.: 'glib warning', file
 /build/firefox/parts/firefox/build/toolkit/xre/nsSigHandlers.cpp:187

(firefox:36928): Gtk-WARNING **: 10:51:06.537: GTK+ module /snap/firefox/4173/gnome-platform/usr/lib/gtk-2.0/mod
ules/libcanberra-gtk-module.so cannot be loaded.
GTK+ 2.x symbols detected. Using GTK+ 2.x and GTK+ 3 in the same process is not supported.
Gtk-Message: 10:51:06.537: Failed to load module "canberra-gtk-module"
```

Les erreurs concernent uniquement le moment où minikube essaie de forcer l'ouverture du navigateur web. À cause des sécurités strictes d'Ubuntu (Snap confinement), Firefox râle parce qu'il n'aime pas être lancé de cette manière par un autre processus. Toutefois, le dashboard fonctionne bel et bien.



Qu’est ce que le DashBoard, que présente t’il ?

Le Dashboard est une interface utilisateur Web officielle pour Kubernetes. Il présente une vue graphique de l'état du cluster et permet de gérer ces ressources sans utiliser des lignes de commande.

Lister les nœuds d’un profil

On peut faire “kubectl get nods”.

```
ubuntu@ubuntu2204:~$ kubectl  get nodes
NAME       STATUS   ROLES           AGE   VERSION
minikube   Ready    control-plane   16d   v1.34.0
```

On peut aussi faire “minikube node list”.

```
ubuntu@ubuntu2204:~$ minikube node list
minikube        192.168.49.2
```

Ajouter un nœud à un profil minikube, supprimer ce même nœud

Il nous suffit de faire “minikube node add” puis “minikube node delete <nom du noeud>”.

```
ubuntu@ubuntu2204:~$ minikube node add
😄  Adding node m02 to cluster minikube as [worker]
❗  Cluster was created without any CNI, adding a node to it might cause broken networking.
👍  Starting "minikube-m02" worker node in "minikube" cluster
🚜  Pulling base image v0.0.48 ...
🔥  Creating docker container (CPUs=2, Memory=2200MB) ...
🐳  Preparing Kubernetes v1.34.0 on Docker 28.4.0 ...
🔎  Verifying Kubernetes components...
🏄  Successfully added m02 to minikube!
ubuntu@ubuntu2204:~$ minikube node delete minikube-m02
🔥  Deleting node minikube-m02 from cluster minikube
✋  Stopping node "minikube-m02"  ...
🛑  Powering off "minikube-m02" via SSH ...
🔥  Deleting "minikube-m02" in docker ...
💀  Node minikube-m02 was successfully deleted.
```

Consulter les logs de minikube, comment faire ?

Pour consulter les logs, il suffit de faire “minikube logs”.

```
ubuntu@ubuntu2204:~$ minikube logs

==> Audit <==
```

COMMAND	ARGS	PROFILE	USER	VERSION	START TIME	END TIME
start		minikube	ubuntu	v1.37.0	06 Jan 26 09:50 EST	06 Jan 26 09:54 EST
config	get driver	minikube	ubuntu	v1.37.0	06 Jan 26 09:56 EST	
config	get driver	minikube	ubuntu	v1.37.0	06 Jan 26 09:58 EST	
addons	list	minikube	ubuntu	v1.37.0	06 Jan 26 10:08 EST	06 Jan 26 10:08 EST
addons	enable dashboard	minikube	ubuntu	v1.37.0	06 Jan 26 10:12 EST	06 Jan 26 10:12 EST

Gestion des pods et services sous kubernetes

Lister les images actuellement en exécution dans votre environnement minikube

```
ubuntu@ubuntu2204:~$ minikube image ls
registry.k8s.io/pause:3.10.1
registry.k8s.io/kube-scheduler:v1.34.0
registry.k8s.io/kube-proxy:v1.34.0
registry.k8s.io/kube-controller-manager:v1.34.0
registry.k8s.io/kube-apiserver:v1.34.0
registry.k8s.io/etcd:3.6.4-0
registry.k8s.io/coredns/coredns:v1.12.1
gcr.io/k8s-minikube/storage-provisioner:v5
docker.io/kubernetesui/metrics-scraper:<none>
docker.io/kubernetesui/dashboard:<none>
```

Lancer une image nginx dans un pod et/ou un deployment en mode impératif

Lancement dans un pod :

```
ubuntu@ubuntu2204:~$ kubectl run nginx-pod --image=nginx --restart=Never
pod/nginx-pod created
```

Lancement dans un deployment :

```
ubuntu@ubuntu2204:~$ kubectl create deployment nginx-deploy --image=nginx
deployment.apps/nginx-deploy created
```

Créer un service en mode impératif permettant d'accéder à votre service nginx

```
ubuntu@ubuntu2204:~$ kubectl expose pod nginx-pod --type=NodePort --port=80 --name=nginx-service
service/nginx-service exposed
```

Visualiser les informations du pod et du service ?

```
ubuntu@ubuntu2204:~$ kubectl describe pod nginx-pod
Name:             nginx-pod
Namespace:        default
Priority:         0
Service Account:  default
Node:             minikube/192.168.49.2
Start Time:       Fri, 09 Jan 2026 09:24:34 -0500
Labels:           run=nginx-pod
Annotations:      <none>
Status:           Running
IP:               10.244.0.25
IPs:
  IP:  10.244.0.25
Containers:
  nginx-pod:
    Container ID:   docker://749a1e3277fdb23f942659e89f7bfd33f696eb1a2a8f0e98cee25b215fc1f87a
    Image:          nginx
    Image ID:       docker-pullable://nginx@sha256:ca871a86d45a3ec6864dc45f014b11fe626145569ef0e74dea
ffc95a3b15b430
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Fri, 09 Jan 2026 09:24:46 -0500
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-sblh8 (ro)
Conditions:
  Type                        Status
```

```
ubuntu@ubuntu2204:~$ kubectl describe service nginx-service
Name:                     nginx-service
Namespace:                default
Labels:                   run=nginx-pod
Annotations:              <none>
Selector:                 run=nginx-pod
Type:                     NodePort
IP Family Policy:         SingleStack
IP Families:              IPv4
IP:                       10.97.101.130
IPs:                      10.97.101.130
Port:                     <unset>  80/TCP
TargetPort:               80/TCP
NodePort:                 <unset>  31012/TCP
Endpoints:                10.244.0.5:80
Session Affinity:         None
External Traffic Policy:  Cluster
Internal Traffic Policy:  Cluster
Events:                   <none>
```

Obtenir l'url du service

```
ubuntu@ubuntu2204:~$ minikube service nginx-service --url
http://192.168.58.2:31012
```

Exécuter le service dans lynx ou Firefox ou tout autre browser

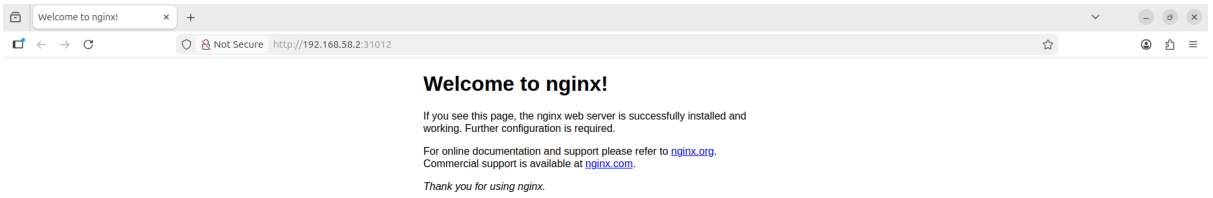
```
ubuntu@ubuntu2204:~$ minikube service nginx-service
```

NAMESPACE	NAME	TARGET PORT	URL
default	nginx-service	80	http://192.168.58.2:31012

```
🎉  Opening service default/nginx-service in default browser...
ubuntu@ubuntu2204:~$ Gtk-Message: 13:17:06.270: Not loading module "atk-bridge": The functionality is provided by GTK natively. Please
 try to not load it.
[213952, Main Thread] WARNING: GTK+ module /snap/firefox/7672/gnome-platform/usr/lib/gtk-2.0/modules/libcanberra-gtk-module.so cannot
be loaded.
GTK+ 2.x symbols detected. Using GTK+ 2.x and GTK+ 3 in the same process is not supported.: 'glib warning', file /build/firefox/parts/
firefox/build/toolkit/xre/nsSigHandlers.cpp:201

(firefox_firefox:213952): Gtk-WARNING **: 13:17:06.379: GTK+ module /snap/firefox/7672/gnome-platform/usr/lib/gtk-2.0/modules/libcanbe
rra-gtk-module.so cannot be loaded.
GTK+ 2.x symbols detected. Using GTK+ 2.x and GTK+ 3 in the same process is not supported.
Gtk-Message: 13:17:06.379: Failed to load module "canberra-gtk-module"
[213952, Main Thread] WARNING: GTK+ module /snap/firefox/7672/gnome-platform/usr/lib/gtk-2.0/modules/libcanberra-gtk-module.so cannot
be loaded.
GTK+ 2.x symbols detected. Using GTK+ 2.x and GTK+ 3 in the same process is not supported.: 'glib warning', file /build/firefox/parts/
firefox/build/toolkit/xre/nsSigHandlers.cpp:201

(firefox_firefox:213952): Gtk-WARNING **: 13:17:06.383: GTK+ module /snap/firefox/7672/gnome-platform/usr/lib/gtk-2.0/modules/libcanbe
rra-gtk-module.so cannot be loaded.
GTK+ 2.x symbols detected. Using GTK+ 2.x and GTK+ 3 in the same process is not supported.
Gtk-Message: 13:17:06.383: Failed to load module "canberra-gtk-module"
```



Comment puis-je lancer une commande bash directement dans mon conteneur nginx ?

Pour exécuter une commande directement dans un pod, on fait “kubectl exec <nom-du-pod> -- <commande>”.

```
ubuntu@ubuntu2204:~$ kubectl exec -it nginx-pod -- ls
bin   docker-entrypoint.d   home   media  proc  sbin  tmp
boot  docker-entrypoint.sh  lib    mnt    root  srv   usr
dev   etc                   lib64  opt    run   sys   var
```

On peut accéder au shell du pod en faisant “kubectl exec -it <nom du pod> -- sh”.

```
ubuntu@ubuntu2204:~$ kubectl exec -it nginx-pod -- sh
# ls
bin   dev                 docker-entrypoint.sh  home  lib64  mnt  proc  run   srv  tmp  var
boot  docker-entrypoint.d  etc                   lib   media  opt  root  sbin  sys  usr
#
```

Lister les logs du conteneur nginx du pod

```
ubuntu@ubuntu2204:~$ kubectl logs nginx-pod
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2026/01/09 14:24:46 [notice] 1#1: using the "epoll" event method
2026/01/09 14:24:46 [notice] 1#1: nginx/1.29.4
2026/01/09 14:24:46 [notice] 1#1: built by gcc 14.2.0 (Debian 14.2.0-19)
2026/01/09 14:24:46 [notice] 1#1: OS: Linux 6.14.0-37-generic
2026/01/09 14:24:46 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2026/01/09 14:24:46 [notice] 1#1: start worker processes
2026/01/09 14:24:46 [notice] 1#1: start worker process 30
2026/01/09 14:24:46 [notice] 1#1: start worker process 31
10.244.0.1 - - [09/Jan/2026:14:34:56 +0000] "GET / HTTP/1.1" 200 615 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; r
v:125.0) Gecko/20100101 Firefox/125.0" "-"
2026/01/09 14:34:57 [error] 30#30: *1 open() "/usr/share/nginx/html/favicon.ico" failed (2: No such file or directo
ry), client: 10.244.0.1, server: localhost, request: "GET /favicon.ico HTTP/1.1", host: "192.168.49.2:31170", refer
rer: "http://192.168.49.2:31170/"
10.244.0.1 - - [09/Jan/2026:14:34:57 +0000] "GET /favicon.ico HTTP/1.1" 404 153 "http://192.168.49.2:31170/" "Mozil
la/5.0 (X11; Ubuntu; Linux x86_64; rv:125.0) Gecko/20100101 Firefox/125.0" "-"
10.244.0.1 - - [09/Jan/2026:14:38:15 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:
125.0) Gecko/20100101 Firefox/125.0" "-"
```

Étape 2 : Projet d'une architecture de type EDA

Vous trouverez le dépôt GitHub du projet à l'adresse suivante :
https://github.com/SimonDum/eda-project.git

Ce projet répond à la demande de mise en place d'une architecture de type Event Driven (EDA). Dans un premier temps, l'objectif était de transformer des échanges de données synchrones (API REST classique) en un système asynchrone basé sur des événements échangés via un bus de messages. Dans un second temps, le but était de déployer l'ensemble de cette architecture via Kubernetes.

Nous avons développé une architecture composée de trois microservices applicatifs et de deux services d'infrastructure, respectant le schéma EDA proposé sur l'énoncé.

Les deux services de l'infrastructure sont :

- **kakfa** : Bus de messages Kafka composé des topics "etudiants-requests" et "etudiants-responses"
- **postgre** : Base de données PostgreSQL composée d'une simple table pour les étudiants

Les trois services applicatifs sont :

- **api-service** : Programme Java Spring Boot qui reçoit des requêtes HTTP (API) pour des opérations CRUD sur les étudiants et qui publie ces demandes sur le topic "etudiants-requests" plutôt que de s'occuper de les réaliser directement.
- **integration-service** : Programme Java Spring Boot qui se met en écoute du topic "etudiants-requests" et qui consomme les évènements de requête. Il effectue les opérations demandées avant de publier des évènements de réponse sur le topic "etudiants-responses" contenant les résultats.
- **front-service** : Serveur NodeJS et client HTML/Javascript qui donne une interface pour envoyer des requêtes HTTP à l'API, et qui se met en écoute du topic "etudiants-responses" pour consommer puis afficher les résultats.

Nous avons utilisé des Dockerfiles Multi-Stage afin de construire et de push des images optimisées de nos services applicatifs sur DockerHub. Un script bash nous a permis de faire le build et le push de toutes les images en un seul coup.

Pour le déploiement de ces images via Kubernetes (minikube), nous avons créé des fichiers yaml qui définissent les ressources Kubernetes nécessaires (Deployments, Services, etc.) et décrivent la configuration associée.

Parmi les remarques intéressantes à ce sujet, on peut évoquer l'utilisation des "readinessProbe". Cette option nous permet de définir des "sondes" pour s'assurer que Kubernetes sache quand le conteneur est réellement prêt (car souvent running != ready). Par exemple, le "readinessProbe" de l'API a été défini sur le endpoint de healthcheck. De cette manière, nous avons pu réaliser un script de déploiement qui attend que les services indispensables soient opérationnels avant de lancer les services qui en dépendent.

Une autre remarque intéressante concerne le choix des contrôleurs Kubernetes selon la nature du service. Pour nos microservices applicatifs (API, Front, Intégration), nous avons utilisé des Deployments. Comme ces services ne stockent rien en local (stateless), leurs conteneurs sont interchangeables : si l'un plante, Kubernetes le remplace par un nouveau avec un nom aléatoire, et cela ne pose aucun problème. En revanche, pour l'infrastructure (PostgreSQL et Kafka), nous avons choisi des StatefulSets. Ceux-ci garantissent une identité stable pour chaque pod (ex. kafka-0), ainsi qu'une adresse réseau persistante. Cette stabilité est particulièrement adaptée aux bases de données et aux systèmes distribués comme Kafka, car elle facilite la reconnexion automatique au volume de stockage associé et évite tout risque de conflit ou d'incohérence des données.

Concernant le développement des services d'API et d'intégration en Sprint Boot, plusieurs choix de conception méritent d'être soulignés. Les fonctions d'endpoints API et de requêtes en base de données étaient déjà en grande partie développées sur le template fourni, donc nous les avons séparé et adapté en 2 programmes distincts, en ajoutant les mécanismes Kafka (producteur/consommateur). De plus, nous avons choisi de définir des classes DTO (Data Transfer Objects) "StudentRequest" et "StudentResponse" pour avoir un format attendu (type d'action, données étudiant, etc) sur chacun des deux topics. La sérialisation et la désérialisation des messages (JSON) en objet se fait automatiquement grâce à Spring Kafka (basé sur Jackson). Les fonctionnalités CRUD manquantes ont également été ajoutées (modification et suppression).

Par ailleurs, il y a quelques points intéressants que nous pouvons mentionner concernant notre implémentation du service de front. L'application est servie par un backend Node.js agissant comme une passerelle : il traduit les actions de l'utilisateur en appels API REST et en parallèle il écoute le bus d'événements. Grâce à l'utilisation de WebSockets, le serveur pousse les notifications en temps réel vers le navigateur dès qu'une réponse est consommée depuis Kafka. Ainsi, si un client demande une opération, celle-ci est reflétée automatiquement dans son interface, mais aussi dans celle de tous les autres clients qui sont connectés sans aucune action nécessaire de leur part.

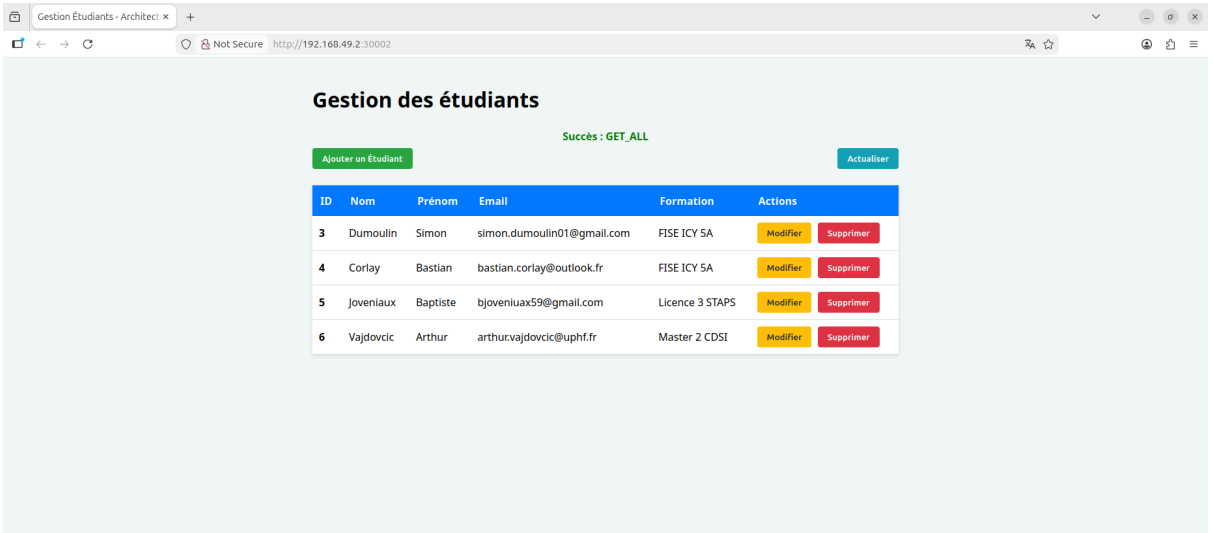
Pour montrer le bon fonctionnement de notre architecture, on exécute simplement le script “deploy.sh” (ça peut être légèrement long car on attend que les services soient prêts) :

```
ubuntu@ubuntu2204:~/Downloads/eda_project$ ./deploy.sh
Démarrage du déploiement de l'architecture Microservices...
[1/3] Déploiement de l'infrastructure (Postgres & Kafka)...
service/postgres-service created
statefulset.apps/postgres created
configmap/postgres-init unchanged
service/kafka-service created
statefulset.apps/kafka created
Attente de la disponibilité de la Base de données et du Broker...
pod/postgres-0 condition met
pod/kafka-0 condition met
[2/3] Déploiement des services applicatifs...
deployment.apps/api-service created
service/api-service created
deployment.apps/integration-service created
deployment.apps/front-service created
service/front-service created
[3/3] Vérification que tous les services sont prêts...
En attente du service API...
deployment.apps/api-service condition met
En attente du service d'intégration...
pod/integration-service-7f8db66f98-5nwzl condition met
En attente du service de front...
deployment.apps/front-service condition met
Déploiement terminé avec succès.
Lancement de l'application...
```

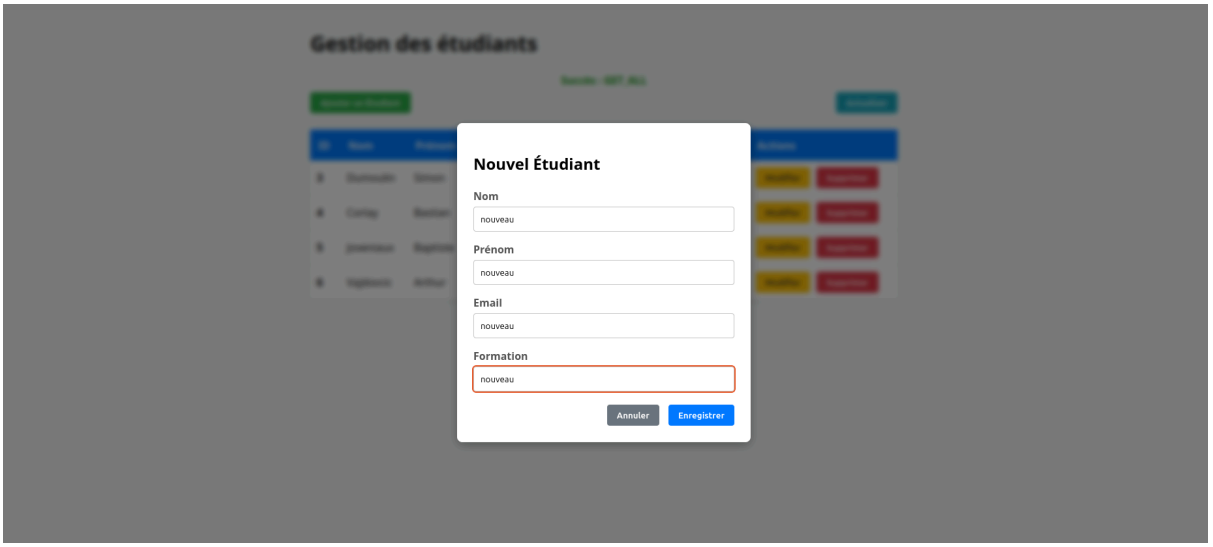
NAMESPACE	NAME	TARGET PORT	URL
default	front-service	3000	http://192.168.49.2:30002

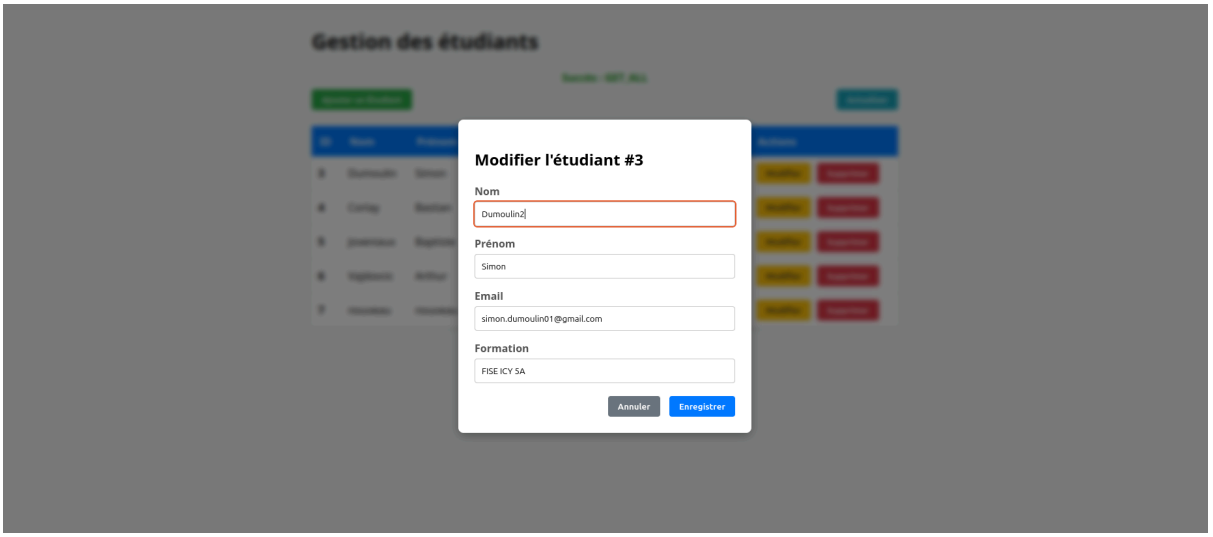
```
🎉  Opening service default/front-service in default browser...
```

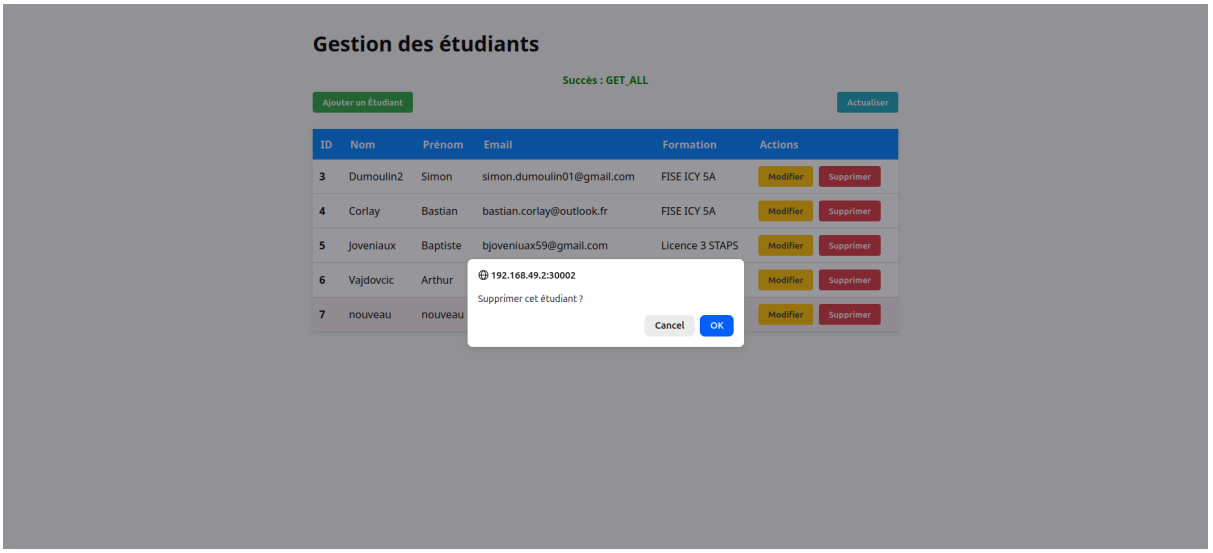
Le navigateur s’ouvre automatiquement une fois que tout est lancé et prêt :



On peut désormais visualiser, ajouter, modifier et supprimer des étudiants :







On peut même ouvrir la page sur un deuxième onglet et observer les modifications effectuées par le premier client en temps réel (rechargement automatique des étudiants).

Enfin, un script permet de supprimer les ressources Kubernetes qui composent l'architecture (y compris les données) :

```
ubuntu@ubuntu2204:~/Downloads/eda_project$ ./cleanup.sh
[1/3] Suppression des services applicatifs...
deployment.apps "front-service" deleted from default namespace
service "front-service" deleted from default namespace
deployment.apps "api-service" deleted from default namespace
service "api-service" deleted from default namespace
deployment.apps "integration-service" deleted from default namespace
[2/3] Suppression de l'infrastructure...
service "kafka-service" deleted from default namespace
statefulset.apps "kafka" deleted from default namespace
service "postgres-service" deleted from default namespace
statefulset.apps "postgres" deleted from default namespace
configmap "postgres-init" deleted from default namespace
[3/3] Nettoyage des volumes persistants (PVC)...
persistentvolumeclaim "pgdata-postgres-0" deleted from default namespace
persistentvolumeclaim "kafka-data-kafka-0" deleted from default namespace
Environnement nettoyé.
```